COMPRESSION, GENERATION, AND INFERENCE VIA SUPERVISED LEARNING

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Jiaming Song
December 2021

Abstract

Artificial intelligence and machine learning methods have seen tremendous advances in the past decade, thanks to deep neural networks. Supervised learning methods enables neural networks to effectively approximate low-level functions of human intelligence, such as identifying an object within an image. However, many complex functions of human intelligence are difficult to solve with supervised learning directly: humans can build concise representations of the world (*compression*), generate works of art based on creative imaginations (*generation*), and infer how others will act from personal experiences (*inference*).

In this dissertation, we focus on machine learning approaches that reduce these complex functions of human intelligence into simpler ones that can be readily solved with supervised learning and thus enabling us to leverage the developments in deep learning. This dissertation comprises of three parts, namely *compression*, *generation*, and *inference*.

The first part discusses how we can apply supervised learning to unsupervised representation learning. We develop algorithms that can learn informative representations from large unlabeled datasets while protecting certain sensitive attributes. The second part extends these ideas to learning high-dimensional probabilistic models of unlabeled data. Combined with the insights from the first part, we introduce a generative model suitable for conditional generation under limited supervision. In the third and final part, we present two applications of supervised learning in probabilistic inference methods: (a) optimizing for efficient Bayesian inference algorithms and (b) inferring the agents' intent under complex, multi-agent environments. These contributions enable machines to overcome existing limitations of supervised learning in real-world compression, generation, and inference problems.

Acknowledgments

First and foremost, I would like to thank my Ph.D. advisor, Stefano Ermon. I had a fantastic Ph.D. experience with Stefano in the past four and a half years, which would not have been possible without Stefano's patient advice over my research and the liberty he gave me to pursue my personal research interests. Stefano has an amazing depth and breadth of knowledge about various fields in machine learning and can answer my questions whenever I need to. I will never forget his advice on type checks and rigorous theory statements during many hours of paper writing, and I often feel that he treats the rigor in my papers and talks more seriously than I do; I am both sorry and thankful for these moments. Outside of core machine learning, Stefano also has keen insights about valuable scientific domains to work on, such as computational sustainability, instead of simply pursuing what are "hot". I hope that I will learn these essential qualities of an advisor in my future academic career and that our collaborations will remain fruitful.

My journey at Stanford would not be complete without the company I enjoyed with peers, especially at the Stanford AI lab. Aditya Grover's research has shaped my own a lot, and he was of tremendous help during every important step of my Ph.D. Shengjia Zhao, my roommate for four years, continues to be a huge inspiration for life and research. Jonathan Kuck and Rui Shu were amazing lab mates with a lot of fun. I learned a lot from Yang Song, and we had a blast sailing, golfing, and playing computer games. Kuno was a great gym buddy (and instructor) before covid, and my dissertation talk would not be the same without him. Lantao Yu is a very reliable collaborator, and we had amazing times skiing and playing board games. Kristy Choi was an incredible host at her home parties and my Ph.D. "farewell" dinner. Tri Dao was extremely helpful during my quarter as course assistant and an inspirational gym buddy. Chenlin Meng has unparalleled work ethic and many projects were made possible by her. I learned a lot from more senior lab mates: Volodymyr Kuleshov, Russell Stewart, Jonathan Ho, and Burak Ukzent, and hope to learn even more from newer ones: Chris Cundy, Andy Shih, Willie Neiswanger, Charlie Marx, and Gengchen Mai. I also miss hanging out with the many friends at StatsML group and InfoLab,

and others from places such as UC Berkeley, MIT, Tsinghua and Toronto.

I have also had the privilege of the guidance of various amazing mentors over the years. Jun Zhu gave me my first experiences in machine learning research when I was an undergraduate at Tsinghua University, and I extended this experience with Wenwu Zhu and Peng Cui. Lawrence Carin and many of his students, including Zhe Gan, Chunyuan Li, Changyou Chen, Yizhe Zhang, Yunchen Pu, were accomodating during my stay at Duke University and are one of the reasons why I pursued a Ph.D. after undergrad. Chelsea Finn, Dorsa Sadigh, Emma Brunskill, Mykel Kochenderfer, Peter Bailis, Jiajun Wu, and Tengyu Ma were generous with their insightful feedback and time during rotations, quals, oral exams and the dissertation. John Schulman, Yan Duan, Tengyu Ma, Yann Dauphin and Michael Auli broadened my research perspectives in summer internships at OpenAI and Facebook. I have also enjoyed discussions with collaborators who helped me with interesting ideas in research: Abhishek Sinha, Ali Malik, Divyansh Garg, Hongyu Ren, Laëtitia Shao, Nate Gruver, Samarth Sinha, Yilun Xu, Yuhuai Wu, Yunzhu Li, and Yusuke Tashiro.

This dissertation would not have been possible with my many other co-authors: Aidan Perreault, Akshat Jindal, Alekh Agarwal, Ali Malik, Animesh Garg, Arianna Yuan, Ashish Kapoor, Ashish Sabharwal, Ayush Kumar, Danny Nemer, Eric J. Horvitz, Hao Tang, Harlan Seymour, Hongxia Jin, Huazhong Yang, Huimin Ma, Jiayu Chen, Jun-Yan Zhu, Kenneth Tran, Noah Goodman, Rachel Luo, Silvio Savarese, Shuvam Chakraborty, Yanan Sui, Yihong Gu, Yu Wang, Yuanfan Xu, Yuanxin Zhang, and Yutong He. I thank them for their contributions.

During my job search, I was extremely fortunate to have received encouraging guidance from researchers in various institutions. In addition to those mentioned above, I especially thank David McAllester for supporting my application, Jeff Ullman and Keith Weinstein for your honest and helpful opinion on my job talk, Yang Gao, Yi Wu, and Huazhe Xu for sound advice for the various statements, and Danfei Xu and Wei Hu for sharing ups and downs during the interview process. I also appreciate the support of all funding agencies, including the Qualcomm Innovation Fellowship and research grants from NSF, the Toyota Research Institute, the Future of Life Institute, Siemens, Sony, and Lam Research.

I would also like to thank many other friends who were a crucial part of the journey. I have had many fond memories having fun, meeting and travelling around the world with you, and I will cherish these moments forever.

Finally, I would like to thank my family for their love and support, especially my parents and my fiancé Chenchen. I am forever indebted to you.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Deep function approximators, such as neural networks, are highly successful in supervised learning; given abundant labels that are directly provided by humans, neural networks can learn to map training data to the corresponding labels [HZRS16, SHM+16]. However, this learning paradigm does not suit complex algorithms that describe more sophisticated intelligent behaviors; these include algorithms for effective compression of data, algorithms that synthesize novel data, and algorithms that infer the reward models of human drivers. These algorithms are difficult to solve with direct supervised learning, since the acquisition of "ground-truth" labels (*e.g.*, compressed representation of a data point, realism of a synthesized sample, reward of a driving trajectory) is either too expensive to acquire in mass quantities or challenging to define even for domain experts [Zho18, Bar89, Gha03, HTF09].

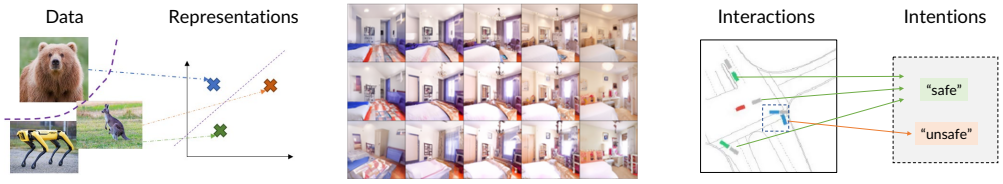
The goal of this dissertation is to build machines that learn these complex algorithms with minimal direct supervision from humans. In particular, we discuss three key problems in unsupervised machine learning that are instances of the tasks[1]:

Compression (Figure 1.1a) : How can we extract compact and informative representations of the data that are useful for downstream goals (*e.g.*, accuracy, privacy, fairness)? These representations can vastly reduce the need for labeled data in a new task.

Generation (Figure 1.1b) : How can we learn powerful generative models that capture complex, multi-modal distributions from data samples (*e.g.* images, videos, demonstrations)? These models can be used to manipulate existing data, evaluate test data, or synthesize new samples.

[1] We provide a detailed account of these tasks in Chapter 2.

Inference (Figure 1.1c) : How can we efficiently infer crucial latent variables (or factors) from existing observations (*e.g.* sample efficiency, reward functions)? The inferred latent variables can be used to optimize certain algorithms (*e.g.*, a Markovian policy) to a desired state (*e.g.*, an instance of the Markov chain Monte Carlo sampler, or a policy for autonomous driving).



(a) Compression. (b) Generation. (c) Inference.

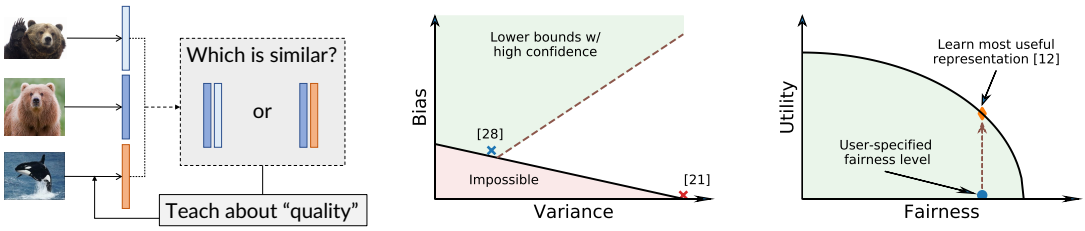
Figure 1.1: Overview of the dissertation topic.

In this dissertation, we apply supervised learning to compression, generation, and inference. We adopt ideas from information theory [BA03], probabilistic inference [Lev18], and physics [N+11] that reduces these difficult problems into simpler ones that can be solved with supervised learning with minimal human supervision. Methods that are discussed in this dissertation can compress neural networks by 60% yet achieving better accuracy (Chapter 4, [SE20b]), perform realistic image manipulation 100× faster than state-of-the-art methods such as StyleGAN2 (Chapter 8, [SSME21]), reduce wall-clock time of probabilistic inference algorithms by more than 90% (Chapter 9, [SZE17]), and reduce the number of crashes of autonomous driving agents by 50% to 70% (Chapter 10, [SRSE18]), illustrating their effectiveness in concrete applications.

1.1 Overview

1.1.1 Compression via Supervised Learning

Representation learning is a fundamental problem in machine learning, which processes raw data into structured representations that are easier to understand. While representations can be learned via direct supervised learning (*e.g.*, from inputs to labels), labeled data can be much more expensive and difficult to acquire than unlabeled ones [HFW+19]. As a result, methods based on unsupervised learning can be more promising, since it allows the use of large amounts of unlabeled data to boost performance on downstream tasks and alleviate the use of labels [MSC+13, DCLT18, MK13, RNSS18, BMR+20]. Ideally, these representations should retain information about real-world data as much as possible, useful for downstream applications (*e.g.*, linear classification) and sufficiently



(a) InfoMax via classification. (b) MI estimation. (c) Fair representation learning.

Figure 1.2: **Compression via Supervised Learning.** (a) InfoMax is a simple principle for representation learning but hard to apply in practice since mutual information is hard to estimate from samples. (b) We found that unbiased estimators have very high variance and proposed a practical high-confidence lower bound estimator with low bias; (c) We find the most informative representations under fairness constraints with alternative notions of information.

concise (*i.e.*, compressed). Given our goal of learning concise and useful representations of the data, we will use the term "compression" for representation learning.

Mutual information estimation for representation learning Information Maximization (InfoMax, [BS95]) is a fundamental principle for learning representations from vast amounts of unlabeled data, where the *(Shannon) mutual information* (MI), a measurement of mutual dependence, is maximized between the observations and learned representations (Figure 1.2a). As mutual information is difficult to measure directly from samples, the InfoMax principle can be instantiated by "supervised learning": given the observations, we can *learn* a classifier to measure mutual information, which can then be used to learn better representations. However, accurately estimating mutual information remains a challenging problem, and it is unclear whether the estimated mutual information learning in unsupervised representation learning are accurate enough to reflect the actual amount of "information" that is learned.

Many existing methods maximize unbiased estimators of mutual information lower bounds. In Chapter 3, I showed that these may not apply well in practical scenarios because of exponentially high variances [SE20c]. I proposed a multi-label classification approach [SE20b] that balances the bias-variance trade-off and outperforms existing state-of-the-art methods in terms of compressing neural networks and learning compact representations of high-dimensional data.

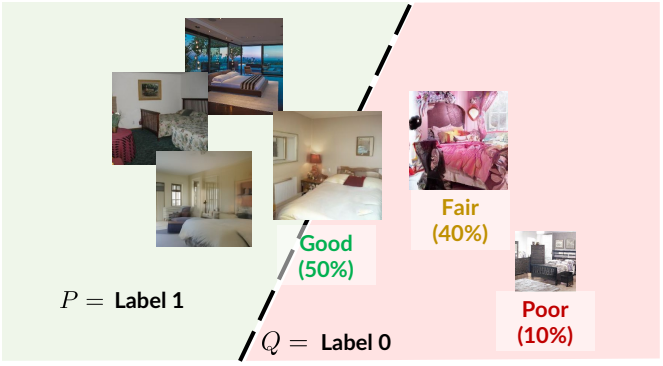
Fair representation learning Large datasets are often collected with certain biases against minority groups [Naj20], and our models may inherit such biases without careful design. This motivates our project in fair representation learning, which we detail in Chapter 5. As an example,

suppose that we wish to distribute a dataset (*e.g.*, hospital diagnostics) to a downstream vendor (hospital or research institute), but we do not wish to leak certain sensitive information about the patients (*e.g.*, gender or ethnicity). This requires us to process the dataset (*i.e.*, obtain a representation) that is sufficiently informative but also protects against the sensitive attributes that we care about. In fair representation learning, we often wish to learn informative representations while limiting the "unfairness" of the representations. To this end, we introduce the first framework of controllable fair representation learning [SKG+18] (Figure 1.2c), where owners of a dataset with certain sensitive attributes (*e.g.*, gender) can prevent compute-constrained downstream users from exploiting these attributes and making unfair decisions.

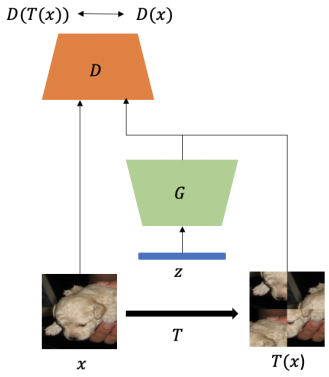
1.1.2 Generation via Supervised Learning

Generative models trained on large amounts of unlabeled data have achieved great success in various domains including images [BDS18, KLA+20, ROV19, HJA20], text [LGL+20, AABS19], audio [DJP+20, PPZS20, vdODZ+16, MEHS21], and graphs [GZE19, NSS+20]. These models are designed to model high-dimensional probability distributions that can be highly complex and rich in modalities. Unlike direct supervised learning which models a conditional distribution – typically from a complex input to a less complex output – a generative model may learn from unlabeled data, and thus does not necessarily have complex input modalities. This makes it challenging to train complex, multi-modal generative models despite the amazing success in deep learning for supervised learning. In this dissertation, I discuss how we can leverage advances in supervised learning to improve generative models.

Reweighted examples for generative models In supervised learning, one common objective function is to perform reweighting over samples. For example, one can assign higher weights to data points that are harder, which encourages the model to learn these harder examples. In contrast, if the dataset is assumed to contain labels that do not reflect the ground truth (*i.e.*, labels gathered from crowdsourcing or web-supervision), then we might be inclined to assign lower weights to examples that might be incorrect in the first place. In [SE20a], I leveraged a similar idea in learning generative adversarial networks, where we assign higher weights to examples that appear more realistic (Figure 1.3a); this simple approach has been shown to consistently improve the performance of state-of-the-art generative adversarial networks with negligible computational overhead.



(a) Improved supervised learning with reweighted labels.



(b) Improved supervised learning with "negative" data.

Figure 1.3: **Supervised learning for generative models.** Generative models can benefit from advances in how the "supervised learning" approach is involved.

Negative examples for generative models A fundamental problem surrounding generative models is how to define "generalization" [ZRY+18]. Generalization is necessary in generative models because without such a notion we can simply memorize the entire training dataset and achieve the minimum loss; however it is difficult to specify the boundaries to which we wish to generalize. If we have a image dataset with every image containing two objects, then it would be natural to **not** generate images with one or three objects; however, if the dataset contains images with two, three, and four objects, then suddenly it could be difficult to determine whether we should generate images with one or five objects.

The extent to which the model should generalize beyond the dataset is typically encoded as an inductive bias that is implicit within the model (and mostly beyond user control). However, there are cases where we wish to tell the model when it should not generalize. In [SKS+21], we discuss how we can incorporate such prior knowledge into the generative model, by simply performing data augmentations that produces "negative examples". This "negative data augmentation" technique directly impacts the supervised learning procedure and improves the resulting generative model (Figure 1.2a).

Few-show conditions for generative models Downstream applications of generative models are often based on various conditioning signals, such as labels [MO14], text descriptions [MPBS15] and reward values [YLY+18]. If we attempt to directly train conditional models, we would have to acquire large amounts of paired data [LMB+14, RPG+21] that are costly. Therefore, it is often

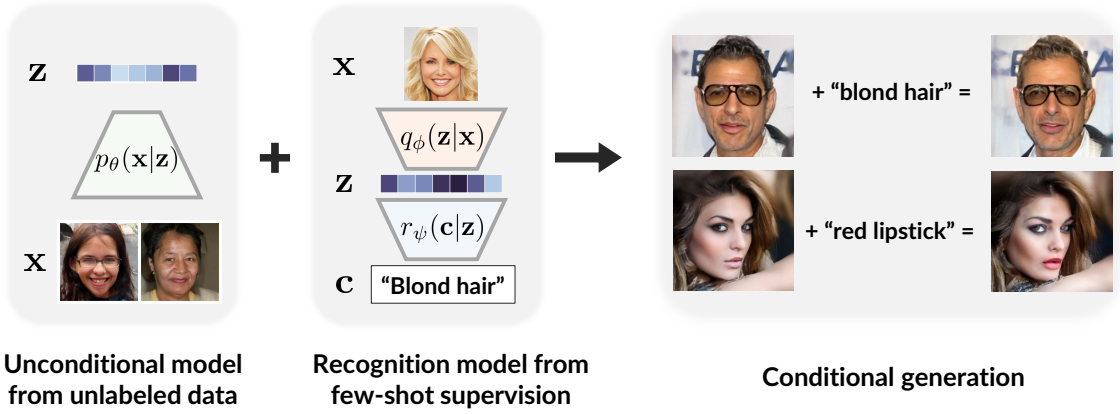


Figure 1.4: Overview of few-shot conditional generation.

desirable to learn good generative models from large amounts of unlabeled data, and then embed new conditions via supervised learning; if we are able to learn high-quality representations, then the supervised learning procedure here would not have to involve lots of data. In [SSME21], we discuss how advances in representation learning can be leveraged to perform few-shot conditional generative modeling, where a few labeled examples are all that is needed to introduce conditions to an unconditional generative model (Figure 1.4).

1.1.3 Inference via Supervised Learning

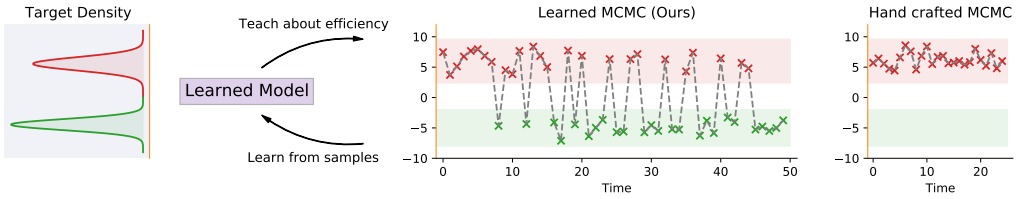


Figure 1.5: **Supervised learning for Bayesian inference.** We can use supervised learning to measure efficiency of MCMC proposals.

Bayesian Inference Markov Chain Monte Carlo (MCMC) methods have played important roles in statistics and machine learning as a fundamental method for probabilistic inference and sampling. Despite their immense success, they have seen little use with deep neural networks. This is because evaluating and selecting a good proposal distribution – a key element in MCMC – is often *more*

art than science [HLZ16], *i.e.*, it requires design choices from domain experts that are difficult to describe via automated procedures.

I addressed this challenge via supervised learning, where a model learns to evaluate the efficiency of a proposal distribution from supervised learning. In turn, this model can be used to optimize for more efficient MCMC proposals. This has lead to the first end-to-end method that learns an efficient MCMC proposal with neural networks [SZE17]. The key insight is that higher-quality MCMC proposals produce samples that are less correlated, which can be quantified through a classifier that distinguishes correlated samples from decorrelated ones. The classifier then provides a differentiable objective that can be used to guide more efficient proposals. My learned proposals outperformed the best hand-crafted ones by $3\times$ to $500\times$ in terms of statistical efficiency, opening the avenue for MCMC methods to benefit from deep learning.

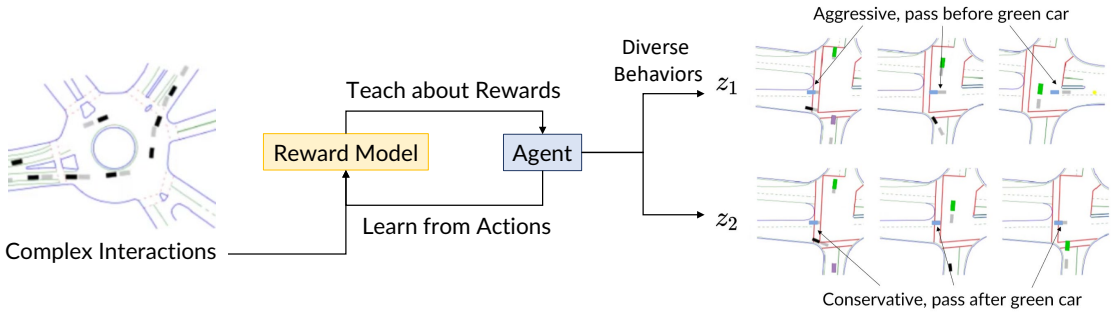


Figure 1.6: **Supervised learning for reward inference.** These rewards can then be used to learn diverse behaviors, such as in autonomous driving.

Inference in Inverse Reinforcement Learning A key aspect of intelligent agents is the ability to perform sequential decision making. While reinforcement learning is successful for this goal when reward functions are well-defined, their applications are limited in real-world multi-agent scenarios where it is difficult to engineer the correct reward functions, such as autonomous driving; even the slightest reward mis-specification can have catastrophic effects [HMMA+17, AC16].

I address this issue via supervised learning: given multi-agent interactions, we learn multiple classifiers to distinguish demonstrated interactions from learned ones; these classifiers can then serve as reward functions to guide multi-agent policies. These learned classifiers can guide agents towards desired complex behaviors, which do not have to be cooperative or zero-sum as often assumed by prior work. Based on this, I proposed the first framework that infers reward functions from general multi-agent interactions, which only assumes that the demonstrators form a Nash Equilibrium [SRSE18].

For more realistic scenarios with many agents such as traffic, I further extended this framework to a novel equilibrium concept that assumes bounded rationality of the demonstrators [YSE19]. I made it scalable to many more agents, learning from real-world traffic data and inferring the latent behavior of the agents [LSE17a, GSKE20]. On real-world traffic datasets, our learned policies provide a 50% - 70% reduction in the number of crashes compared to other learned policies.

1.2 Dissertation Structure

This dissertation consists of three parts, where we discuss how supervised learning can be integrated and improved in three distinctive yet connected problems in unsupervised machine learning: compression, generation and inference.

Part I is about compression – learning rich and compact representations of data without labels. In particular, we consider the information maximization paradigm and explore the fundamental limitations and practical challenges of applying this notion to unsupervised representation learning.

In Chapter 3, we start by a summary of how mutual information are estimated and optimized in unsupervised representation learning and discuss their limitations. We then introduce an estimator that has significantly lower variance. This chapter was previously published as [SE20c].

In Chapter 4, we further discuss the bias and variance trade-offs of mutual information lower bound estimators, which can also be optimized for representation learning. We introduce an estimator based on multi-label classification that achieves the lowest bias among all valid lower bound estimators. This chapter was previous published as [SE20b].

In Chapter 5, we consider a regression approach to mutual information estimation and apply it to learning fair representations. This chapter is primarily based on [SKG+19] and also includes materials from [XZS+20].

Part II is about generation – learning generative models from unlabeled data that can be used to synthesize additional novel data. We can improve generative models by improving the corresponding supervised learning components. We also leverage methods and observations introduced in Part I.

In Chapter 6, we discuss a reweighted objective function in supervised learning, how it is related to traditional estimators of probability divergences, and how it can be used to improve generative modeling. This chapter was previously published as [SE20a].

In Chapter 7, we introduce negative data augmentations in supervised learning, which reduces bias and improves generalization of generative models. This chapter was previous published as [SKS+21].

In Chapter 8, we consider a generative model that is suitable for few-shot conditional generation, *i.e.*, new conditions can be learned from a few examples. Thanks to supervised learning methods for representation learning, our approach can learn new conditions from as few as 100 examples, which can then be applied to image generation and manipulation. This chapter is primarily based on [SSME21] and also includes materials from [SME20].

Part III is about inference – how machines can efficiently infer latent factors from observations when these factors are hard to provide by humans. We can use supervised learning to infer these latent factors or improve the inference process itself.

In Chapter 9, we introduce the first method that learns to optimize Markov chain Monte Carlo methods, which is a hallmark in Bayesian inference (and one of the top-10 algorithms in the 20th century). This chapter was previously published as [SZE17].

In Chapter 10, we discuss how we can infer latent variables that reflect the intentions of agents operating in a multi-agent scenario, which can be applied to transportation systems. This chapter is primarily based on [SRSE18] and [YSE19] and also includes materials from [GSKE20].

We conclude and discuss future directions in Chapter 11.2.

Chapter 2

Background

We begin by providing some background on *learning distributions*, *i.e.*, comparing and optimizing high-dimensional distributions, which includes images, videos, audio, graphs, and demonstrations sequences among other data modalities. We first formally setup the task of comparing and optimizing distributions. Next, we discuss several machine learning problems that belong to learning distributions, such as compression and generation. Finally, we categorize and compare two major approaches of learning distributions that leverages supervised learning techniques.

2.1 Comparing and Optimizing Distributions

Notations We use uppercase letters to denote a probability measure (e.g., P, Q) and corresponding lowercase letters to denote its density (likelihood) functions (e.g., p, q) unless specified otherwise (in certain cases we may use notations such as $\mathrm{d}P$ and $\mathrm{d}Q$). We use X, Y to denote random variables with sample spaces denoted as $\mathcal{X}$ and $\mathcal{Y}$ respectively, and $\mathcal{P}(\mathcal{X})$ (or $\mathcal{P}(\mathcal{Y})$) to denote the set of all probability measures over the Borel σ-algebra on $\mathcal{X}$ (or $\mathcal{Y}$).

Under $Q \in \mathcal{P}(\mathcal{X})$, the p-norm of a function $r : \mathcal{X} \to \mathbb{R}$ is defined as

$$\|r\|_p := \left(\int |r|^p \mathrm{d}Q\right)^{1/p}$$

with $\|r\|_\infty = \lim_{p\to\infty} \|r\|_p$. The set of locally p-integrable functions is defined as

$$L^p(Q) := \{r : \mathcal{X} \to \mathbb{R} \mid \|r\|_p < \infty\}.$$

The space of probability measures wrt. Q is defined as

$$\Delta(Q) := \{r \in L^1(Q) \mid \|r\|_1 = 1, r \geq 0\};$$

we also call this the space of "valid probability ratios" / "valid density ratios" wrt. Q. For example, $\mathrm{d}P/\,\mathrm{d}Q \in \Delta(Q)$ because $\|\mathrm{d}P/\,\mathrm{d}Q\|_1 = \int(\mathrm{d}P/\,\mathrm{d}Q)\,\mathrm{d}Q = 1$. We use $P \ll Q$ to denote that P is absolutely continuous with respect to Q.

Let us assume that we have two distributions P and Q, under the same domain. In many machine learning problems, we access to these distributions via *i.i.d.* samples but do not have further knowledge about them. For example, we have access to a dataset of high-resolution images and assume these are collected *i.i.d.* from an underlying data distribution P, yet we cannot query the probability mass/density function of the data distribution P. A general scenario for machine learning tasks is to estimate and/or optimize some notion of discrepancy (*e.g.*, a probabilistic divergence) between the two distributions P and Q (for which we often access only via samples):

$$\mathrm{D}(P, Q) \quad \text{or alternatively,} \quad \mathrm{D}(p, q) \tag{2.1}$$

which are both based on common notation practices in machine learning. One common divergence is the KL divergence:

$$\mathrm{D}_{\mathrm{KL}}(P, Q) = \mathbb{E}_{\boldsymbol{x}\sim P}[\log \mathrm{d}P(\boldsymbol{x}) - \log \mathrm{d}Q(\boldsymbol{x})] \tag{2.2}$$

$$:= \mathbb{E}_{\boldsymbol{x}\sim P}[\log p(\boldsymbol{x}) - \log q(\boldsymbol{x})] \tag{2.3}$$

minimizing which is highly related to maximum likelihood. KL divergence also belongs to a large family of divergence called f-divergences. For any convex and semi-continuous function $f : [0, \infty) \to \mathbb{R}$ satisfying $f(1) = 0$, the f-divergence [Csi64, AS66] between two probabilistic measures $P, Q \in \mathcal{P}(\mathcal{X})$ is defined as:

$$D_f(P, Q) := \mathbb{E}_Q\left[f\left(\frac{\mathrm{d}P}{\mathrm{d}Q}\right)\right] \tag{2.4}$$

$$= \int_{\mathcal{X}} f\left(\frac{\mathrm{d}P}{\mathrm{d}Q}(\boldsymbol{x})\right)\mathrm{d}Q(\boldsymbol{x}), \tag{2.5}$$

if $P \ll Q$ and $+\infty$ otherwise. The f for KL-divergence is simply $f(u) = u \log u$. We defer the discussion of other divergences, such as the integral probability metric [Mül97], to the relevant

chapter.

Next, we discuss specific problems of compression, generation, and inference that belong to the general problem of estimating/optimizing distributions.

2.1.1 Compression

We use the term *compression* as an alias to learning compact representations from unlabeled data. In representation learning, we are interested in learning a (possibly stochastic) network $h : \mathcal{X} \to \mathcal{Y}$ that maps some data $\boldsymbol{x} \in \mathcal{X}$ to a compact representation $h(\boldsymbol{x}) \in \mathcal{Y}$. For ease of notation, we denote $p(\boldsymbol{x})$ as the data distribution, $p(\boldsymbol{x}, \boldsymbol{y})$ as the joint distribution for data and representations (denoted as $\boldsymbol{y}$), $p(\boldsymbol{y})$ as the marginal distribution of the representations, and X, Y as the random variables associated with data and representations. The InfoMax principle [Lin88, BS95, DHFLM+18] for learning representations considers variational maximization of the mutual information $I(X; Y)$:

$$I(X; Y) := \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim p(\boldsymbol{x},\boldsymbol{y})} \left[\log \frac{p(\boldsymbol{x}, \boldsymbol{y})}{p(\boldsymbol{x})p(\boldsymbol{y})} \right] \tag{2.6}$$

Mutual information is also the KL-divergence between two distributions $P = p(\boldsymbol{x}, \boldsymbol{y})$ and $q = p(\boldsymbol{x})p(\boldsymbol{y})$, which is maximized for the purpose of compression. Thus the objective function becomes:

$$\text{maximize} \quad D_{\mathrm{KL}}(p(\boldsymbol{x}, \boldsymbol{y}), p(\boldsymbol{x})p(\boldsymbol{y})), \tag{2.7}$$

which is our main focus in Part I.

2.1.2 Generation

In generative modeling, our goal is to learn a data distribution $P := p_{\mathrm{data}}$ given access to a training set (*e.g.*, a dataset of images). To this end, we parametrize a family of models $\mathcal{M}$ whose elements $Q := q_\theta$ are specified by a set of real-valued parameters θ. Our goal is to find the distribution $q_\theta \in \mathcal{M}$ such that some notion of discrepancy between p_{data} and q_θ is minimized:

$$\text{minimize} \quad D(P, Q), \tag{2.8}$$

and the learned model distribution would be close enough to the desired data distribution. This is our main focus in Part II.

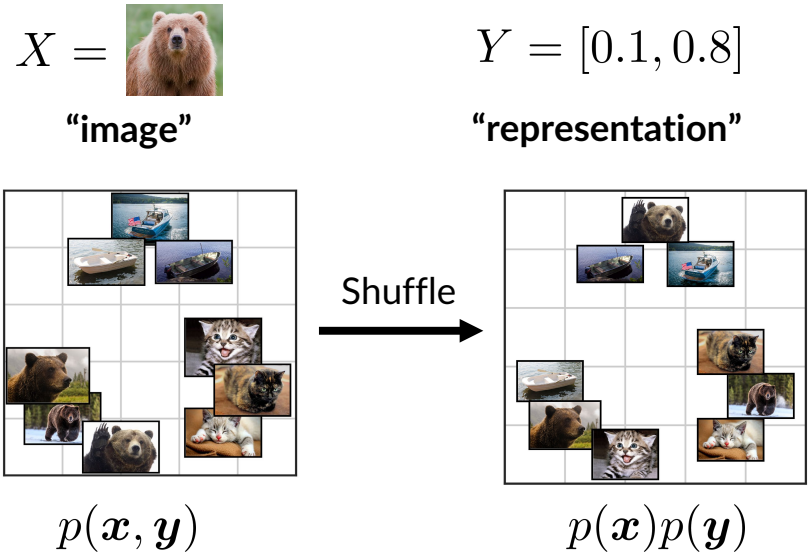


Figure 2.1: **Compression as optimizing a discrepancy.** In this illustration, $\boldsymbol{x}$ are the raw images and $\boldsymbol{y}$ are the corresponding embeddings obtained from a neural networks. Our goal is to maximize the discrepancy between $p(\boldsymbol{x}, \boldsymbol{y})$ and $p(\boldsymbol{x})p(\boldsymbol{y})$ such that the learned representations are more suitable for other downstream tasks.

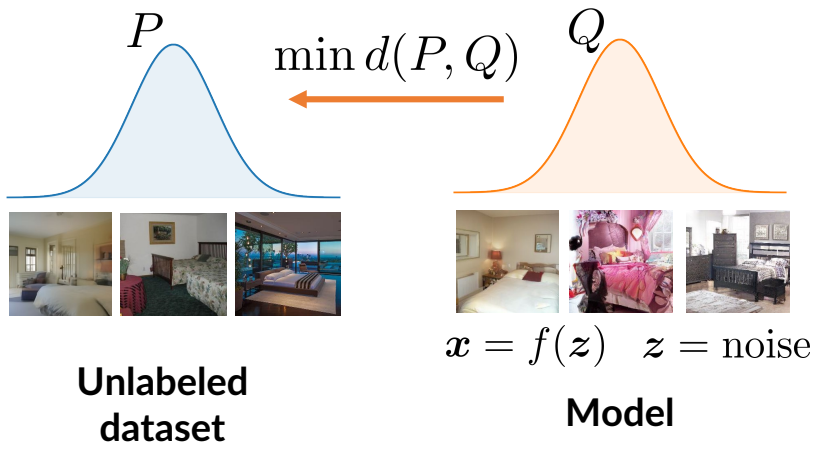


Figure 2.2: **Generation as optimizing a discrepancy.** Our goal is to minimize the discrepancy between the data distribution (represented as the image dataset) and model distribution, such that the model can be used to synthesis new images.

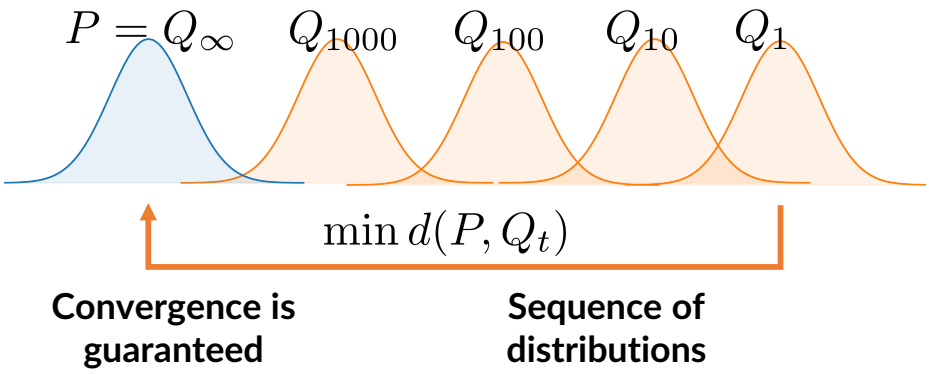


Figure 2.3: **Inference as optimizing a discrepancy.** In Markov chain Monte Carlo methods, our goal is to select an appropriate Markov chain that converges quickly to the stationary distribution; specifically, we minimize the discrepancy between the stationary distribution distribution and the distribution obtained at a certain timestep t.

2.1.3 Inference

Probabilistic inference (or Bayesian inference) is a method of statistical inference which uses Bayes' rule to update the probability for a hypothesis. Suppose that we have a prior distribution for hypotheses $p(\Theta)$ and a likelihood function of the data $\boldsymbol{y}$ conditioned on a certain hypothesis $p(\boldsymbol{y}|\Theta)$, then updated posterior distribution $p(\Theta|\boldsymbol{y})$ can be computed from the Bayes' rule:

$$P := p(\Theta|\boldsymbol{y}) \propto p(\boldsymbol{y}|\Theta)p(\Theta) \tag{2.9}$$

In this dissertation, we discuss two cases where supervised learning can be applied to inference. In Chapter 9, we discuss how we can optimize for more efficient inference procedures; in Chapter 10, we introduce how to use supervised learning to solve an inference problem. Here we discuss more extensively the setup in Chapter 9.

Specifically, we consider the case of Markov chain Monte Carlo (MCMC), which is a general purpose method for sampling from a probability distribution (*e.g.*, the posterior P) [N+11]. Intuitively, MCMC methods construct ergodic Markov chains; let us denote the resulting distribution at time t to be Q_t. A nice property of MCMC is that its stationary distribution (*i.e.*, the distribution reached when the Markov chain was sampled for infinite time, or Q_∞) is guaranteed to be the desired probability distribution for a large family of valid transitions. However, as we have only limited computational power, it is more desirable to select more efficient transitions that converge

faster; this gives rise to the following objective:

$$\text{minimize} \quad D(P, Q_t) \tag{2.10}$$

for some time step t (or many different time steps). Smaller divergence values would suggest that the Markov chain converges faster to the stationary distribution P.

2.2 Variational Methods for Comparing Distributions

As we do not have much additional information of the distributions P and Q beyond samples, we would have to rely on approximations to estimate and/or optimize the divergences. In this section, we briefly describe two families of variational methods of estimating divergences, one based on a lower bound and the other based on the upper bound; choice of the specific variational approach may depend on the specific setup. For example, if we wish to minimize the divergence (*e.g.*, generative modeling), then upper bounds can be more stable to optimize than lower bounds; if the reverses is true (*e.g.*, representation learning), then we might prefer lower bounds instead.

Jensen's inequality Jensen's inequality is a general inequality that appears in many forms depending on the context, from which many results in the dissertation are derived.

Lemma 1 (Jensen's inequality)**.** *Let $\psi : \mathcal{X} \to \mathbb{R}$ be a convex function, and $P \in \mathcal{P}(\mathcal{X})$ a distribution with sample space $\mathcal{X}$. Then:*

$$\mathbb{E}_{\boldsymbol{x} \sim P}[\psi(\boldsymbol{x})] \geq \psi(\mathbb{E}_{\boldsymbol{x} \sim P}[\boldsymbol{x}]) \tag{2.11}$$

Fenchel duality For functions $g : \mathcal{X} \to \mathbb{R}$ defined over a Banach space $\mathcal{X}$, the Fenchel dual of g, $g^* : \mathcal{X}^* \to \mathbb{R}$ is defined over the dual space $\mathcal{X}^*$ by:

$$g^*(\boldsymbol{x}^*) := \sup_{\boldsymbol{x} \in \mathcal{X}} \langle \boldsymbol{x}^*, \boldsymbol{x} \rangle - g(\boldsymbol{x}), \tag{2.12}$$

where $\langle \cdot, \cdot \rangle$ is the duality paring. For example, the dual space of $\mathbb{R}^d$ is also $\mathbb{R}^d$ and $\langle \cdot, \cdot \rangle$ is the usual inner product [Roc70].

2.2.1 Lower bounds of f-divergences

We first introduce a known variational lower bound to f-divergences [NWJ08], which serves as the basis of using supervised learning methods. [NWJ10] derive a general variational method to estimate f-divergences given only samples from P and Q.

Lemma 2 ([NWJ10]). *$\forall P, Q \in \mathcal{P}(\mathcal{X})$ such that $P \ll Q$, and differentiable f:*

$$D_f(P\|Q) = \sup_{T \in L^\infty(Q)} I_f(T; P, Q), \tag{2.13}$$

$$\textit{where} \quad I_f(T; P, Q) := \mathbb{E}_P[T(\boldsymbol{x})] - \mathbb{E}_Q[f^*(T(\boldsymbol{x}))] \tag{2.14}$$

and the supremum is achieved when $T = f'(\mathrm{d}P/\,\mathrm{d}Q)$.

Example 1: Generative Adversarial Networks In generative adversarial networks (GANs, [GPAM+14]), the goal is to fit an (empirical) data distribution P_{data} with an implicit generative model over $\mathcal{X}$, denoted as $Q_\theta \in \mathcal{P}(\mathcal{X})$. Q_θ is defined implicitly via the process $X = G_\theta(Z)$, where Z is a random variable with a fixed prior distribution. Assuming access to *i.i.d.* samples from P_{data} and Q_θ, a discriminator $T_\phi : \mathcal{X} \to [0, 1]$ is used to classify samples from the two distributions, leading to the following objective:

$$\min_\theta \max_\phi \mathbb{E}_{\boldsymbol{x} \sim P_{\text{data}}}[\log T_\phi(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim Q_\theta}[\log(1 - T_\phi(\boldsymbol{x}))].$$

If we have infinite samples from P_{data}, and T_ϕ and Q_θ are sufficiently expressive, then the above minimax objective will reach an equilibrium where $Q_\theta = P_{\text{data}}$ and $T_\phi(\boldsymbol{x}) = 1/2$ for all $\boldsymbol{x} \in \mathcal{X}$.

In the context of GANs, [NCT16] proposed variational f-divergence minimization where one estimates $D_f(p_{\text{data}}\|Q_\theta)$ with the variational lower bound in equation 2.13 while minimizing over θ the estimated divergence. This leads to the f-GAN objective:

$$\min_\theta \max_\phi \mathbb{E}_{\boldsymbol{x} \sim P_{\text{data}}}[T_\phi(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x} \sim Q_\theta}[f^*(T_\phi(\boldsymbol{x}))], \tag{2.15}$$

where the original GAN objective is a special case for $f(u) = u \log u - (u+1)\log(u+1) + 2\log 2$.

Example 2: Contrastive Predictive Coding A variety of mutual information estimators have been proposed for representation learning [NWJ08, vdOKV+16, BBR+18, POvdO+19]. Contrastive

predictive coding (CPC, also known as InfoNCE [vdOLV18]), poses the mutual information estimation problem as an m-class classification problem. Here, the goal is to distinguish a *positive* pair $(\boldsymbol{x}, \boldsymbol{y}) \sim p(\boldsymbol{x}, \boldsymbol{y})$ from $(m-1)$ *negative* pairs $(\boldsymbol{x}, \overline{\boldsymbol{y}}) \sim p(\boldsymbol{x})p(\boldsymbol{y})$. If the optimal classifier is able to distinguish positive and negative pairs easily, it means $\boldsymbol{x}$ and $\boldsymbol{y}$ are tied to each other, indicating high mutual information.

For a batch of n positive pairs $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$, the CPC objective is defined as[1]:

$$L(g) := \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} \log \frac{m \cdot g(\boldsymbol{x}_i, \boldsymbol{y}_i)}{g(\boldsymbol{x}_i, \boldsymbol{y}_i) + \sum_{j=1}^{m-1} g(\boldsymbol{x}_i, \overline{\boldsymbol{y}_{i,j}})}\right] \tag{2.16}$$

for some positive critic function $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+$, where the expectation is taken over n positive pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i) \sim p(\boldsymbol{x}, \boldsymbol{y})$ and $n(m-1)$ negative pairs $(\boldsymbol{x}_i, \overline{\boldsymbol{y}_{i,j}}) \sim p(\boldsymbol{x})p(\boldsymbol{y})$.

In Chapter 4, we present a more detailed argument showing that the CPC objective is a lower bound to mutual information.

2.2.2 Upper bounds of f-divergences

Let us consider the case of hypothetical latent variable models $p(\boldsymbol{x}, \boldsymbol{z})$ and $q(\boldsymbol{x}, \boldsymbol{z})$ where:

$$\int p(\boldsymbol{x}, \boldsymbol{z})\, \mathrm{d}\boldsymbol{z} = p(\boldsymbol{x}), \quad \int q(\boldsymbol{x}, \boldsymbol{z})\, \mathrm{d}\boldsymbol{z} = q(\boldsymbol{x}). \tag{2.17}$$

We have the following upper bound of $D_f(p(\boldsymbol{x}), q(\boldsymbol{x}))$ from Jensen's inequality:

$$D_f(p(\boldsymbol{x}, \boldsymbol{z}), q(\boldsymbol{x}, \boldsymbol{z})) = \int q(\boldsymbol{x}) \int q(\boldsymbol{z}|\boldsymbol{x}) f\left(\frac{p(\boldsymbol{x}, \boldsymbol{z})}{q(\boldsymbol{x}, \boldsymbol{z})}\right) \mathrm{d}\boldsymbol{z}\, \mathrm{d}\boldsymbol{x} \tag{2.18}$$

$$\geq \int q(\boldsymbol{x}) f\left(\int q(\boldsymbol{z}|\boldsymbol{x}) \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q(\boldsymbol{x}, \boldsymbol{z})}\, \mathrm{d}\boldsymbol{z}\right) \mathrm{d}\boldsymbol{x} = D_f(p(\boldsymbol{x}), q(\boldsymbol{x})) \tag{2.19}$$

A special case is the KL divergence, where

$$D_{\mathrm{KL}}(q(\boldsymbol{x}, \boldsymbol{z}), p(\boldsymbol{x}, \boldsymbol{z})) = \mathbb{E}_{q(\boldsymbol{x})}\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log q(\boldsymbol{z}|\boldsymbol{x}) - \log p(\boldsymbol{x}, \boldsymbol{z})] - H(q(\boldsymbol{z})) \tag{2.20}$$

where $H(q(\boldsymbol{x})) = \mathbb{E}_{q(\boldsymbol{x})}[\log q(\boldsymbol{x})]$ denotes the entropy. If we denote $q(\boldsymbol{x})$ as the data distribution, then its entropy is constant with respect to the model distribution and thus can be safely ignored; this is the foundation to the evidence lower bound (ELBO) in variational inference, and can be used to derive the variational autoencoder (VAE) objective function [KW13, RMW14]. Other examples

[1] We suppress the dependencies on n and m in $L(g)$ (and in subsequent objectives) for conciseness.

of applying upper bounds of f-divergences include the Neural Density Imitation method [KJS+20] and Denoising Diffusion Probabilistic Models [HJA20, SME20]; for conciseness, we only discuss the VAE objective in this chapter.

Example: Variational Autoencoders A latent variable generative model (LVGM) is posed as a conditional distribution $p_\theta : \mathcal{Z} \to \mathcal{P}(\mathcal{X})$ from a latent variable $\boldsymbol{z}$ to a generated sample $\boldsymbol{x}$, parametrized by θ. To acquire new samples, LVGMs draw random latent variables $\boldsymbol{z}$ from some distribution $p(\boldsymbol{z})$ and map them to image samples through $\boldsymbol{x} \sim p_\theta(\boldsymbol{x}|\boldsymbol{z})$. Most LVGMs are built on top of four paradigms: variational autoencoders (VAEs, [KW13, RM15]), Normalizing Flows (NFs, [DKB14, DSDB16]), Generative Adversarial Networks (GANs, [GPAM+14]), and diffusion / score-based generative models [HJA20, SE19c].

Particularly, VAEs use an inference model from $\boldsymbol{x}$ to $\boldsymbol{z}$ for training. Denoting the inference distribution from $\boldsymbol{x}$ to $\boldsymbol{z}$ as $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and the generative distribution from $\boldsymbol{z}$ to $\boldsymbol{x}$ as $p_\theta(\boldsymbol{x}|\boldsymbol{z})$, VAEs are trained by minimizing the following upper bound of negative log-likelihood:

$$
\begin{aligned}
L_{\text{VAE}} &= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[-\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] + D_{\text{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x}), p(\boldsymbol{z}))] && (2.21)\\
&= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[-\log p_\theta(\boldsymbol{x},\boldsymbol{z}) - \log q_\phi(\boldsymbol{z}|\boldsymbol{x}))] && (2.22)\\
&= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[-\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right]\right] && (2.23)\\
&\geq \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[-\log \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right]\right] = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[-\log p_\theta(\boldsymbol{x})] && (2.24)
\end{aligned}
$$

where p_{data} is the data distribution and D_{KL} is the KL-divergence.

Part I

Compression via Supervised Learning

Chapter 3

Compression via Mutual Information

Let us begin our journey with supervised learning for compression, *i.e.*, representation learning. Much prior work on data compression rely on hand-designed procedures for specific applications (such as recovering original images from JPEG and PNG), which are not necessarily optimal for other data modalities and tasks (such as image retrieval). This task is also difficult to supervise directly: unlike labels of an image, it is hard to have a consensus over how to compress the data (*e.g.* from crowdsourcing). A common principle that is adopted in representation learning is based on the mutual information maximization principle (InfoMax), namely, we wish to maximize the mutual information between the original data and the learned representations. Variational approaches based on neural networks are showing promise for estimating mutual information (MI) between high dimensional variables. However, they can be difficult to use in practice due to poorly understood bias/variance trade-offs.

In this chapter, we theoretically show that, under some conditions, estimators such as MINE exhibit variance that could grow exponentially with the true amount of underlying MI. We also empirically demonstrate that existing estimators fail to satisfy basic self-consistency properties of MI, such as data processing and additivity under independence. Based on a unified perspective of variational approaches, we develop a new estimator that focuses on variance reduction. Empirical results on standard benchmark tasks demonstrate that our proposed estimator exhibits improved bias-variance trade-offs on standard benchmark tasks.

This chapter is previously published as [SE20c].

3.1 Introduction

Mutual information (MI) estimation and optimization are crucial to many important problems in machine learning, such as representation learning [CDH+16, ZSE18a, TZ15, HAP+18] and reinforcement learning [PAED17, vdOLV18]. However, estimating mutual information from samples is challenging [MS18] and traditional parametric and non-parametric approaches [NBVS04, GVSG15, GKOV17] struggle to scale up to modern machine learning problems, such as estimating the MI between images and learned representations.

Recently, there has been a surge of interest in MI estimation with variational approaches [BA03, NWJ10, DV75], which can be naturally combined with deep learning methods [AFDM16, vdOLV18, POvdO+19]. Despite their empirical effectiveness in downstream tasks such as representation learning [HFLM+18, VFH+18], their effectiveness *for MI estimation* remains unclear. In particular, higher estimated MI between observations and learned representations do not seem to indicate improved predictive performance when the representations are used for downstream supervised learning tasks [TDR+19].

In this chapter, we discuss two limitations of variational approaches to MI estimation. First, we theoretically demonstrate that the variance of certain estimators, such as the Mutual Information Neural Estimator (MINE, [BBR+18]), could grow *exponentially* with the ground truth MI, leading to poor bias-variance trade-offs. Second, we propose a set of *self-consistency tests* over basic properties of MI, and empirically demonstrate that all considered variational estimators fail to satisfy critical properties of MI, such as data processing and additivity under independence. These limitations challenge the effectiveness of these methods for estimating or optimizing MI.

To mitigate these issues, we propose a unified perspective over variational estimators treating variational MI estimation as an optimization problem over (valid) likelihood ratios. This view highlights the role of estimating partition functions, which is the culprit of high variance issues in MINE. To address this issue, we propose to improve MI estimation via variance reduction techniques for partition function estimation. Empirical results demonstrate that our estimators have much better bias-variance trade-off compared to existing methods on standard benchmark tasks.

3.2 Background and Related Work

Additional Notations We use $\hat{I}_E$ to denote an estimator for I_E where we replace expectations with sample averages.

3.2.1 Variational Mutual Inforamtion Estimation

The mutual information between two random variables X and Y is the KL divergence between the joint and the product of marginals:

$$I(X;Y) = D_{\mathrm{KL}}(P(X,Y)\|P(X)P(Y)) \tag{3.1}$$

which we wish to estimate using samples from $P(X,Y)$; in certain cases we may know the density of marginals (e.g. $P(X)$). There are a wide range of variational approaches to variational MI estimation. A traditional variational information maximization approach uses the following result [BA03]:

Lemma 3 (Barber-Agakov (BA))**.** *For two random variables X and Y:*

$$I(X;Y) = \sup_{q_\phi} \left\{ \mathbb{E}_{P(X,Y)} \left[\log q_\phi(\boldsymbol{x}|\boldsymbol{y}) - \log p(\boldsymbol{x}) \right] =: I_{\mathrm{BA}}(q_\phi) \right\} \tag{3.2}$$

where $q_\phi : \mathcal{Y} \to \mathcal{P}(\mathcal{X})$ is a valid conditional distribution over $\mathcal{X}$ given $\boldsymbol{y} \in \mathcal{Y}$ and $p(\boldsymbol{x})$ is the probability density function of the marginal distribution $P(X)$.

Another family of approaches perform MI estimation through variational lower bounds to KL divergences. For example, the Mutual Information Neural Estimator (MINE, [BBR+18]) applies the following lower bound to KL divergences [DV75].

Lemma 4 (Donsker-Varadahn (DV))**.** *$\forall P, Q \in \mathcal{P}(\mathcal{X})$ such that $P \ll Q$,*

$$D_{\mathrm{KL}}(P\|Q) = \sup_{T \in L^\infty(Q)} \left\{ \mathbb{E}_P[T] - \log \mathbb{E}_Q[e^T] =: I_{\mathrm{MINE}}(T) \right\}. \tag{3.3}$$

In order to estimate mutual information, one could set $P = P(X,Y)$ and $Q = P(X)P(Y)$, T as a parametrized neural network (e.g. $T_\theta(\boldsymbol{x}, \boldsymbol{y})$ parametrized by θ), and obtain the estimate by optimizing the above objective via stochastic gradient descent over mini-batches. However, the corresponding estimator $\hat{I}_{\mathrm{MINE}}$ (where we replace the expectations in Eq. (3.3) with sample averages) is biased, leading to biased gradient estimates; [BBR+18] propose to reduce bias via estimating the partition function $\mathbb{E}_Q[e^T]$ with exponential moving averages of mini-batches.

The variational f-divergence estimation approach [NWJ10, NCT16] considers lower bounds on f-divergences which can be specialize to KL divergence, and subsequently to mutual information estimation (this is a special case to Lemma 2 in Chapter 2):

Lemma 5 (Nyugen et al. (NWJ)). *$\forall P, Q \in \mathcal{P}(\mathcal{X})$ such that $P \ll Q$,*

$$D_{\mathrm{KL}}(P\|Q) = \sup_{T \in L^\infty(Q)} \left\{ \mathbb{E}_P[T] - \mathbb{E}_Q[e^{T-1}] =: I_{\mathrm{NWJ}}(T) \right\} \tag{3.4}$$

and $D_{\mathrm{KL}}(P\|Q) = I_{\mathrm{NWJ}}(T)$ when $T = \log(\mathrm{d}P/\,\mathrm{d}Q) + 1$.

The supremum over T is a invertible function of the density ratio $\mathrm{d}P/\,\mathrm{d}Q$, so one could use this approach to estimate density ratios by inverting the function [NWJ10, NCT16, GE17]. The corresponding mini-batch estimator (denoted as $\hat{I}_{\mathrm{NWJ}}$) is unbiased, so unlike MINE, this approach does not require special care to reduce bias in gradients.

Contrastive Predictive Coding (CPC, [vdOLV18]) considers the following objective:

$$I_{\mathrm{CPC}}(f_\theta) := \mathbb{E}_{P^n(X,Y)}\left[\frac{1}{n}\sum_{i=1}^{n} \log \frac{f_\theta(\boldsymbol{x}_i, \boldsymbol{y}_i)}{\frac{1}{n}\sum_{j=1}^{n} f_\theta(\boldsymbol{x}_i, \boldsymbol{y}_j)}\right] \tag{3.5}$$

where $f_\theta : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ is a neural network parametrized by θ and $P^n(X, Y)$ denotes the joint pdf for n i.i.d. random variables sampled from $P(X, Y)$. CPC generally has less variance but is more biased because its estimate does not exceed $\log n$, where n is the batch size [vdOLV18, POvdO+19]. While one can further reduce the bias with larger n, the number of evaluations needed for estimating each batch with f_θ is n^2, which scales poorly. To address the high-bias issue of CPC, Poole *et al.*, [POvdO+19] proposed an interpolation between I_{CPC} and I_{NWJ} to obtain more fine-grained bias-variance trade-offs.

Table 3.1: **Summarization of variational estimators of mutual information.** The $\in \Delta(Q)$ column denotes whether the estimator is a valid density ratio wrt. Q. (✓) means any parameterization is valid; $(n \to \infty)$ means any parameterization is valid as the batch size grows to infinity; $(\mathrm{tr} \to \infty)$ means only the optimal parametrization is valid (infinite training cost).

Category	Estimator	Params	$\Gamma(r; Q_n)$	$\in \Delta(Q)$
Gen.	$\hat{I}_{\mathrm{BA}}$	q_ϕ	$q_\phi(\boldsymbol{x}\|\boldsymbol{y})/p(\boldsymbol{x})$	✓
	$\hat{I}_{\mathrm{GM}}$ (Eq. (3.9))	p_θ, p_ϕ, p_ψ	$p_\theta(\boldsymbol{x}, \boldsymbol{y})/p_\phi(\boldsymbol{x})p_\psi(\boldsymbol{y})$	$\mathrm{tr} \to \infty$
Disc.	$\hat{I}_{\mathrm{MINE}}$	T_θ	$e^{T_\theta(\boldsymbol{x},\boldsymbol{y})}/\mathbb{E}_{Q_n}[e^{T_\theta(\boldsymbol{x},\boldsymbol{y})}]$	$n \to \infty$
	$\hat{I}_{\mathrm{CPC}}$	f_θ	$f_\theta(\boldsymbol{x}, \boldsymbol{y})/\mathbb{E}_{P_n(Y)}[f_\theta(\boldsymbol{x}, \boldsymbol{y})]$	✓
	$\hat{I}_{\mathrm{SMILE}}$	T_θ, τ	$e^{T_\theta(\boldsymbol{x},\boldsymbol{y})}/\mathbb{E}_{Q_n}[e^{\mathrm{clip}(T_\theta(\boldsymbol{x},\boldsymbol{y}),-\tau,\tau)}]$	$n, \tau \to \infty$

3.3 Variational mutual information estimation as optimization over density ratios

In this section, we unify several existing methods for variational mutual information estimation. We first show that variational mutual information estimation can be formulated as a constrained optimization problem, where the feasible set is $\Delta(Q)$, i.e. the valid density ratios with respect to Q.

Theorem 1. *$\forall P, Q \in \mathcal{P}(\mathcal{X})$ such that $P \ll Q$ we have*

$$D_{\mathrm{KL}}(P\|Q) = \sup_{r \in \Delta(Q)} \mathbb{E}_P[\log r] \tag{3.6}$$

where the supremum is achived when $r = \mathrm{d}P/\,\mathrm{d}Q$.

We defer the proof in Appendix A.1. The above argument works for KL divergence between general distributions, but in this paper we focus on the special case of mutual information estimation. For the remainder of the paper, we use P to represent the short-hand notation for the joint distribution $P(X, Y)$ and use Q to represent the short-hand notation for the product of marginals $P(X)P(Y)$.

3.3.1 A summary of existing variational methods

From Theorem 1, we can describe a general approach to variational MI estimation:

1. Obtain a density ratio estimate – denote the solution as r;
2. Project r to be close to $\Delta(Q)$ – in practice we only have samples from Q, so we denote the solution as $\Gamma(r; Q_n)$, where Q_n is the empirical distribution of n i.i.d. samples from Q;
3. Estimate mutual information with $\mathbb{E}_P[\log \Gamma(r; Q_n)]$.

We illustrate two examples of variational mutual information estimation that can be summarized with this approach. In the case of Barber-Agakov, the proposed density ratio estimate is $r_{\mathrm{BA}} = q_\phi(\boldsymbol{x}|\boldsymbol{y})/p(\boldsymbol{x})$ (assuming that $p(\boldsymbol{x})$ is known), which is guaranteed to be in $\Delta(Q)$ because

$$\mathbb{E}_Q\left[q_\phi(\boldsymbol{x}|\boldsymbol{y})/p(\boldsymbol{x})\right] = \int q_\phi(\boldsymbol{x}|\boldsymbol{y})/p(\boldsymbol{x})\mathrm{d}P(x)\mathrm{d}P(y) = 1, \qquad \Gamma_{\mathrm{BA}}(r_{\mathrm{BA}}, Q_n) = r_{\mathrm{BA}} \tag{3.7}$$

for all conditional distributions q_ϕ. In the case of MINE / Donsker-Varadahn, the *logarithm* of the density ratio is estimated with $T_\theta(\boldsymbol{x}, \boldsymbol{y})$; the corresponding density ratio might not be normalized,

so one could apply the following normalization for n samples:

$$\mathbb{E}_{Q_n}\left[e^{T_\theta}/\mathbb{E}_{Q_n}[e^{T_\theta}]\right] = 1, \qquad \Gamma_{\mathrm{MINE}}(e^{T_\theta}, Q_n) = e^T_\theta/\mathbb{E}_{Q_n}[e^{T_\theta}] \tag{3.8}$$

where $\mathbb{E}_{Q_n}[e^{T_\theta}]$ (the sample average) is an unbiased estimate of the partition function $\mathbb{E}_Q[e^{T_\theta}]$; $\Gamma_{\mathrm{DV}}(e^{T_\theta}, Q_n) \in \Delta(Q)$ is only true when $n \to \infty$. Similarly, we show I_{CPC} is a lower bound to MI in Corollary 7, Appendix A.1, providing an alternative proof to the one in [POvdO+19].

These examples demonstrate that different mutual information estimators can be obtained in a procedural manner by implementing the above steps, and one could involve different objectives at each step. For example, one could estimate density ratio via logistic regression [HFLM+18, POvdO+19, MAK19] while using I_{NWJ} or I_{MINE} to estimate MI. While logistic regression does not optimize for a lower bound for KL divergence, it provides density ratio estimates between P and Q which could be used for subsequent steps.

3.3.2 Generative and discriminative approaches to MI estimation

The above discussed variational mutual information methods can be summarized into two broad categories based on how the density ratio is obtained.

- The *discriminative approach* estimates the density ratio $\mathrm{d}P/\mathrm{d}Q$ directly; examples include the MINE, NWJ and CPC estimators.
- The *generative approach* estimates the densities of P and Q separately; examples include the BA estimator where a conditional generative model is learned. In addition, we describe a generative approach that explicitly learns generative models (GM) for $P(X, Y)$, $P(X)$ and $P(Y)$:

$$I_{\mathrm{GM}}(p_\theta, p_\phi, p_\psi) := \mathbb{E}_P[\log p_\theta(\boldsymbol{x}, \boldsymbol{y}) - \log p_\phi(\boldsymbol{x}) - \log p_\psi(\boldsymbol{y})], \tag{3.9}$$

  where p_θ, p_ϕ, p_ψ are maximum likelihood estimates of $P(X, Y)$, $P(X)$ and $P(Y)$ respectively. We can learn the three distributions with generative models, such as VAE [KW13] or Normalizing flows [DSDB16], from samples.

We summarize various generative and discriminative variational estimators in Table 3.1.

Differences between two approaches While both *generative* and *discriminative* approaches can be summarized with the procedure in Section 3.3.1, they imply different choices in modeling,

estimation and optimization.

- On the modeling side, the *generative approaches* might require more stringent assumptions on the architectures (e.g. likelihood or evidence lower bound is tractable), whereas the *discriminative approaches* do not have such restrictions.
- On the estimation side, *generative approaches* do not need to consider samples from the product of marginals $P(X)P(Y)$ (since it can model $P(X,Y)$, $P(X)$, $P(Y)$ separately), yet the *discriminative approaches* require samples from $P(X)P(Y)$; if we consider a mini-batch of size n, the number of evaluations for generative approaches is $\Omega(n)$ whereas that for discriminative approaches it could be $\Omega(n^2)$.
- On the optimization side, discriminative approaches may need additional projection steps to be close to $\Delta(Q)$ (such as I_{MINE}), while generative approaches might not need to perform this step (such as I_{BA}).

3.4 Limitations of existing variational estimators

3.4.1 Good discriminative estimators require exponentially large batches

In the $\hat{I}_{\mathrm{NWJ}}$ and $\hat{I}_{\mathrm{MINE}}$ estimators, one needs to estimate the "partition function" $\mathbb{E}_Q[r]$ for some density ratio estimator r; for example, $\hat{I}_{\mathrm{MINE}}$ needs this in order to perform the projection step $\Gamma_{\mathrm{MINE}}(r, Q_n)$ in Eq (3.8). Note that the I_{NWJ} and I_{MINE} lower bounds are maximized when r takes the optimal value $r^\star = \mathrm{d}P/\,\mathrm{d}Q$. However, the sample averages $\hat{I}_{\mathrm{MINE}}$ and $\hat{I}_{\mathrm{NWJ}}$ of $\mathbb{E}_Q[r^\star]$ could have a variance that scales *exponentially* with the ground-truth MI; we show this in Theorem 2.

Theorem 2. *Assume that the ground truth density ratio $r^\star = \mathrm{d}P/\,\mathrm{d}Q$ and $\mathrm{Var}_Q[r^\star]$ exist. Let Q_n denote the empirical distribution of n i.i.d. samples from Q and let $\mathbb{E}_{Q_n}$ denote the sample average over Q_n. Then under the randomness of the sampling procedure, we have:*

$$\mathrm{Var}_Q[\mathbb{E}_{Q_n}[r^\star]] \geq \frac{e^{D_{\mathrm{KL}}(P\|Q)} - 1}{n} \tag{3.10}$$

$$\lim_{n\to\infty} n\mathrm{Var}_Q[\log \mathbb{E}_{Q_n}[r^\star]] \geq e^{D_{\mathrm{KL}}(P\|Q)} - 1. \tag{3.11}$$

We defer the proof to Appendix A.1. Note that in the theorem above, we assume the ground truth density ratio $r^\star$ is already obtained, which is the optimal ratio for NWJ and MINE estimators. As a natural consequence, the NWJ and MINE estimators under the optimal solution could exhibit

variances that grow exponentially with the ground truth MI (recall that in our context MI is a KL divergence). One could achieve smaller variances with some $r \neq r^\star$, but this guarantees looser bounds and higher bias.

Corollary 1. *Assume that the assumptions in Theorem 2 hold. Let P_m and Q_n be the empirical distributions of m i.i.d. samples from P and n i.i.d. samples from Q, respectively. Define*

$$I_{\mathrm{NWJ}}^{m,n} := \mathbb{E}_{P_m}[\log r^\star + 1] - \mathbb{E}_{Q_n}[r^\star] \tag{3.12}$$

$$I_{\mathrm{MINE}}^{m,n} := \mathbb{E}_{P_m}[\log r^\star] - \log \mathbb{E}_{Q_n}[r^\star] \tag{3.13}$$

where $r^\star = \mathrm{d}P/\mathrm{d}Q$. Then under the randomness of the sampling procedure, we have $\forall m \in \mathbb{N}$:

$$\mathrm{Var}_{P,Q}[I_{\mathrm{NWJ}}^{m,n}] \geq (e^{D_{\mathrm{KL}}(P\|Q)} - 1)/n \tag{3.14}$$

$$\lim_{n\to\infty} n\mathrm{Var}_{P,Q}[I_{\mathrm{MINE}}^{m,n}] \geq e^{D_{\mathrm{KL}}(P\|Q)} - 1. \tag{3.15}$$

This high variance phenomenon has been empirically observed in [POvdO+19] (Figure 3) for $\hat{I}_{\mathrm{NWJ}}$ under various batch sizes, where the log-variance scales linearly with MI. We also demonstrate this in Figure 3.2 (Section 3.6.1). In order to keep the variance of $\hat{I}_{\mathrm{MINE}}$ and $\hat{I}_{\mathrm{NWJ}}$ relatively constant with growing MI, one would need a batch size of $n = \Theta(e^{D_{\mathrm{KL}}(P\|Q)})$. $\hat{I}_{\mathrm{CPC}}$ has small variance, but it would need $n \geq e^{D_{\mathrm{KL}}(P\|Q)}$ to have small bias, as its estimations are bounded by $\log n$.

3.4.2 Self-consistency issues for mutual information estimators

If we consider $\mathcal{X}$, $\mathcal{Y}$ to be high-dimensional, estimation of mutual information becomes more difficult. The density ratio between $P(X, Y)$ and $P(X)P(Y)$ could be very difficult to estimate from finite samples without proper parametric assumptions [MS18, ZRY+18]. Additionally, the exact value of mutual information is dependent on the definition of the sample space; given finite samples, whether the underlying random variable is assumed to be discrete or continuous will lead to different measurements of mutual information (corresponding to entropy and differential entropy, respectively).

In machine learning applications, however, we are often more interested in maximizing or minimizing mutual information (estimates), rather than estimating its exact value. For example, if an estimator is off by a constant factor, it would still be useful for downstream applications, even though it can be highly biased. To this end, we propose a set of *self-consistency* tests for any MI estimator $\hat{I}$, based on properties of mutual information:

1. (Independence) if X and Y are independent, then $\hat{I}(X;Y) = 0$;

2. (Data processing) for all functions g, h,

$$\hat{I}(X;Y) \geq \hat{I}(g(X);h(Y))$$

   and

$$\hat{I}(X;Y) \approx \hat{I}([X,g(X)];[Y,h(Y)])$$

   where $[\cdot,\cdot]$ denotes concatenation.

3. (Additivity) denote X_1, X_2 as independent random variables that have the same distribution as X (similarly define Y_1, Y_2), then $\hat{I}([X_1,X_2];[Y_1,Y_2]) \approx 2 \cdot \hat{I}(X,Y)$.

These properties holds under both entropy and differential entropy, so they do not depend on the choice of the sample space. While these conditions are necessary but obviously not sufficient for accurate mutual information estimation, we argue that satisfying them is highly desirable for applications such as representation learning [CDH+16] and information bottleneck [TZ15]. Unfortunately, none of the MI estimators we considered above pass all the self-consistency tests when X, Y are images, as we demonstrate below in Section 3.6.2. In particular, the *generative* approaches perform poorly when MI is low (failing in independence and data processing), whereas *discriminative* approaches perform poorly when MI is high (failing in additivity).

3.5 Mutual information estimation via clipped density ratios

To address the high-variance issue in the I_{NWJ} and I_{MINE} estimators, we propose to clip the density ratios when estimating the partition function. We define the following clip function:

$$\mathrm{clip}(v,l,u) = \max(\min(v,u),l) \tag{3.16}$$

For an empirical distribution of n samples Q_n, instead of estimating the partition function via $\mathbb{E}_{Q_n}[r]$, we instead consider $\mathbb{E}_{Q_n}[\mathrm{clip}(r, e^{-\tau}, e^{\tau})]$ where $\tau \geq 0$ is a hyperparameter; this is equivalent to clipping the log density ratio estimator between $-\tau$ and τ.

We can then obtain a following estimator with smoothed partition function estimates:

$$I_{\mathrm{SMILE}}(T_\theta, \tau) := \mathbb{E}_P[T_\theta(\boldsymbol{x},\boldsymbol{y})] - \log \mathbb{E}_Q[\mathrm{clip}(e^{T_\theta(\boldsymbol{x},\boldsymbol{y})}, e^{-\tau}, e^{\tau})] \tag{3.17}$$

where T_θ is a neural network that estimates the log-density ratio (similar to the role of T_θ in $\hat{I}_{\text{MINE}}$). We term this the Smoothed Mutual Information "Lower-bound" Estimator (SMILE) with hyperparameter τ; I_{SMILE} converges to I_{MINE} when $\tau \to \infty$. In our experiments, we consider learning the density ratio with logistic regression, similar to the procedure in Deep InfoMax [HFLM+18].

The selection of τ affects the bias-variance trade-off when estimating the partition function; with a smaller τ, variance is reduced at the cost of (potentially) increasing bias. In the following theorems, we analyze the bias and variance in the worst case for density ratio estimators whose actual partition function is S for some $S \in (0, \infty)$.

Theorem 3. *Let $r(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}_{\geq 0}$ be any non-negative measurable function such that $\int r \mathrm{d}Q = S$, $S \in (0, \infty)$ and $r(\boldsymbol{x}) \in [0, e^K]$. Define $r_\tau(\boldsymbol{x}) = \mathrm{clip}(r(\boldsymbol{x}), e^\tau, e^{-\tau})$ for finite, non-negative τ. If $\tau < K$, then the bias for using r_τ to estimate the partition function of r satisfies:*

$$|\mathbb{E}_Q[r] - \mathbb{E}_Q[r_\tau]| \leq \max\left(e^{-\tau}|1 - Se^{-\tau}|, \left|\frac{1 - e^K e^{-\tau} + S(e^K - e^\tau)}{e^K - e^{-\tau}}\right|\right);$$

if $\tau \geq K$, then

$$|\mathbb{E}_Q[r] - \mathbb{E}_Q[r_\tau]| \leq e^{-\tau}(1 - Se^{-K}).$$

Theorem 4. *The variance of the estimator $\mathbb{E}_{Q_n}[r_\tau]$ (using n samples from Q) satisfies:*

$$\mathrm{Var}[\mathbb{E}_{Q_n}[r_\tau]] \leq \frac{e^\tau - e^{-\tau}}{4n} \tag{3.18}$$

We defer the proofs to Appendix A.1. Theorems 3 and 4 suggest that as we decrease τ, variance is decreased at the cost of potentially increasing bias. However, if S is close to 1, then we could use small τ values to obtain estimators where both variance and bias are small. We further discuss the bias-variance trade-off for a fixed r over changes of τ in Theorem 3 and Corollary 8.

3.6 Experiments

3.6.1 Benchmarking on multivariate Gaussians

First, we evaluate the performance of MI bounds on two toy tasks detailed in [POvdO+19, BBR+18], where the ground truth MI is tractable. The first task (**Gaussian**) is where $(\boldsymbol{x}, \boldsymbol{y})$ are drawn from a 20-d Gaussian distribution with correlation ρ, and the second task (**Cubic**) is the same as **Gaussian**

but we apply the transformation $\boldsymbol{y} \mapsto \boldsymbol{y}^3$. We consider three discriminative approaches (I_{CPC}, I_{NWJ}, I_{SMILE}) and one generative approach (I_{GM}). For the discriminative approaches, we consider the joint critic in [BBR+18] and the separate critic in [vdOLV18]. For I_{GM} we consider invertible flow models [DSDB16]. We train all models for 20k iterations, with the ground truth mutual information increasing by 2 per 4k iterations. More training details are included in Appendix B.1.1[1].

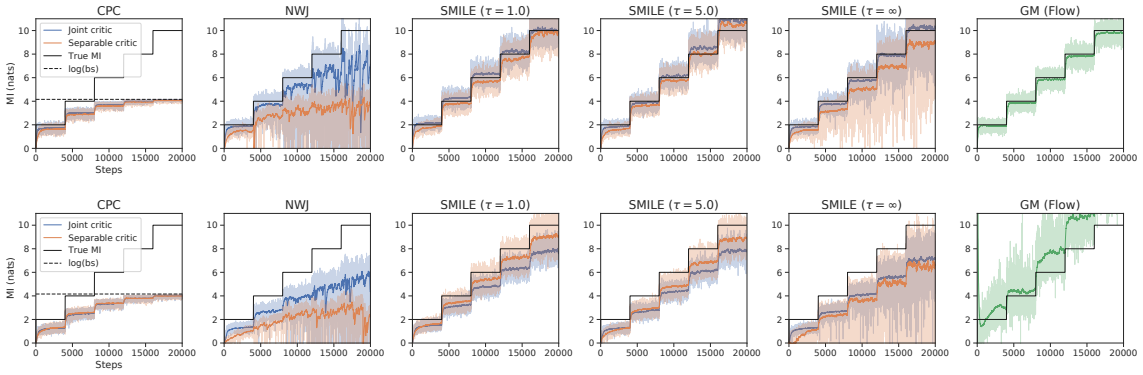


Figure 3.1: **Evaluation over the SMILE estimator.** Performance of mutual information estimation approaches on **Gaussian** (top row) and **Cubic** (bottom row). Left two columns are I_{CPC} and I_{NWJ}, next three columns are I_{SMILE} with $\tau = 1.0, 5.0, \infty$ and the right column is I_{GM} with flow models.

Figure 3.1 shows the estimated mutual information over the number of iterations. In both tasks, I_{CPC} has high bias and I_{NWJ} has high variance when the ground truth MI is high, whereas I_{SMILE} has relatively low bias and low variance across different architectures and tasks. Decreasing τ in the SMILE estimator decreases variances consistently but has different effects over bias; for example, under the joint critic bias is higher for $\tau = 5.0$ in **Gaussian** but lower in **Cubic**. I_{GM} with flow models has the best performance on **Gaussian**, yet performs poorly on **Cubic**, illustrating the importance of model parametrization in the *generative approaches*.

In Figure 3.2, we compare the bias, variance and mean squared error (MSE) of the *discriminative methods*. We observe that the variance of I_{NWJ} increases exponentially with mutual information, which is consistent with our theory in Corollary 1. On the other hand, the SMILE estimator is able to achieve much lower variances with small τ values; in comparison the variance of SMILE when $\tau = \infty$ is similar to that of I_{NWJ} in **Cubic**. In Table B.1, we show that I_{SMILE} can have nearly two orders of magnitude smaller variance than I_{NWJ} while having similar bias. Therefore I_{SMILE} enjoys lower MSE in this benchmark MI estimation task compared to I_{NWJ} and I_{CPC}.

[1]We release our code in https://github.com/ermongroup/smile-mi-estimator.

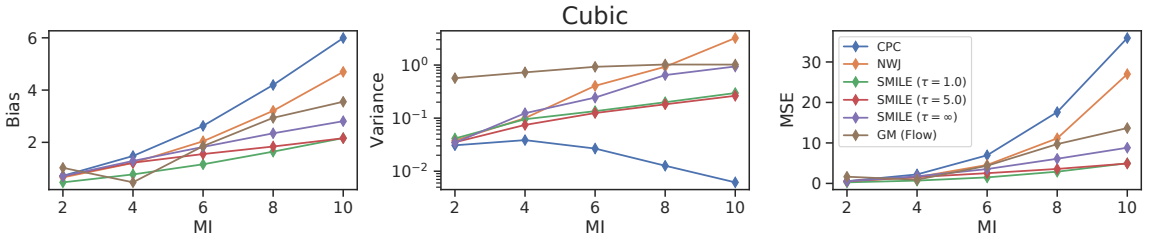


Figure 3.2: Bias / Variance / MSE of various estimators on **Cubic** (right). We display more results for **Gaussian** in Appendix B.1.1.

3.6.2 Self-consistency tests on Images

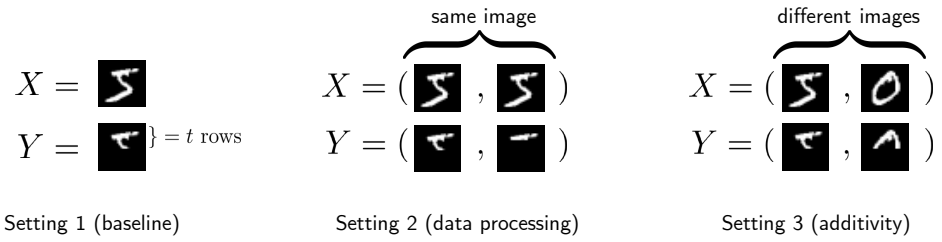


Figure 3.3: Three settings in the self-consistency experiments.

Next, we perform our proposed *self-consistency* tests on high-dimensional images (MNIST and CIFAR10) under three settings, where the ground truth MI is difficult to obtain (if not impossible). These settings are illustrated in Figure 3.3.

1. The first setting is where X is an image and Y is the same image where we mask the bottom rows, leaving the top t rows from X (t is selected before evaluation). The rationale behind this choice of Y is twofold: 1) $I(X;Y)$ should be non-decreasing with t; 2) it is easier (compared to low-d representations) to gain intuition about the amount of information remaining in Y.

2. In the second setting, X corresponds to two identical images, and Y to the top t_1, t_2 rows of the two images ($t_1 \geq t_2$); this considers the "data-processing" property.

3. In the third setting, X corresponds to two independent images, and Y to the top t rows of both; this considers the "additivity" property.

We compare four approaches: I_{CPC}, I_{MINE}, I_{SMILE} and I_{GM}. We use the same CNN architecture for I_{CPC}, I_{MINE} and I_{SMILE}, and use VAEs [KW13] for I_{GM}. We include more experimental details and alternative image processing approaches in Appendix B.1.1.

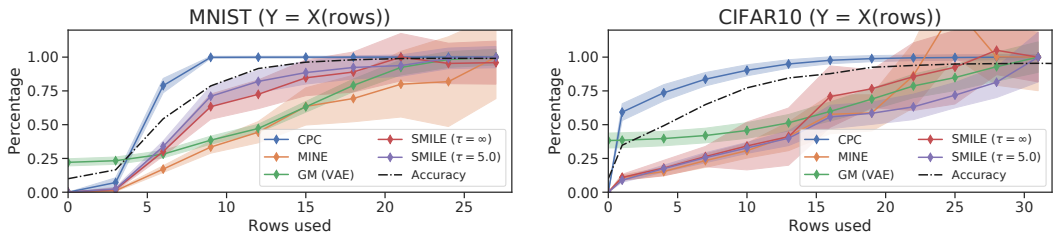


Figure 3.4: Evaluation of $\hat{I}(X;Y)/\hat{I}(X;X)$. X is an image and Y contains the top t rows of X.

Baselines We evaluate the first setting with Y having varying number of rows t in Figure 3.4, where the estimations are normalized by the estimated $\hat{I}(X;X)$. Most methods (except for I_{GM}) predicts zero MI when X and Y are independent, passing the first self-consistency test. Moreover, the estimated MI is non-decreasing with increasing t, but with different slopes. As a reference, we show the validation accuracy of predicting the label where only the top t rows are considered.

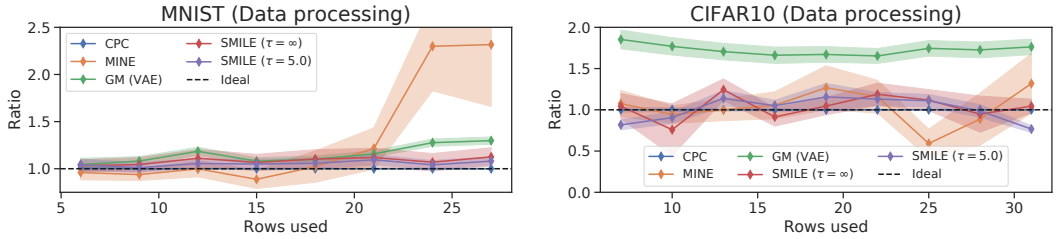


Figure 3.5: Evaluation of $\hat{I}([X,X];[Y,h(Y)])/\hat{I}(X,Y)$, where the ideal value is 1.

Data-processing In the second setting we set $t_2 = t_1 - 3$. Ideally, the estimator should satisfy $\hat{I}([X,X];[Y,h(Y)])/\hat{I}(X,Y) \approx 1$, as additional processing should not increase information. We show the above ratio in Figure 3.5 under varying t_1 values. All methods except for I_{MINE} and I_{GM} performs well in both datasets; I_{GM} performs poorly in CIFAR10 (possibly due to limited capacity of VAE), whereas I_{MINE} performs poorly in MNIST (possibly due to numerical stability issues).

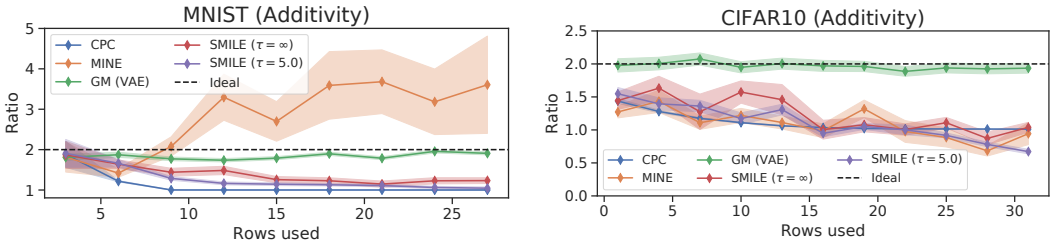


Figure 3.6: Evaluation of $\hat{I}([X_1, X_2]; [Y_1, Y_2])/\hat{I}(X, Y)$, where the ideal value is 2.

Additivity In the third setting, the estimator should double its value compared to the baseline with the same t, i.e. $\hat{I}([X_1, X_2]; [Y_1, Y_2])/\hat{I}(X, Y) \approx 2$. Figure 3.6 shows the above ratio under different values of t. None of the *discriminative* approaches worked well in this case except when t is very small, when t is large this ratio converges to 1 (possibly due to initialization and saturation of the training objective). I_{GM} however, performs near perfectly on this test for all values of t.

3.7 Discussion

In this chapter, we discuss *generative* and *discriminative* approaches to variational mutual information estimation and demonstrate their limitations. We show that estimators based on I_{NWJ} and I_{MINE} are prone to high variances when estimating with mini-batches, inspiring our I_{SMILE} estimator that improves performances on benchmark tasks. However, none of the approaches are good enough to pass the *self-consistency* tests. The *generative* approaches perform poorly when MI is small (failing independence and data-processing tests) while the *discriminative* approaches perform poorly when MI is large (failing additivity tests).

These empirical evidences suggest that optimization over these variational estimators are not necessarily related to optimizing MI, so the empirical successes with these estimators might have little connections to optimizing mutual information. Therefore, it would be helpful to acknowledge these limitations and consider alternative measurements of information that are more suited for modern machine learning applications [OLB+19, TDR+19].

Chapter 4

Representation Learning via Classification

In the previous chapter, we have discussed the limitations of existing variational mutual information estimators and how we can improve the estimation by variance reduction. However, being able to accurately estimate mutual information is not enough for unsupervised representation learning, since we often would like to maximize the informativeness of the representations efficiently.

In this chapter, we discuss how we can pose unsupervised representation learning as a multi-label classification problem. Variational mutual information (MI) estimators are widely used in unsupervised representation learning methods such as contrastive predictive coding (CPC). A lower bound on MI can be obtained from a multi-class classification problem, where a critic attempts to distinguish a positive sample drawn from the underlying joint distribution from $(m-1)$ negative samples drawn from a suitable proposal distribution. Using this approach, MI estimates are bounded above by $\log m$, and could thus severely underestimate unless m is very large.

To overcome this limitation, we introduce a novel estimator based on a multi-label classification problem, where the critic needs to jointly identify *multiple* positive samples at the same time. We show that using the same amount of negative samples, multi-label CPC is able to exceed the $\log m$ bound, while still being a valid lower bound of mutual information. We demonstrate that the proposed approach is able to lead to better mutual information estimation, gain empirical improvements in unsupervised representation learning, and beat a current state-of-the-art knowledge distillation method over 10 out of 13 tasks.

This chapter was previous published as [SE20b].

4.1 Introduction

Learning efficient representations from data with minimal supervision is a critical problem in machine learning with significant practical impact [MSC+13, DCLT18, MK13, RNSS18, BMR+20]. Representations obtained using large amounts of unlabeled data can boost performance on downstream tasks where labeled data is scarce. This paradigm is already successful in a variety of domains; for example, representations trained on large amounts of unlabeled images can be used to improve performance on detection [ZZY19, HFW+19, CKNH20].

In the context of learning visual representations, contrastive objectives based on variational mutual information (MI) estimation are among the most successful ones [vdOLV18, BBR+18, DHFLM+18, POvdO+19, TKI19]. One such approach, named Contrastive Predictive Coding (CPC, [vdOLV18]), obtains a lower bound to MI via a multi-class classification problem. In CPC, a critic is generally trained to distinguish a pair of representations from two augmentations of the same image (positive), apart from $(m-1)$ pairs of representations from different images (negative). The representation network is then trained to increase the MI estimates given by the critic. This brings together the two representations from the positive pair and pushes apart the two representations from the negative pairs.

It has been empirically observed that factors leading to better MI estimates, such as training for more iterations and increasing the complexity of the critic [CKNH20, CFGH20], can generally result in improvements over downstream tasks. In the context of CPC, increasing the number of negative samples per positive sample (i.e. increasing m) also helps with downstream performance [WXYL18, HFW+19, CKNH20, TKI19]. This can be explained from a mutual information estimation perspective that CPC estimates are upper bounded by $\log m$, so increasing m could reduce bias when the actual mutual information is much higher [MC18]. However, due to constraints over compute, memory and data, there is a limit to how many negative samples we can obtain per positive sample.

In this chapter, we propose generalizations to CPC that can increase the $\log m$ bound without additional computational costs, thus decreasing bias. We first generalize CPC through by re-weighting the influence of positive and negative samples in the underlying the classification problem. This increases the $\log m$ bound and leads to bias reduction, yet the re-weighted CPC objective is no longer guaranteed to be a lower bound to mutual information.

To this end, we introduce multi-label CPC (ML-CPC) which poses mutual information estimation as a multi-label classification problem. Instead of identifying one positive sample for each

classification task (as in CPC), the critic now simultaneously identifies multiple positive samples that come from the same batch. We prove for ML-CPC that under certain choices of the weights, we can increase the $\log m$ bound and reduce bias, while guaranteeing that the new objective is still lower bounded by mutual information. This provides an practical algorithm whose upper bound is close to the theoretical upper limit by any distribution-free, high-confidence lower bound estimators of mutual information [MS20].

Re-weighted ML-CPC encompasses a range of mutual information lower bound estimators with different bias-variance trade-offs, which can be chosen with minimal impact on the computational costs. We demonstrate the effectiveness of re-weighted ML-CPC over CPC empirically on several tasks, including mutual information estimation, knowledge distillation and unsupervised representation learning. In particular, ML-CPC is able to beat the current state-of-the-art method in knowledge distillation [TKI19] on 10 out of 13 distillation tasks for CIFAR-100.

4.2 Contrastive Predictive Coding and Mutual Information

Oord et al. [vdOLV18] interpreted the CPC objective as a lower bound to MI, but only proved the case for a lower bound approximation of CPC, where a term containing $-\mathbb{E}[\log g]$ is replaced by $-\log \mathbb{E}[g]$; so their arguments alone cannot prove that CPC is a lower bound of mutual information. Poole et al. [POvdO+19] proved a lower bound argument for the objective where $m = n$ and negative samples are tied to other positive samples in the same batch. To bridge the gap between theory (that CPC instantiates InfoMax) and practice (where negative samples can be chosen independently from positive samples of the same batch, such as MoCo [HFW+19]), we present another proof for the general CPC objective as presented in $L(g)$. First, we show the following result for variational lower bounds of KL divergences between general distributions where batches of negative samples are used to estimate the divergence. Then, as mutual information is a KL divergence between two specific distributions, the lower bound argument for CPC simply follows.

Theorem 5. *For all probability measures P, Q over sample space $\mathcal{X}$ such that $P \ll Q$, the following holds for all functions $r : \mathcal{X} \to \mathbb{R}_+$ and integers $m \geq 2$:*

$$D_{\mathrm{KL}}(P\|Q) \geq \mathbb{E}_{\boldsymbol{x}\sim P, \boldsymbol{y}_{1:m-1}\sim Q^{m-1}} \left[\log \frac{m \cdot r(\boldsymbol{x})}{r(\boldsymbol{x}) + \sum_{i=1}^{m-1} r(\boldsymbol{y}_i)} \right]. \tag{4.1}$$

Proof. In Appendix A.2, using the variational representations of f-divergences [NWJ08] and Prop. 6. □

The argument about CPC being a lower bound to MI is simply a corollary of the above statement where P is $p(\boldsymbol{x}, \boldsymbol{y})$ (joint) and Q is $p(\boldsymbol{x})p(\boldsymbol{y})$ (product of marginals); we state the claim below.

Corollary 2. *$\forall n \geq 1, m \geq 2, \forall g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+$, the following is true: $L(g) \leq I(X;Y)$.*

Therefore, one can train g and h to maximize $L(g)$ (recall that L depends on h via $\boldsymbol{y} = h(\boldsymbol{x})$), which is guaranteed to be lower than $I(X;Y)$ in expectation.

CPC is an estimator with high bias For finite m, since $g(\boldsymbol{x}_i, \boldsymbol{y}_i)$ appears in both the numerator and denominator of Equation 2.16 and g is positive, the density ratio estimates can be no larger than m, and the value of $L(g)$ is thus upper bounded by $\log m$ [vdOLV18]. While this is acceptable for certain low dimensional scenarios, this can lead to high-bias if the true mutual information is much larger than $\log m$. In fact, the required m can be unacceptable in high dimensions since MI can scale linearly with dimension, which means an exponential number of negative samples are needed to achieve low bias.

For example, if X and Y are 1000-dimensional random variables where the marginal distribution for each dimension is standard Gaussian, and for each dimension d, X_d and Y_d has a correlation of 0.2, then the mutual information $I(X;Y)$ is around 20.5, which means that m has to be greater than 4×10^8 in order for CPC estimates to approach this value. In comparison, state-of-the-art image representation learning methods use a m that is around 65536 and representation dimensions between 128 to 2048 [WXYL18, HFW+19, CKNH20] due to batch size and memory limitations, as one would need a sizeable batch of positive samples in order to apply batch normalization [IS15].

4.2.1 Re-weighted Contrastive Predictive Coding

Under the computational limitations imposed by m (i.e., we cannot obtain too many negative samples per positive sample), we wish to develop generalizations to CPC that reduce bias while still being lower bounds to the mutual information. We do not consider other types of estimators such as MINE [BBR+18] or NWJ [NWJ08] because they would exhibit high variance on the order of $O(e^{I(X;Y)})$ [SE19b], and thus are much less stable to optimize.

One possible approach is to decrease the weights of the positive sample when calculating the sum in the denominator; this leads to the following objective, called α-CPC:

$$L_\alpha(g) := \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} \log \frac{m \cdot g(\boldsymbol{x}_i, \boldsymbol{y}_i)}{\alpha g(\boldsymbol{x}_i, \boldsymbol{y}_i) + \frac{m-\alpha}{m-1}\sum_{j=1}^{m-1} g(\boldsymbol{x}_i, \overline{\boldsymbol{y}_{i,j}})}\right] \tag{4.2}$$

where the positive sample is weighted by α and negative samples are weighted by $\frac{m-\alpha}{m-1}$. The purpose of adding weights to negative samples is to make sure the the weights sum to m, like in the original case where each sample has weight 1 and there are m samples in total. Clearly, the original CPC objective is a special case when $\alpha = 1$.

On the one hand, $L_\alpha(g)$ is now upper bounded by $\log \frac{m}{\alpha}$, which is larger than $\log m$ when $\alpha \in (0, 1)$. Thus, α-CPC has the potential to reduce bias when $\log m$ is much smaller than $I(X; Y)$. On the other hand, when we set a smaller α, the variance of the estimator becomes larger, and the objective $L_\alpha(g)$ becomes more difficult to optimize [HLLT16, HLCT19]. Therefore, selecting an appropriate α to balance the bias-variance trade-off is helpful for optimization of the objective in practice.

However, it is now possible for $L_\alpha(g)$ to be larger than $I(X; Y)$ as the number of classes m grows to infinity, so optimizing $L_\alpha(g)$ does not necessarily recover a lower bound to mutual information. We illustrate this via the following example (more details in Appendix B.2.2).

Example 1. *Let X, Y be two binary r.v.s such that $\Pr(X = 1, Y = 1) = \Pr(X = 0, Y = 0) = 0.5$. Then $I(X; Y) = \log 2 \approx 0.69$. However, when $\alpha = 0.5$ and $n = m = 3$, we can analytically compute $L_\alpha(g) \approx 0.72 \geq I(X; Y)$ for $g(x, y) = 1$ if $x = y$ and near 0 otherwise.*

4.3 Multi-label Contrastive Predictive Coding

While α-CPC could be useful empirically, we lack a principled way of selecting proper values of α as $L_\alpha(g)$ may no longer be a lower bound to mutual information. In the following sections, we propose an approach that allows us to achieve both, *i.e.*, for all α in a certain range (that only depends on n and m), we can achieve an upper bound of $\log \frac{m}{\alpha}$ while ensuring that the objective is still a lower bound on mutual information. This allows us to select different values of α to reflect different preferences over bias and variance, all while keeping the computational cost identical.

We consider solving a "nm-class, n-label" classification problem, where given n positive samples and $n(m-1)$ negative samples $\overline{\boldsymbol{y}_{j,k}} \sim p(\boldsymbol{y})$, we wish to jointly identify the top-n samples that are most likely to be the positive ones. Concretely, this has the following objective function:

$$J(g) := \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} \log \frac{nm \cdot g(\boldsymbol{x}_i, \boldsymbol{y}_i)}{\sum_{j=1}^{n} g(\boldsymbol{x}_j, \boldsymbol{y}_j) + \sum_{j=1}^{n}\sum_{k=1}^{m-1} g(\boldsymbol{x}_j, \overline{\boldsymbol{y}_{j,k}})}\right] \tag{4.3}$$

where the expectation is taken over the n positive samples $(\boldsymbol{x}_i, \boldsymbol{y}_i) \sim p(\boldsymbol{x}, \boldsymbol{y})$ for $i \in [n]$ and the $n(m-1)$ negative samples $\overline{\boldsymbol{y}_{j,k}} \sim p(\boldsymbol{y})$ for $j \in [n], k \in [m-1]$. We call this *multi-label*

contrastive predictive coding (ML-CPC), since the classifier now needs to predict n positive labels from nm options at the same time, instead of 1 positive label from m options as in traditional CPC (performed for n times for a batch size of n).

Distinctions from CPC Despite its similarity compared to CPC (both are based on classification), we note that the multi-label perspective is fundamentally different from the CPC paradigm in three aspects, and cannot be treated as simply increasing the number of negative samples.

1. The ML-CPC objective value depends on the batch size n, whereas the CPC objective does not.
2. In CPC the positive pair and negative pairs share a same element ($\boldsymbol{x}_i$ in equation 2.16 where the positive sample is $(\boldsymbol{x}_i, \boldsymbol{y}_i)$), whereas in ML-CPC the negative pairs no longer have such restrictions; this could be useful for smaller datasets $\mathcal{D}$ when the number of possible negative pairs increases from $O(|\mathcal{D}|)$ to $O(|\mathcal{D}|^2)$.
3. The optimal critic for CPC is $g^\star = c(\boldsymbol{x}) \cdot p(\boldsymbol{x}, \boldsymbol{y})/(p(\boldsymbol{x})p(\boldsymbol{y}))$, where c is any positive function of $\boldsymbol{x}$ [MC18]. In ML-CPC, different $\boldsymbol{x}$ values are tied within the same batch, so the optimal critic for ML-CPC is $g^\star = c \cdot p(\boldsymbol{x}, \boldsymbol{y})/(p(\boldsymbol{x})p(\boldsymbol{y}))$, where c is a positive constant. As a result, ML-CPC reduces the amount of optimal solutions, and forces the similarity of *any* positive pair to be higher than that of *any* negative pair, unlike CPC where the positive pair only needs to have higher similarity than any negative pairs with the same x.

Computational cost of ML-CPC To compute CPC with a batch size of n, one would need nm critic evaluations and compute n sums in the denominator, each over a different set of m evaluations. To compute ML-CPC, one needs nm critic evaluations, and compute 1 sum over all nm evaluations. Therefore, ML-CPC has almost the same computational cost compared to CPC which is $O(mn)$. We perform a similar analysis in Appendix A.2 to show that evaluating the gradients of the objectives also has similar costs, so ML-CPC is computationally as efficient as CPC.

4.3.1 Re-weighted Multi-label Contrastive Predictive Coding

Similar to α-CPC, we can modify the multi-label objective $J(g)$ by re-weighting the critic predictions, which results in the following objective called α-ML-CPC:

$$J_\alpha(g) := \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} \log \frac{nm \cdot g(\boldsymbol{x}_i, \boldsymbol{y}_i)}{\alpha \sum_{j=1}^{n} g(\boldsymbol{x}_j, \boldsymbol{y}_j) + \frac{m-\alpha}{m-1} \sum_{j=1}^{n} \sum_{k=1}^{m-1} g(\boldsymbol{x}_j, \overline{\boldsymbol{y}_{j,k}})}\right] \tag{4.4}$$

For $\alpha \in (0, 1)$, we down-weight the positive critic outputs by α and up-weight the negative critic outputs by $\frac{m-\alpha}{m-1}$ (similar to α-CPC). Setting a smaller α has the potential to reduce bias, since the upper bound of $\log m$ is changed to $\log \frac{m}{\alpha}$, which is larger when $\alpha \in (0, 1)$. In contrast to α-CPC, $J_\alpha(g)$ is now guaranteed to be a lower bound of mutual information for a wide range of α, as we show in the following statements. Similar to the case of CPC, we first show a more general argument, for which the weighted ML-CPC is a special case.

Theorem 6. *For all probability measures P, Q over sample space $\mathcal{X}$ such that $P \ll Q$, the following holds for all functions $r : \mathcal{X} \to \mathbb{R}_+$, integers $n \geq 1, m \geq 2$, and real numbers $\alpha \in [\frac{m}{n(m-1)+1}, 1]$:*

$$D_{\mathrm{KL}}(P\|Q) \geq \mathbb{E}_{\boldsymbol{x}_{1:n}\sim P^n, \boldsymbol{y}_{i,1:m-1}\sim Q^{m-1}} \left[\frac{1}{n} \sum_{i=1}^{n} \log \frac{mn \cdot r(\boldsymbol{x}_i)}{\alpha \sum_{j=1}^{n} r(\boldsymbol{x}_j) + \frac{m-\alpha}{m-1} \sum_{k=1}^{m-1} r(\boldsymbol{y}_{j,k})} \right]. \tag{4.5}$$

Proof. In Appendix A.2, using the variational representations of f-divergences [NWJ08] and Prop. 7. □

The above theorem extends existing variational lower bound estimators of KL divergences (that are generally interpreted as binary classification [SSK12, NWJ08]) into a family of lower bounds that can be interpreted as multi-label classification. The argument about re-weighted ML-CPC being a lower bound to MI is simply a corollary where P is $p(\boldsymbol{x}, \boldsymbol{y})$ and Q is $p(\boldsymbol{x})p(\boldsymbol{y})$; we state the claim below.

Corollary 3. *$\forall n \geq 1, m \geq 2$, define $\alpha_{m,n} = \frac{m}{n(m-1)+1}$. If $\alpha \in [\alpha_{m,n}, 1]$, then $\forall g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+$,*

$$J_\alpha(g) \leq I(X; Y) \tag{4.6}$$

The above theorem shows that for an appropriate range of α values, the objective $J_\alpha(g)$ is still guaranteed to be a variational lower bound to mutual information, like the original CPC objective. Selecting α within this range results in estimators with different bias-variance trade-offs. Here, a smaller α could lead to low-bias high-variance estimates; this achieves a similar effect to increasing the number of classes m to nearly m/α, but without the actual additional computational costs that comes with obtaining more negative samples in CPC.

Illustrative example We consider the case of X, Y being binary and equal random variables in Example 1, where $I(X; Y) = \log 2 \approx 0.69$, the optimal critic g is known, and both $L_\alpha(g)$ and

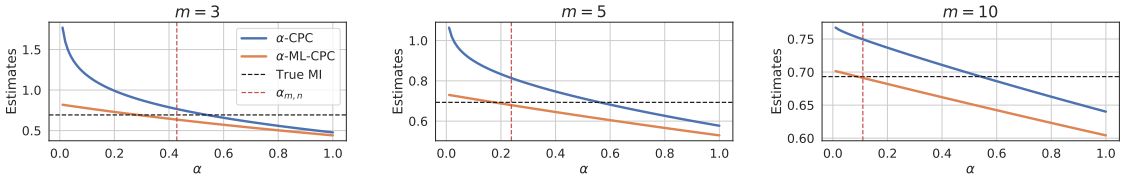


Figure 4.1: MI estimates with CPC and ML-CPC under different α.

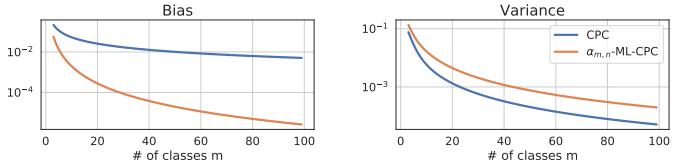


Figure 4.2: **Bias-variance trade-offs for different m.** Lower is better.

$J_\alpha(g)$ can be computed in closed-form for any α and g in $O(m)$ time (details in Appendix B.2.2). We plot the CPC (equation 4.2) and ML-CPC (equation 4.4) objectives with different choices of α and m in Figure 4.1. The estimates of ML-CPC when $\alpha \geq \alpha_{m,n}$ are lower bounds to the ground truth MI, which indeed aligns with our theory.

Furthermore, in Figure 4.2 we illustrate the bias-variance trade-offs for CPC and $\alpha_{m,n}$-ML-CPC as we vary the number of classes m (for simplicity, we choose $n = m$). Despite having slightly higher variance in the estimates, $\alpha_{m,n}$-ML-CPC has significantly less bias than CPC, which suggests that it is helpful in cases where lower bias is preferable than lower variance. In practice, the user could select different values of α to indicate the desired trade-off, all without having to change the number of negative samples and increase computational costs.

We include the pseudo-code and a PyTorch implementation to α-ML-CPC in Appendix B.2.1.

4.4 Related Work

Contrastive methods for representation learning The general principle of contrastive methods for representation learning encourages representations to be closer between "positive" pairs and further between "negative" pairs, which has been applied to learning representations in various domains such as images [HRD+19, WXYL18, HFW+19, CKNH20], words [MSC+13, DCLT18], graphs [VFH+18] and videos [HXZ19]. Commonly used objectives include the logistic loss [MSC+13], margin triplet loss [SKP15], the noise contrastive estimation loss [GH12] and other objectives based

on variational lower bounds of mutual information, such as MINE [BBR+18] and CPC [vdOLV18]. CPC-based approaches have gained interest due to its superior performance in downstream tasks compared to other losses such as the logistic and margin loss [CKNH20].

Variational mutual information estimators Estimating mutual information from samples is challenging [MS20, XZS+20]. Most variational approaches to mutual information estimation are based on the Fenchel dual representation of f-divergences [NWJ08, SE19a], where a critic function is trained to learn the density ratio $p(\boldsymbol{x}, \boldsymbol{y})/(p(\boldsymbol{x})p(\boldsymbol{y}))$. These approaches mostly vary in terms of how the critics are modeled and optimized [BA03, POvdO+19], and exhibit different bias-variance trade-offs from these choices.

CPC would tend to underestimate the density ratio (since it is capped at m) and generally requires $O(e^{I(X;Y)})$ samples to achieve low bias; MINE [BBR+18] (based on the Donsker-Varadhan inequality [DV75]) is a biased estimator and requires $O(e^{I(X;Y)})$ samples to achieve low variance [SE19b, SE19a]. Poole et al. [POvdO+19] proposed an estimator that interpolates between two types of estimators, allowing for certain bias-variance trade-offs; this is relevant to our proposed re-weighted CPC in the sense that positive samples are down-weighted, but an additional baseline model is required during training. Through ML-CPC, we introduce a family of unbiased mutual information lower bound estimators, and reflect a wide range of bias-variance trade-offs without using more negative samples.

Relevance to the limitations of mutual information lower bound estimators Furthermore, we note that ML-CPC is upper bounded by $\log(n(m-1)+1)$ for the smallest possible α, which appears to be very close to (but smaller than) the general upper limit of $O(\log nm)$ that can be achieved by any distribution-free high-confidence lower bound (namely, estimates from samples are lower bounds to the true mutual infomation with high probability) on mutual information for nm samples [MS20]. However, we note that the assumptions in [MS20] are slightly different to our settings, in the sense that they assumed complete access to the distribution $p(\boldsymbol{x}, \boldsymbol{y})$ and only required samples from $p(\boldsymbol{x})p(\boldsymbol{y})$, whereas we have to estimate $p(\boldsymbol{x}, \boldsymbol{y})$ from the samples as well; and the amount of samples we obtain from $p(\boldsymbol{x})p(\boldsymbol{y})$ is $n(m-1)$ instead of nm. We hypothesize that we can reach the theoretical limit with a method derived from ML-CPC, but leave it as an interesting future direction.

Re-weighted softmax loss Generalizations to the softmax loss have been proposed in which different weights are assigned to different classes or samples [LWYY16, LLW17, WLLC18], which

are commonly used with regularization [BL18]. When the dataset has extremely imbalanced classes, higher weights are given to classes with less frequency [HLLT16, HLCT19, WRH17] or classes with less effective samples [CJL+19]. Cao et al. [CWG+19] investigate re-weighting approaches that encourages large margins to the decision boundary for minority classes; such a context is also studied for detection [LKG19] and segmentation [KHZ+19] where class imbalance exists. Our work introduce re-weighting approaches to the context of unsupervised representation learning (where class labels do not exist in the traditional sense), where we aim for flexible bias-variance trade-offs in contrastive mutual information estimators.

4.5 Experiments

We evaluate our proposed methods on mutual information estimation, knowledge distillation and unsupervised representation learning. *To ensure fair comparisons are made, we only make adjustments to the training objective, and keep the remaining experimental setup identical to that of the baselines.* We describe details to the experimental setup in Appendix B.2.2. Our code is available at https://github.com/jiamings/ml-cpc.

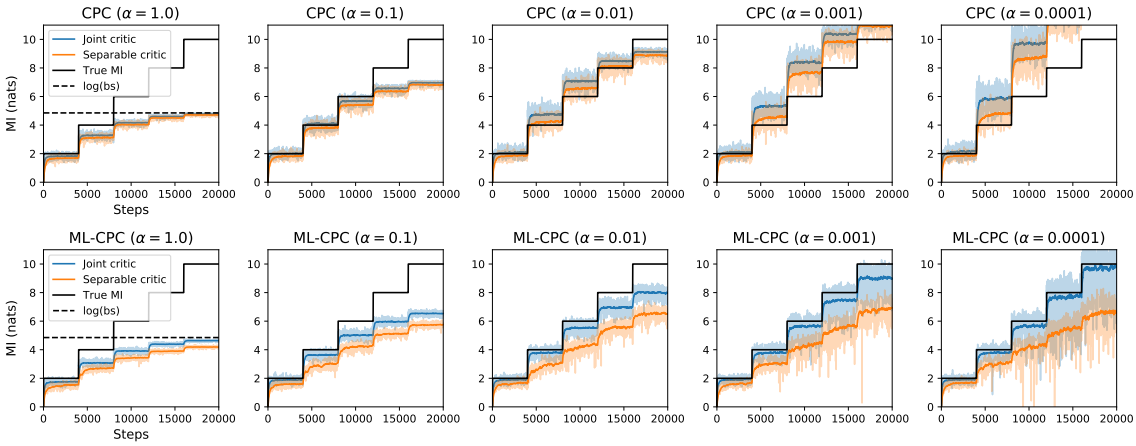


Figure 4.3: Mutual information estimation with CPC and ML-CPC, with different choices of α.

4.5.1 Mutual Information Estimation

Setup We first consider mutual information estimation between two correlated Gaussians of 20 dimensions, following the setup in [POvdO+19, SE19b] where the ground truth mutual information is known and increases by 2 every 4k iterations, for a total of 20k iterations. We evaluate CPC and

ML-CPC with different choices of α (ranging from 1.0 to 0.0001, which might not guarantee that they are lower bounds to mutual information) under two types of critic, named joint [BBR+18] and separable [vdOLV18]. We use $m = n = 128$ in our experiments.

Results We illustrate the estimates and the ground truth MI in Figure 4.3. Both CPC and ML-CPC estimates are bounded by $\log m$ when $\alpha = 1$, which is no longer the case when we set smaller values of α; however, as we decrease α, CPC estimates are no longer guaranteed to be lower bounds to mutual information, whereas ML-CPC estimates still provide lower bound estimates in general. Moreover, a reduction in α for ML-CPC reduces bias at the cost of increasing variance, as the problem becomes more difficult with re-weighting. The time to compute 200 updates on a Nvidia 1080 Ti GPU with the a PyTorch implementation is 1.15 ± 0.06 seconds with CPC and 1.14 ± 0.04 seconds with ML-CPC, so the computational costs are indeed near identical.

4.5.2 Knowledge Distillation

Setup We apply re-weighted CPC and ML-CPC to knowledge distillation (KD, [HVD15]), in which one neural network model (teacher) transfers its knowledge to another model (student, typically smaller) so that the student's performance is higher than training from labels alone. Contrastive representation distillation (CRD, [TKI19]) is a state-of-the-art method that regularizes the student model so that its features have higher mutual information with that of the teacher; CRD is implemented via a type of noise contrastive estimation objective [GH12]. We replace this objective with CPC and ML-CPC, using different choices of α that are fixed throughout training, and *keeping the remaining hyperparameters identical to the CRD ones* in [TKI19]. Two baselines are considered: the original KD objective in [HVD15] and the state-of-the-art CRD objective in [TKI19], since other baselines [KKBZ19, AHD+19, HW17, KPK18] are shown to have inferior performance in general.

Results Following the procedure in [TKI19], we evaluate over 13 different student-teacher pairs on CIFAR-100 [KH+09]. The student and teacher have the same type of architecture in 7 cases and different types in 6 cases. We report top-1 test accuracy in Table 4.1 (same type) and Table 4.2 (different types), where each case is the mean evaluation from 3 random seeds. We omit the standard deviation across different random seeds of each setup to fit the table in the paper, but we note that deviation among different random seeds is fairly small (at around 0.05 to 0.1 for most cases). While CPC and ML-CPC are generally inferior to that of CRD when $\alpha = 1.0$ (this aligns with the

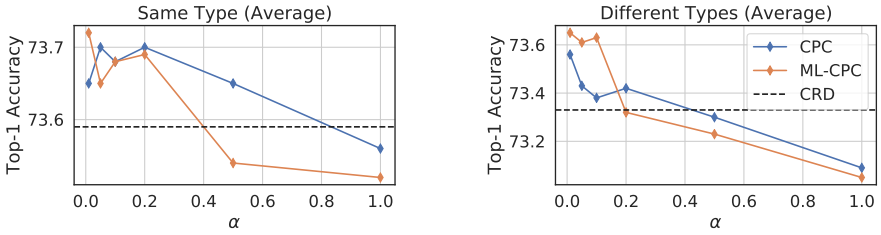


Figure 4.4: **Ablation studies for knowledge distillation** with CPC and ML-CPC at different values of α. Left: student and teacher are of the same type. Right: student and teacher are from different types.

observation in [TKI19]), they outperform CRD in 10 out of 13 cases when a smaller α is selected.

To demonstrate the effect of improved performance of smaller α, we evaluate average top-1 accuracies with $\alpha \in \{0.01, 0.05, 0.1, 0.2, 0.5, 1.0\}$ in Figure 4.4. Both CPC and ML-CPC are generally inferior to CRD when $\alpha = 1.0$ or 0.5, but as we select smaller values of α, they become superior to CRD and reaches the highest values at around 0.01 to 0.05, with ML-CPC being slightly better. Moreover, $n = 64, m = 16384$ so $\alpha_{m,n} \approx 0.015$, which achieves the lowest bias while ensuring ML-CPC to be a lower bound to MI. Thus this observation aligns with our claims on $\alpha_{m,n}$ in Theorem 3.

Table 4.1: Top-1 *test accuracy* (%) of students networks on CIFAR100 where the student and teacher networks are of the same type. (↑) and (↓) denotes superior and inferior performance relative to CRD. Each result is the mean of 3 random runs. L_α and J_α denote α-CPC and α-ML-CPC.

Teacher Student	WRN-40-2 WRN-16-2	WRN-40-2 WRN-40-1	resnet56 resnet20	resnet110 resnet20	resnet110 resnet32	resnet32x4 resnet8x4	vgg13 vgg8
Teacher	75.61	75.61	72.34	74.31	74.31	79.42	74.64
Student	73.26	71.98	69.06	69.06	71.14	72.50	70.36
KD	74.92	73.54	70.66	70.67	73.08	73.33	72.98
CRD	75.48	74.14	71.16	**71.46**	73.48	**75.51**	73.94
$L_{1.0}$	75.42 (↓)	74.16 (↑)	71.32 (↑)	71.39 (↓)	73.57 (↑)	75.50 (↓)	73.60 (↓)
$L_{0.1}$	75.69 (↑)	74.17 (↑)	71.48 (↑)	71.38 (↓)	73.66 (↑)	75.41 (↓)	73.61 (↓)
$J_{1.0}$	75.39 (↓)	74.18 (↑)	71.28 (↑)	71.28 (↓)	73.58 (↑)	75.32 (↓)	73.67 (↓)
$J_{0.05}$	75.64 (↑)	**74.27** (↑)	71.33 (↑)	71.24 (↓)	73.57 (↑)	75.50 (↓)	**74.01** (↑)
$J_{0.01}$	**75.83** (↑)	74.24 (↑)	**71.50** (↑)	71.27 (↓)	**73.90** (↑)	75.37 (↓)	73.95 (↑)

Table 4.2: Top-1 *test accuracy* (%) of students networks on CIFAR100 where the student and teacher networks are from different types. (↑) and (↓) denotes superior and inferior performance relative to CRD. Each result is the mean of 3 random runs. L_α and J_α denote α-CPC and α-ML-CPC.

Teacher Student	vgg13 MobileNetV2	ResNet50 MobileNetV2	ResNet50 vgg8	resnet32x4 ShuffleNetV1	resnet32x4 ShuffleNetV2	WRN-40-2 ShuffleNetV1
Teacher	74.64	79.34	79.34	79.42	79.42	75.61
Student	64.60	64.60	70.36	70.50	71.82	70.50
KD	67.37	67.35	73.81	74.07	74.45	74.83
CRD	**69.73**	69.11	74.30	75.11	75.65	76.05
$L_{1.0}$	69.24 (↓)	69.02 (↓)	73.66 (↓)	75.00 (↓)	75.93 (↑)	75.72 (↓)
$L_{0.1}$	69.26 (↓)	69.33 (↑)	74.24 (↓)	75.34 (↑)	76.01 (↑)	76.12 (↑)
$J_{1.0}$	68.92 (↓)	68.80 (↓)	73.65 (↓)	75.39 (↑)	75.88 (↑)	75.70 (↓)
$J_{0.05}$	69.25 (↓)	**70.04** (↑)	**74.84** (↑)	75.51 (↑)	76.24 (↑)	76.03 (↑)
$J_{0.01}$	69.25 (↓)	69.90 (↑)	74.81 (↑)	**75.47** (↑)	**76.04** (↑)	**76.19** (↑)

4.5.3 Representation Learning

Setup Finally, we consider ML-CPC for unsupervised representation learning as a replacement to CPC. We follow the experiment procedures in MoCo-v2 [CFGH20] (which used the CPC objective), where negative samples are obtained from a key encoder that updates more slowly than the representation network. We use the "linear evaluation protocol" where the learned representations are evaluated via the test top-1 accuracy when a linear classifier is trained to predict labels from representations. Different from knowledge distillation, we do not have labels and fixed teacher representations, so the problem becomes much more difficult and using small values of α alone will lead to high variance in initial estimates which could hinder the final performance. To this end, we use a curriculum learning [BLCW09] approach where we select α values from high to low throughout training: higher α has higher bias, lower variance and easier to learn, whereas lower α has lower bias, higher variance and harder to learn. For ML-CPC, we consider 4 types of geometrically decreasing schedules for α: fixed at 1.0; from 2.0 to 0.5; from 5.0 to 2.0; and from 10.0 to 0.1; so $\alpha = 1.0$ for all cases when we reached half of the training epochs. We use the same values for other hyperparameters as those used in the MoCo-v2 CPC baseline (more details in Appendix B.2.2).

Results We show the top-1 accuracy of the learned representations under the linear evaluation protocol in Table 4.3 for CIFAR10 and CIFAR100. While the original ML-CPC objective (denoted

as $J_{1.0} \rightarrow J_{1.0}$) already outperforms the CPC baseline in most cases, we observe that using a curriculum from easy to hard objective has the potential to further improve performance of the representations. Notably, the $J_{10.0} \rightarrow J_{0.1}$ schedule improves the performance on both datasets by almost 2.5 percent when trained for 200 epochs, which demonstrates its effectiveness when the number of epochs used during training is limited.

Table 4.3: **Top-1 accuracy of unsupervised representation learning**.

(a) CIFAR-10

Epochs	200	500	1000
$L_{1.0}$	83.28	89.31	91.20
$J_{1.0} \rightarrow J_{1.0}$	83.61 (↑)	89.43 (↑)	91.48 (↑)
$J_{2.0} \rightarrow J_{0.5}$	84.31 (↑)	89.47 (↑)	91.43 (↑)
$J_{5.0} \rightarrow J_{0.2}$	85.52 (↑)	**89.85** (↑)	91.50 (↑)
$J_{10.0} \rightarrow J_{0.1}$	**86.16** (↑)	89.49 (↑)	**91.86** (↑)

(b) CIFAR-100

Epochs	200	500	1000
$L_{1.0}$	61.42	67.72	69.63
$J_{1.0} \rightarrow J_{1.0}$	61.80 (↑)	67.68 (↓)	**70.85** (↑)
$J_{2.0} \rightarrow J_{0.5}$	62.92 (↑)	68.01 (↑)	70.22 (↑)
$J_{5.0} \rightarrow J_{0.2}$	63.58 (↑)	**68.04** (↑)	70.07 (↑)
$J_{10.0} \rightarrow J_{0.1}$	**64.05** (↑)	67.94 (↑)	70.03 (↑)

In Table 4.4, we include additional results for ImageNet under a compute-constrained scenario, where the representations are trained for only 30 epochs on a ResNet-18 architecture. Similar to the observations in CIFAR-10, we observe improvements in terms of linear classification accuracy of the learned representations. This demonstrates that the curriculum learning approach (specific to ML-CPC with re-weighting schedules, where the objective remains a lower bound to mutual information) could be useful to unsupervised representation learning in general.

4.6 Discussion

In this chapter, we proposed multi-label contrastive predictive coding for representation learning, which provides a generalization to contrastive predictive coding via multi-label classification. Re-weighted ML-CPC is able to enjoy less bias while being a lower bound to mutual information. Our upper bounds for the smallest α is close to the theoretical limit [MS20] of any distribution-free high-confidence lower bound on mutual information estimation. We demonstrate the effectiveness of

Table 4.4: ImageNet representation learning for 30 epochs.

Objective	$L_{1.0}$	$J_{1.0} \rightarrow J_{1.0}$	$J_{2.0} \rightarrow J_{0.5}$	$J_{5.0} \rightarrow J_{0.2}$	$J_{10.0} \rightarrow J_{0.1}$
Top1	43.45	43.24 (↓)	43.52 (↑)	**43.86** (↑)	43.81 (↑)
Top5	67.42	67.43 (↑)	**67.82** (↑)	67.67 (↑)	67.71 (↑)

ML-CPC on mutual information, knowledge distillation and unsupervised representation learning.

It would be interesting to further apply this method to other application domains, investigate alternative methods to control the re-weighting procedure (such as using angular margins [LWYY16]), and develop more principled approaches towards curriculum learning for unsupervised representation learning. From a theoretical standpoint, it is also interesting to formally investigate the bias-variance trade-off of ML-CPC, and see whether simple modifications to ML-CPC based on a slightly different assumption over $p(\boldsymbol{x}, \boldsymbol{y})$ could approach the theoretical limit by McAllester and Stratos [MS20].

Chapter 5

Fair Representation Learning via Regression

In the previous chapter, our goal was to learn compressed representations of the data that are as informative as possible. However, in many real-world applications, raw data often contain certain sensitive information, such as age and gender. If we wish to protect the user from data misuse, then removing the sensitive information alone would not be enough: one can still infer the sensitive attributes from other features (for example, one's favorite activity could be correlated to their gender). This motivates us to produce not only informative representations, but also "fair" ones that reduces the undesirable information about sensitive information.

In this chapter, we propose an information-theoretically motivated objective for learning maximally expressive fair representations. We unify a range of existing approaches by showing they optimize approximations to the Lagrangian dual of our objective. These existing approaches may learn somewhat expressive and fair representations, but users wanting to control the fairness of representations must tune the expressiveness-fairness trade-off over many runs and will never know whether they have obtained the maximally expressive representations under certain fairness constraints. We are the first to provide user control over the fairness of representations through a user-specifiable limit on unfairness. Through a dual optimization method (optimizing the model as well as the expressiveness-fairness trade-off), we obtain representations achieving high expressiveness while satisfying the user-specified limits on unfairness.

This chapter was previous published as [SKG+19]. Pratyusha Kalluri, Aditya Grover and Shengjia Zhao contributed to the contents of this chapter. My contributions are conceiving the idea, implementing the algorithm, executing the experiments and writing the chapter. Other coauthors

helped with discussions and writing.

5.1 Introduction

Statistical learning systems are increasingly being used to assess individuals, influencing consequential decisions such as bank loans, college admissions, and criminal sentences. This yields a growing demand for systems guaranteed to output decisions that are fair with respect to sensitive attributes such as gender, race, and disability.

In the typical classification and regression settings with fairness and privacy constraints, one is concerned about performing a single, specific task. However, situations arise where a data owner needs to release data to downstream users without prior knowledge of the tasks that will be performed [MCPZ18]. In such cases, it is crucial to find representations of the data that can be used on a wide variety of tasks while preserving fairness [CWRV17].

This gives rise to two desiderata. On the one hand, the representations need to be *expressive*, so that they can be used effectively for as many tasks as possible. On the other hand, the representations also need to satisfy certain *fairness* constraints to protect sensitive attributes. Further, many notions of fairness are possible, and it may not be possible to simultaneously satisfy all of them [KMR16, Cho17]. Therefore, the ability to effectively trade off multiple notions of fairness is crucial to fair representation learning.

To this end, we present an information theoretically motivated constrained optimization framework (Section 5.2). The goal is to maximize the expressiveness of representations while satisfying certain fairness constraints. We represent expressiveness as well as three dominant notions of fairness (demographic parity [ZWS+13], equalized odds, equalized opportunity [HPS16]) in terms of mutual information, obtain tractable upper/lower bounds of these mutual information objectives, and connect them with existing objectives such as maximum likelihood, adversarial training [GPAM+14], and variational autoencoders [KW13, RM15].

As we demonstrate in Section 5.3, this serves as a unifying framework for existing work [ZWS+13, LSL+15, ES15, MCPZ18] on learning fair representations. A range of existing approaches to learning fair representations, which do not draw connections to information theory, optimize an approximation of the Lagrangian dual of our objective with fixed values of the Lagrange multipliers. These thus require the user to obtain different representations for different notions of fairness as in [MCPZ18].

Instead, we consider a dual optimization approach (Section 5.4), in which we optimize the model

as well as the Lagrange multipliers during training [ZSE18b], thereby also learning the trade-off between expressiveness and fairness. We further show that our proposed framework is strongly convex in distribution space.

Our work is the first to provide direct user control over the fairness of representations through fairness constraints that are interpretable by non-expert users. Empirical results in Section 5.5 demonstrate that our notions of expressiveness and fairness based on mutual information align well with existing definitions, our method encourages representations that satisfy the fairness constraints while being more expressive, and that our method is able to balance the trade-off between multiple notions of fairness with a single representation and a significantly lower computational cost.

5.2 An Information-Theoretic Objective for Controllable Fair Representations

We are given a dataset $\mathcal{D}_u = \{(\mathbf{x}_i, \mathbf{u}_i)\}_{i=1}^M$ containing pairs of observations $\mathbf{x} \in \mathcal{X}$ and sensitive attributes $\mathbf{u} \in \mathcal{U}$. We assume the dataset is sampled i.i.d. from an unknown data distribution $q(\mathbf{x}, \mathbf{u})$. Our goal is to transform each data point $(\mathbf{x}, \mathbf{u})$ into a new representation $\mathbf{z} \in \mathcal{Z}$ that is (1) *transferable*, i.e., it can be used in place of $(\mathbf{x}, \mathbf{u})$ by multiple unknown vendors on a variety of downstream tasks, and (2) *fair*, i.e., the sensitive attributes $\mathbf{u}$ are protected. For conciseness, we focus on the *demographic parity* notion of fairness [CKP09, Zli15, ZVRG15], which requires the decisions made by a classifier over $\mathbf{z}$ to be independent of the sensitive attributes $\mathbf{u}$. We discuss in Appendix B.3.3 how our approach can be extended to control other notions of fairness simultaneously, such as the *equalized odds* and *equalized opportunity* notions of fairness [HPS16].

We assume the representations $\mathbf{z} \in \mathcal{Z}$ of $(\mathbf{x}, \mathbf{u})$ are obtained by sampling from a conditional probability distribution $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{u})$ parameterized by $\phi \in \Phi$. The joint distribution of $(\mathbf{x}, \mathbf{z}, \mathbf{u})$ is then given by $q_\phi(\mathbf{x}, \mathbf{z}, \mathbf{u}) = q(\mathbf{x}, \mathbf{u})q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{u})$. We formally express our desiderata for learning a controllable fair representation $\mathbf{z}$ through the concept of mutual information:

1. **Fairness**: $\mathbf{z}$ should have low mutual information with the sensitive attributes $\mathbf{u}$.
2. **Expressiveness**: $\mathbf{z}$ should have high mutual information with the observations $\mathbf{x}$, conditioned on $\mathbf{u}$ (in expectation over possible values of $\mathbf{u}$).

The first condition encourages $\mathbf{z}$ to be independent of $\mathbf{u}$; if this is indeed the case, the downstream vendor cannot learn a classifier over the representations $\mathbf{z}$ that discriminates based on $\mathbf{u}$. Intuitively, the mutual information $I_q(\mathbf{z}, \mathbf{u})$ is related to the optimal predictor of $\mathbf{u}$ given $\mathbf{z}$. If $I_q(\mathbf{z}, \mathbf{u})$ is zero,

then no such predictor can perform better than chance; if $I_q(\mathbf{z}, \mathbf{u})$ is large, vendors in downstream tasks could utilize $\mathbf{z}$ to predict the sensitive attributes $\mathbf{u}$ and make unfair decisions.

The second condition encourages $\mathbf{z}$ to contain as much information as possible from $\mathbf{x}$ conditioned on the knowledge of $\mathbf{u}$. By conditioning on $\mathbf{u}$, we ensure we do not encourage information in $\mathbf{x}$ that is correlated with $\mathbf{u}$ to leak into $\mathbf{z}$. The two desiderata allow $\mathbf{z}$ to encode non-sensitive information from $\mathbf{x}$ (expressiveness) while excluding information in $\mathbf{u}$ (fairness).

Our goal is to choose parameters $\phi \in \Phi$ for $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{u})$ that meet both these criteria[1]. Because we wish to ensure our representations satisfy fairness constraints even at the cost of using less expressive $\mathbf{z}$, we synthesize the two desiderata into the following constrained optimization problem:

$$\max_{\phi \in \Phi} I_q(\mathbf{x}; \mathbf{z}|\mathbf{u}) \qquad \text{s.t. } I_q(\mathbf{z}; \mathbf{u}) < \epsilon \tag{5.1}$$

where $I_q(\mathbf{x}; \mathbf{z}|\mathbf{u})$ denotes the mutual information of $\mathbf{x}$ and $\mathbf{z}$ conditioned on $\mathbf{u}$, $I_q(\mathbf{z}; \mathbf{u})$ denotes mutual information between $\mathbf{z}$ and $\mathbf{u}$, and the hyperparameter $\epsilon > 0$ controls the maximum amount of mutual information allowed between $\mathbf{z}$ and $\mathbf{u}$. The motivation of our "hard" constraint on $I_q(\mathbf{z}; \mathbf{u})$ – as opposed to a "soft" regularization term – is that even at the cost of learning less expressive $\mathbf{z}$ and losing some predictive power, we view as important ensuring that our representations are fair to the extent dictated by ϵ.

Both mutual information terms in Equation 5.1 are difficult to compute and optimize. In particular, the optimization objective in Equation 5.1 can be expressed as the following expectation:

$$I_q(\mathbf{x}; \mathbf{z}|\mathbf{u}) = \mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z},\mathbf{u})}[\log q_\phi(\mathbf{x}, \mathbf{z}|\mathbf{u}) - \log q(\mathbf{x}|\mathbf{u}) - \log q_\phi(\mathbf{z}|\mathbf{u})]$$

while the constraint on $I_q(\mathbf{z}; \mathbf{u})$ involves the following expectation:

$$I_q(\mathbf{z}; \mathbf{u}) = \mathbb{E}_{q_\phi(\mathbf{z},\mathbf{u})}[\log q_\phi(\mathbf{z}|\mathbf{u}) - \log q_\phi(\mathbf{z})]$$

Even though $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{u})$ is known analytically and assumed to be easy to evaluate, both mutual information terms are difficult to estimate and optimize.

To offset the challenge in estimating mutual information, we introduce upper and lower bounds with tractable Monte Carlo gradient estimates. We introduce the following lemmas, with the proofs provided in Appendix A.3. We note that similar bounds have been proposed in [BA03, AFDM16, ZSE17a, ZSE18b, GE19].

[1] Simply ignoring $\mathbf{u}$ as an input is insufficient, as $\mathbf{x}$ may still contain information about $\mathbf{u}$.

5.2.1 Tractable Lower Bound for $I_q(\mathbf{x}; \mathbf{z}|\mathbf{u})$

We begin with a (variational) *lower* bound on the objective function $I_q(\mathbf{x}; \mathbf{z}|\mathbf{u})$ related to expressiveness which we would like to *maximize* in Equation 5.1.

Lemma 6. *For any conditional distribution $p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{u})$ (parametrized by θ)*

$$I_q(\mathbf{x}; \mathbf{z}|\mathbf{u}) = \mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z},\mathbf{u})}[\log p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{u})] + H_q(\mathbf{x}|\mathbf{u}) + \mathbb{E}_{q_\phi(\mathbf{z},\mathbf{u})} D_{\mathrm{KL}}(q_\phi(\mathbf{x}|\mathbf{z}, \mathbf{u}) \| p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{u}))$$

where $H_q(\mathbf{x}|\mathbf{u})$ is the entropy of $\mathbf{x}$ conditioned on $\mathbf{u}$, and D_{KL} denotes KL-divergence.

Since entropy and KL divergence are non-negative, the above lemma implies the following lower bound:

$$I_q(\mathbf{x}; \mathbf{z}|\mathbf{u}) \geq \mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z},\mathbf{u})}[\log p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{u})] := \mathcal{L}_r. \tag{5.2}$$

5.2.2 Tractable Upper Bound for $I_q(\mathbf{z}; \mathbf{u})$

Next, we provide an *upper* bound for the constraint term $I_q(\mathbf{z}; \mathbf{u})$ that specifies the limit on unfairness. In order to satisfy this fairness constraint, we wish to implicitly *minimize* this term.

Lemma 7. *For any distribution $p(\mathbf{z})$, we have:*

$$I_q(\mathbf{z}; \mathbf{u}) \leq I_q(\mathbf{z}; \mathbf{x}, \mathbf{u}) = \mathbb{E}_{q(\mathbf{x},\mathbf{u})} D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{u}) \| p(\mathbf{z})) - D_{\mathrm{KL}}(q_\phi(\mathbf{z}) \| p(\mathbf{z})). \tag{5.3}$$

Again, using the non-negativity of KL divergence, we obtain the following upper bound:

$$I_q(\mathbf{z}; \mathbf{u}) \leq \mathbb{E}_{q(\mathbf{x},\mathbf{u})} D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{u}) \| p(\mathbf{z})) := C_1. \tag{5.4}$$

In summary, Equation 5.2 and Equation 5.4 imply that we can compute tractable Monte Carlo estimates for the lower and upper bounds to $I_q(\mathbf{x}; \mathbf{z}|\mathbf{u})$ and $I_q(\mathbf{z}; \mathbf{u})$ respectively, as long as the variational distributions $p(\mathbf{x}|\mathbf{z}, \mathbf{u})$ and $p(\mathbf{z})$ can be evaluated tractably, e.g., Bernoulli and Gaussian distributions. Note that the distribution $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{u})$ is assumed to be tractable.

5.2.3 A Tighter Upper Bound to $I_q(\mathbf{z}, \mathbf{u})$ via Adversarial Training

It would be tempting to use C_1, the tractable upper bound from Equation 5.4, as a replacement for $I_q(\mathbf{z}, \mathbf{u})$ in the constraint of Equation 5.1. However, note from Equation 5.3 that C_1 is *also* an upper

bound to $I_q(\mathbf{x}, \mathbf{z}|\mathbf{u})$, which is the objective function (expressiveness) we would like to maximize in Equation 5.1. If this was constrained too tightly, we would constrain the expressiveness of our learned representations. Therefore, we introduce a tighter bound via the following lemma.

Lemma 8. *For any distribution $p(\mathbf{u})$, we have:*

$$I_q(\mathbf{z}; \mathbf{u}) = \mathbb{E}_{q_\phi(\mathbf{z})} D_{\mathrm{KL}}(q_\phi(\mathbf{u}|\mathbf{z}) \| p(\mathbf{u})) - D_{\mathrm{KL}}(q(\mathbf{u}) \| p(\mathbf{u})). \tag{5.5}$$

Using the non-negativity of KL divergence as before, we obtain the following upper bound on $I_q(\mathbf{z}; \mathbf{u})$:

$$I_q(\mathbf{z}; \mathbf{u}) \leq \mathbb{E}_{q_\phi(\mathbf{z})} D_{\mathrm{KL}}(q_\phi(\mathbf{u}|\mathbf{z}) \| p(\mathbf{u})) := \hat{C}_2. \tag{5.6}$$

As $\mathbf{u}$ is typically low-dimensional (e.g., a binary variable, as in [HPS16, ZWS+13]), we can choose $p(\mathbf{u})$ in Equation 5.5 to be a kernel density estimate based on the dataset $\mathcal{D}$. By making $D_{\mathrm{KL}}(q(\mathbf{u}) \| p(\mathbf{u}))$ as small as possible, our upper bound $\hat{C}_2$ gets closer to $I_q(\mathbf{z}, \mathbf{u})$.

While $\hat{C}_2$ is a valid upper bound to $I_q(\mathbf{z}; \mathbf{u})$, the term $q_\phi(\mathbf{u}|\mathbf{z})$ appearing in $\hat{C}_2$ is intractable to evaluate, requiring an integration over $\mathbf{x}$. Our solution is to approximate $q_\phi(\mathbf{u}|\mathbf{z})$ with a parametrized model $p_\psi(\mathbf{u}|\mathbf{z})$ with parameters $\psi \in \Psi$ obtained via the following objective:

$$\min_\psi \mathbb{E}_{q_\phi(\mathbf{z})} D_{\mathrm{KL}}(q_\phi(\mathbf{u}|\mathbf{z}) \| p_\psi(\mathbf{u}|\mathbf{z})). \tag{5.7}$$

Note that the above objective corresponds to maximum likelihood prediction with inputs $\mathbf{z}$ and labels $\mathbf{u}$ using $p_\psi(\mathbf{u}|\mathbf{z})$. In contrast to $q_\phi(\mathbf{u}|\mathbf{z})$, the distribution $p_\psi(\mathbf{u}|\mathbf{z})$ is tractable and implies the following lower bound to $\hat{C}_2$:

$$\begin{aligned} &\mathbb{E}_{q_\phi(\mathbf{z},\mathbf{u})}[\log p_\psi(\mathbf{u}|\mathbf{z}) - \log p(\mathbf{u})] = \mathbb{E}_{q_\phi(\mathbf{z})}[D_{\mathrm{KL}}(q_\phi(\mathbf{u}|\mathbf{z}) \| p(\mathbf{u})) - D_{\mathrm{KL}}(q_\phi(\mathbf{u}|\mathbf{z}) \| p_\psi(\mathbf{u}|\mathbf{z}))] \\ &\leq \mathbb{E}_{q_\phi(\mathbf{z})} D_{\mathrm{KL}}(q_\phi(\mathbf{u}|\mathbf{z}) \| p(\mathbf{u})) = \hat{C}_2. \end{aligned}$$

It follows that we can approximate $I_q(\mathbf{z}; \mathbf{u})$ through the following adversarial training objective:

$$\min_\phi \max_\psi \mathbb{E}_{q_\phi(\mathbf{z},\mathbf{u})}[\log p_\psi(\mathbf{u}|\mathbf{z}) - \log p(\mathbf{u})] \tag{5.8}$$

Here, the goal of the adversary p_ψ is to minimize the difference between the tractable approximation given by $\mathbb{E}_{q_\phi(\mathbf{z},\mathbf{u})}[\log p_\psi(\mathbf{u}|\mathbf{z}) - \log p(\mathbf{u})]$ and the intractable true upper bound $\hat{C}_2$. We summarize this observation in the following result:

Corollary 4. *If* $D_{\mathrm{KL}}(q_\phi(\mathbf{u}|\mathbf{z})\|p_\psi(\mathbf{u}|\mathbf{z})) \leq \ell$, *then*

$$I_q(\mathbf{z};\mathbf{u}) \leq \mathbb{E}_{q_\phi(\mathbf{z},\mathbf{u})}[\log p_\psi(\mathbf{u}|\mathbf{z}) - \log p(\mathbf{u})] - D_{\mathrm{KL}}(q(\mathbf{u})\|p(\mathbf{u})) + \ell$$

for any distribution $p(\mathbf{u})$.

It immediately follows that when $\ell \to 0$, i.e., the adversary approaches global optimality, we obtain the true upper bound. For any other finite value of ℓ, we have:

$$I_q(\mathbf{z};\mathbf{u}) \leq \mathbb{E}_{q_\phi(\mathbf{z},\mathbf{u})}[\log p_\psi(\mathbf{u}|\mathbf{z}) - \log p(\mathbf{u})] + \ell := C_2 + \ell. \tag{5.9}$$

5.2.4 A practical objective for controllable fair representations

Recall that our goal is to find tractable estimates to the mutual information terms in Equation 5.1 to make the objective and constraints tractable. In the previous sections, we have derived a lower bound for $I_q(\mathbf{x},\mathbf{u}|\mathbf{z})$ (which we want to maximize) and upper bounds for $I_q(\mathbf{u},\mathbf{z})$ (which we want to implicitly minimize to satisfy the constraint). Therefore, by applying these results to the optimization problem in Equation 5.1, we obtain the following constrained optimization problem:

$$\begin{aligned} \min_{\theta,\phi}\max_{\psi\in\Psi} \quad & \mathcal{L}_r = -\mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z},\mathbf{u})}[\log p_\theta(\mathbf{x}|\mathbf{z},\mathbf{u})] \\ \text{s.t.} \quad & C_1 = \mathbb{E}_{q(\mathbf{x},\mathbf{u})} D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x},\mathbf{u})\|p(\mathbf{z})) < \epsilon_1 \\ & C_2 = \mathbb{E}_{q_\phi(\mathbf{z},\mathbf{u})}[\log p_\psi(\mathbf{u}|\mathbf{z}) - \log p(\mathbf{u})] < \epsilon_2 \end{aligned} \tag{5.10}$$

where $\mathcal{L}_r$, C_1, and C_2 are introduced in Equations 5.2, 5.4 and 5.6 respectively.

Both C_1 and C_2 provide a way to limit $I_q(\mathbf{z},\mathbf{u})$. C_1 is guaranteed to be an upper bound to $I_q(\mathbf{z};\mathbf{u})$ but also upper-bounds $I_q(\mathbf{x};\mathbf{z}|\mathbf{u})$ (which we would like to maximize), so it is more suitable when we value true guarantees on fairness over expressiveness. C_2 may more accurately approximate $I_q(\mathbf{z};\mathbf{u})$ but is guaranteed to be an upper bound only in the case of an optimal adversary. Hence, it is more suited for scenarios where the user is satisfied with guarantees on fairness in the limit of adversarial training, and we wish to learn more expressive representations. Depending on the underlying application, the user can effectively remove either of the constraints C_1 or C_2 (or even both) by setting the corresponding ϵ to infinity.

5.3 A Unifying Framework for Related Work

Multiple methods for learning fair representations have been proposed in the literature. [ZWS+13] propose a method for clustering individuals into a small number of discrete fair representations. Discrete representations, however, lack the representational power of distributed representations, which vendors desire. In order to learn distributed fair representations, [ES15], [ELS+18] and [MCPZ18] each propose adversarial training, where the latter (LAFTR) connects different adversarial losses to multiple notions of fairness. [LSL+15] propose VFAE for learning distributed fair representations by using a variational autoencoder architecture with additional regularization based on Maximum Mean Discrepancy (MMD) [GBR+07]. Each of these methods is limited to the case of a binary sensitive attribute because their measurements of fairness are based on statistical parity [ZWS+13], which is defined only for two groups.

Interestingly, each of these methods can be viewed as optimizing an *approximation* of the Lagrangian dual of our objective in Equation 5.10, with particular *fixed* settings of the Lagrangian multipliers:

$$\begin{aligned} &\arg\min_{\theta,\phi}\max_{\psi} \mathcal{L}_r + \lambda_1(C_1 - \epsilon_1) + \lambda_2(C_2 - \epsilon_2) \\ =&\arg\min_{\theta,\phi}\max_{\psi} \mathcal{L}_r + \lambda_1 C_1 + \lambda_2 C_2 \end{aligned} \tag{5.11}$$

where $\mathcal{L}_r$, C_i and ϵ_i are defined as in Equation 5.10, and the multipliers $\lambda_i \geq 0$ are hyperparameters controlling the relative strengths of the constraints (which now act as "soft" regularizers).

We use "approximation" to suggest these objectives are not exactly the same as ours, as ours can deal with more than two groups in the fairness criterion C_2 and theirs cannot. However, all the fairness criteria achieve $\mathbf{z} \perp \mathbf{u}$ at a global optimum; in the following discussions, for brevity we use C_2 to indicate their objectives, even when they are not identical to ours[2].

Here, the values of ϵ do not affect the final solution. Therefore, if we wish to find representations that satisfy specific constraints, we would have to search over the hyperparameter space to find feasible solutions, which could be computationally inefficient. We call this class of approaches *Mutual Information-based Fair Representations* (MIFR[3]). In Table 5.1, we summarize these existing methods.

[2]We also have not included the task classification error in their methods, as we do not assume a single, specific task or assume access to labels in our setting.

[3]Pronounced "Mipha".

Table 5.1: **Summarizing the components in existing methods.** The hyperparameters (e.g. A_z, α, β) are from the original notations of the corresponding methods.

	λ_1	λ_2
Zemel *et al.* [ZWS+13]	0	A_z/A_x
Edwards and Storkey [ES15]	0	α/β
Madras *et al.* [MCPZ18]	0	γ/β
Louizos *et al.* [LSL+15]	1	β

- [ZWS+13] considers $\mathcal{L}_r$ as well as minimizing statistical parity (Equation 4 in their paper); they assume $\mathbf{z}$ is discrete, bypassing the need for adversarial training. Their objective is equivalent to Equation 5.11 with $\lambda_1 = 0, \lambda_2 = A_z/A_x$.
- [ES15] considers $\mathcal{L}_r$ (where $p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{u})$ is Gaussian) and adversarial training where the adversary tries to distinguish the representations from two groups (Equation 9). Their objective is equivalent to Equation 5.11 with $\lambda_1 = 0, \lambda_2 = \alpha/\beta$.
- [MCPZ18] considers $\mathcal{L}_r$ and adversarial training, which optimizes over surrogates to the demographic parity distance between two groups (Equation 4). Their objective is equivalent to Equation 5.11 with $\lambda_1 = 0, \lambda_2 = \gamma/\beta$.
- [LSL+15] considers $\mathcal{L}_r$, C_1 with $\lambda_1 = 1$ and the maximum mean discrepancy between two sensitive groups (C_2) (Equation 8). However, as $\mathcal{L}_r + C_1$ is the VAE objective, their solutions does not prefer high mutual information between $\mathbf{x}$ and $\mathbf{z}$ (referred to as the "information preference" property [CKS+16, ZSE17b, ZSE17a, ZSE18b]). Their objective is equivalent to Equation 5.11 with $\lambda_1 = 1, \lambda_2 = \beta$.

All of the above methods requires hand-tuning λ to govern the trade-off between the desiderata, because each of these approaches optimizes the dual with *fixed* multipliers instead of *optimizing* the multipliers to satisfy the fairness constraints, ϵ is ignored, so these approaches cannot ensure that the fairness constraints are satisfied. Using any of these approaches to empirically achieve a desirable limit on unfairness requires manually tuning the multipliers (e.g., increase some λ_i until the corresponding constraint is satisfied) over many experiments and is additionally difficult because there is no interpretable relationship between the multipliers and a *limit* on unfairness.

Our method is also related to other works on fairness [MGB+18] and information theory. [KTHS18] solve least square regression under multiple fairness constraints. [CWRV17] transform the dataset to prevent discrimination on specific classification tasks. [ZSE18b] discussed

information-theoretic constraints in the context of learning latent variable generative models, but did not discuss fairness.

5.4 Dual Optimization for Controllable Fair Representations

In order to exactly solve the dual of our practical objective from Equation 5.10 and guarantee that the fairness constraints are satisfied, we must optimize the model parameters as well as the Lagrangian multipliers, which we do using the following dual objective:

$$\max_{\boldsymbol{\lambda}\geq 0} \min_{\theta,\phi} \max_{\psi} \mathcal{L} = \mathcal{L}_r + \boldsymbol{\lambda}^\top (\mathbf{C} - \boldsymbol{\epsilon}) \tag{5.12}$$

where $\boldsymbol{\lambda} = [\lambda_1, \lambda_2]$ are the multipliers and $\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2]$ and $\mathbf{C} = [C_1, C_2]$ represent the constraints.

If we assume we are optimizing in the distribution space (i.e. Φ, Θ corresponds to the set of all valid distributions $(q_\phi(\mathbf{z}|\mathbf{x},\mathbf{u}), p_\theta(\mathbf{x}|\mathbf{z},\mathbf{u}), p_\theta(\mathbf{z}))$), then we can show that strong duality holds (our primal objective from Equation 5.10 equals our dual objective from Equation 5.12).

Theorem 7. *If $\epsilon_1, \epsilon_2 > 0$, then strong duality holds for the following optimization problem over distributions p_θ and q_ϕ:*

$$\min_{p_\theta, q_\phi} \quad -\mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z},\mathbf{u})}[\log p_\theta(\mathbf{x}|\mathbf{z},\mathbf{u})] \tag{5.13}$$

$$\begin{aligned} \textit{s.t.} \quad & \mathbb{E}_{q(\mathbf{x},\mathbf{u})} D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x},\mathbf{u}) \| p_\theta(\mathbf{z})) < \epsilon_1 \\ & \mathbb{E}_{q_\phi(\mathbf{z})} D_{\mathrm{KL}}(q_\phi(\mathbf{u}|\mathbf{z}) \| p(\mathbf{u})) < \epsilon_2 \end{aligned} \tag{5.14}$$

where q_ϕ denotes $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{u})$ and p_θ denotes $p_\theta(\mathbf{z})$ and $p_\theta(\mathbf{x}|\mathbf{z},\mathbf{u})$.

We show the complete proof in Appendix A.3.4. Intuitively, we utilize the convexity of KL divergence (over the pair of distributions) and mutual information (over the conditional distribution) to verify that Slater's conditions hold for this problem.

In practice, we can perform standard iterative gradient updates in the parameter space: standard gradient descent over θ, ϕ, gradient ascent over ψ (which parameterizes only the adversary), and gradient ascent over $\boldsymbol{\lambda}$. Intuitively, the gradient ascent over $\boldsymbol{\lambda}$ corresponds to a multiplier $\boldsymbol{\lambda}$ increasing when its constraint is not being satisfied, encouraging the representations to satisfy the fairness constraints even at a cost to representation expressiveness. Empirically, we show that this scheme is effective despite non-convexity in the parameter space.

Note that given finite model capacity, an $\boldsymbol{\epsilon}$ that is too small may correspond to no feasible solutions in the parameter space; that is, it may be impossible for the model to satisfy the specified fairness constraints. Here we introduce heuristics to estimate the mimimum feasible $\boldsymbol{\epsilon}$. The minimum feasible ϵ_1 and ϵ_3 can be estimated by running the standard conditional VAE algorithm on the same model and estimating the value of each divergence. Feasible ϵ_2 can be approximated by $H_q(\mathbf{u})$, since $I_q(\mathbf{z}; \mathbf{u}) \leq H_q(\mathbf{u})$; This can easily be estimated empirically when $\mathbf{u}$ is binary or discrete.

5.5 Experiments

We aim to experimentally answer the following:

- Do our information-theoretical objectives align well with existing notions of fairness?
- Do our constraints achieve their intended effects?
- How do MIFR and L-MIFR compare when learning controllable fair representations?
- How are the learned representations affected by other hyperparameters, such as the number of iterations used for adversarial training in C_2?
- Does L-MIFR have the potential to balance different notions of fairness?

5.5.1 Experimental Setup

We evaluate our results on three datasets [ZWS+13, LSM+17, MCPZ18]. The first is the UCI *German* credit dataset[4], which contains information about 1000 individuals, with a binary sensitive feature being whether the individual's age exceeds a threshold. The downstream task is to predict whether the individual is offered credit or not. The second is the UCI *Adult* dataset[5], which contains information of over 40,000 adults from the 1994 US Census. The downstream task is to predict whether an individual earns more than $50K/year. We consider the sensitive attribute to be gender, which is pre-processed to be a binary value. The third is the Heritage *Health* dataset[6], which contains information of over 60,000 patients. The downstream task is to predict whether the Charlson Index (an estimation of patient mortality) is greater than zero. Diverging from previous

[4] https://archive.ics.uci.edu/ml/datasets
[5] https://archive.ics.uci.edu/ml/datasets/adult
[6] https://www.kaggle.com/c/hhp

work [MCPZ18], we consider sensitive attributes to be age and gender, where there are 9 possible age values and 2 possible gender values; hence the sensitive attributes have 18 configurations. This prevents VFAE [LSL+15] and LAFTR [MCPZ18] from being applied, as both methods reply on some statistical distance between two groups, which is not defined when there are 18 groups in question[7].

We assume that the model does not have access to labels during training; instead, it supplies its representations to an unknown vendor's classifier, whose task is to achieve high prediction with labels. We compare the performance of *MIFR*, the model with fixed multipliers, and *L-MIFR*, the model using the Lagrangian dual optimization method. We provide details of the experimental setup in Appendix B.3.1. Specifically, we consider the simpler form for $p(\mathbf{z})$ commonly used in VAEs, where $p(\mathbf{z})$ is a fixed prior; the use of other more flexible parametrized forms of $p(\mathbf{z})$, such as normalizing flows [DSDB16, RM15] and autoregressive models [KSJ+16b, vdOKK16], is left as future work.

We estimate the mutual information values $I_q(\mathbf{x}; \mathbf{z}|\mathbf{u})$ and $I_q(\mathbf{u}; \mathbf{z})$ on the test set using the following equations:

$$I_q(\mathbf{x}; \mathbf{z}|\mathbf{u}) = \mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z},\mathbf{u})}[\log q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{u}) - \log q_\phi(\mathbf{z}|\mathbf{u})]$$
$$I_q(\mathbf{u}; \mathbf{z}) = \mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z},\mathbf{u})}[\log q_\phi(\mathbf{u}|\mathbf{z}) - \log q(\mathbf{u})]$$

where $q_\phi(\mathbf{z}|\mathbf{u})$ is estimated via kernel density estimation over samples from $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{u})$ with $(\mathbf{x}, \mathbf{u})$ sampled from the training set. Kernel density estimates are reasonable since both $\mathbf{z}$ and $\mathbf{u}$ are low dimensional (for example, *Adult* considers a 10-dimension $\mathbf{z}$ for 40,000 individuals). However, computing $q_\phi(\mathbf{z}|\mathbf{u})$ requires a summation over the training set, so we only compute these mutual information quantities during evaluation. We include our implementations in https://github.com/ermongroup/lag-fairness.

5.5.2 Mutual Information, Prediction Accuracy, and Fairness

We investigate the relationship between mutual information and prediction performance by considering area under the ROC curve (AUC) for prediction tasks. We also investigate the relationship between mutual information and traditional fairness metrics by considering the Δ_{DP} fairness metric in [MCPZ18], which compares the absolute expected difference in classifier outcomes between two groups. Δ_{DP} is only defined on two groups of classifier outcomes, so it is not defined for

[7] Δ_{DP} is only defined for binary sensitive variables in [MCPZ18].

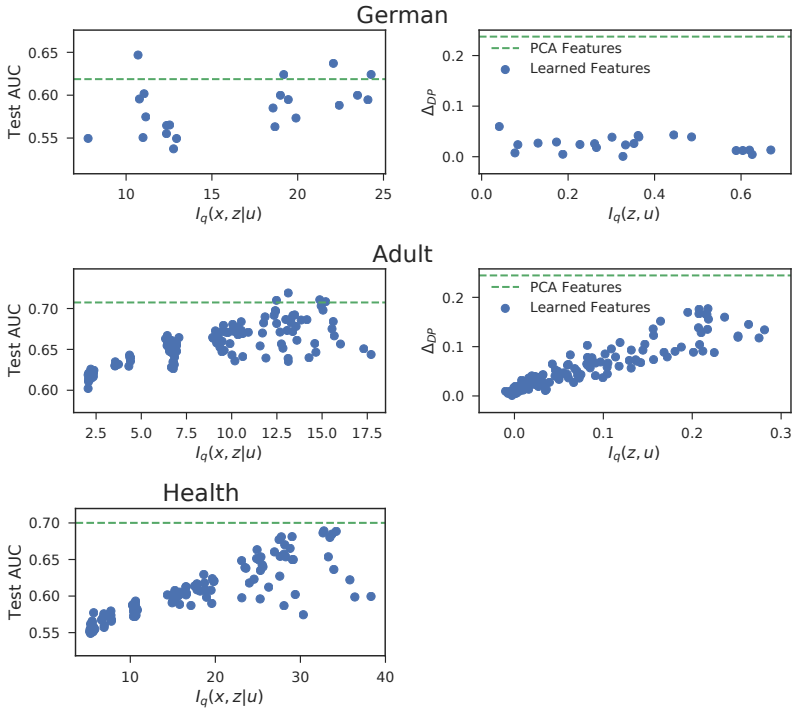


Figure 5.1: The relationship between mutual information and fairness related quantities. Each dot is the representations from an instance of MIFR with a different set of hyperparameters. Green line represents features obtained via principle component analysis. Increased mutual information between inputs and representations increase task performance (left) and unfairness (right). For *Health* we do not include Δ_{DP} since it is not defined for more than two groups.

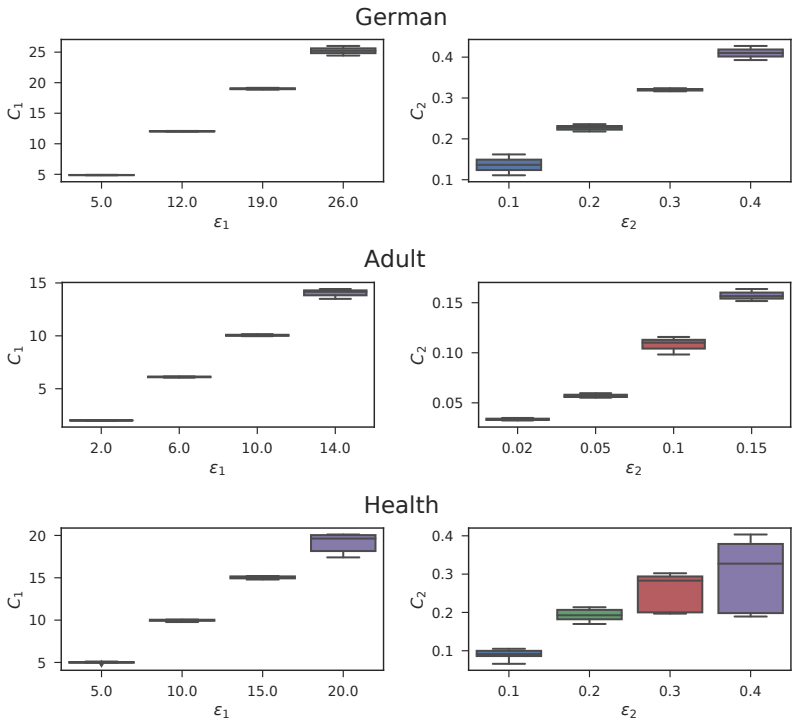


Figure 5.2: Corresponding C_i values under different ϵ_i with L-MIFR. After ϵ_i is fixed, we consider a range of values for the other constraint, leading to a distribution of C_i for each ϵ_i (hence the box plot).

the *Health* dataset when considering the sensitive attributes to be "age and gender", which has 18 groups. We use logistic regression classifiers for prediction tasks.

From the results results in Figure 5.1, we show that there are strong positive correlations between $I_q(\mathbf{x}; \mathbf{z}|\mathbf{u})$ and test AUC, and between $I_q(\mathbf{z}, \mathbf{u})$ and Δ_{DP}; increases in $I_q(\mathbf{z}, \mathbf{u})$ decrease fairness. We also include a baseline in Figure 5.1 where the features are obtained via the top-k principal components (where k is the dimension of $\mathbf{z}$), which has slightly better AUC but significantly worse fairness as measured by Δ_{DP}. Therefore, our information theoretic notions of fairness/expressiveness align well with existing notions such as Δ_{DP}/test AUC.

5.5.3 Controlling Representation Fairness with L-MIFR

Keeping all other constraint budgets fixed, any increase in ϵ_i for an arbitrary constraint C_i implies an increase in the unfairness budget; consequently, we are able to trade-off fairness for more informative representations when desired.

We demonstrate this empirically via an experiment where we note the C_i values corresponding to a range of budgets ϵ_i at a fixed configuration of the other constraint budgets ϵ_j ($j \neq i$). From Figure 5.2, C_i increases as ϵ_i increases, and $C_i < \epsilon_i$ holds under different values of the other constraints ϵ_j. This suggest that we can use ϵ_i to control C_i (our fairness criteria) of the learned representations.

We further show the changes in Δ_{DP} (a traditional fairness criteria) values as we vary ϵ_i in Figure 5.3. In *Adult*, Δ_{DP} clearly increases as ϵ_i increases; this is less obvious in *German*, as Δ_{DP} is already very low. These results suggest that the L-MIFR user can control the level of fairness of the representations quantitatively via ϵ.

5.5.4 Improving Representation Expressiveness with L-MIFR

Recall that our goal is to perform controlled fair representation learning, which requires us to learn expressive representations subject to fairness constraints. We compare two approaches that could achieve this: 1) MIFR, which has to consider a range of Lagrange multipliers (e.g. from a grid search) to obtain solutions that satisfy the constraints; 2) L-MIFR, which finds feasible solutions directly by optimizing the Lagrange multipliers.

We evaluate both methods on 4 sets of constraints by modifying the values of ϵ_2 (which is the tighter estimate of $I_q(\mathbf{z}; \mathbf{u})$) while keeping ϵ_1 fixed, and we compare the expressiveness of the features learned by the two methods in Figure 5.4. For MIFR, we perform a grid search running

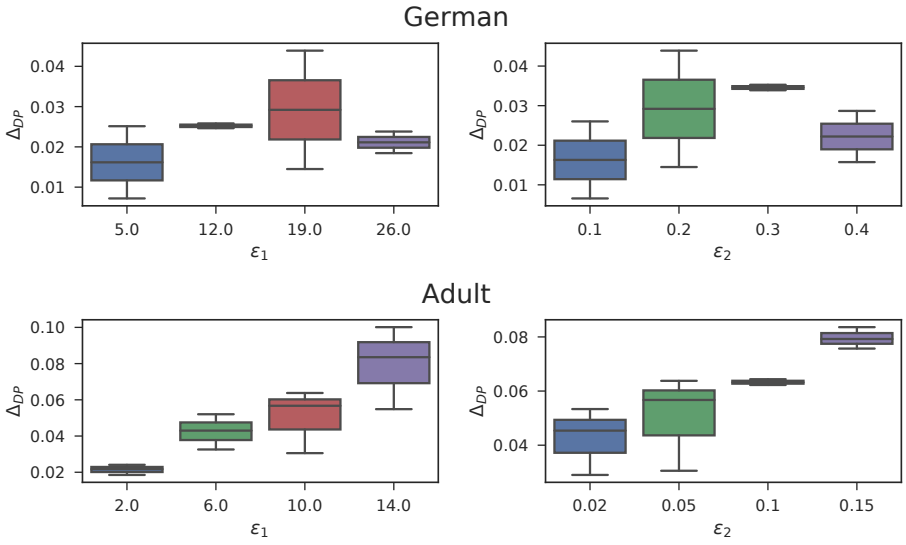


Figure 5.3: Δ_{DP} under different levels of ϵ with L-MIFR. Δ_{DP} generally increases as ϵ increases.

$5^2 = 25$ configurations. In contrast, we run *one* instance of L-MIFR for each ϵ setting, which takes roughly the same time to run as one instance of MIFR (the only overhead is updating the two scalar values λ_1 and λ_2).

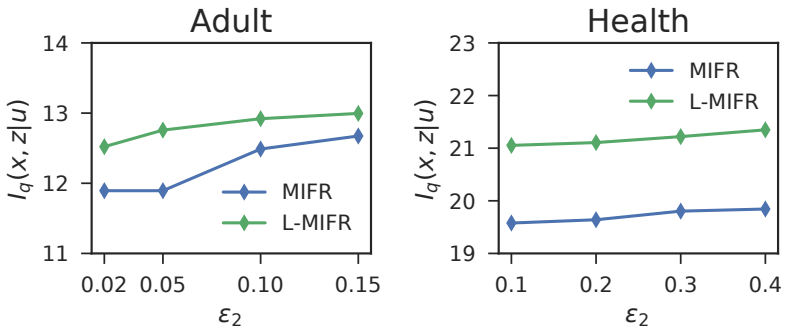


Figure 5.4: Expressiveness vs. ϵ_2. A larger feasible region (as measured by ϵ_2) leads to more expressive representations (as measured by $I_q(\mathbf{x}, \mathbf{z}|\mathbf{u})$).

In terms of representation expressiveness, L-MIFR outperforms MIFR even though MIFR took almost 25x the computational resources. Therefore, L-MIFR is significantly more computationally efficient than MIFR at learning controlled fair representation.

		D =1	D =2	D =5	D =10
Adult	$I_q(\mathbf{x};\mathbf{z}\|\mathbf{u})$	10.46	10.94	9.75	9.54
	$I_q(\mathbf{z};\mathbf{u})$	0.10	0.07	0.08	0.06
Health	$I_q(\mathbf{x};\mathbf{z}\|\mathbf{u})$	16.60	16.47	16.65	16.75
	$I_q(\mathbf{z};\mathbf{u})$	0.17	0.17	0.22	0.28

Table 5.2: Expressiveness and fairness of the representations from L-MIFR under various D.

5.5.5 Ablation Studies

The C_2 objective requires adversarial training, which involves iterative training of (θ, ϕ) with ψ. We assess the sensitivity of the expressiveness and fairness of the learned representations to the number of iterations D for ψ per iteration for (θ, ϕ). Following practices in [GAA+17] to have more iterations for critic, we consider $D = \{1, 2, 5, 10\}$, and use the same number of total iterations for training.

In Table 5.2, we evaluate $I_q(\mathbf{x};\mathbf{z}|\mathbf{u})$ and $I_q(\mathbf{z};\mathbf{u})$ obtained L-MIFR on *Adult* ($\epsilon_2 = 0.10$) and *Health* ($\epsilon_2 = 0.30$). This suggests that the final solution of the representations is not very sensitive to D, although larger D seem to find solutions that are closer to ϵ_2.

5.5.6 Fair Representations under Multiple Notions

Finally, we demonstrate how L-MIFR could control multiple fairness constraints simultaneously, thereby finding representations that are reasonably fair when there are multiple fairness notions being considered. We consider the *Adult* dataset, and describe the *demographic parity*, *equalized odds* and *equalized opportunity* notions of fairness in terms of mutual information, which we denote as $I_{DP} := I_q(\mathbf{z};\mathbf{u})$, I_{EO}, I_{EOpp} respectively (see details in Appendix B.3.3 about how I_{EO} and I_{EOpp} are derived).

For L-MIFR, we set $\epsilon_1 = 10$ and other ϵ values to 0.1. For MIFR, we consider a more efficient approach than random grid search. We start by setting every $\lambda = 0.1$; then we multiply the λ value for a particular constraint by 2 until the constraint is satisfied by MIFR; we finish when all the constraints are satisfied[8]. We find that this requires us to update the λ of I_{DP}, I_{EO} and I_{EOpp} four times each (so corresponding $\lambda = 1.6$); this costs 12x the computational resources needed by L-MIFR.

[8]This allows MIFR to approach the feasible set from outside, so the solution it finds will generally have high expressiveness.

	$I_q(\mathbf{x};\mathbf{z}\|\mathbf{u})$	C_1	I_{DP}	I_{EO}	I_{EOpp}
MIFR	9.34	**9.39**	0.09	0.10	0.07
L-MIFR	**9.94**	9.95	**0.08**	**0.09**	**0.04**

Table 5.3: Learning one representation for multiple notions of fairness on *Adult*. L-MIFR learns representations that are better than MIFR on all the measurements instead of only C_1. Here $\epsilon_1 = 10$ for C_1 and $\epsilon = 0.1$ for other constraints.

We compare the representations learned by L-MIFR and MIFR in Figure 5.3. L-MIFR outperforms MIFR in terms of $I_q(\mathbf{x};\mathbf{z}|\mathbf{u})$, I_{DP}, I_{EO} and I_{EOpp}, while only being slightly worse in terms of C_1. Since $\epsilon_1 = 10$, the L-MIFR solution is still feasible. This demonstrates that even with a thoughtfully designed method for tuning λ, MIFR is still much inferior to L-MIFR in terms of computational cost and representation expressiveness.

5.6 Discussion

In this chapter, we introduced an objective for learning controllable fair representations based on mutual information. This interpretation allows us to unify and explain existing work. In particular, we have shown that a range of existing approaches optimize an approximation to the Lagrangian dual of our objective with *fixed* multipliers, fixing the trade-off between fairness and expressiveness. We proposed a dual optimization method that allows us to achieve higher expressiveness while satisfying the user-specified limit on unfairness.

In future work, we are interested in formally and empirically extending this framework and the corresponding dual optimization method to other notions of fairness. It is also valuable to investigate alternative approaches to training the adversary [GAA+17], the usage of more flexible $p(\mathbf{z})$ [RM15], and alternative solutions to bounding $I_q(\mathbf{z},\mathbf{u})$.

Part II

Generation via Supervised Learning

Chapter 6

Generation with Reweighted Objectives

Generative adversarial networks (GANs) variants approximately minimize divergences between the model and the data distribution using a discriminator. Wasserstein GANs (WGANs) enjoy superior empirical performance, however, unlike in f-GANs, the discriminator does not provide an estimate for the ratio between model and data densities, which is useful in applications such as inverse reinforcement learning.

In this chapter, we propose an new training objective where we additionally optimize over a set of importance weights over the generated samples. By suitably constraining the feasible set of importance weights, we obtain a family of objectives which includes and generalizes the original f-GAN and WGAN objectives. We show that a natural extension outperforms WGANs while providing density ratios as in f-GAN, and demonstrate empirical success on distribution modeling, density ratio estimation and image generation.

6.1 Introduction

Learning generative models to sample from complex, high-dimensional distributions is an important task in machine learning with many important applications, such as image generation [KW13], imitation learning [HE16] and representation learning [CDH+16]. Generative adversarial networks (GANs, [GPAM+14]) are likelihood-free deep generative models [ML16] based on finding the equilibrium of a two-player minimax game between a generator and a critic (discriminator). Assuming the optimal critic is obtained, one can cast the GAN learning procedure as minimizing a discrepancy

measure between the distribution induced by the generator and the training data distribution.

Various GAN learning procedures have been proposed for different discrepancy measures. f-GANs [NCT16] minimize a variational approximation of the f-divergence between two distributions [Csi64, NWJ08]. In this case, the critic acts as a density ratio estimator [USS+16, GE17], i.e., it estimates if points are more likely to be generated by the data or the generator distribution. This includes the original GAN approach [GPAM+14] which can be seen as minimizing a variational approximation to the Jensen-Shannon divergence. Knowledge of the density ratio between two distributions can be used for importance sampling and in a range of practical applications such as mutual information estimation [DHFLM+18], off-policy policy evaluation [LLTZ18], and de-biasing of generative models [GSA+19].

Another family of GAN approaches are developed based on Integral Probability Metrics (IPMs, [Mül97]), where the critic (discriminator) is restricted to particular function families. For the family of Lipschitz-1 functions, the IPM reduces to the Wasserstein-1 or earth mover's distance [RTG00], which motivates the Wasserstein GAN (WGAN, [ACB17]) setting. Various approaches have been applied to enforce Lipschitzness, including weight clipping [ACB17], gradient penalty [GAA+17] and spectral normalization [MKKY18]. Despite its strong empirical success in image generation [KALL17, BDS18], the learned critic cannot be interpreted as a density ratio estimator, which limits its usefulness for importance sampling or other GAN-related applications such as inverse reinforcement learning [YSE19].

In this chapter, we address this problem via a generalized view of f-GANs and WGANs. The generalized view introduces importance weights over the generated samples in the critic objective, allowing prioritization over the training of different samples. The algorithm designer can select suitable feasible sets to constrain the importance weights; we show that both f-GAN and WGAN are special cases to this generalization when specific feasible sets are considered. We further discuss cases that select alternative feasible sets where divergences other than f-divergence and IPMs can be obtained.

To derive concrete algorithms, we turn to a case where the importance weights belong to the set of valid density ratios over the generated distribution. In certain cases, the optimal importance weights can be obtained via closed-form solutions, bypassing the need to perform an additional inner-loop optimization. We discuss one such approach, named KL-Wasserstein GAN (KL-WGAN), that is easy to implement from existing WGAN approaches, and is compatible with state-of-the-art GAN architectures. We evaluate KL-WGAN empirically on distribution modeling, density estimation and image generation tasks. Empirical results demonstrate that KL-WGAN enjoys

superior quantitative performance compared to its WGAN counterparts on several benchmarks.

6.2 Preliminaries

Notations Let X denote a random variable with separable sample space $\mathcal{X}$ and let $\mathcal{P}(\mathcal{X})$ denote the set of all probability measures over the Borel σ-algebra on $\mathcal{X}$. We use P, Q to denote probabiliy measures, and $P \ll Q$ to denote P is absolutely continuous with respect to Q, *i.e.*, the Radon-Nikodym derivative $\mathrm{d}P/\,\mathrm{d}Q$ exists. Under $Q \in \mathcal{P}(\mathcal{X})$, the p-norm of a function $r : \mathcal{X} \to \mathbb{R}$ is defined as

$$\|r\|_p := \left(\int |r(\boldsymbol{x})|^p \mathrm{d}Q(\boldsymbol{x}) \right)^{1/p}, \tag{6.1}$$

with $\|r\|_\infty = \lim_{p\to\infty} \|r\|_p$. The set of locally p-integrable functions is defined as

$$L^p(Q) := \{r : \mathcal{X} \to \mathbb{R} : \|r\|_p < \infty\}, \tag{6.2}$$

i.e. its norm with respect to Q is finite. We denote $L^p_{\geq 0}(Q) := \{r \in L^p(Q) : \forall \boldsymbol{x} \in \mathcal{X}, r(\boldsymbol{x}) \geq 0\}$ which considers non-negative functions in $L^p(Q)$. The space of probability measures wrt. Q is defined as

$$\Delta(Q) := \{r \in L^1_{\geq 0}(Q) : \|r\|_1 = 1\}. \tag{6.3}$$

For example, for any $P \ll Q$, $\mathrm{d}P/\,\mathrm{d}Q \in \Delta(Q)$ because $\int (\mathrm{d}P/\,\mathrm{d}Q)\,\mathrm{d}Q = 1$. We define $\mathbb{1}$ such that $\forall \boldsymbol{x} \in \mathcal{X}$, $\mathbb{1}(\boldsymbol{x}) = 1$, and define $\mathrm{im}(\cdot)$ and $\mathrm{dom}(\cdot)$ as image and domain of a function respectively.

Integral Probability Metrics and Wasserstein GANs For a fixed class of real-valued bounded Borel measurable functions $\mathcal{F}$ on $\mathcal{X}$, the integral probability metric (IPM) based on $\mathcal{F}$ and between $P, Q \in \mathcal{P}(\mathcal{X})$ is defined as:

$$\mathrm{IPM}_{\mathcal{F}}(P, Q) := \sup_{T \in \mathcal{F}} \left| \int T(\boldsymbol{x})\,\mathrm{d}P(\boldsymbol{x}) - \int T(\boldsymbol{x})\,\mathrm{d}Q(\boldsymbol{x}) \right|.$$

If for all $T \in \mathcal{F}$, $-T \in \mathcal{F}$ then $\mathrm{IPM}_{\mathcal{F}}$ forms a metric over $\mathcal{P}(\mathcal{X})$ [Mül97]; we assume this is always true for $\mathcal{F}$ in this paper (so we can remove the absolute values). In particular, if $\mathcal{F}$ is the set of all bounded 1-Lipschitz functions with respect to the metric over $\mathcal{X}$, then the corresponding IPM

becomes the Wasserstein distance between P and Q [Vil08]. This motivates the Wasserstein GAN objective [ACB17]:

$$\min_{\theta} \max_{\phi} \mathbb{E}_{\boldsymbol{x}\sim P_{\text{data}}}[T_\phi(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim Q_\theta}[T_\phi(\boldsymbol{x})], \tag{6.4}$$

where T_ϕ is regularized to be approximately k-Lipschitz for some k. Various approaches have been applied to enforce Lipschitzness of neural networks, including weight clipping [ACB17], gradient penalty [GAA+17], and spetral normalization over the weights [MKKY18].

Despite its strong empirical performance, WGAN has two drawbacks. First, unlike f-GAN (Lemma 2), it does not naturally recover a density ratio estimator from the critic. Granted, the WGAN objective corresponds to an f-GAN one [SFG+09] when $f(x) = 0$ if $x = 1$ and $f(x) = +\infty$ otherwise, so that $f^*(x) = x$; however, we can no longer use Lemma 2 to recover density ratios given an optimal critic T, because the derivative $f'(x)$ does not exist. Second, WGAN places the same weight on the objective for each generated sample, which could be sub-optimal when the generated samples are of different qualities.

6.3 A Generalization of f-GANs and WGANs

In order to achieve the best of both worlds, we propose an alternative generalization to the critic objectives to *both* f-GANs and WGANs. Consider the following functional:

$$\ell_f(T, r; P, Q) := \mathbb{E}_{\boldsymbol{x}\sim Q}[f(r(\boldsymbol{x}))] + \mathbb{E}_{\boldsymbol{x}\sim P}[T(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim Q}[r(\boldsymbol{x}) \cdot T(\boldsymbol{x})] \tag{6.5}$$

which depends on the distributions P and Q, the critic function $T : \mathcal{X} \to \mathbb{R}$, and an additional function $r : \mathcal{X} \to \mathbb{R}$. For conciseness, we remove the dependency on the argument $\boldsymbol{x}$ for T, r, P, Q in the remainder of the paper.

The function $r : \mathcal{X} \to \mathbb{R}$ here plays the role of "importance weights", as they changes the weights to the critic objective over the generator samples. When $r = \mathrm{d}P/\,\mathrm{d}Q$, the objective above simplifies to $\mathbb{E}_Q[f(\mathrm{d}P/\,\mathrm{d}Q)]$ which is exactly the definition of the f-divergence between P and Q (Eq. 2.5).

To recover an objective over only the critic T, we minimize ℓ_f as a function of r over a suitable set $\mathcal{R} \subseteq L^\infty_{\geq 0}(Q)$, thus eliminating the dependence over r:

$$\mathcal{L}_f^{\mathcal{R}}(T; P, Q) := \inf_{r\in\mathcal{R}} \ell_f(T, r; P, Q) \tag{6.6}$$

We note that the minimization step is performed within a particular set $\mathcal{R} \subseteq L^\infty(Q)$, which can be selected by the algorithm designer. The choice of the set $\mathcal{R}$ naturally gives rise to different critic objectives. As we demonstrate below (and in Figure 6.1), we can obtain critic objectives for f-GAN as well as WGANs as special cases via different choices of $\mathcal{R}$ in $\mathcal{L}_f^{\mathcal{R}}(T; P, Q)$.

6.3.1 Recovering the f-GAN Critic Objective

First, we can recover the critic in the f-GAN objective by setting $\mathcal{R} = L^\infty_{\geq 0}(Q)$, which is the set of all non-negative functions in $L^\infty(Q)$. Recall from Lemma 2 the f-GAN objective:

$$D_f(P\|Q) = \sup_{T \in L^\infty(Q)} I_f(T; P, Q) \tag{6.7}$$

where $I_f(T; P, Q) := \mathbb{E}_P[T] - \mathbb{E}_Q[f^*(T)]$ as defined in Lemma 2. The following proposition shows that when $\mathcal{R} = L^\infty_{\geq 0}(Q)$, we recover $I_f = \mathcal{L}_f^{\mathcal{R}}$.

Proposition 1. *Assume that f is differentiable at $[0, \infty)$. $\forall P, Q \in \mathcal{P}(\mathcal{X})$ such that $P \ll Q$, and $\forall T \in \mathcal{F} \subseteq L^\infty(Q)$ such that $\mathrm{im}(T) \subseteq \mathrm{dom}((f')^{-1})$,*

$$I_f(T; P, Q) = \inf_{r \in L^\infty_{\geq 0}(Q)} \ell_f(T, r; P, Q). \tag{6.8}$$

where $I_f(T; P, Q) := \mathbb{E}_P[T] - E_Q[f^*(T)]$.

Proof. From Fenchel's inequality we have for convex $f : \mathbb{R} \to \mathbb{R}$, $\forall T(\boldsymbol{x}) \in \mathbb{R}$ and $\forall r(\boldsymbol{x}) \geq 0$, $f(r(\boldsymbol{x})) + f^*(T(\boldsymbol{x})) \geq r(\boldsymbol{x})T(\boldsymbol{x})$ where equality holds when $T(\boldsymbol{x}) = f'(r(\boldsymbol{x}))$. Taking the expectation over Q, we have

$$\mathbb{E}_Q[f(r)] - \mathbb{E}_Q[rT] \geq -\mathbb{E}_Q[f^*(T)]; \tag{6.9}$$

applying this to the definition of $\ell_f(T, r; P, Q)$, we have:

$$\begin{aligned} \ell_f(T, r; P, Q) &:= \mathbb{E}_Q[f(r)] + \mathbb{E}_P[T] - \mathbb{E}_Q[rT] \\ &\geq \mathbb{E}_P[T] - \mathbb{E}_Q[f^*(T)] = I_f(T; P, Q). \end{aligned} \tag{6.10}$$

where the inequality comes from Equation 6.9. The inequality becomes an equality when $r(\boldsymbol{x}) = (f')^{-1}(T(\boldsymbol{x}))$ for all $\boldsymbol{x} \in \mathcal{X}$. We note that such a case can be achieved, i.e., $(f')^{-1}(T) \in L^\infty_{\geq 0}(Q)$, because $\forall \boldsymbol{x} \in \mathcal{X}, (f')^{-1}(T(\boldsymbol{x})) \in \mathrm{dom}(f) = [0, \infty)$ from the assumption over $\mathrm{im}(T)$. Therefore,

taking the infimum over $r \in L^{\infty}_{\geq 0}(Q)$, we have:

$$I_f(T; P, Q) = \inf_{r \in L^{\infty}_{\geq 0}(Q)} \ell_f(T, r; P, Q), \tag{6.11}$$

which completes the proof. □

6.3.2 Recovering the WGAN Critic Objective

Next, we recover the WGAN critic objective (IPM) by setting $\mathcal{R} = \{\mathbb{1}\}$, where $\mathbb{1}(x) = 1$ is a constant function. First, we can equivalently rewrite the definition of an IPM using the following notation:

$$\text{IPM}_{\mathcal{F}}(P, Q) = \sup_{T \in \mathcal{F}} I_W(T; P, Q) \tag{6.12}$$

where I_W represents the critic objective. We show that $I_W = \mathcal{L}_f^{\mathcal{R}}$ when $\mathcal{R} = \{\mathbb{1}\}$ as follows.

Proposition 2. *$\forall P, Q \in \mathcal{P}(\mathcal{X})$ such that $P \ll Q$, and $\forall T \in \mathcal{F} \subseteq L^{\infty}(Q)$:*

$$I_W(T; P, Q) = \inf_{r \in \{\mathbb{1}\}} \ell_f(T, r; P, Q) \tag{6.13}$$

where $I_W(T; P, Q) := \mathbb{E}_P[T] - \mathbb{E}_Q[T]$.

Proof. As $\{\mathbb{1}\}$ has only one element, the infimum is:

$$\ell_f(T, \mathbb{1}; P, Q) = \mathbb{E}_Q[f(1)] + \mathbb{E}_P[T] - \mathbb{E}_Q[T] = I_W(T; P, Q) \tag{6.14}$$

where we used $f(1) = 0$ for the second equality. □

The above propositions show that $\mathcal{L}_f^{\mathcal{R}}$ generalizes both f-GAN and WGANs critic objectives by setting $\mathcal{R} = L^{\infty}_{\geq 0}(Q)$ and $\mathcal{R} = \{\mathbb{1}\}$ respectively.

6.3.3 Extensions to Alternative Constraints

The generalization with $\mathcal{L}_f^{\mathcal{R}}$ allows us to introduce new objectives when we consider alternative choices for the constraint set $\mathcal{R}$. We consider sets $\mathcal{R}$ such that $\{\mathbb{1}\} \subseteq \mathcal{R} \subseteq L^{\infty}_{\geq 0}(Q)$. The following proposition shows for some fixed T, the corresponding objective with $\mathcal{R}$ is bounded between the f-GAN objective (where $\mathcal{R} = L^{\infty}_{\geq 0}(Q)$) and the WGAN objective (where $\mathcal{R} = \{\mathbb{1}\}$).

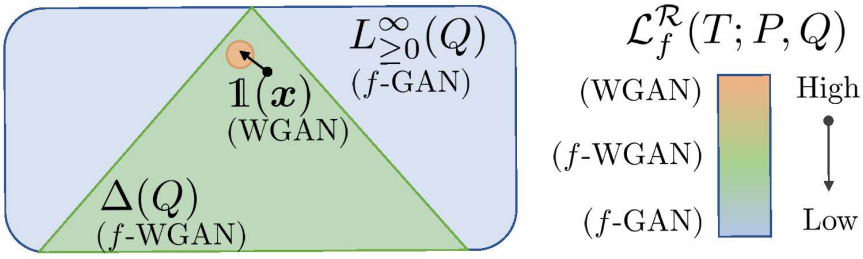


Figure 6.1: **High-level idea of f-WGAN.** (Left) Minimization over different $\mathcal{R}$ in $\mathcal{L}^{\mathcal{R}}_f$ gives different critic objectives. Minimizing over $L^{\infty}_{\geq 0}(Q)$ recovers f-GAN (blue set), minimizing over $\{\mathbb{1}\}$ recovers WGAN (orange set), and minimizing over $\Delta(Q)$ recovers f-WGAN (green set). (Right) Naturally, as we consider smaller sets $\mathcal{R}$ to minimize over, the critic objective becomes larger for the same T.

Proposition 3. *$\forall P, Q \in \mathcal{P}(\mathcal{X})$ such that $P \ll Q$, $\forall T \in L^{\infty}(Q)$ such that $\mathrm{im}(T) \subseteq \mathrm{dom}((f')^{-1})$, and $\forall \mathcal{R} \subseteq L^{\infty}_{\geq 0}(Q)$ such that $\{\mathbb{1}\} \subseteq \mathcal{R}$ we have:*

$$I_f(T;P,Q) \leq \mathcal{L}^{\mathcal{R}}_f(T;P,Q) \leq I_W(T;P,Q). \tag{6.15}$$

Proof. In Appendix A.4. □

We visualize this in Figure 6.1. Selecting the set $\mathcal{R}$ allows us to control the critic objective in a more flexible manner, interpolating between the f-GAN critic and the IPM critic objective and finding suitable trade-offs. Moreover, if we additionally take the supremum of $\mathcal{L}^{\mathcal{R}}_f(T;P,Q)$ over T, the result will be bounded between the supremum of I_f over T (corresponding to the f-divergence) and the supremum of I_W over T, as stated in the following theorem.

Theorem 8. *For $\{\mathbb{1}\} \subseteq \mathcal{R} \subseteq L^{\infty}_{\geq 0}(Q)$, define*

$$D_{f,\mathcal{R}}(P\|Q) := \sup_{T \in \mathcal{F}} \mathcal{L}^{\mathcal{R}}_f(T;P,Q) \tag{6.16}$$

where $\mathcal{F} := \{T : \mathcal{X} \to \mathrm{dom}((f')^{-1}), T \in L^{\infty}(Q)\}$. Then

$$D_f(P\|Q) \leq D_{f,\mathcal{R}}(P\|Q) \leq \sup_{T \in \mathcal{F}} I_W(T;P,Q). \tag{6.17}$$

Proof. In Appendix A.4. □

A natural corollary is that $D_{f,\mathcal{R}}$ defines a divergence between two distributions.

Corollary 5. *$D_{f,\mathcal{R}}(P\|Q)$ defines a divergence between P and Q: $D_{f,\mathcal{R}}(P\|Q) \geq 0$ for all $P, Q \in \mathcal{P}(\mathcal{X})$, and $D_{f,\mathcal{R}}(P\|Q) = 0$ if and only if $P = Q$.*

This allows us to interpret the corresponding GAN algorithm as variational minimization of a certain divergence bounded between the corresponding f-divergence and IPM.

6.4 Practical f-Wasserstein GANs

As a concrete example, we consider the set $\mathcal{R} = \Delta(Q)$, which is the set of all valid density ratios over Q. We note that $\{\mathbb{1}\} \subset \Delta(Q) \subset L^{\infty}_{\geq 0}(Q)$ (see Figure 6.1), so the corresponding objective is a divergence (from Corollary 5). We can then consider the variational divergence minimization objective over $\mathcal{L}_f^{\Delta(Q)}(T; P, Q)$:

$$\inf_{Q \in \mathcal{P}(\mathcal{X})} \sup_{T \in \mathcal{F}} \inf_{r \in \Delta(Q)} \ell_f(T, r; P, Q), \tag{6.18}$$

We name this the "f-Wasserstein GAN" (f-WGAN) objective, since it provides an interpolation between f-GAN and Wasserstein GANs while recovering a density ratio estimate between two distributions.

6.4.1 KL-Wasserstein GANs

For the f-WGAN objective in equation 6.18, the trivial algorithm would have to perform iterative updates to three quantities Q, T and r, which involves three nested optimizations. While this seems impractical, we show that for certain choices of f-divergences, we can obtain closed-form solutions for the optimal $r \in \Delta(Q)$ in the innermost minimization; this bypasses the need to perform an inner-loop optimization over $r \in \Delta(Q)$, as we can simply assign the optimal solution from the close-form expression.

Theorem 9. *Let $f(u) = u \log u$ and $\mathcal{F}$ a set of real-valued bounded measurable functions on $\mathcal{X}$. For any fixed choice of P, Q, and $T \in \mathcal{F}$, we have*

$$\arg\min_{r \in \Delta(Q)} \mathbb{E}_Q[f(r)] + \mathbb{E}_P[T] - \mathbb{E}_Q[r \cdot T] = \frac{e^T}{\mathbb{E}_Q[e^T]} \tag{6.19}$$

Proof. In Appendix A.4. □

The above theorem shows that if the f-divergence of interest is the KL divergence, we can directly obtain the optimal $r \in \Delta(Q)$ using equation 6.19 for any fixed critic T. Then, we can apply this r to the f-WGAN objective, and perform gradient descent updates on Q and T only. Avoiding the optimization procedure over r allows us to propose practical algorithms that are similar to existing WGAN procedures. In Appendix B.4.2, we show a similar argument with χ^2-divergence, another f-divergence admitting a closed-form solution, and discuss its connections with the χ^2-GAN approach [TCH+18].

6.4.2 Implementation Details

In Algorithm 1, we describe KL-Wasserstein GAN (KL-WGAN), a practical algorithm motivated by the f-WGAN objectives based on the observations in Theorem 9. We note that r_0 corresponds to selecting the optimal value for r from Theorem 9; once r_0 is selected, we ignore the effect of $E_Q[f(r_0)]$ to the objective and optimize the networks with the remaining terms, which corresponds to weighting the generated samples with r_0; the critic will be updated as if the generated samples are reweighted. In particular, $\nabla_\phi(D_0 - D_1)$ corresponds to the critic gradient (T, which is parameterized by ϕ) and $\nabla_\theta D_1$ corresponds to the generator gradient (Q, parameterized by θ).

In terms of implementation, the only differences between KL-WGAN and WGAN are between lines 8 and 11, where WGAN will assign $r_0(\boldsymbol{x}) = 1$ for all $\boldsymbol{x} \sim Q_m$. In contrast, KL-WGAN "importance weights" the samples using the critic, in the sense that it will assign higher weights to samples that have large $T_\phi(\boldsymbol{x})$ and lower weights to samples that have low $T_\phi(\boldsymbol{x})$. This will encourage the generator $Q_\theta(\boldsymbol{x})$ to put more emphasis on samples that have high critic scores. It is relatively easy to implement the KL-WGAN algorithm from an existing WGAN implementation, as we only need to modify the loss function. We present an implementation of KL-WGAN losses (in PyTorch) in Appendix B.4.1.

While the mini-batch estimation for $r_0(\boldsymbol{x})$ provides a biased estimate to the optimal $r \in \Delta(Q)$ (which according to Theorem 9 is $e^{T_\theta(\boldsymbol{x})}/\mathbb{E}_Q[e^{T_\theta(\boldsymbol{x})}]$, i.e., normalized with respect to Q instead of over a minibatch of m samples as done in line 8), we found that this does not affect performance significantly. We further note that computing $r_0(\boldsymbol{x})$ does not require additional network evaluations, so the computational cost for each iteration is nearly identical between WGAN and KL-WGAN.

Algorithm 1 Pseudo-code for KL-Wasserstein GAN

Input: the (empirical) data distribution p_{data};
Output: implicit generative model Q_θ.
Initialize generator Q_θ and discriminator T_ϕ.
repeat
    Draw $P_m := m$ *i.i.d.* samples from p_{data};
    Draw $Q_m := m$ *i.i.d.* samples from $Q_\theta(\boldsymbol{x})$.
    Compute $D_1 := \mathbb{E}_{P_m}[T_\phi(\boldsymbol{x})]$ (real samples)
    for all $\boldsymbol{x} \in Q_m$ (fake samples) **do**
        Compute $r_0(\boldsymbol{x}) := e^{T_\phi(\boldsymbol{x})}/\mathbb{E}_{Q_m}[e^{T_\phi(\boldsymbol{x})}]$
    end for
    Compute $D_0 := \mathbb{E}_{Q_m}[r_0(\boldsymbol{x})T_\phi(\boldsymbol{x})]$.
    Perform SGD over θ with $-\nabla_\theta D_0$;
    Perform SGD over ϕ with $\nabla_\phi(D_0 - D_1)$.
    Regularize T_ϕ to satisfy k-Lipschitzness.
until Stopping criterion **return** learned implicit generative model Q_θ.

6.5 Related Work

6.5.1 f-divergences, IPMs and GANs

Variational f-divergence minimization and IPM minimization paradigms are widely adopted in GANs. A non-exhaustive list includes f-GAN [NCT16], Wasserstein GAN [ACB17], MMD-GAN [LCC+17], WGAN-GP [GAA+17], SNGAN [MKKY18], LSGAN [MLX+17], etc. The f-divergence paradigms enjoy better interpretations over the role of learned discriminator (in terms of density ratio estimation), whereas IPM-based paradigms enjoy better training stability and empirical performance. Prior work have connected IPMs with χ^2 divergences between mixtures of data and model distributions [MLX+17, TCH+18, MS17]; our approach can be applied to χ^2 divergences as well, and we discuss its connections with χ^2-GAN in Appendix B.4.2.

Several works [LBC17, FT18] considered restricting function classes directly over the f-GAN objective; [HNW19] show that restricted f-GAN objectives are lower bounds to Wasserstein autoencoder [TBGS17] objectives, aligning with our argument for f-GAN and WGAN (Figure 6.1).

Our approach is most related to regularized variational f-divergence estimators [NWJ10, RRGGP12] and linear f-GANs [LBC17, LC18] where the function family $\mathcal{F}$ is a RKHS with fixed "feature maps". Different from these approaches, ours naturally allows the "feature maps" to be learned. Moreover, considering both restrictions allows us to bypass inner-loop optimization via closed-form solutions in certain cases (such as KL or χ^2 divergences); this leads to our KL-WGAN

approach which is easy to implement from existing WGAN implementations, and also have similar computational cost per iteration.

6.5.2 Reweighting of Generated Samples

The learned discriminators in GANs can further be used to perform reweighting over the generated samples [TCH+18]; these include rejection sampling [AOD+18], importance sampling [GSA+19, TCH+18], and Markov chain monte carlo [THF+18]. These approaches can only be performed after training has finished, unlike our KL-WGAN case where discriminator-based reweighting are performed during training.

Moreover, prior reweighting approaches assume that the discriminator learns to approximate some (fixed) function of the density ratio $\mathrm{d}p_{\mathrm{data}}/\,\mathrm{d}Q_\theta$, which does not apply directly to general IPM-based GAN objectives (such as WGAN); in KL-WGAN, we interpret the discriminator outputs as (un-normalized, regularized) log density ratios, introducing the density ratio interpretation to the IPM paradigm. We note that post-training discriminator-based reweighting can also be applied to our approach, and is orthogonal to our contributions; we leave this as future work.

Table 6.1: Negative Log-likelihood (NLL) and Maximum mean discrepancy (MMD, multiplied by 10^3) results on six 2-d synthetic datasets. Lower is better. W denotes the original WGAN objective, and KL-W denotes the proposed KL-WGAN objective.

Metric	GAN	MoG	Banana	Rings	Square	Cosine	Funnel
NLL	W	2.65 ± 0.00	3.61 ± 0.02	$\mathbf{4.25} \pm 0.01$	3.73 ± 0.01	$\mathbf{3.98} \pm 0.00$	3.60 ± 0.01
	KL-W	$\mathbf{2.54} \pm 0.00$	$\mathbf{3.57} \pm 0.00$	$\mathbf{4.25} \pm 0.00$	$\mathbf{3.72} \pm 0.00$	4.00 ± 0.01	$\mathbf{3.57} \pm 0.00$
MMD	W	25.45 ± 7.78	3.33 ± 0.59	2.05 ± 0.47	2.42 ± 0.24	$\mathbf{1.24} \pm 0.40$	1.71 ± 0.65
	KL-W	$\mathbf{6.51} \pm 3.16$	$\mathbf{1.45} \pm 0.12$	$\mathbf{1.20} \pm 0.10$	$\mathbf{1.10} \pm 0.23$	1.33 ± 0.23	$\mathbf{1.08} \pm 0.23$

6.6 Experiments

We release code for our experiments in https://github.com/ermongroup/f-wgan.

6.6.1 Synthetic and UCI Benchmark Datasets

We first demonstrate the effectiveness of KL-WGAN on synthetic and UCI benchmark datasets [AN07] considered in [WSSG18]. The 2-d synthetic datasets include Mixture of Gaussians (MoG), Banana, Ring, Square, Cosine and Funnel; these datasets cover different modalities and geometries. We use

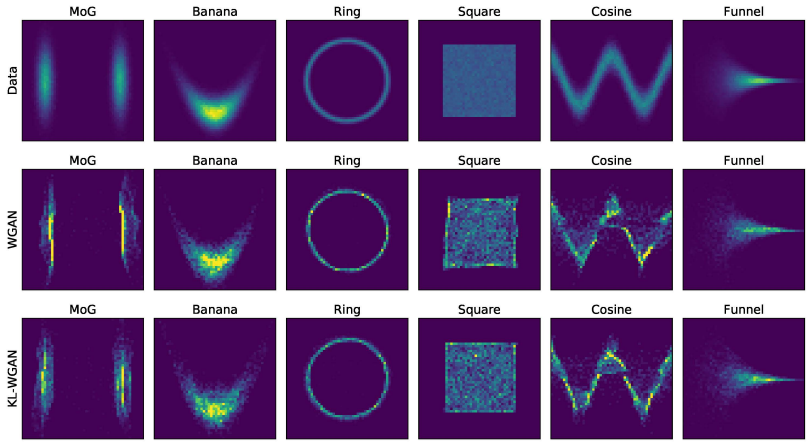


Figure 6.2: **KL-WGAN samples on 2d distributions.** Histograms of samples from the data distribution (top), WGAN (middle) and our KL-WGAN (bottom).

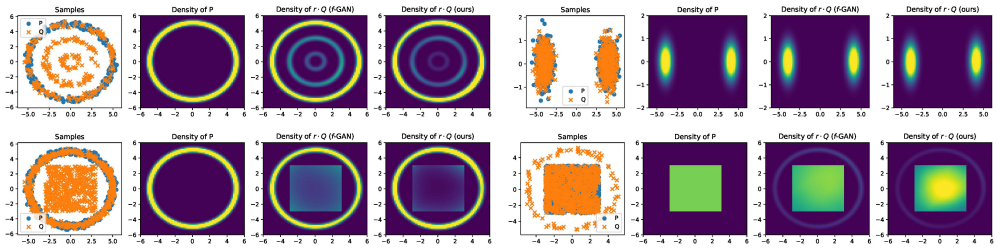


Figure 6.3: **Estimating density ratios with KL-WGAN.** The first column contains the samples used for training, the second column is the ground truth density of P, the third and fourth columns are the density of Q times the estimated density ratios from original f-GAN (third column) and our KL-WGAN (fourth column).

RedWine, WhiteWine and Parkinsons from the UCI datasets. We use the same SNGAN [MKKY18] arhictetures for WGAN and KL-WGANs, which uses spectral normalization to enforce Lipschitzness (detailed in Appendix B.4.3).

After training, we draw 5,000 samples from the generator and then evaluate two metrics over a fixed validation set. One is the negative log-likelihood (NLL) of the validation samples on a kernel density estimator fitted over the generated samples; the other is the maximum mean discrepancy (MMD, [BGR+06]) between the generated samples and validation samples. To ensure a fair comparison, we use identical kernel bandwidths for all cases.

Distribution modeling We report the mean and standard error for the NLL and MMD results in Tables 6.1 and 6.2 (with 5 random seeds in each case) for the synthetic datasets and UCI datasets

Table 6.2: Negative Log-likelihood (NLL, top two rows) and Maximum mean discrepancy (MMD, multiplied by 10^3, bottom two rows) results on real-world datasets. Lower is better for both evaluation metrics. W denotes the original WGAN objective, and KL denotes the proposed KL-WGAN objective.

	RedWine	WhiteWine	Parkinsons
W	14.55 ± 0.04	14.12 ± 0.02	20.24 ± 0.08
KL	$\mathbf{14.41} \pm 0.03$	$\mathbf{14.08} \pm 0.02$	$\mathbf{20.16} \pm 0.05$
W	2.61 ± 0.37	1.32 ± 0.10	1.30 ± 0.09
KL	$\mathbf{2.55} \pm 0.11$	$\mathbf{1.23} \pm 0.17$	$\mathbf{0.84} \pm 0.04$

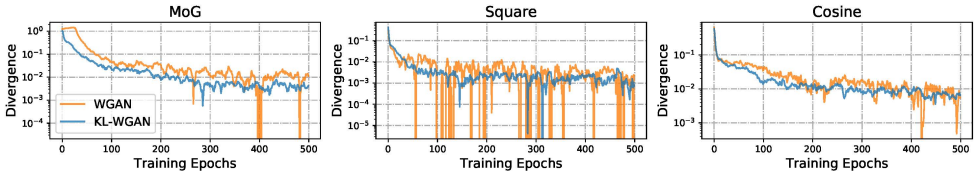


Figure 6.4: **Estimated divergence with KL-WGAN.** The results are shown with respect to training epochs (smoothed with a window of 10).

respectively. The results demonstrate that our KL-WGAN approach outperforms its WGAN counterpart on all but the Cosine dataset. From the histograms of samples in Figure 6.2, we can visually observe where our KL-WGAN performs significantly better than WGAN. For example, WGAN fails to place enough probability mass in the center of the Gaussians in MoG and fails to learn a proper square in Square, unlike our KL-WGAN approaches.

Density ratio estimation We demonstrate that adding the constraint $r \in \Delta(Q)$ leads to effective density ratio estimators. We consider measuring the density ratio from synthetic datasets, and compare them with the original f-GAN with KL divergence. We evaluate the density ratio estimation quality by multiplying $\mathrm{d}Q$ with the estimated density ratios, and compare that with the density of P; ideally the two quantities should be identical. We demonstrate empirical results in Figure 6.3, where we plot the samples used for training, the ground truth density of P and the two estimates given by two methods. In terms of estimating density ratios, our proposed approach is comparable to the f-GAN one.

Table 6.3: **Inception and FID scores for CIFAR10 image generation.** We list comparisons with results reported by WGAN-GP [GAA+17], Fisher GAN [MS17], χ^2 GAN [TCH+18], MoLM [RMRV18], SNGAN [MKKY18], NCSN [SE19c], BigGAN [BDS18] and Sphere GAN [PK19]. (*) denotes our experiments with the PyTorch BigGAN implementation.

Method	Inception score	FID score
CIFAR10 Unconditional		
WGAN-GP	$7.86 \pm .07$	-
Fisher GAN	$7.90 \pm .05$	-
MoLM	$7.90 \pm .10$	18.9
SNGAN	$8.22 \pm .05$	21.7
Sphere GAN	$8.39 \pm .08$	**17.1**
NCSN	**8.91**	25.32
BigGAN*	$8.60 \pm .10$	16.38
KL-BigGAN*	**8.66** $\pm .09$	**15.23**
CIFAR10 Conditional		
Fisher GAN	$8.16 \pm .12$	-
WGAN-GP	$8.42 \pm .10$	-
χ^2-GAN	$8.44 \pm .10$	-
SNGAN	$8.60 \pm .08$	17.5
BigGAN	**9.22**	**14.73**
BigGAN*	$9.08 \pm .11$	9.51
KL-BigGAN*	**9.20** $\pm .09$	**9.17**

Stability of critic objectives For the MoG, Square and Cosine datasets, we further show the estimated divergences over a batch of 256 samples in Figure 6.4, where WGAN uses I_W and KL-WGAN uses the proposed $\mathcal{L}_f^{\Delta(Q)}$. While both estimated divergences decrease over the course of training, our KL-WGAN divergence is more stable on all three cases. In addition, we evaluate the number of occurrences when a negative estimate of the divergences was produced for an epoch (which contradicts the fact that divergences should be non-negative); over 500 batches, WGAN has 46, 181 and 55 occurrences on MoG, Square and Cosine respectively, while KL-WGAN only has 29, 100 and 7 occurrences. This suggests that the proposed objective is easier to estimate and optimize, and is more stable across different iterations.

Table 6.4: **FID scores for CelebA image generation.** The mean and standard deviation are obtained from 4 instances trained with different random seeds.

Method	Image Size	FID score
BigGAN	64×64	18.07 ± 0.47
KL-BigGAN		$\mathbf{17.70} \pm 0.32$

6.6.2 Image Generation

We further evaluate our KL-WGAN's practical on image generation tasks on CIFAR10 and CelebA datasets. Our experiments are based on the BigGAN [BDS18] PyTorch implementation. We use a smaller network than the one reported in [BDS18] (implemented on TensorFlow), using the default architecture in the PyTorch implementation.

We compare training a BigGAN network with its original objective and training same network with our proposed KL-WGAN algorithm, where we add steps 8 to 11 in Algorithm 1. In addition, we also experimented with the original f-GAN with KL divergence; this failed to train properly due to numerical issues where exponents of very large critic values gives infinity values in the objective.

We report two common benchmarks for image generation, Inception scores [SGZ+16] and Fréchet Inception Distance (FID) [HRU+17] [1] in Table 6.3 (CIFAR10) and Table 6.4 (CelebA). We do not report inception score on CelebA since the real dataset only has a score of less than 3, so the score is not very indicative of generation performance [HRU+17]. We show generated samples from the model in Appendix B.4.4.

Despite the strong performance of BigGAN, our method is able to consistently achieve superior inception scores and FID scores consistently on all the datasets and across different random seeds. This demonstrates that the KL-WGAN algorithm is practically useful, and can serve as a viable drop-in replacement for the existing WGAN objective even on state-of-the-art GAN models, such as BigGAN.

6.7 Discussion

In this chapter, we introduce a generalization of f-GANs and WGANs based on optimizing a (regularized) objective over importance weighted samples. This perspective allows us to recover

[1] Based on https://github.com/mseitzer/pytorch-fid

both f-GANs and WGANs when different sets to optimize for the importance weights are considered. In addition, we show that this generalization leads to alternative practical objectives for training GANs and demonstrate its effectiveness on several different applications, such as distribution modeling, density ratio estimation and image generation. The proposed method only requires a small change in the original training algorithm and is easy to implement in practice.

In future work, we are interested in considering other constraints that could lead to alternative objectives and/or inequalities and their practical performances. It would also be interesting to investigate the KL-WGAN approaches on high-dimensional density ratio estimation tasks such as off-policy policy evaluation, inverse reinforcement learning and contrastive representation learning.

Chapter 7

Generation with Negative Data Augmentation

In the previous chapter, we discussed how we can incorporate reweighting to the supervised learning objective and improve GAN training. In this chapter, we consider another improvement that introduces certain data augmentations to the binary classification objective. Similar to traditional data augmentations in supervised learning, this allows us to improve generative modeling performance that cannot be achieved by the objective function alone.

Data augmentation is often used to enlarge datasets with synthetic samples generated in accordance with the underlying data distribution. To enable a wider range of augmentations, we explore *negative* data augmentation strategies (NDA) that intentionally create out-of-distribution samples. We show that such negative out-of-distribution samples provide information on the support of the data distribution, and can be leveraged for generative modeling and representation learning.

We introduce a new GAN training objective where we use NDA as an additional source of synthetic data for the discriminator. We prove that under suitable conditions, optimizing the resulting objective still recovers the true data distribution but can directly bias the generator towards avoiding samples that lack the desired structure. Empirically, models trained with our method achieve improved conditional/unconditional image generation along with improved anomaly detection capabilities. Further, we incorporate the same negative data augmentation strategy in a contrastive learning framework for self-supervised representation learning on images and videos, achieving improved performance on downstream image classification, object detection, and action recognition tasks. These results suggest that prior knowledge on what does not constitute valid

data is an effective form of weak supervision across a range of unsupervised learning tasks.

This chapter is based on [SKS+21]. Abhishek Sinha and Ayush Kumar contributed to the contents of this chapter. My contributions include providing the theory and initial code for the GAN and the contrastive learning setups.

7.1 Introduction

Data augmentation strategies for synthesizing new data in a way that is consistent with an underlying task are extremely effective in both supervised and unsupervised learning [vdOLV18, ZIE16, NF16, ARV19]. Because they operate at the level of samples, they can be combined with most learning algorithms. They allow for the incorporation of prior knowledge (inductive bias) about properties of typical samples from the underlying data distribution [JWAAN18, ASE17], e.g., by leveraging invariances to produce additional "positive" examples of how a task should be solved.

To enable users to specify an even wider range of inductive biases, we propose to leverage an alternative and complementary source of prior knowledge that specifies how a task should *not* be solved. We formalize this intuition by assuming access to a way of generating samples that are guaranteed to be out-of-support for the data distribution, which we call a *Negative Data Augmentation* (NDA). Intuitively, negative out-of-distribution (OOD) samples can be leveraged as a useful inductive bias because they provide information about the support of the data distribution to be learned by the model. For example, in a density estimation problem we can bias the model to avoid putting any probability mass in regions which we know a-priori should have zero probability. This can be an effective prior if the negative samples cover a sufficiently large area. The best NDA candidates are ones that expose common pitfalls of existing models, such as prioritizing local structure over global structure [GRM+18]; this motivates us to consider known transformations from the literature that intentionally destroy the spatial coherence of an image [NF16, DT17, YHO+19], such as Jigsaw transforms.

Building on this intuition, we introduce a new GAN training objective where we use NDA as an additional source of fake data for the discriminator as shown in Fig. 7.1. Theoretically, we can show that if the NDA assumption is valid, optimizing this objective will still recover the data distribution in the limit of infinite data. However, in the finite data regime, there is a need to generalize beyond the empirical distribution [ZRY+18]. By explicitly providing the discriminator with samples we want to avoid, we are able to bias the generator towards avoiding undesirable samples thus improving generation quality.

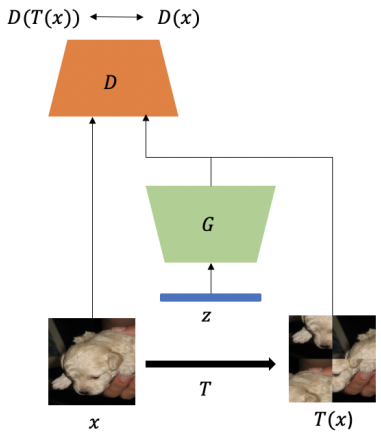


Figure 7.1: **Negative Data Augmentation for GANs.**

Furthermore, we propose a way of leveraging NDA for unsupervised representation learning. We propose a new contrastive predictive coding [HFW+19, HXZ19] (CPC) objective that encourages the distribution of representations corresponding to in-support data to become disjoint from that of NDA data. Empirically, we show that applying NDA with our proposed transformations (e.g., forcing the representation of normal and jigsaw images to be disjoint) improves performance in downstream tasks.

With appropriately chosen NDA strategies, we obtain superior empirical performance on a variety of tasks, with almost no cost in computation. For generative modeling, models trained with NDA achieve better image generation, image translation and anomaly detection performance compared with the same model trained without NDA. Similar gains are observed on representation learning for images and videos over downstream tasks such as image classification, object detection and action recognition. These results suggest that NDA has much potential to improve a variety of self-supervised learning techniques.

7.2 Negative Data Augmentation

The input to most learning algorithms is a dataset of samples from an underlying data distribution p_{data}. While p_{data} is unknown, learning algorithms always rely on prior knowledge about its properties (inductive biases [WM97]), e.g., by using specific functional forms such as neural networks. Similarly, data augmentation strategies exploit known invariances of p_{data}, such as the conditional label distribution being invariant to semantic-preserving transformations.

While typical data augmentation strategies exploit prior knowledge about what is in support of p_{data}, in this paper, we propose to exploit prior knowledge about what is *not* in the support of

p_{data}. This information is often available for common data modalities (e.g., natural images and videos) and is under-exploited by existing approaches. Specifically, we assume: (1) there exists an alternative distribution $\overline{p}$ such that its support is disjoint from that of p_{data}; and (2) access to a procedure to efficiently sample from $\overline{p}$. We emphasize $\overline{p}$ need not be explicitly defined (e.g., through an explicit density) – it may be implicitly defined by a dataset or by a procedure that transforms samples from p_{data} into ones from $\overline{p}$ by suitably altering their structure.

Figure 7.2: **Negative augmentations produce out-of-distribution samples.** Samples by negative augmentations lack the typical structure of natural images, and can be used to inform a model on what it should *not* learn.

Analogous to typical data augmentations, NDA strategies are by definition domain and task specific. In this paper, we focus on natural images and videos, and leave the application to other domains (such as natural language processing) as future work. How do we select a good NDA strategy? According to the manifold hypothesis [FMN16], natural images lie on low-dimensional manifolds: p_{data} is supported on a low-dimensional manifold of the ambient (pixel) space. This suggests that many negative data augmentation strategies exist. Indeed, sampling random noise is in most cases a valid NDA. However, while this prior is generic, it is not very informative, and this NDA will likely be ineffective for most learning problems. Intuitively, NDA is informative if its support is close (in a suitable metric) to that of p_{data}, while being disjoint. These negative samples will provide information on the "boundary" of the support of p_{data}, which we will show is helpful in several learning problems. In most of our tasks, the images are processed by convolutional neural networks (CNNs) that are good at processing local features but not necessarily global features [GRM+18]. Therefore, we may consider NDA examples to be ones that preserve local features ("informative") and break global features, so that it forces the CNNs to learn global features (by realizing NDAs are different from real data).

Leveraging this intuition, we show several image transformations from the literature that can be viewed as generic NDAs over natural images in Figure 7.2, that we will use for generative modeling and representation learning in the following sections. Details about these transformations can be found in Appendix B.5.2.

7.3 NDA for Generative Adversarial Networks

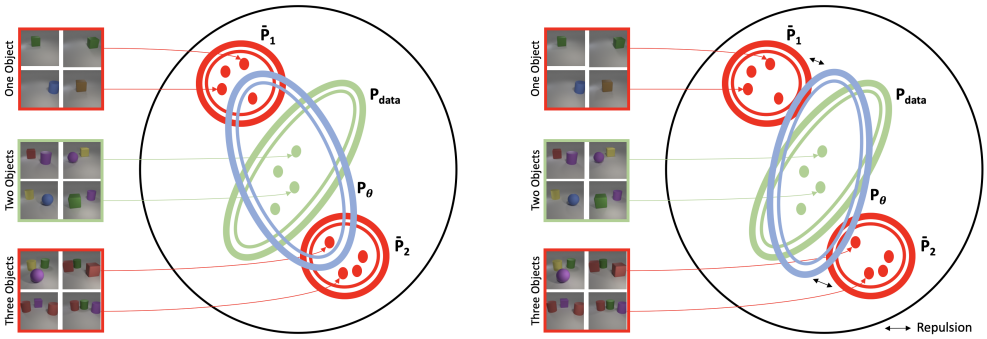


Figure 7.3: **Schematic overview of our NDA framework. Left**: In the absence of NDA, the support of a generative model P_θ (blue oval) learned from samples (green dots) may "over-generalize" and include samples from $\overline{P_1}$ or $\overline{P_2}$. **Right**: With NDA, the learned distribution P_θ becomes disjoint from NDA distributions $\overline{P_1}$ and $\overline{P_2}$, thus pushing P_θ closer to the true data distribution p_{data} (green oval). As long as the prior is consistent, i.e. the supports of $\overline{P_1}$ and $\overline{P_2}$ are truly disjoint from p_{data}, the best fit distribution in the infinite data regime does not change.

In GANs, we are interested in learning a generative model G_θ from samples drawn from some data distribution p_{data} [GPAM+14]. GANs use a binary classifier, the so-called discriminator D_ϕ, to distinguish real data from generated (fake) samples. The generator G_θ is trained via the following mini-max objective that performs variational Jensen-Shannon divergence minimization:

$$\min_{G_\theta \in \mathcal{P}(\mathcal{X})} \max_{D_\phi} L_{\text{JS}}(G_\theta, D_\phi) \quad \text{where} \tag{7.1}$$

$$L_{\text{JS}}(G_\theta, D_\phi) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} [\log(D_\phi(\boldsymbol{x}))] + \mathbb{E}_{\boldsymbol{x} \sim G_\theta} [\log(1 - D_\phi(\boldsymbol{x}))] \tag{7.2}$$

This is a special case to the more general variational f-divergence minimization objective [NCT16]. The optimal D_ϕ for any G_θ is $(p_{\text{data}}/G_\theta)/(1 + p_{\text{data}}/G_\theta)$, so the discriminator can serve as a density ratio estimator between p_{data} and G_θ.

With sufficiently expressive models and infinite capacity, G_θ will match p_{data}. In practice, however, we have access to finite datasets and limited model capacity. This means that the generator needs to generalize beyond the empirical distribution, which is challenging because the number of possible discrete distributions scale *doubly exponentially* w.r.t. to the data dimension. Hence, as studied in [ZRY+18], the role of the inductive bias is critical. For example, [ZRY+18] report that when trained on images containing 2 objects only, GANs and other generative models can sometimes "generalize" by generating images with 1 or 3 objects (which were never seen in the

training set). The generalization behavior – which may or may not be desirable – is determined by factors such as network architectures, hyperparameters, etc., and is difficult to characterize analytically.

Here we propose to bias the learning process by directly specifying what the generator should *not* generate through NDA. We consider an adversarial game based on the following objective:

$$\min_{G_\theta \in \mathcal{P}(\mathcal{X})} \max_{D_\phi} L_{\mathrm{JS}}(\lambda G_\theta + (1-\lambda)\overline{P}, D_\phi) \tag{7.3}$$

where the negative samples are generated from a mixture of G_θ (the generator distribution) and $\overline{P}$ (the NDA distribution); the mixture weights are controlled by the hyperparameter λ. Intuitively, this can help addresses the above "over-generalization" issue, as we can directly provide supervision on what should not be generated and thus guide the support of G_θ (see Figure 7.3) . For instance, in the object count example above, we can empirically prevent the model from generating images with an undesired number of objects (see Appendix Section A for experimental results on this task).

In addition, the introduction of NDA samples will not affect the solution of the original GAN objective in the limit. In the following theorem, we show that given infinite training data and infinite capacity discriminators and generators, using NDA will not affect the optimal solution to the generator, *i.e.* the generator will still recover the true data distribution.

Theorem 10. *Let $\overline{P} \in \mathcal{P}(\mathcal{X})$ be any distribution over $\mathcal{X}$ with disjoint support than p_{data}, i.e., such that $\mathrm{supp}(p_{\mathrm{data}}) \cap \mathrm{supp}(\overline{P}) = \varnothing$. Let $D_\phi : \mathcal{X} \to \mathbb{R}$ be the set of all discriminators over $\mathcal{X}$, $f : \mathbb{R}_{\geq 0} \to \mathbb{R}$ be a convex, semi-continuous function such that $f(1) = 0$, $f^\star$ be the convex conjugate of f, f' its derivative, and G_θ be a distribution with sample space $\mathcal{X}$. Then $\forall \lambda \in (0, 1]$, we have:*

$$\arg\min_{G_\theta \in \mathcal{P}(\mathcal{X})} \max_{D_\phi : \mathcal{X} \to \mathbb{R}} L_f(G_\theta, D_\phi) = \arg\min_{G_\theta \in \mathcal{P}(\mathcal{X})} \max_{D_\phi : \mathcal{X} \to \mathbb{R}} L_f(\lambda G_\theta + (1-\lambda)\overline{P}, D_\phi) = p_{\mathrm{data}} \tag{7.4}$$

where $L_f(Q, D_\phi) = \mathbb{E}_{\boldsymbol{x} \sim p_{\mathrm{data}}}[D_\phi(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x} \sim Q}[f^\star(D_\phi(\boldsymbol{x}))]$ is the objective for f-GAN [NCT16]. However, the optimal discriminators are different for the two objectives:

$$\arg\max_{D_\phi : \mathcal{X} \to \mathbb{R}} L_f(G_\theta, D_\phi) = f'(p_{\mathrm{data}}/G_\theta) \tag{7.5}$$

$$\arg\max_{D_\phi : \mathcal{X} \to \mathbb{R}} L_f(\lambda G_\theta + (1-\lambda)\overline{P}, D_\phi) = f'(p_{\mathrm{data}}/(\lambda G_\theta + (1-\lambda)\overline{P})) \tag{7.6}$$

Proof. See Appendix A.5.1. □

The above theorem shows that in the limit of infinite data and computation, adding NDA changes the optimal discriminator solution but not the optimal generator. In practice, when dealing with finite data, existing regularization techniques such as weight decay and spectral normalization [MKKY18] allow potentially many solutions that achieve the same objective value. The introduction of NDA samples allows us to filter out certain solutions by providing additional inductive bias through OOD samples. In fact, the optimal discriminator will reflect the density ratio between p_{data} and $\lambda G_\theta + (1-\lambda)\overline{P}$ (see Eq.(7.6)), and its values will be higher for samples from p_{data} compared to those from $\overline{P}$. As we will show in Section 7.5, a discriminator trained with this objective and suitable NDA performs better than relevant baselines for other downstream tasks such as anomaly detection.

7.4 NDA for Constrastive Representation Learning

Using a classifier to estimate a density ratio is useful not only for estimating f-divergences (as in the previous section) but also for estimating mutual information between two random variables. In representation learning, mutual information (MI) maximization is often employed to learn compact yet useful representations of the data, allowing one to perform downstream tasks efficiently [TZ15, NWJ08, POvdO+19, vdOLV18]. Here, we show that NDA samples are also beneficial for representation learning.

In contrastive representation learning (such as CPC [vdOLV18]), the goal is to learn a mapping $h_\theta(\boldsymbol{x}) : \mathcal{X} \to \mathcal{P}(\mathcal{Z})$ that maps a datapoint $\boldsymbol{x}$ to some distribution over the representation space $\mathcal{Z}$; once the network h_θ is learned, representations are obtained by sampling from $\boldsymbol{z} \sim h_\theta(\boldsymbol{x})$. CPC *maximizes* the following objective:

$$I_{\text{CPC}}(h_\theta, g_\phi) := \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x}),\boldsymbol{z}\sim h_\theta(\boldsymbol{x}),\widehat{\boldsymbol{z}}_i\sim p_\theta(\boldsymbol{z})}\left[\log \frac{n g_\phi(\boldsymbol{x},\boldsymbol{z})}{g_\phi(\boldsymbol{x},\boldsymbol{z}) + \sum_{j=1}^{n-1} g_\phi(\boldsymbol{x},\widehat{\boldsymbol{z}}_j)}\right] \tag{7.7}$$

where $p_\theta(\boldsymbol{z}) = \int h_\theta(\boldsymbol{z}|\boldsymbol{x}) p_{\text{data}}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}$ is the marginal distribution of the representations associated with p_{data}. Intuitively, the CPC objective involves an n-class classification problem where g_ϕ attempts to identify a matching pair (i.e. $(\boldsymbol{x}, \boldsymbol{z})$) sampled from the joint distribution from the $(n-1)$ non-matching pairs (i.e. $(\boldsymbol{x}, \widehat{\boldsymbol{z}}_j)$) sampled from the product of marginals distribution. Note that g_ϕ plays the role of a discriminator/critic, and is implicitly estimating a density ratio. As $n \to \infty$, the optimal g_ϕ corresponds to an un-normalized density ratio between the joint distribution and the product of marginals, and the CPC objective matches its upper bound which is

the mutual information between X and Z [POO+19, SE19b]. However, this objective is no longer able to control the representations for data that are out of support of p_{data}, so there is a risk that the representations are similar between p_{data} samples and out-of-distribution ones.

To mitigate this issue, we propose to use NDA in the CPC objective, where we additionally introduce a batch of NDA samples, for each positive sample:

$$\overline{I_{\text{CPC}}}(h_\theta, g_\phi) := \mathbb{E}\left[\log \frac{(n+m)g_\phi(\boldsymbol{x}, \boldsymbol{z})}{g_\phi(\boldsymbol{x}, \boldsymbol{z}) + \sum_{j=1}^{n-1} g_\phi(\boldsymbol{x}, \widehat{\boldsymbol{z}}_j) + \sum_{k=1}^{m} g_\phi(\boldsymbol{x}, \overline{\boldsymbol{z}}_k)}\right] \tag{7.8}$$

where the expectation is taken over $\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x}), \boldsymbol{z} \sim h_\theta(\boldsymbol{x}), \widehat{\boldsymbol{z}}_i \sim p_\theta(\boldsymbol{z}), \overline{\boldsymbol{x}}_k \sim \overline{p}$ (NDA distribution), $\overline{\boldsymbol{z}}_k \sim h_\theta(\overline{\boldsymbol{x}}_k)$ for all $k \in [m]$. Here, the behavior of $h_\theta(\boldsymbol{x})$ when $\boldsymbol{x}$ is NDA is optimized explicitly, allowing us to impose additional constraints to the NDA representations. This corresponds to a more challenging classification problem (compared to basic CPC) that encourages learning more informative representations. In the following theorem, we show that the proposed objective encourages the representations for NDA samples to become disjoint from the representations for p_{data} samples, *i.e.* NDA samples and p_{data} samples do not map to the same representation.

Theorem 11. *(Informal) The optimal solution to h_θ in the NDA-CPC objective maps the representations of data samples and NDA samples to disjoint regions.*

Proof. See Appendix A.5.2 for a detailed statement and proof. □

7.5 NDA-GAN Experiments

In this section we report experiments with different types of NDA for image generation. Additional details about the network architectures and hyperparameters can be found in Appendix B.5.7.

Unconditional Image Generation. We conduct experiments on various datasets using the BigGAN architecture [BDS18] for unconditional image generation[1]. We first explore various image transformations from the literature to evaluate which ones are effective as NDA. For each transformation, we evaluate its performance as NDA (training as in Eq. 7.3) and as a traditional data augmentation strategy, where we enlarge the training set by applying the transformation to real images (denoted PDA for positive data augmentation). Table 7.1 shows the FID scores for different types of transformations as PDA/NDA. The results suggest that transformations that

[1]We feed a single label to all images to make the architecture suitable for unconditional generation.

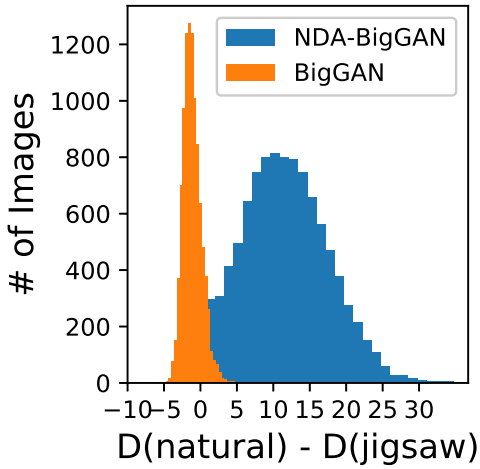


Figure 7.4: **Discriminator outputs for images.** Histogram of difference in the discriminator output for a real image and it's Jigsaw version. Without NDA training, the discriminator has trouble differentiating real or NDA samples.

spatially corrupt the image are strong NDA candidates. It can be seen that Random Horizontal Flip is not effective as an NDA; this is because flipping does not spatially corrupt the image but is rather a semantic preserving transformation, hence the NDA distribution $\overline{P}$ is not disjoint from p_{data}. On the contrary, it is reasonable to assume that if an image is likely under p_{data}, its flipped variant should also be likely. This is confirmed by the effectiveness of this strategy as PDA.

We believe spatially corrupted negatives perform well as NDA in that they push the discriminator to focus on global features instead of local ones (e.g., texture). We confirm this by plotting the histogram of differences in the discriminator output for a real image and it's Jigsaw version as shown in Fig. 7.4. We show that the difference is *(a)* centered close to zero for normal BigGAN (so *without NDA training, the discriminator cannot distinguish real and Jigsaw samples well*), and *(b)* centered at a positive number (logit 10) for our method (NDA-BigGAN). Following our findings, in our remaining experiments we use Jigsaw, Cutout, Stitch, Mixup and Cutmix as they achieve significant improvements when used as NDA for unconditional image generation on CIFAR-10.

Table 7.2 shows the FID scores for BigGAN when trained with five types of negative data augmentation on four different benchmarks. Almost all the NDA augmentations improve the baseline across datasets. For all the datasets except CIFAR-100, $\lambda = 0.25$, whereas for CIFAR-100 it is 0.5. We show the effect of λ on CIFAR-10 performance in Appendix B.5.6. We additionally performed an experiment using a mixture of augmentation policy. The results (FID 16.24) were

Table 7.1: **FID scores over CIFAR-10**, using different transformations as PDA and NDA in BigGAN. The results indicate that some transformations yield better results when used as NDA. The common feature of such transformations is they all spatially corrupt the images.

w/o Aug.	Jigsaw		Cutout		Stitch		Mixup		Cutmix		Random Crop		Random Flip		Gaussian	
	PDA	NDA	PDA	NDA	PDA	NDA	PDA	NDA	PDA	NDA	PDA	NDA	PDA	NDA	PDA	NDA
18.64	98.09	12.61	79.72	14.69	108.69	13.97	70.64	17.29	90.81	15.01	20.02	15.05	16.65	124.32	44.41	18.72

Table 7.2: **Comparison of FID scores of different types of NDA**, for unconditional image generation on various datasets. The numbers in bracket represent the corresponding image resolution in pixels. Jigsaw consistently achieves the **best** or second best result.

	BigGAN	Jigsaw	Stitching	Mixup	Cutout	Cutmix	CR-BigGAN
CIFAR-10 (32)	18.64	**12.61**	13.97	17.29	14.69	15.01	14.56
CIFAR-100 (32)	22.19	**19.72**	20.99	22.21	22.08	20.78	–
CelebA (64)	38.14	37.24	**37.17**	37.51	37.39	37.46	–
STL10 (32)	26.80	**23.94**	26.08	24.45	24.91	25.34	–

better than the baseline method (18.64) but not as good as using a single strategy.

Conditional Image Generation. We also investigate the benefits of NDA in conditional image generation using BigGAN. The results are shown in Table 7.3. In this setting as well, NDA gives a significant boost over the baseline model. We again use $\lambda = 0.25$ for CIFAR-10 and $\lambda = 0.5$ for CIFAR-100. For both unconditional and conditional setups we find the Jigsaw and Stitching augmentations to achieve a better FID score than the other augmentations.

Table 7.3: **FID scores for conditional image generation using different NDAs.**[2]

	BigGAN	Jigsaw	Stitching	Mixup	Cutout	Cutmix	CR-BigGAN
C-10	11.51	**9.42**	9.47	13.87	10.52	10.3	11.48
C-100	15.04	14.12	**13.90**	15.27	14.21	13.99	–

Image Translation. Next, we apply the NDA method to image translation. In particular, we use the Pix2Pix model [IZZE17] that can perform image-to-image translation using GANs provided paired training data. Here, the generator is conditioned on an image $\mathcal{I}$, and the discriminator takes as input the concatenation of generated/real image and $\mathcal{I}$. We use Pix2Pix for semantic segmentation on Cityscapes dataset [COR+16] (i.e. photos $\rightarrow$ labels). Table 7.4 shows the quantitative gains obtained by using Jigsaw NDA[3] while Figure B.7 in Appendix B.5.4 highlights the qualitative

[2]We use a PyTorch code for BigGAN. The number reported in [BDS18] for C-10 is 14.73.
[3]We use the official PyTorch implementation and show the best results.

improvements. The NDA-Pix2Pix model avoids noisy segmentation on objects including buildings and trees.

Table 7.4: **Results on CityScapes**, using per pixel accuracy (Pp.), per class accuracy (Pc.) and mean Intersection over Union (mIOU). We compare Pix2Pix and its NDA version.

Metric	Pp.	Pc.	mIOU
Pix2Pix (cGAN)	0.80	0.24	0.27
NDA (cGAN)	**0.84**	**0.34**	**0.28**
Pix2Pix (L1+cGAN)	0.72	0.23	0.18
NDA (L1+cGAN)	**0.75**	**0.28**	**0.22**

Table 7.5: **AUROC scores for different OOD datasets.** OOD-1 contains different datasets, while OOD-2 contains the set of 19 different corruptions in CIFAR-10-C [HD18] (the average score is reported).

		BigGAN	Jigsaw	EBM
OOD-1	DTD	0.70	0.69	0.48
	SVHN	0.75	0.61	0.63
	Places-365	0.35	0.58	0.68
	TinyImageNet	0.40	0.62	0.67
	CIFAR-100	0.63	0.64	0.50
	Average	0.57	**0.63**	0.59
OOD-2	CIFAR-10-C	0.56	**0.63**	0.60

Anomaly Detection. As another added benefit of NDA for GANs, we utilize the output scores of the BigGAN discriminator for anomaly detection. We experiment with 2 different types of OOD datasets. The first set consists of SVHN [NWC+11], DTD [CMK+14], Places-365 [ZLK+17], TinyImageNet, and CIFAR-100 as the OOD datapoints following the protocol in [DM19, HMD18]. We train BigGAN w/ and w/o Jigsaw NDA on the train set of CIFAR-10 and then use the output value of discriminator to classify the test set of CIFAR-10 (not anomalous) and different OOD datapoints (anomalous) as anomalous or not. We use the AUROC metric as proposed in [HG16] to evaluate the anomaly detection performance. Table 7.5 compares the performance of NDA with a likelihood based model (Energy Based Models (EBM [DM19]). Results show that Jigsaw NDA performs much better than baseline BigGAN and other generative models. We did not include other NDAs as Jigsaw achieved the best results.

We consider the extreme corruptions in CIFAR-10-C [HD18] as the second set of OOD datasets. It consists of 19 different corruptions, each having 5 different levels of severity. We only consider the corruption of highest severity for our experiment, as these constitute a significant shift from the true data distribution. Averaged over all the 19 different corruptions, the AUROC score for the normal BigGAN is **0.56**, whereas the BigGAN trained with Jigsaw NDA achieves **0.63**. The histogram of difference in discriminator's output for clean and OOD samples are shown in Figure B.8 in the appendix. High difference values imply that the Jigsaw NDA is better at distinguishing OOD samples than the normal BigGAN.

7.6 Representation Learning using Contrastive Loss and NDA

Unsupervised Learning on Images. In this section, we perform experiments on three benchmarks: (a) CIFAR10 (C10), (b) CIFAR100 (C100), and (c) ImageNet-100 [DDS+09] to show the benefits of NDA on representation learning with the contrastive loss function. In our experiments, we use the momentum contrast method [HFW+19], *MoCo-V2*, as it is currently the state-of-the-art model on unsupervised learning on ImageNet. For C10 and C100, we train the MoCo-V2 model for unsupervised learning (w/ and w/o NDA) for 1000 epochs. On the other hand, for ImageNet-100, we train the MoCo-V2 model (w/ and w/o NDA) for 200 epochs. To evaluate the representations, we train a linear classifier on the representations on the same dataset with labels. Table 7.6 shows the top-1 accuracy of the classifier. We find that across all the three datasets, different NDA approaches outperform *MoCo-V2*. While Cutout NDA performs the best for C10, the best performing NDA for C100 and ImageNet-100 are Jigsaw and Mixup respectively. Figure B.9 compares the cosine distance of the representations learned w/ and w/o NDA (jigsaw) and shows that jigsaw and normal images are projected far apart from each other when trained using NDA whereas with original MoCo-v2 they are projected close to each other.

Table 7.6: Top-1 accuracy results on image recognition w/ and w/o NDA on MoCo-V2.

	MoCo-V2	Jigsaw	Stitching	Cutout	Cutmix	Mixup
CIFAR-10	91.20	91.66	91.59	**92.26**	91.51	91.36
CIFAR-100	69.63	**70.17**	69.21	69.81	69.83	69.99
ImageNet-100	69.41	69.95	69.54	69.77	69.61	**70.01**

Transfer Learning for Object Detection. We transfer the network pre-trained over ImageNet-100 for the task of Pascal-VOC object detection using a Faster R-CNN detector (C4 backbone) [RHGS15]. We fine-tune the network on Pascal VOC 2007+2012 trainval set and test it on the 2007 test set. The baseline MoCo achieves 38.47 AP, 65.99 AP50, 38.81 AP75 whereas the MoCo trained with mixup NDA gets 38.72 AP, 66.23 AP50, 39.16 AP75 (an improvement of ≈ 0.3).

Unsupervised Learning on Videos. In this section, we investigate the benefits of NDA in self-supervised learning of spatio-temporal embeddings from video, suitable for human action recognition. We apply NDA to Dense Predictive Coding [HXZ19], which is a single stream (RGB only) method for self-supervised representation learning on videos. For videos, we create NDA samples by performing the same transformation on all frames of the video (e.g. the same jigsaw permutation is applied to all the frames of a video). We evaluate the approach by first training the

DPC model with NDA on a large-scale dataset (UCF101), and then evaluate the representations by training a supervised action classifier on UCF101 and HMDB51 datasets. As shown in Table 7.7, Jigsaw and Cutmix NDA improve downstream task accuracy on UCF-101 and HMDB-51, achieving new state-of-the-art performance among single stream (RGB only) methods for self-supervised representation learning (when pre-trained using UCF-101).

Table 7.7: Top-1 accuracy results on action recognition in videos w/ and w/o NDA in DPC.

	DPC	Jigsaw	Stitching	Cutout	Cutmix	Mixup
UCF-101 (Pre-trained on UCF-101)	61.35	64.54	**66.07**	64.52	63.52	63.65
HMDB51 (Pre-trained on UCF-101)	45.31	**46.88**	45.31	45.31	**48.43**	43.75

7.7 Related work

In several machine learning settings, negative samples are produced from a statistical generative model. [SHPL19] aim to generate negative data using GANs for semi-supervised learning and novelty detection while we are concerned with efficiently creating negative data to improve generative models and self-supervised representation learning. [HKKT18] also propose an alternative theoretical framework that relies on access to an oracle which classifies a sample as valid or not, but do not provide any practical implementation. [BLC18] use adversarial training to generate hard negatives that fool the discriminator for NLP tasks whereas we obtain NDA data from positive data to improve image generation and representation learning. [HCDLZ18] use a GAN to learn the negative data distribution with the aim of classifying positive-unlabeled (PU) data whereas we do not have access to a mixture data but rather generate negatives by transforming the positive data.

In contrastive unsupervised learning, common negative examples are ones that are assumed to be further than the positive samples semantically. Word2Vec [MSC+13] considers negative samples to be ones from a different context and CPC-based methods [vdOLV18] such as momentum contrast [HFW+19], the negative samples are data augmentations from a different image. Our work considers a new aspect of "negative samples" that are neither generated from some model, nor samples from the data distribution. Instead, by applying negative data augmentation (NDA) to existing samples, we are able to incorporate useful inductive biases that might be difficult to capture otherwise [ZRY+18].

7.8 Discussion

We proposed negative data augmentation as a method to incorporate prior knowledge through out-of-distribution (OOD) samples. NDAs are complementary to traditional data augmentation strategies, which are typically focused on in-distribution samples. Using the NDA framework, we interpret existing image transformations (e.g., jigsaw) as producing OOD samples and develop new learning algorithms to leverage them. Owing to rigorous mathematical characterization of the NDA assumption, we are able to theoretically analyze their properties. As an example, we bias the generator of a GAN to avoid the support of negative samples, improving results on conditional/unconditional image generation tasks. Finally, we leverage NDA for unsupervised representation learning in images and videos. By integrating NDA into MoCo-v2 and DPC, we improve results on image and action recognition on CIFAR10, CIFAR100, ImageNet-100, UCF-101, and HMDB-51 datasets.

Chapter 8

Generation with Learned Conditions

In the previous chapters, we focused on improving the supervised learning pipeline for generative adversarial networks (GANs), where the supervised learning procedure is tied more closely to generation quality. However, unconditional image generation does not highlight all the tasks that can be useful; for example, there can be cases where we wish to generate from certain conditions. Conditional generative models of high-dimensional images have many applications, but supervision signals from conditions to images can be expensive to acquire; this makes it difficult to train conditional generative models from input-label pairs.

In this chapter, we describe Diffusion-Decoding models with Contrastive representations (D2C), a paradigm for training unconditional variational autoencoders (VAEs) for few-shot conditional image generation. D2C uses a learned diffusion-based prior over the latent representations to improve generation and contrastive self-supervised learning to improve representation quality. D2C can adapt to novel generation tasks conditioned on labels or manipulation constraints, by learning from as few as 100 labeled examples.

On conditional generation from new labels, D2C achieves superior performance over state-of-the-art VAEs and diffusion models. On conditional image manipulation, D2C generations are two orders of magnitude faster to produce over StyleGAN2 ones and are preferred by $50\% - 60\%$ of the human evaluators in a double-blind study.

This chapter is based on [SSME21] with materials from [SME20]. Abhishek Sinha and Chenlin Meng contributed significantly to this chapter. My contributions include conceving the idea, implementing the diffusion model, executed most of the experiments on CIFAR-10/100 and image manipulation experiments and writing the chapter.

8.1 Introduction

Generative models trained on large amounts of unlabeled data have achieved great success in various domains including images [BDS18, KLA+20, ROV19, HJA20], text [LGL+20, AABS19], audio [DJP+20, PPZS20, vdODZ+16, MEHS21], and graphs [GZE19, NSS+20]. However, downstream applications of generative models are often based on various conditioning signals, such as labels [MO14], text descriptions [MPBS15], reward values [YLY+18], or similarity with existing data [IZZE17]. While it is possible to directly train conditional models, this often requires large amounts of paired data [LMB+14, RPG+21] that are costly to acquire. Hence, it would be desirable to learn conditional generative models using large amounts of unlabeled data and as little paired data as possible.

Contrastive self-supervised learning (SSL) methods can greatly reduce the need for labeled data in discriminative tasks by learning effective representations from unlabeled data [vdOLV18, HFW+19, GSA+20], and have also been shown to improve few-shot learning [Hen20]. It is therefore natural to ask if they can also be used to improve few-shot generation. Latent variable generative models (LVGM) are a natural candidate for this, since they already involve a low-dimensional, structured latent representation of the data they generate. However, popular LVGMs, such as generative adversarial networks (GANs, [GPAM+14, KLA+20]) and diffusion models [HJA20, SME20], lack explicit tractable functions to map inputs to representations, making it difficult to optimize latent variables with SSL. Variational autoencoders (VAEs, [KW13, RM15]), on the other hand, can naturally adopt SSL through their encoder model, but they typically have worse sample quality.

In this paper, we propose Diffusion-Decoding models with Contrastive representations (D2C), a special VAE that is suitable for conditional few-shot generation. D2C uses contrastive self-supervised learning methods to obtain a latent space that inherits the transferrability and few-shot capabilities of self-supervised representations. Unlike other VAEs, D2C learns a diffusion model over the latent representations. This latent diffusion model ensures that D2C uses the same latent distribution for both training and generation. We provide a formal argument to explain why this approach may lead to better sample quality than existing hierarchical VAEs. We further discuss how to apply D2C to few-shot conditional generation where the conditions are defined through labeled examples and/or manipulation constraints. Our approach combines a discriminative model providing conditioning signal and generative diffusion model over the latent space, and is computationally more efficient than methods that act directly over the image space (Figure 8.1).

We evaluate and compare D2C with several state-of-the-art generative models over 6 datasets.

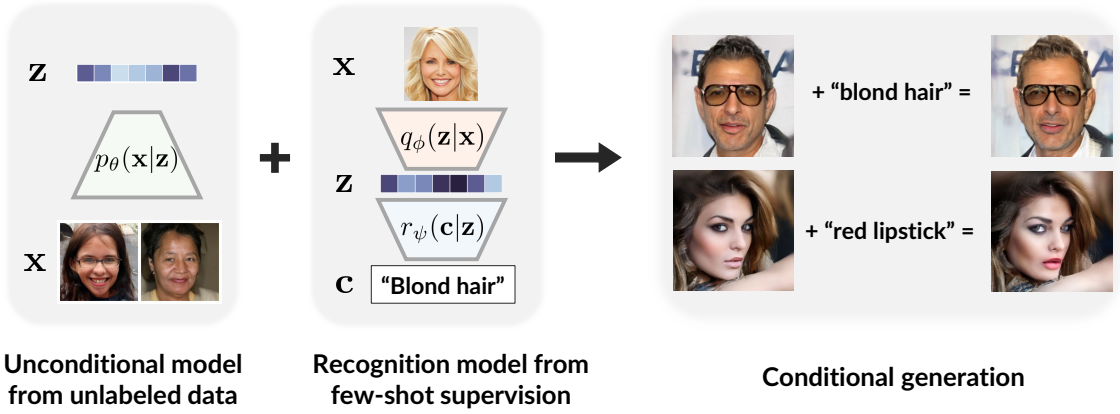


Figure 8.1: Few-shot conditional generation with the unconditional D2C model (left). With a recognition model over the latent space (middle), D2C can generate samples for novel conditions, such as image manipulation (right). These conditions can be defined with very few labels.

On unconditional generation, D2C outperforms state-of-the-art VAEs and is competitive with diffusion models under similar computational budgets. On conditional generation with 100 labeled examples, D2C significantly outperforms state-of-the-art VAE [VK20] and diffusion models [SME20]. D2C can also learn to perform certain image manipulation tasks from as few as 100 labeled examples. Notably, for manipulating images, D2C is two orders of magnitude faster than StyleGAN2 [ZSZZ20] and preferred by $50\% - 60\%$ of human evaluations, which to our best knowledge is the first for any VAE model.

8.2 Background

Self-supervised learning of representations In self-supervised learning (SSL), representations are learned by completing certain pretext tasks that do not require extra manual labeling [NF16, DCLT18]; these representations can then be applied to other downstream tasks, often in few-shot or zero-shot scenarios. In particular, contrastive representation learning encourages representations to be closer between "positive" pairs and further between "negative" pairs; contrastive predictive coding (CPC, [vdOLV18]), based on multi-class classification, have been commonly used in state-of-the-art methods [HFW+19, CFGH20, CXH21, CKNH20, SE20b]. Other non-contrastive methods exist, such as BYOL [GSA+20] and SimSiam [CH20], but they usually require additional care to prevent the representation network from collapsing.

8.3 Problem Statement

Few-shot conditional generation Our goal is to learn an unconditional generative model $p_\theta(\mathbf{x})$ such that it is suitable for conditional generation. Let $\mathcal{C}(\mathbf{x}, \mathbf{c}, f)$ describe an event that "$f(\mathbf{x}) = \mathbf{c}$", where $\mathbf{c}$ is a property value and $f(\mathbf{x})$ is a property function that is unknown at training. In conditional generation, our goal is to sample $\mathbf{x}$ such that the event $\mathcal{C}(\mathbf{x}, \mathbf{c}, f)$ occurs for a chosen $\mathbf{c}$. If we have access to some "ground-truth" model that gives us $p(\mathcal{C}|\mathbf{x}) := p(f(\mathbf{x}) = \mathbf{c}|\mathbf{x})$, then the conditional model can be derived from Bayes' rule: $p_\theta(\mathbf{x}|\mathcal{C}) \propto p(\mathcal{C}|\mathbf{x})p_\theta(\mathbf{x})$. These properties $\mathbf{c}$ include (but are not limited to[1]) labels [MO14], text descriptions [MPBS15, RAY+16], noisy or partial observations [CRT06, AAH19, KS20, DDJD21], and manipulation constraints [PZW+20]. In many cases, we do not have direct access to the true $f(\mathbf{x})$, so we need to learn an accurate model from labeled data [BV18] (*e.g.*, $(\mathbf{c}, \mathbf{x})$ pairs).

Desiderata Many existing methods are optimized for some known condition (*e.g.*, labels in conditional GANs [BDS18]) or assume abundant pairs between images and conditions that can be used for pretraining (*e.g.*, DALL-E [RPG+21] and CLIP [RKH+21] over image-text pairs). Neither is the case in this paper, as we do not expect to train over paired data.

While high-quality latent representations are not essential to unconditional image generation (*e.g.*, autoregressive [vdOKK16], energy-based [DM19], and some diffusion models [HJA20]), they can be beneficial when we wish to specify certain conditions with limited supervision signals, similar to how SSL representations can reduce labeling efforts in downstream tasks. A compelling use case is detecting and removing biases in datasets via image manipulation, where we should not only address well-known biases a-priori but also address other hard-to-anticipate biases, adapting to societal needs [Naj20].

Therefore, a desirable generative model should not only have high sample quality but also contain informative latent representations.

8.4 Denoising Diffusion Implicit Models

Diffusion models [SDWMG15, HJA20, SME20] produce samples by reversing a Gaussian diffusion process. We use the index $\alpha \in [0, 1]$ to denote the particular noise level of an noisy observation $\mathbf{x}^{(\alpha)} = \sqrt{\alpha}\mathbf{x} + \sqrt{1-\alpha}\epsilon$, where $\mathbf{x}$ is the clean observation and $\epsilon \sim \mathcal{N}(0, \boldsymbol{I})$ is a standard Gaussian distribution; as $\alpha \to 0$, the distribution of $\mathbf{x}^{(\alpha)}$ converges to $\mathcal{N}(0, \boldsymbol{I})$. Diffusion models are typically

[1] When $\mathcal{C}$ refers to an event that is always true, we recover unconditioned generation.

Table 8.1: A comparison of several common paradigms for generative modeling. [Explicit $\mathbf{x} \to \mathbf{z}$]: the mapping from $\mathbf{x}$ to $\mathbf{z}$ is directly trainable, which enables SSL; [No prior hole]: latent distributions used for generation and training are identical (Sec. 8.5.2), which improves generation; [Non-adversarial]: training procedure does not involve adversarial optimization, which improves training stability.

Model family	Explicit $\mathbf{x} \to \mathbf{z}$ (Enables SSL)	No prior hole (Better generation)	Non-Adversarial (Stable training)
VAE [KW13, RM15], NF [DKB14]	✔	✗	✔
GAN [GPAM+14]	✗	✔	✗
BiGAN [DKD16, DBP+16]	✔	✔	✗
DDIM [SME20]	✗	✔	✔
D2C	✔	✔	✔

parametrized as reverse noise models $\epsilon_\theta(\mathbf{x}^{(\alpha)}, \alpha)$ that predict the noise component of $\mathbf{x}^{(\alpha)}$ given a noise level α, and trained to minimize $\|\epsilon_\theta(\mathbf{x}^{(\alpha)}, \alpha) - \epsilon\|_2^2$, the mean squared error loss between the true noise and predicted noise. Given any non-increasing series $\{\alpha_i\}_{i=0}^T$ between 0 and 1, the diffusion objective for a clean sample from the data $\mathbf{x}$ is:

$$\ell_{\text{diff}}(\mathbf{x}; w, \theta) := \sum_{i=1}^{T} w(\alpha_i) \mathbb{E}_{\epsilon \sim \mathcal{N}(0, \boldsymbol{I})}[\|\epsilon - \epsilon_\theta(\mathbf{x}^{(\alpha_i)}, \alpha_i)\|_2^2], \quad \mathbf{x}^{(\alpha_i)} := \sqrt{\alpha_i}\mathbf{x} + \sqrt{1 - \alpha_i}\epsilon \tag{8.1}$$

where $w : \{\alpha_i\}_{i=1}^T \to \mathbb{R}_+$ controls the loss weights for each α. When $w(\alpha) = 1$ for all α, we recover the denoising score matching objective for training score-based generative models [SE19c].

Given an initial sample $\mathbf{x}_0 \sim \mathcal{N}(0, \boldsymbol{I})$, diffusion models acquires clean samples (*i.e.*, samples of $\mathbf{x}_1$) through a gradual denoising process, where samples with reducing noise levels α are produced (*e.g.*, $\mathbf{x}_0 \to \mathbf{x}_{0.3} \to \mathbf{x}_{0.7} \to \mathbf{x}_1$). In particular, Denoising Diffusion Implicit Models (DDIMs, [SME20]) uses an Euler discretization of some neural ODE [CRBD18] to produce samples (Figure 8.2, left).

For conciseness, we use the notation $p^{(\alpha)}(\mathbf{x}^{(\alpha)})$ to denote the marginal distribution of $\mathbf{x}^{(\alpha)}$ under the diffusion model, and $p^{(\alpha_1, \alpha_2)}(\mathbf{x}^{(\alpha_2)} \mid \mathbf{x}^{(\alpha_1)})$ to denote the diffusion sampling process from $\mathbf{x}^{(\alpha_1)}$ to $\mathbf{x}^{(\alpha_2)}$ (assuming $\alpha_1 < \alpha_2$). This notation abstracts away the exact sampling procedure of the diffusion model, which depends on choices of α.

8.4.1 Training diffusion models

We use the notations in [SME20] to denote the α values and consider the forward diffusion model in [HJA20]; a non-Markovian version that motivates other sampling procedures can be found in [SME20], but the training procedure is largely identical. We refer to the reader to these two papers for more details.

First, we define the following diffusion forward process for a series $\{\alpha_t\}_{t=0}^T$:

$$q(\mathbf{x}^{(\alpha_{1:T})}|\mathbf{x}^{(\alpha_0)}) := \prod_{t=1}^{T} q(\mathbf{x}^{(\alpha_t)}|\mathbf{x}^{(\alpha_{t-1})}), \tag{8.2}$$

$$q(\mathbf{x}^{(\alpha_t)}|\mathbf{x}^{(\alpha_{t-1})}) := \mathcal{N}\left(\sqrt{\frac{\alpha_t}{\alpha_{t-1}}}\mathbf{x}^{(\alpha_{t-1})}, \left(1 - \frac{\alpha_t}{\alpha_{t-1}}\right)\boldsymbol{I}\right), \tag{8.3}$$

and from standard derivations for Gaussian we have that:

$$q(\mathbf{x}^{(\alpha_{t-1})}|\mathbf{x}^{(\alpha_t)}, \mathbf{x}^{(\alpha_0)}) = \mathcal{N}\Bigg(\underbrace{\frac{\sqrt{\alpha_{t-1} - \alpha_t}}{1-\alpha_t}\mathbf{x}^{(\alpha_0)} + \frac{\alpha_t(1-\alpha_{t-1})}{\alpha_{t-1}(1-\alpha_t)}\mathbf{x}^{(\alpha_t)}}_{\tilde{\mu}(\mathbf{x}^{(\alpha_t)}, \mathbf{x}^{(\alpha_0)}; \alpha_t, \alpha_{t-1})}, \frac{1-\alpha_{t-1}}{1-\alpha_t}\left(1 - \frac{\alpha_t}{\alpha_{t-1}}\right)\boldsymbol{I}\Bigg). \tag{8.4}$$

As a variational approximation to the above, [HJA20] considered a specific type of $p_\theta(\mathbf{x}^{(\alpha_{t-1})}|\mathbf{x}^{(\alpha_t)})$:

$$p_\theta(\mathbf{x}^{(\alpha_{t-1})}|\mathbf{x}^{(\alpha_t)}) = \mathcal{N}\left(\mu_\theta(\mathbf{x}^{(\alpha_t)}; \alpha_t, \alpha_{t-1}), (\sigma^{(\alpha_t)})^2\boldsymbol{I}\right), \tag{8.5}$$

where μ_θ and $\sigma^{(\alpha_t)}$ are parameters, and we remove the superscript of p_θ to indicate that there are no additional discretization steps in between (the sampling process is explicitly defined). Then, we have the standard variational objective as follows:

$$\begin{aligned} L &:= \mathbb{E}_q\left[\log q(\mathbf{x}^{(\alpha_T)}|\mathbf{x}^{(\alpha_0)}) + \sum_{t=2}^{T}\log q(\mathbf{x}^{(\alpha_{t-1})}|\mathbf{x}^{(\alpha_t)}, \mathbf{x}^{(\alpha_0)}) - \sum_{t=1}^{T}\log p_\theta^{(\alpha_t, \alpha_{t-1})}(\mathbf{x}^{(\alpha_{t-1})}|\mathbf{x}^{(\alpha_t)})\right] \\ &\equiv \mathbb{E}_q\left[\sum_{t=2}^{T}\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}^{(\alpha_{t-1})}|\mathbf{x}^{(\alpha_t)}, \mathbf{x}^{(\alpha_0)}))\|p_\theta(\mathbf{x}^{(\alpha_{t-1})}|\mathbf{x}^{(\alpha_t)}))}_{L_{t-1}} - \log p_\theta(\mathbf{x}^{(\alpha_0)}|\mathbf{x}^{(\alpha_1)})\right], \end{aligned}$$

where $\equiv$ denotes "equal up to a constant that does not depend on θ" and each L_{t-1} is a KL divergence between two Gaussian distributions. Let us assume that the standard deviation of $p_\theta(\mathbf{x}^{(\alpha_{t-1})}|\mathbf{x}^{(\alpha_t)})$

is equal to that of $q(\mathbf{x}^{(\alpha_{t-1})}|\mathbf{x}^{(\alpha_t)}, \mathbf{x}^{(\alpha_0)}))$, which we denote as $\sigma^{(\alpha_t)}$. And thus:

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2(\sigma^{(\alpha_t)})^2} \|\mu_\theta(\mathbf{x}^{(\alpha_t)}; \alpha_t, \alpha_{t-1}) - \tilde{\mu}(\mathbf{x}^{(\alpha_t)}, \mathbf{x}^{(\alpha_0)}; \alpha_t, \alpha_{t-1})\|_2^2 \right]. \tag{8.6}$$

With a particular reparametrization from μ_θ to ϵ_θ (which tries to model the noise vector at α_t):

$$\mu_\theta(\mathbf{x}^{(\alpha_t)}; \alpha_t, \alpha_{t-1}) = \sqrt{\frac{\alpha_{t-1}}{\alpha_t}} \left(\mathbf{x}^{(\alpha_t)} - \frac{\sqrt{\alpha_{t-1} - \alpha_t}}{\sqrt{(1-\alpha_t)\alpha_t}} \cdot \epsilon_\theta(\mathbf{x}^{(\alpha_t)}; \alpha_t) \right), \tag{8.7}$$

the objective function can be simplified to:

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{(\alpha_{t-1} - \alpha_t)}{2(\sigma^{(\alpha_t)})^2 (1-\alpha_t)\alpha_t} \|\epsilon - \epsilon_\theta(\mathbf{x}^{(\alpha_t)}; \alpha_t, \alpha_{t-1})\|_2^2 \right] \tag{8.8}$$

where $\mathbf{x}^{(\alpha_t)} = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\epsilon$. Intuitively, this is a weighted sum of mean-square errors between the noise model ϵ_θ and the actual noise ϵ. Other weights can also be derived with different forward processes that are non-Markovian [SME20], and in practice, setting the weights to 1 is observed to achieve decent performance for image generation.

8.4.2 DDIM sampling procedure

In this section, we discuss the detailed sampling procedure from $\mathbf{x}^{(0)} \sim \mathcal{N}(0, \boldsymbol{I})$ (which is the distribution with "all noise"[2]) to $\mathbf{x}^{(1)}$ (which is the model distribution with "no noise"). More specifically, we discuss a deterministic sampling procedure, which casts the generation procedure as an implicit model [SME20]. Compared to other procedures (such as the one in DDPM [HJA20]), this has the advantage of better sample quality when few steps are allowed to produce each sample, as well as a near-invertible mapping between $\mathbf{x}^{(0)}$ and $\mathbf{x}^{(1)}$. We describe this procedure in Algorithm 2, where we can choose different series of α to control how many steps (and through which steps) we wish to draw a sample. The DDIM sampling procedure corresponds to a particular discretization to an ODE, we note that it is straightforward to also define the sampling procedure between any two α values. Similarly, given an observation $\mathbf{x}^{(1)}$ we can obtain the corresponding latent code $\mathbf{x}^{(0)}$ by sampling running Algorithm 2 with the sequence of α reversed.

[2]Technically, the maximum noise level α_T should have $\alpha_T \to 0$ but not equal to 0, but we can approximate the distribution of $\mathbf{x}^{(\alpha_T)}$ with that of $\mathbf{x}^{(0)}$ arbitrarily well in practice.

Algorithm 2 Sampling with the DDIM procedure

1: **Input**: non-increasing series $\{\alpha_t\}_{t=0}^{T}$ with $\alpha_T = 0$ and $\alpha_0 = 1$.
2: Sample $\mathbf{x}^{(1)} \sim \mathcal{N}(0, \boldsymbol{I})$.
3: **for** $k \leftarrow T$ to 1 **do**
4: Update $\mathbf{x}^{(\alpha_{t-1})}$ from $\mathbf{x}^{(\alpha_t)}$ such that

$$\sqrt{\frac{1}{\alpha_{t-1}}}\mathbf{x}^{(\alpha_{t-1})} = \sqrt{\frac{1}{\alpha_t}}\mathbf{x}^{(\alpha_t)} + \left(\sqrt{\frac{1-\alpha_{t-1}}{\alpha_{t-1}}} - \sqrt{\frac{1-\alpha_t}{\alpha_t}}\right) \cdot \epsilon_\theta(\mathbf{x}^{(\alpha_t)}; \alpha_t)$$

5: **end for**
6: **Output** $\mathbf{x}^{(0)}$.

8.5 Diffusion-Decoding Generative Models with Contrastive Learning

While VAEs are ideal for learning rich latent representations due to being able to incorporate SSL within the encoder, they generally do not achieve the same level of sample quality as GANs and diffusion models.

To address this issue, we present Diffusion-Decoding generative models with Contrastive Learning (D2C), an extension to VAEs with high-quality samples and high-quality latent representations, and are thus well suited to few-shot conditional generation. Moreover, unlike GAN-based methods, D2C does not involve unstable adversarial training (Table 8.1).

As its name suggests, the generative model for D2C has two components – *diffusion* and *decoding*; the *diffusion* component operates over the latent space and the *decoding* component maps from latent representations to images. Let us use the α index notation for diffusion random variables: $\mathbf{z}^{(0)} \sim p^{(0)}(\mathbf{z}^{(0)}) := \mathcal{N}(0, \boldsymbol{I})$ is the "noisy" latent variable with $\alpha = 0$, and $\mathbf{z}^{(1)}$ is the "clean" latent variable with $\alpha = 1$. The generative process of D2C, which we denote $p_\theta(\mathbf{x}|\mathbf{z}^{(0)})$, is then defined as:

$$\mathbf{z}^{(0)} \sim p^{(0)}(\mathbf{z}^{(0)}), \quad \mathbf{z}^{(1)} \sim \underbrace{p_\theta^{(0,1)}(\mathbf{z}^{(1)}|\mathbf{z}^{(0)})}_{\text{diffusion}}, \quad \mathbf{x} \sim \underbrace{p_\theta(\mathbf{x}|\mathbf{z}^{(1)})}_{\text{decoding}}, \tag{8.9}$$

where $p^{(0)}(\mathbf{z}^{(0)}) = \mathcal{N}(0, \boldsymbol{I})$ is the prior distribution for the diffusion model, $p_\theta^{(0,1)}(\mathbf{z}^{(1)}|\mathbf{z}^{(0)})$ is the diffusion process from $\mathbf{z}^{(0)}$ to $\mathbf{z}^{(1)}$, and $p_\theta(\mathbf{x}|\mathbf{z}^{(1)})$ is the decoder from $\mathbf{z}^{(1)}$ to $\mathbf{x}$. Intuitively, D2C models produce samples by drawing $\mathbf{z}^{(1)}$ from a diffusion process and then decoding $\mathbf{x}$ from $\mathbf{z}^{(1)}$.

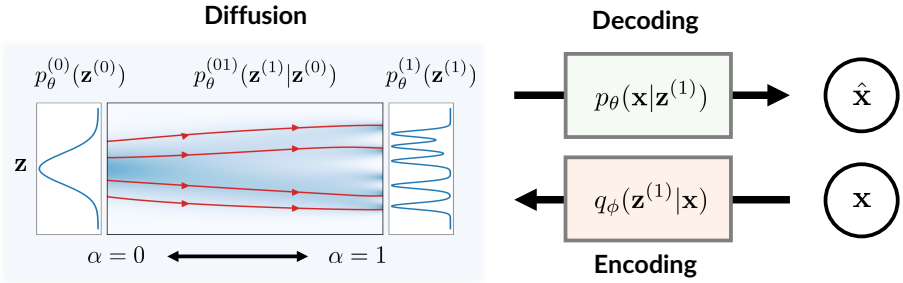


Figure 8.2: **Illustration of components of a D2 model.** On top of the encoding and decoding between $\mathbf{x}$ and $\mathbf{z}^{(1)}$, we use a diffusion model to generate $\mathbf{z}^{(1)}$ from a Gaussian $\mathbf{z}^{(0)}$. The red lines describe several smooth ODE trajectories from $\alpha = 0$ to $\alpha = 1$ corresponding to DDIM.

In order to train a D2C model, we use an inference model $q_\phi(\mathbf{z}^{(1)}|\mathbf{x})$ that predicts proper $\mathbf{z}^{(1)}$ latent variables from $\mathbf{x}$; this can directly incorporate SSL methods [XLZ+21], leading to the following objective:

$$L_{\text{D2C}}(\theta, \phi; w) := L_{\text{D2}}(\theta, \phi; w) + \lambda L_{\text{C}}(q_\phi), \tag{8.10}$$

$$L_{\text{D2}}(\theta, \phi; w) := \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}, \mathbf{z}^{(1)}\sim q_\phi(\mathbf{z}^{(1)}|\mathbf{x})}[-\log p(\mathbf{x}|\mathbf{z}^{(1)}) + \ell_{\text{diff}}(\mathbf{z}^{(1)}; w, \theta)], \tag{8.11}$$

where ℓ_{diff} is defined as in equation 8.1, $L_C(q_\phi)$ denotes any contrastive predictive coding objective [vdOLV18] with rich data augmentations [HFW+19, CFGH20, CXH21, CKNH20, SE20b] (details in Appendix B.6.1) and $\lambda > 0$ is a weight hyperparameter. The first two terms, which we call L_{D2}, contains a "reconstruction loss" ($-\log p(\mathbf{x}|\mathbf{z}^{(1)})$) and a "diffusion loss" over samples of $\mathbf{z}^{(1)} \sim q_\phi(\mathbf{z}^{(1)}|\mathbf{x})$. We illustrate the D2C generative and inference models in Figure 8.2, and its training procedure in Appendix B.6.1.

8.5.1 Relationship to maximum likelihood

The D2 objective (L_{D2}) appears similar to the original VAE objective (L_{VAE}). Here, we make an informal statement that the D2 objective function is deeply connected to the variational lower bound of log-likelihood; we present the full statement and proof in Appendix A.6.1.

Theorem 12. *(informal) For any valid $\{\alpha_i\}_{i=0}^T$, there exists some weights $\hat{w} : \{\alpha_i\}_{i=1}^T \to \mathbb{R}_+$ for the diffusion objective such that $-L_{\text{D2}}$ is a variational lower bound to the log-likelihood, i.e.,*

$$-L_{\text{D2}}(\theta, \phi; \hat{w}) \leq \mathbb{E}_{p_{\text{data}}}[\log p_\theta(\mathbf{x})], \tag{8.12}$$

where $p_\theta(\mathbf{x}) := \mathbb{E}_{\mathbf{x}_0 \sim p^{(0)}(\mathbf{z}^{(0)})}[p_\theta(\mathbf{x}|\mathbf{z}^{(0)})]$ *is the marginal probability of* $\mathbf{x}$ *under the D2C model.*

Proof. (sketch) The diffusion term ℓ_{diff} upper bounds the KL divergence between $q_\phi(\mathbf{z}_1|\mathbf{x})$ and $p_\theta^{(1)}(\mathbf{z}^{(1)})$ for suitable weights [HJA20, SME20], which recovers a VAE objective. □

8.5.2 D2C models address latent posterior mismatch in VAEs

While D2C is a special case of VAE, we argue that D2C is non-trivial in the sense that it addresses a long-standing problem in VAE methods [TW17, TIY+19], namely the mismatch between the prior distribution $p_\theta(\mathbf{z})$ and the aggregate (approximate) posterior distribution $q_\phi(\mathbf{z}) := \mathbb{E}_{p_{\text{data}}(\mathbf{x})}[q_\phi(\mathbf{z}|\mathbf{x})]$. A mismatch could create "holes" [RLM18, HJ16, ASKV20] in the prior that the aggregate posterior fails to cover during training, resulting in worse sample quality, as many latent variables used during generation are likely to never have been trained on. We formalize this notion in the following definition.

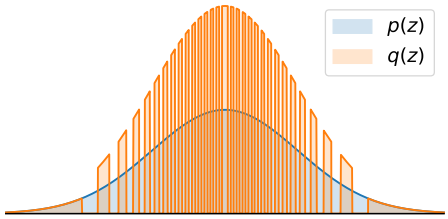
Definition 1 (Prior hole)**.** *Let* $p(\mathbf{z}), q(\mathbf{z})$ *be two distributions with* $\text{supp}(q) \subseteq \text{supp}(p)$*. We say that* q *has an* (ϵ, δ)***-prior hole*** *with respect to (the prior)* p *for* $\epsilon, \delta \in (0, 1)$*,* $\delta > \epsilon$*, if there exists a set* $S \in \text{supp}(P)$*, such that* $\int_S p(\mathbf{z})\,\mathrm{d}\mathbf{z} \geq \delta$ *and* $\int_S q(\mathbf{z})\,\mathrm{d}\mathbf{z} \leq \epsilon$*.*

Intuitively, if $q_\phi(\mathbf{z})$ has a prior hole with large δ and small ϵ (*e.g.*, inversely proportional to the number of training samples), then it is very likely that latent variables within the hole are never seen during training (small ϵ), yet frequently used to produce samples (large δ). Most existing methods address this problem by optimizing certain statistical divergences between $q_\phi(\mathbf{z})$ and $p_\theta(\mathbf{z})$, such as the KL divergence or Wasserstein distance [TBGS17]. However, we argue in the following statement that prior holes might not be eliminated even if we optimize certain divergence values to be reasonably low, especially when $q_\phi(\mathbf{z})$ is very flexible. We present the formal statement and proof in Appendix A.6.2.

Theorem 13. *(informal) Let* $p_\theta(\mathbf{z}) = \mathcal{N}(0, 1)$*. For any* $\epsilon > 0$*, there exists a distribution* $q_\phi(\mathbf{z})$ *with an* $(\epsilon, 0.49)$*-prior hole, such that* $D_{\text{KL}}(q_\phi \| p_\theta) \leq \log 2$[3] *and* $W_2(q_\phi, p_\theta) < \gamma$ *for any* $\gamma > 0$*, where* W_2 *is the 2-Wasserstein distance.*

Proof. (sketch) We construct a q_ϕ that satisfies these properties (see figure). First, we truncate the Gaussian and divide them into regions with same probability mass; then we support q_ϕ over half of these regions (so $\delta > 0.49$); finally, we show that the divergences are small enough. □

[3] This is reasonably low for realistic VAE models (NVAE [VK20] reports a KL divergence of around 2810 nats).



In contrast to addressing prior holes by optimization, diffusion models eliminate prior holes by construction, since the diffusion process from $\mathbf{z}^{(1)}$ to $\mathbf{z}^{(0)}$ is constructed such that the distribution of $\mathbf{z}^{(\alpha)}$ always converges to a standard Gaussian as $\alpha \to 0$. As a result, the distribution of latent variables used during training is arbitrarily close to that used in generation[4], which is also the case in GANs. Therefore, our argument provides an explanation as to why we observe better sample quality results from GANs and diffusion models than VAEs and NFs.

8.6 Few-shot Conditional Generation with D2C

In this section, we discuss how D2C can be used to learn to perform conditional generation from few-shot supervision. We note that D2C is only trained on images and not with any other data modalities (*e.g.*, image-text pairs [RPG+21]) or supervision techniques (*e.g.*, meta-learning [CD19, BV18]).

Algorithm We describe the general algorithm for conditional generation from a few images in Algorithm 3. With a model over the latent space (denoted as $r_\psi(\mathbf{c}|\mathbf{z}^{(1)})$), we draw conditional latents from an unnormalized distribution with the diffusion prior (line 4). This can be implemented in many ways such as rejection sampling or Langevin dynamics [NCB+17, SSDK+20, DN21].

Algorithm 3 Conditional generation with D2C

1: **Input** n examples $\{(\mathbf{x}_i, \mathbf{c}_i)\}_{i=1}^n$, property $\mathbf{c}$.
2: Acquire latents $\mathbf{z}_i^{(1)} \sim q_\phi(\mathbf{z}^{(1)}|\mathbf{x}_i)$ for $i \in [n]$;
3: Train model $r_\psi(\mathbf{c}|\mathbf{z}^{(1)})$ over $\{(\mathbf{z}_i^{(1)}, \mathbf{c}_i)\}_{i=1}^n$
4: Sample latents with $\hat{\mathbf{z}}^{(1)} \sim r_\psi(\mathbf{c}|\mathbf{z}^{(1)}) \cdot p_\theta^{(1)}(\mathbf{z}^{(1)})$ (unnormalized);
5: Decode $\hat{\mathbf{x}} \sim p_\theta(\mathbf{x}|\hat{\mathbf{z}}^{(1)})$.
6: **Output** $\hat{\mathbf{x}}$.

Conditions from labeled examples Given a few labeled examples, we wish to produce diverse samples with a certain label. For labeled examples we can directly train a classifier over the latent space, which we denote as $r_\psi(\mathbf{c}|\mathbf{z}^{(1)})$ with $\mathbf{c}$ being the class label and $\mathbf{z}^{(1)}$ being the latent

[4]We expand this argument in Appendix A.6.2.

representation of $\mathbf{x}$ from $q_\phi(\mathbf{z}^{(1)}|\mathbf{x})$. If these examples do not have labels (*i.e.*, we merely want to generate new samples similar to given ones), we can train a positive-unlabeled (PU) classifier [EN08] where we assign "positive" to the new examples and "unlabeled" to training data. Then we use the classifier with the diffusion model $p_\theta(\mathbf{z}^{(1)}|\mathbf{z}^{(0)})$ to produce suitable values of $\mathbf{z}^{(1)}$, such as by rejecting samples from the diffusion model that has a small $r_\psi(\mathbf{c}|\mathbf{z}^{(1)})$.

Conditions from manipulation constraints Given a few labeled examples, here we wish to learn how to manipulate images. Specifically, we condition over the event that "$\mathbf{x}$ has label $\mathbf{c}$ but is similar to image $\bar{\mathbf{x}}$". Here $r_\psi(\mathbf{c}|\mathbf{z}^{(1)})$ is the unnormalized product between the classifier conditional probability and closeness to the latent $\bar{\mathbf{z}}^{(1)}$ of $\bar{\mathbf{x}}$ (*e.g.*, measured with RBF kernel). We implement line 4 of Alg. 3 with a Lanvegin-like procedure where we take a gradient step with respect to the classifier probability and then correct this gradient step with the diffusion model. Unlike many GAN-based methods [CK17, PHS+18, WLZ+18, IZZE17, XYXW21], D2C does not need to optimize an inversion procedure at evaluation time, and thus the latent value is much faster to compute; D2C is also better at retaining fine-grained features of the original image due to the reconstruction loss.

Detailed Algorithms In order to perform few-shot conditional generation, we need to implement line 4 in Algorithm 3, where an unnormalized (energy-based) model is defined over the representations. After we have defined the energy-based model, we implement a procedure to draw samples from this unnormalized model. We note that our approach (marked in teal boxes) is only one way of drawing valid samples, and not necessarily the optimal one. Furthermore, these implementations can also be done over the image space (which is the case for DDIM-I), which may costs more to compute than over the latent space since more layers are needed in a neural network to process it.

For generation from labels, we would define the energy-based model over latents as the product of two components: the first is the "prior" over $\mathbf{z}^{(1)}$ as defined by the diffusion model and the second is the "likliehood" of the label $\mathbf{c}$ being true given the latent variable $\mathbf{z}^{(1)}$. This places high energy values to the latent variables that are likely to occur under the diffusion prior (so generated images are likely to have high quality) as well as latent variables that have the label $\mathbf{c}$. To sample from this energy-based model, we perform a rejection sampling procedure, where we reject latent samples from the diffusion model that have low discrminator values. This procedure is describe in Algorithm 4.

Algorithm 4 Generate from labels

Input model $r_\psi(\mathbf{c}|\mathbf{z}^{(1)})$, target label $\mathbf{c}$.

Define latent energy-based model

$E(\hat{\mathbf{z}}^{(1)}) = r_\psi(\mathbf{c}|\hat{\mathbf{z}}^{(1)}) \cdot p_\theta^{(1)}(\hat{\mathbf{z}}^{(1)})$

Sample from $E(\hat{\mathbf{z}}^{(1)})$

while True **do**
  Sample $\hat{\mathbf{z}}^{(1)} \sim p_\theta^{(1)}(\hat{\mathbf{z}}^{(1)})$;
  Sample $u \sim \text{Uniform}(0, 1)$;
  If $u < r_\psi(\mathbf{c}|\hat{\mathbf{z}}^{(1)})$ **then** break.
end while

Output $\hat{\mathbf{x}} \sim p_\theta(\mathbf{x}|\hat{\mathbf{z}}^{(1)})$.

For generation from manipulation constraints, we need to further define a prior that favors closeness to the given latent variable so that the manipulated generation is close to the given image except for the label $\mathbf{z}$. If the latent variable for the original image is $\mathbf{z}^{(1)} \sim q_\phi(\mathbf{z}^{(1)}|\mathbf{x})$, then we define the closeness via the L2 distance between the it and the manipulated latent. We obtain the energy-based model by multiplying this with the diffusion "prior" and the classifier "likelihood". Then, we approximately draw samples from this energy by taking a gradient step from the original latent value $\mathbf{z}^{(1)}$ and then regularizing it with the diffusion prior; this is described in Algorithm 5. A step size η, diffusion noise magnitude α, and the diffusion steps from α to 1 are chosen as hyperparameters. We choose one η for each attribute, $\alpha \approx 0.9$, and number of discretization steps to be 5[5]; we tried $\alpha \in [0.65, 0.9]$ and found that our results are not very sensitive to values within this range. We list the η values for each attribute (details in Appendix B.6.2).

We note that a more principled approach is to take gradient with respect to the entire energy function (*e.g.*, for Langevin dynamics), where the gradient over the DDIM can be computed with instantaneous change-of-variables formula [CRBD18]; we observe that our current version is computationally efficient enough to perform well.

[5]The results are not particularly sensitive to how the discretization steps are chosen. For example, one can take $0.9 \to 0.92 \to 0.96 \to 0.98 \to 0.99 \to 1$.

Algorithm 5 Generate from manipulation constraints

Input model $r_\psi(\mathbf{c}|\mathbf{z}^{(1)})$, target label $\mathbf{c}$, original image $\mathbf{x}$.

Acquire latent $\mathbf{z}^{(1)} \sim q_\phi(\mathbf{z}^{(1)}|\mathbf{x})$;

Fit a model $r_\psi(\mathbf{c}|\mathbf{z}^{(1)})$ over $\{(\mathbf{z}_i^{(1)}, \mathbf{c}_i)\}_{i=1}^n$

Define latent energy-based model

$$E(\hat{\mathbf{z}}^{(1)}) = r_\psi(\mathbf{c}|\hat{\mathbf{z}}^{(1)}) \cdot p_\theta^{(1)}(\hat{\mathbf{z}}^{(1)}) \cdot \|\mathbf{z}^{(1)} - \hat{\mathbf{z}}^{(1)}\|_2^2$$

Sample from $E(\hat{\mathbf{z}}^{(1)})$ (approximate)

Choose hyperparameters $\eta > 0, \alpha \in (0, 1)$.

Take a gradient step $\bar{\mathbf{z}}^{(1)} \leftarrow \mathbf{z}^{(1)} + \eta \nabla_{\mathbf{z}} r_\psi(\mathbf{c}|\mathbf{z})|_{\mathbf{z}=\mathbf{z}^{(1)}}$.

Add noise $\tilde{\mathbf{z}}^{(\alpha)} \leftarrow \sqrt{\alpha}\bar{\mathbf{z}}^{(1)} + \sqrt{1-\alpha}\epsilon$.

Sample $\hat{\mathbf{z}}^{(1)} \sim p_\theta^{(\alpha,1)}(\mathbf{z}^{(1)}|\tilde{\mathbf{z}}^{(\alpha)})$ with DDIM, *i.e.*, use the diffusion prior to "denoise".

Output $\hat{\mathbf{x}} \sim p_\theta(\mathbf{x}|\hat{\mathbf{z}}^{(1)})$.

8.7 Related Work

Latent variable generative models Most deep generative models explicitly define a latent representation, except for some energy-based models [Hin02, DM19] and autoregressive models [vdOKK16, vdODZ+16, BMR+20]. Unlike VAEs and NFs, GANs do not explicitly define an inference model and instead optimize a two-player game. In terms of sample quality, GANs currently achieve superior performance over VAEs and NFs, but they can be difficult to invert even with additional optimization [KALL17, XSZ+20, BZW+19]. This can be partially addressed by training reconstruction-based losses with GANs [LSLW16, LLC+17]. Moreover, the GAN training procedure can be unstable [BLRW16, BDS18, MKKY18], lack a informative objective for measuring progress [ACB17], and struggle with discrete data [YZWY17]. Diffusion models [DN21] achieves high sample quality without adversarial training, but its latent dimension must be equal to the image dimension.

Addressing posterior mismatch in VAEs Most methods address this mismatch problem by improving inference models [ML16, KSJ+16a, TW16], prior models [TW17, ASKV20, TIY+19], or objective functions [ZSE17a, ZSE17b, ZSE18b, APF+17, MSJ+15]; all these approaches optimize the posterior model to be close to the prior. In Section 8.5.2, we explain why these approaches do

not necessarily remove large "prior holes", so their sample qualities remain relatively poor even after many layers [VK20, Chi20]. Other methods adopt a "two-stage" approach [DW19], which fits a generative model over the latent space of autoencoders [vdOVK17, ROV19, DJP+20, RPG+21].

Conditional generation with unconditional models To perform conditional generation over an unconditional LVGM, most methods assume access to a discriminative model (*e.g.*, a classifier); the latent space of the generator is then modified to change the outputs of the discriminative model. The disciminative model can operate on either the image space [NCB+17, PWS+21, DN21] or the latent space [SGTZ20, XYXW21]. For image space discriminative models, plug-and-play generative networks [NCB+17] control the attributes of generated images via Langevin dynamics [RR98]; these ideas are also explored in diffusion models [SSDK+20]. Image manipulation methods are based on GANs often operate with latent space discriminators [SGTZ20, XYXW21]. However, these methods have some trouble manipulating real images because of imperfect reconstruction [ZZZZ19, BZW+19]. This is not a problem in D2C since a reconstruction objective is optimized.

8.8 Experiments

We examine the conditional and unconditional generation qualities of D2C over CIFAR-10 [KSH12], CIFAR-100 [KSH12], fMoW [CFWM18], CelebA-64 [LLWT15], CelebA-HQ-256 [KALL17], and FFHQ-256 [KLA18]. Our D2C implementation is based on the state-of-the-art NVAE [VK20] autoencoder structure, the U-Net diffusion model [HJA20], and the MoCo-v2 contrastive representation learning method [CFGH20]. We keep the diffusion series hyperparameter $\{\alpha_i\}_{i=1}^T$ identical to ensure a fair comparison with different diffusion models. For the contrastive weight hyperparameter λ in Equation 8.10, we consider the value of $\lambda = 10^{-4}$ based on the relative scale between the L_{C} and L_{D2}; we find that the results are relatively insensitive to λ. We use 100 diffusion steps for DDIM and D2C unless mentioned otherwise, as running with longer steps is not computationally economical despite tiny gains in FID [SME20]. We include additional training details, such as architectures, optimizers and learning rates in Appendix B.6.2.

8.8.1 Unconditional generation

For unconditional generation, we measure the sample quality of images using the Frechet Inception Distance (FID, [HRU+17]) with 50,000 images. In particular, we extensively evaluate NVAE [VK20] and DDIM [SME20], a competitive VAE model and a competitive diffusion model as baselines

Table 8.2: Quality of representations and generations with LVGMs.

Model	CIFAR-10			CIFAR-100			fMoW		
	FID ↓	MSE ↓	Acc ↑	FID ↓	MSE ↓	Acc ↑	FID ↓	MSE ↓	Acc ↑
NVAE [VK20]	36.4	**0.25**	18.8	42.5	0.53	4.1	82.25	**0.30**	27.7
DDIM [SME20]	**4.16**	2.5	22.5	**10.16**	3.2	2.2	**37.74**	3.0	23.5
D2C (Ours)	10.15	0.76	**76.02**	14.62	**0.44**	**42.75**	44.7	2.33	**66.9**

Figure 8.3: Generated samples on CIFAR-10, fMoW, and FFHQ.

because we can directly obtain features from them without additional optimization steps[6]. For them, we report mean-squared reconstruction error (MSE, summed over all pixels, pixels normalized to $[0, 1]$) and linear classification accuracy (Acc., measured in percentage) over $\mathbf{z}_1$ features for the test set.

We report sample quality results[7] in Tables 8.2, and 8.3. For FID, we outperform NVAE in all datasets and outperform DDIM on CelebA-64 and CelebA-HQ-256, which suggests our results are competitive with state-of-the-art non-adversarial generative models. In Table 8.2, we additionally compare NVAE, DDIM and D2C in terms of reconstruction and linear classification accuracy. As all three methods contain reconstruction losses, the MSE values are low and comparable. However, D2C enjoys much better linear classification accuracy than the other two thanks to the contrastive SSL component. We further note that training the same contrastive SSL method without L_{D2} achieves slightly higher 78.3% accuracy on CIFAR-10. We tried improving this via ResNet [HZRS15] encoders, but this significantly increased reconstruction error, possibly due to loss of information in average pooling layers.

[6]For DDIM, the latent representations $\mathbf{x}^{(0)}$ are obtained by reversing the neural ODE process.
[7]Due to space limits, we place additional CIFAR-10 results in Appendix B.6.3.

Table 8.3: FID scores over different faces dataset with LVGMs.

Model	CelebA-64	CelebA-HQ-256	FFHQ-256
NVAE [VK20]	13.48	40.26	26.02
DDIM [SME20]	6.53	25.6	-
D2C (Ours)	**5.7**	**18.74**	**13.04**

Table 8.4: Sample quality as a function of diffusion steps.

	CIFAR-10			CIFAR-100			CelebA-64		
Steps	10	50	100	10	50	100	10	50	100
DDPM [HJA20]	41.07	8.01	5.78	50.27	21.37	16.72	33.12	18.48	13.93
DDIM [SME20]	**13.36**	**4.67**	**4.16**	**23.34**	**11.69**	**10.16**	17.33	9.17	6.53
D2C (Ours)	17.71	10.11	10.15	23.16	14.62	14.46	**17.32**	**6.8**	**5.7**

8.8.2 Few-shot conditional generation from examples

We demonstrate the advantage of D2C representations by performing few-shot conditional generation over labels. We consider two types of labeled examples: one has binary labels for which we train a binary classifier; the other is positive-only labeled (*e.g.*, images of female class) for which we train a PU classifier. Our goal here is to generate a diverse group of images with a certain label. We evaluate and compare three models: D2C, NVAE and DDIM. We train a classifier $r_\psi(\mathbf{c}|\mathbf{z})$ over the latent space of these models; we also train a image space classifier and use it with DDIM (denoted as DDIM-I). We run Algorithm 3 for these models, where line 4 is implemented via rejection sampling. As our goal is to compare different models, we leave more sophisticated methods [DN21] as future work.

We consider performing 8 conditional generation tasks over CelebA-64 with 2 binary classifiers (trained over 100 samples, 50 for each class) and 4 PU classifiers (trained over 100 positively labeled and 10k unlabeled samples). We also report a "naive" approach where we use all the training images (regardless of labels) and compute its FID with the corresponding subset of images (*e.g.*, all images versus blond images). In Table 8.5, we report the FID score between generated images (5k samples) and real images of the corresponding label. These results suggest that D2C outperforms the other approaches, and is the only one that performs better than the "naive" approach in most cases, illustrating the advantage of contrastive representations for few-shot conditional generation.

Table 8.5: FID scores for few-shot conditional generation with various types of labeled examples. Naive performs very well for non-blond due to class percentages.

Method	Classes (% in train set)	D2C	DDIM	NVAE	DDIM-I	Naive
Binary	Male (42%)	**13.44**	38.38	41.07	29.03	26.34
	Female (58%)	**9.51**	19.25	16.57	15.17	18.72
	Blond (15%)	**17.61**	31.39	31.24	29.09	27.51
	Non-Blond (85%)	**8.94**	9.67	16.73	19.76	3.77
PU	Male (42%)	**16.39**	37.03	42.78	19.60	26.34
	Female (58%)	**12.21**	15.42	18.36	14.96	18.72
	Blond (15%)	**10.09**	30.20	31.06	76.52	27.51
	Non-Blond (85%)	**9.09**	9.70	17.98	9.90	3.77

8.8.3 Few-shot conditional generation from manipulation constraints

Finally, we consider image manipulation where we use binary classifiers that are learned over 50 labeled instances for each class. We perform Amazon Mechanical Turk (AMT) evaluations over two attributes in the CelebA-256 dataset, *blond* and *red lipstick*, over D2C, DDIM, NVAE and StyleGAN2 [KLA+20] (see Figure 8.4). The evaluation is double-blinded: neither we nor the evaluators know the correspondence between generated image and underlying model during the study. We include more details (algorithm, setup and human evaluation) in Appendix B.6.2 and additional qualitative results (such as *beard* and *gender* attributes) in Appendix B.6.3.

In Figure 8.5, we show the percentage of manipulations preferred by AMT evaluators for each model; D2C slightly outperforms StyleGAN2 for *blond* and significantly outperforms StyleGAN2 for *red lipstick*. When we compare D2C with only StyleGAN2, D2C is preferred over 51.5% for *blond* and 60.8% for *red lipstick*. An additional advantage of D2C is that the manipulation is much faster than StyleGAN2, since the latter requires additional optimization over the latent space to improve reconstruction [ZSZZ20]. On the same Nvidia 1080Ti GPU, it takes 0.013 seconds to obtain the latent code in D2C, while the same takes 8 seconds [ZSZZ20] for StyleGAN2 ($615\times$ slower). As decoding is very fast for both models, D2C generations are around two orders of magnitude faster to produce.

8.9 Discussions

We introduced D2C, a VAE-based generative model with a latent space suitable for few-shot conditional generation. To our best knowledge, our model is the first unconditional VAE to

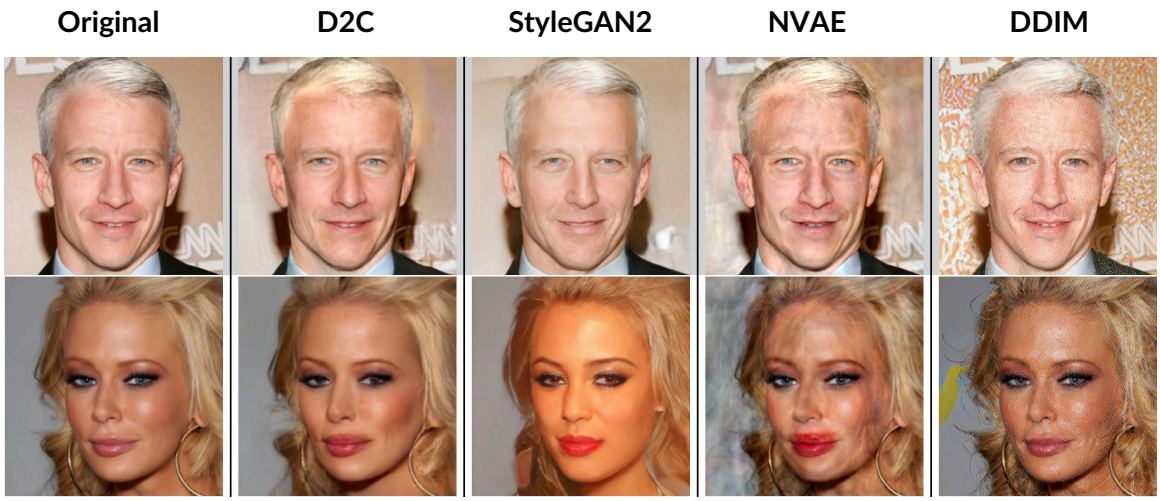


Figure 8.4: Image manipulation results for *blond* (top) and *red lipstick* (bottom). D2C is better than StyleGAN2 at preserving details of the original image, such as eyes, earrings, and background.

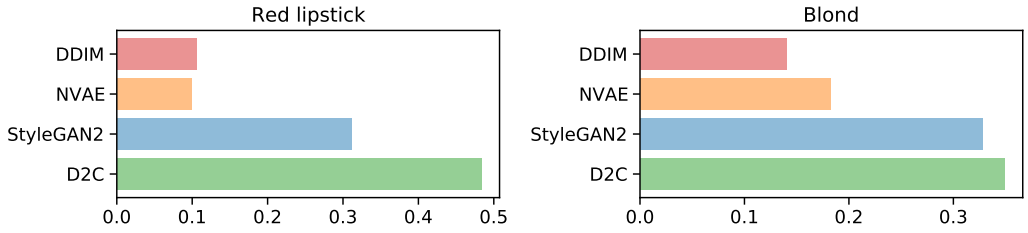


Figure 8.5: **AMT evaluation over image manipulations.** x-axis shows the percentage that the evaluator selects the image generated from the corresponding model out of 4 images from each model.

demonstrate superior image manipulation performance than StyleGAN2, which is surprising given our use of a regular NVAE architecture. We believe that with better architectures, such as designs from StyleGAN2 or Transformers [HZ21], D2C can achieve even better performance. It is also interesting to formally investigate the integration between D2C and other types of conditions on the latent space, as well as training D2C in conjunction with other domains and data modalities, such as text [RPG+21], in a fashion that is similar to semi-supervised learning. Nevertheless, we note that our model have to be used properly in order to mitigate potential negative societal impacts, such as deep fakes.

Part III

Inference via Supervised Learning

Chapter 9

Learning Kernels for Markov Chain Monte Carlo

In the previous part, we discussed how supervised learning methods can be integrated into unconditional and conditional generative models, such as generative adversarial networks and variational autoencoders. In this part, we discuss how supervised learning can be used to improve inference methods, starting with Bayesian inference.

Markov Chain Monte Carlo (MCMC) methods are regarded as the "golden standard" of Bayesian inference, but existing MCMC methods are either based on general-purpose and domain-agnostic schemes that can lead to slow convergence, or based on problem-specific proposals hand-crafted by an expert. As a result, MCMC methods are falling out of favor compared to approximate inference methods (such as variational inference) despite being asymptotically exact.

In this chapter, we propose A-NICE-MC, a novel method that uses supervised learning to automatically design efficient Markov chain kernels tailored for a specific domain. First, we propose an efficient likelihood-free supervised learning (adversarial training) method to train a Markov chain and mimic a given data distribution. Then, we leverage flexible volume preserving flows to obtain parametric kernels for MCMC. Using a bootstrap approach, we show how to train efficient Markov chains to sample from a prescribed posterior distribution by iteratively improving the quality of both the model and the samples. Empirical results demonstrate that A-NICE-MC combines the strong guarantees of MCMC with the expressiveness of deep neural networks, and is able to significantly outperform competing methods such as Hamiltonian Monte Carlo.

This chapter is previously published in [SZE17]. Shengjia Zhao and Daniel Levy contributed to the contents of this chapter. My contributions include conceiving the idea, implementing the

algorithms and executing the experiments.

9.1 Introduction

Variational inference (VI) and Monte Carlo (MC) methods are two key approaches to deal with complex probability distributions in machine learning. The former approximates an intractable distribution by solving a variational optimization problem to minimize a divergence measure with respect to some tractable family. The latter approximates a complex distribution using a small number of typical states, obtained by sampling ancestrally from a proposal distribution or iteratively using a suitable Markov chain (Markov Chain Monte Carlo, or MCMC).

Recent progress in deep learning has vastly advanced the field of variational inference. Notable examples include black-box variational inference and variational autoencoders [RGB14, KW13, RMW14], which enabled variational methods to benefit from the expressive power of deep neural networks, and adversarial training [GPAM$^+$14, ML16], which allowed the training of new families of implicit generative models with efficient ancestral sampling.

MCMC methods, on the other hand, have not benefited as much from these recent advancements. Unlike variational approaches, MCMC methods are iterative in nature and do not naturally lend themselves to the use of expressive function approximators [SKW15, DFHSJR01]. Even evaluating an existing MCMC technique is often challenging, and natural performance metrics are intractable to compute [GM15, GDVM16, GM17, EGSS14]. Defining an objective to improve the performance of MCMC that can be easily optimized in practice over a large parameter space is itself a difficult problem [MWHDF12, BDX04].

To address these limitations, we introduce A-NICE-MC, a new method for training flexible MCMC kernels, e.g., parameterized using (deep) neural networks. Given a kernel, we view the resulting Markov Chain as an implicit generative model, i.e., one where sampling is efficient but evaluating the (marginal) likelihood is intractable. We then propose supervised training as an effective, likelihood-free method for training a Markov chain to match a target distribution.

First, we show it can be used in a learning setting to directly approximate an (empirical) data distribution. We then use the approach to train a Markov Chain to sample efficiently from a model prescribed by an analytic expression (e.g., a Bayesian posterior distribution), the classic use case for MCMC techniques. We leverage flexible volume preserving flow models [DKB14] and a "bootstrap" technique to automatically design powerful domain-specific proposals that combine the guarantees of MCMC and the expressiveness of neural networks. Finally, we propose a method that

decreases autocorrelation and increases the effective sample size of the chain as training proceeds. We demonstrate that these trained operators are able to significantly outperform traditional ones, such as Hamiltonian Monte Carlo, in various domains.

9.2 Notations and Problem Setup

A sequence of continuous random variables $\{x_t\}_{t=0}^{\infty}$, $x_t \in \mathbb{R}^n$, is drawn through the following Markov chain:

$$x_0 \sim \pi^0 \qquad x_{t+1} \sim T_\theta(x_{t+1}|x_t)$$

where $T_\theta(\cdot|x)$ is a time-homogeneous stochastic transition kernel parametrized by $\theta \in \Theta$ and π^0 is some initial distribution for x_0. In particular, we assume that T_θ is defined through an implicit generative model $f_\theta(\cdot|x, v)$, where $v \sim p(v)$ is an auxiliary random variable, and f_θ is a deterministic transformation (e.g., a neural network). Let π_θ^t denote the distribution for x_t. If the Markov chain is both irreducible and positive recurrent, then it has an unique stationary distribution $\pi_\theta = \lim_{t\to\infty} \pi_\theta^t$. We assume that this is the case for all the parameters $\theta \in \Theta$.

Let $p_d(x)$ be a target distribution over $x \in \mathbb{R}^n$, e.g, a data distribution or an (intractable) posterior distribution in a Bayesian inference setting. Our objective is to find a T_θ such that it achieves:

1. **Low bias:** The stationary distribution is close to the target distribution (minimize $|\pi_\theta - p_d|$).
2. **Efficiency:** $\{\pi_\theta^t\}_{t=0}^{\infty}$ converges quickly (minimize t such that $|\pi_\theta^t - p_d| < \delta$).
3. **Low variance:** Samples from one chain $\{x_t\}_{t=0}^{\infty}$ should be as uncorrelated as possible (minimize autocorrelation of $\{x_t\}_{t=0}^{\infty}$).

We think of π_θ as a stochastic generative model, which can be used to efficiently produce samples with certain characteristics (specified by p_d), allowing for efficient Monte Carlo estimates. We consider two settings for specifying the target distribution. The first is a *learning* setting where we do not have an analytic expression for $p_d(x)$, but we have access to typical samples $\{s_i\}_{i=1}^m \sim p_d$; in the second case we have an analytic expression for $p_d(x)$, possibly up to a normalization constant, but no access to samples. The two cases are discussed in Sections 9.3 and 9.4 respectively.

9.3 Adversarial Training for Markov Chains

Consider the setting where we have direct access to samples from $p_d(x)$. Assume that the transition kernel $T_\theta(x_{t+1}|x_t)$ is the following implicit generative model:

$$v \sim p(v) \quad x_{t+1} = f_\theta(x_t, v) \tag{9.1}$$

Assuming a stationary distribution $\pi_\theta(x)$ exists, the value of $\pi_\theta(x)$ is typically intractable to compute. The marginal distribution $\pi_\theta^t(x)$ at time t is also intractable, since it involves integration over all the possible paths (of length t) to x. However, we can directly obtain samples from π_θ^t, which will be close to π_θ if t is large enough (assuming ergodicity). This aligns well with the idea of generative adversarial networks (GANs), a likelihood free method which only requires samples from the model.

Generative Adversarial Network (GAN) [GPAM+14] is a framework for training deep generative models using a two player minimax game. A generator network G generates samples by transforming a noise variable $z \sim p(z)$ into $G(z)$. A discriminator network $D(\mathbf{x})$ is trained to distinguish between "fake" samples from the generator and "real" samples from a given data distribution p_d. Formally, this defines the following objective (Wasserstein GAN, from [ACB17])

$$\min_G \max_D V(D, G) = \min_G \max_D \mathbb{E}_{x\sim p_d}[D(x)] - \mathbb{E}_{z\sim p(z)}[D(G(z))] \tag{9.2}$$

In our setting, we could assume $p_d(x)$ is the empirical distribution from the samples, and choose $z \sim \pi^0$ and let $G_\theta(z)$ be the state of the Markov Chain after t steps, which is a good approximation of π_θ if t is large enough. However, optimization is difficult because we do not know a reasonable t in advance, and the gradient updates are expensive due to backpropagation through the entire chain.

Therefore, we propose a more efficient approximation, called *Markov GAN* (MGAN):

$$\min_\theta \max_D \mathbb{E}_{x\sim p_d}[D(x)] - \lambda \mathbb{E}_{\bar{x}\sim \pi_\theta^b}[D(\bar{x})] - (1-\lambda)\mathbb{E}_{x_d\sim p_d, \bar{x}\sim T_\theta^m(\bar{x}|x_d)}[D(\bar{x})] \tag{9.3}$$

where $\lambda \in (0, 1), b \in \mathbb{N}^+, m \in \mathbb{N}^+$ are hyperparameters, $\bar{x}$ denotes "fake" samples from the generator and $T_\theta^m(x|x_d)$ denotes the distribution of x when the transition kernel is applied m times, starting from some "real" sample x_d.

We use two types of samples from the generator for training, optimizing θ such that the samples will fool the discriminator:

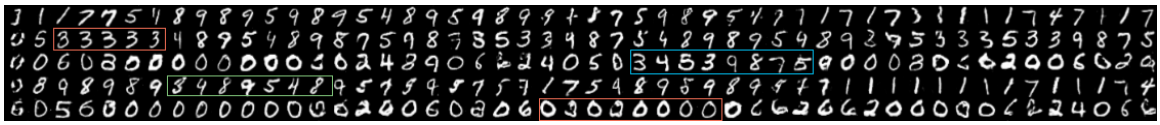


Figure 9.1: Visualizing samples of π_1 to π_{50} (each row) from a model trained on the MNIST dataset. Consecutive samples can be related in label (red box), inclination (green box) or width (blue box).

1. Samples obtained after b transitions $\bar{x} \sim \pi_\theta^b$, starting from $x_0 \sim \pi^0$.

2. Samples obtained after m transitions, starting from a data sample $x_d \sim p_d$.

Intuitively, the first condition encourages the Markov Chain to converge towards p_d over relatively short runs (of length b). The second condition enforces that p_d is a fixed point for the transition operator.[1] Instead of simulating the chain until convergence, which will be especially time-consuming if the initial Markov chain takes many steps to mix, the generator would run only $(b+m)/2$ steps on average. Empirically, we observe better training times by uniformly sampling b from $[1, B]$ and m from $[1, M]$ respectively in each iteration, so we use B and M as the hyperparameters for our experiments.

9.3.1 Example: Generative Model for Images

We experiment with a distribution p_d over images, such as digits (MNIST) and faces (CelebA). In the experiments, we parametrize f_θ to have an autoencoding structure, where the auxiliary variable $v \sim \mathcal{N}(0, I)$ is directly added to the latent code of the network serving as a source of randomness:

$$z = \text{encoder}_\theta(x_t) \quad z' = \text{ReLU}(z + \beta v) \quad x_{t+1} = \text{decoder}_\theta(z') \tag{9.4}$$

where β is a hyperparameter we set to 0.1. While sampling is inexpensive, evaluating probabilities according to $T_\theta(\cdot|x_t)$ is generally intractable as it would require integration over v. The starting distribution π_0 is a factored Gaussian distribution with mean and standard deviation being the mean and standard deviation of the training set. We include all the details, which ares based on the DCGAN [RMC15] architecture, in Appendix B.7.4. All the models are trained with the gradient penalty objective for Wasserstein GANs [GAA+17, ACB17], where $\lambda = 1/3$, $B = 4$ and $M = 3$.

We visualize the samples generated from our trained Markov chain in Figures 9.1 and 9.3, where each row shows consecutive samples from the same chain. From Figure 9.1, it is clear that x_{t+1} is related to x_t in terms of high-level properties, such as digit identity (label). Our model learns to find

[1] We provide a more rigorous justification in Appendix B.7.2.

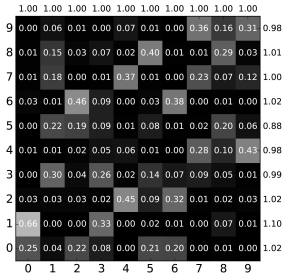


Figure 9.2: $T_\theta(y_{t+1}|y_t)$.

Figure 9.3: Samples of π_1 to π_{30} from models (top: without shortcut connections; bottom: with shortcut) trained on the CelebA dataset.

and move between the modes of the dataset, instead of generating a single sample ancestrally. This is drastically different from other iterative generative models trained with maximum likelihood, such as Generative Stochastic Networks (GSN, [BTLAY14]) and Infusion Training (IF, [BHV17]), because when we train $T_\theta(x_{t+1}|x_t)$ we are not specifying a particular target for x_{t+1}. In fact, to maximize the discriminator score the model (generator) may choose to generate some x_{t+1} near a different mode.

To further investigate the frequency of various modes in the stationary distribution, we consider the class-to-class transition probabilities for MNIST. We run one step of the transition operator starting from real samples where we have class labels $y \in \{0, \ldots, 9\}$, and classify the generated samples with a CNN. We are thus able to quantify the transition matrix for labels in Figure 9.2. Results show that class probabilities are fairly uniform and range between 0.09 and 0.11.

Although it seems that the MGAN objective encourages rapid transitions between different modes, it is not always the case. In particular, as shown in Figure 9.3, adding residual connections [HZRS16] and highway connections [SGS15] to an existing model can significantly increase the time needed to transition between modes. This suggests that the time needed to transition between modes can be affected by the architecture we choose for $f_\theta(x_t, v)$. If the architecture introduces an information bottleneck which forces the model to "forget" x_t, then x_{t+1} will have higher chance to occur in another mode; on the other hand, if the model has shortcut connections, it tends to generate x_{t+1} that are close to x_t. The increase in autocorrelation will hinder performance if samples are used for Monte Carlo estimates.

9.4 Adversarial Training for Markov Chain Monte Carlo

We now consider the setting where the target distribution p_d is specified by an analytic expression:

$$p_d(x) \propto \exp(-U(x)) \tag{9.5}$$

where $U(x)$ is a known energy function and the normalization constant in Equation (9.5) might be intractable to compute. This form is very common in Bayesian statistics [Gre95], computational physics [JM12] and graphics [LB14]. Compared to the setting in Section 9.3, there are two additional challenges:

1. We want to train a Markov chain such that the stationary distribution π_θ is *exactly* p_d;
2. We do not have direct access to samples from p_d during training.

9.4.1 Exact Sampling Through MCMC

We use ideas from the Markov Chain Monte Carlo (MCMC) literature to satisfy the first condition and guarantee that $\{\pi_\theta^t\}_{t=0}^\infty$ will asymptotically converge to p_d. Specifically, we require the transition operator $T_\theta(\cdot|x)$ to satisfy the *detailed balance* condition:

$$p_d(x)T_\theta(x'|x) = p_d(x')T_\theta(x|x') \tag{9.6}$$

for all x and x'. This condition can be satisfied using Metropolis-Hastings (MH), where a sample x' is first obtained from a *proposal distribution* $g_\theta(x'|x)$ and accepted with the following probability:

$$A_\theta(x'|x) = \min\left(1, \frac{p_d(x')}{p_d(x)}\frac{g_\theta(x|x')}{g_\theta(x'|x)}\right) = \min\left(1, \exp(U(x) - U(x'))\frac{g_\theta(x|x')}{g_\theta(x'|x)}\right) \tag{9.7}$$

Therefore, the resulting MH transition kernel can be expressed as $T_\theta(x'|x) = g_\theta(x'|x)A_\theta(x'|x)$ (if $x \neq x'$), and it can be shown that p_d is stationary for $T_\theta(\cdot|x)$ [Has70].

The idea is then to optimize for a good proposal $g_\theta(x'|x)$. We can set g_θ directly as in Equation (9.1) (if f_θ takes a form where the probability g_θ can be computed efficiently), and attempt to optimize the MGAN objective in Eq. (9.3) (assuming we have access to samples from p_d, a challenge we will address later). Unfortunately, Eq. (9.7) is not differentiable - the setting is similar to policy gradient optimization in reinforcement learning. In principle, score function gradient estimators (such as REINFORCE [Wil92]) could be used in this case; in our experiments, however, this approach leads

to extremely low acceptance rates. This is because during initialization, the ratio $g_\theta(x|x')/g_\theta(x'|x)$ can be extremely low, which leads to low acceptance rates and trajectories that are not informative for training. While it might be possible to optimize directly using more sophisticated techniques from the RL literature, we introduce an alternative approach based on volume preserving dynamics.

9.4.2 Hamiltonian Monte Carlo and Volume Preserving Flow

To gain some intuition to our method, we introduce Hamiltonian Monte Carlo (HMC) and volume preserving flow models [N+11]. HMC is a widely applicable MCMC method that introduces an auxiliary "velocity" variable v to $g_\theta(x'|x)$. The proposal first draws v from $p(v)$ (typically a factored Gaussian distribution) and then obtains (x', v') by simulating the dynamics (and inverting v at the end of the simulation) corresponding to the Hamiltonian

$$H(x, v) = v^\top v/2 + U(x) \tag{9.8}$$

where x and v are iteratively updated using the *leapfrog integrator* (see [N+11]). The transition from (x, v) to (x', v') is deterministic, invertible and volume preserving, which means that

$$g_\theta(x', v'|x, v) = g_\theta(x, v|x', v') \tag{9.9}$$

MH acceptance (9.7) is computed using the distribution $p(x, v) = p_d(x)p(v)$, where the acceptance probability is $p(x', v')/p(x, v)$ since $g_\theta(x', v'|x, v)/g_\theta(x, v|x', v') = 1$. We can safely discard v' after the transition since x and v are independent.

Let us return to the case where the proposal is parametrized by a neural network; if we could satisfy Equation 9.9 then we could significantly improve the acceptance rate compared to the "REINFORCE" setting. Fortunately, we can design such an proposal by using a volume preserving flow model [DKB14].

A flow model [DKB14, RM15, KSJ+16b, GDE17] defines a generative model for $x \in \mathbb{R}^n$ through a bijection $f : h \to x$, where $h \in \mathbb{R}^n$ have the same number of dimensions as x with a fixed prior $p_H(h)$ (typically a factored Gaussian). In this form, $p_X(x)$ is tractable because

$$p_X(x) = p_H(f^{-1}(x)) \left| \det \frac{\partial f^{-1}(x)}{\partial x} \right|^{-1} \tag{9.10}$$

and can be optimized by maximum likelihood.

In the case of a *volume preserving flow model* f, the determinant of the Jacobian $\frac{\partial f(h)}{\partial h}$ is one.

Such models can be constructed using *additive coupling layers*, which first partition the input into two parts, y and z, and then define a mapping from (y, z) to (y', z') as:

$$y' = y \qquad z' = z + m(y) \tag{9.11}$$

where $m(\cdot)$ can be an expressive function, such as a neural network. By stacking multiple coupling layers the model becomes highly expressive. Moreover, once we have the forward transformation f, the backward transformation f^{-1} can be easily derived. This family of models are called *Non-linear Independent Components Estimation* (NICE) [DKB14].

9.4.3 A NICE Proposal

HMC has two crucial components. One is the introduction of the auxiliary variable v, which prevents random walk behavior; the other is the symmetric proposal in Equation (9.9), which allows the MH step to only consider $p(x, v)$. In particular, if we simulate the Hamiltonian dynamics (the deterministic part of the proposal) twice starting from any (x, v) (without MH or resampling v), we will always return to (x, v).

Auxiliary variables can be easily integrated into neural network proposals. However, it is hard to obtain symmetric behavior. If our proposal is deterministic, then $f_\theta(f_\theta(x, v)) = (x, v)$ should hold for all (x, v), a condition which is difficult to achieve[2]. Therefore, we introduce a proposal which satisfies Eq. (9.9) for any θ, while preventing random walk in practice by resampling v after every MH step.

Our proposal considers a NICE model $f_\theta(x, v)$ with its inverse f_θ^{-1}, where $v \sim p(v)$ is the auxiliary variable. We draw a sample x' from the proposal $g_\theta(x', v'|x, v)$ using the following procedure:

1. Randomly sample $v \sim p(v)$ and $u \sim \text{Uniform}[0, 1]$;
2. If $u > 0.5$, then $(x', v') = f_\theta(x, v)$;
3. If $u \leq 0.5$, then $(x', v') = f_\theta^{-1}(x, v)$.

We call this proposal a *NICE proposal* and introduce the following theorem.

[2]The cycle consistency loss (as in CycleGAN [ZPIE17]) introduces a regularization term for this condition; we added this to the REINFORCE objective but were not able to achieve satisfactory results.

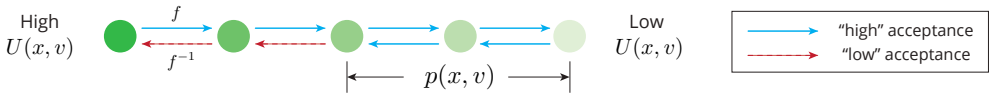


Figure 9.4: Sampling process of A-NICE-MC. Each step, the proposal executes f_θ or f_θ^{-1}. Outside the high probability regions f_θ will guide x towards $p_d(x)$, while MH will tend to reject f_θ^{-1}. Inside high probability regions both operations will have a reasonable probability of being accepted.

Theorem 14. *For any (x, v) and (x', v') in their domain, a NICE proposal g_θ satisfies*

$$g_\theta(x', v'|x, v) = g_\theta(x, v|x', v')$$

Proof. In Appendix A.7. □

9.4.4 Training A NICE Proposal

Given any NICE proposal with f_θ, the MH acceptance step guarantees that p_d is a stationary distribution, yet the ratio $p(x', v')/p(x, v)$ can still lead to low acceptance rates unless θ is carefully chosen. Intuitively, we would like to train our proposal g_θ to produce samples that are likely under the joint distribution $p(x, v)$.

Although the proposal itself is non-differentiable w.r.t. x and v, we do not require score function gradient estimators to train it. In fact, if f_θ is a bijection between samples in high probability regions, then f_θ^{-1} is automatically also such a bijection. Therefore, we ignore f_θ^{-1} during training and only train $f_\theta(x, v)$ to reach the target distribution $p(x, v) = p_d(x)p(v)$. For $p_d(x)$, we use the MGAN objective in Equation (9.3); for $p(v)$, we minimize the distance between the distribution for the generated v' (tractable through Equation (9.10)) and the prior distribution $p(v)$ (which is a factored Gaussian):

$$\min_\theta \max_D L(x; \theta, D) + \gamma L_d(p(v), p_\theta(v')) \tag{9.12}$$

where L is the MGAN objective, L_d is an objective that measures the divergence between two distributions and γ is a parameter to balance between the two factors; in our experiments, we use KL divergence for L_d and $\gamma = 1$ [3].

Our transition operator includes a trained NICE proposal followed by a Metropolis-Hastings step, and we call the resulting Markov chain *Adversarial NICE Monte Carlo* (A-NICE-MC). The

[3] The results are not very sensitive to changes in γ; we also tried Maximum Mean Discrepancy (MMD, see [LSZ15] for details) and achieved similar results.

Figure 9.5: *Left*: Samples from a model with shortcut connections trained with ordinary discriminator. *Right*: Samples from the same model trained with a pairwise discriminator.

sampling process is illustrated in Figure 9.4. Intuitively, if (x, v) lies in a high probability region, then both f_θ and f_θ^{-1} should propose a state in another high probability region. If (x, v) is in a low-probability probability region, then f_θ would move it closer to the target, while f_θ^{-1} does the opposite. However, the MH step will bias the process towards high probability regions, thereby suppressing the random-walk behavior.

9.4.5 Bootstrap

The main remaining challenge is that we do not have direct access to samples from p_d in order to train f_θ according to the adversarial objective in Equation (9.12), whereas in the case of Section 9.3, we have a dataset to get samples from the data distribution.

In order to retrieve samples from p_d and train our model, we use a bootstrap process [ET94] where the quality of samples used for adversarial training should increase over time. We obtain initial samples by running a (possibly) slow mixing operator T_{θ_0} with stationary distribution p_d starting from an arbitrary initial distribution π_0. We use these samples to train our model f_{θ_i}, and then use it to obtain new samples from our trained transition operator T_{θ_i}; by repeating the process we can obtain samples of better quality which should in turn lead to a better model.

9.4.6 Reducing Autocorrelation by Pairwise Discriminator

An important metric for evaluating MCMC algorithms is the effective sample size (ESS), which measures the number of "effective samples" we obtain from running the chain. As samples from MCMC methods are not *i.i.d.*, to have higher ESS we would like the samples to be as independent as possible (low autocorrelation). In the case of training a NICE proposal, the objective in Equation (9.3) may lead to high autocorrelation even though the acceptance rate is reasonably high. This is because the coupling layer contains residual connections from the input to the output; as shown in Section 9.3.1, such models tend to learn an identity mapping and empirically they have high autocorrelation.

We propose to use a *pairwise discriminator* to reduce autocorrelation and improve ESS. Instead

of scoring one sample at a time, the discriminator scores two samples (x_1, x_2) at a time. For "real data" we draw two independent samples from our bootstrapped samples; for "fake data" we draw $x_2 \sim T_\theta^m(\cdot|x_1)$ such that x_1 is either drawn from the data distribution or from samples after running the chain for b steps, and x_2 is the sample after running the chain for m steps, which is similar to the samples drawn in the original MGAN objective.

The optimal solution would match both distributions of x_1 and x_2 to the target distribution. Moreover, if x_1 and x_2 are correlated, then the discriminator should be able distinguish the "real" and "fake" pairs, so the model is forced to generate samples with little autocorrelation. More details are included in Appendix B.7.3. The pairwise discriminator is conceptually similar to the minibatch discrimination layer [SGZ+16]; the difference is that we provide correlated samples as "fake" data, while [SGZ+16] provides independent samples that might be similar.

To demonstrate the effectiveness of the pairwise discriminator, we show an example for the image domain in Figure 9.5, where the same model with shortcut connections is trained with and without pairwise discrimination (details in Appendix B.7.4); it is clear from the variety in the samples that the pairwise discriminator significantly reduces autocorrelation.

9.5 Experiments

Code for reproducing the experiments is available at https://github.com/ermongroup/a-nice-mc.

To demonstrate the effectiveness of A-NICE-MC, we first compare its performance with HMC on several synthetic 2D energy functions: **ring** (a ring-shaped density), **mog2** (a mixture of 2 Gaussians) **mog6** (a mixture of 6 Gaussians), **ring5** (a mixture of 5 distinct rings). The densities are illustrated in Figure 9.6 (Appendix B.7.4 has the analytic expressions). *ring* has a single connected component of high-probability regions and HMC performs well; *mog2*, *mog6* and *ring5* are selected to demonstrate cases where HMC fails to move across modes using gradient information. A-NICE-MC performs well in all the cases.

We use the same hyperparameters for all the experiments (see Appendix B.7.4 for details). In particular, we consider $f_\theta(x, v)$ with three coupling layers, which update v, x and v respectively. This is to ensure that both x and v could affect the updates to x' and v'.

How does A-NICE-MC perform? We evaluate and compare ESS and ESS per second (ESS/s) for both methods in Table 9.1. For *ring*, *mog2*, *mog6*, we report the smallest ESS of all the dimensions (as in [GC11]); for *ring5*, we report the ESS of the distance between the sample and the origin,

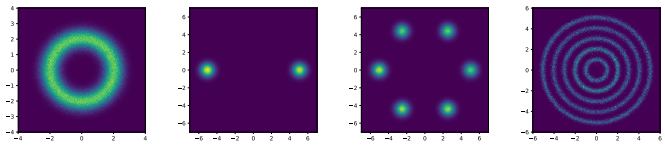


Figure 9.6: Densities of **ring**, **mog2**, **mog6** and **ring5** (from left to right).

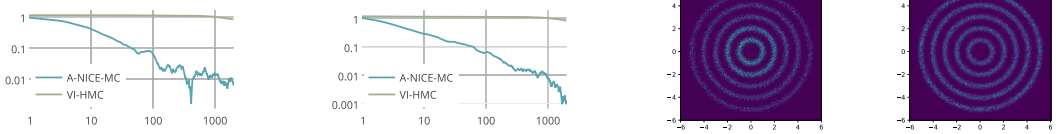
Table 9.1: **Performance of MCMC samplers as measured by Effective Sample Size (ESS).** Higher is better (1000 maximum). Averaged over 5 runs under different initializations. See Appendix B.7.1 for the ESS formulation, and Appendix B.7.4 for how we benchmark the running time of both methods.

ESS	A-NICE-MC	HMC
ring	**1000.00**	**1000.00**
mog2	**355.39**	1.00
mog6	**320.03**	1.00
ring5	**155.57**	0.43

ESS/s	A-NICE-MC	HMC
ring	**128205**	**121212**
mog2	**50409**	78
mog6	**40768**	39
ring5	**19325**	29

which indicates mixing across different rings. In the four scenarios, HMC performed well only in *ring*; in cases where modes are distant from each other, there is little gradient information for HMC to move between modes. On the other hand, A-NICE-MC is able to freely move between the modes since the NICE proposal is parametrized by a flexible neural network.

We use *ring5* as an example to demonstrate the results. We assume $\pi_0(x) = \mathcal{N}(0, \sigma^2 I)$ as the initial distribution, and optimize σ through maximum likelihood. Then we run both methods, and use the resulting particles to estimate p_d. As shown in Figures 9.7a and 9.7b, HMC fails and there is a large gap between true and estimated statistics. This also explains why the ESS is lower than 1 when we use HMC for *ring5* in Table 9.1.



(a) $\mathbb{E}[\sqrt{x_1^2 + x_2^2}]$ (b) Std$[\sqrt{x_1^2 + x_2^2}]$ (c) HMC (d) A-NICE-MC

Figure 9.7: **Evaluation for A-NICE-MC.** (a-b) Mean absolute error for estimating the statistics in *ring5* w.r.t. simulation length. Averaged over 100 chains. (c-d) Density plots for both methods. When the initial distribution is a Gaussian centered at the origin, HMC overestimates the densities of the rings towards the center.

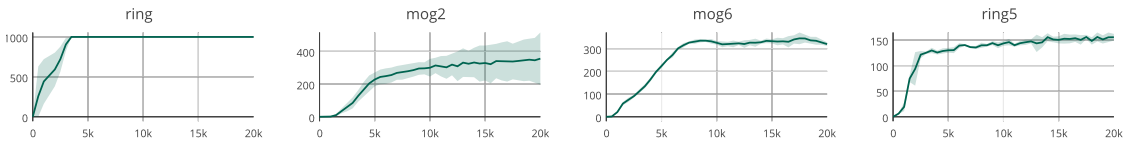


Figure 9.8: ESS with respect to the number of training iterations.

Another reasonable measurement to consider is Gelman's R hat diagnostic [BG98], which evaluates performance across multiple sampled chains. We evaluate this over the rings5 domain (where the statistics is the distance to the origin), using 32 chains with 5000 samples and 1000 burn-in steps for each sample. HMC gives a R hat value of 1.26, whereas A-NICE-MC gives a R hat value of 1.002[4]. This suggest that even with 32 chains, HMC does not succeed at estimating the distribution reasonably well.

Does training increase ESS? We show in Figure 9.8 that in all cases ESS increases with more training iterations and bootstrap rounds, which also indicates that using the pairwise discriminator is effective at reducing autocorrelation.

Admittedly, training introduces an additional computational cost which HMC could utilize to obtain more samples initially (not taking parameter tuning into account), yet the initial cost can be amortized thanks to the improved ESS. For example, in the *ring5* domain, we can reach an ESS of 121.54 in approximately 550 seconds (2500 iterations on 1 thread CPU, bootstrap included). If we then sample from the trained A-NICE-MC, it will catch up with HMC in less than 2 seconds.

Next, we demonstrate the effectiveness of A-NICE-MC on Bayesian logistic regression, where the posterior has a single mode in a higher dimensional space, making HMC a strong candidate for the task. However, in order to achieve high ESS, HMC samplers typically use many leap frog steps and require gradients at every step, which is inefficient when $\nabla_x U(x)$ is computationally expensive. A-NICE-MC only requires running f_θ or f_θ^{-1} once to obtain a proposal, which is much cheaper computationally. We consider three datasets - *german* (25 covariates, 1000 data points), *heart* (14 covariates, 532 data points) and *australian* (15 covariates, 690 data points) - and evaluate the lowest ESS across all covariates (following the settings in [GC11]), where we obtain 5000 samples after 1000 burn-in samples. For HMC we use 40 leap frog steps and tune the step size for the best ESS possible. For A-NICE-MC we use the same hyperparameters for all experiments (details in Appendix B.7.4). Although HMC outperforms A-NICE-MC in terms of ESS, the NICE proposal is less expensive to compute than the HMC proposal by almost an order of magnitude,

[4]For R hat values, the perfect value is 1, and 1.1-1.2 would be regarded as too high.

Table 9.2: ESS and ESS per second for Bayesian logistic regression tasks.

ESS	A-NICE-MC	HMC
german	926.49	**2178.00**
heart	1251.16	**5000.00**
australian	1015.75	**1345.82**

ESS/s	A-NICE-MC	HMC
german	**1289.03**	216.17
heart	**3204.00**	1005.03
australian	**1857.37**	289.11

which leads to higher ESS *per second* (see Table 9.2).

9.6 Discussion

We present a likelihood-free method to train a parametric MCMC operator with good mixing properties. The resulting Markov Chains can be used to target both empirical and analytic distributions. We showed that using our novel training objective we can leverage flexible neural networks and volume preserving flow models to obtain domain-specific transition kernels. These kernels significantly outperform traditional ones which are based on elegant yet very simple and general-purpose analytic formulas. Our hope is that these ideas will allow us to bridge the gap between MCMC and neural network function approximators, similarly to what "black-box techniques" did in the context of variational inference [RGB14].

Combining the guarantees of MCMC and the expressiveness of neural networks unlocks the potential to perform fast and accurate inference in high-dimensional domains, such as Bayesian neural networks. This would likely require us to gather the initial samples through other methods, such as variational inference, since the chances for untrained proposals to "stumble upon" low energy regions is diminished by the curse of dimensionality. Therefore, it would be interesting to see whether we could bypass the bootstrap process and directly train on $U(x)$ by leveraging the properties of flow models. Another promising future direction is to investigate proposals that can rapidly adapt to changes in the data. One use case is to infer the latent variable of a particular data point, as in variational autoencoders. We believe it should be possible to utilize meta-learning algorithms with data-dependent parametrized proposals.

Chapter 10

Learning Intentions for Multi-agent Interactions

Reinforcement learning agents are prone to undesired behaviors due to reward mis-specification. Finding a set of reward functions to properly guide agent behaviors is particularly challenging in multi-agent scenarios. Imitation learning and inverse reinforcement learning algorithms provide a framework to automatically acquire suitable policies and reward functions from expert demonstrations. Its extension to multi-agent settings, however, is difficult due to the more complex notions of rational behaviors. Hence, existing approaches are not applicable in multi-agent settings due to the existence of multiple (Nash) equilibria and non-stationary environments. This problem can be posed as inferring the underlying reward function of certain expert demonstrations.

In this chapter, we show how we can use supervised learning techniques to address this problem. First, we propose MAGAIL new framework for multi-agent imitation learning for general Markov games, where we build upon a generalized notion of inverse reinforcement learning. We further introduce a practical multi-agent actor-critic algorithm with good empirical performance. Next, we focus on settings with a large, variable number of agents and attempt to resolve these settings by exploiting similarities between agent behaviors. In particular, we learn a shared reward function using a variant of adversarial inverse reinforcement learning. This reward function is able to fit a broad array of behavior by means of a latent variable learned using a variational autoencoder. Finally, we illustrate the effectiveness of our method in multiple settings. Our methods can be used to imitate complex behaviors in high-dimensional environments with multiple cooperative or competing agents, recover reward functions that are highly correlated with ground truth ones. Our methods can also work on two large real-world datasets and instances of traffic congestion,

including cars on highways and aircraft in terminal airspace, learning reward functions in crowded environments at scale.

This chapter is based on [SRSE18, YSE19, GSKE20]. Lantao Yu and Nate Gruver contributed to the materials in this chapter. Apart from writing the paper and regular discussions of the project, my contributions include conceiving the idea, implementing the algorithms and executing the experiments for [SRSE18], providing the idea and initial code base for [YSE19], and providing the idea and dataset for [GSKE20].

10.1 Introduction

Reinforcement learning (RL) methods are becoming increasingly successful at optimizing reward signals in complex, high dimensional environments [ESM+18]. A key limitation of RL, however, is the difficulty of designing suitable reward functions for complex and not well-specified tasks [HMMA+17, AOS+16]. If the reward function does not cover all important aspects of the task, the agent could easily learn undesirable behaviors [AC16]. This problem is further exacerbated in multi-agent scenarios, such as multiplayer games [PYW+17], multi-robot control [MJMO12] and social interactions [LZL+17]; in these cases, agents do not even necessarily share the same reward function, especially in competitive settings where the agents might have conflicting rewards. In multi-agents systems, since different agents may have completely different goals and state-action representations, hand-tuning reward functions becomes increasingly more challenging as we take more agents into consideration.

Imitation learning methods address these problems via expert demonstrations [ZMBD08, ET15, FLA16, SAS17]; the agent directly learns desirable behaviors by imitating an expert. Notably, inverse reinforcement learning (IRL) frameworks assume that the expert is (approximately) optimizing an underlying reward function, and attempt to recover a reward function that rationalizes the demonstrations; an agent policy is subsequently learned through RL [NRO00, AN04]. Alternatively, because the reward function is often considered as the most succinct, robust and transferable representation of a task [AN04, FLL17], we can also consider the problem of inferring reward functions from expert demonstrations, which we refer to as inverse reinforcement learning (IRL).

Unfortunately, this paradigm is not suitable for general multi-agent settings due to environment being non-stationary to individual agents [LWT+17] and the existence of multiple equilibrium solutions [HWO98]. The optimal policy of one agent could depend on the policies of other agents, and vice versa, so there could exist multiple solutions in which each agents' policy is the optimal

response to others.

In this chapter, we propose a new framework for multi-agent imitation learning – provided with demonstrations of a set of experts interacting with each other within the same environment, we aim to learn multiple parametrized policies that imitate the behavior of each expert respectively. Using the framework of Markov games, we integrate multi-agent RL with a suitable extension of multi-agent inverse RL. The resulting procedure strictly generalizes Generative Adversarial Imitation Learning (GAIL, [HE16]) in the single agent case. Imitation learning corresponds to a two-player game between a generator and a discriminator. The generator controls the policies of all the agents in a distributed way, and the discriminator contains a classifier for each agent that is trained to distinguish that agent's behavior from that of the corresponding expert. Upon training, the behaviors produced by the policies are indistinguishable from the training data through the discriminator. We can incorporate prior knowledge into the discriminators, including the presence of cooperative or competitive agents. In addition, we propose a novel multi-agent natural policy gradient algorithm that addresses the issue of high variance gradient estimates commonly observed in reinforcement learning [LWT+17, FAdFW16]. Empirical results demonstrate that our method can imitate complex behaviors in high-dimensional environments, such as particle environments and cooperative robotic control tasks, with multiple cooperative or competitive agents; the imitated behaviors are close to the expert behaviors with respect to "true" reward functions which the agents do not have access to during training.

Furthermore, we focus on settings with a large, variable number of agents and attempt to resolve these settings by exploiting similarities between agent behaviors. In particular, we learn a shared reward function using adversarial inverse reinforcement learning and a continuous latent variable. We demonstrate our algorithm on two real-world settings: traffic on highways and in terminal airspace.

10.2 Preliminaries

10.2.1 Markov games

We consider an extension of Markov decision processes (MDPs) called Markov games [Lit94]. A Markov game (MG) for N agents is defined via a set of states $\mathcal{S}$, N sets of actions $\{\mathcal{A}_i\}_{i=1}^N$. The function $P : \mathcal{S} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_N \to \mathcal{P}(\mathcal{S})$ describes the (stochastic) transition process between states, where $\mathcal{P}(\mathcal{S})$ denotes the set of probability distributions over the set $\mathcal{S}$. Given that we are in state s_t at time t, the agents take actions $(a_1, \ldots, a_N)$ and the state transitions to s_{t+1} with

probability $P(s_{t+1}|s_t, a_1, \ldots, a_N)$.

Each agent i obtains a (bounded) reward given by a function $r_i : \mathcal{S} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_N \to \mathbb{R}$. Each agent i aims to maximize its own total expected return $R_i = \sum_{t=0}^{\infty} \gamma^t r_{i,t}$, where γ is the discount factor, by selecting actions through a (stationary and Markovian) stochastic policy $\pi_i : \mathcal{S} \times \mathcal{A}_i \to [0, 1]$. The initial states are determined by a distribution $\eta : \mathcal{S} \to [0, 1]$.

The joint policy is defined as $\boldsymbol{\pi}(a|s) = \prod_{i=1}^{N} \pi_i(a_i|s)$, where we use bold variables without subscript i to denote the concatenation of all variables for all agents (e.g. $\boldsymbol{\pi}$ denotes the joint policy $\prod_{i=1}^{N} \pi_i$ in a multi-agent setting, v denotes all rewards, $\mathbf{a}$ denotes actions of all agents).

We use expectation with respect to a policy π to denote an expectation with respect to the trajectories it generates. For example,

$$\mathbb{E}_{\pi}\left[r(s, a)\right] \triangleq \mathbb{E}_{s_t, a_t \sim \pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$$

denotes the following sample process for the right hand side: $s_0 \sim \eta$, $a_t \sim \pi(a_t|s_t)$, $s_{t+1} \sim P(s_{t+1}|a_t, s_t)$, yet if we do not take expectation over the state s, then

$$\mathbb{E}_{\pi}\left[r(s, a) + \sum_{s' \in \mathcal{S}} P(s'|s, a)v(s')\right] \triangleq \mathbb{E}_{a \sim \boldsymbol{\pi}(\cdot|s)}\left[r(s, a) + \sum_{s' \in \mathcal{S}} P(s'|s, a)v(s')\right]$$

assumes the policy samples only the next-step action a.

We use subscript $-i$ to denote *all agents except for* i. For example, (a_i, a_{-i}) represents $(a_1, \ldots, a_N)$, the actions of all N agents.

10.2.2 Reinforcement learning and Nash equilibrium

In reinforcement learning (RL), the goal of each agent is to maximize total expected return $\mathbb{E}_{\pi}[r(s, a)]$ given access to the reward signal r. In single agent RL, an optimal Markovian policy exists but the optimal policy might not be unique (e.g., all policies are optimal for an identically zero reward; see [SB98], Chapter 3.8). An entropy regularizer can be introduced to resolve this ambiguity. The optimal policy is found via the following RL procedure:

$$\mathrm{RL}(r) = \arg\max_{\pi \in \Pi} H(\pi) + \mathbb{E}_{\pi}[r(s, a)], \tag{10.1}$$

where $H(\pi)$ is the γ-discounted causal entropy [BB14] of policy $\pi \in \Pi$.

Definition 2 (γ-discounted Causal Entropy). *The γ-discounted causal entropy for a policy π is defined as follows:*

$$H(\pi) \triangleq \mathbb{E}_{\pi}[-\log \pi(a|s)] = \mathbb{E}_{s_t, a_t \sim \pi}\left[-\sum_{t=0}^{\infty} \gamma^t \log \pi(a_t|s_t)\right]$$

If we scale the reward function by any positive value, the addition of $H(\pi)$ resolves ambiguity by selecting the policy among the set of optimal policies that have the highest causal entropy[1] – the policy with both the highest reward and the highest entropy is unique because the entropy function is concave with respect to π and the set of optimal policies is convex.

In Markov games, however, the optimal policy of an agent depends on other agents' policies. One approach is to use an equilibrium solution concept, such as Nash equilibrium [HWO98]. Informally, a set of policies $\{\pi_i\}_{i=1}^N$ is a Nash equilibrium if no agent can achieve higher reward by unilaterally changing its policy, i.e. $\forall i \in [1, N], \forall \hat{\pi}_i \neq \pi_i, \mathbb{E}_{\pi_i, \pi_{-i}}[r_i] \geq \mathbb{E}_{\hat{\pi}_i, \pi_{-i}}[r_i]$. The process of finding a Nash equilibrium can be defined as a constrained optimization problem ([FV12], Theorem 3.7.2):

$$\min_{\pi \in \Pi, \boldsymbol{v} \in \mathbb{R}^{\mathcal{S} \times N}} f_r(\pi, \boldsymbol{v}) = \sum_{i=1}^{N} \left(\sum_{s \in \mathcal{S}} v_i(s) - \mathbb{E}_{a_i \sim \pi_i(\cdot|s)} q_i(s, a_i) \right) \tag{10.2}$$

$$v_i(s) \geq q_i(s, a_i) \triangleq \mathbb{E}_{\pi_{-i}}\left[r_i(s, \mathbf{a}) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, \mathbf{a}) v_i(s') \right] \quad \forall i \in [N], s \in \mathcal{S}, a_i \in \mathcal{A}_i \tag{10.3}$$

$$\mathbf{a} \triangleq (a_i, a_{-i}) \triangleq (a_1, \ldots, a_N) \qquad \boldsymbol{v} \triangleq [v_1; \ldots; v_N]$$

where the joint action $\mathbf{a}$ includes actions a_{-i} sampled from π_{-i} and a_i. Intuitively, $\boldsymbol{v}$ could represent some estimated value function for each state and $\mathbf{q}$ represents the Q-function that corresponds to $\boldsymbol{v}$. The constraints enforce the Nash equilibrium condition – when the constraints are satisfied, $(v_i(s) - q_i(s, a_i))$ is non-negative for every $i \in [N]$. Hence $f_r(\boldsymbol{\pi}, \boldsymbol{v})$ is always non-negative for a feasible $(\boldsymbol{\pi}, \boldsymbol{v})$. Moreover, this objective has a global minimum of zero if a Nash equilibrium exists, and π forms a Nash equilibrium if and only if $f_r(\boldsymbol{\pi}, \boldsymbol{v})$ reaches zero while being a feasible solution ([PB15], Theorem 2.4).

[1] For the remainder of the chapter, we may use the term "entropy" to denote the γ-discounted causal entropy for policies.

10.2.3 Inverse reinforcement learning

Suppose we do not have access to the reward signal r, but have demonstrations $\mathcal{D}$ provided by an expert (N expert agents in Markov games). Imitation learning aims to learn policies that behave similarly to these demonstrations. In Markov games, we assume all experts/players operate in the same environment, and the demonstrations $\mathcal{D} = \{(s_j, a_j)\}_{j=1}^{M}$ are collected by sampling $s_0 \sim \eta(s), \mathbf{a}_t = \pi_E(\mathbf{a}_t|s_t), s_{t+1} \sim P(s_{t+1}|s_t, \mathbf{a}_t)$; we assume knowledge of $N, \gamma, \mathcal{S}, \mathcal{A}$, as well as access to T and η as black boxes. We further assume that once we obtain $\mathcal{D}$, we cannot ask for additional expert interactions with the environment (unlike in DAgger [RGB11] or CIRL [HMRAD16]).

Let us first consider imitation in Markov decision processes (as a special case to Markov games) and the framework of single-agent Maximum Entropy IRL [ZMBD08, HE16] where the goal is to recover a reward function r that rationalizes the expert behavior π_E:

$$\text{IRL}(\pi_E) = \arg\max_{r \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}} \mathbb{E}_{\pi_E}[r(s,a)] - \left(\max_{\pi \in \Pi} H(\pi) + \mathbb{E}_{\pi}[r(s,a)] \right)$$

In practice, expectations with respect to π_E are evaluated using samples from $\mathcal{D}$.

The IRL objective is ill-defined [NRO00, FLA16] and there are often multiple valid solutions to the problem when we consider all $r \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$. For example, we can assign the reward function for trajectories that are not visited by the expert arbitrarily so long as these trajectories yields lower rewards than the expert trajectories. To resolve this ambiguity, [HE16] introduce a convex reward function regularizer $\psi : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \mathbb{R}$, which can be used to restrict rewards to be linear in a pre-determined set of features [HE16]:

$$\text{IRL}_{\psi}(\pi_E) = \arg\max_{r \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}} -\psi(r) + \mathbb{E}_{\pi_E}[r(s,a)] - \left(\max_{\pi \in \Pi} H(\pi) + \mathbb{E}_{\pi}[r(s,a)] \right) \tag{10.4}$$

10.2.4 Solution Concepts for Markov Games

A correlated equilibrium (CE) for a Markov game [ZBD11] is a joint strategy profile, where no agent can achieve higher expected reward through unilaterally changing its own policy. CE first introduced by [Aum74, Aum87] is a more general solution concept than the well-known Nash equilibrium (NE) [HWO98], which further requires agents' actions in each state to be independent, *i.e.* $\boldsymbol{\pi}(\boldsymbol{a}|s) = \Pi_{i=1}^{N} \pi_i(a_i|s)$. It has been shown that many decentralized, adaptive strategies will converge to CE instead of a more restrictive equilibrium such as NE [GGM08, HMC00]. To take bounded rationality into consideration, [MP95, MP98] further propose logistic quantal response

equilibrium (LQRE) as a stochastic generalization to NE and CE.

Definition 3. *A logistic quantal response equilibrium for Markov game corresponds to any strategy profile satisfying a set of constraints, where for each state and action, the constraint is given by:*

$$\pi_i(a_i|s) = \frac{\exp\left(\lambda \mathrm{ExpRet}_i^{\boldsymbol{\pi}}(s, a_i, \boldsymbol{a}_{-i})\right)}{\sum_{a_i'} \exp\left(\lambda \mathrm{ExpRet}_i^{\boldsymbol{\pi}}(s, a_i', \boldsymbol{a}_{-i})\right)}$$

Intuitively, in LQRE, agents choose actions with higher expected return with higher probability.

10.2.5 Imitation by matching occupancy measures

[HE16] interpret the imitation learning problem as matching two occupancy measures, i.e., the distribution over states and actions encountered when navigating the environment with a policy. Formally, for a policy π, it is defined as $\rho_{\boldsymbol{\pi}}(s, a) = \boldsymbol{\pi}(a|s) \sum_{t=0}^{\infty} \gamma^t P(s_t = s|\pi)$. [HE16] draw a connection between IRL and occupancy measure matching, showing that the former is a dual of the latter:

Proposition 4 (Proposition 3.1 in [HE16])**.**

$$\mathrm{RL} \circ \mathrm{IRL}_{\psi}(\pi_E) = \arg\min_{\pi \in \Pi} -H(\pi) + \psi^{\star}(\rho_{\pi} - \rho_{\pi_E})$$

Here $\psi^{\star}(x) = \sup_y x^{\top} y - \psi(y)$ is the convex conjugate of ψ, which could be interpreted as a measure of similarity between the occupancy measures of expert policy and agent's policy. One instance of $\psi = \psi_{\mathrm{GA}}$ gives rise to the Generative Adversarial Imitation Learning (GAIL) method:

$$\psi_{\mathrm{GA}}^{\star}(\rho_{\pi} - \rho_{\pi_E}) = \max_{D \in (0,1)^{\mathcal{S} \times \mathcal{A}}} \mathbb{E}_{\pi_E}[\log(D(s, a))] + \mathbb{E}_{\pi}[\log(1 - D(s, a))] \tag{10.5}$$

The resulting imitation learning method from Proposition 4 involves a *discriminator* (a classifier D) competing with a *generator* (a policy π). The discriminator attempts to distinguish real vs. synthetic trajectories (produced by π) by optimizing (10.5). The generator, on the other hand, aims to perform optimally under the reward function defined by the discriminator, thus "fooling" the discriminator with synthetic trajectories that are difficult to distinguish from the expert ones.

10.2.6 Adversarial Inverse Reinforcement Learning

Besides resolving the ambiguity that many optimal rewards can explain a set of demonstrations, another advantage of MaxEnt IRL is that it can be interpreted as solving the following maximum

likelihood estimation (MLE) problem:

$$p_\omega(\tau) \propto \left[\eta(s^1)\prod_{t=1}^{T} P(s^{t+1}|s^t,a^t)\right]\exp\left(\sum_{t=1}^{T} r_\omega(s^t,a^t)\right) \tag{10.6}$$

$$\max_\omega \mathbb{E}_{\pi_E}\left[\log p_\omega(\tau)\right] = \mathbb{E}_{\tau\sim\pi_E}\left[\sum_{t=1}^{T} r_\omega(s^t,a^t)\right] - \log Z_\omega$$

Here, ω are the parameters of the reward function and Z_ω is the partition function, *i.e.* an integral over all possible trajectories consistent with the environment dynamics. Z_ω is intractable to compute when the state-action spaces are large or continuous, and the environment dynamics are unknown.

Combining Guided Cost Learning (GCL) [FLA16] and generative adversarial training, [FCAL16, FLL17] proposed adversarial IRL framework as an efficient sampling based approximation to the MaxEnt IRL, where the discriminator takes on a particular form:

$$D_\omega(s,a) = \frac{\exp(f_\omega(s,a))}{\exp(f_\omega(s,a)) + q(a|s)}$$

where $f_\omega(s,a)$ is the learned function, $q(a|s)$ is the probability of the adaptive sampler pre-computed as an input to the discriminator, and the policy is trained to maximize $\log D - \log(1-D)$.

To alleviate the reward shaping ambiguity [NHR99], where many reward functions can explain an optimal policy, [FLL17] further restricted f to a reward estimator g_ω and a potential shaping function h_ϕ:

$$f_{\omega,\phi}(s,a,s') = g_\omega(s,a) + \gamma h_\phi(s') - h_\phi(s)$$

It has been shown that under suitable assumptions, g_ω and h_ϕ will recover the true reward and value function up to a constant.

10.3 Generalizing Imitation Learning to Markov games

Extending imitation learning to multi-agent settings is difficult because there are multiple rewards (one for each agent) and the notion of optimality is complicated by the need to consider an equilibrium solution [HWO98]. We use MARL(r) to denote the set of (stationary and Markovian) policies that form a Nash equilibrium under r and have the maximum γ-discounted causal entropy

(among all equilibria):

$$\begin{aligned} \text{MARL}(\mathrm{v}) = & \underset{\boldsymbol{\pi}\in\Pi, \boldsymbol{v}\in\mathbb{R}^{\mathcal{S}\times N}}{\arg\min} f_r(\boldsymbol{\pi}, \boldsymbol{v}) - H(\boldsymbol{\pi}) \\ & v_i(s) \geq q_i(s, a_i) \quad \forall i \in [N], s \in \mathcal{S}, a_i \in \mathcal{A}_i \end{aligned} \tag{10.7}$$

where q is defined as in Eq. 10.3. Our goal is to define a suitable inverse operator MAIRL, in analogy to IRL in Eq. 10.4. The key idea of Eq. 10.4 is to choose a reward that creates a *margin* between the expert and every other policy. However, the *constraints* in the Nash equilibrium optimization (Eq. 10.7) can make this challenging. To that end, we derive an equivalent Lagrangian formulation of (10.7), where we "move" the constraints into the objective function, so that we can define a margin between the expected reward of two sets of policies that captures their "difference".

10.3.1 Equivalent constraints via temporal difference learning

Intuitively, the Nash equilibrium constraints imply that any agent i cannot improve π_i via 1-step temporal difference learning; if the condition for Equation 10.3 is not satisfied for some v_i, q_i, and (s, a_i), this would suggest that we can update the policy for agent i and its value function. Based on this notion, we can derive equivalent versions of the constraints corresponding to t-step temporal difference (TD) learning.

Theorem 15. *For a certain policy $\boldsymbol{\pi}$ and reward* v*, let $\hat{v}_i(s; \boldsymbol{\pi}, \mathrm{v})$ be the unique solution to the Bellman equation:*

$$\hat{v}_i(s; \boldsymbol{\pi}, \mathrm{v}) = \mathbb{E}_\pi \left[r_i(s, \mathbf{a}) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, \mathbf{a}) \hat{v}_i(s'; \boldsymbol{\pi}, \mathrm{v}) \right] \qquad \forall s \in \mathcal{S}$$

Denote $\hat{q}_i^{(t)}(\{s^{(j)}, \mathbf{a}^{(j)}\}_{j=0}^{t-1}, s^{(t)}, a_i^{(t)}; \boldsymbol{\pi}, \mathrm{v})$ as the discounted expected return for the i-th agent conditioned on visiting the trajectory $\{s^{(j)}, \mathbf{a}^{(j)}\}_{j=0}^{t-1}, s^{(t)}$ in the first $t-1$ steps and choosing action $a_i^{(t)}$ at the t step, when other agents use policy π_{-i}:

$$\begin{aligned} & \hat{q}_i^{(t)}(\{s^{(j)}, \mathbf{a}^{(j)}\}_{j=0}^{t-1}, s^{(t)}, a_i^{(t)}; \boldsymbol{\pi}, \mathrm{v}) \\ = & \sum_{j=0}^{t-1} \gamma^j r_i(s^{(j)}, a^{(j)}) + \gamma^t \mathbb{E}_{\pi_{-i}} \left[r_i(s^{(t)}, \mathbf{a}^{(t)}) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, \mathbf{a}^{(t)}) \hat{v}_i(s'; \boldsymbol{\pi}, \mathrm{v}) \right] \end{aligned}$$

Then π is Nash equilibrium if and only if

$$\hat{v}_i(s^{(0)}; \boldsymbol{\pi}, \mathrm{v}) \geq \mathbb{E}_{\pi_{-i}}\left[\hat{q}_i^{(t)}(\{s^{(j)}, \mathbf{a}^{(j)}\}_{j=0}^{t-1}, s^{(t)}, a_i^{(t)}; \boldsymbol{\pi}, \mathrm{v})\right] \triangleq Q_i^{(t)}(\{s^{(j)}, a_i^{(j)}\}_{j=0}^{t}; \boldsymbol{\pi}, \mathrm{v}) \quad (10.8)$$
$$\forall t \in \mathbb{N}^+, i \in [N], j \in [t], s^{(j)} \in \mathcal{S}, a^{(j)} \in \mathcal{A}$$

Intuitively, Theorem 15 states that if we replace the 1-step constraints with $(t+1)$-step constraints, we obtain the same solution as MARL(r), since $(t+1)$-step TD updates (over one agent at a time) is still stationary with respect to a Nash equilibrium solution. So the constraints can be unrolled for t steps and rewritten as $\hat{v}_i(s^{(0)}) \geq Q_i^{(t)}(\{s^{(j)}, a_i^{(j)}\}_{j=0}^{t}; \boldsymbol{\pi}, \mathrm{v})$ (corresponding to Equation 10.8).

10.3.2 Multi-agent imitation learning

We are now ready to construct the Lagrangian dual of the primal in Equation 10.7, using the equivalent formulation from Theorem 15. The first observation is that for any policy $\boldsymbol{\pi}$, $f(\boldsymbol{\pi}, \hat{v}) = 0$ when $\hat{v}$ is defined as in Theorem 15 (see Lemma 10 in appendix). Therefore, we only need to consider the "unrolled" constraints from Theorem 15, obtaining the following dual problem

$$\max_{\lambda \geq 0} \min_{\pi} L_{\mathrm{v}}^{(t+1)}(\boldsymbol{\pi}, \lambda) \triangleq \sum_{i=1}^{N} \sum_{\tau_i \in \mathcal{T}_i^t} \lambda(\tau_i) \left(Q_i^{(t)}(\tau_i; \boldsymbol{\pi}, \mathrm{v}) - \hat{v}_i(s^{(0)}; \boldsymbol{\pi}, \mathrm{v})\right) \quad (10.9)$$

where $\mathcal{T}_i(t)$ is the set of all length-t trajectories of the form $\{s^{(j)}, a_i^{(j)}\}_{j=0}^{t}$, with $s^{(0)}$ as initial state, λ is a vector of $N \cdot |\mathcal{T}_i(t)|$ Lagrange multipliers, and $\hat{v}$ is defined as in Theorem 15. This dual formulation is a sum over agents and trajectories, which uniquely corresponds to the constraints in Equation 10.8.

In the following theorem, we show that for a specific choice of λ we can recover the difference of the sum of expected rewards between two policies, a performance gap similar to the one used in single agent IRL in Eq. (10.4). This amounts to "relaxing" the primal problem.

Theorem 16. *For any two policies $\boldsymbol{\pi}^\star$ and $\boldsymbol{\pi}$, let*

$$\lambda_\pi^\star(\tau_i) = \eta(s^{(0)})\pi_i(a_i^{(0)}|s^{(0)}) \prod_{j=1}^{t} \pi_i(a_i^{(j)}|s^{(j)}) \sum_{a_{-i}^{(j-1)}} P(s^{(j)}|s^{(j-1)}, a^{(j-1)})\pi_{-i}^\star(a_{-i}^{(j)}|s^{(j)})$$

be the probability of generating the sequence τ_i using policy π_i and $\pi^\star_{-i}$. Then

$$\lim_{t\to\infty} L_r^{(t+1)}(\boldsymbol{\pi}^\star, \lambda^\star_{\boldsymbol{\pi}}) = \sum_{i=1}^{N} \mathbb{E}_{\pi_i,\pi^\star_{-i}}[r_i(s,a)] - \sum_{i=1}^{N} \mathbb{E}_{\pi^\star_i,\pi^\star_{-i}}[r_i(s,a)] \tag{10.10}$$

where $L_r^{(t+1)}(\boldsymbol{\pi}^\star, \lambda^\star_{\pi})$ corresponds to the dual function where the multipliers are the probability of generating their respective trajectories of length t.

We provide a proof in Appendix A.8.3. Intuitively, the $\lambda^\star(\tau_i)$ weights correspond to the probability of generating trajectory τ_i when the policy is π_i for agent i and $\pi^\star_{-i}$ for the other agents. As $t \to \infty$, the first term of left hand side in Equation 10.10, $\sum_{i=1}^{N}\sum_{\tau_i\in\mathcal{T}_i^t} \lambda(\tau_i) Q_i^{(t)}(\tau_i)$, converges to the expected total reward $\mathbb{E}_{\pi_i,\pi^\star_{-i}}[r_i]$, which is the first term of right hand side. The marginal of $\lambda^\star$ over the initial states is the initial state distribution, so the second term of left hand side, $\sum_s \hat{v}(s)\eta(s)$, converges to $\mathbb{E}_{\pi^\star_i,\pi^\star_{-i}}[r_i]$, which is the second term of right hand side. Thus, the left hand side and right hand side of Equation 10.10 are the same as $t \to \infty$.

Theorem 16 motivates the following definition of multi-agent IRL with regularizer ψ.

$$\text{MAIRL}_\psi(\boldsymbol{\pi}_E) = \arg\max_{\text{v}} -\psi(\text{v}) + \sum_{i=1}^{N}(\mathbb{E}_{\boldsymbol{\pi}_E}[r_i]) - \left(\max_{\boldsymbol{\pi}} \sum_{i=1}^{N}(\beta H_i(\pi_i) + \mathbb{E}_{\pi_i,\pi_{E_{-i}}}[r_i])\right), \tag{10.11}$$

where $H_i(\pi_i) = \mathbb{E}_{\pi_i,\pi_{E_{-i}}}[-\log \pi_i(a_i|s)]$ is the discounted causal entropy for policy π_i when other agents follow $\pi_{E_{-i}}$, and β is a hyper-parameter controlling the strength of the entropy regularization term as in [HE16]. This formulation is a strict generalization to the single agent IRL in [HE16].

Corollary 6. *If $N = 1$, $\beta = 1$ then $\text{MAIRL}_\psi(\boldsymbol{\pi}_E) = \text{IRL}_\psi(\boldsymbol{\pi}_E)$.*

Furthermore, if the regularization ψ is additively separable, and for each agent i, π_{E_i} is the unique optimal response to other experts $\pi_{E_{-i}}$, we obtain the following:

Theorem 17. *Assume that $\psi(\text{v}) = \sum_{i=1}^{N} \psi_i(r_i)$, ψ_i is convex for each $i \in [N]$, and that $\text{MARL}(r)$ has a unique solution[2] for all $r \in \text{MAIRL}_\psi(\pi_E)$, then*

$$\text{MARL} \circ \text{MAIRL}_\psi(\pi_E) = \arg\min_{\pi\in\Pi} \sum_{i=1}^{N} -\beta H_i(\pi_i) + \psi_i^\star(\rho_{\pi_{i,E_{-i}}} - \rho_{\pi_E})$$

[2]The set of Nash equilibria is not always convex, so we have to assume $\text{MARL}(r)$ returns a unique solution.

where $\pi_{i,E_{-i}}$ denotes π_i for agent i and $\pi_{E_{-i}}$ for other agents.

The above theorem suggests that ψ-regularized multi-agent inverse reinforcement learning is seeking, for each agent i, a policy whose occupancy measure is close to one where we replace policy π_i with expert π_{E_i}, as measured by the convex function $\psi_i^\star$.

However, we do not assume access to the expert policy π_E during training, so it is not possible to obtain $\rho_{\pi_{i,E_{-i}}}$. In the settings of this chapter, we consider an alternative approach where we match the occupancy measure between ρ_{π_E} and ρ_π instead. We can obtain our practical algorithm if we select an adversarial reward function regularizer and remove the effect from entropy regularizers.

Proposition 5. *If $\beta = 0$, and $\psi(\mathrm{v}) = \sum_{i=1}^N \psi_i(r_i)$ where $\psi_i(r_i) = \mathbb{E}_{\pi_E}[g(r_i)]$ if $r_i > 0$; $+\infty$ otherwise, and*

$$g(x) = \begin{cases} -x - \log(1 - e^x) & \text{if} \quad r_i > 0 \\ +\infty & \text{otherwise} \end{cases}$$

then

$$\arg\min_\pi \sum_{i=1}^N \psi_i^\star(\rho_{\pi_i,\pi_{E_{-i}}} - \rho_{\pi_E}) = \arg\min_\pi \sum_{i=1}^N \psi_i^\star(\rho_{\pi_i,\pi_{-i}} - \rho_{\pi_E}) = \pi_E$$

Theorem 17 and Proposition 5 discuss the differences from the single agent scenario. On the one hand, in Theorem 17 we make the assumption that MARL(v) has a unique solution, which is always true in the single agent case due to convexity of the space of the optimal policies. On the other hand, in Proposition 5 we remove the entropy regularizer because here the causal entropy for π_i may depend on the policies of the other agents, so the entropy regularizer on two sides are not the same quantity. Specifically, the entropy for the left hand side conditions on $\pi_{E_{-i}}$ and the entropy for the right hand side conditions on π_{-i} (which would disappear in the single-agent case).

10.4 Practical multi-agent imitation learning

Despite the recent successes in deep RL, it is notoriously hard to train policies with RL algorithms-because of high variance gradient estimates. This is further exacerbated in Markov games since an agent's optimal policy depends on other agents [LWT+17, FAdFW16]. In this section, we address these problems and propose practical algorithms for multi-agent imitation.

10.4.1 Multi-agent generative adversarial imitation learning

We select ψ_i to be our reward function regularizer in Proposition 5; this corresponds to the two-player game introduced in Generative Adversarial Imitation Learning (GAIL, [HE16]). For each

agent i, we have a discriminator (denoted as D_{ω_i}) mapping state action-pairs to *scores* optimized to discriminate expert demonstrations from behaviors produced by π_i. Implicitly, D_{ω_i} plays the role of a reward function for the generator, which in turn attempts to train the agent to maximize its reward thus fooling the discriminator. We optimize the following objective:

$$\min_{\theta} \max_{\omega} \mathbb{E}_{\boldsymbol{\pi}_\theta} \left[\sum_{i=1}^{N} \log D_{\omega_i}(s, a_i) \right] + \mathbb{E}_{\boldsymbol{\pi}_E} \left[\sum_{i=1}^{N} \log(1 - D_{\omega_i}(s, a_i)) \right] \tag{10.12}$$

We update π_θ through reinforcement learning, where we also use a baseline V_ϕ to reduce variance. We outline the algorithm – Multi-Agent GAIL (MAGAIL) – in Appendix B.8.1.

We can augment the reward regularizer $\psi(r)$ using an indicator $y(r)$ denoting whether r fits our prior knowledge; the augmented reward regularizer $\hat{\psi} : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \mathbb{R} \cup \{\infty\}$ is then: $\psi(r)$ if $y(r) = 1$ and ∞ if $y(r) = 0$. We introduce three types of $y(r)$ for common settings.

Centralized The easiest case is to assume that the agents are fully cooperative, i.e. they share the same reward function. Here $y(r) = \mathbb{I}(r_1 = r_2 = \dots r_n)$ and $\psi(r) = \psi_{\text{GA}}(r)$. One could argue this corresponds to the GAIL case, where the RL procedure operates on multiple agents (a joint policy).

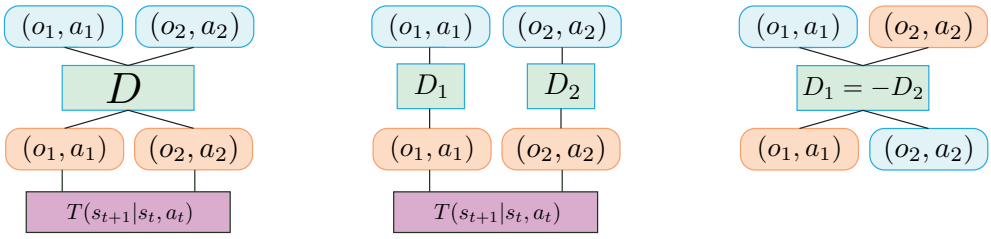
Decentralized We make no prior assumptions over the correlation between the rewards. Here $y(r) = \mathbb{I}(r_i \in \mathbb{R}^{\mathcal{O}_i\times\mathcal{A}_i})$ and $\psi_i(r_i) = \psi_{\text{GA}}(r_i)$. This corresponds to one discriminator for each agent which discriminates the trajectories as observed by agent i. However, these discriminators are not learned independently as they interact indirectly via the environment.

Zero Sum Assume there are two agents that receive opposite rewards, so $r_1 = -r_2$. As such, ψ is no longer additively separable. Nevertheless, an adversarial training procedure can be designed using the following fact:

$$v(\pi_{E_1}, \pi_2) \geq v(\pi_{E_1}, \pi_{E_2}) \geq v(\pi_1, \pi_{E_2})$$

where $v(\pi_1, \pi_2) = \mathbb{E}_{\pi_1,\pi_2}[r_1(s, a)]$ is the expected outcome for agent 1. The discriminator could maximize the reward for trajectories in (π_{E_1}, π_2) and minimize the reward for trajectories in (π_2, π_{E_1}).

These three settings are in summarized in Figure 10.1.



(a) Centralized (Cooperative) (b) Decentralized (Mixed) (c) Zero-sum (Competitive)

Figure 10.1: Different MAGAIL algorithms obtained with different priors on the reward structure. The discriminator tries to assign higher rewards to top row and low rewards to bottom row. In centralized and decentralized, the policy operates with the environment to match the expert rewards. In zero-sum, the policy do not interact with the environment; expert and policy trajectories are paired together as input to the discriminator.

10.4.2 Multi-agent actor-critic with Kronecker factors

To optimize over the generator parameters θ in Eq. (10.12) we wish to use an algorithm for multi-agent RL that has good sample efficiency in practice. Our algorithm, which we refer to as Multi-agent Actor-Critic with Kronecker-factors (MACK), is based on Actor-Critic with Kronecker-factored Trust Region (ACKTR, [WMG+17]), a state-of-the-art natural policy gradient [Ama98, Kak02] method in deep RL. MACK uses the framework of centralized training with decentralized execution [FAdFW16]; policies are trained with additional information to reduce variance but such information is not used during execution time. We let the advantage function of every agent agent be a function of all agents' observations and actions:

$$A^{\pi_i}_{\phi_i}(s, a_t) = \sum_{j=0}^{k-1} (\gamma^j r(s_{t+j}, a_{t+j}) + \gamma^k V^{\pi_i}_{\phi_i}(s_{t+k}, a_{-i,t})) - V^{\pi_i}_{\phi_i}(s_t, a_{-i,t}) \tag{10.13}$$

where $V^{\pi_i}_{\phi_i}(s_k, a_{-i})$ is the baseline for i, utilizing the additional information (a_{-i}) for variance reduction. We use (approximated) natural policy gradients to update both θ and ϕ but without trust regions to schedule the learning rate – a linear decay learning rate schedule achieves similar empirical performance.

MACK has some notable differences from Multi-Agent Deep Deterministic Policy Gradient [LWT+17]. On the one hand, MACK does not assume knowledge of other agent's policies nor tries to infer them; the value estimator merely collects experience from other agents (and treats them as black boxes). On the other hand, MACK does not require gradient estimators such as

Gumbel-softmax [JGP16] to optimize over discrete actions, which is necessary for DDPG [LHP+15].

10.5 Scaling up Multi-agent Inverse Reinforcement Learning

Accurate models of human behavior are increasingly important to safe and effective deployment of autonomous systems. Despite this need, behavior modeling remains difficult for various common problem settings. Urban environments, for example, still pose significant challenges for autonomous planning because of the uncertainty resulting from a high density of people [SAMR18]. To describe these scenarios robustly, a model must capture multi-modality in agent motivations and complex interactions that often scale super-linearly with the number of agents.

10.5.1 Scalable Modeling with Latent Variables

To address the scalability issue, we propose to model the policy and reward functions with latent variables. Specifically, we may assume the reward function for agent i can be modeled using a latent variable z^i, and is defined as $r_\phi(a_t^i, s_t|z^i)$ with parameters ϕ respectively. We further assume that the latent variable has the prior $p(z) = \mathcal{N}(\mu_z, \sigma_z)$.

With the latent variables and conditional reward model, the objective becomes:

$$\mathcal{L}(\phi) = \mathbb{E}_{\tau\sim\pi_E}\left[\sum_{i=1}^{N}\log\left(\int_{z^i} p(z^i)\frac{\exp(\sum_t r_\phi(a_t^i, s_t|z^i))}{Z(r_\phi, z^i)}\mathrm{d}z^i\right)\right] \tag{10.14}$$

To remove the summation over $p(z^i)$ in the log, we introduce a inference model $q_\omega(z^i|\tau^i)$, where $\tau^i = \{(s_t^i, a_t^i)\}$ is the trajectory for agent i. This leads to an evidence lower bound to $L(\phi)$ [KW13]:

$$\mathcal{L}(\phi) \geq \mathbb{E}_{\tau\sim\pi_E}\left[\sum_{i=1}^{N}\mathbb{E}_{z^i\sim q_\omega(z^i|\tau^i)}[\mathrm{ELBO}_{\phi,\omega}(\tau^i, z^i)]\right] \tag{10.15}$$

where $\mathrm{ELBO}_{\phi,\omega}(\tau^i, z^i)$ is defined as:

$$\sum_t r_\phi(a_t^i, s_t|z^i) - \log Z(r_\phi, z^i) - \log q_\omega(z^i|\tau^i) + \log p(z^i) \tag{10.16}$$

Given z^i, we can then optimize the first two terms in $\mathrm{ELBO}_{\phi,\omega}(\tau^i, z^i)$ with AIRL, which provides both the reward function and the corresponding policy as discussed next.

10.5.2 Multi-agent AIRL with Latent Variables

We propose an Adversarial Inverse Reinforcement Learning (AIRL, [FLL17]) algorithm that maximizes the evidence lower bound objective in Equation 10.16. First, for the latent variable model q_ω, we introduce a inference network $q_\omega(z|\tau)$ that predicts the latent variable from trajectories. From Equation 10.16, this corresponds to the following objective:

$$\mathcal{L}_{q_\omega} = -\mathbb{E}_{\tau\sim\pi_E, z\sim q_\omega(z|\tau)}[\log q_\omega(z|\tau) - \log p(z)] \tag{10.17}$$

Then, conditioned on the latent variable z, we can transform the first term of Equation 10.16 using an AIRL approach. Here we need an additional discriminator $D_{\theta,\phi}(s, a, z)$ that depends on state s, action a and the latent variable z, whose goal is to discriminate generated trajectories and the demonstrations. Specifically, one provides a parameterized policy $\pi_\theta(a|s)$, and the discriminator is denoted as:

$$D_{\theta,\phi}(s, a, z) = \frac{\exp(r_\phi(s, a|z))}{\exp(r_\phi(s, a|z)) + \pi_\theta(a|s, z)} \tag{10.18}$$

The discriminator then minimizes the following objective:

$$\begin{aligned}\mathcal{L}_D = &- \mathbb{E}_{\tau_E\sim\pi_E, z_E\sim q_\omega(z|\tau_E)}[\log D_{\theta,\phi}(s, a, z_E)] &(10.19)\\ &- \mathbb{E}_{\tau\sim\pi_\theta(z), z\sim\mathcal{N}(0,1)}[\log(1 - D_{\theta,\phi}(s, a, z))] &(10.20)\\ &- \mathbb{E}_{\hat{\tau_E}\sim\pi_\theta(z_E), z_E\sim q_\omega(z|\tau_E)}[\log(1 - D_{\theta,\phi}(s, a, z_E))] &(10.21)\end{aligned}$$

where the first term encourages higher $D_{\theta,\phi}$ for demonstrations, and the second and third term encourages lower $D_{\theta,\phi}$ for trajectories generated by the policy when the latent variables are sampled from $p(z)$ or inferred from demonstrations.

The learned policy $\pi_\theta(a|s, z)$ produces an action distribution based on the latent variable and the current state, and its primary objective is to reach higher $D_{\theta,\phi}$ values. We use $\pi_\theta(z)$ for the shorthand notation for the policy $\pi_\theta(a|s, z)$ with latent variable z. To encourage the latent variables to be informative for generating the trajectories, we add a reconstruction loss such that trajectories in τ_E could be reconstructed via $q_\omega(z|\tau_E)$ (encoder) and $\pi_\theta(a|s, z)$ (decoder), similar to InfoGAIL [LSE17a]. This leads to the objective:

$$\mathcal{L}_G = -\mathbb{E}_{\tau\sim\pi_\theta(z), z\sim p(z)}[D_{\theta,\phi}(s, a, z)] + \mathbb{E}_{\hat{\tau}_E\sim\pi_\theta(z_E), z_E\sim q_\omega(z|\tau_E), \tau_E\sim\pi_E}[\|\hat{\tau_E} - \tau_E\|_2] \tag{10.22}$$

These loss functions are iteratively minimized with stochastic gradient descent.

10.6 Experiments

We evaluate the performance of (centralized, decentralized, and zero-sum versions) of MAGAIL under two types of environments. One is a particle environment which allows for complex interactions and behaviors; the other is a control task, where multiple agents try to cooperate and move a plank forward. We collect results by averaging over 5 random seeds. Our implementation is based on OpenAI baselines [DHK+17]; please refer to Appendix B.8.2 for implementation details.

We compare our methods (centralized, decentralized, zero-sum MAGAIL) with two baselines. The first is behavior cloning (BC), which learns a maximum likelihood estimate for a_i given each state s and does not require actions from other agents. The second baseline is the GAIL IRL baseline that operates on each agent separately – for each agent we first pretrain the other agents with BC, and then train the agent with GAIL; we then gather the trained GAIL policies from all the agents and evaluate their performance.

10.6.1 Particle environments

We first consider the particle environment proposed in [LWT+17], which consists of several agents and landmarks. We consider two cooperative environments and two competitive ones. All environments have an underlying true reward function that allows us to evaluate the performance of learned agents.

The environments include: **Cooperative Communication** – two agents must cooperate to reach one of three colored landmarks. One agent ("speaker") knows the goal but cannot move, so it must convey the message to the other agent ("listener") that moves but does not observe the goal. **Cooperative Navigation** – three agents must cooperate through physical actions to reach three landmarks; ideally, each agent should cover a single landmark. **Keep-Away** – two agents have contradictory goals, where agent 1 tries to reach one of the two targeted landmarks, while agent 2 (the adversary) tries to keep agent 1 from reaching its target. The adversary does not observe the target, so it must act based on agent 1's actions. **Predator-Prey** – three slower cooperating adversaries must chase the faster agent in a randomly generated environment with obstacles; the adversaries are rewarded by touching the agent while the agent is penalized.

For the cooperative tasks, we use an analytic expression defining the expert policy; for the competitive tasks, we use MACK to train expert policies based on the true underlying rewards

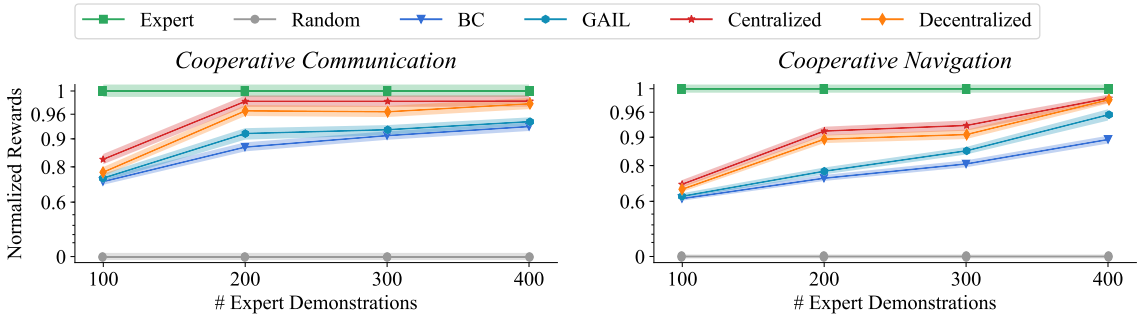


Figure 10.2: Average true reward from cooperative tasks. Performance of experts and random policies are normalized to one and zero respectively. We use inverse log scale for better comparison.

(using larger policy and value networks than the ones that we use for imitation). We then use the expert policies to simulate trajectories $\mathcal{D}$, and then do imitation learning on $\mathcal{D}$ as demonstrations, where we assume the underlying rewards are unknown. Following [LSE17b], we pretrain our Multi-Agent GAIL methods and the GAIL baseline using behavior cloning as initialization to reduce sample complexity for exploration. We consider 100 to 400 episodes of expert demonstrations, each with 50 timesteps, which is close to the amount of timesteps used for the control tasks in [HE16]. Moreover, we randomly sample the starting position of agent and landmarks each episode, so our policies have to learn to generalize when they encounter new settings.

Cooperative tasks

We evaluate performance in cooperative tasks via the average expected reward obtained by all the agents in an episode. In this environment, the starting state is randomly initialized, so generalization is crucial. We do not consider the zero-sum case, since it violates the cooperative nature of the task. We display the performance of centralized, decentralized, GAIL and BC in Figure 10.2.

Naturally, the performance of BC and MAGAIL increases with more expert demonstrations. MAGAIL performs consistently better than BC in all the settings; interestingly, in the cooperative communication task, centralized MAGAIL is able to achieve expert-level performance with only 200 demonstrations, but BC fails to come close even with 400 trajectories. Moreover, the centralized MAGAIL performs slightly better than decentralized MAGAIL due to the better prior, but decentralized MAGAIL still learns a highly correlated reward between two agents.

Table 10.1: Average agent rewards in competitive tasks. We compare behavior cloning (BC), GAIL (G), Centralized (C), Decentralized (D), and Zero-Sum (ZS) methods. Best marked in bold (high vs. low rewards is preferable depending on the agent vs. adversary role).

Task	Predator-Prey								
Agent	Behavior Cloning					G	C	D	ZS
Adversary	BC	G	C	D	ZS	Behavior Cloning			
Rewards	-93.20	-93.71	-93.75	-95.22	**-95.48**	-90.55	-91.36	**-85.00**	-89.4
Task	Keep-Away								
Agent	Behavior Cloning					G	C	D	ZS
Adversary	BC	G	C	D	ZS	Behavior Cloning			
Rewards	24.22	24.04	23.28	23.56	**23.19**	26.22	26.61	**28.73**	27.80

Competitive tasks

We consider all three types of Multi-Agent GAIL (centralized, decentralized, zero-sum) and BC in both competitive tasks. Since there are two opposing sides, it is hard to measure performance directly. Therefore, we compare by letting (agents trained by) BC play against (adversaries trained by) other methods, and vice versa. From Table 10.1, decentralized and zero-sum MAGAIL often perform better than centralized MAGAIL and BC, which suggests that the selection of the suitable prior $\hat{\psi}$ is important for good empirical performance. More details for all the particle environments are in the appendix.

10.6.2 Cooperative control with MA-GAIL

In some cases we are presented with sub-optimal expert demonstrations because the environment has changed; we consider this case in a cooperative control task [KGE17], where N bipedal walkers cooperate to move a long plank forward; the agents have incentive to collaborate since the plank is much longer than any of the agents. The expert demonstrates its policy on an environment with no bumps on the ground and heavy weights, while we perform imitation in an new environment with bumps and lighter weights (so one is likely to use too much force). Agents trained with BC tend to act more aggressively and fail, whereas agents trained with centralized MAGAIL can adapt to the new environment. With 10 (imperfect) expert demonstrations, BC agents have a chance of failure of 39.8% (with a reward of 1.26), while centralized MAGAIL agents fail only 26.2% of the time (with a reward of 26.57). We show videos of respective policies in the supplementary.

10.6.3 Transportation Environments

In this section, we consider multi-agent imitation learning and inverse reinforcement learning in transportation environments where scalability is crucial.

Deterministic Transition Model

In the transportation environments, we assume that the state s_t contains the location $\vec{l}$ and velocities $\vec{v}$ of all agents:

$$s_t = \{s_t^i\}$$
$$s_t^i = \begin{cases} (\vec{l}_t, \vec{v}_t)^i & \mathcal{O}_t^i = \mathcal{O}[a] \\ s_{\mathcal{O}} & \text{else} \end{cases}$$

where we consider $\vec{l}, \vec{v} \in \mathbb{R}^2$ for car trajectories (2d) and $\vec{l}, \vec{v} \in \mathbb{R}^3$ for airplane trajectories (3d). We assume that the action for each agent is to change its location and velocity (denoted as $\Delta\vec{l}$ and $\Delta\vec{v}$ respectively). The extra null state $s_{\mathcal{O}}$ and action $a_{\mathcal{O}}$ are taken to be values far from the data distribution such as the largest number possible in 64-bit floating point. The transition model of alive agents is defined as

$$T'((\vec{l}_{t+1}, \vec{v}_{t+1})^i \mid (\vec{l}_t,\ \vec{v}_t)^i,\ (\Delta\vec{l},\ \Delta\vec{v})_t^i) = (\vec{l}_t + (\Delta\vec{l})_t,\ \vec{v}_t + (\Delta\vec{v})_t)^i$$

which is deterministic. Learning $\Delta\vec{l}$ as well as $\Delta\vec{v}$ (instead of using second order ODEs) is useful in this case because it compensates for temporal downsampling of the data and discretization of time.

Data

HighD Dataset The HighD dataset [KBKE18] comprises over 100,000 short vehicle trajectories gathered from drone footage of German highways. We used the locations with 3 lanes (majority of dataset), and preprocessed by making all velocities positive and rescaling and shifting to a shared coordinate system. As lane markings differed slightly due to drone camera angle, we took the true location to be the average across all the recordings after scaling. To aid our optimization objective, we also downsampled the trajectories without lane changes so that the resulting trajectories were 30% lane changes. This prevented the highly biased dataset from biasing the generative model in the early phases of its training. Additionally, for convenience, we aggressively downsampled the

trajectories in time, sacrificing a smoother dynamics model for a smaller computational burden.

As vehicles are not point particles in the HighD dataset and have lateral dimensions, we found training could be accelerated by including the relative locations of the edges making up each of the two vehicle's bounding boxes as inputs to the state encoder. Though not strictly necessary, this proved to aid convergence.

FAA Dataset The aircraft trajectory dataset consists of tracks gathered from the Federal Aviation Administration (FAA) multi-sensor fusion tracker [JSV08]. The dataset contains six months of flights from locations in Central Florida, New York City, and Southern California, though for this work we only used flights to and from JFK airport in New York City. Raw tracks contain the lat long and altitude of each aircraft. These (x, y, z) points are smoothed using the optimization procedure described in [BKB18] and approximate velocities are calculated by temporal differences. All trajectories are bounded at 40 km ($\sim$25 miles) from the airport laterally and 3 km ($\sim$10000 ft) in altitude. As with the HighD vehicle trajectories, we also temporally downsampled the aircraft trajectories which in their raw form often had sequence lengths on the order of hundreds or thousands. In order to provide as much contextual information to the model as possible, we also joined trajectory data with hourly historical weather data from the NOAA, which included wind speed, wind direction, and visibility, among other features.

Results

Learned Policies We used two types of metrics to evaluated the policies learned by our models. To measure how effectively the learned distribution can cover the expert distribution, we calculate the average and final displacement between sampled trajectories drawn from a particular starting state $s^i_{t_0}$ and the ground truth trajectory, holding the policies of the other agents fixed as expert. The reported number is the minimum over 10 sampled latent variables. Additionally, we measure "emergent" properties of the learned policies in the setting of multi-agent control. These emergent properties include the distribution of speeds, distance between agents, and rate of anomalous events (such as driving off the road or going to zero altitude far from the airport). For the HighD data distance between agents is reported as distance headway (DHW)–the distance to the preceding car in the lane if it exists–and time-to-collision (TTC) as these are the typical metrics used in driving simulation. The benchmark our models, we compare them with the policy with the LSTM removed as well as our full model with the latent variable z removed. Results for the HighD and FAA datasets are shown in Tables 10.2 and 10.3. From these ablations, *we see that the proposed*

Table 10.2: Results on the HighD dataset.

	HighD			
metric	w/o lstm & latent	w/o latent	w/ latent	expert
avg. displacement (m)	5.89	5.03	**4.93**	
final displacement (m)	6.57	5.68	**5.61**	
min speed (m/s)	22.4	21.6	23.7	21.2
max speed (m/s)	44.8	40.3	37.2	39.5
rate bad final state (%)	2.6	1.15	1.24	0.0
avg. DHW (m)	51.3	58.2	63.8	56.7
avg. TTC (s)	143	152	168	159

Table 10.3: Results on the FAA dataset.

	FAA			
metric	w/o lstm & latent	w/o latent	w/ latent	expert
avg. displacement (m)	823	670	**596**	
final displacement (m)	897	808	**785**	
min speed (m/s)	58.7	61.0	59.3	64.8
max speed (m/s)	310	296	302	287
rate bad final state (%)	7.52	6.01	5.32	0.0
avg. inter-agent dist. (m)	28000	26100	25200	24300

method is able to achieve low displacement from expert trajectory and emergent properties similar to those of expert trajectories. The rate of bad final states is perhaps the most relevant among these emergent properties and this is minimal with the full model on both datasets.

As the learned latent variables are a focus of this work, we show visualizations of rollouts where latent variables are drawn from across the distribution. Figure 10.3 shows a distribution of highway driving trajectories. This visualization illustrates the primary role of the latent variable for the HighD dataset is modeling lane-changing behavior. Figure 10.4 shows a distribution of takeoff trajectories out of JFK airport. Here we see the latent variable captures the shape of the trajectory, which contains information about direction and altitude changes[3].

Learned Reward Functions

Verifying learned reward function in an IRL setting is challenging in general. In our setting, the

[3]Video demonstrations of our learned policies in http://bit.ly/multi-agent-traffic.

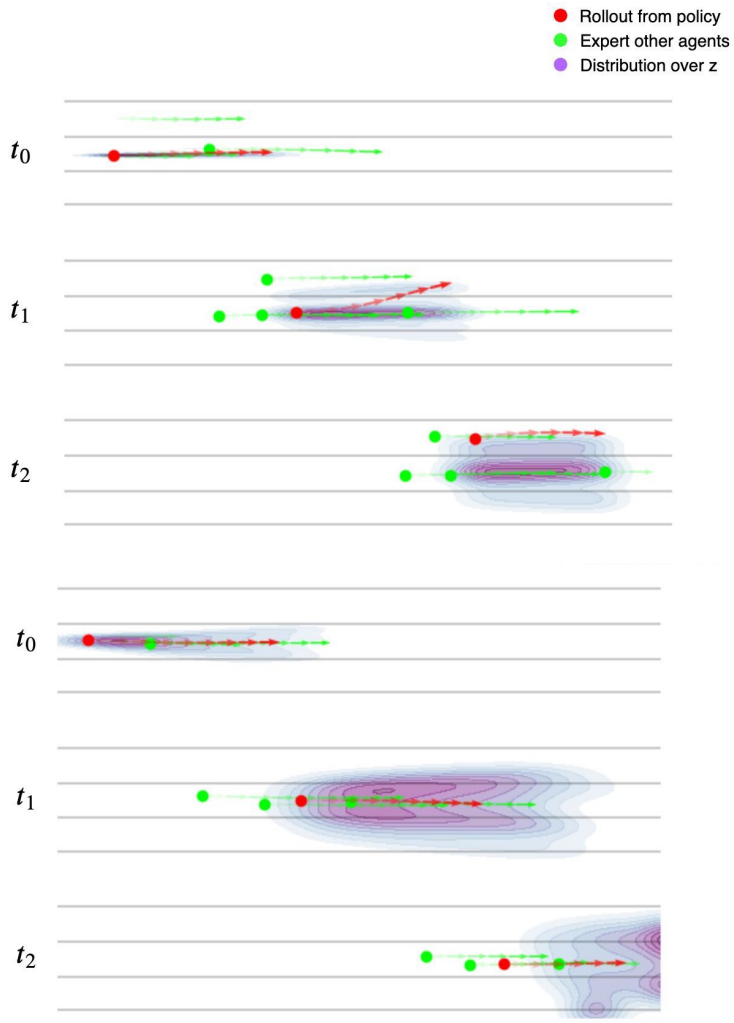


Figure 10.3: Expert trajectories are shown in green while a rollout for one value of z is shown in red. A full distribution of positions from trajectories resulting from many $z \sim \mathcal{N}(0, 1)$ are shown as a density. The latent variable primarily captures the driver's willingness to change lanes, with concentrations of density moving out to the adjacent lanes as well as behind the preceding car. Agents further away from the rolled out agent are not shown for visual clarity.

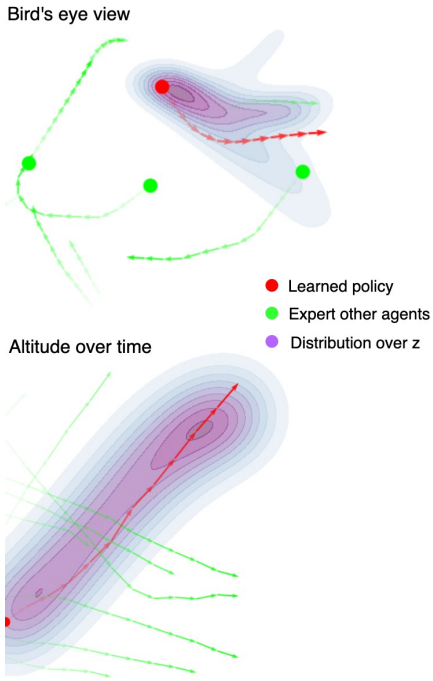


Figure 10.4: Expert trajectories are shown in green while a rollout for one value of z is shown in red. A full distribution of positions from trajectories resulting from many $z \sim \mathcal{N}(0, 1)$ are shown as a density. Here the latent variable captures the path of ascent chosen by the pilot.

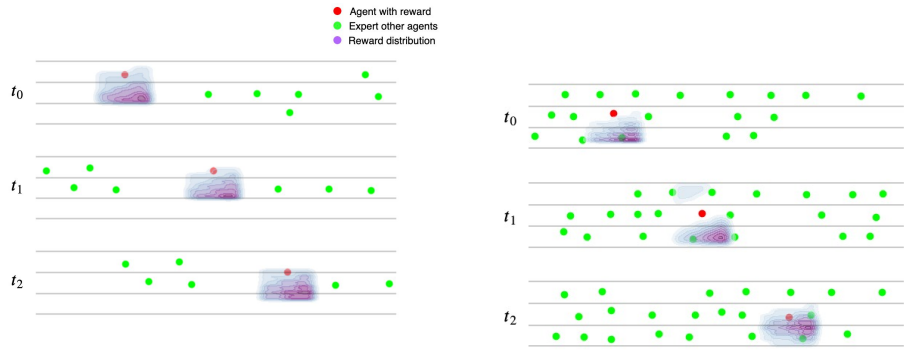


Figure 10.5: Visualization of a learned reward function when the state of the vehicle is rolled forward in time. Cars appear closer horizontally than they are in real life, as the true aspect ratio of the road is much higher.

challenge remains because we do not have access to any ground truth rewards. Luckily, however, visualization in this setting is feasible and human intuition can be used for qualitatively validating the rewards. We visualize learned rewards by plotting the distribution of rewards around a vehicle. Specifically, we calculate the rewards over a grid of possible locations surrounding the vehicle, leaving the locations of the other vehicles fixed. Rewards are normalized between zero and one and the resulting density is smoothed with a Gaussian kernel.

Figure 10.5 shows two visualizations of reward functions learned from the HighD dataset. Both reward functions reflect a lane-change objective–one in less congestion and another in a higher level of congestion where rewards are more affected by the locations of other agents. Figure 10.6 shows a learned reward function on the FAA dataset. We see a similar reward function that reflects near-term navigational goals. In general, reward functions learned from the FAA dataset show less dramatic dependence on other agents, as we would expect given the implicit effect of terminal air traffic control.

10.7 Related work and discussion

There is a vast literature on single-agent imitation learning [Bag15]. Behavior Cloning (BC) learns the policy through supervised learning [Pom91]. Inverse Reinforcement Learning (IRL) assumes the expert policy optimizes over some unknown reward, recovers the reward, and learns the policy through reinforcement learning (RL). BC does not require knowledge of transition probabilities or access to the environment, but suffers from compounding errors and covariate shift [RAB10, RGB11].

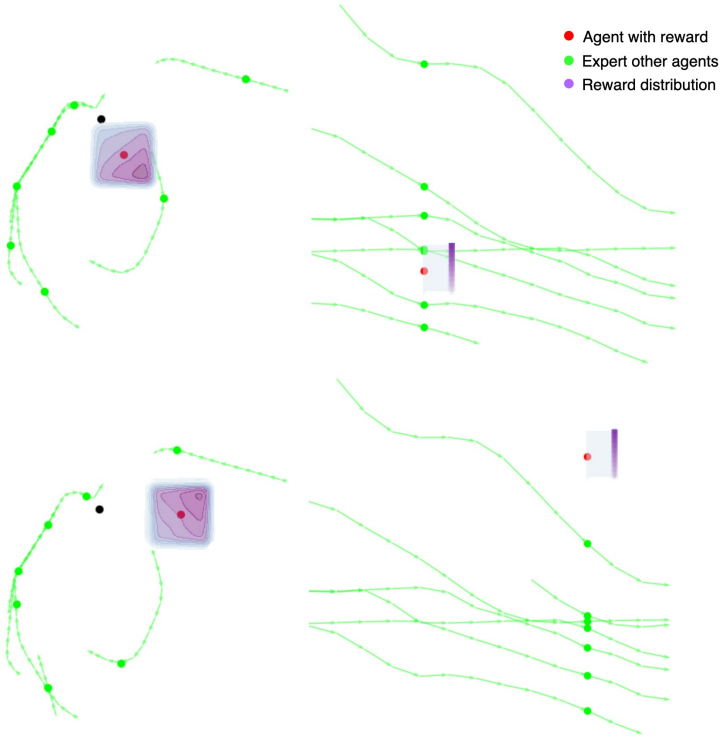


Figure 10.6: Visualization of reward function learned from the FAA dataset. Rewards are clearly correlated with the flight path of the pilot, in this case a looping ascent to the NE.

Most existing work in multi-agent imitation learning assumes the agents have very specific reward structures. The most common case is fully cooperative agents, where the challenges mainly lie in other factors, such as unknown role assignments [LYC17], scalability to swarm systems [ŠKZK16] and agents with partial observations [BD14]. In non-cooperative settings, [LBC14] consider the case of IRL for two-player zero-sum games and cast the IRL problem as Bayesian inference, while [RGZH12] assume agents are non-cooperative but the reward function is a linear combination of pre-specified features.

The use of latent variables to learn multi-modal policies have been considered in single agent imitation learning, such as learning multi-modal policies [LSE17a] or meta-learning [YYFE19]. Our approach makes two distinctions from these previous ones: different from [LSE17a], our latent variables are continuous since there is no explicit prior about the number of different policies; different from [YYFE19], we assume a prior distribution on the latent variable so as to generate new policies unconditionally. These differences allow us to generate diverse multi-agent behaviors within the same environment; using latent variables would also decrease sample complexity as opposed to learning the policy of each agent separately.

Experimental results demonstrate that our methods are able to imitate complex behaviors and learn suitable reward functions in high-dimensional environments with both cooperative and adversarial interactions. An interesting research direction is to explore new techniques for gathering expert demonstration; for example, when the expert is allowed to aid the agents by participating in part of the agent's learning process [HMRAD16].

Chapter 11

Conclusions

11.1 Summary of Contributions

The motivating theme of this dissertation was to develop artificial intelligence (AI) agents that adapts to our various needs in complex, high-dimensional, non-stationary, real-world environments. In particular, we focus on machine learning (ML)-based AI systems that leverages data, models and computational systems. A key challenge in current ML systems is the need for supervision, which does not apply well to cases where supervision signals are difficult to capture – these cases include compression, generation, and inference. To address the unique problems in these scenarios, we developed theory and algorithms that applies supervised learning techniques and implemented solutions to various application domains. In the following, we summarize the contribution within each of these parts.

In **Part I**, we discuss the problem of compression (unsupervised representation learning) and frame the problem within the context of variational mutual information estimation / maximization. We start by analyzing variational mutual information objectives for representation learning, discuss their limitations, and introduce a low-variance estimator of mutual information (Chapter 2). We then follow up by introducing an estimator based on multi-label classification that can also be optimized efficiently for representation learning (Chapter 3). Finally, we apply the above ideas and consider information-theoretical approaches to learning fair representations, which uses regression to estimate and optimize mutual information (Chapter 4). These approaches have been applied to learning informative and fair representations from data.

In **Part II**, we discuss how supervised learning can be advanced for generation, *i.e.*, learning high-dimensional probabilistic models of data (generation). First, we start by improving generative

adversarial networks (GANs) by reweighting the supervised learning objective. We proposed a unified objective for f-GANs and Wasserstein GANs, which allows us to interpret them as objective functions that can be reweighted for better generative modeling performance. Next, we improve GANs by introducing a novel negative data augmentation technique. This introduces negative samples into the supervised learning procedure of training GANs, and is able to encourage GAN discriminators to capture additional global information of the data. Negative data augmentation can also be used the compression algorithms that we discussed in Part I. Finally, we discuss how we can incorporate compression techniques from the previous part to enable few-shot conditional generation capabilities of generative models. We introduced an efficient sampling procedure for diffusion generative models, applied it to the latent space of variational autoencoders, which allows us to learn high-fidelity generative models with informative latent spaces. Such a latent space can be used for various few-shot conditional generation tasks, such as conditional image generation and image manipulation.

In **Part III** we discuss how supervised learning can be used to improve probabilistic inference methods. In Chapter 8, we discuss how supervised learning is used to automate design choices in Bayesian inference that were previously hand-designed. We replace hand-designed heuristics with automated supervised learning algorithms, allowing us to vastly improve the efficiency of Bayesian inference algorithms. In Chapter 9, we apply supervised learning to multi-agent imitation learning and inverse reinforcement learning, where machines can imitate the behavior of real-world multi-agent demonstrations while learning flexible reward functions from them. We demonstrated our methods on a range of environments, such as cooperative robotic control, road traffic, and air traffic.

11.2 Future Work

The contributions of this dissertation suggest new opportunities and challenges for future work, some of which we highlight below.

Flexible, hierarchical representation learning In representation learning, the optimal level of abstraction is task-dependent: information about color and texture is crucial if our goal is to reproduce an image, but much less so if our goal is to merely recognize objects. Most existing works on unsupervised representation learning only have a single level of abstraction that is "biased" towards certain tasks, limiting the ability to adapt to novel ones. I believe that the next

generation of intelligent systems learning from real-world data should form a flexible hierarchy of abstractions that is suited to more general tasks and adapt them to novel tasks. Having more flexibility in the representations also opens up the possibility to incorporate structured domain knowledge, which we have explored in the context of structured prediction [RSS+18]. Learning flexible representations can also lead to new computational tools that can facilitate application areas where high quality supervision is scarce, such as generative modeling, scientific discovery, and computational sustainability.

Generative modeling with conditions As intelligent systems grow increasingly complicated, interpreting them becomes more challenging, so it is crucial for these systems to be able to represent their knowledge in more interpretable ways, such as images, text, or graphs. These generative models should also produce high-dimensional signals conditioned on our In recent works, I explored new paradigms to training deep neural networks that accurately and efficiently model high-dimensional data from few-shot conditions (Chapter 8). Combining this with other high-fidelity generative models can allow humans to better understand the model and provide more accurate feedback. This also includes pressing concerns in using generative models for fair generative modeling, as we show in Chapter 5 and hopefully will extend to conditional few-shot generation in future research.

Data-driven sequential decision making Much practical progress in machine learning over the past few years have been driven by advances in large-scale datasets. However, in sequential decision making problems with human interactions (*e.g.*, autonomous driving and dialog systems), accurate real-world data are often costly, and cheaper sources of data (*e.g.*, simulators) are often inaccurate. Bridging the gap between these two sources of data could benefit real-world problems that are difficult to describe with human labels; this requires breakthroughs in many relevant fields such as causal inference, uncertainty quantification, generative modeling, and domain adaptation. One example is our work on re-calibration techniques for model-based reinforcement learning [MKS+19], which improves policy learning through models that better quantify uncertainty. I believe that we can learn from precise yet expensive real-world data to better teach our models under massive, cheap sources of data, and make progress in many real-world domains where machines interact with humans.

Appendix A

Proofs

A.1 Proofs for Chapter 3

A.1.1 Proofs in Section 3.3

Theorem 1. *$\forall P, Q \in \mathcal{P}(\mathcal{X})$ such that $P \ll Q$ we have*

$$D_{\mathrm{KL}}(P\|Q) = \sup_{r \in \Delta(Q)} \mathbb{E}_P[\log r] \tag{3.6}$$

where the supremum is achived when $r = \mathrm{d}P/\,\mathrm{d}Q$.

Proof. For every $T \in L^\infty(Q)$, define $r_T = \frac{e^T}{\mathbb{E}_Q[e^T]}$, then $r_T \in \Delta(Q)$ and from the Donsker-Varadhan inequality [DV75]

$$D_{\mathrm{KL}}(P\|Q) = \sup_{T \in L^\infty(Q)} \mathbb{E}_P[T] - \log \mathbb{E}_Q[e^T] \tag{A.1}$$

$$= \sup_{T \in L^\infty(Q)} \mathbb{E}_P\left[\log \frac{e^T}{\mathbb{E}_Q[e^T]}\right] = \sup_{r_T \in \Delta(Q)} \mathbb{E}_P[\log r_T] \tag{A.2}$$

Moreover, we have:

$$D_{\mathrm{KL}}(P\|Q) = \mathbb{E}_P[\log \mathrm{d}P - \log \mathrm{d}Q] = \mathbb{E}_P\left[\log \frac{\mathrm{d}P}{\mathrm{d}Q}\right] \tag{A.3}$$

which completes the proof. □

Corollary 7. *$\forall P, Q \in \mathcal{P}(\mathcal{X})$ such that $P \ll Q, \forall f_\theta : \mathcal{X} \to \mathbb{R}_{\geq 0}$ we have*

$$I(X;Y) \geq I_{\mathrm{CPC}}(f_\theta) := \mathbb{E}_{P^n(X,Y)}\left[\frac{1}{n}\sum_{i=1}^{n} \log \frac{f_\theta(\boldsymbol{x}_i, \boldsymbol{y}_i)}{\frac{1}{n}\sum_{j=1}^{n} f_\theta(\boldsymbol{x}_i, \boldsymbol{y}_j)}\right] \tag{A.4}$$

Proof.

$$nI_{\mathrm{CPC}}(f_\theta) := \mathbb{E}_{P^n(X,Y)}\left[\sum_{i=1}^{n} \log \frac{f_\theta(\boldsymbol{x}_i, \boldsymbol{y}_i)}{\frac{1}{n}\sum_{j=1}^{n} f_\theta(\boldsymbol{x}_i, \boldsymbol{y}_j)}\right] \tag{A.5}$$

$$= \mathbb{E}_{P^n(X,Y)}\left[\sum_{i=1}^{n} \log \frac{nf_\theta(\boldsymbol{x}_i, \boldsymbol{y}_i)}{\sum_{j=1}^{n} f_\theta(\boldsymbol{x}_i, \boldsymbol{y}_j)}\right] \tag{A.6}$$

Since

$$\mathbb{E}_{P(X)P^n(Y)}\left[\frac{nf_\theta(\boldsymbol{x}, \boldsymbol{y})}{\sum_{j=1}^{n} f_\theta(\boldsymbol{x}, \boldsymbol{y}_j)}\right] = 1, \tag{A.7}$$

we can apply Theorem 1 to obtain:

$$nI_{\mathrm{CPC}}(f_\theta) = \mathbb{E}_{P^n(X,Y)}\left[\sum_{i=1}^{n} \log \frac{nf_\theta(\boldsymbol{x}_i, \boldsymbol{y}_i)}{\sum_{j=1}^{n} f_\theta(\boldsymbol{x}_i, \boldsymbol{y}_j)}\right] \tag{A.8}$$

$$= \sum_{i=1}^{n} \mathbb{E}_{P(X_i, Y_1^n)}\left[\log \frac{nf_\theta(\boldsymbol{x}_i, \boldsymbol{y}_i)}{\sum_{j=1}^{n} f_\theta(\boldsymbol{x}_i, \boldsymbol{y}_j)}\right] \tag{A.9}$$

$$\leq \sum_{i=1}^{n} I(X_i; Y_1^n) = nI(X;Y) \tag{A.10}$$

where Y_1^n denotes the concatenation of n independent random variables $(Y_1, \ldots, Y_n)$ and

$$P(X_i, Y_1^n) = P(X_i, Y_i)P(Y_1^{i-1})P(Y_{i+1}^n)$$

is the joint distribution of $P(X_i, Y_1^n)$. □

A.1.2 Proofs in Section 3.4

Theorem 2. *Assume that the ground truth density ratio $r^\star = \mathrm{d}P/\mathrm{d}Q$ and $\mathrm{Var}_Q[r^\star]$ exist. Let Q_n denote the empirical distribution of n i.i.d. samples from Q and let $\mathbb{E}_{Q_n}$ denote the sample average*

over Q_n. Then under the randomness of the sampling procedure, we have:

$$\mathrm{Var}_Q[\mathbb{E}_{Q_n}[r^\star]] \geq \frac{e^{D_{\mathrm{KL}}(P\|Q)} - 1}{n} \tag{3.10}$$

$$\lim_{n\to\infty} n \,\mathrm{Var}_Q[\log \mathbb{E}_{Q_n}[r^\star]] \geq e^{D_{\mathrm{KL}}(P\|Q)} - 1. \tag{3.11}$$

Proof. Consider the variance of $r^\star(\boldsymbol{x})$ when $\boldsymbol{x} \sim Q$:

$$\mathrm{Var}_Q[r^\star] = \mathbb{E}_Q\left[\left(\frac{\mathrm{d}P}{\mathrm{d}Q}\right)^2\right] - \left(\mathbb{E}_Q\left[\frac{\mathrm{d}P}{\mathrm{d}Q}\right]\right)^2 \tag{A.11}$$

$$= \mathbb{E}_P\left[\frac{\mathrm{d}P}{\mathrm{d}Q}\right] - 1 \tag{A.12}$$

$$\geq e^{\mathbb{E}_P[\log \frac{\mathrm{d}P}{\mathrm{d}Q}]} - 1 \tag{A.13}$$

$$= e^{D_{\mathrm{KL}}(P\|Q)} - 1 \tag{A.14}$$

where (A.11) uses the definition of variance, (A.12) uses the definition of Radon-Nikodym derivative to change measures, (A.13) uses Jensen's inequality over $\log$, and (A.14) uses the definition of KL divergences.

The variance of the mean of n i.i.d. random variables then gives us:

$$\mathrm{Var}_Q[\mathbb{E}_{Q_n}[r]] = \frac{\mathrm{Var}[r]}{n} \geq \frac{e^{D_{\mathrm{KL}}(P\|Q)} - 1}{n} \tag{A.15}$$

which is the first part of the theorem.

As $n \to \infty$, $\mathrm{Var}_Q[\mathbb{E}_{Q_n}[r]] \to 0$, so we can apply the delta method:

$$\mathrm{Var}_Q[f(X)] \approx (f'(\mathbb{E}(X)))^2 \,\mathrm{Var}_Q[X] \tag{A.16}$$

Applying $f = \log$ and $\mathbb{E}[X] = 1$ gives us the second part of the theorem:

$$\lim_{n\to\infty} n \,\mathrm{Var}_Q[\log \mathbb{E}_{Q_n}[r]] = \lim_{n\to\infty} n \,\mathrm{Var}[\mathbb{E}_{Q_n}[r]] \geq e^{D_{\mathrm{KL}}(P\|Q)} - 1 \tag{A.17}$$

which describes the variance in the asymptotic sense. □

Corollary 1. *Assume that the assumptions in Theorem 2 hold. Let P_m and Q_n be the empirical*

distributions of m i.i.d. samples from P and n i.i.d. samples from Q, respectively. Define

$$I^{m,n}_{\mathrm{NWJ}} := \mathbb{E}_{P_m}[\log r^\star + 1] - \mathbb{E}_{Q_n}[r^\star] \tag{3.12}$$

$$I^{m,n}_{\mathrm{MINE}} := \mathbb{E}_{P_m}[\log r^\star] - \log \mathbb{E}_{Q_n}[r^\star] \tag{3.13}$$

where $r^\star = \mathrm{d}P/\mathrm{d}Q$. Then under the randomness of the sampling procedure, we have $\forall m \in \mathbb{N}$:

$$\mathrm{Var}_{P,Q}[I^{m,n}_{\mathrm{NWJ}}] \geq (e^{D_{\mathrm{KL}}(P\|Q)} - 1)/n \tag{3.14}$$

$$\lim_{n\to\infty} n \,\mathrm{Var}_{P,Q}[I^{m,n}_{\mathrm{MINE}}] \geq e^{D_{\mathrm{KL}}(P\|Q)} - 1. \tag{3.15}$$

Proof. Since P_m and Q_n are independent, we have

$$\mathrm{Var}[I^{m,n}_{\mathrm{NWJ}}] \geq \mathrm{Var}[\mathbb{E}_{Q_n}[r^\star]] \tag{A.18}$$

$$= \mathrm{Var}[\mathbb{E}_{Q_n}[r^\star]] \geq \frac{e^{D_{\mathrm{KL}}(P\|Q)} - 1}{n} \tag{A.19}$$

and

$$\lim_{n\to\infty} n \,\mathrm{Var}[I^{m,n}_{\mathrm{MINE}}] \geq \lim_{n\to\infty} n \,\mathrm{Var}[\log \mathbb{E}_{Q_n}[r^\star]] \geq e^{D_{\mathrm{KL}}(P\|Q)} - 1 \tag{A.20}$$

which completes the proof. □

A.1.3 Proofs in Section 3.5

Theorem 3. *Let $r(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}_{\geq 0}$ be any non-negative measurable function such that $\int r \mathrm{d}Q = S$, $S \in (0, \infty)$ and $r(\boldsymbol{x}) \in [0, e^K]$. Define $r_\tau(\boldsymbol{x}) = \mathrm{clip}(r(\boldsymbol{x}), e^\tau, e^{-\tau})$ for finite, non-negative τ. If $\tau < K$, then the bias for using r_τ to estimate the partition function of r satisfies:*

$$|\mathbb{E}_Q[r] - \mathbb{E}_Q[r_\tau]| \leq \max\left(e^{-\tau}|1 - Se^{-\tau}|, \left|\frac{1 - e^K e^{-\tau} + S(e^K - e^\tau)}{e^K - e^{-\tau}}\right|\right);$$

if $\tau \geq K$, then

$$|\mathbb{E}_Q[r] - \mathbb{E}_Q[r_\tau]| \leq e^{-\tau}(1 - Se^{-K}).$$

Proof. We establish the upper bounds by finding a worst case r to find the largest $|\mathbb{E}_Q[r] - \mathbb{E}_Q[r_\tau]|$. First, without loss of generality, we may assume that $r(\boldsymbol{x}) \in (-\infty, e^{-\tau}] \cup [e^\tau, \infty)$ for all $\boldsymbol{x} \in \mathcal{X}$.

Otherwise, denote $\mathcal{X}_\tau(r) = \{\boldsymbol{x} \in \mathcal{X} : e^{-\tau} < r(\boldsymbol{x}) < e^{\tau}\}$ as the (measurable) set where the $r(\boldsymbol{x})$ values are between $e^{-\tau}$ and e^{τ}. Let

$$V_\tau(r) = \int_{\boldsymbol{x} \in \mathcal{X}_\tau(r)} r(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \in (e^{-\tau}|\mathcal{X}_\tau(r)|, e^{\tau}|\mathcal{X}_\tau(r)|) \tag{A.21}$$

be the integral of r over $\mathcal{X}_\tau(r)$. We can transform $r(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{X}_\tau(r)$ to have values only in $\{e^{-\tau}, e^{\tau}\}$ and still integrate to $V_\tau(r)$, so the expectation under Q is not changed.

Then we show that we can rescale all the values above e^{τ} and below e^{τ} to the same value without changing the expected value under Q. We denote

$$K_1 = \log \int I(r(\boldsymbol{x}) \leq e^{-\tau}) r(\boldsymbol{x}) \mathrm{d}Q(\boldsymbol{x}) - \log \int I(r(\boldsymbol{x}) \leq e^{-\tau}) \mathrm{d}Q(\boldsymbol{x}) \tag{A.22}$$

$$K_2 = \log \int I(r(\boldsymbol{x}) \geq e^{\tau}) r(\boldsymbol{x}) \mathrm{d}Q(\boldsymbol{x}) - \log \int I(r(\boldsymbol{x}) \geq e^{\tau}) \mathrm{d}Q(\boldsymbol{x}) \tag{A.23}$$

where e^{K_1} and e^{K_2} represents the mean of $r(\boldsymbol{x})$ for all $r(\boldsymbol{x}) \leq e^{-\tau}$ and $r(\boldsymbol{x}) \geq e^{\tau}$ respectively. We then have:

$$\mathbb{E}_Q[r] = e^{K_1} \int I(r(\boldsymbol{x}) \leq e^{-\tau}) \mathrm{d}Q(\boldsymbol{x}) + e^{K_2} \int I(r(\boldsymbol{x}) \geq e^{\tau}) \mathrm{d}Q(\boldsymbol{x}) \tag{A.24}$$

$$1 = \int I(r(\boldsymbol{x}) \leq e^{-\tau}) \mathrm{d}Q(\boldsymbol{x}) + \int I(r(\boldsymbol{x}) \geq e^{\tau}) \mathrm{d}Q(\boldsymbol{x}) \tag{A.25}$$

so we can parametrize $\mathbb{E}_Q[r]$ via K_1 and K_2. Since $E_Q[r] = S$ by assumption, we have:

$$\int I(r(\boldsymbol{x}) \leq e^{-\tau}) \mathrm{d}Q(\boldsymbol{x}) = \frac{e^{K_2} - S}{e^{K_2} - e^{-K_1}} \tag{A.26}$$

and from the definition of $r_\tau(\boldsymbol{x})$:

$$\mathbb{E}_Q[r_\tau] = \frac{e^{K_2}e^{-\tau} - Se^{-\tau} + Se^{\tau} - e^{-K_1}e^{\tau}}{e^{K_2} - e^{-K_1}} := g(K_1, K_2) \tag{A.27}$$

We can obtain an upper bound once we find $\max g(K_1, K_2)$ and $\min g(K_1, K_2)$. First, we have:

$$\begin{aligned} \frac{\partial g(K_1, K_2)}{\partial K_1} &= \frac{e^{-K_1}e^{\tau}(e^{K_2} - e^{-K_1}) - e^{-K_1}(e^{K_2}e^{-\tau} - Se^{-\tau} + Se^{\tau} - e^{-K_1}e^{\tau})}{(e^{K_2} - e^{-K_1})^2} \\ &= \frac{e^{-K_1}(e^{\tau} - e^{-\tau})(e^{K_2} - S)}{(e^{K_2} - e^{-K_1})^2} \geq 0 \end{aligned} \tag{A.28}$$

$$\frac{\partial g(K_1,K_2)}{\partial K_2} = \frac{e^{K_2}e^{-\tau}(e^{K_2}-e^{-K_1}) - e^{K_2}(e^{K_2}e^{-\tau} - Se^{-\tau} + Se^{\tau} - e^{-K_1}e^{\tau})}{(e^{K_2}-e^{-K_1})^2}$$
$$= \frac{e^{K_2}(e^{\tau}-e^{-\tau})(e^{-K_1}-S)}{(e^{K_2}-e^{-K_1})^2} \leq 0 \tag{A.29}$$

Therefore, $g(K_1, K_2)$ is largest when $K_1 \to \infty, K_2 = \tau$ and smallest when $K_1 = \tau, K_2 \to \infty$.

$$\max g(K_1,K_2) = \lim_{K\to\infty} \frac{1-e^{-K}e^{\tau}+S(e^{\tau}-e^{-\tau})}{e^{\tau}-e^{-K}} = S + e^{-\tau} - Se^{-2\tau} \tag{A.30}$$
$$\min g(K_1,K_2) = \lim_{K\to\infty} \frac{e^{K}e^{-\tau}-1+S(e^{\tau}-e^{-\tau})}{e^{K}-e^{-\tau}} = e^{-\tau} \tag{A.31}$$

Therefore,

$$|\mathbb{E}_Q[r] - \mathbb{E}_Q[r_\tau]| \leq \max(|\max g(K_1,K_2) - S|, |S - \min g(K_1,K_2)|) \tag{A.32}$$
$$= \max\left(|e^{-\tau} - Se^{-2\tau}|, |S - e^{-\tau}|\right) \tag{A.33}$$

The proof for Theorem 3 simply follows the above analysis for fixed K. When $\tau < K$, we consider the case when $K_1 \to \infty, K_2 = \tau$ and $K_1 = \tau, K_2 = K$; when $\tau > K$ only the smaller values will be clipped, so the increased value is no larger than the case where $K_1 \to \infty, K_2 = K$:

$$\frac{e^K - S}{e^K} \cdot e^{\tau} = e^{-\tau}(1 - Se^{-K}) \tag{A.34}$$

where $e^K \geq S$ from the fact that $\int r \, \mathrm{d}Q = S$. □

Theorem 4. *The variance of the estimator $\mathbb{E}_{Q_n}[r_\tau]$ (using n samples from Q) satisfies:*

$$\mathrm{Var}[\mathbb{E}_{Q_n}[r_\tau]] \leq \frac{e^{\tau}-e^{-\tau}}{4n} \tag{3.18}$$

Proof. Since $r_\tau(\boldsymbol{x})$ is bounded between e^{τ} and $e^{-\tau}$, we have

$$\mathrm{Var}[r_\tau] \leq \frac{e^{\tau}-e^{-\tau}}{4} \tag{A.35}$$

Taking the mean of n independent random variables gives us the result. □

Combining Theorem 3 and 4 with the bias-variance trade-off argument, we have the following:

Corollary 8. *Let $r(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}_{\geq 0}$ be any non-negative measurable function such that $\int r \mathrm{d}Q = S$, $S \in (0, \infty)$ and $r(\boldsymbol{x}) \in [0, e^K]$. Define $r_\tau(\boldsymbol{x}) = \mathrm{clip}(r(\boldsymbol{x}), e^\tau, e^{-\tau})$ for finite, non-negative τ and $\mathbb{E}_{Q_n}$ as the sample average of n i.i.d. samples from Q. If $\tau < K$, then*

$$\mathbb{E}_Q[(r - \mathbb{E}_{Q_n}[r_\tau])^2] \leq \max\left(e^{-\tau}|1 - Se^{-\tau}|, \left|\frac{1 - e^K e^{-\tau} + S(e^K - e^\tau)}{e^K - e^{-\tau}}\right|\right)^2 + \frac{e^\tau - e^{-\tau}}{4n};$$

If $\tau \geq K$, then:

$$\mathbb{E}_Q[(r - \mathbb{E}_{Q_n}[r_\tau])^2] \leq e^{-2\tau}(1 - Se^{-K})^2 + \frac{e^\tau - e^{-\tau}}{4n} \tag{A.36}$$

A.2 Proofs for Chapter 4

A.2.1 Preliminary Lemma and Propositions

To prove the main results, we need the following Lemma and Propositions 6 and 7. The Lemma is a special case to the dual representation of f-divergences discussed in [NWJ08].

Lemma 9 (Nguyen et al. [NWJ08])**.** *$\forall P, Q \in \mathcal{P}(\mathcal{X})$ such that $P \ll Q$,*

$$D_{\mathrm{KL}}(P\|Q) = \sup_{T \in L^\infty(Q)} \mathbb{E}_P[T] - \mathbb{E}_Q[e^T] + 1 \tag{A.37}$$

Proof. (Sketch) Please refer to [NWJ08] for a more formal proof.

Denote $f(t) = t \log t$ whose convex conjugate is $f^\star(u) = \exp(u - 1)$, we have that

$$\begin{aligned}
D_{\mathrm{KL}}(P\|Q) &= \mathbb{E}_{\boldsymbol{x}\sim Q}\left[f\left(\frac{\mathrm{d}P}{\mathrm{d}Q}(\boldsymbol{x})\right)\right] = \mathbb{E}_{\boldsymbol{x}\sim Q}\left[\sup_u u \cdot \frac{\mathrm{d}P}{\mathrm{d}Q}(\boldsymbol{x}) - f^\star(u)\right] && \text{(A.38)}\\
&= \sup_{T \in L^\infty(Q)} \mathbb{E}_{\boldsymbol{x}\sim Q}\left[T(\boldsymbol{x}) \cdot \frac{\mathrm{d}P}{\mathrm{d}Q}(\boldsymbol{x}) - f^\star(T(\boldsymbol{x}))\right] && \text{(A.39)}\\
&= \sup_{T \in L^\infty(Q)} \mathbb{E}_{\boldsymbol{x}\sim P}[T(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim Q}\left[f^\star(T(\boldsymbol{x}))\right] && \text{(A.40)}\\
&= \sup_{T \in L^\infty(Q)} \mathbb{E}_{\boldsymbol{x}\sim P}[T(\boldsymbol{x}) + 1] - \mathbb{E}_{\boldsymbol{x}\sim Q}\left[\exp(T(\boldsymbol{x}))\right], && \text{(A.41)}
\end{aligned}$$

which completes the proof. □

Proposition 6. *For all positive integers $n \geq 1, m \geq 2$, and for any collection of positive random variables $\{X_i\}_{i=1}^n, \{\overline{X_{i,j}}\}_{j=1}^m$ such that $\forall i \in [n]$, $X_i, \overline{X_{i,1}}, \overline{X_{i,2}}, \ldots, \overline{X_{i,m-1}}$ are exchangeable, then*

$\forall \alpha \in (0, \frac{2m}{m+1}]$, *the following is true:*

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\frac{mX_i}{\alpha X_i + \frac{m-\alpha}{m-1}\sum_{j=1}^{m-1}\overline{X_{i,j}}}\right] \leq \frac{1}{\alpha}. \tag{A.42}$$

Proof. First, for $\alpha \in (0, 2m/(m+1)]$ we have:

$$n\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\frac{mX_i}{\alpha X_i + \frac{m-\alpha}{m-1}\sum_{j=1}^{m-1}\overline{X_{i,j}}}\right] \tag{A.43}$$

$$= \mathbb{E}\left[\sum_{i=1}^{n}\frac{m\frac{m-1}{m-\alpha}X_i}{(\Sigma_i) - \left(1 - \frac{m-1}{m-\alpha}\alpha\right)X_i}\right] \tag{A.44}$$

$$= \mathbb{E}\left[\sum_{i=1}^{n}\frac{m\frac{m-1}{m-\alpha}}{\Sigma_i/X_i - (1 - \frac{m-1}{m-\alpha}\alpha)}\right] \tag{A.45}$$

$$= m\frac{m-1}{m-\alpha}\mathbb{E}\left[\sum_{i=1}^{n}\sum_{p=0}^{\infty}\left(\frac{X_i}{\Sigma_i}\right)^{p+1}\left(1 - \frac{m-1}{m-\alpha}\alpha\right)^{p}\right] \qquad \text{(Taylor expansion)} \tag{A.46}$$

$$= m\frac{m-1}{m-\alpha}\sum_{i=1}^{n}\sum_{p=0}^{\infty}\mathbb{E}\left[\left(\frac{X_i}{\Sigma_i}\right)^{p+1}\right]\left(1 - \frac{m-1}{m-\alpha}\alpha\right)^{p} \tag{A.47}$$

where we simplify the notation with $\Sigma_i := X_i + \sum_{j=1}^{m-1}\overline{X_{i,j}}$. Furthermore, we note that the Taylor series converges because $(1 - \frac{m-1}{m-\alpha}\alpha) \in (-1, 1)$.

Since the random variables are exchangeable, switching the ordering of $X_i, \overline{X_{i,1}}, \ldots, \overline{X_{i,m-1}}$ does not affect the joint distribution, and the summing function is permutation invariant. Therefore, for all $i \in [n], p \geq 0$,

$$\mathbb{E}\left[\left(\frac{X_i}{\Sigma_i}\right)^{p+1}\right] = \frac{1}{m}\mathbb{E}\left[\left(\frac{X_i}{\Sigma_i}\right)^{p+1} + \sum_{j=1}^{m-1}\left(\frac{\overline{X_{i,j}}}{\Sigma_i}\right)^{p+1}\right] \tag{A.48}$$

$$\leq \frac{1}{m}\mathbb{E}\left[\left(\frac{X_i}{\Sigma_i} + \sum_{j=1}^{m-1}\frac{\overline{X_{i,j}}}{\Sigma_i}\right)^{p+1}\right] = \frac{1}{m} \tag{A.49}$$

where the last inequality comes from the fact that $\left(X_i + \sum_{j=1}^{m-1}\overline{X_{i,j}}\right)/\Sigma_i = 1$ and all the random

variables are positive. Continuing from equation A.47, we have:

$$n\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\frac{mX_i}{\alpha X_i + \frac{m-\alpha}{m-1}\sum_{j=1}^{m-1}\overline{X_{i,j}}}\right] \tag{A.50}$$

$$\leq nm\frac{m-1}{m-\alpha}\sum_{p=0}^{\infty}\frac{1}{m}\left(1-\frac{m-1}{m-\alpha}\alpha\right)^p = \frac{m-1}{m-\alpha}\frac{n}{\alpha\frac{m-1}{m-\alpha}} = \frac{n}{\alpha} \tag{A.51}$$

Dividing both sides by n completes the proof for $\alpha \in (0, \frac{2m}{m+1}]$. □

Proposition 7. *$\forall n \geq 1, m \geq 2$, and for any collection of positive random variables $\{X_i\}_{i=1}^n$, $\{\overline{X_{i,j}}\}_{j=1}^m$ such that $\forall i \in [n]$, $X_i, \overline{X_{i,1}}, \overline{X_{i,2}}, \ldots, \overline{X_{i,m-1}}$ are exchangeable, then $\forall \alpha \in [1, \frac{m}{2}]$,*

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\frac{mX_i}{\alpha X_i + \frac{m-\alpha}{m-1}\sum_{j=1}^{m-1}\overline{X_{i,j}}}\right] \leq 1 \tag{A.52}$$

Proof. The case for $\alpha \in [1, \frac{2m}{m+1}]$ is apparent from Proposition 6.

For $\alpha \in (2m/(m+1), m/2]$, we have for all $t \in [m-1]$:

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\frac{mX_i}{\alpha X_i + \frac{m-\alpha}{m-1}\sum_{j=1}^{m-1}\overline{X_{i,j}}}\right] \tag{A.53}$$

$$\leq \mathbb{E}\left[\frac{1}{n}\frac{1}{m-1}\sum_{i=1}^{n}\sum_{j=1}^{m-1}\frac{mX_i}{\alpha X_i + \sum_{k=j}^{j+t-1}\overline{X_{i,k}}}\right] \tag{A.54}$$

$$= \mathbb{E}\left[\frac{1}{n}\frac{1}{m-1}\sum_{i=1}^{n}\sum_{j=1}^{m-1}\frac{tX_i}{\frac{t\alpha}{m}X_i + \frac{t-\frac{t\alpha}{m}}{t-1}\sum_{k=j}^{j+t-1}\overline{X_{i,k}}}\right] \tag{A.55}$$

where we define $\overline{X_{i,k}} = \overline{X_{i,k-(m-1)}}$ when $k > (m-1)$ and use the concavity of the inverse function (or equivalently the HM-AM inequality) to establish equation A.54. For any $\alpha \in (2m/(m+1), m/2]$, we can choose t to be any integer from the interval $[\frac{m}{\alpha}, \frac{2m}{\alpha} - 1]$; we note that such an integer always exists because the length of the interval is greater or equal to 1:

$$\frac{2m}{\alpha} - 1 - \frac{m}{\alpha} = \frac{m}{\alpha} - 1 \geq 1$$

Then we can apply the result in Proposition 6, for t samples and the new α being $\frac{t\alpha}{m}$; from our

construction of t, this satisfies the condition in Proposition 6 that:

$$1 \leq \frac{t\alpha}{m} \leq \frac{2t}{t+1}$$

Therefore we can apply Proposition 6 to a valid choice of t to obtain

$$\begin{aligned}
&\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\frac{mX_i}{\alpha X_i + \frac{m-\alpha}{m-1}\sum_{j=1}^{m-1}\overline{X_{i,j}}}\right] \\
&\leq \mathbb{E}\left[\frac{1}{n}\frac{1}{m-1}\sum_{i=1}^{n}\sum_{j=1}^{m-1}\frac{tX_i}{\frac{t\alpha}{m}X_i + \frac{t-\frac{t\alpha}{m}}{t-1}\sum_{k=j}^{j+t-1}\overline{X_{i,k}}}\right] \leq \frac{m}{t\alpha} \leq 1
\end{aligned}$$

which proves the result. □

A.2.2 Proof for CPC

Theorem 5. *For all probability measures P, Q over sample space $\mathcal{X}$ such that $P \ll Q$, the following holds for all functions $r : \mathcal{X} \to \mathbb{R}_+$ and integers $m \geq 2$:*

$$D_{\mathrm{KL}}(P\|Q) \geq \mathbb{E}_{\boldsymbol{x}\sim P, \boldsymbol{y}_{1:m-1}\sim Q^{m-1}}\left[\log \frac{m \cdot r(\boldsymbol{x})}{r(\boldsymbol{x}) + \sum_{i=1}^{m-1} r(\boldsymbol{y}_i)}\right]. \tag{4.1}$$

Proof. From Lemma 9, we have that:

$$\begin{aligned}
&D_{\mathrm{KL}}(P\|Q) && \text{(A.56)}\\
&\geq \mathbb{E}_{\boldsymbol{y}_{1:m-1}\sim Q^{m-1}}\left[\mathbb{E}_{\boldsymbol{x}\sim P}\left[\log \frac{m \cdot r(\boldsymbol{x})}{r(\boldsymbol{x}) + \sum_{i=1}^{m-1} r(\boldsymbol{y}_i)}\right] - \mathbb{E}_{\boldsymbol{x}\sim Q}\left[\frac{m \cdot r(\boldsymbol{x})}{r(\boldsymbol{x}) + \sum_{i=1}^{m-1} r(\boldsymbol{y}_i)}\right] + 1\right] \\
&\geq \mathbb{E}_{\boldsymbol{y}_{1:m-1}\sim Q^{m-1}}\left[\mathbb{E}_{\boldsymbol{x}\sim P}\left[\log \frac{m \cdot r(\boldsymbol{x})}{r(\boldsymbol{x}) + \sum_{i=1}^{m-1} r(\boldsymbol{y}_i)}\right]\right] - 1 + 1 && \text{(A.57)}\\
&= \mathbb{E}_{\boldsymbol{x}\sim P, \boldsymbol{y}_{1:m-1}\sim Q^{m-1}}\left[\log \frac{m \cdot r(\boldsymbol{x})}{r(\boldsymbol{x}) + \sum_{i=1}^{m-1} r(\boldsymbol{y}_i)}\right], && \text{(A.58)}
\end{aligned}$$

where the second inequality comes from Proposition 6 where $\boldsymbol{x} \sim Q$ and $\boldsymbol{y}_{1:m-1} \sim Q^{m-1}$ are exchangeable, thus proving the statement. □

A.2.3 Proof for ML-CPC

Theorem 6. *For all probability measures P, Q over sample space $\mathcal{X}$ such that $P \ll Q$, the following holds for all functions $r : \mathcal{X} \to \mathbb{R}_+$, integers $n \geq 1, m \geq 2$, and real numbers $\alpha \in [\frac{m}{n(m-1)+1}, 1]$:*

$$D_{\mathrm{KL}}(P\|Q) \geq \mathbb{E}_{\boldsymbol{x}_{1:n}\sim P^n, \boldsymbol{y}_{i,1:m-1}\sim Q^{m-1}} \left[\frac{1}{n} \sum_{i=1}^{n} \log \frac{mn \cdot r(\boldsymbol{x}_i)}{\alpha \sum_{j=1}^{n} r(\boldsymbol{x}_j) + \frac{m-\alpha}{m-1} \sum_{k=1}^{m-1} r(\boldsymbol{y}_{j,k})} \right]. \tag{4.5}$$

Proof. First, we have

$$\mathbb{E}\left[\frac{1}{n} \sum_{i=1}^{n} \log \frac{nm \cdot r(\boldsymbol{x}_i)}{\alpha \sum_{j=1}^{n} r(\boldsymbol{x}_j) + \frac{m-\alpha}{m-1} \sum_{j=1}^{n} \sum_{k=1}^{m-1} r(\boldsymbol{y}_{j,k})} \right] \tag{A.59}$$

$$= \mathbb{E}\left[\frac{1}{n} \sum_{i=1}^{n} \log(nm \cdot r(\boldsymbol{x}_i)) - \log \left(\alpha \sum_{j=1}^{n} r(\boldsymbol{x}_j) + \frac{m-\alpha}{m-1} \sum_{j=1}^{n} \sum_{k=1}^{m-1} r(\boldsymbol{y}_{j,k}) \right) \right]$$

$$\leq \mathbb{E}\left[\frac{1}{n} \sum_{i=1}^{n} \log(nm \cdot r(\boldsymbol{x}_i)) - \log \left(n\alpha r(\boldsymbol{x}_i) + \frac{m-\alpha}{m-1} \sum_{j=1}^{n} \sum_{k=1}^{m-1} r(\boldsymbol{y}_{j,k}) \right) \right] \tag{A.60}$$

$$= \mathbb{E}\left[\frac{1}{n} \sum_{i=1}^{n} \frac{nm \cdot r(\boldsymbol{x}_i)}{n\alpha r(\boldsymbol{x}_i) + \frac{m-\alpha}{m-1} \sum_{j=1}^{n} \sum_{k=1}^{m-1} r(\boldsymbol{y}_{j,k})} \right] \tag{A.61}$$

$$\leq D_{\mathrm{KL}}(P\|Q) - 1 + \mathbb{E}_{\boldsymbol{x}_{1:n}\sim Q^n}\left[\frac{1}{n} \sum_{i=1}^{n} \frac{nm \cdot r(\boldsymbol{x}_i)}{n\alpha r(\boldsymbol{x}_i) + \frac{m-\alpha}{m-1} \sum_{j=1}^{n} \sum_{k=1}^{m-1} r(\boldsymbol{y}_{j,k})} \right] \tag{A.62}$$

where we use Jensen's inequality over log in equation A.60 and Lemma 9 in equation A.62.

Since $\boldsymbol{x}_i \sim Q$ and all the $\boldsymbol{y}_{j,k}$ are $(n(m-1)+1)$ exchangeable random variables, and

$$m \geq 2, \alpha \in \left[\frac{m}{n(m-1)+1}, 1 \right] \quad \Rightarrow \quad \frac{n(m-1)+1}{m} \alpha \in \left[1, \frac{n(m-1)+1}{2} \right],$$

we can apply Proposition 7 to the $(n(m-1)+1)$ exchangeable variables

$$\mathbb{E}_{\boldsymbol{x}_{1:n}\sim Q^n}\left[\frac{1}{n} \sum_{i=1}^{n} \frac{nm \cdot r(\boldsymbol{x}_i)}{n\alpha r(\boldsymbol{x}_i) + \frac{m-\alpha}{m-1} \sum_{j=1}^{n} \sum_{k=1}^{m-1} r(\boldsymbol{y}_{j,k})} \right]$$

$$= \mathbb{E}_{\boldsymbol{x}_{1:n}\sim Q^n}\left[\frac{1}{n} \sum_{i=1}^{n} \frac{(n(m-1)+1) \cdot r(\boldsymbol{x}_i)}{\frac{n(m-1)+1}{m} \alpha r(\boldsymbol{x}_i) + \frac{m - \frac{n(m-1)+1}{nm}\alpha}{m-1} \sum_{j=1}^{n} \sum_{k=1}^{m-1} r(\boldsymbol{y}_{j,k})} \right] \leq 1$$

Combining the above with equation A.62, proves the result for the given range of α. □

A.3 Proofs for Chapter 5

A.3.1 Proof of Lemma 6

Proof.

$$\begin{aligned}
I_q(\mathbf{x};\mathbf{z}|\mathbf{u}) &= \mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z},\mathbf{u})}[\log q_\phi(\mathbf{x},\mathbf{z}|\mathbf{u}) - \log q(\mathbf{x}|\mathbf{u}) - \log q_\phi(\mathbf{z}|\mathbf{u})] \\
&= \mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z},\mathbf{u})}[\log q_\phi(\mathbf{x},\mathbf{z}|\mathbf{u}) - \log q_\phi(\mathbf{z}|\mathbf{u})] + \mathbb{E}_{q_\phi(\mathbf{z},\mathbf{u})}[-\log q(\mathbf{x}|\mathbf{u})] \\
&= \mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z},\mathbf{u})}[\log q_\phi(\mathbf{x}|\mathbf{z},\mathbf{u})] + H_q(\mathbf{x}|\mathbf{u}) \\
&= \mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z},\mathbf{u})}[\log q_\phi(\mathbf{x}|\mathbf{z},\mathbf{u}) + \log p(\mathbf{x}|\mathbf{z},\mathbf{u}) - \log p(\mathbf{x}|\mathbf{z},\mathbf{u})] + H_q(\mathbf{x}|\mathbf{u}) \\
&= \mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z},\mathbf{u})}[\log p(\mathbf{x}|\mathbf{z},\mathbf{u})] + H_q(\mathbf{x}|\mathbf{u}) + \mathbb{E}_{q_\phi(\mathbf{z},\mathbf{u})} D_{\mathrm{KL}}(q_\phi(\mathbf{x}|\mathbf{z},\mathbf{u}) \| p(\mathbf{x}|\mathbf{z},\mathbf{u})) \\
&\geq \mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z},\mathbf{u})}[\log p(\mathbf{x}|\mathbf{z},\mathbf{u})] + H_q(\mathbf{x}|\mathbf{u})
\end{aligned}$$

where the last inequality holds because KL divergence is non-negative. □

A.3.2 Proof of Lemma 7

Proof.

$$\begin{aligned}
I_q(\mathbf{z};\mathbf{u}) &\leq I_q(\mathbf{z};\mathbf{x},\mathbf{u}) \\
&= \mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z},\mathbf{u})}[\log q_\phi(\mathbf{z}|\mathbf{x},\mathbf{u}) - \log q_\phi(\mathbf{z})] \\
&= \mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z},\mathbf{u})}[\log q_\phi(\mathbf{z}|\mathbf{x},\mathbf{u}) - \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}) + \log p(\mathbf{z})] \\
&= \mathbb{E}_{q(\mathbf{x},\mathbf{u})} D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x},\mathbf{u}) \| p(\mathbf{z})) - D_{\mathrm{KL}}(q_\phi(\mathbf{z}) \| p(\mathbf{z}))
\end{aligned}$$

□

A.3.3 Proof of Lemma 8

Proof.

$$\begin{aligned}
I_q(\mathbf{z};\mathbf{u}) &= \mathbb{E}_{q_\phi(\mathbf{z},\mathbf{u})}[\log q_\phi(\mathbf{u}|\mathbf{z}) - \log q(\mathbf{u})] \\
&= \mathbb{E}_{q_\phi(\mathbf{z},\mathbf{u})}[\log q_\phi(\mathbf{u}|\mathbf{z}) - \log p(\mathbf{u}) - \log q(\mathbf{u}) + \log p(\mathbf{u})]
\end{aligned}$$

$$= \mathbb{E}_{q_\phi(\mathbf{z})} D_{\mathrm{KL}}(q_\phi(\mathbf{u}|\mathbf{z}) \| p(\mathbf{u})) - D_{\mathrm{KL}}(q(\mathbf{u}) \| p(\mathbf{u}))$$
$$\leq \mathbb{E}_{q_\phi(\mathbf{z})} D_{\mathrm{KL}}(q_\phi(\mathbf{u}|\mathbf{z}) \| p(\mathbf{u}))$$

Again, the last inequality holds because KL divergence is non-negative. □

A.3.4 Proof of Theorem 7

Proof. Let us first verify that this problem is convex.

- Primal: $-\mathbb{E}_{q_\phi(\mathbf{x},\mathbf{z},\mathbf{u})}[\log p_\theta(\mathbf{x}|\mathbf{z},\mathbf{u})]$ is affine in $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{u})$, convex in $p_\theta(\mathbf{x}|\mathbf{z},\mathbf{u})$ due to the concavity of $\log$, and independent of $p_\theta(\mathbf{z})$.
- First condition: $\mathbb{E}_{q(\mathbf{u})} D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x},\mathbf{u}) \| p_\theta(\mathbf{z}))$ is convex in $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{u})$ and $p_\theta(\mathbf{z})$ (because of convexity of KL-divergence), and independent of $p_\theta(\mathbf{x}|\mathbf{z},\mathbf{u})$.
- Second condition: since $\mathbb{E}_{q_\phi(\mathbf{z})} D_{\mathrm{KL}}(q_\phi(\mathbf{u}|\mathbf{z}) \| p(\mathbf{u})) - D_{\mathrm{KL}}(q(\mathbf{u}) \| p(\mathbf{u})) = I_q(\mathbf{z};\mathbf{u})$ and

$$I_q(\mathbf{z};\mathbf{u}) = D_{\mathrm{KL}}(q_\phi(\mathbf{z},\mathbf{u}) \| q(\mathbf{u}) q_\phi(\mathbf{z})) \tag{A.63}$$
$$= D_{\mathrm{KL}}(\sum_{\mathbf{x}} q_\phi(\mathbf{z}|\mathbf{x},\mathbf{u}) q(\mathbf{x},\mathbf{u}) \| q(\mathbf{u}) \sum_{\mathbf{x},\mathbf{u}} q_\phi(\mathbf{z}|\mathbf{x},\mathbf{u}) q(\mathbf{x},\mathbf{u})) \tag{A.64}$$

  Let $q = \beta q_1 + (1-\beta) q_2, \forall \beta \in [0,1], q_1, q_2$. We have

$$\begin{aligned} I_q(\mathbf{z};\mathbf{u}) &= D_{\mathrm{KL}}(\sum_{\mathbf{x}} q(\mathbf{z}|\mathbf{x},\mathbf{u}) q(\mathbf{x},\mathbf{u}) \| q(\mathbf{u}) \sum_{\mathbf{x},\mathbf{u}} q(\mathbf{z}|\mathbf{x},\mathbf{u}) q(\mathbf{x},\mathbf{u})) \\ &\geq \beta D_{\mathrm{KL}}(\sum_{\mathbf{x}} q_1(\mathbf{z}|\mathbf{x},\mathbf{u}) q(\mathbf{x},\mathbf{u}) \| q(\mathbf{u}) \sum_{\mathbf{x},\mathbf{u}} q_1(\mathbf{z}|\mathbf{x},\mathbf{u}) q(\mathbf{x},\mathbf{u})) \\ &\quad + (1-\beta) D_{\mathrm{KL}}(\sum_{\mathbf{x}} q_2(\mathbf{z}|\mathbf{x},\mathbf{u}) q(\mathbf{x},\mathbf{u}) \| q(\mathbf{u}) \sum_{\mathbf{x},\mathbf{u}} q_2(\mathbf{z}|\mathbf{x},\mathbf{u}) q(\mathbf{x},\mathbf{u})) \\ &= \beta I_{q_1}(\mathbf{z};\mathbf{u}) + (1-\beta) I_{q_2}(\mathbf{z};\mathbf{u}) \end{aligned}$$

  where we use the convexity of KL divergence in the inequality. Since $D_{\mathrm{KL}}(q(\mathbf{u}) \| p(\mathbf{u}))$ is independent of $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{u})$, both $I_q(\mathbf{z};\mathbf{u})$ and $\mathbb{E}_{q_\phi(\mathbf{z})} D_{\mathrm{KL}}(q_\phi(\mathbf{u}|\mathbf{z}) \| p(\mathbf{u}))$ are convex in $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{u})$.

Then we show that the problem has a feasible solution by construction. In fact, we can simply let $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{u}) = p_\theta(\mathbf{z})$ be some fixed distribution over $\mathbf{z}$, and $p_\theta(\mathbf{x}|\mathbf{z},\mathbf{u}) = q_\phi(\mathbf{x}|\mathbf{z},\mathbf{u})$ for all $\mathbf{x},\mathbf{u}$. In this case, $\mathbf{z}$ and $\mathbf{u}$ are independent, so $D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x},\mathbf{u}) \| p_\theta(\mathbf{z})) = 0 < \epsilon_1$, $D_{\mathrm{KL}}(q_\phi(\mathbf{u}|\mathbf{z}) \| p(\mathbf{u})) =$

$0 < \epsilon_2$. This corresponds to the case where $\mathbf{z}$ is simply random noise that does not capture anything in $\mathbf{u}$.

Hence, Slater's condition holds, which is a sufficient condition for strong duality. □

A.4 Proofs for Chapter 6

Proposition 3. *$\forall P, Q \in \mathcal{P}(\mathcal{X})$ such that $P \ll Q$, $\forall T \in L^\infty(Q)$ such that $\operatorname{im}(T) \subseteq \operatorname{dom}((f')^{-1})$, and $\forall \mathcal{R} \subseteq L^\infty_{\geq 0}(Q)$ such that $\{\mathbb{1}\} \subseteq \mathcal{R}$ we have:*

$$I_f(T; P, Q) \leq \mathcal{L}_f^{\mathcal{R}}(T; P, Q) \leq I_W(T; P, Q). \tag{6.15}$$

Proof. From Propositions 1, and that $\mathcal{R} \subseteq L^\infty_{\geq 0}(Q)$, we have:

$$I_f(T; P, Q) = \inf_{r \in L^\infty_{\geq 0}(Q)} \ell_f(T, r; P, Q) \leq \inf_{r \in \mathcal{R}} \ell_f(T, r; P, Q) = \mathcal{L}_f^{\mathcal{R}}(T; P, Q). \tag{A.65}$$

From Proposition 2 and that $\{\mathbb{1}\} \subseteq \mathcal{R}$, we have:

$$\mathcal{L}_f^{\mathcal{R}}(T; P, Q) = \inf_{r \in \mathcal{R}} \ell_f(T, r; P, Q) \leq \inf_{r \in \mathbb{1}} \ell_f(T, r; P, Q) = \mathcal{L}_f^{\mathcal{R}}(T; P, Q) \leq I_W(T; P, Q). \tag{A.66}$$

Combining the two inequalities completes the proof. □

Theorem 8. *For $\{\mathbb{1}\} \subseteq \mathcal{R} \subseteq L^\infty_{\geq 0}(Q)$, define*

$$D_{f,\mathcal{R}}(P\|Q) := \sup_{T \in \mathcal{F}} \mathcal{L}_f^{\mathcal{R}}(T; P, Q) \tag{6.16}$$

where $\mathcal{F} := \{T : \mathcal{X} \to \operatorname{dom}((f')^{-1}), T \in L^\infty(Q)\}$. Then

$$D_f(P\|Q) \leq D_{f,\mathcal{R}}(P\|Q) \leq \sup_{T \in \mathcal{F}} I_W(T; P, Q). \tag{6.17}$$

Proof. From Proposition 1, we have the following upper bound for $D_{f,\mathcal{R}}(P\|Q)$:

$$\begin{aligned} & \sup_{T \in \mathcal{F}} \inf_{r \in \mathcal{R}} \mathbb{E}_P[f(r)] + \mathbb{E}_P[T] - \mathbb{E}_Q[r \cdot T] \\ \leq & \sup_{T \in \mathcal{F}} \inf_{r \in \{\mathbb{1}\}} \mathbb{E}_P[f(r)] + \mathbb{E}_P[T] - \mathbb{E}_Q[r \cdot T] \\ = & \sup_{T \in \mathcal{F}} \mathbb{E}_P[T] - \mathbb{E}_Q[T] = \mathrm{IPM}_{\mathcal{F}}(P, Q), \end{aligned} \tag{A.67}$$

We also have the following lower bound for $D_{f,\mathcal{R}}(P\|Q)$:

$$\begin{aligned} & \sup_{T\in\mathcal{F}} \inf_{r\in\mathcal{R}} \mathbb{E}_P[f(r)] + \mathbb{E}_P[T] - \mathbb{E}_Q[r\cdot T] \\ \geq\ & \sup_{T\in\mathcal{F}} \inf_{r\in L^\infty_{\geq 0}(Q)} \mathbb{E}_P[f(r)] + \mathbb{E}_P[T] - \mathbb{E}_Q[r\cdot T] \\ =\ & \sup_{T\in\mathcal{F}} \mathbb{E}_P[T] - \mathbb{E}_Q[f_*(T)] = D_f(P\|Q). \end{aligned} \tag{A.68}$$

Therefore, $D_{f,\mathcal{R}}(P\|Q)$ is bounded between $D_f(P\|Q)$ and $\mathrm{IPM}_{\mathcal{F}}(P,Q)$ and thus it is a valid divergence over $\mathcal{P}(\mathcal{X})$. □

Theorem 9. *Let $f(u) = u\log u$ and $\mathcal{F}$ a set of real-valued bounded measurable functions on $\mathcal{X}$. For any fixed choice of P, Q, and $T \in \mathcal{F}$, we have*

$$\arg\min_{r\in\Delta(Q)} \mathbb{E}_Q[f(r)] + \mathbb{E}_P[T] - \mathbb{E}_Q[r\cdot T] = \frac{e^T}{\mathbb{E}_Q[e^T]} \tag{6.19}$$

Proof. Consider the following Lagrangian:

$$h(r,\lambda) := \mathbb{E}_Q[f(r)] - \mathbb{E}_Q[r\cdot T] + \lambda(\mathbb{E}_Q[r] - 1) \tag{A.69}$$

where $\lambda\in\mathbb{R}$ and we formalize the constraint $r\in\Delta(r)$ with $\mathbb{E}_Q[r]-1=0$. Taking the functional derivative $\partial h/\partial r$ and setting it to zero, we have:

$$\begin{aligned} & f'(r)\,\mathrm{d}Q - T\,\mathrm{d}Q + \lambda \\ =\ & (\log r + 1)\,\mathrm{d}Q - T\,\mathrm{d}Q + \lambda = 0, \end{aligned} \tag{A.70}$$

so $r = \exp(T-(\lambda+1))$. We can then apply the constraint $\mathbb{E}_Q[r]=1$, where we solve $\lambda+1 = \mathbb{E}_Q[e^T]$, and consequently the optimal $r = e^T/\mathbb{E}_Q[e^T] \in \Delta(Q)$. □

A.5 Proofs for Chapter 7

A.5.1 NDA for GANs

Theorem 10. *Let $\overline{P} \in \mathcal{P}(\mathcal{X})$ be any distribution over $\mathcal{X}$ with disjoint support than p_{data}, i.e., such that $\mathrm{supp}(p_{\mathrm{data}}) \cap \mathrm{supp}(\overline{P}) = \varnothing$. Let $D_\phi : \mathcal{X}\to\mathbb{R}$ be the set of all discriminators over $\mathcal{X}$, $f:\mathbb{R}_{\geq 0}\to\mathbb{R}$ be a convex, semi-continuous function such that $f(1)=0$, $f^\star$ be the convex conjugate*

of f, f' its derivative, and G_θ be a distribution with sample space $\mathcal{X}$. Then $\forall \lambda \in (0, 1]$, we have:

$$\arg\min_{G_\theta \in \mathcal{P}(\mathcal{X})} \max_{D_\phi : \mathcal{X} \to \mathbb{R}} L_f(G_\theta, D_\phi) = \arg\min_{G_\theta \in \mathcal{P}(\mathcal{X})} \max_{D_\phi : \mathcal{X} \to \mathbb{R}} L_f(\lambda G_\theta + (1-\lambda)\overline{P}, D_\phi) = p_{\text{data}} \tag{7.4}$$

where $L_f(Q, D_\phi) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[D_\phi(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x} \sim Q}[f^\star(D_\phi(\boldsymbol{x}))]$ is the objective for f-GAN [NCT16]. However, the optimal discriminators are different for the two objectives:

$$\arg\max_{D_\phi : \mathcal{X} \to \mathbb{R}} L_f(G_\theta, D_\phi) = f'(p_{\text{data}}/G_\theta) \tag{7.5}$$

$$\arg\max_{D_\phi : \mathcal{X} \to \mathbb{R}} L_f(\lambda G_\theta + (1-\lambda)\overline{P}, D_\phi) = f'(p_{\text{data}}/(\lambda G_\theta + (1-\lambda)\overline{P})) \tag{7.6}$$

Proof. Let us use $p(x), \overline{p}(x), q(x)$ to denote the density functions of $p_{\text{data}}, \overline{P}$ and G_θ respectively (and P, $\overline{P}$, Q for the respective distributions). First, from Lemma 1 in [NWJ08], we have that

$$\max_{D_\phi : \mathcal{X} \to \mathbb{R}} L_f(G_\theta, D_\phi) = D_f(P \| G_\theta) \tag{A.71}$$

$$\max_{D_\phi : \mathcal{X} \to \mathbb{R}} L_f(\lambda G_\theta + (1-\lambda)\overline{P}, D_\phi) = D_f(P \| \lambda Q + (1-\lambda)\overline{P}) \tag{A.72}$$

where D_f refers to the f-divergence. Then, we have

$$\begin{aligned}
& D_f(P||\lambda Q + (1-\lambda)\overline{P}) \\
=& \int_{\mathcal{X}} (\lambda q(x) + (1-\lambda)\overline{p}(x))\, f\left(\frac{p(x)}{\lambda q(x) + (1-\lambda)\overline{p}(x)}\right) \\
=& \int_{\mathcal{X}} \lambda q(x) f\left(\frac{p(x)}{\lambda q(x) + (1-\lambda)\overline{p}(x)}\right) + (1-\lambda) f(0) \\
\geq& \lambda f\left(\int_{\mathcal{X}} q(x) \frac{p(x)}{\lambda q(x) + (1-\lambda)\overline{p}(x)}\right) + (1-\lambda) f(0) \qquad \text{(A.73)} \\
=& \lambda f\left(\frac{1}{\lambda} \int_{\mathcal{X}} \lambda q(x) \frac{p(x)}{\lambda q(x) + (1-\lambda)\overline{p}(x)}\right) + (1-\lambda) f(0) \\
=& \lambda f\left(\frac{1}{\lambda} \int_{\mathcal{X}} (\lambda q(x) + (1-\lambda)\overline{p}(x) - (1-\lambda)\overline{p}(x)) \frac{p(x)}{\lambda q(x) + (1-\lambda)\overline{p}(x)}\right) + (1-\lambda) f(0) \\
=& \lambda f\left(\frac{1}{\lambda} - \int_{\mathcal{X}} ((1-\lambda)\overline{p}(x)) \frac{p(x)}{\lambda q(x) + (1-\lambda)\overline{p}(x)}\right) + (1-\lambda) f(0) \\
=& \lambda f\left(\frac{1}{\lambda}\right) + (1-\lambda) f(0) \qquad \text{(A.74)}
\end{aligned}$$

where we use the fact that f is convex with Jensen's inequality in Eq.(A.73) and the fact that

$p(x)\overline{p}(x) = 0, \forall x \in \mathcal{X}$ in Eq.(A.74) since P and $\overline{P}$ has disjoint support.

We also have

$$\begin{aligned} D_f(P||\lambda P + (1-\lambda)\overline{P}) &= \int_{\mathcal{X}} (\lambda p(x) + (1-\lambda)\overline{p}(x))\, f\left(\frac{p(x)}{\lambda p(x) + (1-\lambda)\overline{p}(x)}\right) \\ &= \int_{\mathcal{X}} (\lambda p(x))\, f\left(\frac{p(x)}{\lambda p(x) + (1-\lambda)\overline{p}(x)}\right) + (1-\lambda) f(0) \\ &= \int_{\mathcal{X}} (\lambda p(x))\, f\left(\frac{p(x)}{\lambda p(x) + 0}\right) + (1-\lambda) f(0) \\ &= \lambda f\left(\frac{1}{\lambda}\right) + (1-\lambda) f(0) \end{aligned}$$

Therefore, in order for the inequality in Equation A.73 to be an equality, we must have that $q(\boldsymbol{x}) = p(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{X}$. Therefore, the generator distribution recovers the data distribution at the equlibrium posed by the NDA-GAN objective, which is also the case for the original GAN objective.

Moreover, from Lemma 1 in [NWJ08], we have that:

$$\arg\max_{D_\phi} L_f(Q, D_\phi) = f'(p_{\text{data}}/Q) \tag{A.75}$$

Therefore, by replacing Q with G_θ and $(\lambda G_\theta + (1-\lambda)\overline{P})$, we have:

$$\arg\max_{D_\phi : \mathcal{X} \to \mathbb{R}} L_f(G_\theta, D_\phi) = f'(p_{\text{data}}/G_\theta) \tag{A.76}$$

$$\arg\max_{D_\phi : \mathcal{X} \to \mathbb{R}} L_f(\lambda G_\theta + (1-\lambda)\overline{P}, D_\phi) = f'(p_{\text{data}}/(\lambda G_\theta + (1-\lambda)\overline{P})) \tag{A.77}$$

which shows that the optimal discriminators are indeed different for the two objectives. □

A.5.2 NDA for Contrastive Representation Learning

We describe the detailed statement of Theorem 2 and proof as follows.

Theorem 18. *For some distribution $\overline{p}$ over $\mathcal{X}$ such that $\text{supp}(\overline{p}) \cap \text{supp}(p_{\text{data}}) = \varnothing$, and for any maximizer of the NDA-CPC objective*

$$\hat{h} \in \arg\max_{h_\theta} \max_{g_\phi} \overline{I_{\text{CPC}}}(h_\theta, g_\phi)$$

the representations of negative samples are disjoint from that of positive samples for $\hat{h}$; i.e., $\forall \boldsymbol{x} \in \mathrm{supp}(p_{\mathrm{data}}), \bar{\boldsymbol{x}} \in \mathrm{supp}(\overline{p})$,

$$\mathrm{supp}(\hat{h}(\bar{\boldsymbol{x}})) \cap \mathrm{supp}(\hat{h}(\boldsymbol{x})) = \varnothing$$

Proof. We use a contradiction argument to establish the proof. For any representation mapping that maximizes the NDA-CPC objective,

$$\hat{h} \in \arg\max_{h_\theta} \max_{g_\phi} \overline{I_{\mathrm{CPC}}}(h_\theta, g_\phi)$$

suppose that the positive and NDA samples share some support, i.e., $\exists \boldsymbol{x} \in \mathrm{supp}(p_{\mathrm{data}}), \bar{\boldsymbol{x}} \in \mathrm{supp}(\overline{p})$,

$$\mathrm{supp}(\hat{h}(\bar{\boldsymbol{x}})) \cap \mathrm{supp}(\hat{h}(\boldsymbol{x})) \neq \varnothing$$

We can always construct $\hat{h}'$ that shares the same representation with $\hat{h}$ for p_{data} but have disjoint representations for NDA samples; i.e., $\forall \boldsymbol{x} \in \mathrm{supp}(p_{\mathrm{data}}), \bar{\boldsymbol{x}} \in \mathrm{supp}(\overline{p})$, the following two statements are true:

1. $\hat{h}(\boldsymbol{x}) = \hat{h}'(\boldsymbol{x})$;
2. $\mathrm{supp}(\hat{h}'(\bar{\boldsymbol{x}})) \cap \mathrm{supp}(\hat{h}'(\boldsymbol{x})) = \varnothing$.

Our goal is to prove that:

$$\max_{g_\phi} \overline{I_{\mathrm{CPC}}}(\hat{h}', g_\phi) > \max_{g_\phi} \overline{I_{\mathrm{CPC}}}(\hat{h}, g_\phi) \tag{A.78}$$

which shows a contradiction.

For ease of exposition, let us allow zero values for the output of g, and define $0/0 = 0$ (in this case, if g assigns zero to positive values, then the CPC objective becomes $-\infty$, so it cannot be a maximizer to the objective).

Let $\hat{g} \in \arg\max \overline{I_{\mathrm{CPC}}}(\hat{h}, g_\phi)$ be an optimal critic to the representation model $\hat{h_\theta}$. We then define a following critic function:

$$\hat{g}'(\boldsymbol{x}, \boldsymbol{z}) = \begin{cases} \hat{g}(\boldsymbol{x}, \boldsymbol{z}) & \text{if } \exists \boldsymbol{x} \in \mathrm{supp}(p_{\mathrm{data}}) \quad s.t. \quad \boldsymbol{z} \in \mathrm{supp}(\hat{h}'(\boldsymbol{x})) \\ 0 & \text{otherwise} \end{cases} \tag{A.79}$$

In other words, the critic assigns the same value for data-representation pairs over the support of p_{data} and zero otherwise. From the assumption over $\hat{h}$, $\exists \boldsymbol{x} \in \mathrm{supp}(p_{\mathrm{data}}), \bar{\boldsymbol{x}} \in \mathrm{supp}(\overline{p})$, and

$\overline{\boldsymbol{z}} \in \operatorname{supp}(\hat{h}(\bar{\boldsymbol{x}}))$,

$$\overline{\boldsymbol{z}} \in \operatorname{supp}(\hat{h}(\boldsymbol{x}))$$

so $(\boldsymbol{x}, \overline{\boldsymbol{z}})$ can be sampled as a positive pair and $\hat{g}(\boldsymbol{x}, \overline{\boldsymbol{z}}) > 0$.

Therefore,

$$\max_{g_\phi} \overline{I_{\mathrm{CPC}}}(\hat{h}', g_\phi) \geq \overline{I_{\mathrm{CPC}}}(\hat{h}', \hat{g}') \tag{A.80}$$

$$= \mathbb{E}\left[\log \frac{(n+m)\hat{g}'(\boldsymbol{x}, \boldsymbol{z})}{\hat{g}'(\boldsymbol{x}, \boldsymbol{z}) + \sum_{j=1}^{n-1} \hat{g}'(\boldsymbol{x}, \widehat{\boldsymbol{z}}_j) + \sum_{k=1}^{m} \underbrace{\hat{g}'(\boldsymbol{x}, \overline{\boldsymbol{z}}_k)}_{=0}}\right] \qquad \text{(plug in definition for NDA-CPC)}$$

$$\geq \mathbb{E}\left[\log \frac{(n+m)\hat{g}(\boldsymbol{x}, \boldsymbol{z})}{\hat{g}(\boldsymbol{x}, \boldsymbol{z}) + \sum_{j=1}^{n-1} \hat{g}(\boldsymbol{x}, \widehat{\boldsymbol{z}}_j) + \sum_{k=1}^{m} \hat{g}(\boldsymbol{x}, \overline{\boldsymbol{z}}_k)}\right] \qquad \text{(existence of some} \hat{g}(\boldsymbol{x}, \overline{\boldsymbol{z}}) > 0\text{)}$$

$$= \max_{g_\phi} \overline{I_{\mathrm{CPC}}}(\hat{h}, g_\phi) \qquad \text{(Assumption that } \hat{g} \text{ is optimal critic)}$$

which proves the theorem via contradiction. □

A.6 Proofs for Chapter 8

A.6.1 Relationship to maximum likelihood

Theorem 12. *(informal) For any valid $\{\alpha_i\}_{i=0}^T$, there exists some weights $\hat{w} : \{\alpha_i\}_{i=1}^T \to \mathbb{R}_+$ for the diffusion objective such that $-L_{\mathrm{D2}}$ is a variational lower bound to the log-likelihood, i.e.,*

$$-L_{\mathrm{D2}}(\theta, \phi; \hat{w}) \leq \mathbb{E}_{p_{\mathrm{data}}}[\log p_\theta(\mathbf{x})], \tag{8.12}$$

where $p_\theta(\mathbf{x}) := \mathbb{E}_{\mathbf{x}_0 \sim p^{(0)}(\mathbf{z}^{(0)})}[p_\theta(\mathbf{x}|\mathbf{z}^{(0)})]$ is the marginal probability of $\mathbf{x}$ under the D2C model.

Theorem 19. *(formal) Suppose that $\mathbf{x} \in \mathbb{R}^d$. For any valid $\{\alpha_i\}_{i=0}^T$, let $\hat{w}$ satisfy:*

$$\forall t \in [2, \ldots, T], \quad \hat{w}(\alpha_t) = \frac{(1-\alpha_t)\alpha_{t-1}}{2(1-\alpha_{t-1})^2 \alpha_t} \tag{A.81}$$

$$\hat{w}(\alpha_1) = \frac{1-\alpha_1}{2(2\pi)^d \alpha_1} \tag{A.82}$$

then:

$$-L_{\mathrm{D2}}(\theta, \phi; \hat{w}) + H(q_\phi(\mathbf{z}^{(1)}|\mathbf{x})) \leq \mathbb{E}_{p_{\mathrm{data}}}[\log p_\theta(\mathbf{x})] \tag{A.83}$$

where $p_\theta(\mathbf{x}) := \mathbb{E}_{\mathbf{x}_0 \sim p^{(0)}(\mathbf{z}^{(0)})}[p_\theta(\mathbf{x}|\mathbf{z}^{(0)})]$ *is the marginal probability of* $\mathbf{x}$ *under the D2C model.*

Proof. First, we have that:

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x})}[\log p_\theta(\mathbf{x})] = \mathbb{E}_{p_{\text{data}}(\mathbf{x})}\left[\log \sum_{\mathbf{z}^{(1)}} p_\theta(\mathbf{x}|\mathbf{z}^{(1)}) p_\theta(\mathbf{z}^{(1)})\right] \tag{A.84}$$

$$\geq \mathbb{E}_{p_{\text{data}}(\mathbf{x}), q_\phi(\mathbf{z}^{(1)})}[\log p_\theta(\mathbf{x}|\mathbf{z}^{(1)}) + \log p_\theta(\mathbf{z}^{(1)}) - \log q_\phi(\mathbf{z}^{(1)}|\mathbf{x})] \tag{A.85}$$

$$= \mathbb{E}_{p_{\text{data}}(\mathbf{x}), q_\phi(\mathbf{z}^{(1)}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z}^{(1)}) - D_{\text{KL}}(q_\phi(\mathbf{z}^{(1)}|\mathbf{x}) \| p_\theta(\mathbf{z}^{(1)}))]. \tag{A.86}$$

where we use Jensen's inequality here. Compared with the objective for D2:

$$-L_{\text{D2}}(\theta, \phi; w) := \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}, \mathbf{z}^{(1)} \sim q_\phi(\mathbf{z}^{(1)}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z}^{(1)}) - \ell_{\text{diff}}(\mathbf{z}^{(1)}; w, \theta)], \tag{A.87}$$

and it is clear the proof is complete if we show that:

$$H(q_\phi(\mathbf{z}^{(1)}|\mathbf{x})) - \mathbb{E}_{\mathbf{z}^{(1)} \sim q_\phi(\mathbf{z}^{(1)}|\mathbf{x})}[\ell_{\text{diff}}(\mathbf{z}^{(1)}; \hat{w}, \theta)] \tag{A.88}$$

$$\leq - D_{\text{KL}}(q_\phi(\mathbf{z}^{(1)}|\mathbf{x}) \| p_\theta(\mathbf{z}^{(1)})) \tag{A.89}$$

$$= H(q_\phi(\mathbf{z}^{(1)}|\mathbf{x})) + \mathbb{E}_{\mathbf{z}^{(1)} \sim q_\phi(\mathbf{z}^{(1)}|\mathbf{x})}[\log p_\theta(\mathbf{z}^{(1)})] \tag{A.90}$$

or equivalently:

$$\mathbb{E}_{\mathbf{z}^{(1)} \sim q_\phi(\mathbf{z}^{(1)}|\mathbf{x})}[\ell_{\text{diff}}(\mathbf{z}^{(1)}; \hat{w}, \theta)] \leq \mathbb{E}_{\mathbf{z}^{(1)} \sim q_\phi(\mathbf{z}^{(1)}|\mathbf{x})}[\log p_\theta(\mathbf{z}^{(1)})] \tag{A.91}$$

Let us apply variational inference with an inference model $q(\mathbf{z}^{(\alpha_{1:T})}|\mathbf{z}^{(1)})$ where $\alpha_0 = 1$ and $\alpha_T = 0$:

$$\mathbb{E}_{\mathbf{z}^{(1)} \sim q_\phi(\mathbf{z}^{(1)}|\mathbf{x})}[\log p_\theta(\mathbf{z}^{(1)})] = \mathbb{E}_{\mathbf{z}^{(1)} \sim q_\phi(\mathbf{z}^{(1)}|\mathbf{x})}[\log \sum_{\mathbf{z}} \left(p_\theta(\mathbf{z}^{(\alpha_T)}) \prod_{t=1}^{T} p_\theta(\mathbf{z}^{(\alpha_{t-1})}|\mathbf{z}^{(\alpha_t)})\right)]$$

$$\geq \mathbb{E}_{\mathbf{z}^{(\alpha_{0:T})}}[\log p_\theta(\mathbf{z}^{(\alpha_T)}) + \sum_{t=1}^{T} \log p_\theta(\mathbf{z}^{(\alpha_{t-1})}|\mathbf{z}^{(\alpha_t)}) - \log q(\mathbf{z}^{(\alpha_{1:T})}|\mathbf{z}^{(\alpha_0)})] \tag{A.92}$$

$$\geq \mathbb{E}_{\mathbf{z}^{(\alpha_{0:T})}}\Big[\log p_\theta(\mathbf{z}^{(\alpha_T)}) - \log q(\mathbf{z}^{(\alpha_T)}|\mathbf{z}^{(\alpha_0)}) \tag{A.93}$$

$$- \sum_{t=2}^{T} \underbrace{D_{\text{KL}}(q(\mathbf{z}^{(\alpha_{t-1})}|\mathbf{z}^{(\alpha_t)}, \mathbf{z}^{(\alpha_0)}) \| p_\theta(\mathbf{z}^{(\alpha_{t-1})}|\mathbf{z}^{(\alpha_t)}))}_{L_{t-1}} + \log p_\theta(\mathbf{z}^{(\alpha_0)}|\mathbf{z}^{(\alpha_1)})\Big]$$

where we remove the superscript of p_θ to indicate that there are no intermediate discretization steps between α_{t-1} and α_t. Now, for $t \geq 2$, let us consider p_θ and q_ϕ with the form in Equations 8.4 and 8.5 respectively, which are both Gaussian distributions (restrictions to p_θ will still give lower bounds). Then we can model the standard deviation of $p_\theta(\mathbf{x}^{(\alpha_{t-1})}|\mathbf{x}^{(\alpha_t)})$ to be equal to that of $q(\mathbf{x}^{(\alpha_{t-1})}|\mathbf{x}^{(\alpha_t)}, \mathbf{x}^{(\alpha_0)}))$. Under this formulation, the KL divergence for L_{t-1} is just one between two Gaussians with the same standard deviations and is a weighted Euclidean distance between the means. Using the derivation from Equation 8.6 to Equation 8.8, we have that:

$$L_{t-1} = \mathbb{E}_{\mathbf{z}_0,\epsilon}\left[\frac{(1-\alpha_t)\alpha_{t-1}}{2(1-\alpha_{t-1})^2\alpha_t}\|\epsilon - \epsilon_\theta(\mathbf{z}^{(\alpha_t)}; \alpha_t, \alpha_{t-1})\|_2^2\right] \tag{A.94}$$

which gives us the weights for $\hat{w}$ for $\alpha_{2:T}$. For $p_\theta(\mathbf{z}^{(\alpha_0)}|\mathbf{z}^{(\alpha_1)})$ let us model it to be a Gaussian with mean

$$\mu_\theta(\mathbf{z}^{(\alpha_1)}; \alpha_1, \alpha_0) = \frac{\mathbf{z}^{(\alpha_1)} - \sqrt{1-\alpha_t}\epsilon_\theta(\mathbf{z}^{(\alpha_1)}; \alpha_1, \alpha_0)}{\sqrt{\alpha_1}}$$

and standard deviation $1/\sqrt{2\pi}$ (chosen such that normalization constant is 1). Thus, with

$$\mathbf{z}^{(0)} = \frac{\mathbf{z}^{(\alpha_1)} - \sqrt{1-\alpha_t}\epsilon}{\sqrt{\alpha_1}}$$

we have that:

$$\log p_\theta(\mathbf{z}^{(\alpha_0)}|\mathbf{z}^{(\alpha_1)}) = \frac{1-\alpha_1}{2(2\pi)^d\alpha_1}\|\epsilon - \epsilon_\theta(\mathbf{z}^{(\alpha_1)}; \alpha_1, \alpha_0)\|_2^2 \tag{A.95}$$

which gives us the weight of $\hat{w}$ for α_1. Furthermore:

$$\mathbb{E}_{\mathbf{z}^{(\alpha_{0:T})}}[\log p_\theta(\mathbf{z}^{(\alpha_T)}) - q(\mathbf{z}^{(\alpha_T)}|\mathbf{z}^{(\alpha_0)})] = 0 \tag{A.96}$$

because $\mathbf{z}^{(\alpha_T)} \sim \mathcal{N}(0, \boldsymbol{I})$ for both p_θ and q. Therefore, we have that:

$$\mathbb{E}_{\mathbf{z}^{(1)}\sim q_\phi(\mathbf{z}^{(1)}|\mathbf{x})}[\ell_{\text{diff}}(\mathbf{z}^{(1)}; \hat{w}, \theta)] \leq \mathbb{E}_{\mathbf{z}^{(1)}\sim q_\phi(\mathbf{z}^{(1)}|\mathbf{x})}[\log p_\theta(\mathbf{z}^{(1)})] \tag{A.97}$$

which completes the proof. □

A.6.2 D2 models address latent posterior mismatch in VAEs

Theorem 13. *(informal) Let $p_\theta(\mathbf{z}) = \mathcal{N}(0, 1)$. For any $\epsilon > 0$, there exists a distribution $q_\phi(\mathbf{z})$ with an $(\epsilon, 0.49)$-prior hole, such that $D_{\mathrm{KL}}(q_\phi \| p_\theta) \leq \log 2$[1] and $W_2(q_\phi, p_\theta) < \gamma$ for any $\gamma > 0$, where W_2 is the 2-Wasserstein distance.*

Theorem 20. *(formal) Let $p_\theta(\mathbf{z}) = \mathcal{N}(0, \boldsymbol{I})$ where $\mathbf{z} \in \mathbb{R}^d$. For any $\epsilon > 0, \delta < 0.5$, there exists a distribution $q_\phi(\mathbf{z})$ with an (ϵ, δ)-prior hole, such that $D_{\mathrm{KL}}(q_\phi \| p_\theta) \leq \log 2$ and $W_2(q_\phi, p_\theta) < \gamma$ for any $\gamma > 0$, where W_2 is the 2-Wasserstein distance.*

Proof. Let us define a function $f : \mathbb{R}_{\geq 0} \to [0, 1]$ such that for any Euclidean ball $B(0, R)$ centered at 0 with radius R:

$$f(R) := \int_{B(0,R)} p_\theta(\mathbf{z})\, \mathrm{d}\mathbf{z}, \tag{A.98}$$

i.e., $f(R)$ measures the probability mass of the Gaussian distribution $p_\theta(\mathbf{z})$ within $B(0, R)$. As $\mathrm{d}f/\mathrm{d}R > 0$ for $R > 0$, f is invertible.

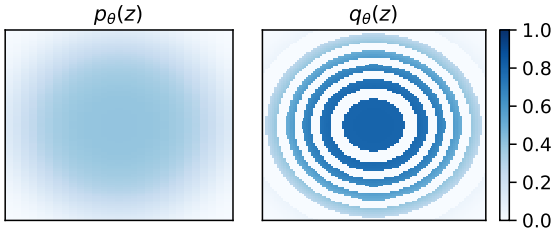


Figure A.1: Illustration of the construction in 2d. When we use more rings, the prior hole and upper bound of KL divergence are constant but the upper bound of Wasserstein distance decreases.

Now we shall construct $q_\phi(\mathbf{z})$. First, let $q_\phi(\mathbf{z}) = p_\theta(\mathbf{z})$ whenever $\|\mathbf{z}\|_2 \geq f^{-1}(2\delta)$; then for any n, we can find a sequence $\{r_0, r_1, \ldots, r_{2n}\}$ such that:

$$r_0 = 0, \quad r_{2n} = f^{-1}(2\delta), \quad f(r_i) - f(r_{i-1}) = f^{-1}(2\delta)\delta/n \text{ for all } k \in \{1, \ldots, 2n\}, \tag{A.99}$$

Intuitively, we find $2n$ circles with radii $\{r_0, \ldots, r_{2n}\}$ whose masses measured by $p_\theta(\mathbf{z})$ is an

[1] This is reasonably low for realistic VAE models (NVAE [VK20] reports a KL divergence of around 2810 nats).

arithmetic progression $\{0, \delta/2n, \ldots, 2\delta\}$. We then define $q_\phi(\mathbf{z})$ for $\|\mathbf{z}\| < f^{-1}(2\delta)$ as follows:

$$q_\phi(\mathbf{z}) = \begin{cases} 2 \cdot p_\theta(\mathbf{z}) & \text{if } \|\mathbf{z}\| \in \bigcup_{k=0}^{n-1}[r_{2k}, r_{2k+1}) \\ 0 & \text{otherwise} \end{cases} \tag{A.100}$$

Intuitively, q_ϕ is defined by moving all the mass from ring $(2k+1)$ to ring $2k$. Note that this $q_\phi(\mathbf{z})$ is a valid probability distribution because:

$$\int_{\mathbb{R}^d} q_\phi(\mathbf{z})\,\mathrm{d}\mathbf{z} = \int_{B(0,f^{-1}(2\delta))} q_\phi(\mathbf{z})\,\mathrm{d}\mathbf{z} + \int_{B^c(0,f^{-1}(2\delta))} q_\phi(\mathbf{z})\,\mathrm{d}\mathbf{z} \tag{A.101}$$

$$= 2\int_{B(0,f^{-1}(2\delta))} p_\theta(\mathbf{z})\mathbb{I}\left(\|\mathbf{z}\| \in \bigcup_{i=0}^{n-1}[r_{2k}, r_{2k+1})\right)\mathrm{d}\mathbf{z} + \int_{B^c(0,f^{-1}(2\delta))} p_\theta(\mathbf{z})\,\mathrm{d}\mathbf{z} \tag{A.102}$$

$$= \int_{B(0,f^{-1}(2\delta))} p_\theta(\mathbf{z})\,\mathrm{d}\mathbf{z} + \int_{B^c(0,f^{-1}(2\delta))} p_\theta(\mathbf{z})\,\mathrm{d}\mathbf{z} = 1 \tag{A.103}$$

Next, we validate that q_ϕ satisfies our constraints in the statement.

Prior hole Apparently, if we choose $\mathcal{S} = \bigcup_{k=0}^{n-1}[r_{2k+1}, r_{2k+2})$, then $\int_{\mathcal{S}} p_\theta(\mathbf{z})\,\mathrm{d}\mathbf{z} = \delta$ and $\int_{\mathcal{S}} q_\phi(\mathbf{z})\,\mathrm{d}\mathbf{z} = 0$; so $\mathcal{S}$ instantiates a (ϵ, δ)-prior hole.

KL divergence We note that $q_\phi(\mathbf{z}) \leq 2p_\theta(\mathbf{z})$ is true for all $\mathbf{z}$, so

$$D_{\mathrm{KL}}(q_\phi(\mathbf{z})\|p_\theta(\mathbf{z})) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z})}[\log q_\phi(\mathbf{z}) - \log p_\theta(\mathbf{z})] \leq \log 2.$$

2 Wasserstein Distance We use the Monge formulation:

$$W_2(q_\phi(\mathbf{z}), 2p_\theta(\mathbf{z})) = \min_{T:q_\phi = T_\sharp p_\theta} \int_{R^d} \|\mathbf{z} - T(\mathbf{z})\|_2^2 p_\theta(\mathbf{z})\,\mathrm{d}\mathbf{z} \tag{A.104}$$

where T is any transport map from p_θ to q_ϕ. Consider the transport map $\hat{T}$ such that:

$$\hat{T}(\mathbf{z}) = \begin{cases} \mathbf{z} & \text{if} \quad q_\phi(\mathbf{z}) \geq 0 \\ \mathbf{z} \cdot f^{-1}(f(\|\mathbf{z}\|) - f(r_{2k+1}) + k\delta/n) & \text{otherwise, for } k \text{ such that } \|\mathbf{z}\|_2 \leq [r_{2k+1}, r_{2k+2}) \end{cases} \tag{A.105}$$

which moves the mass in $[r_{2k+1}, r_{2k+2})$ to $[r_{2k}, r_{2k+1})$. From this definition, we have that $\|\hat{T}(\mathbf{z}) - \mathbf{z}\|_2 \leq \max_{k\in\{0,\dots,n-1\}}(r_{2k+2} - r_{2k})$. Moreover, since by definition,

$$2\delta/n = \int_{B(0,r_{2k+2})} p_\theta(\mathbf{z})\,\mathrm{d}\mathbf{z} - \int_{B(0,r_{2k})} p_\theta(\mathbf{z})\,\mathrm{d}\mathbf{z} \tag{A.106}$$

$$> \pi(r_{2k+2}^2 - r_{2k}^2) \min_{\mathbf{z}:\|\mathbf{z}\|\in[r_{2k},r_{2k+2})} p_\theta(\mathbf{z}) \tag{A.107}$$

$$> \pi(r_{2k+2} - r_{2k})^2 \min_{\mathbf{z}:\|\mathbf{z}\|\in[r_{2k},r_{2k+2})} p_\theta(\mathbf{z}) \tag{A.108}$$

We have that

$$W_2(q_\phi(\mathbf{z}), 2p_\theta(\mathbf{z})) \leq \max_{k\in\{0,\dots,n-1\}} (r_{2k+2} - r_{2k})^2 < \frac{2\delta}{\pi n \min_{\mathbf{z}:\|\mathbf{z}\|_2\leq r_{2n}} p_\theta(\mathbf{z})} \tag{A.109}$$

$$< \frac{2\delta}{\pi n \min_{\mathbf{z}:\|\mathbf{z}\|_2\leq r_{2n}} p_\theta(f^{-1}(2\delta)\mathbf{n})} \tag{A.110}$$

for any vector $\mathbf{n}$ with norm 1. Note that the above inequality is inversely proportional to n, which can be any integer. Therefore, for a fixed δ, $W_2(q_\phi(\mathbf{z}), 2p_\theta(\mathbf{z})) = O(1/n)$; so for any γ, there exists n such that $W_2(q_\phi(\mathbf{z}), 2p_\theta(\mathbf{z})) < \gamma$, completing the proof. □

Note on DDIM prior preventing the prior hole For a noise level α, we have that:

$$q^{(\alpha)}(\mathbf{z}^{(\alpha)}) = \mathbb{E}_{\mathbf{z}^{(1)}\sim q^{(1)}(\mathbf{z}^{(1)})}[\mathcal{N}(\sqrt{\alpha}\mathbf{z}^{(1)}, (1-\alpha)\boldsymbol{I})] \tag{A.111}$$

as $\alpha \to 0$, $D_{\mathrm{KL}}(q^{(\alpha)}(\mathbf{z}^{(\alpha)})\|\mathcal{N}(0, \boldsymbol{I})) \to 0$. From Pinsker's inequality and the definition of (ϵ, δ)-prior hole:

$$\delta - \epsilon \leq D_{\mathrm{TV}}(q^{(\alpha)}(\mathbf{z}^{(\alpha)}), \mathcal{N}(0, \boldsymbol{I}))) \leq \sqrt{\frac{1}{2} D_{\mathrm{KL}}(q^{(\alpha)}(\mathbf{z}^{(\alpha)})\|\mathcal{N}(0, \boldsymbol{I}))}, \tag{A.112}$$

we should not expect to see any (ϵ, δ)-prior hole where the difference between δ and ϵ is large.

A.7 Proofs for Chapter 9

Proof. For any (x, v) and (x', v'), g satisfies:

$$g(x', v'|x, v) = \frac{1}{2}\left|\det \frac{\partial f(x,v)}{\partial(x,v)}\right|^{-1} \mathbb{I}(x', v' = f(x, v)) + \frac{1}{2}\left|\det \frac{\partial f(x,v)}{\partial(x,v)}\right| \mathbb{I}(x', v' = f^{-1}(x, v))$$

$$
\begin{aligned}
&= \frac{1}{2}\mathbb{I}(x', v' = f(x, v)) + \frac{1}{2}\mathbb{I}(x', v' = f^{-1}(x, v)) \\
&= \frac{1}{2}\mathbb{I}(x, v = f^{-1}(x', v')) + \frac{1}{2}\mathbb{I}(x, v = f(x', v')) \\
&= g(x, v|x', v') \qquad \text{(A.113)}
\end{aligned}
$$

where $\mathbb{I}(\cdot)$ is the indicator function, the first equality is the definition of $g(x', v'|x, v)$, the second equality is true since $f(x, v)$ is volume preserving, the third equality is a reparametrization of the conditions, and the last equality uses the definition of $g(x, v|x', v')$ and f is volume preserving, so the determinant of the Jacobian is 1. □

Theorem 14 allows us to use the ration $p(x', v')/p(x, v)$ when performing the MH step.

A.8 Proofs for Chapter 10

We use $\hat{v}_i(s)$, $\hat{q}_i(s, a_i)$ and $Q(\tau)$ to represent $\hat{v}_i(s; \boldsymbol{\pi}, \mathrm{v})$, $\hat{q}_i(s, a_i; \boldsymbol{\pi}, \mathrm{v})$ and $Q(\tau; \boldsymbol{\pi}, \mathrm{v})$, where we implicitly assume dependency over $\boldsymbol{\pi}$ and v.

A.8.1 Proof to Lemma 10

For any policy $\boldsymbol{\pi}$, $f_{\mathrm{v}}(\boldsymbol{\pi}, \hat{v}) = 0$ when $\hat{v}$ is the value function of $\boldsymbol{\pi}$ (due to Bellman equations). However, only policies that form a Nash equilibrium satisfies the constraints in Eq. 10.2; we formalize this in the following Lemma.

Lemma 10. *Let $\hat{v}_i(s; \boldsymbol{\pi}, \mathrm{v})$ be the solution to the Bellman equation*

$$\hat{v}_i(s) = \mathbb{E}_{\boldsymbol{\pi}}[r_i(s, \mathbf{a}) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, \mathbf{a})\hat{v}_i(s')]$$

and $\hat{q}_i(s, a_i) = \mathbb{E}_{\pi_{-i}}[r_i(s, \mathbf{a}) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, \mathbf{a})\hat{v}_i(s')]$. Then for any π,

$$f_{\mathrm{v}}(\boldsymbol{\pi}, \hat{v}(\boldsymbol{\pi})) = 0$$

Furthermore, $\boldsymbol{\pi}$ is Nash equilibrium under r if and only if $\hat{v}_i(s) \geq \hat{q}_i(s, a_i)$ for all $i \in [N], s \in \mathcal{S}, a_i \in \mathcal{A}_i$.

Proof. By definition of $\hat{v}_i(s)$ we have:

$$\begin{aligned}\hat{v}_i(s) &= \mathbb{E}_{\boldsymbol{\pi}}[r_i(s,\mathbf{a}) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,\mathbf{a})\hat{v}_i(s')] \\ &= \mathbb{E}_{\pi_i}\mathbb{E}_{\pi_{-i}}[r_i(s,\mathbf{a}) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,\mathbf{a})\hat{v}_i(s')] \\ &= \mathbb{E}_{\pi_i}[\hat{q}_i(s,a_i)]\end{aligned}$$

which uses the fact that a_i and a_{-i} are independent conditioned on s. Hence $f_{\mathrm{v}}(\boldsymbol{\pi}, \hat{v}) = 0$ immediately follows.

If $\boldsymbol{\pi}$ is a Nash equilibrium, and at least one of the constrains does not hold, i.e. there exists some i and s, a_i such that $\hat{v}(s) < \hat{q}(s,a_i)$, then agent i can achieve a strictly higher expected return if it chooses to take actions a_i whenever it encounters state s_i and follow π_i for rest of the states, which violates the Nash equilibrium assumption.

If the constraints hold, i.e. for all i and (s,a_i), $\hat{v}_i(s) \geq \hat{q}_i(s,a_i)$ then

$$\hat{v}_i(s) \geq \mathbb{E}_{\pi_i}[\hat{q}_i(s,a_i)] = \hat{v}_i(s)$$

so value iteration over $\hat{v}_i(s)$ converges. If we can find another policy $\boldsymbol{\pi}'$ such that $\hat{v}_i(s) < \mathbb{E}_{\pi_i'}[\hat{q}_i(s,a_i)]$, then there should be at least one violation in the constraints since π_i' must be a convex combination (expectation) over actions a_i. Therefore, for any policy π_i' and action a_i for any agent i, $\mathbb{E}_{\pi_i}[\hat{q}_i(s,a_i)] \geq \mathbb{E}_{\pi_i'}[\hat{q}_i(s,a_i)]$ always hold, so π_i is the optimal response to π_{-i}, and π constitutes a Nash equilibrium when we repeat this argument for all agents.

Notably, Theorem 3.8.2 in [FV12] discusses the equivalence by assuming $f_{\mathrm{v}}(\boldsymbol{\pi}, v) = 0$ for some v; if v satisfies the assumptions, then $v = \hat{v}'$. □

A.8.2 Proof to Theorem 15

Proof. If π is a Nash equilibrium, and at least one of the constraints does not hold, i.e. there exists some i and $\{s^{(j)}, a_i^{(j)}\}_{j=0}^{t}$, such that

$$\hat{v}_i(s^{(0)}) < \mathbb{E}_{\pi_{-i}}[\hat{q}_i^{(t)}(\{s^{(j)}, \mathbf{a}^{(j)}\}_{j=0}^{t-1}, s^{(t)}, a_i^{(t)})]$$

Then agent i can achieve a strictly higher expected return on its own if it chooses a particular sequence of actions by taking $a_i^{(j)}$ whenever it encounters state $s^{(j)}$, and follow π_i for the remaining

states. We note that this is in expectation over the policy of other agents. Hence, we construct a policy for agent i that has strictly higher value than π_i without modifying π_{-i}, which contradicts the definition of Nash equilibrium.

If the constraints hold, i.e for all i and $\{s^{(j)}, a_i^{(j)}\}_{j=0}^t$,

$$\hat{v}_i(s^{(0)}) \geq \mathbb{E}_{\pi_{-i}}[\hat{q}_i^{(t)}(\{s^{(j)}, \mathbf{a}^{(j)}\}_{j=0}^{t-1}, s^{(t)}, a_i^{(t)})]$$

then we can construct any $\hat{q}_i(s^{(0)}, a_i^{(0)})$ via a convex combination by taking the expectation over π_i:

$$\hat{q}_i(s^{(0)}, a_i^{(0)}) = \mathbb{E}_{\pi_i}[\mathbb{E}_{\pi_{-i}}[\hat{q}_i^{(t)}(\{s^{(j)}, \mathbf{a}^{(j)}\}_{j=0}^{t-1}, s^{(t)}, a_i^{(t)})]]$$

where the expectation over π_i is taken over actions $\{a_i^{(j)}\}_{j=0}^t$ (the expectation over states are contained in the inner expectation over π_{-i}). Therefore, $\forall i \in [N], s \in \mathcal{S}, a_i \in \mathcal{A}_i$,

$$\hat{v}_i(s) \geq \hat{q}_i(s, a_i)$$

and we recover the constraints in Eq. 10.2. By Lemma 10, π is a Nash equilibrium. □

A.8.3 Proof to Theorem 16

Proof. We use $Q^\star, \hat{q}^\star, \hat{v}^\star$ to denote the Q, $\hat{q}$ and $\hat{v}$ quantities defined for policy $\boldsymbol{\pi}^\star$. For the two terms in $L_r^{(t+1)}(\boldsymbol{\pi}^\star, \lambda_{\boldsymbol{\pi}}^\star)$ we have:

$$L_r^{(t+1)}(\boldsymbol{\pi}^\star, \lambda_{\boldsymbol{\pi}}^\star) = \sum_{i=1}^N \sum_{\tau_i \in \mathcal{T}_i} \lambda^\star(\tau_i)(Q_i^\star(\tau_i) - \hat{v}_i^\star(s^{(0)})) \tag{A.114}$$

For any agent i, we note that

$$\sum_{\tau_i \in \mathcal{T}_i} \lambda^\star(\tau_i) Q_i^\star(\tau_i) = \mathbb{E}_{\pi_i}\mathbb{E}_{\pi_{-i}^\star}[\sum_{j=0}^{t-1} \gamma^j r_i(s^{(j)}, a^{(j)}) + \gamma^t \hat{q}_i^\star(s^t, a_i^{(t)})]$$

which amounts to using π_i for agent i for the first t steps and using $\pi_i^\star$ for the remaining steps, whereas other agents follow $\pi_{-i}^\star$. As $t \to \infty$, this converges to $\mathbb{E}_{\pi_i, \pi_{-i}^\star}[r_i]$ since $\gamma^t \to 0$ and $q_i^\star(s^{(t)}, a_i^{(t)})$ is bounded. Moreover, for $\hat{v}_i^\star(s^{(0)})$, we have

$$\sum_{\tau_i \in \mathcal{T}_i} \lambda^\star(\tau_i) \hat{v}_i^\star(s^{(0)}) = \mathbb{E}_{s^{(0)} \sim \eta}[\hat{v}_i^\star(s^{(0)})] = \mathbb{E}_{\boldsymbol{\pi}^\star}[r_i]$$

Combining the two we have

$$L_r^{(t+1)}(\boldsymbol{\pi}^\star, \lambda_{\boldsymbol{\pi}}^\star) = \sum_{i=1}^{N} \mathbb{E}_{\pi_i, \pi_{-i}^\star}[r_i] - \sum_{i=1}^{N} \mathbb{E}_{\boldsymbol{\pi}^\star}[r_i]$$

which describes the differences in expected rewards. □

A.8.4 Proof to Theorem 17

Proof. Define the "MARL" objective for a single agent i where other agents have policy π_{E_i}:

$$\text{MARL}_i(r_i) = \max_{\pi_i} H_i(\pi_i) + \mathbb{E}_{\pi_i, \pi_{E_{-i}}}[r_i]$$

Define the "MAIRL" objective for a single agent i where other agents have policy π_E:

$$\text{MAIRL}_{i,\psi}(\pi^\star) = \arg\max_{r_i} \psi_i(r_i) + \mathbb{E}_{\pi_E}[r_i] - (\max_{\pi_i} H_i(\pi_i) + \mathbb{E}_{\pi_i, \pi_{E_{-i}}}[r_i])$$

Since r_i and π_i's are independent in the MAIRL objective, the solution to MAIRL_ψ can be represented by the solutions of $\text{MAIRL}_{i,\psi}$ for each i:

$$\text{MAIRL}_\psi = [\text{MAIRL}_{1,\psi}, \ldots, \text{MAIRL}_{N,\psi}]$$

Moreover, the single agent "MARL" objective $\text{MARL}_i(r_i)$ has a unique solution π_{E_i}, which also composes the (unique) solution to MARL (which we assumed in Section 10.3. Therefore,

$$\text{MARL}(\text{v}) = [\text{MARL}_1(r_1), \ldots, \text{MARL}_N(r_N)]$$

So we can use Proposition 3.1 in [HE16] for each agent i with $\text{MARL}_i(r_i)$ and $\text{MAIRL}_{i,\psi}(\pi^\star)$ and achieve the same solution as $\text{MARL} \circ \text{MAIRL}_\psi$. □

A.8.5 Proof to Proposition 5

Proof. From Corollary A.1.1 in [HE16], we have

$$\psi_{GA}^\star(\rho_{\boldsymbol{\pi}} - \rho_{\boldsymbol{\pi}_E}) = \max_{D \in (0,1)^{\mathcal{S} \times \mathcal{A}}} \mathbb{E}_{\boldsymbol{\pi}}[\log D(s,a)] + \mathbb{E}_{\boldsymbol{\pi}_E}[\log(1 - D(s,a))] \equiv D_{JS}(\rho_{\boldsymbol{\pi}}, \rho_{\boldsymbol{\pi}_E})$$

where D_{JS} denotes Jensen-Shannon divergence (which is a squared metric), and $\equiv$ denotes equivalence up to shift and scaling.

Taking the min over this we obtain

$$\arg\min_{\pi} \sum_{i=1}^{N} \psi^{\star}_{GA}(\rho_{\pi} - \rho_{\boldsymbol{\pi}_E}) = \boldsymbol{\pi}_E$$

Similarly,

$$\arg\min_{\pi} \sum_{i=1}^{N} \psi^{\star}_{GA}(\rho_{\pi_i, \pi_{E_{-i}}} - \rho_{\boldsymbol{\pi}_E}) = \boldsymbol{\pi}_E$$

So these two quantities are equal. □

Appendix B

Additional Experimental Details & Results

B.1 Chapter 2

B.1.1 Additional Experimental Details

Benchmark Tasks

Tasks We sample each dimension of $(\boldsymbol{x}, \boldsymbol{y})$ independently from a correlated Gaussian with mean 0 and correlation of ρ, where $\mathcal{X} = \mathcal{Y} = \mathbb{R}^{20}$. The true mutual information is computed as:

$$I(\boldsymbol{x}, \boldsymbol{y}) = -\frac{d}{2} \log \left(1 - \frac{\rho}{2}\right) \tag{B.1}$$

The initial mutual information is 2, and we increase the mutual information by 2 every $4k$ iterations, so the total training iterations is $20k$.

Architecture and training procedure For all the *discriminative* methods, we consider two types of architectures – *joint* and *separable*. The *joint* architecture concatenates the inputs $\boldsymbol{x}, \boldsymbol{y}$, and then passes through a two layer MLP with 256 neurons in each layer with ReLU activations at each layer. The *separable architecture* learns two separate neural networks for $\boldsymbol{x}$ and $\boldsymbol{y}$ (denoted as $g(\boldsymbol{x})$ and $h(\boldsymbol{y})$) and predicts $g(\boldsymbol{x})^\top h(\boldsymbol{y})$; g and h are two neural networks, each is a two layer MLP with 256 neurons in each layer with ReLU activations at each layer; the output of g and h are 32 dimensions.

For the generative method, we consider the invertible flow architecture described in [DKB14,

DSDB16]. p_θ, p_ϕ, p_ψ are flow models with 5 coupling layers (with scaling), where each layer contains a neural network with 2 layers of 100 neurons and ReLU activation. For all the cases, we use with the Adam optimizer [KB14] with learning rate 5×10^{-4} and $\beta_1 = 0.9, \beta_2 = 0.999$ and train for $20k$ iterations with a batch size of 64, following the setup in [POvdO+19].

Additional results We show the bias, variance and mean squared error of the *discriminative* approaches in Table B.1. We include additional results for I_{SMILE} with $\tau = 10.0$.

Table B.1: Bias, Variance and MSE of the estimators under the joint critic.

		Gaussian					Cubic				
	MI	2	4	6	8	10	2	4	6	8	10
Bias	CPC	0.25	0.99	2.31	4.00	5.89	0.72	1.48	2.63	4.20	5.99
	NWJ	0.12	0.30	0.75	2.30	2.97	0.66	1.21	2.04	3.21	4.70
	SMILE ($\tau = 1.0$)	0.15	0.30	0.32	0.18	0.03	0.47	0.77	1.16	1.64	2.16
	SMILE ($\tau = 5.0$)	0.13	0.11	0.19	0.54	0.86	0.71	1.22	1.55	1.84	2.16
	SMILE ($\tau = 10.0$)	0.14	0.21	0.22	0.11	0.19	0.70	1.28	1.83	2.44	3.02
	SMILE ($\tau = \infty$)	0.15	0.21	0.22	0.12	0.22	0.71	1.29	1.82	2.35	2.81
	GM (Flow)	0.11	0.14	0.15	0.14	0.17	1.02	0.47	1.85	2.93	3.55
Var	CPC	0.04	0.04	0.02	0.01	0.00	0.03	0.04	0.03	0.01	0.01
	NWJ	0.06	0.22	1.36	16.50	99.0	0.04	0.10	0.41	0.93	3.23
	SMILE ($\tau = 1.0$)	0.05	0.12	0.20	0.28	0.34	0.04	0.10	0.14	0.20	0.30
	SMILE ($\tau = 5.0$)	0.05	0.11	0.19	0.31	0.51	0.04	0.07	0.12	0.18	0.26
	SMILE ($\tau = 10.0$)	0.05	0.13	0.31	0.69	1.35	0.03	0.10	0.21	0.46	0.79
	SMILE ($\tau = \infty$)	0.05	0.14	0.36	0.75	1.54	0.03	0.12	0.24	0.65	0.94
	GM (Flow)	0.05	0.10	0.13	0.16	0.19	0.56	0.72	0.92	1.02	1.02
MSE	CPC	0.10	1.02	5.33	16.00	34.66	0.55	2.22	6.95	17.62	35.91
	NWJ	0.07	0.32	2.19	33.37	28.43	0.47	1.55	4.56	11.13	27.00
	SMILE ($\tau = 1.0$)	0.08	0.21	0.30	0.32	0.31	0.26	0.69	1.49	2.90	4.98
	SMILE ($\tau = 5.0$)	0.07	0.13	0.22	0.57	1.26	0.54	1.56	2.53	3.58	4.92
	SMILE ($\tau = 10.0$)	0.07	0.18	0.36	0.67	1.33	0.52	1.75	3.54	6.41	9.91
	SMILE ($\tau = \infty$)	0.08	0.19	0.40	0.76	1.62	0.54	1.75	3.55	6.09	8.81
	GM (Flow)	0.07	0.11	0.14	0.17	0.22	1.65	0.91	4.36	9.70	13.67

We show the bias, variance and MSE results in Figure B.1. We also evaluate the variance of estimating $\mathbb{E}_{Q_n}[r_\tau]$ (partition function with clipped ratios) for different values of τ in the SMILE estimator in Figure B.2b. With smaller τ we see a visible decrease in terms of variance in this term; this is consistent with the variance estimates in Figure B.1, as there the variance of $\mathbb{E}_{P_n}[\log r]$ is also considered.

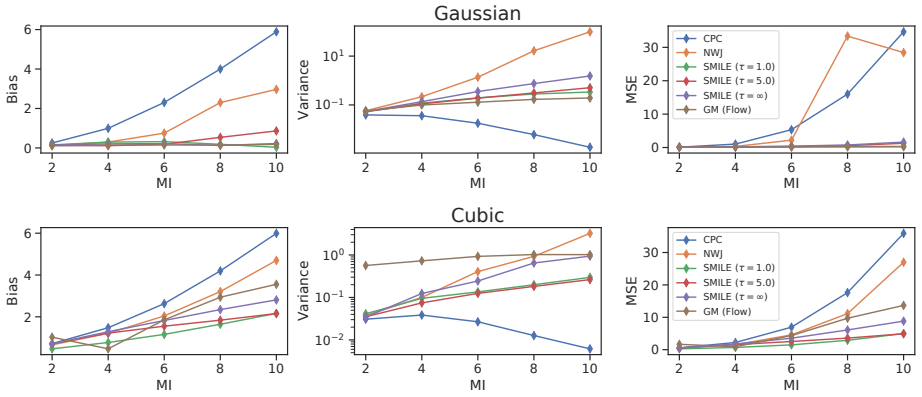
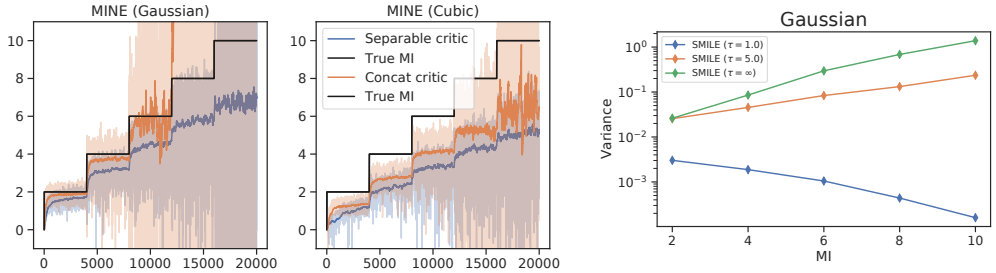


Figure B.1: Bias / Variance / MSE of various estimators, on **Gaussian** (top) and **Cubic** (down).



(a) Additional benchmark results on MINE estimator.

(b) Variance of $\mathbb{E}_{Q_n}[r_\tau]$

Figure B.2: Additional benchmark results.

Self-consistency Experiments

Tasks We consider three tasks with the mutual information estimator $\hat{I}$:

1. $\hat{I}(X;Y)$ where X is an image from MNIST [LBBH98] or CIFAR10 [KSH12] and Y is the top t rows of X. To simplify architecture designs, we simply mask out the bottom rows to be zero, see Figure 3.3.

2. $\hat{I}([X,X];[Y;h(Y)])$ where X is an image, Y is the top t rows of X, $h(Y)$ is the top $(t-3)$ rows of Y and $[\cdot,\cdot]$ denotes concatenation. Ideally, the prediction should be close to $\hat{I}(X;Y)$.

3. $\hat{I}([X_1,X_2],[Y_1,Y_2])$ where X_1 and X_2 are independent images from MNIST or CIFAR10, Y_1 and Y_2 are the top t rows of X_1 and X_2 respectively. Ideally, this prediction should be close to $2\cdot\hat{I}(X;Y)$.

Architecture and training procedure We consider the same architecture for all the *discriminative* approaches. The first layer is a convolutional layer with 64 output channels, kernel size of 5, stride of 2 and padding of 2; the second layer is a convolutional layer with 128 output channels, kernel size of 5, stride of 2 and padding of 2. This is followed another fully connected layer with 1024 neurons and finally a linear layer that produces an output of 1. All the layers (except the last one) use ReLU activations. We stack variables over the channel dimension to perform concatenation.

For the generative approach, we consider the following VAE architectures. The encoder architecture is identical to the *discriminative approach* except the last layer has 20 outputs that predict the mean and standard deviations of 10 Gaussians respectively. The decoder for MNIST is a two layer MLP with 400 neurons each; the decoder for CIFAR10 is the corresponding transposed convolution network for the encoder. All the layers (except the last layers for encoder and decoder) use ReLU activations. For concatenation we stack variables over the channel dimension. For all the cases, we use with the Adam optimizer [KB14] with learning rate 10^{-4} and $\beta_1 = 0.9, \beta_2 = 0.999$. For I_{GM} we train for 10 epochs, and for the *discriminative* methods, we train for 2 epochs, due to numerical stability issues of I_{MINE}.

Additional experiments on scaling, rotation and translation We consider additional benchmark experiments on MNIST where instead of removing rows, we apply alternative transformations such as random scaling, rotation and translations. For random scaling, we upscale the image randomly by 1x to 1.2x; for random rotation, we randomly rotate the image between ± 20 degrees; for random translation, we shift the image randomly by no more than 3 pixels horizontally and vertically. We consider evaluating the data processing and additivity properties, where the ideal value for the former is no more than 1, and the ideal value for the latter is 2. From the results in Table B.2, none of the considered approaches achieve good results in all cases.

Table B.2: Self-consistency experiments on other image transforms.

		CPC	MINE	GM (VAE)	SMILE ($\tau = 5.0$)	SMILE ($\tau = \infty$)
DP	Scaling	1.00	1.03	1.12	1.19	1.04
	Rotation	1.00	1.30	1.13	1.03	1.27
	Translation	1.00	1.28	1.01	1.07	1.08
Additivity	Scaling	1.00	1.55	1.89	1.04	1.18
	Rotation	1.00	2.09	1.58	1.50	1.78
	Translation	1.00	1.41	1.28	1.32	1.33

B.2 Chapter 3

Time complexity of gradient calculation in ML-CPC

Suppose g is a neural network parametrized by θ, then the gradient to the ML-CPC objective is

$$\nabla_\theta J(g_\theta) = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\frac{\nabla_\theta g_\theta(\boldsymbol{x}_i, \boldsymbol{y}_i)}{g_\theta(\boldsymbol{x}_i, \boldsymbol{y}_i)} - \frac{\sum_{j=1}^{n}\nabla_\theta g_\theta(\boldsymbol{x}_j, \boldsymbol{y}_j) + \sum_{j=1}^{n}\sum_{k=1}^{m-1}\nabla_\theta g_\theta(\boldsymbol{x}_j, \overline{\boldsymbol{y}_{j,k}})}{\sum_{j=1}^{n} g(\boldsymbol{x}_j, \boldsymbol{y}_j) + \sum_{j=1}^{n}\sum_{k=1}^{m-1} g(\boldsymbol{x}_j, \overline{\boldsymbol{y}_{j,k}})}\right]$$

Computing the gradient through the an empirical estimate of $J(g_\theta)$ requires us to perform back-propagation through all nm critic evaluations, which is identical to the amount of back-propagation passes needed for CPC. So the time complexity to compute the ML-CPC gradient is $\mathcal{O}(nm)$.

B.2.1 Pseudo-code and PyTorch implementation to ML-CPC

We include a PyTorch implementation to α-ML-CPC as follows.

```
def ml_cpc(logits, alpha):
    """
    We assume that logits are of shape (n, m),
    and the predictions over positive are logits[:, 0].
    Alternatively, one can use kl_div() to ensure that the loss is non-negative.
    """
    n, m = logits.size(0), logits.size(1)
    beta = (m - alpha) / (m - 1)
    pos = logits.select(1, 0)
    neg = logits.narrow(1, 1, m)
    denom = torch.cat([pos + torch.log(torch.tensor(alpha)).float(),
                       neg + torch.log(torch.tensor(beta)).float()], dim=1)
    denom = denom.logsumexp(dim=1).logsumexp(dim=0)
    loss = denom - pos.sum()
    return loss / n
```

To ensure that the loss value is non-negative, one can alternatively use the `kl_div()` function that evaluates the KL divergence between the predicted label distribution with a ground truth label distribution. This is equivalent to the negative of the α-ML-CPC objective shifted by a constant. We describe this idea in the following algorithm.

B.2.2 Experimental Details

Binary simulation experiments

Let X, Y be two binary r.v.s such that $\Pr(X = 1, Y = 1) = p$, $\Pr(X = 0, Y = 0) = 1 - p$. We can simulate the case of a batch size of n with $n - 1$ negative samples. For the example of CPC, we

Algorithm 6 Pseudo-code for α-ML-CPC

1: **Input**: the critic g, input values $\boldsymbol{x}_i$, $\boldsymbol{y}_i$, $\overline{\boldsymbol{y}_{j,k}}$
2: **Output**: shifted negative objective value $J_\alpha(g)$ for optimization
3: Compute logit values $\ell_{i,i} = \log g(\boldsymbol{x}_i, \boldsymbol{y}_i) + \log \alpha$ and $\overline{\ell_{j,k}} = \log g(\boldsymbol{x}_j, \overline{\boldsymbol{y}_{j,k}}) + \log \frac{m-\alpha}{m-1}$.
4: Compute the normalization value $Z = \sum_i \exp(\ell_{i,i}) + \sum_{j,k} \exp(\overline{\ell_{j,k}})$.
5: Compute the predicted probabilities $p'_{i,i} = \exp(\ell_{i,i})/Z$, $\overline{p'_{j,k}} = \exp(\overline{\ell_{j,k}})/Z$
6: Assign the ground truth probabilities $p_{i,i} = 1/n$, $\overline{p_{j,k}} = 0$.
7: Compute the KL divergence between p and p'.

have:

$$L(g) := \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} \log \frac{n \cdot g(\boldsymbol{x}_i, \boldsymbol{y}_i)}{\sum_{j=1}^{n} g(\boldsymbol{x}_i, \boldsymbol{y}_j)}\right] \tag{B.2}$$

Since we are drawing from the above distribution, $\boldsymbol{x}_i = \boldsymbol{y}_i$ is always true; therefore, we only need to enumerate how many $\boldsymbol{y}_j$ are different from $\boldsymbol{x}_i$ in order to compute one term of the expectation. In the case where we have t pairs of $(1, 1)$ and $(n-t)$ pairs of $(0, 0)$, then for $g(1, 1) = g(0, 0) = 1$, $g(0, 1) = g(1, 0) = 0$ we have that:

$$\frac{1}{n}\sum_{i=1}^{n} \log \frac{n \cdot g(\boldsymbol{x}_i, \boldsymbol{y}_i)}{\sum_{j=1}^{n} g(\boldsymbol{x}_i, \boldsymbol{y}_j)} = \frac{1}{n}\left(t \log \frac{n}{t} + (n-t) \log \frac{n}{n-t}\right) \tag{B.3}$$

Moreover the probability of such an arrangement can be computed from the Binomial distribution

$$\Pr(t \text{ pairs of } (1, 1)) = \binom{n}{t} p^t (1-p)^{n-t} \tag{B.4}$$

Therefore, we can compute the expectation that is $L(g)$ in closed form by computing the sum for t from 0 to n. We can apply a similar argument to computing the mean of ML-CPC values as well as the variance of the empirical estimates. This allows us to analytically compute the optimal value of the objective values, which allows us to perform direct comparisons over them.

Mutual information estimation

The general procedure follows that in [POvdO+19] and [SE19b].

Tasks We sample each dimension of $(\boldsymbol{x}, \boldsymbol{y})$ independently from a correlated Gaussian with mean 0 and correlation of ρ, where $\mathcal{X} = \mathcal{Y} = \mathbb{R}^{20}$. The true mutual information is computed as: $I(\boldsymbol{x}, \boldsymbol{y}) = -\frac{d}{2} \log\left(1 - \frac{\rho}{2}\right)$ The initial mutual information is 2, and we increase the mutual

information by 2 every $4k$ iterations.

Architecture and training procedure We consider two types of architectures – *joint* and *separable*. The *joint* architecture concatenates the inputs $\boldsymbol{x}, \boldsymbol{y}$, and then passes through a two layer MLP with 256 neurons in each layer with ReLU activations at each layer. The *separaable architecture* learns two separate neural networks for $\boldsymbol{x}$ and $\boldsymbol{y}$ (denoted as $g(\boldsymbol{x})$ and $h(\boldsymbol{y})$) and predicts $g(\boldsymbol{x})^\top h(\boldsymbol{y})$; g and h are two neural networks, each is a two layer MLP with 256 neurons in each layer with ReLU activations at each layer; the output of g and h are 32 dimensions. For all the cases, we use with the Adam optimizer [KB14] with learning rate 1×10^{-3} and $\beta_1 = 0.9, \beta_2 = 0.999$ and train for $20k$ iterations with a batch size of 128.

Knowledge distillation

The general procedure follows that in [TKI19], where we use the same training hyperparameters. Specifically, we train for 240 epochs with the SGD optimizer with a momentum of 0.9 and weight decay of 5×10^{-4}. We use a default initial learning rate of 0.1, and divide the learning rate by 10 at 150, 180 and 210 epochs. We use 16384 negative samples per positive sample [1], and a temperature of 0.07 for the critic. We did not additiaonlly include the knowledge distillation loss to reduce potential compounding effects over the representation learning performance.

Unsupervised representation learning

For CIFAR10, the general procedure follows that of MoCo-v2 [HFW+19, CFGH20], with some slight changes adapted to CIFAR-10. First, we use the standard ResNet50 adaptation of 3×3 kernels instead of 7×7 kernels used for the larger resolution ImageNet, with representation learning dimension of 2048. Next, we use a temperature of $\tau = 0.07$, a batch size of 256 and a learning rate of 0.3 for the representation learner, and a learning rate of 3 for the linear classifier; we observe that these hyperparameters combinations is able to achieve higher performance on the CPC objective for CIFAR-10, so we use these for all our other experiments. The remaining hyperparameters are identical to the ImageNet setup for MoCo-v2. For ImageNet, we use the same procedure as that of MoCo-v2, except that we trained the representations for merely 30 epochs with a ResNet-18 network, instead of training on ResNet-50 for 800 epochs.

[1] We note that this is smaller than what is used in [TKI19], and it is possible to achieve additional (though not much) improvements by using more negative samples.

B.3 Chapter 4

B.3.1 Experimental Setup Details

We consider the following setup for our experiments.

- For MIFR, we modify the weight for reconstruction error $\alpha = 1$, $\lambda_1 \in \{0.0, 0.1, 0.2, 1.0, 2.0\}$ and $\lambda_2 \in \{0.1, 0.2, 1.0, 2.0, 5.0\}$ for the constraints, which creates a total of $5^2 = 25$ configurations; λ_1 values smaller since high values of λ_1 prefers solutions with low $I_q(\mathbf{x}; \mathbf{z}|\mathbf{u})$.
- For L-MIFR, we modify ϵ_1 and ϵ_2 according to the estimated values for each dataset. This allows us to claim results that holds for a certain hyperparameter in general (even as other hyperparameter change).
- We use the Adam optimizer with initial learning rate $1e-3$ and $\beta_1 = 0.5$ where the learning rate is multiplied by 0.98 every 1000 optimization iterations, following common settings for adversarial training [GAA+17].
- For L-MIFR, we initialize the λ_i parameters to 1.0, and allow for a range of $(0.01, 100)$.
- Unless otherwise specified, we update $p_\psi(\mathbf{u}|\mathbf{z})$ 10 times per update of $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{u})$ and $p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{u})$.
- For *Adult* and *Health* we optimize for 2000 epochs; for *German* we optimize for 10000 epochs (since there are only 1000 low dimensional data points).
- For both cases, we consider $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{u}), p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{u}), p_\psi(\mathbf{u}|\mathbf{z})$ as a two layer neural networks with a hidden layer of 50 neurons with softplus activations, and use $\mathbf{z}$ of dimension 10 for *German* and *Adult*, and 30 for *Health*. For the joint of two variables (i.e. $(\mathbf{x}, \mathbf{u})$) we simply concatenate them at the input layer. We find that our conclusions are insensitive to a reasonable change in architectures (e.g. reduce number of neurons to 50 and $\mathbf{z}$ to 25 dimensions).

B.3.2 Comparison with LAFTR

Our work have several notable differences from prior methods (such as LAFTR [MCPZ18]) that make it hard to compare them directly. First, we do not assume access to the prediction task while learning the representation, thus our method does not directly include the "classification error" objective. Second, our method is able to deal with any type of sensitive attributes, as opposed to binary ones.

Nevertheless, we compare the performance of MIFR and LAFTR (Madras et al.) with the demographic parity notion of fairness (measured by $Delta_{DP}$, lower is better). To make a fair comparison, we add a classification error to MIFR during training. MIFR achieves an accuracy of 0.829 and Δ_{DP} of 0.037, whereas LAFTR achieves an accuracy of 0.821 and Δ_{DP} of 0.029. This shows that MIFR and LAFTR are comparable in terms of the accuracy / fairness trade-off. MIFR is still useful for sensitive attributes that are not binary, such as Health, which LAFTR cannot handle.

We further show a comparison of Δ_{DP}, Δ_{EO}, Δ_{EOpp} between L-MIFR and LAFTR [MCPZ18] on the Adult dataset in Table B.3, where L-MIFR is trained with the procedure in Section 5.5.6. While LAFTR achieves better fairness on each notion if it is specifically trained for that notion, it often achieves worse performance on other notions of fairness. We note that L-MIFR uses a logistic regression classifier, whereas LAFTR uses a one layer MLP. Moreover, these measurements are also task-specific as opposed to mutual information criterions.

Table B.3: Comparison between L-MIFR and LAFTR on Δ_{DP}, Δ_{EO}, Δ_{EOpp} metrics from [MCPZ18]. While LAFTR achieves better fairness on individual notions if it is trained for that notion, it often trades that with other notions of fairness.

	Δ_{DP}	Δ_{EO}	Δ_{EOpp}
L-MIFR	0.057	0.123	0.026
LAFTR-DP	0.029	0.244	0.027
LAFTR-EO	0.125	0.074	0.037
LAFTR-EOpp	0.098	0.154	0.022

B.3.3 Extension to Equalized Odds and Equalized Opportunity

If we are also provided labels y for a particular task, in the form of $\mathcal{D}_l = \{(\mathbf{x}_i, \mathbf{u}_i, y_i)\}_{i=1}^M$, we can also use the representations to predict y, which leads to a third condition:

3. **Classification z** can be used to classify y with high accuracy.

We can either add this condition to the primal objective in Equation 5.1, or add an additional constraint that we wish to have accuracy that is no less than a certain threshold.

With access to binary labels, we can also consider information-theoretic approaches to *equalized odds* and *equalized opportunity* [HPS16]. Recall that *equalized odds* requires that the predictor and sensitive attribute are independent conditioned on the label, whereas *equalized opportunity* requires that the predictor and sensitive attribute are independent conditioned on the label being positive.

In the case of learning representations for downstream tasks, our notions should consider any classifier over $\mathbf{z}$.

For *equalized odds*, we require that z and u have low mutual information conditioned on the label, which is $I_q(\mathbf{z}, \mathbf{u}|y)$. For *equalized opportunity*, we require that z and u have low mutual information conditioned on the label $y = 1$, which is $I_q(\mathbf{z}, \mathbf{u})|_{y=1}$.

We can still apply the upper bounds similar to the case in C_2. For *equalized opportunity* we have

$$\begin{aligned} I_q(\mathbf{z}; \mathbf{u})|_{y=1} &\leq \mathbb{E}_{q_\phi(\mathbf{z},\mathbf{u},y|y=1)}[D_{\mathrm{KL}}(q_\phi(\mathbf{u}|\mathbf{z}, y)\|p(\mathbf{u}))] - D_{\mathrm{KL}}(q(\mathbf{u})\|p(\mathbf{u})) := I_{EO} \\ &\leq \mathbb{E}_{q_\phi(\mathbf{z},\mathbf{u},y|y=1)}[D_{\mathrm{KL}}(q_\phi(\mathbf{u}|\mathbf{z}, y)\|p(\mathbf{u}))] \end{aligned}$$

For *equalized odds* we have

$$\begin{aligned} I_q(\mathbf{z}; \mathbf{u}|y) &= q(1)I_q(\mathbf{z}; \mathbf{u})|_{y=1} + q(0)I_q(\mathbf{z}; \mathbf{u})|_{y=0} := I_{EOpp} \\ &\leq q(1)\mathbb{E}_{q_\phi(\mathbf{z},\mathbf{u},y|y=1)}[D_{\mathrm{KL}}(q_\phi(\mathbf{u}|\mathbf{z}, y)\|p(\mathbf{u}))] + q(0)\mathbb{E}_{q_\phi(\mathbf{z},\mathbf{u},y|y=0)}[D_{\mathrm{KL}}(q_\phi(\mathbf{u}|\mathbf{z}, y)\|p(\mathbf{u}))] \end{aligned}$$

which can be implemented by using a separate classifier for each y or using y as input. If y is an input to the classifier, our mutual information formulation of *equalized odds* does not have to be restricted to the case where y is binary.

B.4 Chapter 5

B.4.1 Example KL-WGAN Implementation in PyTorch

```
def get_kl_ratio(v):
    vn = torch.logsumexp(v.view(-1), dim=0) - torch.log(torch.tensor(v.size(0)).float())
    return torch.exp(v - vn)

def loss_kl_dis(dis_fake, dis_real, temp=1.0):
    """
    Critic loss for KL-WGAN.
    dis_fake, dis_real are the critic outputs for generated samples and real samples.
    temp is a hyperparameter that scales down the critic outputs.
    We use the hinge loss from BigGAN PyTorch implementation.
    """
    loss_real = torch.mean(F.relu(1. - dis_real))
    dis_fake_ratio = get_kl_ratio(dis_fake / temp)
    dis_fake = dis_fake * dis_fake_ratio
    loss_fake = torch.mean(F.relu(1. + dis_fake))
    return loss_real, loss_fake

def loss_kl_gen(dis_fake, temp=1.0):
    """
    Generator loss for KL-WGAN.
```

```
dis_fake is the critic outputs for generated samples.
temp is a hyperparameter that scales down the critic outputs.
We use the hinge loss from BigGAN PyTorch implementation.
"""
dis_fake_ratio = get_kl_ratio(dis_fake / temp)
dis_fake = dis_fake * dis_fake_ratio
loss = -torch.mean(dis_fake)
return loss
```

B.4.2 Argument about χ^2-Divergences

We present a similar argument to Theorem 9 to χ^2-divergences, where $f(u) = (u-1)^2$.

Theorem 21. *Let $f(u) = (u-1)^2$ and $\mathcal{F}$ is a set of real-valued bounded measurable functions on $\mathcal{X}$. For any fixed choice of P, Q, and $T \in \mathcal{F}$ such that $T \geq 0, T - \mathbb{E}[T] + 2 \geq 0$, we have*

$$\arg\min_{r \in \Delta(Q)} \mathbb{E}_Q[f(r)] + \mathbb{E}_P[T] - \mathbb{E}_{Q_r}[T] = \frac{T - \mathbb{E}_Q[T] + 2}{2}$$

Proof. Consider the following Lagrangian:

$$h(r, \lambda) := \mathbb{E}_Q[f(r)] - \mathbb{E}_Q[r \cdot T] + \lambda(\mathbb{E}_Q[r] - 1) \tag{B.5}$$

where $\lambda \in \mathbb{R}$ and we formalize the constraint $r \in \Delta(r)$ with $\mathbb{E}_Q[r] - 1 = 0$. Taking the functional derivative $\partial h / \partial r$ and setting it to zero, we have:

$$\begin{aligned} & f'(r)\,\mathrm{d}Q - T\,\mathrm{d}Q + \lambda \\ &= 2r\,\mathrm{d}Q - T\,\mathrm{d}Q + \lambda = 0, \end{aligned} \tag{B.6}$$

so $r = (T - \lambda)/2$. We can then apply the constraint $\mathbb{E}_Q[r] = 1$, where we solve $\lambda = \mathbb{E}_Q[T] - 2$, and consequently the optimal $r = (T - \mathbb{E}_Q[T] + 2)/2 \in \Delta(Q)$. □

In practice, when the constraint $T - \mathbb{E}_Q[T] + 2 \geq 0$ is not true, then one could increase the values when T is small, using

$$\hat{T} = \max(T, c) + b \tag{B.7}$$

where b, c are some constants that satisfies $\hat{T(\boldsymbol{x})} - \mathbb{E}_Q[\hat{T}] + 2 \geq 0$ for all $\boldsymbol{x} \in \mathcal{X}$. Similar to the KL case, we encourage higher weights to be assigned to higher quality samples.

If we plug in this optimal r, we obtain the following objective:

$$\mathbb{E}_P[T] - \mathbb{E}_Q[T] + \frac{1}{4}\mathbb{E}_Q[T^2] + \frac{1}{4}(\mathbb{E}_Q[T])^2 = \mathbb{E}_P[T] - \mathbb{E}_Q[T] - \frac{\mathrm{Var}_Q[T]}{4}. \tag{B.8}$$

Let us now consider $P = p_{\mathrm{data}}$, $Q = \frac{p_{\mathrm{data}}+G_\theta}{2}$, then the f-divergence corresponding to $f(u) = (u-1)^2$:

$$D_f(P\|Q) = \int_{\mathcal{X}} \frac{(P(\boldsymbol{x}) - Q(\boldsymbol{x}))^2}{\frac{P(\boldsymbol{x})+Q(\boldsymbol{x})}{2}}\,\mathrm{d}\boldsymbol{x}, \tag{B.9}$$

is the squared χ^2-distance between P and Q. So the objective becomes:

$$\min_\theta \max_\phi \mathbb{E}_{p_{\mathrm{data}}}[D_\theta] - \mathbb{E}_{G_\theta}[D_\phi] - \mathrm{Var}_{M_\theta}[D_\phi], \tag{B.10}$$

where $M_\theta = (p_{\mathrm{data}} + G_\theta)/2$ and we replace $T/2$ with D_ϕ. In comparison, the χ^2-GAN objective [TCH+18] for θ is:

$$\frac{(\mathbb{E}_{p_{\mathrm{data}}}[D_\theta] - \mathbb{E}_{G_\theta}[D_\phi])^2}{\mathrm{Var}_{M_\theta}[D_\phi]}. \tag{B.11}$$

They do not exactly minimize χ^2-divergence, or a squared χ^2-divergence, but a normalized version of the 4-th power of it, hence the square term over $\mathbb{E}_{p_{\mathrm{data}}}[D_\theta] - \mathbb{E}_{G_\theta}[D_\phi]$.

B.4.3 Additional Experimental Details

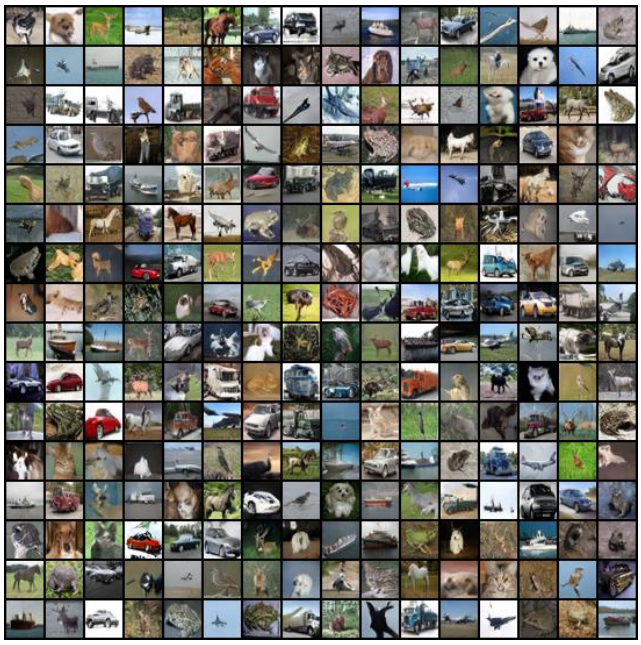
For 2d experiments, we consider the WGAN and KL-WGAN objectives with the same architecture and training procedure. Specifically, our generator is a 2 layer MLP with 100 neurons and LeakyReLU activations on each hidden layer, with a latent code dimension of 2; our discriminator is a 2 layer MLP with 100 neurons and LeakyReLU activations on each hidden layer. We use spectral normalization [MKKY18] over the weights for the generators and consider the hinge loss in [MKKY18]. Each dataset contains 5,000 samples from the distribution, over which we train both models for 500 epochs with RMSProp (learning rate 0.2). The procedure for tabular experiments is identical except that we consider networks with 300 neurons in each hidden layer with a latent code dimension of 10.

B.4.4 Samples

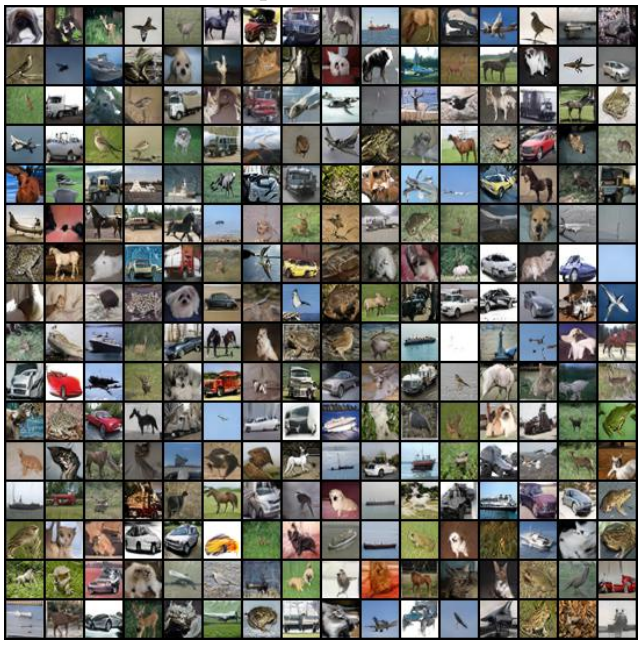
We show uncurated samples from BigGAN trained with WGAN and KL-WGAN loss in Figures B.3a, B.4a, B.3b, and B.4b.

(a) CelebA 64x64 samples trained with WGAN.

(b) CelebA 64x64 Samples trained with KL-WGAN.



(a) CIFAR samples trained with WGAN.



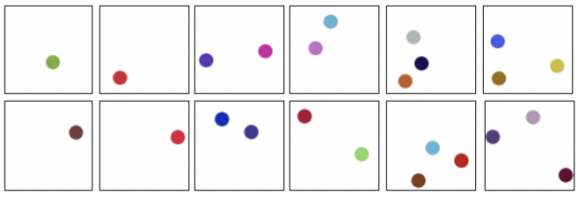
(b) CIFAR samples trained with KL-WGAN.

B.5 Chapter 6

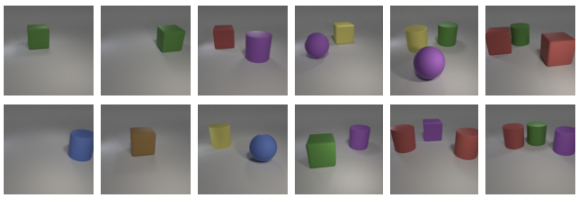
B.5.1 Numerosity Containment

[ZRY+18] systematically investigate generalization in deep generative models using two different datasets: (a) a toy dataset where there are k non-overlapping dots (with random color and location) in the image (see Figure B.5a), and (b) the CLEVR dataset where ther are k objects (with random shape, color, location, and size) in the images (see Figure B.5b). They train a GAN model (WGAN-GP [GAA+17]) with (either) dataset and observe that the learned distribution does not produce the same number of objects as in the dataset it was trained on. The distribution of the numerosity in the generated images is centered at the numerosity from the dataset, with a slight-bias towards over-estimation. For, example when trained on images with six dots, the generated images contain anywhere from two to eight dots (see Figure B.6a). The observation is similar when trained on images with two CLEVR objects. The generated images contain anywhere from one to three dots (see Figure B.6b).

In order to remove samples with numerosity different from the train dataset, we use such samples as negative data during training. For example, while training on images with six dots we use images with four, five and seven dots as negative data for the GAN. The resulting distribution of the numerosity in the generated images is constrained to six. We observe similar behaviour when training a GAN with images containing two CLEVR objects as positive data and images with one or three objects as negative data.

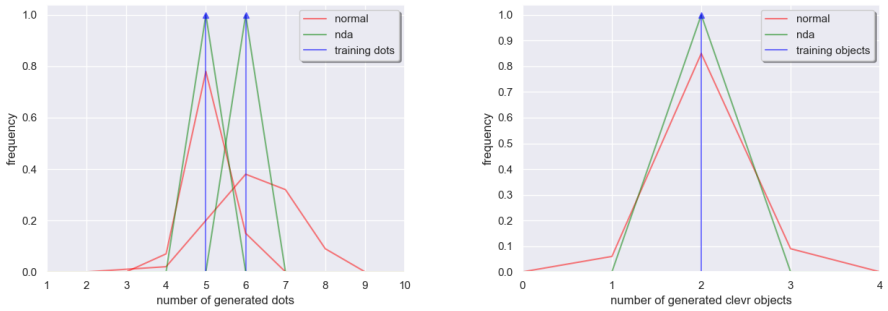


(a) Dots



(b) CLEVR

Figure B.5: Toy Datasets used in Numerosity experiments.



(a) (b)

Figure B.6: **Left**: Distribution over number of dots. The arrows are the number of dots the learning algorithm is trained on, and the solid line is the distribution over the number of dots the model generates. **Right**: Distribution over number of CLEVR objects the model generates. Generating CLEVR is harder so we explore only one, but the behaviour with NDA is similar to dots.

B.5.2 Image Transformations

Given an image of size $H \times W$, the different image transformations that we used are described below.

Jigsaw-K [NF16] We partition the image into a grid of $K \times K$ patches of size $(H/K) \times (W/K)$, indexed by $[1, \ldots, K \times K]$. Then we shuffle the image patches according to a random permutation (different from the original order) to produce the NDA image. Empirically, we find $K = 2$ to work the best for Jigsaw-K NDA.

Stitching We stitch two equal-sized patches of two different images, either horizontally $(H/2 \times W)$ or vertically $(H \times W/2)$, chosen uniformly at random, to produce the NDA image.

Cutout / Cutmix We select a random patch in the image with its height and width lying between one-third and one-half of the image height and width respectively. To construct NDA images, this patch is replaced with the mean pixel value of the patch (like cutout [DT17] with the only difference that they use zero-masking), or the pixel values of another image at the same location (cutmix [YHO+19]).

Mixup-α NDA image is constructed from a linear interpolation between two images $\boldsymbol{x}$ and $\boldsymbol{y}$ [ZCDLP17], $\gamma\boldsymbol{x} + (1-\gamma)\boldsymbol{y}$; $\gamma \sim \mathrm{Beta}(\alpha, \alpha)$. α is chosen so that the distribution has high density at 0.5.

Other classes NDA images are sampled from other classes in the same dataset. See Appendix A.

B.5.3 What does the theory over GANs entail?

Our goal is to show that NDA GAN objectives are principled in the sense that with infinite computation, data, and modeling capacity, NDA GAN will recover the same optimal generator as a regular GAN. In other words, under these assumptions, NDA will not bias the solution in an undesirable way. We note that the NDA GAN objective is as stable as regular GAN in practice since both methods estimate a lower bound to the divergence with the discriminator, and then minimize that lower bound w.r.t. the generator. The estimated divergences are slightly different, but they have the same minimizer (which is the ground truth data distribution). Intuitively, while GAN and NDA GAN will give the same solution asymptotically, NDA GAN might get there faster (with less data) because it leverages a stronger prior over what the support should (not) be.

B.5.4 Pix2Pix

Figure B.7 highlights the qualitative improvements when we apply the NDA method to Pix2Pix model [IZZE17].

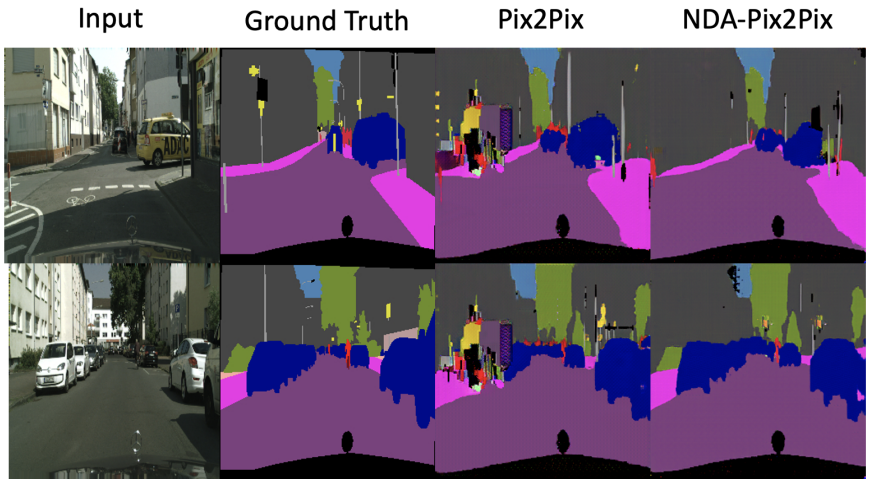
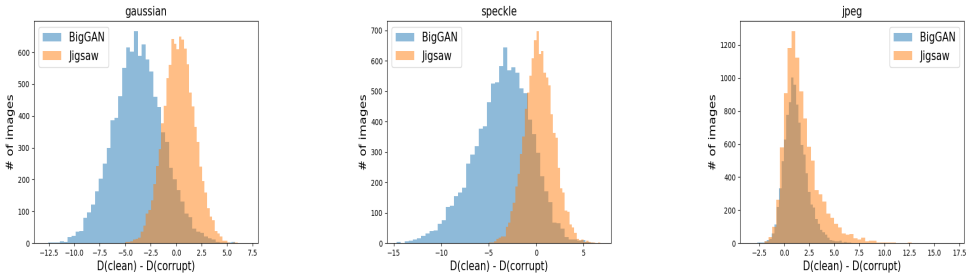


Figure B.7: Qualitative results on Cityscapes.

B.5.5 Anomaly Detection

Here, we show the histogram of difference in discriminator's output for clean and OOD samples in Figure B.8. High difference values imply that the Jigsaw NDA is better at distinguishing OOD samples than the normal BigGAN.



(a) Gaussian Noise (b) Speckle Noise (c) JPEG Compression

Figure B.8: Histogram of D(clean) - D(corrupt) for 3 different corruptions.

B.5.6 Effect of hyperparameter on Unconditional Image generation

Here, we show the effect of λ for unconditional image generation on CIFAR-10 dataset.

Table B.4: Effect of λ on the FID score for unconditional image generation on CIFAR-10 using Jigsaw as NDA.

λ	1.0	0.75	0.5	0.25	0.15
FID	18.64	16.61	14.95	**12.61**	13.01

B.5.7 Unsupervised Learning on Images

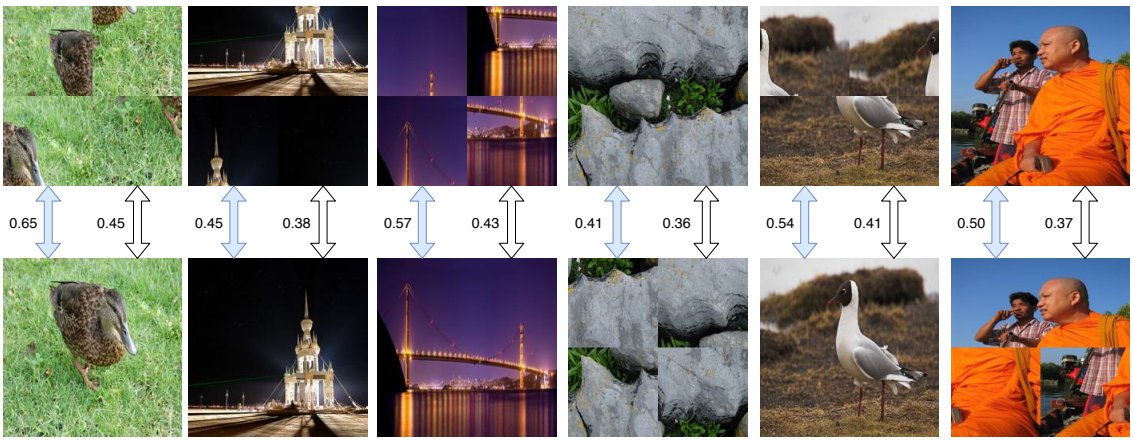


Figure B.9: Comparing the cosine distance of the representations learned with Jigsaw NDA and Moco-V2 (**shaded blue**), and original Moco-V2 (**white**). With NDA, we project normal and its jigsaw image representations further away from each other than the one without NDA.

Dataset Preparation for FID evaluation For dataset preparation, we follow the the following procedures: (a) CIFAR-10 contains 60K 32×32 images with 10 labels, out of which 50K are used for training and 10K are used for testing, (b) CIFAR-100 contains 60K 32 × 32 images with 100 labels, out of which 50K are used for training and 10K are used for testing, (c) CelebA contains 162,770 train images and 19,962 test images (we resize the images to 64×64px), (d) STL-10 contains 100K (unlabeled) train images and 8K (labeled) test images (we resize the images to 32×32px). In our experiments the FID is calculated on the test dataset. In particular, we use 10K generated images vs. 10K test images for CIFAR-10, 10K vs. 10K for CIFAR-100, 19,962 vs. 19,962 for CelebA, and 8K vs 8K for STL-10.

Hyperparameters and Network Architecture

Generative Modeling. We use the same network architecture in BigGAN [BDS18] for our experiments. The code used for our experiments is based over the author's PyTorch code. For CIFAR-10, CIFAR-100, and CelebA we train for 500 epochs whereas for STL-10 we train for 300 epochs. For all the datasets we use the following hyperparameters: batch-size = 64, generator learning rate = 2e-4, discriminator learning rate = 2e-4, discriminator update steps per generator update step = 4. The best model was selected on the basis of FID scores on the test set (as explained above).

Momentum Contrastive Learning. We use the official PyTorch implementation for our experiments. For CIFAR-10 and CIFAR-100, we perform unsupervised pre-training for 1000 epochs and supervised training (linear classifier) for 100 epochs. For Imagenet-100, we perform unsupervised pre-training for 200 epochs and supervised training (linear classifier) for 100 epochs. For CIFAR-10 and CIFAR-100, we use the following hyperparameters during pre-training: batch-size = 256, learning-date = 0.3, temperature = 0.07, feature dimensionality = 2048. For ImageNet-100 pre-training we have the following: batch-size = 128, learning-date = 0.015, temperature = 0.2, feature dimensionality = 128. During linear classification we use a batch size of 256 for all the datasets and learning rate of 10 for CIFAR-10, CIFAR-100, whereas for ImageNet-100 we use learning rate of 30.

Dense Predictive Coding. We use the same network architecture and hyper-parameters in DPC [HXZ19] for our experiments and use the official PyTorch implementation. We perform

self-supervised training on UCF-101 for 200 epochs and supervised training (action classifier) for 200 epochs on both UCF-101 and HMDB51 datasets.

Code The code to reproduce our experiments is given here.

Implementation Details For our experiment over GAN, we augment the batch of real samples with a negative augmentation of the same batch, and we treat the augmented images as fake images for the discriminator. Similarly, for the contrastive learning experiments, we consider negative augmentation of the query image batch as negatives for that batch.

For all our experiments we used existing open-source models. For experiments over GAN, we use the open-source implementations of BigGAN and Pix2Pix models, and for contrastive learning, we use the open-source implementation of the MoCo-v2 model and Dense Predictive Coding. Hence, we did not explain in detail each of the models. Implementing NDA is quite simple as we only need to generate NDA samples from the images in a mini-batch which only takes several lines of code.

Ablation study on negative samples We perform the experiments over MoCo-v2 which maintains a queue of negative samples. The number of negatives is around 65,536. With our approach, we use the augmented versions of images in the same batch as negative. We transform both the key and query images to create NDA samples. Thus, the number of negatives for our approach is 65,536 + 2 (one NDA sample created using query image and other using key image), only 0.00003051664 times more than the original number of negatives samples in MoCo-v2. Thus our experiments are comparable to the baseline MoCo-v2. In terms of computation, we need an additional forward pass in each batch to get the representations of the NDA samples. The normal MoCo-v2 requires 1.09 secs for entire forward computation, which includes forward pass through the network, momentum update of the key encoder and dot product between the positive and negative samples. With NDA, 1 forward computation requires 1.36 secs.

What happens when negative data augmentations are noisy? Regarding the performance of negative data augmentation, we perform 2 different experiments:

a) When the noise is low - When using jigsaw as our NDA strategy with a 2 x 2 grid, one out of the 24 permutations will be the original image. We find that when this special permutation is not removed, or there is 4% "noise", the FID score is 12.61, but when it is removed the FID score is 12.59. So, we find that when the noise is low, the performance of our approach is not greatly affected and is robust in such scenarios.

b) When the noise is large - We use random vertical flipping as our NDA strategy, where with 50% probability the image is vertically flipped during NDA. In this case, the "noise" is large, as 50% of the time, the negative sample is actually the original image. We contrast this with the "noise-free" NDA strategy where the NDA image is always vertically flipped. We find that for the random vertical flipping NDA, the FID score of BigGAN is 15.84, whereas, with vertical flipping NDA, the FID score of BigGAN is 14.74. So performance degrades with larger amounts of noise.

B.6 Chapter 7

B.6.1 Additional Details for D2C

Contrastive representation learning

In contrastive representation learning, the goal is to distinguish a *positive* pair $(\mathbf{y}, \mathbf{w}) \sim p(\boldsymbol{x}, \boldsymbol{y})$ from $(m-1)$ *negative* pairs $(\mathbf{y}, \overline{\mathbf{w}}) \sim p(\boldsymbol{x})p(\boldsymbol{y})$. In our context, the positive pairs are representations from the same image, and negative pairs are representations from different images; these images are pre-processed with strong data augmentations [CKNH20] to encourage rich representations. With two random, independent data augmentation procedures defined as aug_1 and aug_2, we define $p(\boldsymbol{x}, \boldsymbol{y})$ and $p(\boldsymbol{x})p(\boldsymbol{y})$ via the following sampling procedure:

$$(\mathbf{y}, \mathbf{w}) \sim p(\boldsymbol{x}, \boldsymbol{y}) : \mathbf{y} \sim q_\phi(\boldsymbol{z}^{(1)}|\mathrm{aug}_1(\mathbf{x})), \mathbf{w} \sim q_\phi(\boldsymbol{z}^{(1)}|\mathrm{aug}_2(\mathbf{x})), \mathbf{x} \sim p_{\mathrm{data}}(\mathbf{x}),$$
$$(\mathbf{y}, \mathbf{w}) \sim p(\boldsymbol{x})p(\boldsymbol{y}) : \mathbf{y} \sim q_\phi(\boldsymbol{z}^{(1)}|\mathrm{aug}_1(\mathbf{x}_1)), \mathbf{w} \sim q_\phi(\boldsymbol{z}^{(1)}|\mathrm{aug}_2(\mathbf{x}_2)), \mathbf{x}_1, \mathbf{x}_2 \sim p_{\mathrm{data}}(\mathbf{x}).$$

For a batch of n positive pairs $\{(\mathbf{y}_i, \mathbf{w}_i)\}_{i=1}^n$, the contrastive predictive coding (CPC, [vdOLV18]) objective is defined as:

$$L_{\mathrm{CPC}}(g; q_\phi) := \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} \log \frac{m \cdot g(\boldsymbol{x}_i, \boldsymbol{y}_i)}{g(\boldsymbol{x}_i, \boldsymbol{y}_i) + \sum_{j=1}^{m-1} g(\boldsymbol{x}_i, \overline{\boldsymbol{y}_{i,j}})}\right] \tag{B.12}$$

for some positive critic function $g : \mathcal{Y} \times \mathcal{Z} \to \mathbb{R}_+$, where the expectation is taken over n positive pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i) \sim p(\boldsymbol{x}, \boldsymbol{y})$ and $n(m-1)$ negative pairs $(\boldsymbol{x}_i, \overline{\boldsymbol{y}_{i,j}}) \sim p(\boldsymbol{x})p(\boldsymbol{y})$. Another interpretation to CPC is that it performs m-way classification where the ground truth label is assigned to the positive pair. The representation learner q_ϕ then aims to maximize the CPC objective, or to minimize

the following objective:

$$-L_{\mathrm{C}}(q_\phi) := \min_g -L_{\mathrm{CPC}}(g; q_\phi), \tag{B.13}$$

Different specific implementations, such as MoCo [HFW+19, CFGH20, CXH21] and SimCLR [CKNH20] can all be treated as specific implementations of this objective function. In this paper, we considered using MoCo-v2 [CKNH20] as our implementation for L_{C} objective; in principle, other implementations to CPC can also be integrated into D2C as well.

Training D2C

In Algorithm 7, we describe a high-level procedure that trains the D2C model; we note that this procedure does not have any adversarial components. On the high-level, this is the integration of three objectives: the reconstruction objective via the autoencoder, the diffusion objective over the latent space, and the contrastive objective over the latent space. In principle, the **[reconstruction]**, **[constrastive]**, and **[diffusion]** components can be optimized jointly or separately; we observe that normalizing the latent $\mathbf{z}^{(1)}$ with a global mean and standard deviation before applying the diffusion objective helps learning the diffusion model with a fixed α series.

Algorithm 7 Training D2C

Input: Data distribution p_{data}.

while training **do**

[Draw samples with data augmentation]

Draw m samples $\mathbf{x}_{0:m-1} \sim p_{\text{data}}(\mathbf{x})$.

Draw $(m+1)$ data augmentations $\text{aug}_0, \dots \text{aug}_{m-1}$ and $\overline{\text{aug}}_0$.

for $i \leftarrow 0$ to $m-1$ **do**

Draw $\mathbf{z}_i^{(1)} \sim q_\phi(\mathbf{z}^{(1)}|\text{aug}_i(\mathbf{x}))$.

end for

Draw $\overline{\mathbf{z}}_0^{(1)} \sim q_\phi(\mathbf{z}^{(1)}|\overline{\text{aug}}_0(\mathbf{x}))$.

[Reconstruction]

Reconstruct $\mathbf{x}_0 \sim p_\theta(\mathbf{x}|\mathbf{z}_0^{(1)})$

Minimize $L_{\text{recon}} = -\log p_\theta(\mathbf{x}|\mathbf{z}_0^{(1)})$ over θ and ϕ with gradient descent.

[Contrastive]

Define a classification task: assign label 1 to $(\mathbf{z}_0^{(1)}, \overline{\mathbf{z}}_0^{(1)})$ and label 0 to $(\mathbf{z}_0^{(1)}, \mathbf{z}_i^{(1)})$ for $i \neq 0$.

Define $L_{\text{CPC}}(g; q_\phi)$ as the loss to minimize for the above task, with g as the classifier.

Define $\hat{g}$ as a minimizer to the classifier objective $L_{\text{CPC}}(g; q_\phi)$.

Minimize $L_{\text{CPC}}(\hat{g}; q_\phi)$ over ϕ with gradient descent.

[Diffusion]

Sample $\epsilon \sim \mathcal{N}(0, I)$, $t \sim \text{Uniform}(1, \dots, T)$.

Define $\boldsymbol{z}_0^{(\alpha_t)} = \sqrt{\alpha_t}\boldsymbol{z}_0^{(0)} + \sqrt{1-\alpha_t}\epsilon$.

Minimize $\|\epsilon - \epsilon_\theta(\boldsymbol{z}_0^{(\alpha_t)}; \alpha_t)\|_2^2$ over θ with gradient descent.

end while

B.6.2 Experimental details

Architecture details and hyperparameters used for training

We modify the NVAE [VK20] architecture by removing the "Combiner Cells" in both encoder and decoder. For the diffusion model, we use the same architecture with different number of channel multiplications, as used in [HJA20, SME20]. For Contrastive learning, we use the MoCo-v2 [CFGH20] algorithm with augmentations such as *RandomResizedCrop, ColorJitter, RandomGrayscale, RandomHorizontalFlip.*

Table B.5: Hyperparameters across different datasets

Hyperparameter	**CIFAR-10** 32x32	**CIFAR-100** 32x32	**CelebA-64** 64x64	**fMoW** 64x64	**CelebA-HQ-256** 256x256	**FFHQ-256** 256x256
# of epochs	1000	1000	300	300	200	100
batch size per GPU	32	32	16	16	3	3
# initial channels in enc,	128	128	64	64	24	24
spatial dims of z	16*16	16*16	32*32	32*32	64*64	64*64
# channel in z	8	8	5	5	8	8
MoCo-v2 queue size	65536	65536	65536	65536	15000	15000
Diffusion feature map res.	16,8,4,2	16,8,4,2	32,16,8,4,1	32,16,8,4,1	64,32,16,8,2	64,32,16,8,2
λ^{-1}	17500	17500	17500	17500	17500	17500
learning rate	0.001	0.001	0.001	0.001	0.001	0.001
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW
# GPUs	8	8	4	4	8	8
GPU Type	16 GB V100	16 GB V100	12 GB Titan X	12 GB Titan X	16 GB V100	16 GB V100
Total training time (h)	24	24	120	120	96	96

Additional details about the hyperparameters used are provided in Table B.5.

Additional details for conditional generation

For $r_\psi(\mathbf{c}|\mathbf{z}^{(1)})$ we consider training a linear model over the latent space, which has the advantage of being computationally efficient. For conditional generation on labels, we reject samples if their classifier return are lower than a certain threshold (we used 0.5 for all our experiments). For conditional image manipulation, we consider the same step size η for each attribute: $\eta = 10$ for *red lipstick* and $\eta = 15$ for *blond*. We note that these values are not necessarily the optimal ones, as the intensity of the change can grow with a choice of larger η values.

Amazon Mechanical Turk procedure

The mechanical turk evaluation is done for different attributes to find out how evaluators evaluate the different approaches. The evaluators are asked to compare a pair of images, and find the best image, which retains the identity as well as contains the desired attribute. Figure B.10 a) shows the instructions that was given to the evaluators before starting the test and Figure B.10 b) contains the UI shown to the evaluators when doing comparison. Each evaluation task contains 10 pairwise comparisons, and we perform 15 such evaluation tasks for each attribute. The reward per task is kept as 0.25$. Since each task takes around 2.5 mins, so the hourly wage comes to be 6$ per hour.

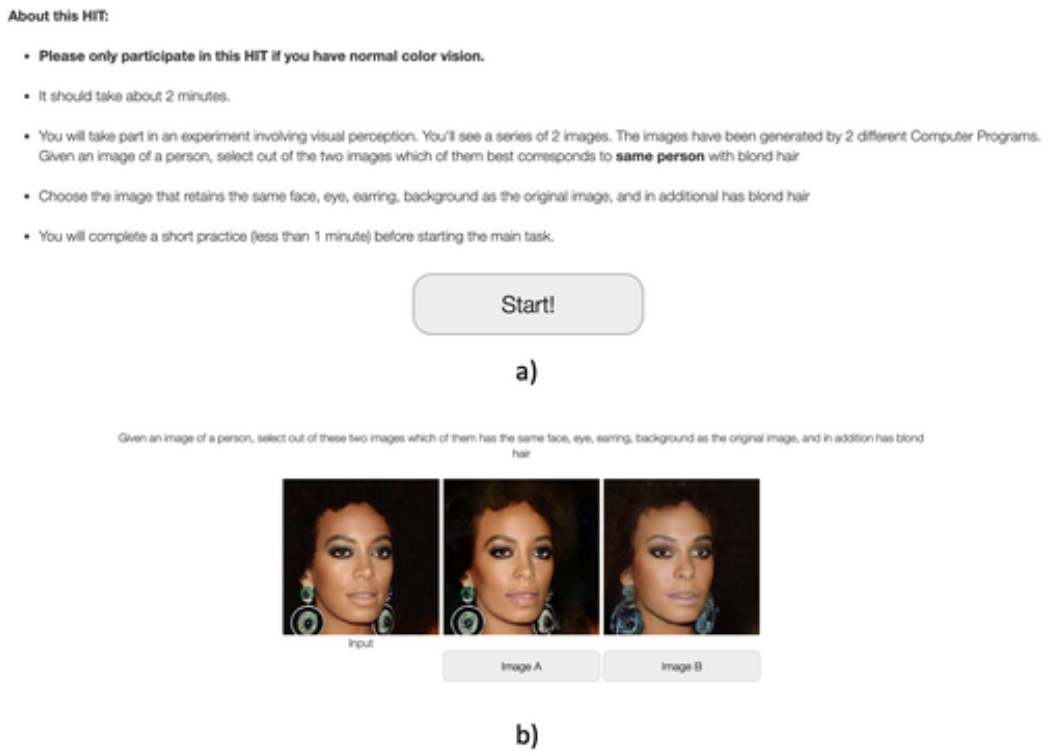


Figure B.10: a) Instructions shown to human evaluators for Amazon Mechanical Turk for blond hair before starting the evaluation and b) UI shown to the evaluators when doing comparison.

Table B.6: CIFAR-10 image generation results.

Method	FID
NVAE [VK20]	51.71
NCP-VAE [ASKV20]	24.08
EBM [DM19]	40.58
StyleGAN2 [KLA+20]	3.26
DDPM [HJA20]	3.17
DDIM [SME20]	4.04
D2C	10.15

B.6.3 Additional Results

We list results for unconditional CIFAR-10 image generation for various types of generative models in Table B.6. While our results are slightly worse than state-of-the-art diffusion models, we note that our D2C models are trained with relatively fewer resources that some of the baselines; for example, our D2C models is trained on 8 GPUs for 24 hours, whereas NVAE is trained on 8 GPUs for 100 hours and DDPM is trained on v3-8 TPUs for 24 hours. We also note that these comparisons are not necessarily fair in terms of the architecture and compute used to produce the samples.

We list additional image generation results in Figure B.11 (unconditional), Figures B.12, B.13, B.14, and B.15 (conditional on manipulation constraints), and Figures B.16, B.17, B.18, and B.19 (conditional on labels)[2].

B.7 Chapter 8

B.7.1 Estimating Effective Sample Size

Assume a target distribution $p(x)$, and a Markov chain Monte Carlo (MCMC) sampler that produces a set of N correlated samples $\{x_i\}_1^N$ from some distribution $q(\{x_i\}_1^N)$ such that $q(x_i) = p(x_i)$. Suppose we are estimating the mean of $p(x)$ through sampling; we assume that increasing the number of samples will reduce the variance of that estimate.

Let $V = \text{Var}_q[\sum_{i=1}^N x_i/N]$ be the variance of the mean estimate through the MCMC samples. The effective sample size (ESS) of $\{x_i\}_1^N$, which we denote as $M = ESS(\{x_i\}_1^N)$, is the number of independent samples from $p(x)$ needed in order to achieve the same variance, i.e.

[2]We will list more results online after publication.

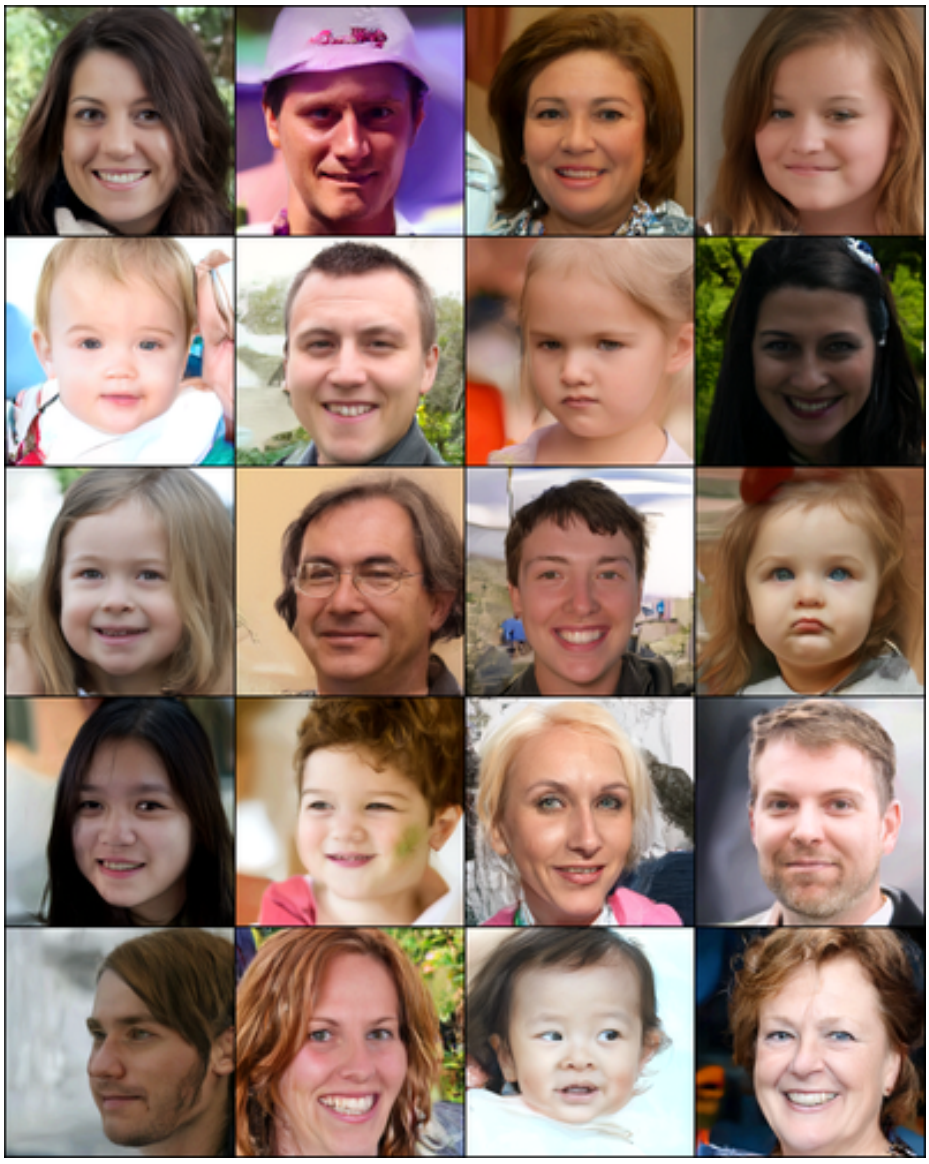


Figure B.11: Additional image samples for the FFHQ-256 dataset.

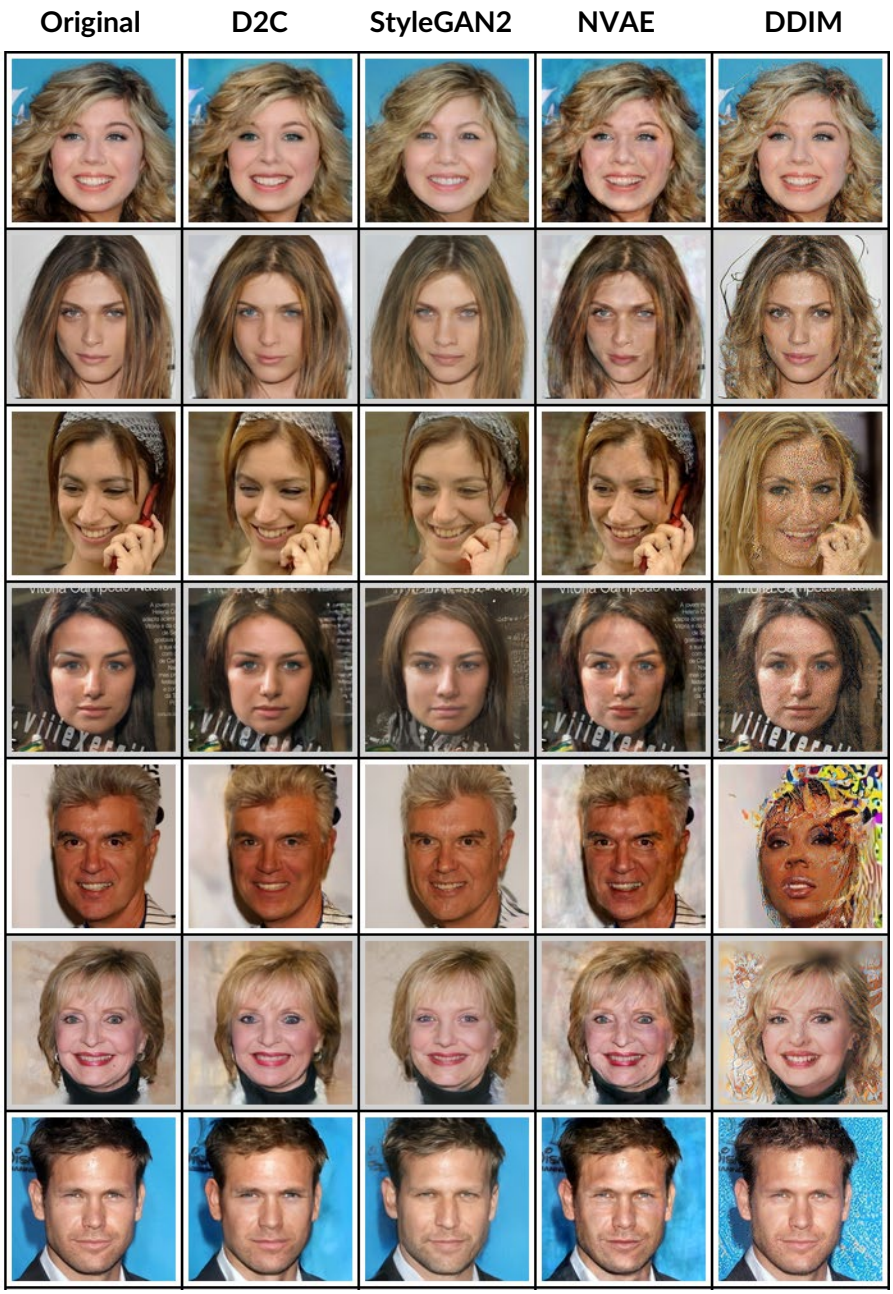


Figure B.12: Image manipulation results for *blond hair*.

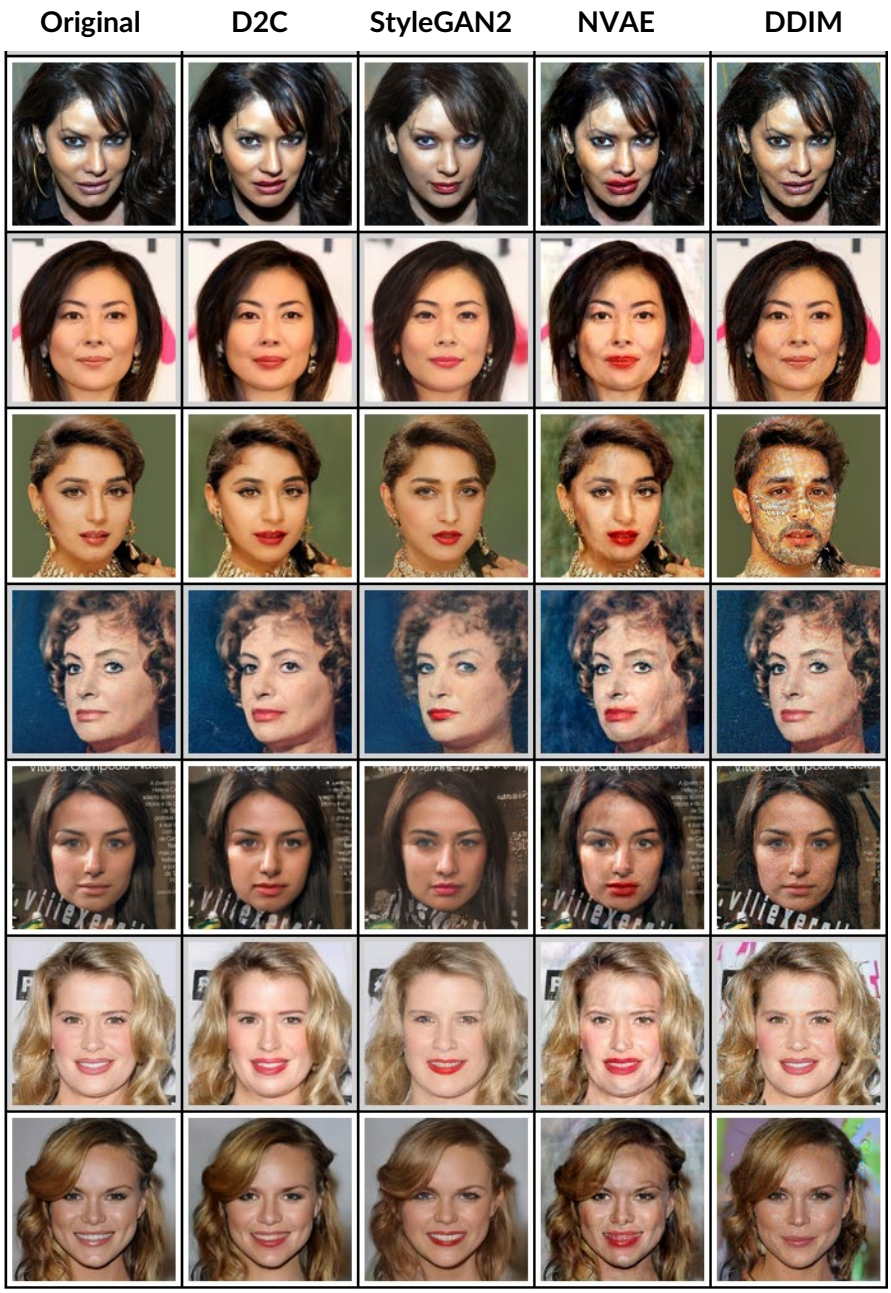


Figure B.13: Image manipulation results for *red lipstick*.

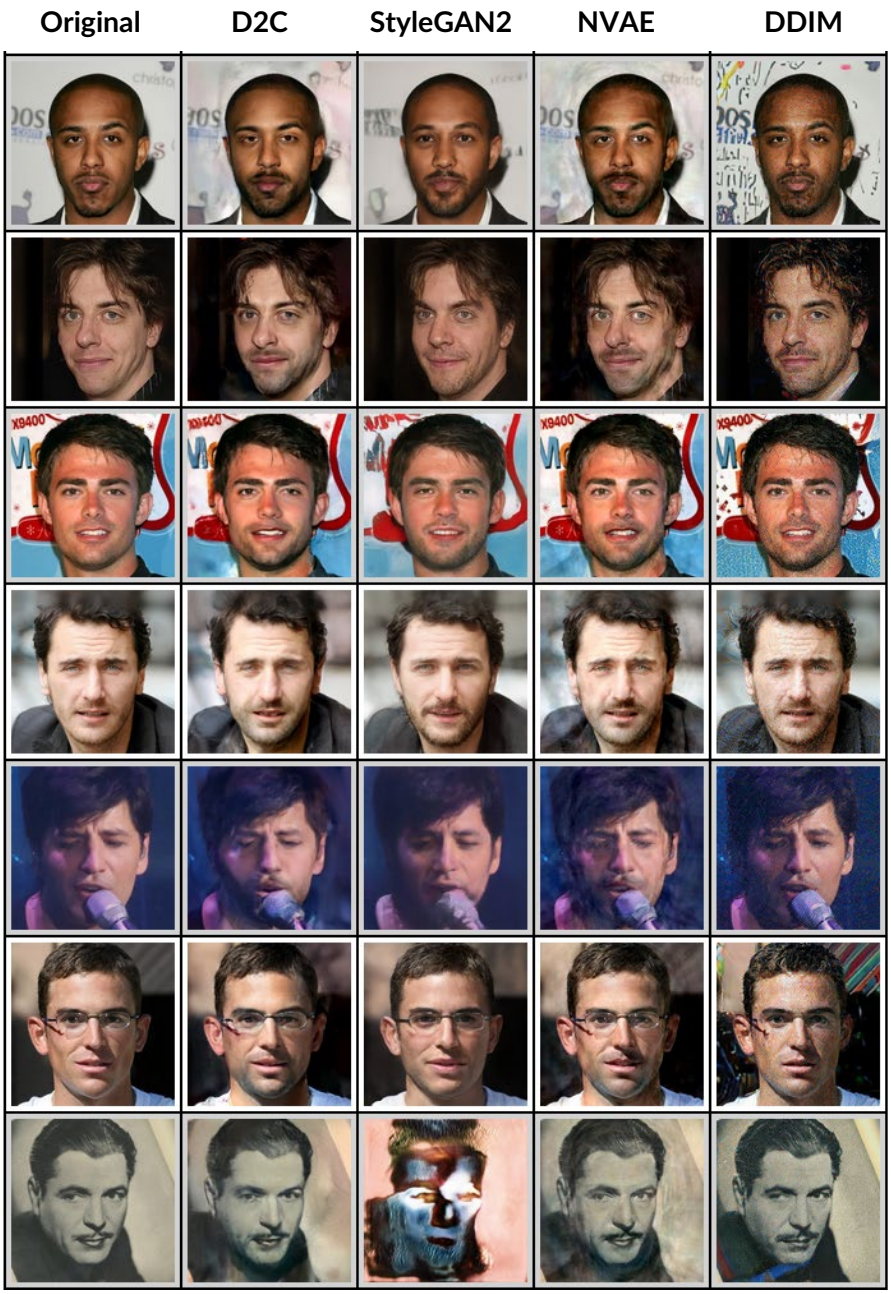
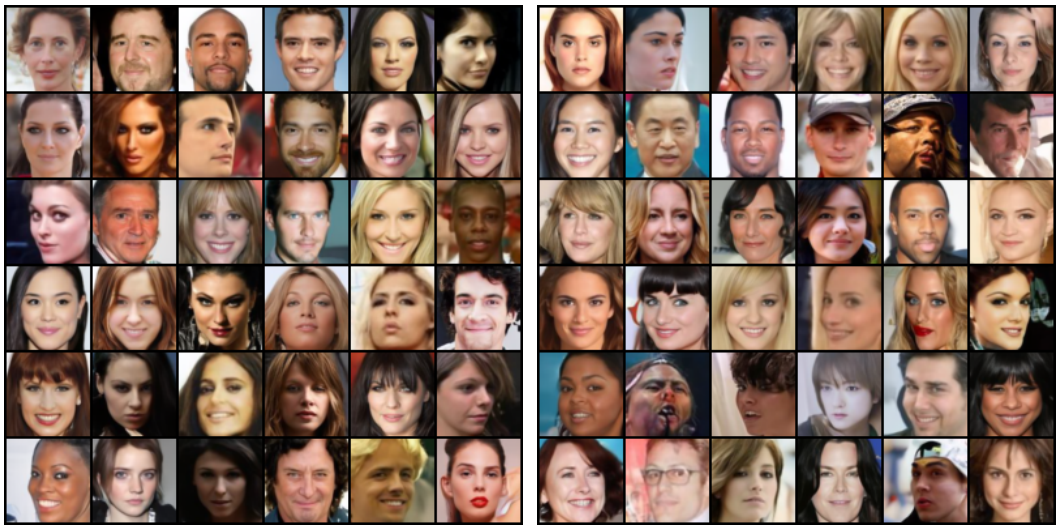


Figure B.14: Image manipulation results for *beard*.

Figure B.15: Image manipulation results for *gender*.

(a) Conditioned on *non-blond* label (b) Conditioned on *blond* label

Figure B.16: Conditional generation with D2C by learning from 100 labeled examples.



(a) Conditioned on *non-blond* label (b) Conditioned on *blond* label

Figure B.17: Conditional generation with DDIM by learning from 100 labeled examples.

$\text{Var}_p[\sum_{j=1}^{M} x_j/M] = V$. A practical algorithm to compute the ESS given $\{x_i\}_1^N$ is provided by:

$$ESS(\{x_i\}_1^N) = \frac{N}{1 + 2\sum_{s=1}^{N-1}(1 - \frac{s}{N})\rho_s} \tag{B.14}$$

where ρ_s denotes the autocorrelation under q of x at lag s. We compute the following empirical

(a) Conditioned on *female* label (b) Conditioned on *male* label

Figure B.18: Conditional generation with D2C by learning from 100 labeled examples.

(a) Conditioned on *female* label (b) Conditioned on *male* label

Figure B.19: Conditional generation with DDIM by learning from 100 labeled examples.

estimate $\hat{\rho}_s$ for ρ_s:

$$\hat{\rho}_s = \frac{1}{\hat{\sigma}^2(N-s)} \sum_{n=s+1}^{N} (x_n - \hat{\mu})(x_{n-s} - \hat{\mu}) \tag{B.15}$$

where $\hat{\mu}$ and $\hat{\sigma}$ are the empirical mean and variance obtained by an independent sampler.

Due to the noise in large lags s, we adopt the approach of [HG14] where we truncate the sum over the autocorrelations when the autocorrelation goes below 0.05.

B.7.2 Justifications for Objective in Equation 9.3

We consider two necessary conditions for p_d to be the stationary distribution of the Markov chain, which can be translated into a new algorithm with better optimization properties, described in Equation 9.3.

Proposition 8. *Consider a sequence of ergodic Markov chains over state space $\mathcal{S}$. Define π_n as the stationary distribution for the n-th Markov chain, and π_n^t as the probability distribution at time step t for the n-th Markov chain. If the following two conditions hold:*

1. *$\exists b > 0$ such that the sequence $\{\pi_n^b\}_{n=1}^{\infty}$ converges to p_d in total variation;*
2. *$\exists \epsilon > 0$, $\rho < 1$ such that $\exists M > 0, \forall n > M$, if $\|\pi_n^t - p_d\|_{TV} < \epsilon$, then $\|\pi_n^{t+1} - p_d\|_{TV} < \rho\|\pi_n^t - p_d\|_{TV}$;*

then the sequence of stationary distributions $\{\pi_n\}_{n=1}^{\infty}$ converges to p_d in total variation.

Proof. The goal is to prove that $\forall \delta > 0, \exists K > 0, T > 0$, such that $\forall n > N, t > T$, $\|\pi_n^t - p_d\|_{\text{TV}} < \delta$.

According to the first assumption, $\exists N > 0$, such that $\forall n > N$, $\|\pi_n^b - p_d\|_{\text{TV}} < \epsilon$.

Therefore, $\forall n > K = \max(N, M), \forall \delta > 0, \exists T = b + \max(0, \lceil \log_\rho \delta - \log_\rho \epsilon \rceil) + 1$, such that $\forall t > T$,

$$\begin{aligned} &\|\pi_n^t - p_d\|_{\text{TV}} \\ =&\|\pi_n^b - p_d\|_{\text{TV}} \prod_{i=b}^{t-1} \frac{\|\pi_n^{i+1} - p_d\|_{\text{TV}}}{\|\pi_n^i - p_d\|_{\text{TV}}} \\ <&\epsilon\rho^{t-b} < \epsilon\rho^{T-b} < \epsilon \cdot \frac{\delta}{\epsilon} = \delta \end{aligned} \tag{B.16}$$

The first inequality uses the fact that $\|\pi_n^b - p_d\|_{\text{TV}} < \epsilon$ (from Assumption 1), and $\|\pi_n^{t+1} - p_d\|_{\text{TV}} / \|\pi_n^t - p_d\|_{\text{TV}} < \rho$ (from Assumption 2). The second inequality is true because $\rho < 1$ by Assumption 2. The third inequality uses the fact that $T - b > \lceil \log_\rho \delta - \log_\rho \epsilon \rceil$ (from definition of T), so $\rho^{T-b} < \delta/\epsilon$. Hence the sequence $\{\pi_n\}_{n=1}^{\infty}$ converges to p_d in total variation. □

Moreover, convergence in total variation distance is equivalent to convergence in Jensen-Shannon (JS) divergence[ACB17], which is what GANs attempt to minimize [GPAM+14]. This

motivates the use of GANs to achieve the two conditions in Proposition 8. This suggests a new optimization criterion, where we look for a θ that satisfies both conditions in Proposition 8, which translates to Equation 9.3.

B.7.3 Details on the Pairwise Discriminator

Similar to the settings in MGAN objective, we consider two chains to obtain samples:

- Starting from a data point x, sample z_1 in B steps.
- Starting from some noise z, sample z_2 in B steps; and from z_2 sample z_3 in M steps.

For the "generated" (fake) data, we use two type of pairs (x, z_1), and (z_2, z_3). This is illustrated in Figure B.20. We assume equal weights between the two types of pairs.

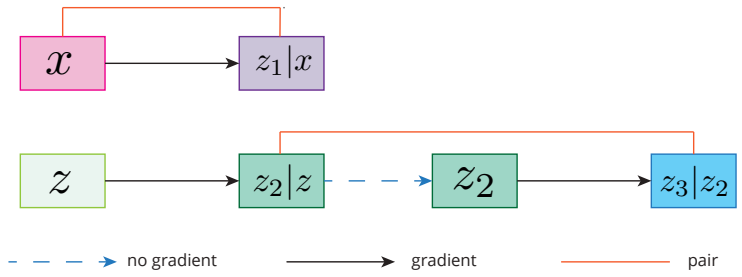


Figure B.20: Illustration of the generative process for the pairwise discriminator. We block the gradient for z_2 to further parallelize the process and improve training speed.

B.7.4 Additional Experimental Details

Architectures for Generative Model for Images

Code is available at https://github.com/ermongroup/markov-chain-gan.

Let 'fc n, (activation)' denote a fully connected layer with n neurons. Let 'conv2d n, k, s, (activation)' denote a convolutional layer with n filters of size k and stride s. Let 'deconv2d n, k, s, (activation)' denote a transposed convolutional layer with n filters of size k and stride s.

We use the following model to generate Figure 9.1 (MNIST).

encoder	decoder	discriminator
fc 600, lrelu	fc 600, lrelu	conv2d 64, 4×4, 2×2, relu
fc 100, linear	fc 784, sigmoid	conv2d 128, 4×4, 2×2, lrelu
		fc 600, lrelu
		fc 1, linear

We use the following model to generate Figure 9.3 (CelebA, top)

encoder	decoder	discriminator
conv2d 64, 4×4, 2×2, lrelu	fc $16 \times 16 \times 64$, lrelu	conv2d 64, 4×4, 2×2, relu
fc 200, linear	deconv2d 3, 4×4, 2×2, tanh	conv2d 128, 4×4, 2×2, lrelu
		conv2d 256, 4×4, 2×2, lrelu
		fc 1, linear

For the bottom figure in Figure 9.3, we add a residual connection such that the input to the second layer of the decoder is the sum of the outputs from the first layers of the decoder and encoder (both have shape $16 \times 16 \times 64$); we add a highway connection from input image to the output of the decoder:

$$\bar{x} = \alpha x + (1 - \alpha)\hat{x}$$

where $\bar{x}$ is the output of the function, $\hat{x}$ is the output of the decoder, and α is an additional transposed convolutional output layer with sigmoid activation that has the same dimension as $\hat{x}$.

We use the following model to generate Figure 9.5 (CelebA, pairwise):

encoder	decoder	discriminator
conv2d 64, 4×4, 2×2, lrelu	fc 1024, relu	conv2d 64, 4×4, 2×2, relu
conv2d 64, 4×4, 2×2	fc $8 \times 8 \times 128$, relu	conv2d 128, 4×4, 2×2, lrelu
fc 1024, lrelu	deconv2d 64, 4×4, 2×2, relu	conv2d 256, 4×4, 2×2, lrelu
fc 200 linear	deconv2d 3, 4×4, 2×2, tanh	fc 1, linear

For the pairwise discriminator, we double the number of filters in each convolutional layer. According to [GAA+17], we only use batch normalization in the generator for all experiments.

Analytic Forms of Energy Functions

Let $f(x|\mu, \sigma)$ denote the log pdf of $\mathcal{N}(\mu, \sigma^2)$.

The analytic form of $U(x)$ for *ring* is:

$$U(x) = \frac{(\sqrt{x_1^2 + x_2^2} - 2)^2}{0.32} \tag{B.17}$$

The analytic form of $U(x)$ for *mog2* is:

$$U(x) = f(x|\mu_1, \sigma_1) + f(x|\mu_2, \sigma_2) - \log 2 \tag{B.18}$$

where $\mu_1 = [5, 0]$, $\mu_2 = [-5, 0]$, $\sigma_1 = \sigma_2 = [0.5, 0.5]$.

The analytic form of $U(x)$ for *mog6* is:

$$U(x) = \sum_{i=1}^{6} f(x|\mu_i, \sigma_i) - \log 6 \tag{B.19}$$

where $\mu_i = [\sin \frac{i\pi}{3}, \cos \frac{i\pi}{3}]$ and $\sigma_i = [0.5, 0.5]$.

The analytic form of $U(x)$ for *ring5* is:

$$U(x) = \min(u_1, u_2, u_3, u_4, u_5) \tag{B.20}$$

where $u_i = (\sqrt{x_1^2 + x_2^2} - i)^2 / 0.04$.

Benchmarking Running Time

Since the runtime results depends on the type of machine, language, and low-level optimizations, we try to make a fair comparison between HMC and A-NICE-MC on TensorFlow [AAB+16].

Our code is written and executed in TensorFlow 1.0. Due to the optimization of the computation graphs in TensorFlow, the wall-clock time does not seem to be exactly linear in some cases, even when we force the program to use only 1 thread on the CPU. The wall-clock time is affected by 2 aspects, batch size and number of steps. We find that the wall-clock time is relatively linear with respect to the number of steps, and not exactly linear with respect to the batch size.

Given a fixed number of steps, the wall-clock time is constant when the batch size is lower than a threshold, and then increases approximately linearly. To perform speed benchmarking on the methods, we select the batch size to be the value around the threshold, in order to prevent significant under-estimates of the efficiency.

We found that the graph is much more optimized if the batch size is determined before execution. Therefore, we perform all the benchmarks on the optimized graph where we specify a batch size prior to running the graph. For the energy functions, we use a batch size of 2000; for Bayesian logistic regression we use a batch size of 64.

Hyperparameters for the Energy Function Experiments

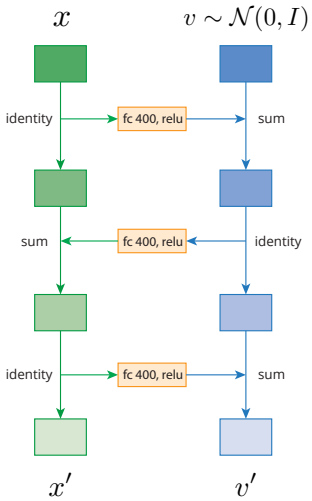
For all the experiments, we use same hyperparameters for both A-NICE-MC and HMC. We sample $x_0 \sim \mathcal{N}(0, I)$ and run the chain for 1000 burn-in steps and evaluate the samples from the next 1000 steps. For HMC we use 40 leapfrog steps and a step size of 0.1. For A-NICE-MC we consider $f_\theta(x, v)$ with three coupling layers, which updates v, x and v respectively. The motivation behind this particular architecture is to ensure that both x and v could affect the updates to x' and v'. In each coupling layer, we select the function $m(\cdot)$ to be a one-layer NN with 400 neurons. The discriminator is a three layer MLP with 400 neurons each. Similar to the settings in Section 9.3.1, we use the gradient penalty method in [GAA+17] to train our model. For bootstrapping, we first collect samples by running the NICE proposal over the untrained f_θ, and for every 500 iterations we replace half of the samples with samples from the latest trained model. All the models are trained with AdaM [KB14] for 20000 iterations with $B = 4$, $M = 2$, batch size of 32 and learning rate of 10^{-4}.

Hyperparameters for the Bayesian Logistic Regression Experiments

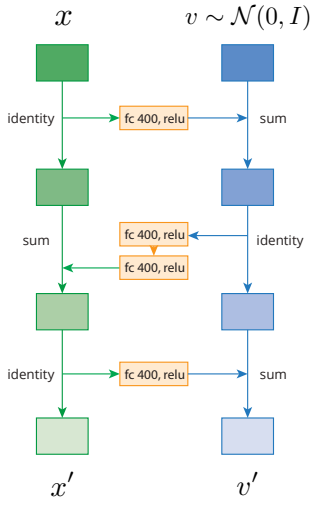
For HMC we tuned the step size parameter to achieve the best ESS possible on each dataset, which is 0.005 for *german*, 0.01 for *heart* and 0.0115 for *australian* (HMC performance on *australian* is extremely sensitive to the step size). For A-NICE-MC we consider $f(x, v)$ with three coupling layers, which updates v, x and v respectively; we set v to have 50 dimensions in all the experiments. $m(\cdot)$ is a one-layer NN with 400 neurons for the top and bottom coupling layer, and a two-layer NN with 400 neurons each for the middle layer. The discriminator is a three layer MLP with 800 neurons each. We use the same training and bootstrapping strategy as in Appendix B.7.4. All the models are trained with AdaM for 20000 iterations with $B = 16$, $M = 2$, batch size of 32 and learning rate of 5×10^{-4}.

Architecture Details

The following figure illustrates the architecture details of $f_\theta(x, v)$ for A-NICE-MC experiments. We do not use batch normalization (or other normalization techniques), since it slows the execution of the network and does not provide much ESS improvement.



(a) NICE architecture for energy functions.



(b) NICE architecture for Bayesian logistic regression.

B.8 Chapter 9

B.8.1 MAGAIL Algorithm

We include the MAGAIL algorithm as follows:

Algorithm 8 Multi-Agent GAIL (MAGAIL)

Input: Initial parameters of policies, discriminators and value (baseline) estimators $\theta_0, \omega_0, \phi_0$; expert trajectories $\mathcal{D} = \{(s_j, a_j)\}_{j=0}^{M}$; batch size B; Markov game as a black box $(N, \mathcal{S}, \mathcal{A}, \eta, T, r, \mathbf{o}, \gamma)$.

Output: Learned policies π_θ and reward functions D_ω.

for $u = 0, 1, 2, \ldots$ **do**

Obtain trajectories of size B from π by the process: $s_0 \sim \eta(s), a_t \sim \pi_{\theta_u}(a_t|s_t), s_{t+1} \sim P(s_t|a_t)$.

Sample state-action pairs from $\mathcal{D}$ with batch size B.

Denote state-action pairs from π and $\mathcal{D}$ as χ and χ_E.

for $i = 1, \ldots, n$ **do**

Update ω_i to increase the objective

$$\mathbb{E}_\chi[\log D_{\omega_i}(s, a_i)] + \mathbb{E}_{\chi_E}[\log(1 - D_{\omega_i}(s, a_i))]$$

end for

for $i = 1, \ldots, n$ **do**

Compute value estimate $V^\star$ and advantage estimate A_i for $(s, a) \in \chi$.

Update ϕ_i to decrease the objective

$$\mathbb{E}_\chi[(V_\phi(s, a_{-i}) - V^\star(s, a_{-i}))^2]$$

Update θ_i by policy gradient with small step sizes:

$$\mathbb{E}_\chi[\nabla_{\theta_i} \pi_{\theta_i}(a_i|o_i) A_i(s, a)]$$

end for

end for

B.8.2 Experiment Details

Hyperparameters

For the particle environment, we use two layer MLPs with 128 cells in each layer, for the policy generator network, value network and the discriminator. We use a batch size of 1000. The policy is trained using K-FAC optimizer [MG15] with learning rate of 0.1. All other parameters for K-FAC

Table B.7: Performance in cooperative navigation.

# Expert Episodes	100	200	300	400
Expert	-13.50 ± 6.3			
Random	-128.13 ± 32.1			
Behavior Cloning	-56.82 ± 18.9	-43.10 ± 16.0	-35.66 ± 15.2	-25.83 ± 12.7
Centralized	**-46.66** ± 20.8	**-23.10** ± 12.4	**-21.53** ± 12.9	**-15.30** ± 7.0
Decentralized	-50.00 ± 18.6	-25.61 ± 12.3	-24.10 ± 13.3	-15.55 ± 6.5
GAIL	-55.01 ± 17.7	-39.21 ± 16.5	-29.89 ± 13.5	-18.76 ± 12.1

Table B.8: Performance in cooperative communication.

# Expert Episodes	100	200	300	400
Expert	-6.22 ± 4.5			
Random	-62.49 ± 28.7			
Behavior Cloning	-21.25 ± 10.6	-13.25 ± 7.4	-11.37 ± 5.9	-10.00 ± 5.36
Centralized	**-15.65** ± 10.0	**-7.11** ± 4.8	**-7.11** ± 4.8	**-7.09** ± 4.8
Decentralized	-18.68 ± 10.4	-8.06 ± 5.3	-8.16 ± 5.5	-7.34 ± 4.9
GAIL	-20.28 ± 10.1	-11.06 ± 7.8	-10.51 ± 6.6	-9.44 ± 5.7

optimizer are the same in [WMG+17].

For the cooperative control task, we use two layer MLPs with 64 cells in each layer for all the networks. We use a batch size of 2048, and learning rate of 0.03. We obtain expert trajectories by training the expert with MACK and sampling demonstrations from the same environment. Hence, the expert's demonstrations are imperfect (or even flawed) in the environment that we test on.

Detailed Results

We use the particle environment introduced in [LWT+17] and the multi-agent control environment [KGE17] for experiments. We list the exact performance in Tables B.7, B.8 for cooperative tasks, and Table B.9 and competitive tasks. The means and standard deviations are computed over 100 episodes. The policies in the cooperative tasks are trained with varying number of expert demonstrations. The policies in the competitive tasks are trained with on a dataset with 100 expert trajectories.

The environment for each episode is drastically different (e.g. location of landmarks are randomly sampled), which leads to the seemingly high standrad deviation across episodes.

Table B.9: Performance in competitive tasks.

Task	Agent Policy	Adversary Policy	Agent Reward
Predator-Prey	Behavior Cloning	Behavior Cloning	-93.20 ± 63.7
		GAIL	-93.71 ± 64.2
		Centralized	-93.75 ± 61.9
		Decentralized	-95.22 ± 49.7
		Zero-Sum	**-95.48** ± 50.4
	GAIL	Behavior Cloning	-90.55 ± 63.7
	Centralized		-91.36 ± 58.7
	Decentralized		**-85.00** ± 42.3
	Zero-Sum		-89.4 ± 48.2
Keep-Away	Behavior Cloning	Behavior Cloning	24.22 ± 21.1
		GAIL	24.04 ± 18.2
		Centralized	23.28 ± 20.6
		Decentralized	23.56 ± 19.9
		Zero-Sum	**23.19** ± 19.9
	GAIL	Behavior Cloning	26.22 ± 19.1
	Centralized		26.61 ± 20.0
	Decentralized		**28.73** ± 18.3
	Zero-Sum		27.80 ± 19.2

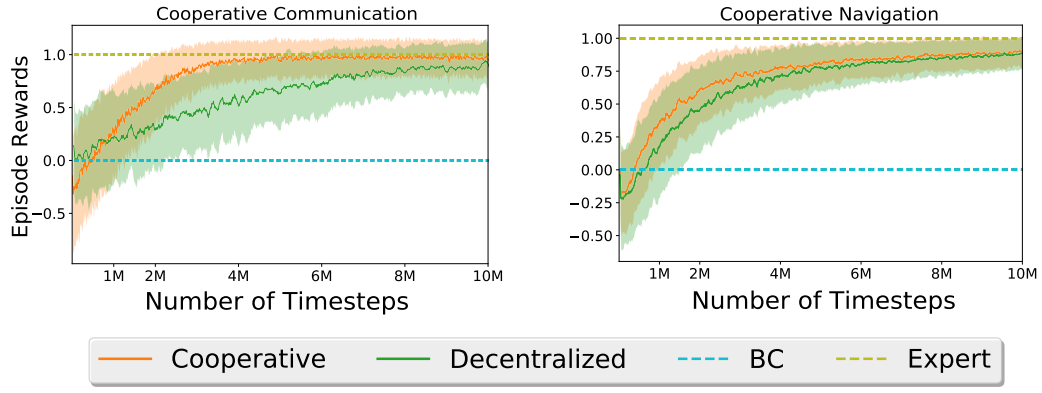


Figure B.22: Sample complexity of multi-agent GAIL methods under cooperative tasks. Performance of experts is normalized to one, and performance of behavior cloning is normalized to zero. The standard deviation is computed with respect to episodes, and is noisy due to randomness in the environment.

Video Demonstrations

We show certain trajectories generated by our methods. The vidoes are here: videos[3].

For the particle case:

Navigation-BC-Agents.gif Agents trained by behavior cloning in the navigation task.

Navigation-GAIL-Agents.gif Agents trained by proposed framework in the navigation task.

Predator-Prey-BC-Agent-BC-Adversary.gif Agent (green) trained by behavior cloning play against adversaries (red) trained by behavior cloning.

Predator-Prey-GAIL-Agent-BC-Adversary.gif Agent (green) trained by proposed framework play against adversaries (red) trained by behavior cloning.

For the cooperative control case:

Multi-Walker-Expert.mp4 Expert demonstrations in the "easy" environment.

Multi-Walker-GAIL.mp4 Centralized GAIL trained on the "hard" environment.

Multi-Walker-BC.mp4 BC trained on the "hard" environment.

Interestingly, the failure modes for the agents in "hard" environment is mostly having the plank fall off or bounce off, since by decreasing the weight of the plank will decrease its friction force and increase its acceleration.

[3] https://drive.google.com/open?id=1Oz4ezMaKiIsPUKtCEOb6YoHJ9jLk6zbj

Appendix C

Code and Data

The following links provide open-source implementations and custom datasets (if applicable) for the various chapters presented in this dissertation.

- Chapter 2
  `https://github.com/ermongroup/smile-mi-estimator`
- Chapter 3
  `https://github.com/jiamings/ml-cpc`
- Chapter 4
  `https://github.com/jiamings/lag-fairness`
- Chapter 5
  `https://github.com/ermongroup/f-wgan`
- Chapter 6
  `https://github.com/ermongroup/NDA`
- Chapter 7
  `https://github.com/jiamings/d2c`
- Chapter 8
  `https://github.com/ermongroup/a-nice-mc`
- Chapter 9
  `https://github.com/ermongroup/multiagent-gail`
  `https://github.com/ermongroup/MA-AIRL`

Bibliography

[AAB+16] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[AABS19] Jyoti Aneja, Harsh Agrawal, Dhruv Batra, and Alexander Schwing. Sequential latent spaces for modeling the intention during diverse image captioning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4261–4270, 2019.

[AAH19] Muhammad Asim, Ali Ahmed, and Paul Hand. Invertible generative models for inverse problems: mitigating representation error and dataset bias. *arXiv preprint arXiv:1905.11672*, May 2019.

[AC16] Dario Amodei and Jack Clark. Faulty reward functions in the wild, 2016.

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*, January 2017.

[AFDM16] Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, December 2016.

[AHD+19] Sungsoo Ahn, Shell Xu Hu, Andreas Damianou, Neil D Lawrence, and Zhenwen Dai. Variational information distillation for knowledge transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9163–9171, 2019.

[Ama98] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, February 1998.

[AN04] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[AN07] Arthur Asuncion and David Newman. UCI machine learning repository, 2007.

[AOD+18] Samaneh Azadi, Catherine Olsson, Trevor Darrell, Ian Goodfellow, and Augustus Odena. Discriminator rejection sampling. *arXiv preprint arXiv:1810.06758*, October 2018.

[AOS+16] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, June 2016.

[APF+17] Alexander A Alemi, Ben Poole, Ian Fischer, Joshua V Dillon, Rif A Saurous, and Kevin Murphy. Fixing a broken ELBO. *arXiv preprint arXiv:1711.00464*, November 2017.

[ARV19] Yuki M Asano, Christian Rupprecht, and Andrea Vedaldi. A critical analysis of self-supervision, or what we can learn from a single image. *arXiv preprint arXiv:1904.13132*, 2019.

[AS66] Syed Mumtaz Ali and Samuel D Silvey. A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society: Series B (Methodological)*, 28(1):131–142, 1966.

[ASE17] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, November 2017.

[ASKV20] Jyoti Aneja, Alexander Schwing, Jan Kautz, and Arash Vahdat. NCP-VAE: Variational autoencoders with noise contrastive priors. *arXiv preprint arXiv:2010.02917*, October 2020.

[Aum74] Robert J Aumann. Subjectivity and correlation in randomized strategies. *Journal of mathematical Economics*, 1(1):67–96, 1974.

[Aum87] Robert J Aumann. Correlated equilibrium as an expression of bayesian rationality. *Econometrica: Journal of the Econometric Society*, pages 1–18, 1987.

[BA03] David Barber and Felix V Agakov. The IM algorithm: a variational approach to information maximization. In *Advances in neural information processing systems*, page None. researchgate.net, 2003.

[Bag15] J Andrew Bagnell. An invitation to imitation. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, 2015.

[Bar89] Horace B Barlow. Unsupervised learning. *Neural computation*, 1(3):295–311, 1989.

[BB14] Michael Bloem and Nicholas Bambos. Infinite time horizon maximum causal entropy inverse reinforcement learning. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 4911–4916. IEEE, 2014.

[BBR+18] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R Devon Hjelm. MINE: Mutual information neural estimation. *arXiv preprint arXiv:1801.04062*, January 2018.

[BD14] Kenneth Bogert and Prashant Doshi. Multi-robot inverse reinforcement learning under occlusion with interactions. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 173–180. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[BDS18] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, September 2018.

[BDX04] Stephen Boyd, Persi Diaconis, and Lin Xiao. Fastest mixing markov chain on a graph. *SIAM review*, 46(4):667–689, 2004.

[BG98] Stephen P Brooks and Andrew Gelman. General methods for monitoring convergence of iterative simulations. *Journal of computational and graphical statistics*, 7(4):434–455, 1998.

[BGR+06] Karsten M Borgwardt, Arthur Gretton, Malte J Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J Smola. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–57, July 2006.

[BHV17] Floria Bordes, Sina Honari, and Pascal Vincent. Learning to generate samples from noise through infusion training. *ICLR*, 2017.

[BKB18] Shane T Barratt, Mykel J Kochenderfer, and Stephen P Boyd. Learning probabilistic trajectory models of aircraft in terminal airspace from position data. *IEEE Transactions on Intelligent Transportation Systems*, 2018.

[BL18] Jonathon Byrd and Zachary C Lipton. What is the effect of importance weighting in deep learning? *arXiv preprint arXiv:1812.03372*, 2018.

[BLC18] Avishek Joey Bose, Huan Ling, and Yanshuai Cao. Adversarial contrastive estimation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1021–1032, 2018.

[BLCW09] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[BLRW16] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016.

[BMR+20] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are Few-Shot learners. *arXiv preprint arXiv:2005.14165*, May 2020.

[BS95] Anthony J Bell and Terrence J Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, 7(6):1129–1159, 1995.

[BTLAY14] Yoshua Bengio, Eric Thibodeau-Laufer, Guillaume Alain, and Jason Yosinski. Deep generative stochastic networks trainable by backprop. 2014.

[BV18] Sergey Bartunov and Dmitry Vetrov. Few-shot generative modelling with generative matching networks. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings*

of the Twenty-First International Conference on Artificial Intelligence and Statistics, volume 84 of *Proceedings of Machine Learning Research*, pages 670–678. PMLR, 2018.

[BZW+19] David Bau, Jun-Yan Zhu, Jonas Wulff, William Peebles, Hendrik Strobelt, Bolei Zhou, and Antonio Torralba. Seeing what a gan cannot generate. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4502–4511, 2019.

[CD19] Louis Clouâtre and Marc Demers. Figr: Few-shot image generation with reptile. *arXiv preprint arXiv:1901.02199*, 2019.

[CDH+16] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. *arXiv preprint arXiv:1606.03657*, pages 2172–2180, June 2016.

[CFGH20] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, March 2020.

[CFWM18] Gordon Christie, Neil Fendley, James Wilson, and Ryan Mukherjee. Functional map of the world. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6172–6180, 2018.

[CH20] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*, November 2020.

[Chi20] Rewon Child. Very deep vaes generalize autoregressive models and can outperform them on images. *arXiv preprint arXiv:2011.10650*, 2020.

[Cho17] Alexandra Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big data*, 5(2):153–163, June 2017.

[CJL+19] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-Balanced loss based on effective number of samples. *arXiv preprint arXiv:1901.05555*, January 2019.

[CK17] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *ICCV*, 2017.

[CKNH20] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, February 2020.

[CKP09] Toon Calders, Faisal Kamiran, and Mykola Pechenizkiy. Building classifiers with independency constraints. In *2009 IEEE International Conference on Data Mining Workshops*, pages 13–18. IEEE, 2009.

[CKS+16] Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, November 2016.

[CMK+14] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[COR+16] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[CRBD18] Ricky T Q Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*, June 2018.

[CRT06] Emmanuel J Candes, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 59(8):1207–1223, 2006.

[Csi64] Imre Csiszár. Eine informationstheoretische ungleichung und ihre anwendung auf beweis der ergodizitaet von markoffschen ketten. *Magyer Tud. Akad. Mat. Kutato Int. Koezl.*, 8:85–108, 1964.

[CWG+19] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with Label-Distribution-Aware margin loss. *arXiv preprint arXiv:1906.07413*, June 2019.

[CWRV17] Flavio P Calmon, Dennis Wei, Karthikeyan Natesan Ramamurthy, and Kush R Varshney. Optimized data Pre-Processing for discrimination prevention. *arXiv preprint arXiv:1704.03354*, April 2017.

[CXH21] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training Self-Supervised vision transformers. *arXiv preprint arXiv:2104.02057*, April 2021.

[DBP+16] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, June 2016.

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, October 2018.

[DDJD21] Giannis Daras, Joseph Dean, Ajil Jalal, and Alexandros G Dimakis. Intermediate layer optimization for inverse problems using deep generative models. *arXiv preprint arXiv:2102.07364*, February 2021.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[DFHSJR01] Nando De Freitas, Pedro Højen-Sørensen, Michael I Jordan, and Stuart Russell. Variational mcmc. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 120–127. Morgan Kaufmann Publishers Inc., 2001.

[DHFLM+18] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, August 2018.

[DHK+17] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Openai baselines. https://github.com/openai/baselines, 2017.

[DJP+20] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford,

and Ilya Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

[DKB14] L Dinh, D Krueger, and Y Bengio. NICE: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[DKD16] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, May 2016.

[DM19] Yilun Du and Igor Mordatch. Implicit generation and generalization in Energy-Based models. *arXiv preprint arXiv:1903.08689*, March 2019.

[DN21] Prafulla Dhariwal and Alex Nichol. Diffusion models beat GANs on image synthesis. *arXiv preprint arXiv:2105.05233*, May 2021.

[DSDB16] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. *arXiv preprint arXiv:1605.08803*, May 2016.

[DT17] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[DV75] Monroe D Donsker and S R Srinivasa Varadhan. Asymptotic evaluation of certain markov process expectations for large time, I. *Communications on Pure and Applied Mathematics*, 28(1):1–47, 1975.

[DW19] Bin Dai and David Wipf. Diagnosing and enhancing VAE models. *arXiv preprint arXiv:1903.05789*, March 2019.

[EGSS14] Stefano Ermon, Carla P Gomes, Ashish Sabharwal, and Bart Selman. Designing fast absorbing markov chains. In *AAAI*, pages 849–855, 2014.

[ELS+18] Stephan Eissman, Daniel Levy, Rui Shu, Stefan Bartzsch, and Stefano Ermon. Bayesian optimization and attribute adjustment. In *Proc. 34th Conference on Uncertainty in Artificial Intelligence*, 2018.

[EN08] Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *14th ACM SIGKDD*, pages 213–220, 2008.

[ES15] Harrison Edwards and Amos Storkey. Censoring representations with an adversary. *arXiv preprint arXiv:1511.05897*, November 2015.

[ESM+18] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed Deep-RL with importance weighted Actor-Learner architectures. *arXiv preprint arXiv:1802.01561*, February 2018.

[ET94] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[ET15] Peter Englert and Marc Toussaint. Inverse KKT–learning cost functions of manipulation tasks from demonstrations. In *Proceedings of the International Symposium of Robotics Research*, 2015.

[FAdFW16] Jakob Foerster, Yannis Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In D D Lee, M Sugiyama, U V Luxburg, I Guyon, and R Garnett, editors, *Advances in Neural Information Processing Systems*, pages 2137–2145. Curran Associates, Inc., 2016.

[FCAL16] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016.

[FLA16] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, pages 49–58, June 2016.

[FLL17] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, October 2017.

[FMN16] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016.

[FT18] Farzan Farnia and David Tse. A convex duality framework for GANs. *arXiv preprint arXiv:1810.11740*, October 2018.

[FV12] Jerzy Filar and Koos Vrieze. *Competitive Markov decision processes*. Springer Science & Business Media, 2012.

[GAA+17] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein GANs. In I Guyon, U V Luxburg,

S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5767–5777. Curran Associates, Inc., 2017.

[GBR+07] Arthur Gretton, Karsten M Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J Smola. A kernel method for the Two-Sample-Problem. In B Schölkopf, J C Platt, and T Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 513–520. MIT Press, 2007.

[GC11] Mark Girolami and Ben Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

[GDE17] Aditya Grover, Manik Dhar, and Stefano Ermon. Flow-GAN: Combining maximum likelihood and adversarial learning in generative models. *arXiv preprint arXiv:1705.08868*, May 2017.

[GDVM16] Jack Gorham, Andrew B Duncan, Sebastian J Vollmer, and Lester Mackey. Measuring sample quality with diffusions. *arXiv preprint arXiv:1611.06972*, 2016.

[GE17] Aditya Grover and Stefano Ermon. Boosted generative models. *arXiv preprint arXiv:1702.08484*, February 2017.

[GE19] Aditya Grover and Stefano Ermon. Uncertainty autoencoders: Learning compressed representations via variational information maximization. In *AISTATS*, 2019.

[GGM08] Geoffrey J Gordon, Amy Greenwald, and Casey Marks. No-regret learning in convex games. In *Proceedings of the 25th international conference on Machine learning*, pages 360–367. ACM, 2008.

[GH12] Michael U Gutmann and Aapo Hyvärinen. Noise-Contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of machine learning research: JMLR*, 13(Feb):307–361, 2012.

[Gha03] Zoubin Ghahramani. Unsupervised learning. In *Summer School on Machine Learning*, pages 72–112. Springer, 2003.

[GKOV17] Weihao Gao, Sreeram Kannan, Sewoong Oh, and Pramod Viswanath. Estimating mutual information for Discrete-Continuous mixtures. *arXiv preprint arXiv:1709.06212*, September 2017.

[GM15] Jackson Gorham and Lester Mackey. Measuring sample quality with stein's method. In *Advances in Neural Information Processing Systems*, pages 226–234, 2015.

[GM17] Jackson Gorham and Lester Mackey. Measuring sample quality with kernels. *arXiv preprint arXiv:1703.01717*, 2017.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z Ghahramani, M Welling, C Cortes, N D Lawrence, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[Gre95] Peter J Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, pages 711–732, 1995.

[GRM+18] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018.

[GSA+19] Aditya Grover, Jiaming Song, Alekh Agarwal, Kenneth Tran, Ashish Kapoor, Eric Horvitz, and Stefano Ermon. Bias correction of learned generative models using Likelihood-Free importance weighting. *arXiv preprint arXiv:1906.09531*, pages 11058–11070, June 2019.

[GSA+20] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to Self-Supervised learning. *arXiv preprint arXiv:2006.07733*, June 2020.

[GSKE20] Nate Gruver, Jiaming Song, Mykel J Kochenderfer, and Stefano Ermon. Multi-agent adversarial inverse reinforcement learning with latent variables. In *International*

Conference on Autonomous Agents and MultiAgent Systems (extended abstract), pages 1855–1857, March 2020.

[GVSG15] Shuyang Gao, Greg Ver Steeg, and Aram Galstyan. Efficient estimation of mutual information for strongly dependent variables. In *Artificial intelligence and statistics*, pages 277–286, 2015.

[GZE19] Aditya Grover, Aaron Zweig, and Stefano Ermon. Graphite: Iterative generative modeling of graphs. In *International conference on machine learning*, pages 2434–2444. PMLR, 2019.

[HAP+18] Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*, 2018.

[Has70] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[HCDLZ18] Ming Hou, Brahim Chaib-Draa, Chao Li, and Qibin Zhao. Generative adversarial positive-unlabeled learning. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 2255–2261, 2018.

[HD18] Dan Hendrycks and Thomas G Dietterich. Benchmarking neural network robustness to common corruptions and surface variations. *arXiv preprint arXiv:1807.01697*, 2018.

[HE16] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *arXiv preprint arXiv:1606.03476*, pages 4565–4573, June 2016.

[Hen20] Olivier Henaff. Data-efficient image recognition with contrastive predictive coding. In *International Conference on Machine Learning*, pages 4182–4192. PMLR, 2020.

[HFLM+18] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

[HFW+19] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, November 2019.

[HG14] Matthew D Hoffman and Andrew Gelman. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

[HG16] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[Hin02] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, August 2002.

[HJ16] Matthew D Hoffman and Matthew J Johnson. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, volume 1, page 2. approximateinference.org, 2016.

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, June 2020.

[HKKT18] Steve Hanneke, Adam Tauman Kalai, Gautam Kamath, and Christos Tzamos. Actively avoiding nonsense in generative models. In *Conference On Learning Theory*, pages 209–227, 2018.

[HLCT19] Chen Huang, Yining Li, Change Loy Chen, and Xiaoou Tang. Deep imbalanced learning for face recognition and attribute prediction. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[HLLT16] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. Learning deep representation for imbalanced classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5375–5384, 2016.

[HLZ16] Lingxiao Huang, Pinyan Lu, and Chihao Zhang. Canonical paths for mcmc: from art to science. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 514–527. SIAM, 2016.

[HMC00] Sergiu Hart and Andreu Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.

[HMD18] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*, December 2018.

[HMMA+17] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. Inverse reward design. In *Advances in Neural Information Processing Systems*, pages 6765–6774, 2017.

[HMRAD16] Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. Cooperative inverse reinforcement learning. In *Advances in neural information processing systems*, pages 3909–3917, 2016.

[HNW19] Hisham Husain, Richard Nock, and Robert C Williamson. Adversarial networks and autoencoders: The Primal-Dual relationship and generalization bounds. *arXiv preprint arXiv:1902.00985*, February 2019.

[HPS16] Moritz Hardt, Eric Price, and Nathan Srebro. Equality of opportunity in supervised learning. *arXiv preprint arXiv:1610.02413*, pages 3315–3323, October 2016.

[HRD+19] Olivier J Hénaff, Ali Razavi, Carl Doersch, S M Ali Eslami, and Aaron van den Oord. Data-Efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, May 2019.

[HRU+17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two Time-Scale update rule converge to a local nash equilibrium. *arXiv preprint arXiv:1706.08500*, June 2017.

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised learning. In *The elements of statistical learning*, pages 485–585. Springer, 2009.

[HVD15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, March 2015.

[HW17] Zehao Huang and Naiyan Wang. Like what you like: Knowledge distill via neuron selectivity transfer. *arXiv preprint arXiv:1707.01219*, 2017.

[HWO98] Junling Hu, Michael P Wellman, and Others. Multiagent reinforcement learning: theoretical framework and an algorithm. In *ICML*, volume 98, pages 242–250. Citeseer, 1998.

[HXZ19] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. *arXiv preprint arXiv:1909.04656*, September 2019.

[HZ21] Drew A Hudson and C Lawrence Zitnick. Generative adversarial transformers. *arXiv preprint arXiv:2103.01209*, 2021.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, December 2015.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, February 2015.

[IZZE17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[JGP16] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, November 2016.

[JM12] Wenzel Jakob and Steve Marschner. Manifold exploration: a markov chain monte carlo technique for rendering scenes with difficult specular transport. *ACM Transactions on Graphics (TOG)*, 31(4):58, 2012.

[JSV08] Chris Jagodnik, Joseph Stella, and Dan Varon. Fusion tracking in air traffic control. *Journal of Air Traffic Control*, 50(1), 2008.

[JWAAN18] Ayush Jaiswal, Rex Yue Wu, Wael Abd-Almageed, and Prem Natarajan. Unsupervised adversarial invariance. In *Advances in Neural Information Processing Systems*, pages 5092–5102, 2018.

[Kak02] Sham M Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538, 2002.

[KALL17] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, October 2017.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, December 2014.

[KBKE18] Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz Eckstein. The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[KGE17] Jayesh K. Gupta and Maxim Egorov. Multi-agent deep reinforcement learning environment. `https://github.com/sisl/madrl`, 2017.

[KH+09] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[KHZ+19] Salman Khan, Munawar Hayat, Waqas Zamir, Jianbing Shen, and Ling Shao. Striking the right balance with uncertainty. *arXiv preprint arXiv:1901.07590*, January 2019.

[KJS+20] Kuno Kim, Akshat Jindal, Yang Song, Jiaming Song, Yanan Sui, and Stefano Ermon. Imitation with neural density models. *arXiv:2010.09808*, January 2020.

[KKBZ19] Animesh Koratana, Daniel Kang, Peter Bailis, and Matei Zaharia. Lit: Learned intermediate representation training for model compression. In *International Conference on Machine Learning*, pages 3509–3518, 2019.

[KLA18] Tero Karras, Samuli Laine, and Timo Aila. A Style-Based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*, December 2018.

[KLA+20] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020.

[KMR16] Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. Inherent Trade-Offs in the fair determination of risk scores. *arXiv preprint arXiv:1609.05807*, September 2016.

[KPK18] Jangho Kim, SeongUk Park, and Nojun Kwak. Paraphrasing complex network: Network compression via factor transfer. In *Advances in Neural Information Processing Systems*, pages 2760–2769, 2018.

[KS20] Zahra Kadkhodaie and Eero P Simoncelli. Solving linear inverse problems using the prior implicit in a denoiser. *arXiv preprint arXiv:2007.13640*, July 2020.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In F Pereira, C J C Burges, L Bottou, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[KSJ+16a] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In D D Lee, M Sugiyama, U V Luxburg, I Guyon, and R Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4743–4751. Curran Associates, Inc., 2016.

[KSJ+16b] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improving variational inference with inverse autoregressive flow. *arXiv preprint arXiv:1606.04934*, June 2016.

[KTHS18] Junpei Komiyama, Akiko Takeda, Junya Honda, and Hajime Shimao. Nonconvex optimization for regression with fairness constraints. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2742–2751, Stockholmsmässan, Stockholm Sweden, 2018. PMLR.

[KW13] Diederik P Kingma and Max Welling. Auto-Encoding variational bayes. *arXiv preprint arXiv:1312.6114v10*, December 2013.

[LB14] David P Landau and Kurt Binder. *A guide to Monte Carlo simulations in statistical physics*. Cambridge university press, 2014.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LBC14] Xiaomin Lin, Peter A Beling, and Randy Cogill. Multi-agent inverse reinforcement learning for zero-sum games. *arXiv preprint arXiv:1403. 6508*, 2014.

[LBC17] Shuang Liu, Olivier Bousquet, and Kamalika Chaudhuri. Approximation and convergence properties of generative adversarial learning. *arXiv preprint arXiv:1705.08991*, May 2017.

[LC18] Shuang Liu and Kamalika Chaudhuri. The inductive bias of restricted f-GANs. *arXiv preprint arXiv:1809.04542*, September 2018.

[LCC+17] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. MMD GAN: Towards deeper understanding of moment matching network. *arXiv preprint arXiv:1705.08584*, May 2017.

[Lev18] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, May 2018.

[LGL+20] Chunyuan Li, Xiang Gao, Yuan Li, Baolin Peng, Xiujun Li, Yizhe Zhang, and Jianfeng Gao. Optimus: Organizing sentences via pre-trained modeling of a latent space. *arXiv preprint arXiv:2004.04092*, 2020.

[LHP+15] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, September 2015.

[Lin88] Ralph Linsker. Self-organization in a perceptual network. *Computer*, 21(3):105–117, March 1988.

[Lit94] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the eleventh international conference on machine learning*, volume 157, pages 157–163, 1994.

[LKG19] Zeju Li, Konstantinos Kamnitsas, and Ben Glocker. Overfitting of neural nets under class imbalance: Analysis and improvements for segmentation. *arXiv preprint arXiv:1907.10982*, July 2019.

[LLC+17] Chunyuan Li, Hao Liu, Changyou Chen, Yuchen Pu, Liqun Chen, Ricardo Henao, and Lawrence Carin. Alice: Towards understanding adversarial learning for joint distribution matching. *Advances in Neural Information Processing Systems*, 30:5495–5503, 2017.

[LLTZ18] Qiang Liu, Lihong Li, Ziyang Tang, and Dengyong Zhou. Breaking the curse of horizon: Infinite-Horizon Off-Policy estimation. *arXiv preprint arXiv:1810.12429*, October 2018.

[LLW17] Yu Liu, Hongyang Li, and Xiaogang Wang. Rethinking feature discrimination and polymerization for large-scale recognition. *arXiv preprint arXiv:1710.00870*, 2017.

[LLWT15] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[LMB+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[LSE17a] Yunzhu Li, Jiaming Song, and Stefano Ermon. InfoGAIL: Interpretable imitation learning from visual demonstrations. In I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3812–3822. Curran Associates, Inc., March 2017.

[LSE17b] Yunzhu Li, Jiaming Song, and Stefano Ermon. Infogail: Interpretable imitation learning from visual demonstrations. *arXiv preprint arXiv:1703.08840*, 2017.

[LSL+15] Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard Zemel. The variational fair autoencoder. *arXiv preprint arXiv:1511.00830*, November 2015.

[LSLW16] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning*, pages 1558–1566. PMLR, 2016.

[LSM+17] Christos Louizos, Uri Shalit, Joris M Mooij, David Sontag, Richard Zemel, and Max Welling. Causal effect inference with deep Latent-Variable models. In I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6446–6456. Curran Associates, Inc., 2017.

[LSZ15] Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In *International Conference on Machine Learning*, pages 1718–1727. PMLR, 2015.

[LWT+17] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-Agent Actor-Critic for mixed Cooperative-Competitive environments. *arXiv preprint arXiv:1706.02275*, June 2017.

[LWYY16] Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. Large-Margin softmax loss for convolutional neural networks. *arXiv preprint arXiv:1612.02295*, December 2016.

[LYC17] Hoang M Le, Yisong Yue, and Peter Carr. Coordinated Multi-Agent imitation learning. *arXiv preprint arXiv:1703.03121*, March 2017.

[LZL+17] Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 464–473. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

[MAK19] Sudipto Mukherjee, Himanshu Asnani, and Sreeram Kannan. CCMI : Classifier based conditional mutual information estimation. *arXiv preprint arXiv:1906.01824*, June 2019.

[MC18] Zhuang Ma and Michael Collins. Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency. *arXiv preprint arXiv:1809.01812*, September 2018.

[MCPZ18] David Madras, Elliot Creager, Toniann Pitassi, and Richard Zemel. Learning adversarially fair and transferable representations. *arXiv preprint arXiv:1802.06309*, February 2018.

[MEHS21] Gautam Mittal, Jesse Engel, Curtis Hawthorne, and Ian Simon. Symbolic music generation with diffusion models. *arXiv preprint arXiv:2103.16091*, March 2021.

[MG15] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International Conference on Machine Learning*, pages 2408–2417, 2015.

[MGB+18] Daniel Moyer, Shuyang Gao, Rob Brekelmans, Greg Ver Steeg, and Aram Galstyan. Invariant representations without adversarial training. *arXiv preprint arXiv:1805.09458*, pages 9102–9111, May 2018.

[MJMO12] Laëtitia Matignon, Laurent Jeanpierre, Abdel-Illah Mouaddib, and Others. Coordinated Multi-Robot exploration under communication constraints using decentralized markov decision processes. In *AAAI*, 2012.

[MK13] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in neural information processing systems*, pages 2265–2273, 2013.

[MKKY18] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, February 2018.

[MKS+19] Ali Malik, Volodymyr Kuleshov, Jiaming Song, Danny Nemer, Harlan Seymour, and Stefano Ermon. Calibrated Model-Based deep reinforcement learning. *arXiv preprint arXiv:1906.08312*, June 2019.

[ML16] Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*, October 2016.

[MLX+17] Xudong Mao, Qing Li, Haoran Xie, Raymond Y K Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802. openaccess.thecvf.com, 2017.

[MO14] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, November 2014.

[MP95] Richard D McKelvey and Thomas R Palfrey. Quantal response equilibria for normal form games. *Games and economic behavior*, 10(1):6–38, July 1995.

[MP98] Richard D McKelvey and Thomas R Palfrey. Quantal response equilibria for extensive form games. *Experimental economics*, 1(1):9–41, 1998.

[MPBS15] Elman Mansimov, Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Generating images from captions with attention. *arXiv preprint arXiv:1511.02793*, 2015.

[MS17] Youssef Mroueh and Tom Sercu. Fisher GAN. *arXiv preprint arXiv:1705.09675*, May 2017.

[MS18] David McAllester and Karl Stratos. Formal limitations on the measurement of mutual information. *arXiv preprint arXiv:1811.04251*, November 2018.

[MS20] David McAllester and Karl Stratos. Formal limitations on the measurement of mutual information. In *International Conference on Artificial Intelligence and Statistics*, pages 875–884, 2020.

[MSC+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C J C Burges, L Bottou, M Welling, Z Ghahramani, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[MSJ+15] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[Mül97] Alfred Müller. Integral probability metrics and their generating classes of functions. *Advances in applied probability*, 29(2):429–443, June 1997.

[MWHDF12] Nimalan Mahendran, Ziyu Wang, Firas Hamze, and Nando De Freitas. Adaptive mcmc with bayesian optimization. In *AISTATS*, volume 22, pages 751–760, 2012.

[N+11] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2:113–162, 2011.

[Naj20] Alex Najibi. Racial Discrimination in Face Recognition Technology. https://sitn.hms.harvard.edu/flash/2020/racial-discrimination-in-face-recognition-technology/, 2020.

[NBVS04] Ilya Nemenman, William Bialek, and Rob De Ruyter Van Steveninck. Entropy and information in neural spike trains: Progress on the sampling problem. *Physical Review E*, 69(5):056111, 2004.

[NCB+17] Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4467–4477, 2017.

[NCT16] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training generative neural samplers using variational divergence minimization. *arXiv preprint arXiv:1606.00709*, June 2016.

[NF16] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer, 2016.

[NHR99] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pages 278–287, 1999.

[NRO00] Andrew Y Ng, Stuart J Russell, and Others. Algorithms for inverse reinforcement learning. In *Icml*, pages 663–670, 2000.

[NSS+20] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics*, pages 4474–4484. PMLR, 2020.

[NWC+11] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. *NIPS workshop*, 2011.

[NWJ08] Xuanlong Nguyen, Martin J Wainwright, and Michael I Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *arXiv preprint arXiv:0809.0853*, (11):5847–5861, September 2008.

[NWJ10] XuanLong Nguyen, Martin J Wainwright, and Michael I Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010.

[OLB+19] Sherjil Ozair, Corey Lynch, Yoshua Bengio, Aaron van den Oord, Sergey Levine, and Pierre Sermanet. Wasserstein dependency measure for representation learning. *arXiv preprint arXiv:1903.11780*, March 2019.

[PAED17] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. *arXiv preprint arXiv:1705.05363*, May 2017.

[PB15] HL Prasad and Shalabh Bhatnagar. A study of gradient descent schemes for general-sum stochastic games. *arXiv preprint arXiv:1507.00093*, 2015.

[PHS+18] Tiziano Portenier, Qiyang Hu, Attila Szabó, Siavash Arjomand Bigdeli, Paolo Favaro, and Matthias Zwicker. Faceshop: Deep sketch-based face image editing. *ACM Transactions on Graphics*, 37(4), 2018.

[PK19] Sung Woo Park and Junseok Kwon. Sphere generative adversarial network based on geometric moment matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4292–4301, 2019.

[Pom91] Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

[POO+19] Ben Poole, Sherjil Ozair, Aaron van den Oord, Alexander A Alemi, and George Tucker. On variational bounds of mutual information. *arXiv preprint arXiv:1905.06922*, 2019.

[POvdO+19] Ben Poole, Sherjil Ozair, Aaron van den Oord, Alexander A Alemi, and George Tucker. On variational bounds of mutual information. *arXiv preprint arXiv:1905.06922*, May 2019.

[PPZS20] Wei Ping, Kainan Peng, Kexin Zhao, and Zhao Song. Waveflow: A compact flow-based model for raw audio. In *International Conference on Machine Learning*, pages 7706–7716. PMLR, 2020.

[PWS+21] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. StyleCLIP: Text-Driven manipulation of StyleGAN imagery. *arXiv preprint arXiv:2103.17249*, March 2021.

[PYW+17] Peng Peng, Quan Yuan, Ying Wen, Yaodong Yang, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent Bidirectionally-Coordinated nets for learning to play StarCraft combat games. *arXiv preprint arXiv:1703. 10069*, 2017.

[PZW+20] Taesung Park, Jun-Yan Zhu, Oliver Wang, Jingwan Lu, Eli Shechtman, Alexei A Efros, and Richard Zhang. Swapping autoencoder for deep image manipulation. *arXiv preprint arXiv:2007.00653*, July 2020.

[RAB10] Stephane Ross and J Andrew Bagnell. Efficient reductions for imitation learning. *AISTATS*, pages 661–668, 2010.

[RAY+16] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *International Conference on Machine Learning*, pages 1060–1069. PMLR, 2016.

[RGB11] Stéphane Ross, Geoffrey J Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, page 6, 2011.

[RGB14] Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *Artificial Intelligence and Statistics*, pages 814–822. jmlr.org, April 2014.

[RGZH12] Tummalapalli Sudhamsh Reddy, Vamsikrishna Gopikrishna, Gergely Zaruba, and Manfred Huber. Inverse reinforcement learning for decentralized non-cooperative multiagent systems. In *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, pages 1930–1935. IEEE, 2012.

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[RKH+21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, February 2021.

[RLM18] Mihaela Rosca, Balaji Lakshminarayanan, and Shakir Mohamed. Distribution matching in variational inference. *arXiv preprint arXiv:1802.06847*, February 2018.

[RM15] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, May 2015.

[RMC15] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[RMRV18] Suman Ravuri, Shakir Mohamed, Mihaela Rosca, and Oriol Vinyals. Learning implicit generative models with the method of learned moments. *arXiv preprint arXiv:1806.11006*, 80:4311–4320, June 2018.

[RMW14] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[RNSS18] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

[Roc70] R Tyrrell Rockafellar. *Convex analysis*, volume 28. Princeton university press, 1970.

[ROV19] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *arXiv preprint arXiv:1906.00446*, 2019.

[RPG+21] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-Shot Text-to-Image generation. *arXiv preprint arXiv:2102.12092*, February 2021.

[RR98] Gareth O Roberts and Jeffrey S Rosenthal. Optimal scaling of discrete approximations to langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268, 1998.

[RRGGP12] Avraham Ruderman, Mark Reid, Dario Garcia-Garcia, and James Petterson. Tighter variational representations of f-divergences via restriction to probability measures. *arXiv preprint arXiv:1206.4664*, June 2012.

[RSS+18] Hongyu Ren, Russell Stewart, Jiaming Song, Volodymyr Kuleshov, and Stefano Ermon. Adversarial constraint learning for structured prediction. *arXiv preprint arXiv:1805.10561*, May 2018.

[RTG00] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

[SAMR18] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:187–210, 2018.

[SAS17] Bradly C Stadie, Pieter Abbeel, and Ilya Sutskever. Third-Person imitation learning. *arXiv preprint arXiv:1703.01703*, March 2017.

[SB98] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[SDWMG15] Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *arXiv preprint arXiv:1503.03585*, March 2015.

[SE19a] Jiaming Song and Stefano Ermon. Bridging the gap between f-gans and wasserstein gans. *arXiv preprint arXiv:1910.09779*, 2019.

[SE19b] Jiaming Song and Stefano Ermon. Understanding the limitations of variational mutual information estimators. *arXiv preprint arXiv:1910.06222*, October 2019.

[SE19c] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *arXiv preprint arXiv:1907.05600*, July 2019.

[SE20a] Jiaming Song and Stefano Ermon. Bridging the gap between f-gans and wasserstein gans. In *International Conference on Machine Learning*, July 2020.

[SE20b] Jiaming Song and Stefano Ermon. Multi-label contrastive predictive coding. *arXiv preprint arXiv:2007.09852*, 2020.

[SE20c] Jiaming Song and Stefano Ermon. Understanding the limitations of variational mutual information estimators. In *International Conference on Learning Representations*, April 2020.

[SFG+09] Bharath K Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, and Gert R G Lanckriet. On integral probability metrics, φ-divergences and binary classification. *arXiv preprint arXiv:0901.2698*, January 2009.

[SGS15] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[SGTZ20] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9243–9252, 2020.

[SGZ+16] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. *arXiv preprint arXiv:1606.03498*, pages 2226–2234, June 2016.

[SHM+16] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[SHPL19] Yi Lin Sung, Sung-Hsien Hsieh, Soo-Chang Pei, and Chun-Shien Lu. Difference-seeking generative adversarial network–unseen sample generation. In *International Conference on Learning Representations*, 2019.

[SKG+18] Jiaming Song, Pratyusha Kalluri, Aditya Grover, Shengjia Zhao, and Stefano Ermon. Learning controllable fair representations. *arXiv preprint arXiv:1812.04218*, December 2018.

[SKG+19] Jiaming Song, Pratyusha Kalluri, Aditya Grover, Shengjia Zhao, and Stefano Ermon. Learning controllable fair representations. In *International Conference on Artificial Intelligence and Statistics*, pages 2164–2173, April 2019.

[SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. *arXiv preprint arXiv:1503.03832*, March 2015.

[SKS+21] Abhishek Sinha*, Ayush Kumar*, Jiaming Song*, Burak Ukzent, Hongxia Jin, and Stefano Ermon. Negative data augmentation. In *International Conference on Learning Representations*, April 2021.

[SKW15] Tim Salimans, Diederik P Kingma, and Max Welling. Markov chain monte carlo and variational inference: Bridging the gap. In *ICML*, volume 37, pages 1218–1226, 2015.

[ŠKZK16] Adrian Šošic, Wasiur R KhudaBukhsh, Abdelhak M Zoubir, and Heinz Koeppl. Inverse reinforcement learning in swarm systems. *stat*, 1050:17, 2016.

[SME20] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, April 2020.

[SRSE18] Jiaming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon. Multi-agent generative adversarial imitation learning. *Advances in neural information processing systems*, pages 7461–7472, 2018.

[SSDK+20] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[SSK12] Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. *Density ratio estimation in machine learning*. Cambridge University Press, 2012.

[SSME21] Abhishek Sinha*, Jiaming Song*, Chenlin Meng, and Stefano Ermon. D2c: Diffusion-denoising models for few-shot conditional generation. *arXiv:2106.06819*, June 2021.

[SZE17] Jiaming Song, Shengjia Zhao, and Stefano Ermon. A-nice-mc: Adversarial training for mcmc. *arXiv preprint arXiv:1706.07561*, June 2017.

[TBGS17] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein Auto-Encoders. *arXiv preprint arXiv:1711.01558*, November 2017.

[TCH+18] Chenyang Tao, Liqun Chen, Ricardo Henao, Jianfeng Feng, and Lawrence Carin Duke. Chi-square generative adversarial network. In *International conference on machine learning*, pages 4887–4896. PMLR, 2018.

[TDR+19] Michael Tschannen, Josip Djolonga, Paul K Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. *arXiv preprint arXiv:1907.13625*, July 2019.

[THF+18] Ryan Turner, Jane Hung, Eric Frank, Yunus Saatci, and Jason Yosinski. Metropolis-Hastings generative adversarial networks. *arXiv preprint arXiv:1811.11357*, November 2018.

[TIY+19] Hiroshi Takahashi, Tomoharu Iwata, Yuki Yamanaka, Masanori Yamada, and Satoshi Yagi. Variational autoencoder with implicit optimal priors. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):5066–5073, July 2019.

[TKI19] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. *arXiv preprint arXiv:1910.10699*, October 2019.

[TW16] Jakub M Tomczak and Max Welling. Improving variational auto-encoders using householder flow. *arXiv preprint arXiv:1611.09630*, 2016.

[TW17] Jakub M Tomczak and Max Welling. VAE with a VampPrior. *arXiv preprint arXiv:1705.07120*, May 2017.

[TZ15] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. *arXiv preprint arXiv:1503.02406*, March 2015.

[USS+16] Masatoshi Uehara, Issei Sato, Masahiro Suzuki, Kotaro Nakayama, and Yutaka Matsuo. Generative adversarial nets from a density ratio estimation perspective. *arXiv preprint arXiv:1610.02920*, October 2016.

[vdODZ+16] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, September 2016.

[vdOKK16] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, January 2016.

[vdOKV+16] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with PixelCNN decoders. *arXiv preprint arXiv:1606.05328*, June 2016.

[vdOLV18] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, July 2018.

[vdOVK17] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *arXiv preprint arXiv:1711.00937*, November 2017.

[VFH+18] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, September 2018.

[Vil08] Cédric Villani. *Optimal Transport: Old and New*. Springer Science & Business Media, October 2008.

[VK20] Arash Vahdat and Jan Kautz. NVAE: A deep hierarchical variational autoencoder. *arXiv preprint arXiv:2007.03898*, July 2020.

[Wil92] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[WLLC18] Feng Wang, Weiyang Liu, Haijun Liu, and Jian Cheng. Additive margin softmax for face verification. *arXiv preprint arXiv:1801.05599*, January 2018.

[WLZ+18] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018.

[WM97] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[WMG+17] Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems*, pages 5285–5294, August 2017.

[WRH17] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Learning to model the tail. In *Advances in Neural Information Processing Systems*, pages 7029–7039, 2017.

[WSSG18] Li Wenliang, Dougal Sutherland, Heiko Strathmann, and Arthur Gretton. Learning deep kernels for exponential family densities. *arXiv preprint arXiv:1811.08357*, November 2018.

[WXYL18] Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. Unsupervised feature learning via Non-Parametric instance-level discrimination. *arXiv preprint arXiv:1805.01978*, May 2018.

[XLZ+21] Zhe Xie, Chengxuan Liu, Yichi Zhang, Hongtao Lu, Dong Wang, and Yue Ding. Adversarial and contrastive variational autoencoder for sequential recommendation. *arXiv preprint arXiv:2103.10693*, March 2021.

[XSZ+20] Yinghao Xu, Yujun Shen, Jiapeng Zhu, Ceyuan Yang, and Bolei Zhou. Generative hierarchical features from synthesizing images. *arXiv e-prints*, pages arXiv–2007, 2020.

[XYXW21] Weihao Xia, Yujiu Yang, Jing-Hao Xue, and Baoyuan Wu. TediGAN: Text-Guided diverse face image generation and manipulation. 2021.

[XZS+20] Yilun Xu, Shengjia Zhao, Jiaming Song, Russell Stewart, and Stefano Ermon. A theory of usable information under computational constraints. *arXiv preprint arXiv:2002.10689*, April 2020.

[YHO+19] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6023–6032, 2019.

[YLY+18] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for Goal-Directed molecular graph generation. *arXiv preprint arXiv:1806.02473*, June 2018.

[YSE19] Lantao Yu, Jiaming Song, and Stefano Ermon. Multi-Agent adversarial inverse reinforcement learning. *arXiv preprint arXiv:1907.13220*, July 2019.

[YYFE19] Lantao Yu, Tianhe Yu, Chelsea Finn, and Stefano Ermon. Meta-Inverse reinforcement learning with probabilistic context variables. *arXiv preprint arXiv:1909.09314*, September 2019.

[YZWY17] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, pages 2852–2858, 2017.

[ZBD11] Brian D Ziebart, J Andrew Bagnell, and Anind K Dey. Maximum causal entropy correlated equilibria for markov games. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '11, pages 207–214, Richland, SC, 2011. International Foundation for Autonomous Agents and Multiagent Systems.

[ZCDLP17] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. October 2017.

[Zho18] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National science review*, 5(1):44–53, 2018.

[ZIE16] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.

[Zli15] Indre Zliobaite. On the relation between accuracy and fairness in binary classification. *arXiv preprint arXiv:1505.05723*, May 2015.

[ZLK+17] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2017.

[ZMBD08] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[ZPIE17] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired Image-to-Image translation using Cycle-Consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, March 2017.

[ZRY+18] Shengjia Zhao, Hongyu Ren, Arianna Yuan, Jiaming Song, Noah Goodman, and Stefano Ermon. Bias and generalization in deep generative models: An empirical study. In *Advances in Neural Information Processing Systems*, pages 10792–10801, 2018.

[ZSE17a] Shengjia Zhao, Jiaming Song, and Stefano Ermon. InfoVAE: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*, June 2017.

[ZSE17b] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Towards deeper understanding of variational autoencoding models. *arXiv preprint arXiv:1702.08658*, February 2017.

[ZSE18a] Shengjia Zhao, Jiaming Song, and Stefano Ermon. The information autoencoding family: A lagrangian perspective on latent variable generative models. *arXiv preprint arXiv:1806.06514*, June 2018.

[ZSE18b] Shengjia Zhao, Jiaming Song, and Stefano Ermon. A lagrangian perspective on latent variable generative models. In *Proc. 34th Conference on Uncertainty in Artificial Intelligence*, 2018.

[ZSZZ20] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *European Conference on Computer Vision*, pages 592–608. Springer, 2020.

[ZVRG15] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. Learning fair classifiers. *arXiv preprint arXiv:1507. 05259*, 2015.

[ZWS+13] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. In *International Conference on Machine Learning*, pages 325–333, February 2013.

[ZZY19] Chengxu Zhuang, Alex Lin Zhai, and Daniel Yamins. Local aggregation for unsupervised learning of visual embeddings. *arXiv preprint arXiv:1903.12355*, March 2019.

[ZZZZ19] Jiapeng Zhu, Deli Zhao, Bolei Zhou, and Bo Zhang. Lia: Latently invertible autoencoder with adversarial learning. 2019.