HEALING FAILURES AND IMPROVING GENERALIZATION IN DEEP GENERATIVE MODELLING

Mingtian Zhang

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Computer Science

University College London

October 7, 2023

Declaration

I, Mingtian Zhang, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Deep generative modeling is a crucial and rapidly developing area of machine learning, with numerous potential applications, including data generation, anomaly detection, data compression, and more. Despite the significant empirical success of many generative models, some limitations still need to be addressed to improve their performance in certain cases. This thesis focuses on understanding the limitations of generative modeling in common scenarios and proposes corresponding techniques to alleviate these limitations and improve performance in practical generative modeling applications. Specifically, the thesis is divided into two sub-topics: one focusing on the training and the other on the generalization of generative models. A brief introduction to each sub-topic is provided below.

Generative models are typically trained by optimizing their fit to the data distribution. This is achieved by minimizing a statistical divergence between the model and data distributions. However, there are cases where these divergences fail to accurately capture the differences between the model and data distributions, resulting in poor performance of the trained model. In the first part of the thesis, we discuss the two situations where the classic divergences are ineffective for training the models:

- KL divergence fails to train implicit models for manifold modeling tasks.
- Fisher divergence cannot distinguish the mixture proportions for modeling target multi-modality distribution.

For both failure modes, we investigate the theoretical reasons underlying the failures of KL and Fisher divergences in modelling certain types of data distributions. We

propose techniques that address the limitations of these divergences, enabling more reliable estimation of the underlying data distributions.

While the generalization of classification or regression models has been extensively studied in machine learning, the generalization of generative models is a relatively under-explored area. In the second part of this thesis, we aim to address this gap by investigating the generalization properties of generative models. Specifically, we investigate two generalization scenarios:

- In-distribution (ID) generalization of probabilistic models, where the test data and the training data are from the same distribution.
- Out-of-distribution (OOD) generalization of probabilistic models, where the test data and the training data can come from different distributions.

In the context of ID generalization, our emphasis rests on the Variational Auto-Encoder (VAE) model, and for OOD generalization, we primarily explore autoregressive models. By studying the generalization properties of the models, we demonstrate how to design new models or training criteria that improve the performance of practical applications, such as lossless compression and OOD detection.

The findings of this thesis shed light on the intricate challenges faced by generative models in both training and generalization scenarios. Our investigations into the inefficacies of classic divergences like KL and Fisher highlight the importance of tailoring modeling techniques to the specific characteristics of data distributions. Additionally, by delving into the generalization aspects of generative models, this work pioneers insights into the ID and OOD scenarios, a domain not extensively covered in current literature. Collectively, the insights and techniques presented in this thesis provide valuable contributions to the community, fostering an environment for the development of more robust and reliable generative models. It's our hope that these take-home messages will serve as a foundation for future research and applications in the realm of deep generative modeling.

Impact statement

The impact of the research presented in this thesis is significant as it contributes to the field of machine learning by proposing novel techniques for improving the training and generalization of generative models. Most parts of the thesis have been published at prestigious machine learning conferences such as ICML and NeurIPS, which demonstrates the impact and relevance of the research. For reference purposes, we provide a detailed list of included publications and pre-prints below:

- *Spread Divergences*
  Mingtian Zhang, Peter Hayes, Thomas Bird, Raza Habib and David Barber
  International Conference on Machine Learning (ICML), 2020

- *Towards Healing the Blindness of Score Matching*
  Mingtian Zhang, Oscar Keys, Peter Hayes, David Barber, Brooks Paige and François-Xavier Briol
  Workshop on Score-Based Methods, NeurIPS, 2022.

- *Generalization Gap in Amortized Inference*
  Mingtian Zhang, Peter Hayes and David Barber
  Neural Information Processing Systems (NeurIPS), 2022

- *On the Out-of-Distribution Generalization of Probabilistic Image Modelling*
  Mingtian Zhang, Andi Zhang and Steven McDonagh
  Neural Information Processing Systems (NeurIPS), 2021

- *Parallel Neural Local Lossless Compression*
  Mingtian Zhang, James Townsend, Ning Kang and David Barber
  Pre-print arXiv:2201.05213.

There are other published works contributed over the course of the PhD that are not included in this thesis are:

- *Moment Matching Denoising Gibbs Sampling*
  Mingtian Zhang, Alex Hawkins-Hooker, Brooks Paige, David Barber
  Neural Information Processing Systems (NeurIPS), 2023

- *Spread Flows for Manifold Modelling*
  Mingtian Zhang, Yitong Sun, Steven McDonagh and Chen Zhang
  Artificial Intelligence and Statistics (AISTATS), 2023

- *Active Forgetting of Negative Transfer in Continual Learning*
  Liyuan Wang, **Mingtian Zhang**, Zhongfan Jia, Qian Li, Kaisheng Ma, Chenglong Bao, Jun Zhu and Yi Zhong
  Neural Information Processing Systems (NeurIPS), 2021.

Acknowledgements

I would like to begin by expressing my deepest gratitude to my supervisor, David Barber, for his unwavering support and invaluable guidance throughout my PhD journey. Without his expertise, mentorship, and encouragement, I would not have been able to undertake this ambitious endeavor and bring it to successful completion.

My heartfelt thanks go to Alex Botev, who patiently addressed my inquiries, even those that might have seemed elementary during my PhD's initial stages. I would also like to express my sincere appreciation to my second supervisor, Brooks Paige, for the stimulating discussions and invaluable feedback on research ideas that have been instrumental in shaping my work. I have also learned a lot from James Townsend and François-Xavier Briol on lossless compression and score-based methods, respectively. I feel very fortunate to work with many brilliant collaborators: Peter Hayes, Thomas Bird, Raza Habib, Alex Hawkins-Hooker, Andi Zhang, Tim Xiao, and Oscar Keys. Special thanks to Peter Hayes, Tianlin Xu, Harshil Shah, and Hippolyt Ritter for making the research life much more enjoyable.

I am also deeply grateful to Huawei for providing me with the opportunity to pursue a part-time internship during my PhD studies. In particular, I would like to extend my heartfelt thanks to my mentors and collaborators, Jun Yao, Zhengguo Li, Wei Zhang, Yongxin Yang, Yitong Sun, Chen Zhang, Steven McDonagh, Shifeng Zhang and Ning Kang, for their invaluable help and guidance. This experience has exposed me to the practical challenges of real-world problems and has played a significant role in shaping the direction of my research.

Lastly, my endless gratitude is for my parents and my girlfriend, Shuyi Huang. Their unwavering love and support have been my bedrock throughout this journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

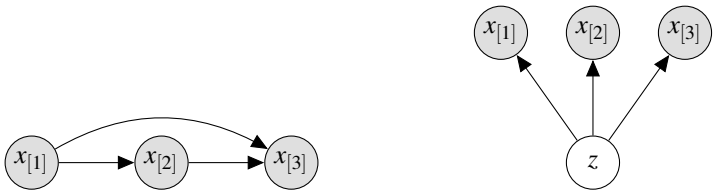
1.1 Probabilistic Modelling

Consider a finite set of training data, denoted as $\mathscr{X}_{train} = \{x_1, x_2, \ldots, x_N\}$, drawn from an underlying and unknown data distribution $\mathbb{P}_d$, where the subscript 'd' represents 'data'. The primary objective of probabilistic modeling is to develop a model, $\mathbb{Q}_\theta$, parameterized by θ, which closely approximates $\mathbb{P}_d$. This modeling process can be divided into two main phases:

1. **Designing a Model $\mathbb{Q}_\theta$ to Represent Data Distribution $\mathbb{P}_d$:** Numerous models have been proposed to encapsulate different data types, such as autoregressive models and latent variable models. Classic machine learning textbooks offer an extensive introduction to these model designs [Bishop, 2006, Barber, 2012, Murphy, 2012].

   In this thesis, our primary attention will be on three types of models: the latent variable model, the autoregressive model, and the energy-based model. These models have been remarkably successful in representing natural data, especially images. We will delve deeper into the intricacies of these models and their training methodologies in subsequent sections.

2. **Defining a Criterion for Learning Model Parameter θ:** To measure the 'closeness', we first need to define a divergence or distance between two distributions $\mathbb{P}$ and $\mathbb{Q}$, denoted as $\mathrm{D}(\mathbb{P}||\mathbb{Q})$. A valid divergence should satisfy



(a) Aggressive Model (b) Latent Variable Model

Figure 1.1: Graphical visualizations of the autoregressive and latent variable models.

the following properties [Dragomir, 2005][1]:

$$\mathrm{D}(\mathbb{P}||\mathbb{Q}) \geq 0 \quad \text{and} \quad \mathrm{D}(\mathbb{P}||\mathbb{Q}) = 0 \Leftrightarrow \mathbb{P} = \mathbb{Q}. \tag{1.1}$$

Several divergences, like the f-divergence [Csiszár, 1972, 1967, Ali and Silvey, 1966], Maximum Mean discrepancy [Gretton et al., 2012] (MMD) or Wasserstein distance [Villani, 2009], have been utilized effectively in probabilistic modeling. Among these, the KL divergence [Kullback and Leibler, 1951] stands out in machine learning literature, owing to its intrinsic connection with Maximum Likelihood Estimation [Casella and Berger, 2021] and its significance in information theory [MacKay, 2003, Lehmann, 2004]. In the following sections, we'll delve into the use of KL divergence (or its upper bounds) for training the autoregressive and latent variable models. Although training energy-based models using KL divergence presents challenges, recent innovations have highlighted the Fisher divergence [Aapo, 2005, Hyvärinen, 2007] as a promising approach, which we will discuss in due course.

1.1.1 KL Divergence and Maximum Likelihood Estimation

For absolutely continuous[2]($a.c.$) (with respect to (w.r.t.) Lebesgue measure) or discrete distributions $\mathbb{P}$, $\mathbb{Q}$ with probability density functions[3] (pdfs), the f-divergence

[1]Two distributions being equal $\mathbb{P} = \mathbb{Q}$ can be interpreted to mean that the cdfs (cumulative distribution functions) of the two distributions match.

[2]A distribution is $a.c.$ if its cumulative distribution function (cdf) is $a.c.$. In this case, it has a density function, see [Tao, 2011] for an introduction.

[3]For simplicity, we can also use the notation $p_d(x)$ to represent the probability mass function (pmf) when $\mathbb{P}_d$ is a discrete distribution.

is defined as

$$\mathrm{D}_f(\mathbb{P}||\mathbb{Q}) \equiv \mathrm{D}_f(p||q) = \int f\left(\frac{p(x)}{q(x)}\right) q(x)dx, \tag{1.2}$$

where $f(x)$ is a convex function with $f(1) = 0$. When p and q have the same support, the f-divergence is a valid divergence such that[4],

$$\mathrm{D}_f(p||q) = 0 \Leftrightarrow p = q \; a.e. \Leftrightarrow \mathbb{P} = \mathbb{Q}, \tag{1.3}$$

see [Csiszár, 1967, Ali and Silvey, 1966]. A popular special case of f-divergence is the Kullback-Leibler (KL) divergence with the convex function $f(\cdot) = -\log(\cdot)$. The KL divergence is defined as follows

$$\mathrm{KL}(p||q) = \int p(x) \log p(x)dx - \int p(x) \log q(x)dx. \tag{1.4}$$

The KL divergence plays a key role in probabilistic modeling. Given a set of training data $\mathscr{X}_{train} = \{x_1, \cdots, x_N\} \sim \mathbb{P}_d$, we can fit a model by minimizing the KL divergence between the data distribution $\mathbb{P}_d$ and the model $\mathbb{Q}_\theta$ to learn the model parameter θ:

$$\mathrm{D}(\mathbb{P}_d||\mathbb{Q}_\theta) \equiv \mathrm{KL}(p_d||q_\theta) = \int \log p_d(x) p_d(x)dx - \int \log q_\theta(x) p_d(x)dx, \tag{1.5}$$

where the first term in the KL divergence $\int \log p_d(x) p_d(x)dx$ is a constant and does not depend on the model parameters. The second cross-entropy in the KL divergence can be approximated by a Monte-Carlo estimation using $\mathscr{X}_{train}$:

$$\mathscr{L}^N(\theta) \equiv \frac{1}{N} \sum_{n=1}^{N} \log q_\theta(x_n), \tag{1.6}$$

which is also commonly referred to as Maximum Likelihood Estimation (MLE). When $N \to \infty$, we have

$$\mathscr{L}^N(\theta) \xrightarrow{N\to\infty} \int \log q_\theta(x) p_d(x)dx. \tag{1.7}$$

[4]We use *a.e.* to represent 'almost everywhere', see Dempster et al. [2019, p. 18] for a definition.

Therefore, learning a model by maximizing the likelihood can be viewed as minimizing the KL divergence between the model and the data distribution. We then introduce how to use KL divergence to train autoregressive and latent variable models.

Autoregressive Model

We assume each data point is a D-dimensional vector $x = [x_{[1]}, \cdots x_{[D]}]$, an autoregressive model can be written as

$$q_\theta(x) = q_\theta(x_{[1]}) \prod_{d=2}^{D} q_\theta(x_{[d]}|x_{[1:d-1]}), \tag{1.8}$$

where the conditional distribution $q_\theta(x_{[d]}|x_{[1:d-1]})$ can be parameterized by a deep neural network, see Figure 1.1a for a graphical model illustration. Learning θ with MLE is straightforward since the log-likelihood has a closed form

$$\log q_\theta(x) = \log q_\theta(x_{[1]}) + \sum_{d=2}^{D} \log q_\theta(x_{[d]}|x_{[1:d-1]}). \tag{1.9}$$

Deep autoregressive models have achieved great success in modeling real-world data like images [Van Oord et al., 2016, Salimans et al., 2017], audio [Oord et al., 2016] or text [Radford et al., 2019, Brown et al., 2020].

Latent Variable Model

A latent variable model assumes the existence of an unobserved latent variable z that captures the correlations between the different dimensions of the data x. Therefore, given the latent variable, each dimension $x_{[d]}$ is conditionally independent of the other dimensions. This property allows the latent variable model to be expressed as:

$$q_\theta(x) = \int p(z) q_\theta(x|z) dz = \int p(z) \prod_{d=1}^{D} q_\theta(x_{[d]}|z) dz. \tag{1.10}$$

See Figure 1.1b for the conditional independence structure of the latent variable model. The most simple latent variable model is the probabilistic principal component analysis (PPCA) [Tipping and Bishop, 1999], where the prior is a standard

Gaussian $p(z) = \mathcal{N}(0, I_z)$ and the observational distribution $q_\theta(x|z) = \mathcal{N}(Az, \sigma_x^2 I_x)$ is also a Gaussian whose mean is a linear transformation of z. In this simple case, the marginal distribution $q_\theta(x)$ is also a Gaussian $q_\theta(x) = \mathcal{N}(0, AA^T + \sigma_x^2 I_x)$ with analytical likelihood evaluation, so the parameters can be directly learned by MLE. However, for a latent variable model with a non-linear parameterization of $p_\theta(x|z)$ (e.g. a deep neural network), the log-likelihood involves an intractable integration

$$\log q_\theta(x) = \log \int q_\theta(x|z)p(z)dz \tag{1.11}$$

In this case, instead of optimizing the log-likelihood, a lower bound of the log-likelihood can be used to train the model, which is also referred to as the evidence lower bound (ELBO)

$$\begin{aligned} \log q_\theta(x) &\geq \int q_{\phi(z|x)} \log q_\theta(x|z)dz - \mathrm{KL}(q_\phi(z|x)||p(z)) \\ &\equiv \mathrm{ELBO}(x, \theta, \phi), \end{aligned} \tag{1.12}$$

where the amortized variational posterior or $q_\phi(z|x)$ is introduced to approximate the true posterior $p_\theta(z|x) \propto q_\theta(x|z)p(z)$. The ELBO can also be written as

$$\mathrm{ELBO}(x, \theta, \phi) = \log q_\theta(x) - \mathrm{KL}(q_\phi(z|x)||p_\theta(z|x)). \tag{1.13}$$

Therefore, maximizing the ELBO is equivalent to simultaneously maximizing the log-likelihood and minimizing the KL divergence between the amortized posterior and the true posterior. When the amortized posterior equals the true posterior $q_\phi(z|x) = p_\theta(z|x)$, the ELBO becomes equal to the log-likelihood. This type of latent variable model, trained with amortized inference, is also known as the Variational Auto-Encoder (VAE) [Kingma and Welling, 2014, Rezende et al., 2014].

If the log-likelihood or ELBO of an *a.c.* model $q_\theta(x)$ is not tractable, but we can access $\nabla_x \log q_\theta(x)$, an alternative divergence - the Fisher divergence, can be used to train the model, we will provide a brief introduction in the next section.

1.1.2 Fisher Divergence and Score Matching

Given two *a.c.* 1D distributions $\mathbb{P}$ and $\mathbb{Q}$ with differentiable densities $p(x)$ and $q(x)$ with support $\mathbb{R}$, the Fisher divergence [Johnson, 2004] is defined as

$$\mathrm{FD}(\mathbb{P}||\mathbb{Q}) \equiv \mathrm{FD}(p(x)||q(x)) = \frac{1}{2}\int p(x)||s_p(x) - s_q(x)||_2^2 dx, \tag{1.14}$$

where we denote by $s_p(x) \equiv \nabla_x \log p(x)$ and $s_q(x) \equiv \nabla_x \log q(x)$ the score functions of p and q respectively. For probabilistic modeling tasks with data density p_d and model density q_θ, the score function of $s_{p_d} \equiv \nabla_x \log p_d(x)$ is unknown. We can further assume q_θ is twice differentiable and rewrite the FD using the following score-matching formulation [Aapo, 2005]

$$\begin{aligned}\mathrm{FD}(p_d(x)||q_\theta(x)) &= \frac{1}{2}\int p_d(x)||s_{p_d}(x) - s_{q_\theta(x)}||_2^2 dx &(1.15)\\ &\doteq \frac{1}{2}\int p_d(x)\left(s_{q_\theta}^2(x) - 2s_{p_d}(x)s_{q_\theta}(x)\right)dx, &(1.16)\end{aligned}$$

where we use $\doteq$ to denote the equivalence up to a constant term that is independent of θ. Using the identity $p_d(x)\nabla_x \log p_d(x) = \nabla_x p_d(x)$, we have

$$\int p_d(x)s_{p_d}(x)s_{p_\theta}(x)dx = \int \nabla_x p_d(x)s_{q_\theta}(x)dx. \tag{1.17}$$

For $p_d(x)s_{p_\theta}(x)$ vanishes at $-\infty$ and ∞, using integration by parts, we have

$$\int \nabla_x p_d(x)s_{q_\theta}(x)dx = \underbrace{p_d(x)s_{q_\theta}(x)\Big|_{-\infty}^{+\infty}}_{=0} - \int p_d(x)\nabla_x s_{q_\theta}(x)dx. \tag{1.18}$$

Therefore, minimizing the FD is equivalent to minimize

$$\mathrm{FD}(p_d(x)||q_\theta(x)) \doteq \frac{1}{2}\int p_d(x)\left(s_{q_\theta}^2(x) + 2\nabla_x s_{q_\theta}(x)\right)dx \tag{1.19}$$

A similar form can also be derived for D-dimensional distributions [Aapo, 2005]

$$\mathrm{FD}(p_d(x)||q_\theta(x)) \doteq \frac{1}{2}\int p_d(x)\left(\left\|s_{q_\theta}(x)\right\|_2^2 + 2\mathrm{Tr}(\nabla_x s_{q_\theta}(x))\right)dx, \tag{1.20}$$

where $\mathrm{Tr}(\cdot)$ denote the trace operator. In both cases, the integration over p_d can be approximated by Monte-Carlo integration with the given training data$\{x_1, \cdots, x_N\} \sim p_d(x)$. The Fisher divergence only depends on the score function, making it a useful tool for training models with intractable likelihood evaluations, such as energy-based models, which we will discuss below.

Energy-Based Model

An energy-based model [LeCun et al., 2006] usually has the following form

$$q_\theta(x) = e^{-f_\theta(x)}/Z(\theta), \quad \text{where} \quad Z(\theta) = \int e^{-f_\theta(x)} dx. \tag{1.21}$$

For the MLE training, the evaluation of the log-likelihood

$$\log q_\theta(x) = -f_\theta(x) - \log Z(\theta) \tag{1.22}$$

requires the normalizer $Z(\theta)$, which is usually intractable and requires approximation for a nonlinear f_θ. There are many methods proposed to train the energy-based model, we recommend [Song and Kingma, 2021] for a detailed overview of different training methods. One of the neat training criteria is the Fisher divergence objective described in Equation 1.19. This objective only requires the evaluations of

$$s_{q_\theta}(x) = -\nabla_x f_\theta(x), \quad \nabla_x s_{q_\theta}(x) = -\mathrm{Tr}(\nabla_x^2 f_\theta(x)), \tag{1.23}$$

which are tractable and independent of the normalizer $Z(\theta)$.

1.2 Research Motivations and Thesis Structure

Machine learning research has heavily revolved around model estimation and generalization. While training generative models are extensively studied, there remain specific failure modes that require exploration and refinement. Additionally, while generalization in classification tasks directly affects test accuracy, its definition and implications in generative models are not explicitly clear. This thesis is structured to address these gaps in two parts, we will start by introducing the research motivations

of each part below.

Part 1: Healing the Divergences for Training Generative Models

We initiated our study by examining the training dynamics of latent variable models using KL divergence. A significant limitation of the KL divergence (or ELBO) arises when the distribution is supported on a lower-dimensional manifold and doesn't possess a valid density function. For instance, let's consider a latent variable model, represented as $\mathbb{Q}_\theta$, with its density function having the density function represented by $q_\theta(x) = \int q(z)q_\theta(x|z)dz$. In situations where $\text{Dim}(z) < \text{Dim}(x)$, and upon substituting $q_\theta(x|z)$ with a delta function $\delta(x - g_\theta(z))$, the model $\mathbb{Q}_\theta$ is no longer $a.c.$ and lacks a density function [Arjovsky et al., 2017], making the the KL divergence and the MLE undefined. This model is also known as the implicit latent variable model [Ravuri et al., 2018], serving as the foundational model for the renowned Generative Adversarial Net (GAN) [Goodfellow et al., 2014]. Consequently, there's a paradigm shift towards utilizing the Wasserstein distance [Arjovsky et al., 2017, Gulrajani et al., 2017a] to train such implicit models, primarily because of its applicability to non-a.c. distributions. Rather than overhauling the entire training process, our goal is to refine the conventional KL divergence and the ELBO objective, ensuring their applicability for training implicit models to fit non-$a.c$ distributions.

Moreover, recent findings suggest that the Fisher divergence exhibits a "blindness problem" in real-world applications where it fails to differentiate between two mixture distributions with different mixture proportions [Wenliang and Kanagawa, 2020]. This motivates us to study the underlying properties of the Fisher divergence and find solutions that can address this issue.

Part 2: Improve the Generalization of the Generative Models

In the second part of the thesis, we focus on the generalization properties of probabilistic models. In this thesis, the generalization of probabilistic models is defined as the log-likelihood evaluated on the test dataset. This definition has a practical implication for lossless compression: given a trained model, the negative log2 likelihood evaluated on this model is approximately equal to the compression length when using the same model for data compression.

Given the practical significance of generative model generalization, we explore factors influencing generalization and strategies to enhance it. Specifically, our study entails:

1. *In-distribution (ID) generalization:* This involves scenarios where training and test data hail from the same distribution and is the common assumption of the model-based compression. We pay special attention to VAE models, renowned for their practical utility in lossless compression [Hinton and Van Camp, 1993, Townsend et al., 2019].

2. *Out-of-distribution (OOD) generalization:* Here, the training and test datasets might originate from distinct distributions. This scenario also appears in practical compression use cases in the deployment of the compression algorithm in the real world, the distribution of the target data is usually unknown.

By understanding these generalization attributes, we aim to design innovative models and training methodologies that bolster lossless compression performance without compromising on compression speed.

Finally, we delve into applications for Out-of-Distribution (OOD) detection. In likelihood-based OOD detection, models trained on in-distribution datasets assess test data. If the likelihood of this data falls below a certain threshold, it's classified as OOD. Unlike the aim of enhancing OOD generalization in lossless compression, our objective is to diminish OOD generalization in OOD detection, enabling models to display lower likelihoods for OOD data, which in turn enhances detection performance.

Summary

While at first glance the two parts of this thesis might appear as distinct domains of investigation, they are, in fact, intertwined in their core objectives of advancing generative models. The training and effective estimation of these models, as discussed in Part 1, lay the foundation for enhancing their generalization abilities, which is the focus of Part 2. Without robust and efficient training methodologies, any pursuit to enhance generalization would be inherently limited.

One of the key takeaways is the criticality of understanding and addressing the limitations of existing methodologies. By exploring the challenges of conventional divergences and their inadequacy in certain scenarios, we were able to propose a simple fix solution to the current method rather than completely shift away from the conventional approach. Similarly, by uncovering the intricacies of generalization in generative models, especially in the context of lossless compression and OOD detection, we recognize the vast landscape of applications and implications these models possess.

In essence, this thesis not only contributes specific techniques and insights to individual challenges but also fosters an integrated perspective on the development and application of generative models. As the field of machine learning continues to evolve, such a dual focus on foundational training and application-oriented generalization will be indispensable.

Part I

Healing the Statistical Divergences

Chapter 2

Healing the KL Divergence for Manifold Modelling

Using an f-divergence $\mathrm{D}_f(\mathbb{P}||\mathbb{Q})$ for model training requires (i) $\mathbb{P}$ and $\mathbb{Q}$ to have valid probability densities p, q and (ii) p and q to have common support. However, these requirements are not satisfied in some important machine learning applications like modeling manifold data with implicit models. To heal this problem, we propose the *Spread Divergence* $\widetilde{\mathrm{D}}_f(\mathbb{P}||\mathbb{Q})$ for distributions $\mathbb{P}, \mathbb{Q}$ and describe sufficient conditions for the existence of such a divergence. We also demonstrate how to maximize the discriminatory power of a given divergence by parameterizing and learning the spread. We apply the proposed spread divergence to train both linear and non-linear implicit generative models and demonstrate the proposed divergence can significantly improve the training stability and sample generation quality.

2.1 Implicit Models for Manifold Modelling

For many datasets of interest, *e.g.*natural images, the data distribution $\mathbb{P}_d$ is commonly believed to be supported on a lower dimensional manifold [Beymer and Poggio, 1996]. We assume a data sample $x \sim \mathbb{P}_d$ to be a D_X dimensional vector $x \in \mathbb{R}^{D_X}$ and define the ambient dimensionality of $\mathbb{P}_d$, denoted by $\texttt{Amdim}(\mathbb{P}_d)$. We also define the intrinsic dimension of $\mathbb{P}_d$, denoted by $\texttt{Indim}(\mathbb{P}_d)$, to be the dimension of this manifold. Figure 2.1 shows an example of a manifold data distribution $\mathbb{P}_d$ that has $\texttt{Amdim}(\mathbb{P}_d) = 3$ and $\texttt{Indim}(\mathbb{P}_d) = 2$. In this case, $\mathbb{P}_d$ is not $a.c.$ (w.r.t.

Lebesgue measure) and doesn't allow a density function.

To model the manifold data distribution $\mathbb{P}_d$, we requires the model $\mathbb{Q}_\theta$ to able to learn to have $\texttt{Indim}(\mathbb{Q}_\theta) = \texttt{Indim}(\mathbb{P}_d)$. However, consider a classic latent variable model with the model density

$$q_\theta(x) = \int q_\theta(x|z)p(z)\,\mathrm{d}z, \tag{2.1}$$

where $q_\theta(x|z)$ is an a.c. Gaussian distribution with a latent dimension of $\mathrm{Dim}(z) < \mathrm{Dim}(x)$. In this case, the intrinsic dimension of the model is $\texttt{Indim}(\mathbb{Q}_\theta) = \texttt{Indim}(q_\theta(x|z)) = \mathrm{Dim}(x)$. Therefore, the classic latent variable model cannot converge to $\mathbb{P}_d$ in principle. Alternatively, the implicit model [Ravuri et al., 2018, Goodfellow et al., 2014, Arjovsky et al., 2017] can be used to model the manifold distributions, which takes the form of a latent variable model

$$\mathbb{Q}_\theta(\mathrm{d}x) = \int \delta\left(x - g_\theta(z)\right) p(z)\,\mathrm{d}z, \tag{2.2}$$

where $\delta(x)$ is the Dirac delta function. To generate a sample from $\mathbb{Q}_\theta$, one can first sample $z' \sim p(z)$ and then generate a sample $x' = g_\theta(z')$. In the common setting where the latent dimension $\mathrm{Dim}(z)$ is lower than the observation dimension $\mathrm{Dim}(x)$: $\mathrm{Dim}(z) < \mathrm{Dim}(x)$, the model $\mathbb{Q}_\theta$ has an intrinsic dimension $\texttt{Indim}(\mathbb{Q}_\theta) \leq \mathrm{Dim}(z)$ [Arjovsky et al., 2017], where the equality is achieved when g_θ is an invertible function. Therefore, if we use a non-invertible function like a neural network and a suitable latent space (e.g. $\mathrm{Dim}(z) \geq \texttt{Indim}(\mathbb{P}_d)$), the model $\mathbb{Q}_\theta$ is flexible to learn have $\texttt{Indim}(\mathbb{Q}_\theta) = \texttt{Indim}(\mathbb{P}_d)$.

We can also generalize the implicit model to accommodate degenerated Gaussian observation noise, represented as:

$$\mathbb{Q}_\theta(\mathrm{d}x) = \int \mathcal{N}_g(g_\theta(z), \Sigma_\theta(z))p(z)\,\mathrm{d}z, \tag{2.3}$$

Here, we use $\mathcal{N}_g$ to denote the 'generalized' Gaussian distributions. These encompass the special case of a degenerated Gaussian covariance function. That is,

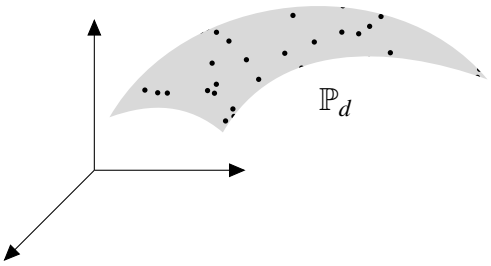


Figure 2.1: This figure shows a data distribution $\mathbb{P}_d$ that is supported on a 2d manifold in a 3d space. In this case, the $\mathbb{P}_d$ is not *a.c.* and doesn't have a density function.

when $\Sigma_\theta(z)$ has $\text{rank}(\Sigma) < \text{Dim}(x)$, the $\mathbb{Q}_\theta$ is not absolutely continuous and can flexibly have any arbitrary intrinsic dimension. This is what we refer to $\mathbb{Q}_\theta$ as the 'generalized implicit model'.

However, in both cases, gradient-based maximum likelihood learning is problematic since $\mathscr{L}(\theta)$ is typically not a continuous function of θ[1]. Furthermore, the Expectation Maximisation (EM) algorithm [Dempster et al., 1977] is not available for models of the form in Equation 2.2 since EM assumes that $\log q(x|z)$ is well defined, which is not the case for singular conditional distribution $\mathbb{Q}_{X|Z}(z, \mathrm{d}x) = \delta(x - g(z))$[Bermond and Cardoso, 1999]. Equally, in Equation 1.4 the ratio $p(x)/q(x)$ may represent a division by zero; the KL divergence between the model and the data-generating process is thus ill-defined.

2.1.1 Model Noise is Not Enough

A common approach to enable maximum likelihood to be used to train implicit generative models is to simply add noise to the model so that it has full support (and a valid density), see for example [Wu et al., 2016]. However, this approach does not guarantee a consistent estimator. To see this, consider the simple implicit generative model $\mathbb{Q}$

$$\mathbb{Q}_\theta(\mathrm{d}x) = \int \delta\left(x - z\theta_q\right) \mathscr{N}(z|0,1)\, \mathrm{d}z, \tag{2.4}$$

[1]For a point x_n that is not on the model manifold, $q(x_n) = 0$. As we adjust θ such that x_n becomes on the manifold, $q(x_n)$ will typically increase to a finite non-zero value, meaning that the (log) likelihood is discontinuous in θ.

where the latent Z is univariate and $\mathrm{Dim}(X) > 1$. Here the vector θ_q defines a one-dimensional line in the X space. For D-dimensional X, adding independent Gaussian noise with mean zero and isotropic covariance $\sigma^2 I_D$ to X results in the noised distribution with density

$$\tilde{q}_\theta(x) = \mathcal{N}(x|0_D, \Sigma), \quad \Sigma \equiv \theta_q \theta_q^\mathsf{T} + \sigma^2 I_D. \tag{2.5}$$

For observed training data $x_1, \ldots, x_N$ the log-likelihood under this model is

$$\mathcal{L}(\theta_q) = -\frac{1}{N} \sum_{n=1}^{N} x_n^\mathsf{T} \Sigma^{-1} x_n - \log\det\Sigma + const. \tag{2.6}$$

We assume that the training data x_n is iid sampled from the distribution $\mathbb{P}$ with generalized density

$$p_d(x) = \int \delta(x - z\theta_p)\, \mathcal{N}(z|0,1)\, \mathrm{d}z. \tag{2.7}$$

Hence $\mathbb{P}$ and $\mathbb{Q}$ are from the same parametric distribution but with different parameters. By the law of large numbers, in the large N limit, the log-likelihood Equation 2.6 tends to

$$-\theta_p^\mathsf{T} \Sigma^{-1} \theta_p - \log\det\Sigma + const. \tag{2.8}$$

which has an optimum when[2] (see Appendix A.1)

$$\theta_q = \sqrt{\frac{\theta_p^2 - \sigma^2}{\theta_p^2}}\, \theta_p. \tag{2.9}$$

Thus adding noise to the model $\mathbb{Q}$ and training using maximum likelihood does not form a consistent estimator; it has an optimum at $\theta_q \neq \theta_p$, resulting in an incorrect estimate of the data generating process. In Appendix A.1 we explain why annealing the noise σ^2 towards zero during numerical optimization will not directly heal this problem.

[2] θ_p^2 is shorthand for the squared length $\theta_p^\mathsf{T} \theta_p = ||\theta_p||_2^2$.

For this reason, alternative (non-likelihood, non-KL) approaches to measure the difference between distributions are commonly used in training implicit generative models (see for example [Mohamed and Lakshminarayanan, 2016]), such as Maximum Mean Discrepancy (MMD) [Gretton et al., 2012] and Wasserstein distance [Arjovsky et al., 2017, Peyré and Cuturi, 2019]. In the next section, we introduce the Spread Divergence, which is well-defined when the distributions do not allow a valid density or the supports of the distributions do not match. As we will demonstrate, the Spread Divergence allows one to use maximum likelihood-based approaches to train implicit generative models, whilst resulting in a consistent estimator.

2.2 Spread Divergence

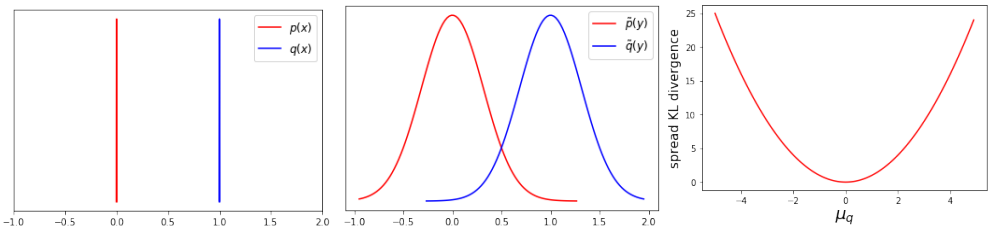
For distributions $\mathbb{Q}$ and $\mathbb{P}$ with generalised densities $q(x)$ and $p(x)$ we first need to define $\tilde{q}(y)$ and $\tilde{p}(y)$ that (i) are valid probability density functions and (ii) have the same support. In contrast to simply noising $\mathbb{Q}$ we define 'noisy' densities for both distributions

$$\tilde{p}(y) = \int p(y|x)p(x)\,\mathrm{d}x, \quad \tilde{q}(y) = \int p(y|x)q(x)\,\mathrm{d}x \tag{2.10}$$

The 'noise' $p(y|x)$ must 'spread' $\mathbb{P}$ and $\mathbb{Q}$ such that $\tilde{p}(y)$ and $\tilde{q}(y)$ satisfy the above two reauirements. The choice of $p(y|x)$ must also ensure that $\mathrm{D}(\tilde{p}||\tilde{q}) = 0 \Leftrightarrow \mathbb{P} = \mathbb{Q}$. If we can define the noise appropriately, this would allow us to define the Spread Divergence

$$\widetilde{\mathrm{D}}(\mathbb{P}||\mathbb{Q}) \equiv \mathrm{D}(\tilde{p}||\tilde{q})\,, \tag{2.11}$$

which satisfies the divergence requirement $\widetilde{\mathrm{D}}(\mathbb{P}||\mathbb{Q}) \geq 0$ and $\widetilde{\mathrm{D}}(\mathbb{P}||\mathbb{Q}) = 0 \Leftrightarrow \mathbb{P} = \mathbb{Q}$. In the following we discuss appropriate choices for the noise distribution $p(y|x)$. We focus on stationary spread noise $p(y|x) = k(y-x)$ since this is simple to implement by adding independent noise to a variable. Non-stationary spread distributions can be easily constructed using a mixture of stationary noise distributions, or through Mercer kernels – these are left for future study. The case of discrete X is discussed



(a) Two delta distributions. (b) Spreaded distributions. (c) Spread KL divergence

Figure 2.2: (a) Delta distributions $p(x) = \delta(x-\mu_p)$, $q(x) = \delta(x-\mu_q)$ where $\mu_p = 0$, $\mu_q = 1$. (b) Spreaded distributions $\tilde{p}(y) = \int p(y|x)p(x)\,\mathrm{d}x$, $\tilde{q}(y) = \int p(y|x)q(x)\,\mathrm{d}x$, where $p(y|x) = \mathcal{N}\left(y|x, \sigma^2 = 0.5\right)$. (c) Spread KL divergence as a function of μ_q.

in Appendix A.2.

2.2.1 Stationary Spread Divergence

Given two random variables $X_{\mathbb{Q}}$ and $X_{\mathbb{P}}$ with distributions $\mathbb{Q}$ and $\mathbb{P}$, respectively. Let K be an *a.c.* random variable that is independent of $X_{\mathbb{Q}}$ and $X_{\mathbb{P}}$ and has density $k(\cdot)$. We define

$$Y_{\tilde{\mathbb{P}}} = X_{\mathbb{P}} + K, \quad Y_{\tilde{\mathbb{Q}}} = X_{\mathbb{Q}} + K, \tag{2.12}$$

with distributions $\tilde{\mathbb{P}}$ and $\tilde{\mathbb{Q}}$ respectively. Then $\tilde{\mathbb{P}}$ and $\tilde{\mathbb{Q}}$ are *a.c.* with density functions

$$\tilde{p}(y) = \int_x k(y-x)d\mathbb{P}, \quad \tilde{q}(y) = \int_x k(y-x)d\mathbb{Q}, \tag{2.13}$$

see Theorem 2.1.16 in [Dempster et al., 2019] for a proof. We then define the stationary spread f-divergence between $\mathbb{P}$ and $\mathbb{Q}$ as

$$\widetilde{\mathrm{D}}_f(\mathbb{P}||\mathbb{Q}) \equiv \mathrm{D}_f\left(\tilde{\mathbb{P}}||\tilde{\mathbb{Q}}\right) \equiv \int f\left(\frac{\tilde{p}(y)}{\tilde{q}(y)}\right)\tilde{q}(y)\,\mathrm{d}x. \tag{2.14}$$

Theorem 1 (Validity of the Spread Divergence)**.** *Let $X_{\mathbb{P}}$ and $X_{\mathbb{Q}}$ be two random variables with Borel probability measure $\mathbb{P}$ and $\mathbb{Q}$. Let the stationary spread noise K be an a.c. random variable that is independent of $X_{\mathbb{P}}$ and $X_{\mathbb{Q}}$ and has support $\mathbb{R}^d$. We further denote the characteristic function of K as ϕ_K. Given $\phi_K \neq 0$ or $\phi_K > 0$ on at most a countable set, then the stationary spread f-divergence is a valid divergence*

with the properties

$$\widetilde{D}_f(\mathbb{P}||\mathbb{Q}) \geq 0, \quad \widetilde{D}_f(\mathbb{P}||\mathbb{Q}) = 0 \Leftrightarrow \mathbb{P} = \mathbb{Q}. \tag{2.15}$$

Proof. The proof contains the following two steps.

First step: We show that if K is $a.c.$ with support $\mathbb{R}^d$, then $\widetilde{\mathrm{D}}_f(\mathbb{P}||\mathbb{Q}) = 0 \Leftrightarrow \widetilde{\mathbb{P}} = \widetilde{\mathbb{Q}}$. Since $Y_{\tilde{\mathbb{P}}}$ and $Y_{\tilde{\mathbb{Q}}}$ are $a.c.$ and allow density functions $\tilde{p}(y)$ and $\tilde{q}(y)$. Since p_K has support $\mathbb{R}$, $\tilde{p}(y)$ and $\tilde{q}(y)$ will also have support $\mathbb{R}^d$. The f-divergence between two $a.c.$ distributions with common support is equal to zero if and only if two distributions are equal [Csiszár, 1967, 1972]. We have $\mathrm{D}_f(\tilde{p}(y)||\tilde{q}(y)) = 0 \Leftrightarrow \mathbb{P} = \mathbb{Q}$. Therefore,

$$\widetilde{\mathrm{D}}_f(\mathbb{P}||\mathbb{Q}) = 0 \Leftrightarrow \widetilde{\mathbb{P}} = \widetilde{\mathbb{Q}}.$$

Second step: We show that if the characteristic function of the spread noise $\phi_K \neq 0$ or $\phi_K = 0$ on at most a countable set then $\widetilde{\mathbb{P}} = \widetilde{\mathbb{Q}} \Leftrightarrow \mathbb{P} = \mathbb{Q}$.

The characteristic function of a probability measure $\mathbb{P}$ is defined as $\phi_{\mathbb{P}}(w) = \int e^{iwx} d\mathbb{P}$. Since a probability measure is uniquely determined by its characteristic function [Kallenberg, 2006, Theorem 4.3], we have

$$\widetilde{\mathbb{P}} = \widetilde{\mathbb{Q}} \Leftrightarrow \phi_{\widetilde{\mathbb{P}}} = \phi_{\widetilde{\mathbb{Q}}}.$$

Using the fact that the characteristic function of the sum of two random variables is equal to the product of their characteristic functions [Dempster et al., 2019, Theorem 3.3.2], we can write

$$\phi_{\widetilde{\mathbb{P}}} = \phi_{\widetilde{\mathbb{Q}}} \Leftrightarrow \phi_{\mathbb{P}}\phi_K = \phi_{\mathbb{Q}}\phi_K.$$

When $\phi_K \neq 0$, we have $\phi_{\mathbb{P}}\phi_K = \phi_{\mathbb{Q}}\phi_K \Leftrightarrow \phi_{\mathbb{P}} = \phi_{\mathbb{Q}}$.

When $\phi_K = 0$ on at most a countable set $\mathscr{C}$, we show that $\phi_{\mathbb{P}}\phi_K = \phi_{\mathbb{Q}}\phi_K \Leftrightarrow \phi_{\mathbb{P}} = \phi_{\mathbb{Q}}$ still holds. We prove this by contradiction: We assume there is a point $w_0 \in \mathscr{C}$ where $\phi_{\mathbb{P}}(w_0) \neq \phi_{\mathbb{Q}}(w_0)$. Without loss of generality, we assume $\phi_{\mathbb{P}}(w_0) - \phi_{\mathbb{Q}}(w_0) = \delta > 0$. For points $w_0 + h$ that are not in $\mathscr{C}$, we have $\phi_K(w_0 + h) \neq 0$, so $\phi_{\mathbb{P}}\phi_K = \phi_{\mathbb{Q}}\phi_K$ implies $\phi_{\mathbb{P}}(w_0 + h) - \phi_{\mathbb{Q}}(w_0 + h) = 0$. Since the characteristic

function of a distribution is uniform continuous [Dempster et al., 2019, Theorem 3.3.1], we have $\delta = \phi_{\mathbb{P}}(w_0 + h) - \phi_{\mathbb{Q}}(w_0 + h) \to 0$ when $h \to 0$, which leads to a contradiction (since δ cannot be zero). Therefore, $\phi_{\mathbb{P}}\phi_K = \phi_{\mathbb{Q}}\phi_K \Leftrightarrow \phi_{\mathbb{P}} = \phi_{\mathbb{Q}}$. By the uniqueness of the characteristic function [Kallenberg, 2006, Theorem 4.3], we have

$$\phi_{\mathbb{P}} = \phi_{\mathbb{Q}} \Leftrightarrow \mathbb{P} = \mathbb{Q}.$$

Using the results of the two steps, we can conclude

$$\widetilde{\mathrm{D}}_f(\mathbb{P}||\mathbb{Q}) = 0 \Leftrightarrow \widetilde{\mathbb{P}} = \widetilde{\mathbb{Q}} \Leftrightarrow \mathbb{P} = \mathbb{Q}.$$

□

Since the spread noise K is $a.c$, its characteristic function is equivalent to the Fourier transform of the density function. Therefore, a set of simplified sufficient conditions of a valid spread noise is (1) K has density function $k(x)$ with support $\mathbb{R}^d$ and (2) the Fourier transform of the $k(x)$ is positive. As an example, consider Gaussian additive spread noise $k(x) = \mathcal{N}\left(x|0, \sigma^2\right)$, its Fourier transform is positive

$$\mathscr{F}\{k\}(\omega) = e^{-\frac{\sigma^2\omega^2}{2}} > 0. \tag{2.16}$$

Similarly, for Laplace noise $k(x) = \frac{1}{2b}e^{-\frac{1}{b}|x|}$,

$$\mathscr{F}\{k\}(\omega) = \sqrt{\frac{2}{\pi}}\frac{b^{-1}}{b^{-2} + \omega^2} > 0. \tag{2.17}$$

In both cases $k(x)>0$ for $x \in \mathbb{R}^d$ and $\mathscr{F}\{k\}>0$. Therefore, additive Gaussian and Laplace noise can be used to define a valid Spread Divergence. This non-zero requirement for the characteristic function is analogous to the characteristic condition on kernels such that the Maximum Mean Discrepancy $\mathrm{MMD}(\mathbb{P}, \mathbb{Q}) = 0 \Leftrightarrow \mathbb{P} = \mathbb{Q}$, see [Sriperumbudur et al., 2011, 2012, Gretton et al., 2012]. As an illustration,

consider the extreme case of two delta distributions $\mathbb{P}$ and $\mathbb{Q}$

$$\mathbb{P}(\mathrm{d}x) = \delta(x - \mu_p), \quad \mathbb{Q}(\mathrm{d}x) = \delta(x - \mu_q). \tag{2.18}$$

In this case $\mathrm{KL}(\mathbb{P}||\mathbb{Q})$ is not well defined. For stationary Gaussian noise $k(x) = \mathcal{N}\left(x|0, \sigma^2\right)$, we can equivalent define a conditional distribution $p(y|x) = \mathcal{N}\left(y|x, \sigma^2\right)$, so the 'spreaded' distributions $\tilde{\mathbb{P}}$ and $\tilde{\mathbb{Q}}$ are *a.c.* and have densities

$$\tilde{p}(y) = \int \delta(x - \mu_p)\mathcal{N}\left(y|x, \sigma^2\right) \mathrm{d}x = \mathcal{N}\left(y|\mu_p, \sigma^2\right), \tag{2.19}$$

$$\tilde{q}(y) = \int \delta(x - \mu_q)\mathcal{N}\left(y|x, \sigma^2\right) \mathrm{d}x = \mathcal{N}\left(y|\mu_q, \sigma^2\right). \tag{2.20}$$

For noise variance $\sigma^2 = 0.5$ this gives:

$$\widetilde{\mathrm{KL}}(\mathbb{P}||\mathbb{Q}) \equiv \mathrm{KL}(\tilde{p}||\tilde{q}) = ||\mu_p - \mu_q||_2^2. \tag{2.21}$$

Hence $\widetilde{\mathrm{KL}}(\mathbb{P}||\mathbb{Q}) \Leftrightarrow \mathbb{P} = \mathbb{Q}$. It is also worth noting that the spread KL divergence, in this case, is equal to the squared 2-Wasserstein distance [Peyré and Cuturi, 2019, Gelbrich, 1990]. Treating μ_q as a variable, Figure 2.2 illustrates the spread KL divergence converging to 0 as μ_q tends to $\mu_p = 0$.

This proposed spread KL divergence allows us to define a valid divergence between manifold distributions. Therefore, an associated training algorithm can be derived for learning manifold models, as we describe below.

2.2.2 Spread Maximum Likelihood Estimation

In Section 2.1.1 we noted that in the context of fitting an implicit generative model $\mathbb{Q}_\theta$ to training data, simply using MLE will be problematic. We can instead minimize the spread KL divergence between data distribution $\mathbb{P}_d$ and model $\mathbb{Q}_\theta$:

$$\begin{aligned}\widetilde{\mathrm{KL}}(\mathbb{P}_d||\mathbb{Q}_\theta) &\equiv \widetilde{\mathrm{KL}}(\tilde{p}_d(y)||\tilde{q}_\theta(y)) \\ &= \underbrace{\int \log \tilde{p}_d(y)\tilde{p}_d(y)\,\mathrm{d}y}_{const.} - \int \log \tilde{q}_\theta(y)\tilde{p}_d(y)\,\mathrm{d}y.\end{aligned} \tag{2.22}$$

Given training data $x_1, \ldots, x_N$ sampled *i.i.d* from $\mathbb{P}_d$, we can define the *spread Maximum Likelihood Estimation* (spread MLE) as

$$\widetilde{\mathscr{L}}^N(\theta) \equiv \frac{1}{N}\sum_{n=1}^{N}\int p(y|x_n)\log \tilde{q}_\theta(y)\,\mathrm{d}y. \tag{2.23}$$

Using the law of large numbers, $N \to \infty$ (using), we have

$$\widetilde{\mathscr{L}}^N(\theta) \xrightarrow{N\to\infty} \int \log \tilde{q}_\theta(y)\tilde{p}_d(y)\,\mathrm{d}y. \tag{2.24}$$

In this case, the spread MLE has a maximum when the spread KL divergence has a minimum. Hence, the spread MLE defines a consistent estimator.

The Maximum Likelihood Estimator (MLE) is a cherished approach due to its consistency (convergence to the true parameters in the large data limit) and asymptotic efficiency (achieves the Cramér-Rao lower bound on the variance of any unbiased estimator) - see [Casella and Berger, 2021] for an introduction. An interesting question is whether these properties also carry over to the spread MLE. In Appendix A.5, we demonstrate, for a certain family of spread noise, *the spread MLE has weaker sufficient conditions than MLE for both consistency and asymptotic efficiency*. Furthermore, a sufficient condition for the existence of the MLE is that the likelihood function is continuous over a compact parameter space Θ. We provide an example in Appendix A.5.1 where the maximum likelihood estimator may not exist (since it violates the compactness requirement), but the spread maximum likelihood estimator still exists.

2.2.3 Spread Evidence Lower Bound

We are interested in training implicit latent variable model

$$\mathbb{Q}_\theta(\mathrm{d}x) = \int \delta(x - g_\theta(z))p(z)\,\mathrm{d}z, \tag{2.25}$$

using the proposed Spread MLE. Therefore, the spreaded model $\tilde{q}_\theta(y)$ with Gaussian $p(y|x) = \mathcal{N}(y|x, \sigma^2 I)$ has the form

$$\tilde{q}_\theta(y) = \int\int p(y|x)\delta(x - g_\theta(z))p(z)\,\mathrm{d}z\,\mathrm{d}x \tag{2.26}$$

$$= \int \underbrace{\mathcal{N}(y|g_\theta(z), \sigma^2 I)}_{q_\theta(y|z)} p(z)\,\mathrm{d}z. \tag{2.27}$$

A similar form can be derived for an implicit model with degenerate Gaussian noise since $p(y|x)$ will be $a.c.$ Gaussian distribution.

For a linear parameterization decoder, e.g. $g_\theta(z) = Fz$ with a linear matrice F, the integration over z can be calculated in a closed form, see Section 4.4 for an example. However, for non-linear g_θ, the integration over the latent variable z is usually intractable so we cannot directly evaluate $\log \tilde{q}_\theta(y)$. Similar to the Evidence Lower Bound (ELBO) that is used in the Variational Auto-Encoder(VAE), we can also derive a *spread ELBO* for the spread log-likelihood

$$\log \tilde{q}_\theta(y) \geq \int \log q_\theta(y|z) q_\phi(z|y)\,\mathrm{d}z - \mathrm{KL}\big(q_\phi(z|y)||p(z)\big), \tag{2.28}$$

$$\equiv \widetilde{\mathrm{ELBO}}(y, \theta, \phi). \tag{2.29}$$

where the amortized posterior $q_\phi(z|y)$ is introduced to approximate the true posterior $p_\theta(z|y) \propto p_\theta(y|z)p(z)$ and when $q_\phi(z|y) = p_\theta(z|y)$, the spread ELBO is equal to the spread log-likelihood. We refer to the implicit model with a variational posterior as the 'δ-VAE'. Different from the standard VAE, both the decoder and encoder in the δ-VAE are operating on the noisy sample y rather than the clean data x.

Similarly, the spread ELBO can also be constructed for the generalized implicit model with a generalized Gaussian observational distribution $p_\theta(x|z) = \mathcal{N}_g(g_\theta(z), \Sigma_\theta(z))$, since for Gaussian spread noise $p(y|x) = \mathcal{N}(y|x, \sigma^2 I)$, $p_\theta(y|z)$ can be calculated in a closed-form:

$$p_\theta(y|z) = \int p(y|x)p_\theta(x|z)\,\mathrm{d}z = \mathcal{N}(y|g_\theta(z), \sigma^2 I + \Sigma_\theta(z))\,\mathrm{d}z. \tag{2.30}$$

We thus refer to the generalized implicit model with a variational posterior as the 'Degenerate Gauss-VAE'.

Given N training data sampled from the data distribution, the lower bound of the spread MLE can also be derived

$$\frac{1}{N}\sum_{n=1}^{N}\int p(y|x_n)\log\tilde{q}_\theta(y) \geq \frac{1}{N}\sum_{n=1}^{N}\int p(y|x_n)\widetilde{\mathrm{ELBO}}(y,\theta,\phi)\,\mathrm{d}y. \tag{2.31}$$

In practice we typically cannot carry out the integral in Equation 2.23 exactly and resort to a sample approximation, sampling L noisy versions $y_{n,k}$, $k = 1,\ldots,K$ of each data point x_n to give

$$\frac{1}{N}\sum_{n=1}^{N}\int p(y|x_n)\widetilde{\mathrm{ELBO}}(y,\theta,\phi) \approx \frac{1}{NK}\sum_{n=1}^{N}\sum_{k=1}^{K}\widetilde{\mathrm{ELBO}}(y_{n,k},\theta,\phi). \tag{2.32}$$

A lower simplified objective also exists for the spread ELBO with Gaussian $p(y|x) = \mathcal{N}(x,\sigma^2 I)$. For a single x_n, we notice that $\int p(y|x_n)\widetilde{\mathrm{ELBO}}(y,\theta,\phi)\,\mathrm{d}y$ is equal to

$$\int p(y|x_n)\log q_\theta(y|z)q_\phi(z|y)\,\mathrm{d}z\,\mathrm{d}y - \int p(y|x_n)\mathrm{KL}\big(q_\phi(z|y)||p(z)\big)\,\mathrm{d}y. \tag{2.33}$$

For a Gaussian $p(\varepsilon) = \mathcal{N}(0,I)$, the first term can be further simplified using the reparameterization trick:

$$\begin{aligned} &-\frac{1}{2\sigma^2 I}\int ||x_n+\varepsilon-g_\theta(z)||_2^2 q_\phi(z|x_n+\varepsilon)p(\varepsilon)\,\mathrm{d}z\,\mathrm{d}\varepsilon \\ =&-\frac{1}{2\sigma^2 I}\int ||x_n-g_\theta(z)||_2^2 q_\phi(z|x_n+\varepsilon)p(\varepsilon)\,\mathrm{d}z\,\mathrm{d}\varepsilon + const.. \end{aligned} \tag{2.34}$$

where the constant term has the following form

$$-\frac{1}{2\sigma^2 I}\int \big(||\varepsilon||_2^2 + 2\varepsilon(x_n-g_\theta(z))\big)\,q_\phi(z|x_n+\varepsilon)p(\varepsilon)\,\mathrm{d}z\,\mathrm{d}\varepsilon = -\frac{1}{2\sigma^2}. \tag{2.35}$$

We observe that this simplified objective has less training variance in practice.

We have discussed how to train the model with spread MLE and fixed spread Gaussian noise. In the next section, we discuss how to learn the noise distribution to

maximize the discriminatory power of the spread divergence.

2.3 Comparisons with Other Divergences

Maximum Mean Discrepancy

In machine learning, convolving distributions using kernels is a widely adopted technique. A particularly prominent application of this is the kernel mean embedding [Berlinet and Thomas-Agnan, 2011, Smola et al., 2007], which represents a distribution $\mathbb{P}$ as a mean function,

$$\mu_{\mathbb{P}} = \int_{\mathscr{X}} k(x,\cdot) d\mathbb{P}(x), \tag{2.36}$$

where $k : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ is a symmetric and positive definite kernel function, see also [Muandet et al., 2017] for a review. The kernel mean embedding has also been used to measure the difference between two distributions. For two distributions $\mathbb{P}$ and $\mathbb{Q}$, the maximum mean discrepancy [Borgwardt et al., 2006, Gretton et al., 2012] (MMD) is defined as the distance in an RKHS $\mathscr{H}$

$$\mathrm{MMD}_{\mathscr{H}}(\mathbb{P},\mathbb{Q}) \equiv \|\mu_{\mathbb{P}} - \mu_{\mathbb{Q}}\|_{\mathscr{H}}. \tag{2.37}$$

The MMD has been used in training implicit latent variable model [Li et al., 2015, Dziugaite et al., 2015, Li et al., 2017]. However, the efficacy of MMD is solely reliant on kernel selection, which becomes challenging when dealing with high-dimensional distributions. Therefore, it is necessary to introduce the dimension reduction or adversarial kernel learning techniques even for learning the MNIST dataset [Li et al., 2015, 2017]. As a result, it often necessitates the inclusion of dimension reduction techniques or adversarial kernel learning, even for simpler datasets like MNIST [Li et al., 2015, 2017]. In contrast, the spread KL divergence merges the advantages of kernel convolution with efficient MLE training (or EM training when using the ELBO), streamlining the process of training high-dimensional datasets.

Wasserstein Distance

The Wasserstein distance, originally introduced by Kantorovich [Kantorovich, 1960], is a metric especially suited for the comparison of manifold distributions. In recent years, it has seen remarkable success across various machine learning contexts [Peyré and Cuturi, 2019]. One notable application is its role in the training of the implicit latent variable model, known as the Wasserstein Generative Adversarial Network (WGAN) [Arjovsky et al., 2017]. Within the WGAN framework, the Kantorovich-Rubinstein dual form of the Wasserstein distance [Villani, 2009] is employed, which is represented as:

$$W(\mathbb{P},\mathbb{Q}) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}}[f(x)] - \mathbb{E}_{x\sim\mathbb{Q}}[f(x)]. \tag{2.38}$$

Here, the supremum is taken over all 1-Lipschitz functions $f : \mathbb{X} \to \mathbb{R}$. In practical implementations, a parameterized function, f_w, is adopted. To approximate the Lipschitz condition, techniques like gradient clipping [Arjovsky et al., 2017] or the gradient penalty [Gulrajani et al., 2017a] are often used.

Considering an implicit model $\mathbb{Q}_\theta$ equipped with a prior $p(z)$, a generator $g_\theta(z)$, and a target data distribution $\mathbb{P}_d$, the training objective for $g_\theta(z)$ is articulated as a min-max problem:

$$\min_\theta \max_w \mathbb{E}_{x\sim\mathbb{P}_d}[f_w(x)] - \mathbb{E}_{z\sim p(z)}[f_w(g_\theta(z))]. \tag{2.39}$$

In this equation, the maximum over w serves to approximate the sup operation in Equation 2.38.

Despite the widespread praise for WGANs, various concerns have emerged challenging whether their empirical success can be attributed to the advantageous properties of the Wasserstein distance. One of the key issues is that the f_w isn't optimally trained during each gradient step of the generative model. Therefore, the WGAN is minimizing an approximate lower bound of the true Wasserstein distance, which doesn't necessarily minimize the Wasserstein distance. Additionally, research has demonstrated that the incorporation of gradient penalty terms can enhance GAN

generation regardless of the specific statistical divergence [Kodali et al., 2017, Fedus et al., 2017, Stanczuk et al., 2021] and no singular GAN loss consistently outperforms the others [Lucic et al., 2018]. More recently, paper [Stanczuk et al., 2021] gives both theoretical and empirical evidence to show that the WGAN loss is not a meaningful approximation of the Wasserstein distance and argues that the Wasserstein distance is not a desirable loss function for deep generative models. Thus, the actual effectiveness of the Wasserstein distance in training implicit models remains an open problem.

In our approach, we aim to subtly adjust the traditional KL (ELBO) training method to suit the training of implicit models to fit the manifold distributions, rather than transitioning entirely to a different training paradigm. Moreover, the spread ELBO objective minimizes a definitive upper bound of the spread KL divergence, ensuring the training converges effectively.

2.4 Maximising Discriminatory Power

Intuitively, spreading out distributions makes them more similar. More formally, from the data processing inequality (see Appendix A.3), using spread noise will always decrease the f-divergence $\mathrm{D}_f(\tilde{p}||\tilde{q}) \leq \mathrm{D}_f(p||q)$ (when $\mathrm{D}_f(p||q)$ is well defined). If we use spread MLE to train a model, too much noise may make the spreaded empirical and spreaded model distributions so similar that it becomes difficult to numerically distinguish them. It is useful therefore to learn spread noise $p_\psi(y|x)$ (parameterised by ψ) to maximally discriminate between the distributions $\max_\psi \mathrm{D}(\tilde{p}||\tilde{q})$. In general, we need to constrain the spread noise to ensure that the divergence remains bounded. The learned noise will discourage overlap between the two spreaded distributions.

We discuss below two complementary approaches to adjust $p_\psi(y|x)$. The first adjusts the covariance for Gaussian $p(y|x)$ and the second uses a mean transformation. In principle, both approaches can be combined and easily generalised to other noise distributions, such as the Laplace distribution. In Section 2.5.3, we empirically investigate the benefit of these approaches when scaling the application of Spread

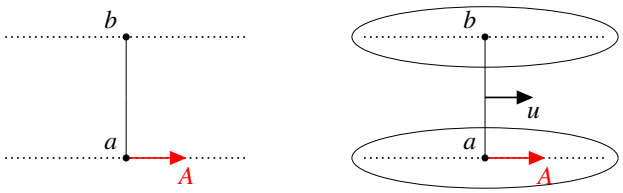


Figure 2.3: Left: The lower dotted line denotes Gaussian distributed data $p(x)$ with support only along the linear subspace defined by the origin a and direction A. The upper dotted line denotes Gaussian distributed data $q(x)$ with support different from $p(x)$. Optimally, to maximise the Spread Divergence between the two distributions, for fixed noise entropy, we should add noise that preferentially spreads out along the directions defined by p and q, as denoted by the ellipses.

Divergence to complex high-dimensional problems.

2.4.1 Learning the Gaussian Noise Covariance

In learning Gaussian stationary spread noise $p(y|x) = \mathcal{N}(y|x,\Sigma)$, the number of parameters in the covariance matrix Σ scales quadratically with the data dimension D. We therefore define $\Sigma = \sigma^2 I + UU^\mathsf{T}$ where $\sigma^2 > 0$ is fixed (to ensure bounded Spread Divergence) and U is a learnable $D \times R$ matrix with $R \ll D$.

As a simple example that can be computed exactly, we consider implicit generative models $\mathbb{P}, \mathbb{Q}$ that generate data in separated linear subspaces,

$$\mathbb{P}(\mathrm{d}x) = \int \delta(x - a - Az)\, p(z)\, \mathrm{d}z, \quad \mathbb{Q}(\mathrm{d}x) = \int \delta(x - b - Bz)\, p(z)\, \mathrm{d}z, \tag{2.40}$$

with $p(z) = \mathcal{N}(z|0, I_Z)$. The spread densities are then

$$\tilde{p}(y) = \mathcal{N}\left(y|a, AA^\mathsf{T} + \Sigma\right), \quad \tilde{q}(y) = \mathcal{N}\left(y|b, BB^\mathsf{T} + \Sigma\right). \tag{2.41}$$

As Σ tends to zero, $\mathrm{KL}(\tilde{p}||\tilde{q})$ tends to infinity. We therefore constrain $\Sigma = \sigma^2 I + uu^\mathsf{T}$, where σ^2 is fixed and $u^\mathsf{T}u = 1$. When $A \neq B$, the optimal μ doesn't have a simple closed-form. Therefore, for calculational simplicity, we assume $A = B$. The Spread Divergence $\mathrm{KL}(\tilde{p}||\tilde{q})$ is then maximised for the noise direction u pointing orthogonal to the vector $\left(AA^\mathsf{T} + \sigma^2 I\right)^{-1}(b - a)$. The noise thus concentrates along directions defined by p and q, see Figure 2.3.

2.4.2 Learning a Mean Transformation

Consider spread noise $p(y|x) = k(y - f(x))$ for injective[3] f and stationary k, e.g. $p(y|x) = \mathcal{N}(y|f(x), \sigma^2 I)$. Then for two distributions $\mathbb{P}$ and $\mathbb{Q}$ that are not $a.c.$, we can define two $a.c.$ distributions $\tilde{\mathbb{P}}$ and $\tilde{\mathbb{Q}}$ with densities

$$\tilde{p}(y) = \int_x k(y - f(x)) d\mathbb{P}, \quad \tilde{q}(y) = \int_x k(y - f(x)) d\mathbb{Q}. \tag{2.42}$$

Therefore, the spread divergence $\widetilde{\mathrm{D}}(\mathbb{P}||\mathbb{Q}) \equiv \mathrm{D}(\tilde{p}(y)||\tilde{q}(y))$ is a valid divergence since

$$\widetilde{\mathrm{D}}(\mathbb{P}||\mathbb{Q}) = 0 \Leftrightarrow \tilde{\mathbb{P}} = \tilde{\mathbb{Q}} \Leftrightarrow \mathbb{P}_f = \mathbb{Q}_f \Leftrightarrow \mathbb{P} = \mathbb{Q}, \tag{2.43}$$

where we denote the distributions $\mathbb{P}_f$ and $\mathbb{Q}_f$ as the transformations of $\mathbb{P}$ and $\mathbb{Q}$ with invertible function f [4] densities

$$\mathbb{P}(ds) = \int \delta(s - f(x)) d\mathbb{P}, \quad \mathbb{Q}(ds) = \int \delta(s - f(x)) d\mathbb{Q}. \tag{2.44}$$

Since injective f gives a different noise $p(y|x)$ and hence we can then define a family of f (e.g. parameterize f with a neural network) and search for the best noise implicitly by learning f.

There are many invertible neural networks that can be used to parameterize f (e.g. normalizing flows [Rezende and Mohamed, 2015]). In this work, we use the invertible residual network [Behrmann et al., 2019] $f_\psi : \mathbb{R}^D \to \mathbb{R}^D$ where $f_\psi = (f_\psi^1 \circ \ldots \circ f_\psi^T)$ denotes a ResNet with blocks $f_\psi^t = I(\cdot) + g_{\psi_t}(\cdot)$. Then f_ψ is invertible if the Lipschitz-constants $Lip(g_{\psi_t}) < 1$, for all $t \in \{1, \ldots, T\}$. Note that when using the Spread Divergence for training we only need samples from $\tilde{p}(y)$ which can be obtained by first sampling x' from $\mathbb{P}$ and then y from $p(y|x = x') = k(y - f(x'))$; this does not require computing the Jacobian or inverse ${f_\psi}^{-1}$.

We then denote the spread divergence parameterized by f_ψ as $\widetilde{\mathrm{KL}}_{f_\psi}$. To maxi-

[3] Since the co-domain of f is determined by its range, injective indicates invertible in this case.

[4] The invertible transformation defines a homeomorphism [Cornish et al., 2020], so $\mathbb{P}_f$ and $\mathbb{Q}_f$ will not be $a.c.$ if $\mathbb{P}$ and $\mathbb{Q}$ are not $a.c.$.

mize the discriminatory power, we train ψ with the following objective

$$\max_{\psi} \min_{\theta} \widetilde{\mathrm{KL}}_f(\mathbb{P}_d || \mathbb{Q}_\theta). \tag{2.45}$$

In contrast to the min-max training technique employed in the Wasserstein GAN [Arjovsky et al., 2017], where the maximum step serves to tighten the lower bound of the Wasserstein divergence, our methodology is different. Here, every instance of ψ establishes a valid divergence. Consequently, for each ψ, we can carry out a full maximization of θ without encountering any instability issues. In practice, we alternate between one-step ψ training and multiple-epoch ϕ training. For a comprehensive understanding of this process, please refer to the experimental section.

2.5 Applications

2.5.1 Deriving Deterministic PPCA

Our aim here is to show how the classical deterministic PCA algorithm can be derived through a maximum-likelihood approach, rather than the classical non-probabilistic least-squares derivation. This is remarkable since the likelihood itself is not defined for this model.

For isotropic Gaussian observation noise with variance γ^2, the Probabilistic PCA (PPCA) model [Tipping and Bishop, 1999] for Dim_X-dimensional observations and Z-dimensional latent is

$$\begin{aligned} &x = Fz + \gamma\varepsilon, \quad z \sim \mathcal{N}(0, I_Z), \quad \varepsilon \sim \mathcal{N}(0, I_X), \\ &q_\theta(x) = \mathcal{N}\left(x | 0, FF^\mathsf{T} + \gamma^2 I_X\right). \end{aligned} \tag{2.46}$$

When $\gamma = 0$, the generative mapping from z to x is deterministic and the model $q_\theta(x)$ has support only on a subset of D_X and the data likelihood is in general not defined for $\mathrm{D}_Z < \mathrm{D}_X$.

In the following we consider general γ, later setting γ to zero throughout the calculation. To fit the model to iid data $\{x_1, \ldots, x_N\}$ using maximum likelihood, the only information required from the dataset is the data covariance $\hat{\Sigma}$. For $\gamma > 0$, the

maximum likelihood solution for PPCA is $F = U_Z \left(\Lambda_Z - \gamma^2 I_Z\right)^{\frac{1}{2}} R$, where Λ_Z, U_Z are the Z largest eigenvalues, eigenvectors of $\hat{\Sigma}$; R is an arbitrary orthogonal matrix. Using spread noise $p(y|x) = \mathcal{N}\left(y|x, \sigma^2 I_X\right)$, the spreaded distribution is a Gaussian $\tilde{q}_\theta(y) = \mathcal{N}\left(y|0, FF^\mathsf{T} + (\gamma^2 + \sigma^2) I_X\right)$. Thus, $\tilde{q}_\theta(y)$ is of the same form as PPCA, albeit with an inflated covariance matrix. Adding Gaussian spread noise to the data also simply inflates the sample covariance to $\hat{\Sigma}' = \hat{\Sigma} + \sigma^2 I_X$.

Since the eigenvalues of $\hat{\Sigma}' \equiv \hat{\Sigma} + \sigma^2 I_X$ are simply $\Lambda' = \Lambda + \sigma^2 I_X$, with unchanged eigenvectors, the optimal deterministic ($\gamma = 0$) latent linear model has solution $F = U_Z \left(\Lambda'_Z - \sigma^2 I_Z\right)^{\frac{1}{2}} R = U_Z \Lambda_Z^{\frac{1}{2}} R$.

We have thus recovered the standard PCA solution; however, the derivation is non-standard since the likelihood of the deterministic latent linear model $\gamma = 0$ is not defined. Since classical deterministic PCA cannot normally be described in terms of a likelihood, the usual derivation of PCA is to define it as the optimal least-squares reconstruction solution based on a linear projection to a lower-dimensional subspace, see for example [Barber, 2012]. Nevertheless, using the Spread Divergence, we learn a sensible model and recover the true data-generating process if the data were exactly generated according to the deterministic model.

2.5.2 Training Degenerate Gauss-VAE

We consider the data distribution $\mathbb{P}_d$ which is defined by the following data generation process to sample $x \sim \mathbb{P}_d$:

$$z \sim \mathcal{B}(0.5), \quad a \sim \mathcal{N}(2z - 1, 0.1), \quad x = (a, 0), \tag{2.47}$$

where $\mathcal{B}(0.5)$ is a Bernoulli distribution with mean 0.5. Therefore, the data distribution $\mathbb{P}_d$ is a mixture of two degenerated Gaussian which is supported on a 1D manifold in a 2D space, we visualize the data distribution in Figure 2.4.

We then learn $\mathbb{P}_d$ with a generalized implicit model with a discrete latent

$$\mathbb{Q}_\theta(\mathrm{d}x) = \int \mathcal{N}_g(x|g_\theta(z), \Sigma_\theta(z)) p(z)\, \mathrm{d}z, \tag{2.48}$$

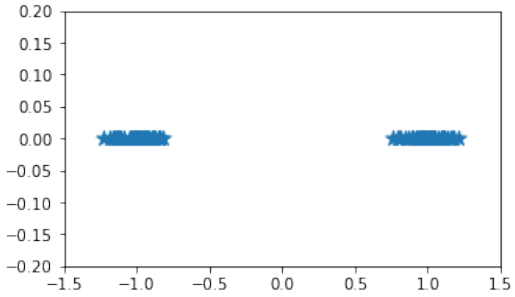
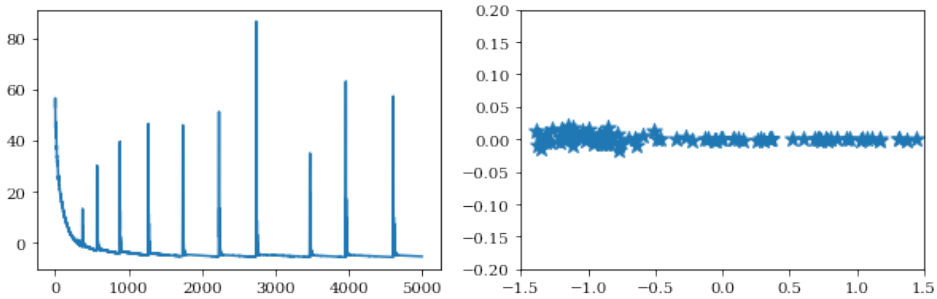


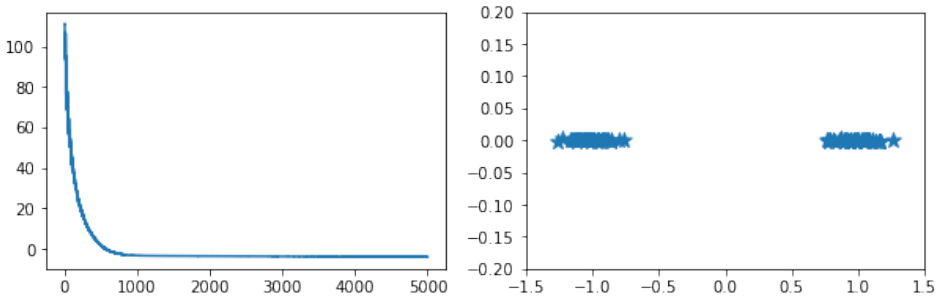
Figure 2.4: A mixture of two degenerated Gaussians.

where the prior $p(z)$ is a 1D Bernoulli distribution and $\mathcal{N}_g(x|g_\theta(z),\Sigma_\theta(z))$ is a generalized Gaussian with learnable mean and learnable diagonal variance $[\sigma_1^2(z),\sigma_2^2(z)]$, both are parameterized by a neural network. We use a softplus activation function on the output of the variance network to allow both $\sigma_1^2(z),\sigma_2^2(z)$ to converge to 0 so the $\mathcal{N}_g(x|g_\theta(z),\Sigma_\theta(z))$ can be a degenerate Gaussian distribution with $\text{rank}(\Sigma_\theta(z)) < \text{Dim}(x)$. Both encoder and decoder contain 2-layer neural networks with ReLU as the activation function and 16 hidden units. We train the model using both the classic ELBO and the proposed spread ELBO for 5k iterations with Adam optimizer [Kingma and Ba, 2014] (with learning rate 1×10^{-3}) and batch-size 100. For the spread ELBO, we use the Gaussian spread noise $p(y|x) = \mathcal{N}(y|x,\sigma^2 I)$ where $\sigma = 0.02$.

In Figure 2.5, we compare the training loss from two degenerated Gauss-VAEs that are trained with the classic ELBO (Figure a) and the proposed spread ELBO (Figure b) respectively. We can see for the classic ELBO training, the training loss is not very smooth. This is because when the $\sigma_2(z)$ in the $p_\theta(x|z)$ converges to 0, the log-density $\log p_\theta(x|z)$ collapses to a δ distribution and the likelihood becomes infinity, which impedes the training. On the other hand, the likelihood of the spread model $p_\theta(y|z)$ is always well-defined when the underlying $p_\theta(x|z)$ becomes a δ distribution, so the spread ELBO objective gives a smooth training curve. We also visualize the samples from both models and find the one trained with the spread ELBO successfully recovers the true data distribution whereas the one trained with the classic ELBO fails to generate valid samples.



(a) Degenerate Gauss-VAE with ELBO



(b) Degenerate Gauss-VAE with Spread ELBO

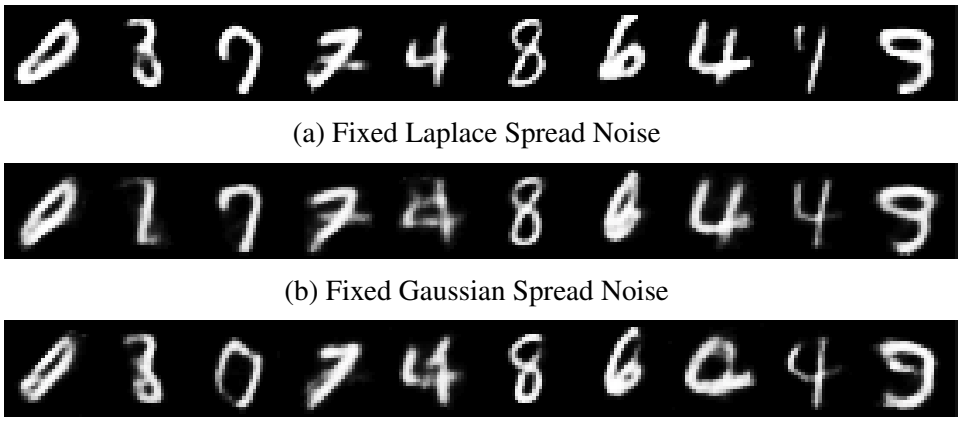
Figure 2.5: Training loss and model samples comparisons for the degenerated Gauss-VAEs trained with the classic ELBO (Figure a) and the proposed spread ELBO (Figure b) respectively We can see the one trained with spread ELBO (Figure b) has smoother training curves and successfully recovers the underlying data distribution whereas the classic ELBO training is unstable for this manifold model.

2.5.3 Image Modelling with δ-VAE

2.5.3.1 MNIST

We trained a δ-VAE with spread ELBO on MNIST [LeCun, 1998] with (i) fixed Laplace spread noise, as in Equation 2.17, (ii) fixed Gaussian spread noise, as in Equation 2.16 and (iii) Gaussian noise with learned covariance, as in Section 2.4.1, with rank $R = 20$; $g_\theta(\cdot)$ is a neural network that contains 3 feed-forward layers.

Figures 2.6(a,b,c) show samples from $p_\theta(x)$ for these models. Given that MNIST is a comparatively simpler problem, it becomes challenging to differentiate between the quality of fixed and learned noise samples. We speculate that the use of Laplace noise enhances image sharpness. This is likely because the Laplace distribution, being leptokurtic, inherently places more emphasis on discriminating between points in close proximity to the data manifold. This is in contrast to the



(a) Fixed Laplace Spread Noise

(b) Fixed Gaussian Spread Noise

(c) Learned Gaussian Spread Noise

Figure 2.6: Samples from a deep implicit generative model trained using δ-VAE with (a) Laplace spread noise with fixed covariance, (b) Gaussian spread noise with fixed covariance and (c) Gaussian spread noise with learned covariance. We first train with one epoch a standard VAE as initialization to all models and keep the latent code $z \sim \mathcal{N}(z|0, I_Z)$ fixed when sampling from these models thereafter, so we can more easily compare the sample quality.

(a) δ Fixed spread noise (b) δ Learned spread noise

Figure 2.7: Samples from a deep implicit generative model trained using δ-VAE with (a) fixed and (b) learned spread with the mean transformation method. We use a similar sampling strategy as in the MNIST experiment to facilitate sample comparison between the different models – see Appendix A.7.

Gaussian distribution, which is less likely to generate points near the data manifold.

2.5.3.2 CelebA

We trained a δ-VAE with spread ELBO on the CelebA dataset [Liu et al., 2015] with (i) fixed and (ii) learned spread using the mean transformation method as discussed in Section 2.4.2. We compare to results from a standard VAE with fixed Gaussian noise $p(x|z) = \mathcal{N}(x|g_\theta(z), 0.5I_X)$ [Tolstikhin et al., 2017], where $g_\theta(\cdot)$ is a neural network contains 4 convolution layers.

For (i) the fixed Spread Divergence uses Gaussian noise $\mathcal{N}(y|x, 0.25I_X)$. For (ii) we use Gaussian noise with a learned injective function in the form of a ResNet: $f_\psi(\cdot) = I(\cdot) + g_\psi(\cdot)$ - see Appendix A.7 for details. Figure 2.7 shows samples from our δ-VAE for (i) and (ii) (with $g_\theta(z)$ initialised to the fixed-noise setting). It is notable how the 'sharpness' of the image samples substantially increases when learning the spread noise. Table 2.1 shows Frechet Inception Distance (FID) [Heusel et al., 2017] score[5] comparisons between different baseline algorithms for implicit generative model training[6]. The δ-VAE significantly improves on the standard VAE result. Learning the mean transformation improves on the fixed-noise δ-VAE. Indeed the mean transformation δ-VAE results are comparable to popular GAN and WAE models [Gulrajani et al., 2017a, Berthelot et al., 2017, Arjovsky et al., 2017, Kodali et al., 2017, Mao et al., 2017, Fedus et al., 2017, Tolstikhin et al., 2017]. Whilst the δ-VAE results are not state-of-the-art, we believe it is the first time that implicit models have been trained using a principled maximum likelihood-based approach. By increasing the complexity of the generative model g_θ and injective function f_ψ, or using better choices of noise, the results may become more competitive with state-of-the-art GAN models[7].

[5]The FID score is a measure of similarity between two datasets of images. FID is calculated by computing the Fréchet distance between two Gaussians fitted to feature representations of the dataset and the samples from the model using an Inception network [Szegedy et al., 2017], see Heusel et al. [2017] for a detailed introduction

[6]FID scores were computed using `github.com/bioinf-jku/TTUR` based on 10000 samples.

[7]We also tried learning the image generator using Laplace spread noise. However, the colour of the sampled images becomes overly intense and we leave it to future work to address this.

Table 2.1: CelebA FID Scores. The δ-VAE results are the average over 5 independent measurements. The scores of the GAN models are based on a large-scale hyperparameter search and take the best FID obtained [Lucic et al., 2018]. The results of the VAE model and both WAE-based models are from [Tolstikhin et al., 2017].

Encoder-Decoder Models	FID	GAN Models	FID
VAE	63.0	WGAN GP	30.0
δ-VAE with fixed spread	**52.7**	BEGAN	38.9
δ-VAE with learned spread	**46.5**	WGAN	41.3
		DRAGAN	42.3
WAE-MMD	55.0	LSGAN	53.9
WAE-GAN	42.0	NS GAN	55.0
		MM GAN	65.6

2.6 Related Work

Instance noise: The instance noise trick to stabilize GAN training [Roth et al., 2017, Sønderby et al., 2016] is a special case of Spread Divergence using fixed Gaussian noise. Whilst other similar tricks, e.g. [Furmston and Barber, 2009], have been proposed previously, we believe it is important to state the more general utility of the spread noise approach.

δ-VAE versus WAE: The Wasserstein Auto-Encoder [Tolstikhin et al., 2017] is another implicit generative model that uses an encoder-decoder architecture. The major difference to our work is that the δ-VAE is based on the KL divergence, which corresponds to MLE, whereas the WAE uses the Wasserstein distance.

δ-VAE versus denoising VAE: The denoising VAE [Im Im et al., 2017] uses a VAE with noise added to the data only. In contrast, for our δ-VAE, spread MLE adds noise to both the data and the model. Therefore, the denoising VAE cannot recover the true data distribution, whereas in principle the δ-VAE with spread MLE can.

MMD GAN with kernel learning: Learning a kernel to increase discrimination is also used in MMD GAN [Li et al., 2017]. Similar to ours, the kernel in MMD GAN is constructed by $\tilde{k} = k \circ f_\psi$, where k is a fixed kernel and f_ψ is a neural network. To ensure the MMD distance $M_{k\circ f_\psi}(p,q) = 0 \Leftrightarrow p = q$, this requires f_ψ to be injective [Gretton et al., 2012]. However, in this framework, $f_\psi(x)$ usually maps x to a lower

dimensional space. This is crucial for MMD because the amount of data required to produce a reliable estimator grows with the data dimension [Ramdas et al., 2015] and the computational cost of MMD scales quadratically with the amount of data. Whilst using a lower-dimensional mapping makes MMD more practical it also makes it difficult to construct an injective function f. For this reason, heuristics such as the auto-encoder regularizer [Li et al., 2017] are considered. In contrast, for the δ-VAE with spread MLE, the cost of estimating the divergence is linear in the number of data points. Therefore, there is no need for f_ψ to be a lower-dimensional mapping; guaranteeing that f_ψ is injective is straightforward for the δ-VAE.

Flow-based generative models: Invertible flow-based functions [Rezende and Mohamed, 2015] have been used to boost the representation power of generative models. Our use of injective functions is quite distinct from the use of flow-based functions to boost generative model capacity. In our case, the injective function f does not change the model - it only changes the divergence. For this reason, the Spread Divergence doesn't require the log determinant of the Jacobian, which is required in [Rezende and Mohamed, 2015, Behrmann et al., 2019], meaning that more general invertible functions can be used to boost the discriminatory power of a Spread Divergence.

2.6.1 Connection to Denoising Score Matching

Recently, Energy-Based Models (EBMs) have gained significant traction within generative model research [Ngiam et al., 2011, Xie et al., 2016, Du and Mordatch, 2019, Song and Ermon, 2019] and play a key role in the success of diffusion models [Ho et al., 2020, Song and Ermon, 2019, 2020]. EBMs are a type of non-normalized probabilistic model that determines the probability density function without a known normalizing constant. For continuous data x, the density function of an EBM is specified as

$$q_\theta(x) = \exp(-f_\theta(x))/Z(\theta), \tag{2.49}$$

where the $f_\theta(x)$ is a nonlinear function with parameter θ and $Z(\theta) = \int \exp(-f_\theta(x))\,\mathrm{d}x$ is the normalization constant that is independent of x. The energy parameterization allows for greater flexibility in model parameterization and the ability to model a wider range of probability distributions.

A popular method to train EBMs is Denoising score matching (DSM) [Vincent, 2011]. This method uses a noise distribution $p(\tilde{x}|x) = \mathcal{N}(x, \sigma^2 I)$ to construct a noised data distribution $\tilde{p}_d(\tilde{x}) = \int p_d(x)p(\tilde{x}|x)\,\mathrm{d}x$. Subsequently, DSM minimizes the Fisher divergence between the noised data distribution $\tilde{p}_d(\tilde{x})$ and an EBM $\tilde{q}_\theta(\tilde{x}) = \exp(-f_\theta(\tilde{x}))/Z(\theta)$, with

$$\begin{aligned}
\mathrm{FD}(\tilde{p}_d||\tilde{q}_\theta) &= \frac{1}{2}\int \tilde{p}_d(\tilde{x})||s_{\tilde{p}_d}(\tilde{x}) - s_{\tilde{q}_\theta}(\tilde{x})||_2^2\,\mathrm{d}\tilde{x} \\
&\doteq \frac{1}{2}\iint p(\tilde{x}|x)p_d(x)||\nabla_{\tilde{x}}\log p(\tilde{x}|x) - s_{\tilde{q}_\theta}(\tilde{x})||_2^2\,\mathrm{d}\tilde{x}\,\mathrm{d}x \\
&\doteq \frac{1}{2}\iint p(\tilde{x}|x)p_d(x)\left\|\frac{\tilde{x}-x}{\sigma^2} + s_{\tilde{q}_\theta}(\tilde{x})\right\|_2^2\,\mathrm{d}\tilde{x}\,\mathrm{d}x,
\end{aligned} \tag{2.50}$$

where the last equation is due to $\nabla_{\tilde{x}}\log p(\tilde{x}|x)$ being tractable for the Gaussian distribution $p(\tilde{x}|x)$. DSM can be extended to handle multi-level noise scenarios, allowing the training of multiple energy-based models. With an amortized energy (or score) network corresponding to each noise level, it is termed a *score-based diffusion model*. A detailed discussion on this can be found in [Song and Ermon, 2019].

The DSM objective trains an EBM to align with the noisy data distribution the noisy data distribution $\tilde{p}_d$, rather than the true underlying distribution p_d. In the ideal scenario when $\tilde{q}_{\theta^*} = \tilde{p}_d$, there exists an underlying clean model $q^*(x)$, such that $\int q^*(x)p(\tilde{x}|x)\,\mathrm{d}x = \tilde{q}_{\theta^*}(\tilde{x})$. This means the clean model would align with the actual data distribution $q^*(x) = p_d(x)$. However, for an imperfect EBM $\tilde{q}_\theta \neq \tilde{p}_d$, the existence of the clean model is not guaranteed, see [Zhang et al., 2023a] for a comprehensive discussion on the existence of the clean model.

In relating to the spread divergence, we posit that it's possible to craft a function family $f_\theta \in \mathscr{F}$ such that there always exists an underlying clean model $q_\theta(x)$ that satisfies $\int q_\theta(x)p(\tilde{x}|x)\,\mathrm{d}x = \exp(-f_\theta(\tilde{x}))/Z(\theta)$, then training the DSM objective is

equivalent to minimizing the *Spread Fisher Divergence* ($\widetilde{\text{FD}}$), defined as

$$\widetilde{\text{FD}}(p_d(x)||q_\theta(x)) = \text{FD}\left(\int p_d(x)p(\tilde{x}|x)\,\mathrm{d}x||\int q_\theta(x)p(\tilde{x}|x)\,\mathrm{d}x\right). \tag{2.51}$$

Designing such a functional family in practical terms continues to be a challenge; however, this close relationship sheds light on the intricacies of DSM. This connection opens up a promising path for deeper exploration, both in understanding and enhancing DSM training. We leave this for future investigations.

2.7 Discussions

In this chapter, we introduced the spread divergence as a solution to the shortcomings of the KL divergence when the two distributions either lack valid probability density functions or share the same support. We outlined a methodology employing this proposed divergence for training implicit generative models, closely aligning with traditional likelihood or ELBO maximization techniques. Additionally, we explored the potential of learning spread noise to enhance discrimination between two distributions, subsequently improving image generation results.

A central insight from our proposed method is the efficacy of introducing equal noise to both data and model. This not only bolsters model training under specific conditions but also ensures the estimator remains consistent. The innovative spread divergence has paved the way for a slew of subsequent research, leading to the development of new models and training methodologies aimed at refining manifold modeling outcomes [Zhang et al., 2023b, Loaiza-Ganem et al., 2023, 2022]. A recent paper [Graves et al., 2023] also uses a similar technique in building the generative model, delivering state-of-the-art image generation results.

There are several potential directions for further exploration of spread divergence. For instance, one might investigate the broader family of spread noise and its associated properties, or determine the optimal noise type for various tasks. As we discussed in Section 2.6.1, introducing noise to the data distribution has become a pivotal technique in the realm of the diffusion model. An enticing avenue is to comprehend and refine the diffusion model's training from the vantage point of

spread divergence, which holds the potential to further enhance its training.

We have addressed the issue of the KL divergence being unsuitable for manifold distributions that lack density functions. Within Section 2.6.1, we touch upon the score-based method employed in training the energy-based model, a pivotal component of diffusion models. Additionally, there exists a notable failure mode in the score-based approach to model multi-modal distributions. We will delve into this failure mode and present a tailored solution in the next chapter.

Chapter 3

Healing the Fisher Divergence for Multi-Modality Modelling

Score-based divergences have been widely used in machine learning and statistics applications. Despite their empirical success, a blindness problem has been observed when using these for multi-modal distributions. In this work, we discuss the blindness problem and propose a new family of divergences that can mitigate the blindness problem. We illustrate our proposed divergence in the context of density estimation and report improved performance compared to traditional approaches.

3.1 Introduction

Score-based divergences such as the Fisher Divergence (FD; also known as score-matching divergence) [Aapo, 2005, Hyvärinen, 2007] and Kernel Stein Discrepancy (KSD) [Liu et al., 2016, Chwialkowski et al., 2016] are widely used in machine learning and statistics [Anastasiou et al., 2021, Song and Kingma, 2021]. Their main advantage is that the score function, a derivative of a log-density, can be evaluated without knowledge of the normalization constant of the density and can be applied to problems where other classical divergences (e.g. KL divergence) are intractable. Unfortunately, this advantage can also be a curse in certain scenarios because the score function only provides local information about the slope of a density, but ignores more global information such as the importance of a point relative to another. This has led to a blindness problem in many applications of score-based methods

where the densities are multi-modal, including in density estimation [Wenliang et al., 2019, Song and Ermon, 2019, Jolicoeur-Martineau et al., 2020], MCMC convergence diagnosis [Gorham et al., 2019], Bayesian inference [Matsubara et al., 2022, D'Angelo and Fortuin, 2021]; see [Wenliang and Kanagawa, 2020] for a detailed discussion.

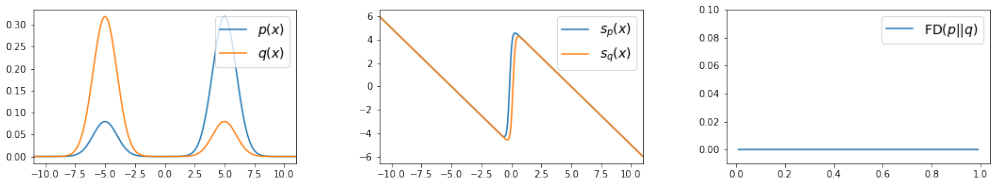
To illustrate this problem, we recall the definition of FD and an example from [Wenliang and Kanagawa, 2020]. Given two distributions with differentiable densities p and q supported on a common domain $\mathscr{X} \subseteq \mathbb{R}^d$, the FD is

$$\mathrm{FD}(p||q) = \frac{1}{2}\int_{\mathscr{X}} p(x)||s_p(x) - s_q(x)||_2^2 \,\mathrm{d}x. \tag{3.1}$$

The classic sufficient conditions [Aapo, 2005, Barp et al., 2019] for the FD to be a valid statistical divergence (i.e. $\mathrm{FD}(p||q) = 0 \Leftrightarrow p = q$) are: (i) p and q are differentiable with support $\mathscr{X} = \mathbb{R}^d$ and (ii) s_p, s_q are square integrable, i.e. $s_p - s_q \in L^2(p)$, where we denote $f \in L^2(p) \equiv \int_{\mathscr{X}} ||f(x)||_2^2 p(x)\,\mathrm{d}x < \infty$. The blindness problem of the FD can be illustrated through the following example due to [Wenliang and Kanagawa, 2020]. Let p and q be a mixtures with the same components but different mixing weights:

$$p(x) = \alpha_p g_1(x) + (1-\alpha_p) g_2(x), \quad q(x) = \alpha_q g_1(x) + (1-\alpha_q) g_2(x), \tag{3.2}$$

where $\alpha_p \neq \alpha_p$, and g_1, g_2 are Gaussian densities with variance σ^2 and means $-\mu$ and μ respectively. Then $\mathrm{FD}(p||q) \to 0$ when $\mu/\sigma^2 \to \infty$ regardless of the mixture proportions α_p and α_q. To build intuition, we let $\mu = 5$, $\sigma = 1$, $\alpha_p = 0.2, \alpha_q = 0.8$ and plot the densities and score functions of p, q in Figure 3.1a and 3.1b. We can find the two distributions are very different but their scores are only different around $x = 0$, which has a negligible density value under p. We then fix $\alpha_p = 0.2$ and plot the $\mathrm{FD}(p||q)$ as a function of α in Figure 3.1c. Here we see the FD is 0 constant function, which shows the FD is 'blind' to the value of the mixture weight. See [Matsubara et al., 2022] for a similar example for discrete $\mathscr{X}$.



(a) Densities of p, q (b) Score functions of p, q (c) FD$(p||q)$ with different α_q

Figure 3.1: We plot the densities and score functions of distributions p and q in Figure (a) and (b). Figure (c) shows FD$(p||q)$ with $\alpha_p = 0.2$ and α_q varies from 0.01 to 0.09 with a grid size 0.01.

3.2 Understanding the Blindness Problem

In the example above, blindness is a numerical problem since the problem occurs despite the fact that the FD is a divergence in that case (i.e. FD$(p||q) = 0 \Leftrightarrow p = q$ since (i) and (ii) are satisfied). When $\mu/\sigma^2 \to \infty$, although the Gaussian distributions still have the same support, the regions that contain most of the mass of g_1 and g_2 tend to be disjoint, which creates numerical issues. However, the blindness problem is not simply a numerical problem, as illustrated in the following example.

Consider the case where p and q are mixtures whose identical components have disjoint supports. For example, let g_1 and g_2 in Equation 3.2 have disjoint support sets $\mathscr{X}_1, \mathscr{X}_2 \subseteq \mathbb{R}^d$ respectively with $\mathscr{X}_1 \cap \mathscr{X}_2 = \emptyset$. Then, $g_2(x') = \nabla_x g_2(x') = 0$ for $x' \in \mathscr{X}_1$ and $g_1(x') = \nabla_x g_1(x') = 0$ for $x' \in \mathscr{X}_2$. In this case, the FD is independent of α_q (see Appendix B.1.1 for a derivation):

$$\text{FD}(p||q) = \frac{\alpha_p}{2}\int_{\mathscr{X}_1} g_1(x)||s_{g_1}(x) - s_{g_1}(x)||_2^2 \,\mathrm{d}x + \frac{1-\alpha_p}{2}\int_{\mathscr{X}_2} g_2(x)||s_{g_2}(x) - s_{g_2}(x)||_2^2 \,\mathrm{d}x = 0. \quad (3.3)$$

Therefore, the FD is not a valid divergence here since FD$(p||q) = 0 \not\Rightarrow p = q$. This example guides us to further study the topology properties of the distributions' support required by the FD. We first extend the Fisher divergence to distributions that have support on the connected space.

Theorem 2 (FD on a connected set)**.** *Assume two distributions (i) have differentiable densities p and q with support on a common*[1] *open connected set $\mathscr{X} \subseteq \mathbb{R}^d$ and (ii)*

[1]The common support condition can be relaxed to $\mathscr{X}_p \subseteq \mathscr{X}_q$, where $\mathscr{X}_p, \mathscr{X}_q$ are the support sets of p and q.

$s_p - s_q \in L^2(p)$. *Then, the FD is a valid divergence i.e.* $\mathrm{FD}(p||q) = 0 \Leftrightarrow p = q$.

See Appendix B.1.2 for a proof. Theorem 2 generalizes the classic FD that is defined on distributions with $\mathscr{X} = \mathbb{R}^d$ [Aapo, 2005, Barp et al., 2019] ($\mathbb{R}^d$ is a special case of the connected set). Secondly, Theorem 3 shows that *connectedness* of the support is a *necessary* condition to define a valid FD.

Theorem 3 (FD is ill-defined on disconnected sets)**.** *Let* $\mathscr{X}$ *be a union of disjoint sets, then the FD is not a valid divergence on* $\mathscr{X}$*, i.e. there exist two different distributions* $p \neq q$ *both supported on* $\mathscr{X}$ *but* $\mathrm{FD}(p||q) = 0$.

See Appendix B.1.3 for a proof. Intuitively, the score function only considers the local derivatives and contains no information on the global normalization constant. If the domain is disconnected, it cannot determine the mass allocation in different domains. This observation can also be extended to the KSD by viewing KSD as a kernelized FD [Liu et al., 2016, Chwialkowski et al., 2016], see Appendix B.1.4 for a detailed discussion.

3.3 Healing the Blindness Problem with Mixture Fisher Divergence

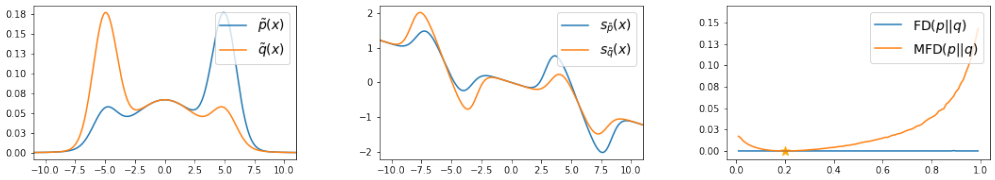
In this section, we propose a new variant of the FD which is well-defined in the disconnected scenario. Consider a distribution with density m with support $\mathscr{X}_m = \mathbb{R}^d$ and define the mixtures

$$\tilde{p}(x) = \beta p(x) + (1-\beta)m(x), \quad \tilde{q}(x) = \beta q(x) + (1-\beta)m(x), \tag{3.4}$$

where $0 < \beta < 1$. We then define the *Mixture Fisher Divergence* (MFD) as

$$\mathrm{MFD}_{m,\beta}(p||q) \equiv \mathrm{FD}(\tilde{p}||\tilde{q}). \tag{3.5}$$

Theorem 4 shows the MFD is well-defined when p and q have support on a disconnected space.



(a) Densities of $\tilde{p}, \tilde{q}$ (b) Score functions of $\tilde{p}, \tilde{q}$ (c) MFD$(p||q)$ with diff. α_q

Figure 3.2: We plot the densities (a) and the score functions (b) of $\tilde{p}$ and $\tilde{q}$. Figure (c) shows MFD$(p||q)$ with $\alpha_p = 0.2$ and α_q varies from 0.0 to 1.0 with a grid size 0.01. The star mark shows the minima of the MFD is achieved when $\alpha_q = \alpha_p = 0.2$, we also plot the original FD for a comparison.

Theorem 4 (Validity of the MFD)**.** *Consider two distributions with differentiable densities p, q supported on $\mathscr{X}_p, \mathscr{X}_q \subseteq \mathbb{R}^d$ with $s_p, s_q \in L^2(p)$ and a differentiable density m with support $\mathscr{X}_m = \mathbb{R}^d, s_m \in L^2(p)$. Then MFD is a valid divergence, i.e.* $\text{MFD}(p||q) = 0 \Leftrightarrow \text{FD}(\tilde{p}||\tilde{q}) \Leftrightarrow p = q$.

See Appendix B.1.5 for a proof. For MFD, we no longer require that p, q have *common connected support*, since $\mathscr{X}_m = \mathbb{R}^d$ results in $\tilde{p}, \tilde{q}$ having connected support $\mathbb{R}^d$[2]. The requirements of $m(x)$ are mild and hold for simple choices of distribution e.g. a Gaussian. To avoid the numerical problem mentioned in Section 1, $m(x)$ should be chosen to effectively connect the different component distributions. As an example, for the toy problem described in Figure 1 with components $\mathscr{N}(-5, 1)$ and $\mathscr{N}(-5, 1)$, we can choose $\beta = 0.5$ and $m(x) = \mathscr{N}(0, 9)$ that covers both components. Figure 3.2 shows the densities and their score functions for $\tilde{p}, \tilde{q}$. We see that the score functions are different on the high-density region of $\tilde{p}$. Figure 3.2c also shows the minimal value of the MFD$(p||q)$ is attained when $\alpha_q = \alpha_p$, which indicates that the proposed MFD heals the blindness problem in this example.

3.4 Density Estimation with Energy-based Models

Given a dataset $\mathscr{X}_{\text{train}} = \{x_1, \cdots, x_N\}$ sampled i.i.d. from a data distribution p_d with support $\mathscr{X}_{p_d} \subseteq \mathbb{R}^d$, we would like to learn a model q_θ to approximate p_d. We are interested in a family of models which can only be evaluated up to a normalization

[2]A weaker condition of m can be obtained by requiring the supports of $\tilde{p}, \tilde{q}$, which we denote as $\mathscr{X}_{\tilde{p}}, \mathscr{X}_{\tilde{q}}$, to be connected and $\mathscr{X}_{\tilde{p}} \subseteq \mathscr{X}_{\tilde{q}}$. We here only study the stronger condition that $m(x)$ has support of $\mathbb{R}^d$ for simplicity.

constant, e.g. an energy-based model $q_\theta(x) = e^{-f_\theta(x)}/Z(\theta)$, where f_θ is a neural network and $Z(\theta) = \int e^{-f_\theta(x)}\,\mathrm{d}x$. In this case, the standard Maximum Likelihood Estimation (MLE) is not applicable (since $Z(\theta)$ cannot be evaluated during training) and an alternative form of the FD [Aapo, 2005] can be applied

$$\mathrm{FD}(p_d||q_\theta) = \frac{1}{2}\int_{\mathscr{X}_{p_d}} p_d(x)\left(||s_{q_\theta}(x)||_2^2 + 2\,\mathrm{Tr}(\nabla_x s_{q_\theta}(x))\right)\mathrm{d}x + const., \tag{3.6}$$

where $\nabla_x s_{q_\theta}(x) = \nabla_x^2 \log q_\theta(x)$ is the Hessian matrix and the constant represents the terms that are independent of θ. The integration over p_d can be approximated by Monte-Carlo using $\mathscr{X}_{train}$. Because both s_{q_θ} and $\nabla_x s_{q_\theta}$ only depend on f_θ, the normalizer $Z_q(\theta)$ is not required during training and we only need to estimate $Z_q(\theta^*)$ once after training. Therefore, density estimation with FD in this setting contains two steps: (1) learn θ^* using Equation 3.6; (2) estimate $Z(\theta^*)$ to obtain the normalized density $q_\theta(x)$. This scheme can result in blindness in practice [Wenliang et al., 2019].

To heal blindness, we can apply the proposed MFD. However, if we directly minimize MFD in step (1), the score

$$s_{\tilde{q}_\theta}(x) = \nabla_x \log\left(\beta \exp(-f_\theta(x))/Z_q(\theta)) + (1-\beta)m(x)\right) \tag{3.7}$$

requires estimating $Z_q(\theta)$. This negates the advantage of using score matching because now $Z_q(\theta)$ must be estimated for every gradient step during training (similar to MLE). To avoid this, we propose to instead directly approximate $\tilde{p}_d$ with an energy-based model $\tilde{q}_\theta(x) \equiv e^{-f_\theta(x)}/Z_{\tilde{q}}(\theta)$ and $\tilde{q}_\theta$ can then be trained using

$$\mathrm{FD}(\tilde{p}_d||\tilde{q}_\theta) = \frac{1}{2}\int_{\mathbb{R}^d} \tilde{p}_d(x)\left(||s_{\tilde{q}_\theta}(x)||_2^2 + 2\,\mathrm{Tr}\left(\nabla_x s_{\tilde{q}_\theta}(x)\right)\right)\mathrm{d}x + const., \tag{3.8}$$

where the integration over $\tilde{p}_d(x)$ can be approximated using the samples from the mixture $\tilde{p}_d(x) = \beta p_d(x) + (1-\beta)m(x)$. Therefore, the learning of θ is independent of $Z_{\tilde{q}}(\theta)$. Optimally we have $\tilde{q}_{\theta^*}(x) = \tilde{p}_d(x) = \beta p_d(x) + (1-\beta)m(x)$. To obtain a model of the underlying true density $q^* = p_d$, we need to remove the mixture

component from $\tilde{q}_{\theta^*}$, which can be done through a 'correction step':

$$q^*(x) = \frac{1}{\beta}\left(\tilde{q}_{\theta^*}(x) - (1-\beta)m(x)\right) = \frac{1}{\beta}\left(\tilde{q}_{\theta^*}(x) - (1-\beta)m(x)\right). \tag{3.9}$$

This procedure for obtaining q^* is equivalent to $q^*(x) = \arg\min_q \mathrm{MFD}(p_d(x)||q(x))$ and when $\mathrm{MFD}(p_d(x)||q(x)) = 0$, we have $q^*(x) = p_d(x)$. Therefore, density estimation with MFD in this setting contains three steps: (1) learn θ^* by minimizing Equation 3.8; (2) estimate $Z_{\tilde{q}}(\theta^*)$; and (3) apply the correction step (Equation 3.9) to obtain q_{θ^*}. Compared to FD, the additional correction step has negligible computation cost.

In practice, for an imperfect model $\tilde{q}_\theta(x) \neq \tilde{p}_d(x)$, $\tilde{q}_\theta(x) - (1-\beta)m(x)$ might take a negative value for certain x. To ensure a positive resulting density, We can apply a $\max(\cdot, 0)$ operation, This yields the estimator:

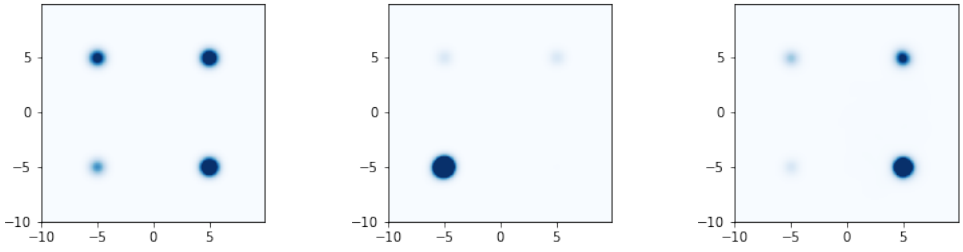
$$\hat{q}(x) = \frac{1}{\beta}\left(\tilde{q}_\theta(x) - (1-\beta)m(x)\right) = \frac{1}{\beta}\max\left(\tilde{q}_\theta(x) - (1-\beta)m(x), 0\right), \tag{3.10}$$

which is still a consistent estimator since $\tilde{q}_\theta(x) \to \tilde{p}_d(x) \Rightarrow \hat{q}(x) \to p_d(x)$.

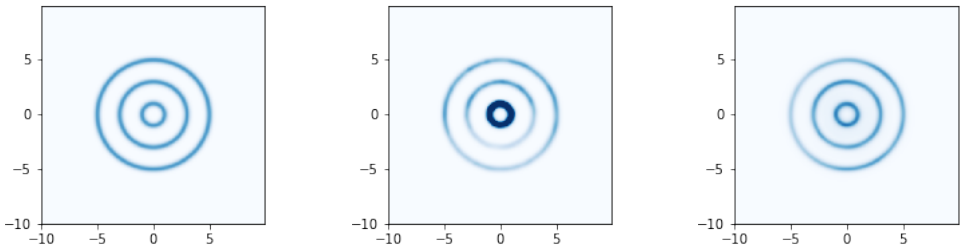
Choice of m and β

As we discussed in Section 3.3, the selection of the introduced $m(x)$ distribution is pivotal to our methodology. Ideally, a good m should be able to bridge disconnected component distributions and has significant mass in the connected paths. For the low-dimensional data distribution, we propose a heuristic solution to choose the $m(x)$: for a given set of data samples $\{x_1, \cdots, x_N\} \sim p_d$, we can simply choose $m(x) = \mathcal{N}(\bar{\mu}, \bar{\Sigma})$, where $\bar{\mu}$ and $\bar{\Sigma}$ are the empirical mean and covariance of the available training data: $\bar{\mu} = \frac{1}{N}\sum_{n=1}^{N} x_n, \bar{\Sigma} = \frac{1}{N}\sum_{n=1}^{N} x_n x_n^T$. This construction corresponds to an empirical moment matching approximation of p_d which is known for its 'mode covering' behavior [Bishop, 2006, Zhang et al., 2019b].

The β is treated as a hyper-parameter in our method. Intuitively, a large beta means that the proportion of data points from p_d is small, and the model is learning m. On the other hand, a small value means we may still have the numerical version of the blindness issue. In this experiment, we use $\beta = 0.8$ can find it can empirically



(a) True (b) FD (c) MFD



(d) True (e) FD (f) MFD

Figure 3.3: Density estimation comparisons with FD and MFD for the energy-based model. The KL$(p_d||p_\theta)$ evaluations are 3.52/0.22 (b/e) for FD and 0.17/0.01 (c/f) for MFD, lower is better.

heal blindness. We leave the theoretical study of choosing the β into future work.

Demonstrations

We apply the proposed method to train a deep energy-based model and examine the performance against two target densities with multiple isolated components: 1) a weighted mixture of four Gaussians $p_d(x) = 0.1g_1(x) + 0.2g_2(x) + 0.3g_3(x) + 0.4g_4(x)$, where g_1, g_2, g_3, g_4 are 2D Gaussians with identity covariance matrix and mean $[-5,-5], [-5,5], [5,5], [5,-5]$ respectively; and 2) a mixture of 3 concentric circles as proposed in [Wenliang et al., 2019]. We use Simpson's rule for the 2D numerical integration to estimate the normalization constant for both methods. The model specifications and training details can be found in Appendix B.2. In Figure 3 we plot the ground truth and the estimated density with classic FD and the proposed MFD methods. We also provide the corresponding KL evaluation (see Appendix B.2) between the ground truth density p_d and the estimated model p_θ. We find the proposed MFD method can significantly improve performance and heal the blindness problem.

3.5 Related Works

The issue of blindness is unique to score-based divergences like FD, KSD [Chwialkowski et al., 2016, Liu et al., 2016], or Diffusion KSD [Barp et al., 2019]. Therefore, the KL divergence (MLE) offers an alternative for training the EBMs, free from the blindness issue [Kim and Bengio, 2016, Zhai et al., 2016]. However, employing MLE necessitates estimating the normalization constant $Z(\theta)$ in every gradient step, potentially introducing errors that result in biased gradients. Contrastingly, our method mandates just a single estimation of the normalization constant post-training, streamlining the training process.

Besides the KL divergence, the generative modeling domain also commonly employs other metrics, including the f-divergence family, Wasserstein distance [Kantorovich, 1960], and MMD [Gretton et al., 2012]. However, training models using these divergences often requires that gradients be back-propagated from samples to model parameters, a step not directly feasible for energy-based models. This calls for leveraging the REINFORCE gradient estimator [Williams, 1992], MCMC sampling, and adversarial training at each optimization juncture; for more details, refer to [Yu et al., 2020]. Our primary objective in this chapter is to refine score-based methods rather than fully revamping training paradigms. Therefore, we'll investigate alternate score-based techniques, examining their potential to address the blindness issue.

Fisher Divergence with Normalizing Flow

Paper [Gong and Li, 2021] proposes to transform p and q with a common differentiable invertible function before defining the FD. Specifically, for distribution $p(x)$ and $q(x)$, we let $p(y) = \int \delta(y - g(x))p(x)dx$ and $q(y) = \int \delta(y - g(x))q(x)dx$, where g is a invertible function, e.g. a normalizing flow [Rezende and Mohamed, 2015]. The Fisher divergence is then defined in the transformed space and have

$$\text{FD}(p(y)||q(y)) = 0 \Leftrightarrow p(x) = q(x), \tag{3.11}$$

which is also shown to be equivalent to the diffusion KSD [Barp et al., 2019]. However, since the invertible transformation is a homeomorphism and will not

change the topology of its domain [Cornish et al., 2020, Zhang et al., 2023b], the invertible transformation will not fix the blindness caused by the disconnected support sets in principle.

Spread Fisher Divergence

In addition to the mixture construction, as we discussed in Chapter 2, conducting a Gaussian convolution on both p_d and q_θ can also bridge the disjoint components and define a valid. Specifically, for two distributions with densities $p_d(x)$ and $q_\theta(x)$ with supports $\mathscr{X}_{p_d}, \mathscr{X}_{q_\theta} \subseteq \mathbb{R}^d$, we can choose $k(\tilde{x}|x) = \mathrm{N}(x, \sigma^2)$ and let

$$\tilde{p}_d(\tilde{x}) = \int_{\mathscr{X}_{p_d}} k(\tilde{x}|x) p_d(x)\,\mathrm{d}x \quad \tilde{q}_\theta(\tilde{x}) = \int_{\mathscr{X}_{q_\theta}} k(\tilde{x}|x) q_\theta(x)\,\mathrm{d}x. \tag{3.12}$$

We follow Equation 2.51 and define *Spread Fisher Divergence*($\widetilde{\mathrm{FD}}$) as

$$\widetilde{\mathrm{FD}}_k(p_d||q_\theta) \equiv \mathrm{FD}(\tilde{p}_d||\tilde{q}_\theta), \tag{3.13}$$

The convolution transform makes $\tilde{p}_d$ and $\tilde{q}_\theta$ have support $X_{\tilde{p}_d} = X_{\tilde{q}_\theta} = \mathbb{R}^d$ (which is a connected space) and $\widetilde{\mathrm{FD}}_k(p_d||q_\theta) \equiv \mathrm{FD}(\tilde{p}_d||\tilde{q}_\theta)$ is a valid discrepancy, i.e. $\widetilde{\mathrm{FD}}_k(p_d||q_\theta) = 0 \Leftrightarrow \tilde{p}_d = \tilde{q}_\theta \Leftrightarrow p_d = q_\theta$. Similar to the FD, we can rewrite the $\widetilde{\mathrm{FD}}$ as

$$\widetilde{\mathrm{FD}}_k(p_d||p_\theta) = \frac{1}{2}\int_{\mathbb{R}^d} \tilde{p}_d(\tilde{x}) \left\| s_{\tilde{p}_d}(\tilde{x}) - s_{\tilde{q}_\theta}(\tilde{x}) \right\|_2^2 d\tilde{x} \tag{3.14}$$

$$= \frac{1}{2}\int_{\mathbb{R}^d} \tilde{p}_d(\tilde{x}) \left(s^2_{\tilde{q}_\theta}(\tilde{x}) + 2\nabla_{\tilde{x}} s_{\tilde{q}_\theta}(\tilde{x}) \right) d\tilde{x} + const., \tag{3.15}$$

where the constant terms are independent of the model parameters. However, for an energy-based model $q_\theta(x) = e^{-f_\theta(x)}/Z(\theta)$, the spread model $\tilde{q}_\theta(\tilde{x}) = \frac{1}{Z(\theta)\sqrt{2\pi\sigma^2}} \int e^{-f_\theta(x) - \frac{1}{2\sigma^2}(\tilde{x}-x)^2}\,\mathrm{d}x$ has an intractable score, which makes the direct training infeasible.

Denoising Score Matching

Alternatively, one can directly parameterize $\tilde{q}_\theta(x)$ to be an energy-based model

$$\tilde{q}_\theta(\tilde{x}) = \exp(-f_\theta(\tilde{x}))/Z(\theta), \tag{3.16}$$

which leads to the denoising score matching (DSM) [Vincent, 2011] objective introduced in Section 2.6.1:

$$\mathrm{FD}(\tilde{p}_d||\tilde{q}_\theta) \doteq \frac{1}{2}\iint p(\tilde{x}|x)p_d(x)\left\|\frac{\tilde{x}-x}{\sigma^2}+s_{\tilde{q}_\theta}(\tilde{x})\right\|_2^2 d\tilde{x}\,\mathrm{d}x. \tag{3.17}$$

where we use $\doteq$ to denote equivalent up to a constant that is independent of θ. However, for a fixed $\sigma > 0$, the DSM objective is not a consistent objective to learn the underlying data distribution p_d since $\mathrm{FD}(\tilde{p}_d||\tilde{q}_\theta) = 0 \implies \tilde{p}_d = \tilde{q}_\theta \neq p_d$. A common solution is to anneal $\sigma \to 0$ during training. However, Equation 3.17 is not defined when $\sigma = 0$ since the division in Equation 3.17 will make $(\tilde{x}-x)/\sigma^2$ unbounded, which results in an inconsistent objective and annealing the noise won't fix the blindness problem.

To see this, we use a deep energy-based model with a 3-layer feedforward neural network with 30 hidden units and a tanh activation function to learn the toy mixture of two Gaussian distributions described in Section 1. We train the model with Adam optimizer with a learning rate $3\times e^{-4}$ for 10k iterations and batch size 300. We add convolutional Gaussian noise to the data samples with a standard deviation of 3.0 and anneal to 0 by multiplying by 0.9999 at each iteration. The noise at the end of training has a standard deviation of less than 0.001. In Figure 3.4 we plot the learned density during training. We find that when the noise is big the model can identify the correct mixture co-efficient, but when the noise is close to 0, the model fails to capture the correct mixing proportions. We also plot the density estimation results with vanilla FD and the proposed MFD in Figure 3.5a and 3.5b and we find that the density estimation with MFD achieves the best performance.

Additionally, unlike the mixture construction, if we directly assume $\tilde{q}_\theta(\tilde{x}) = e^{-f_\theta(\tilde{x})}/Z(\theta)$, the underlying 'correct' model $q_\theta(x)$ can not be recovered from $\tilde{q}_\theta(\tilde{x})$

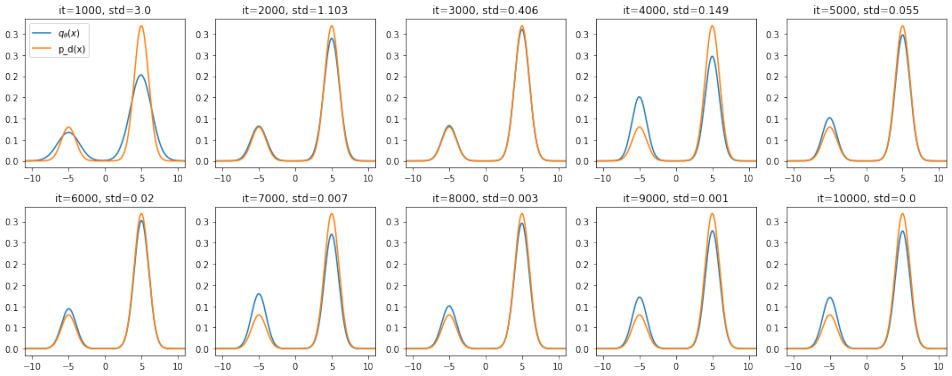
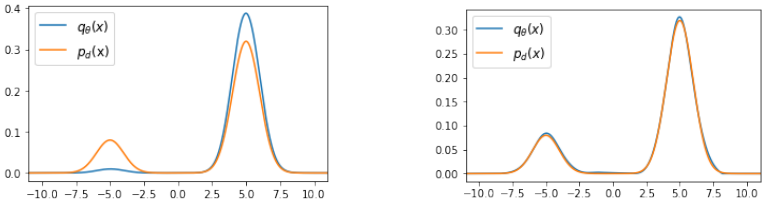


Figure 3.4: FD with training data noise annealing.



(a) FD (b) MFD

Figure 3.5: Density Estimation with FD and MFD.

even if we know $Z(\theta)$, so the DSM is also not directly applicable in this case. We leave the investigation of how to sample from the clean model $q_\theta(x)$, given the noisy energy-based model $\tilde{q}_\theta(\tilde{x})$, for future work.

3.6 Discussions

In this chapter, we explored the nuances of the Fisher divergence and its shortcomings in differentiating between mixture distributions with varying mixture proportions. Building on our theoretical findings, we proposed the mixture Fisher divergence as a preliminary remedy, which exhibited encouraging results in our toy experiments.

However, when transitioning from these elementary datasets to more intricate high-dimensional distributions, such as natural images, a host of challenges emerge:

- The score-matching objective, as highlighted by Equation 3.6, necessitates the computation of the Hessian's trace. This becomes increasingly prohibitive, both in terms of computational power and memory usage, especially for large dimensional distributions.

- The selection of $m(x)$ is pivotal, requiring that $m(x)$ should connect the disconnected regions of the target data distribution. For high-dimensional datasets, such as images, the inherent structure often resides on a lower-dimensional manifold within a vast dimensional space. This poses significant challenges in devising an efficient $m(x)$ that adeptly bridges diverse modes.

- The proposed methodology necessitates the estimation of the normalization constant $Z(\theta)$ post-training to ascertain the distribution's density. This estimation proves to be daunting for expansive high-dimensional distributions.

Addressing these challenges are essential direction for future research in this domain. We will leave a detailed exploration of how to practically apply and scale this method for high-dimensional data to subsequent work.

End of Part I

This marks the conclusion of the first segment of our thesis. In this part, we have delved into the intricacies of the KL divergence, optimizing it for effective training of implicit generative models tailored for non-a.c. distributions (Chapter 2). We also addressed the critical challenges posed by the "blindness problem" inherent to the Fisher divergence (Chapter 3). Our investigations in this section contribute to enhancing the training methodologies for generative models across varied contexts.

In addition to model training, generalization is also a central theme in the realm of classic machine learning research. However, this pivotal attribute is less explored in the context of generative modeling. Moreover, both its definition and practical implications remain somewhat nebulous. Thus, in the second segment of our thesis, we will shift our focus from the theoretical frameworks of training techniques to the pragmatic facets of model generalization. We will investigate the domains of in-distribution and out-of-distribution generalizations, harnessing our research findings to amplify practical applications, especially in the areas of lossless compression and OOD detection.

Part II

Generalizations of Generative Models

Chapter 4

In-distribution Generalization of Variational Auto-Encoder

In this Chapter, we study the generalization of a popular class of probabilistic model - the Variational Auto-Encoder (VAE). We discuss the two generalization gaps that affect VAEs and show that overfitting is usually dominated by amortized inference. Based on this observation, we propose a new training objective that improves the generalization of amortized inference. We demonstrate how our method can improve performance in the context of image modeling and lossless compression.

4.1 Introduction of Variational Auto-Encoder

A popular type of probabilistic model is the Variational Auto-Encoder (VAE) [Kingma and Welling, 2014, Rezende et al., 2014], which assumes a latent variable model $p_\theta(x) = \int p_\theta(x|z)p(z)dz$. For a nonlinear parameterization of $p_\theta(x|z)$ (e.g. a deep neural network), the evaluation of $\log p_\theta(x)$ involves solving an intractable integration over z. In this case, the evidence lower bound (ELBO) can be used to side-step the intractability

$$\int \log p_\theta(x) p_d(x)\,\mathrm{d}x \geq \iint \big(\log p_\theta(x,z) - \log q_\phi(z|x)\big) q_\phi(z|x) p_d(x) dz\,\mathrm{d}x \tag{4.1}$$

$$\equiv \int \mathrm{ELBO}(x,\theta,\phi) p_d(x)\,\mathrm{d}x, \tag{4.2}$$

where $q_\phi(z|x)$ is a variational posterior parameterized by a neural network with parameter ϕ. The use of an approximate posterior of the form $q_\phi(z|x)$ is called *amortized inference*. To better understand this objective, we can rewrite the expected ELBO as the following

$$\int \text{ELBO}(x,\theta,\phi)p_d(x)\,\mathrm{d}x = \int \left(\log p_\theta(x) - \text{KL}(q_\phi(z|x)||p_\theta(z|x))p_d(x)\right)\mathrm{d}x \quad (4.3)$$

$$= \underbrace{\int \log p_\theta(x)p_d(x)\,\mathrm{d}x}_{\text{model learning}} - \underbrace{\int \text{KL}(q_\phi(z|x)||p_\theta(z|x))p_d(x)\,\mathrm{d}x}_{\text{amortized inference}}, \quad (4.4)$$

We denote the posterior family of $q_\phi(z|x)$ as $\mathscr{Q}$, which is indexed by a finite-dimensional θ [Wang and Blei, 2019]. If $\mathscr{Q}$ is flexible enough such that the true posterior $p_\theta(z|x) \in \mathscr{Q}$, where $p_\theta(z|x) \propto p_\theta(x|z)p(z)$, then in the optimum of Equation 4.3, we have $\text{KL}(q_\phi(z|x)||p_\theta(z|x)) = 0 \Rightarrow q_\phi(z|x) = p_\theta(z|x)$ for $x \sim p_d(x)$ and the ELBO will be equal to the log-likelihood $\text{ELBO}(x,\theta,\phi) = \log p_\theta(x)$ [Kingma and Welling, 2014, Blei et al., 2017]. Many methods have been developed to increase the flexibility of $\mathscr{Q}$, e.g. adding auxiliary variables [Agakov and Barber, 2004, Maaløe et al., 2016] or flow-based methods [Challis and Barber, 2012, Rezende and Mohamed, 2015], to obtain a tighter ELBO.

Recent works [Townsend et al., 2019, 2020, Kingma et al., 2019] have successfully applied VAE style models to lossless compression realizing impressive performance. In this setting, the average compression length on the test data set is approximately equal to $-\frac{1}{M}\sum_{m=1}^{M}\text{ELBO}(x'_m,\theta,\phi)$. Hence the better the test ELBO indicates the better the compression performance. This motivates us to study the factors that affect the generalization of VAEs and find practical ways to improve the generalization of VAEs.

In the following section, we show the generalization of VAEs is affected by both the generative model (decoder) and the amortized inference network (encoder); and propose a new training objective that can improve the generalization of the amortized inference without changing the model itself. Additionally, we demonstrate how to improve the compression rate in a practical lossless compression system without sacrificing any computation speed.

4.2 Generalizations of VAEs

During training, we only have access to a finite dataset $\mathscr{X}_{train}$, which leads to the following empirical ELBO approximation:

$$\int \text{ELBO}(x,\theta,\phi)p_d(x)\,\mathrm{d}x \approx \frac{1}{N}\sum_{n=1}^{N}\text{ELBO}(x_n,\theta,\phi), \tag{4.5}$$

which can be further represented as a combination of a *model empirical approximation* and an *amortized inference empirical approximation*:

$$\frac{1}{N}\sum_{n=1}^{N}\text{ELBO}(x,\theta,\phi) = \frac{1}{N}\sum_{n=1}^{N}\log p_\theta(x_n) - \frac{1}{N}\sum_{n=1}^{N}\text{KL}(q_\phi(z|x_n)||p_\theta(z|x_n)). \tag{4.6}$$

Therefore, if either the decoder $p_\theta(x|z)$ or the encoder $q_\phi(z|x)$ is overly flexible, it can cause the VAE to overfit to the training data. We then define the *ELBO generalization gap* (EGG) as the difference between the training and test ELBO

$$\text{EGG} \equiv \underbrace{\frac{1}{N}\sum_{N=1}^{N}\text{ELBO}(x_n,\theta_N^*,\phi_N^*)}_{\text{Training ELBO}} - \underbrace{\frac{1}{M}\sum_{m=1}^{M}\text{ELBO}(x_m,\theta_N^*,\phi_N^*)}_{\text{Test ELBO}}, \tag{4.7}$$

where θ_N^*, ϕ_N^* are defined as the optimal parameters for training the empirical ELBO

$$\theta_N^*,\phi_N^* = \arg\max_{\theta,\phi}\frac{1}{N}\sum_{n=1}^{N}\text{ELBO}(x_n,\theta,\phi). \tag{4.8}$$

By the decomposition in Equation 4.4, ϕ_N^* is also the optimal parameter of the empirical variational inference objective

$$\phi_N^* = \arg\min_{\phi}\frac{1}{N}\sum_{n=1}^{N}\text{KL}\left(q_\phi(z|x_n)||p_{\theta_N^*}(z|x_n)\right). \tag{4.9}$$

For simplicity, considering a flexible amortized inference network, we can assume that for any training data point $x_n \in \mathscr{X}_{train}$

$$q_{\phi_N^*}(z|x_n) = \arg\min_{q\in\mathscr{Q}}\text{KL}(q_\phi(z|x_n)||p_{\theta_N^*}(z|x_n)) \equiv q^*(z|x_n), \tag{4.10}$$

where $q^*(z|x_n)$ is the realizable optimal posterior (in the $\mathscr{Q}$ family) for x_n[1]. However, when $q_{\phi^*}(z|x_n)$ overfits to $\mathscr{X}_{train}$, $q_{\phi^*}(z|x'_m)$ may not be a good approximation to the true posterior $p_{\theta^*_N}(z|x'_m)$ for test data $x'_m \in \mathscr{X}_{test}$. This discrepancy can lead to a suboptimal test ELBO.

To illustrate the generalization of the amortized inference, we denote the ϕ^*_M as the optimal realizable parameter of the amortized inference for the test dataset:

$$\phi^*_M = \arg\min_{\phi} \frac{1}{M} \sum_{m=1}^{M} \mathrm{KL}\left(q_\phi(z|x'_m)||p_{\theta^*_N}(z|x'_m)\right). \tag{4.11}$$

For a flexible $q_\phi(z|x)$, we make a similar assumption to Equation 4.10, such that $q_{\phi^*_M}(z|x'_m)$ can generate the optimal posterior within the variational family $\mathscr{Q}$:

$$q_{\phi^*_M}(z|x'_m) = \arg\min_{q\in\mathscr{Q}} \mathrm{KL}(q_\phi(z|x'_m)||p_{\theta^*_N}(z|x'_m)) \equiv q^*(z|x'_m), \tag{4.12}$$

We then define the *amortized inference generalization gap* (AIGG) as the difference between two averaged KL divergences

$$\mathrm{AIGG} \equiv \frac{1}{M}\sum_{m=1}^{M} \mathrm{KL}(q_{\phi^*_N}(z|x'_m)||p_{\theta^*_N}(z|x'_m)) - \frac{1}{M}\sum_{m=1}^{M} \mathrm{KL}(q_{\phi^*_M}(z|x'_m)||p_{\theta^*_N}(z|x'_m)). \tag{4.13}$$

Intuitively, AIGG assesses the proximity of the posterior $q_{\phi^*_N}(z|x'_m)$, — which is derived from the amortized network trained on $\mathscr{X}_{train}$ — to the optimal realizable posterior for $x'_m \in \mathscr{X}_{test}$. If the family $\mathscr{Q}$ is flexible enough such that $p_\theta(z|x'_m) \in \mathscr{Q}$ for every x'_m, then the AIGG simplifies to $\frac{1}{M}\sum_{m=1}^{M} \mathrm{KL}(q_{\phi^*_N}(z|x'_m)||p_{\theta^*_N}(z|x'_m))$.

Equivalently, the AIGG can also be written as the difference between two ELBOs with ϕ^*_M and ϕ^*_N respectively:

$$\mathrm{AIGG} \equiv \frac{1}{M}\sum_{M=1}^{M} \mathrm{ELBO}(x'_m, \theta^*_N, \phi^*_M) - \frac{1}{M}\sum_{m=1}^{M} \mathrm{ELBO}(x'_m, \theta^*_N, \phi^*_N), \tag{4.14}$$

It is important to emphasize that this gap cannot be reduced by simply using a more flexible variational family $\mathscr{Q}$. While this might reduce the $\mathrm{KL}(q_{\phi^*_N}(z|x_n)||p_{\theta^*_N}(z|x_n))$

[1]For a flexible amortized inference network, we assume that there is no amortization gap [Cremer et al., 2018], which means $q_{\phi^*_N}(z|x)$ can provide the optimal $q^*(z|x_n)$ for any training data $x_n \in \mathscr{X}_{train}$ - see Section 4.6 for further discussion.

for the training data $x_n \in \mathcal{X}_{train}$, it would not explicitly encourage better generalization performance on test data, see also [Shu et al., 2018].

The AIGG is caused by the overfitting of the amortized inference (encoder). To understand the generalization property of the decoder, we can further rewrite the EGG by subtracting and adding the term $\frac{1}{M}\sum_{m=1}^{M}\text{ELBO}(x'_m, \theta_N^*, \phi_M^*)$:

$$\begin{aligned}\text{EGG} = &\frac{1}{N}\sum_{n=1}^{N}\text{ELBO}(x_n, \theta_N^*, \phi_N^*) - \frac{1}{M}\sum_{m=1}^{M}\text{ELBO}(x'_m, \theta_N^*, \phi_M^*) \\ &+ \underbrace{\frac{1}{M}\sum_{m=1}^{M}\text{ELBO}(x'_m, \theta_N^*, \phi_M^*) - \frac{1}{M}\sum_{m=1}^{M}\text{ELBO}(x'_m, \theta_N^*, \phi_N^*)}_{\text{AIGG}}\end{aligned} \tag{4.15}$$

where the second row is the AIGG. Meanwhile, we define the expression in the first row, which is the difference between the training and test ELBO using the optimal amortized inference parameters ϕ_N^*, ϕ_M^* respectively, as the *Generative Model Generalization Gap* (GMGG)

$$\text{GMGG} = \underbrace{\frac{1}{N}\sum_{n=1}^{N}\text{ELBO}(x_n, \theta_N^*, \phi_N^*)}_{\text{Training ELBO with optimal inference}} - \underbrace{\frac{1}{M}\sum_{m=1}^{M}\text{ELBO}(x'_m, \theta_N^*, \phi_M^*)}_{\text{Test ELBO with optimal inference}} \tag{4.16}$$

With this in perspective, we can express the EGG as a combination of both gaps:

$$\text{EGG} = \text{GMGG} + \text{AIGG}, \tag{4.17}$$

This decomposition highlights that the VAE's generalization performance is influenced by the generalization capabilities of both the generative model and the amortized inference. To provide a more in-depth analysis, we'll be illustrating these gaps in the subsequent section.

4.2.1 Visualizations of the Generalization Gaps

The EGG can be easily visualized by plotting the difference between the training ELBO and the test ELBO. To visualize the GMGG and the AIGG, we need to know

the optimal amortized posterior $q_{\phi_M^*}(z|x'_m)$ for the test data $x'_m \in \mathscr{X}_{test}$, where

$$\phi_M^* = \min_{\phi} \frac{1}{M} \sum_{m=1}^{M} \mathrm{KL}(q_\phi(z|x'_m)||p_{\theta_N^*}(z|x'_m)) \tag{4.18}$$

$$= \max_{\phi} \frac{1}{M} \sum_{m=1}^{M} \int \left(\log p_{\theta_N^*}(x'_m, z) - \log q_\phi(z|x'_m)\right) q_\phi(z|x'_m) dz. \tag{4.19}$$

Adopting this optimal inference strategy effectively nullifies the impact of the AIGG. This in turn allows us to separately assess the contributions of both the generative model's generalization gap and the amortized inference generalization gap, offering insights into the nuances of model overfitting.

To visualize the generalization gaps, we trained a VAE on the Binary MNIST training dataset over 1,000 epochs and we saved the encoder/decoder parameter pair, denoted as (θ, ϕ), every 100 epoch. Detailed model specifications and training methods can be found in Section 4.4. Figure 4.1 shows the Bits-per-dimension (BPD)[2] of both training and testing dataset for every 100 epochs, the gap between the training ELBO (blue) and test ELBO (purple) is the EGG. From the figure, it's evident that the VAE model starts to overfit the training dataset, and this overfitting intensifies as the training duration extends.

Additionally, for each saved encoder/decoder parameters, we fix the decoder $p_\theta(x|z)$ and only train $q_\phi(z|x)$ for 1k epochs on the test data using Equation 4.19 to obtain the estimation of the optimal amortized posterior $q_{\phi_M^*}(z|x)$. This process yielded the test BPD associated with the optimal inference strategy, represented by the green line in Figure 4.1. The differential between the purple and green lines showcases the AIGG, while the remaining gap between the green and blue lines is indicative of the GMGG.

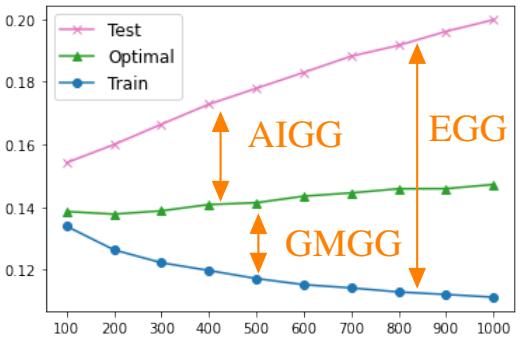


Figure 4.1: BDPs vs epochs. Visualization of the EGG = GMGG + AIGG.

[2] In the case of VAE, the BPD is defined as the negative ELBO (with a base 2 logarithm) normalized by the data dimension, lower BPD indicates higher ELBO.

Utilizing the test ELBO with the test-time optimal inference strategy, the test BPD (green) appears largely stable, showing just a minor increase during training. This pattern hints at the generative model's (decoder's) overfitting being less pronounced compared to that of the amortized inference network (encoder). Consequently, the primary source of significant overfitting is dominated by the overfitting of the amortized inference network.

Although the optimal inference strategy (training the encoder on the test dataset $\mathscr{X}_{test}$) can help reduce the EGG, but it will hinder the speed of both compression and decompression processes, which is one of the central concerns in compression tasks. Therefore, we now focus on improving the generalization of amortized inference *without* access to the test data at training time.

4.3 Consistent Amortized Inference

We now propose an *inference consistency requirement* which, if satisfied, would result in optimal generalization performance for amortized variational inference. Specifically when $p_\theta \to p_d$, the amortized posterior should converge to the true posterior $q_\phi(z|x) \to p_\theta(z|x)$[3] for every $x \sim p_d(x)$. Although this requirement seems natural for variational inference, the classic amortized inference training that is used for VAEs [Kingma and Welling, 2014] doesn't satisfy it. Recall the typical VAE empirical ELBO training objective

$$\frac{1}{N}\sum_{n=1}^{N} \log p_\theta(x_n) - \mathrm{KL}(q_\phi(z|x_n)||p_\theta(z|x_n)). \tag{4.20}$$

When the model is perfect $p_{\theta^*} = p_d$, the training criterion for $q_\phi(z|x)$

$$\min_\phi -\frac{1}{N}\sum_{n=1}^{N} \mathrm{KL}(q_\phi(z|x_n)||p_{\theta^*}(z|x_n)) \tag{4.21}$$

can still result in the amortized posterior $q_\phi(z|x)$ overfitting to the training data. In principle, one could also limit the network capacity and/or add an explicit regularizer to the parameters [Shalev-Shwartz and Ben-David, 2014] in an attempt to improve

[3] We assume the true posterior belongs to the variational family $p_\theta(z|x) \in \mathscr{Q}$.

the generalization. However, this still cannot satisfy the consistency requirement in principle because it still only uses the finite training dataset. Alternatively, there is another classic variational inference method that we now discuss, the wake-sleep training algorithm [Dayan et al., 1995, Hinton et al., 1995], which does in fact satisfy the proposed consistency requirement.

4.3.1 Wake-Sleep Training

Defining $q_\phi(x,z) = q_\phi(z|x)p_d(x)$ and $p_\theta(x,z) = p_\theta(x|z)p(z)$, the two phases of the wake-sleep training [Dayan et al., 1995, Hinton et al., 1995] can be written as minimizing two different KL divergences in both x and z space:

Wake phase model learning: $p_\theta(x|z)$ is trained by

$$\min_\theta \mathrm{KL}(q_\phi(x,z)||p_\theta(x,z)) = \max_\theta \int \mathrm{ELBO}(x,\theta,\phi)p_d(x)\,\mathrm{d}x + const., \tag{4.22}$$

where the integration over $p_d(x)$ is approximated with the training set. This is referred to as the *wake phase* since the model is trained on experience from the 'real environment', i.e. it uses true data samples from $p_d(x)$.

Sleep phase amortized inference: $q_\phi(z|x)$ is trained by

$$\min_\phi \mathrm{KL}(p_\theta(x,z)||q_\phi(x,z)) = \min_\phi \int \mathrm{KL}(p_\theta(z|x)||q_\phi(z|x))p_\theta(x)\,\mathrm{d}x \tag{4.23}$$

Leaving out the terms that are irrelevant to ϕ, the objective can be estimated with Monte-Carlo $-\int \log q_\phi(z|x'_m)p_\theta(x,z) \approx -\frac{1}{K}\sum_{k=1}^K \log q_\phi(z_k|x_k)$, where $z_k \sim p(z)$ and $x_k \sim p_\theta(x|z_k)$. This is referred to as the *sleep phase* because the samples from the model used to train q_ϕ are interpreted as dreamed experience. In contrast, the training criterion for the typical VAE amortized inference (Equation 4.6) uses the true data samples from p_d to train $q_\phi(z|x)$, which we refer to as *wake phase amortized inference*. We notice that if a perfect model $p_{\theta^*}(x) = p_d(x)$ is used in the sleep phase amortized inference, then it is equivalent to minimizing

$$\int \mathrm{KL}(p_\theta(z|x)||q_\phi(z|x))p_{\theta^*}(x)\,\mathrm{d}x = \int \mathrm{KL}(p_\theta(z|x)||q_\phi(z|x))p_d(x)\,\mathrm{d}x. \tag{4.24}$$

Therefore, the training of the inference network satisfies the *inference consistency requirement* since we can access infinite data from p_d by sampling from p_{θ^*}.

However, the wake-sleep algorithm presented lacks convergence guarantees [Dayan et al., 1995] and minimizing $\mathrm{KL}(p_\theta(z|x)||q_\phi(z|x))$ in the sleep phase doesn't necessarily encourage an improvement to the ELBO, which directly relates to the compression rate in the lossless compression application [Townsend et al., 2019]. Therefore, in the next section, we propose a new variational inference scheme: *reverse sleep amortized inference* and demonstrate how it helps improve the generalization of the inference network in practice.

4.3.2 Reverse Sleep Amortized Inference

In the inference time, we propose to fix θ and train ϕ using the *reverse KL divergence*

$$\min_\phi \int \mathrm{KL}(q_\phi(z|x)||p_\theta(z|x)p_\theta(x)) = \max_\phi \iint p_\theta(x)q_\phi(z|x)\log\frac{p_\theta(x,z)}{q_\phi(z|x)}dz\,\mathrm{d}x, \tag{4.25}$$

where the integration over $p_\theta(x)$ is approximated by Monte-Carlo using samples from the generative model $p_\theta(x)$. This reverse KL objective encourages improvements to the ELBO. When we have a perfect model $p_{\theta^*}(x) = p_d(x)$ the reverse sleep phase is equivalent to

$$\min_\phi \int \mathrm{KL}(p_{\theta^*}(z|x)||q_\phi(z|x))p_{\theta^*}(x)\,\mathrm{d}x = \min_\phi \int \mathrm{KL}(p_{\theta^*}(z|x)||q_\phi(z|x))p_d(x)\,\mathrm{d}x \tag{4.26}$$

which satisfies the *inference consistency requirement*.

The consistency requirement can also be validated empirically when the perfect model is known $p_{\theta^*}(x) = p_d(x)$. This can be achieved by using a pre-trained VAE as the true data generation distribution. Therefore, our method has two training stages:

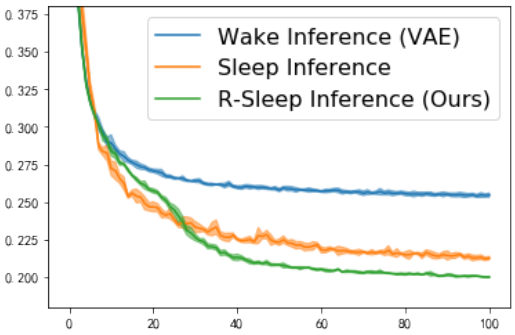


Figure 4.2: Test BPD vs epochs. We compare the consistency property between three amortized inference methods.

1. Pre-train a VAE using Equation 4.2 to fit the training dataset.

2. Fix the decoder and train the encoder using Equation 4.25 and the samples from the Pre-trained VAE.

Unlike the Wake-sleep training method, which alternates between two objectives in each gradient step and lacks a guarantee of algorithmic convergence, our approach optimizes a single divergence objective at each stage, which ensures algorithmic convergence.

As a demonstration, we first train a VAE to fit the binary MNIST problem. The VAE has the same structure as that used in Section 2 and is trained using Adam with $lr = 1\times10^{-3}$ for 100 epochs. After training, we treat the pre-trained decoder $p_{\theta'}(x|z)$ as the training data generator $p_d(x) \equiv \int p_{\theta'}(x|z)p(z)dz$. We then sample 10000 data samples from p_d to form a training set $\mathscr{X}_{train}$ and 1000 samples to form a test set $\mathscr{X}_{test}$. We then train a new $q_\phi(z|x)$ with 1) wake phase inference (VAE) 2) (forward) sleep inference and 3) reverse sleep inference. The network is trained using Adam with $lr = 1\times10^{-3}$ for 100 epochs. Figure 4.2 shows the test BPD calculated after every training epoch. We can see the sleep phase outperforms the wake phase and the reverse sleep inference achieves the best BPD. Intuitively, this is because both the forward and reverse sleep inference use the true model to generate additional training data whereas the wake inference only has access to the finite training dataset X_{train}.

4.3.3 Reverse Half-asleep Inference with Imperfect Models

In practice, our model will not be perfect $p_\theta \neq p_d$. Empirically we find that samples from even a well-trained model p_θ may not always be sufficiently like the samples from the true data distribution. This can lead to degradation in the performance of the inference network when using the reverse-sleep approach. For this reason, we propose to use a mixture distribution between the model and the empirical training data distribution as follows

$$\int \mathrm{KL}\big(q_\phi(z|x)||p_\theta(z|x)\big)m(x)\,\mathrm{d}x \quad \text{where} \quad m(x) \equiv \alpha p_\theta(x) + (1-\alpha)\hat{p}_d. \tag{4.27}$$

When $\alpha = 0$, it reduces to the standard approach used in VAE training. When $\alpha = 1$, we recover the reverse sleep method (Equation 4.25). We find that a setting of $\alpha = 0.5$ works well in practice. This balances samples from the true underlying data distribution with samples from the model.

We thus refer to this method as *reverse half-asleep* since it uses both data and model samples to train the amortized posterior. Intuitively, we can rewrite the Equation 4.27 as a sum of two positive terms

$$\alpha \int \mathrm{KL}\big(q_\phi(z|x)||p_\theta(z|x)\big)\hat{p}(x)\,\mathrm{d}x + (1-\alpha)\int \mathrm{KL}\big(q_\phi(z|x)||p_\theta(z|x)\big)p_\theta(x)\,\mathrm{d}x. \quad (4.28)$$

Therefore, the optimal of this objective will make the first term 0, which is the same requirement as the classic amortized inference (Equation 4.6). The second term, which is equivalent to the reverse sleep amortized inference (Equation 4.25), can encourage the inference consistency requirement: when $p_\theta = p_d$, the optimal of the second term will set $q_\phi(z|x) = p_\theta(z|x)$ for any $x \sim p_d(x)$. When p_θ is not perfect, the second term can be seen as a regularizer added to the classic amortized inference objective, which can be used to penalize the hypothesis space of the amortized network [Shalev-Shwartz and Ben-David, 2014].

To compare with different α, we first fit a VAE (with the same structure as that used in Figure 2) to the Binary MNIST dataset, and then train the amortized posterior using sleep inference (Equation 4.23) and three different α for additional 100 epochs using Adam with learning rate 3×10^{-4}. Figure 4.3 shows the test BPD comparison. We find the proposed reverse half-asleep method ($\alpha = 0.5$) outperforms the reversed sleep method ($\alpha = 1$), whereas the standard amortized inference training in VAE ($\alpha = 0$) leads to overfitting of the inference

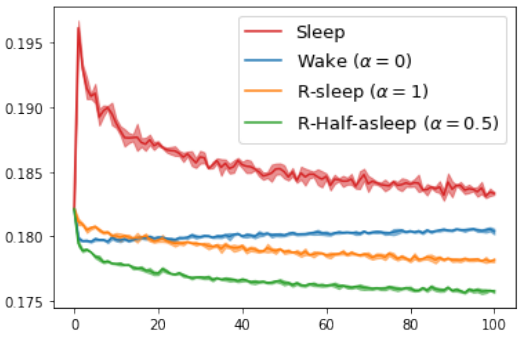


Figure 4.3: Test BPD comparisons of Amortized inference with different α. We find the Reverse Half-asleep method ($\alpha = 0.5$) achieves the best BPD. The mean and std are calculated with three random seeds.

network. We also plot the sleep inference curve, whose BPD is less competitive since it does not directly optimise the ELBO.

4.4 Generalization Experiments

We apply the reverse half-asleep to improve the generalization of VAEs on three different datasets: binary MNIST, grey MNIST [LeCun, 1998] and CIFAR10 [Krizhevsky et al., 2009]. For binary and grey MNIST, we use latent dimension 16/32 and neural nets with 2 layers of 500 hidden units in both the encoder and decoder. We use Bernoulli $p(x|z)$ for binary MNIST and discretized logistic distribution for grey MNIST. We train the VAE with the usual amortized inference approach using Adam with $lr = 3\times10^{-4}$ for 1000 epochs and save the model every 100 epochs. We then use the saved models to 1) evaluate the test data sets, 2) conduct optimal inference by training $q_\phi(z|x)$ on the test data and 3) run the reverse half-asleep method before calculating the test BPD. For the reverse half-asleep, we train the amortized posterior for 100 epochs with Adam and $lr = 5\times10^{-4}$. To sample from $p_\theta(x)$, we firstly sample $z' \sim p(z)$ and sample $x' \sim p(x|z = z')$. For the optimal inference strategy, we train the amortized posterior with the same optimization scheme on the test data set for an additional 500 epochs to ensure the same number of gradient steps are conducted (since the training set is 5 times as big as the test set). Figure 4.4a and 4.4b show the test BPD comparisons of binary and grey MNIST respectively and demonstrate that our approach does not require further training on the test data to improve generalization performance.

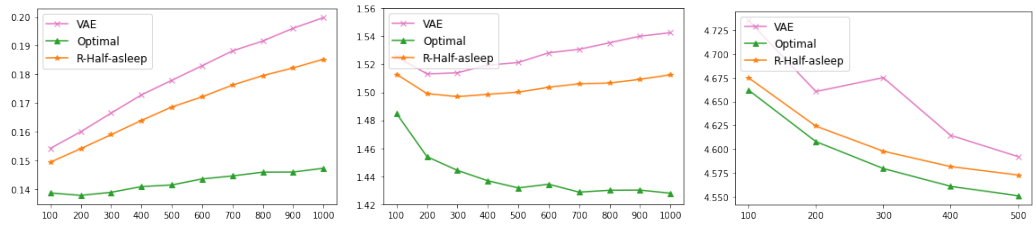
For CIFAR10, we use the convolutional ResNet [He et al., 2016, Van Den Oord et al., 2017] with 2 residual blocks and latent size 128. The observational distribution is a discretized logistic distribution with linear autoregressive parameterization within channels. We train the VAE for 500 epochs with Adam and $lr = 5\times10^{-4}$ and save the model every 100 epoch. The pre-trained VAE achieves 4.592 BPD on the CIFAR10, which is comparable with other single latent VAE models reported in [Van Den Oord et al., 2017]: 4.51 BPD with a VAE with latent dimension 256 and 4.67 BPD with a discrete latent VAE (VQVAE).

Ideally, when the VAE model converges to the true distribution $p_\theta \to p_d$, the aggregate posterior $q_\phi(z) = \int q_\phi(z|x)p_d(x)\,\mathrm{d}x$ will match the prior $p(z)$. However, for a complex distribution like CIFAR10, a significant mismatch between $q_\phi(z)$ and $p(z)$ is usually observed in practice [Zhao et al., 2017, Dai and Wipf, 2019]. In this case, the sample x' that is generated using a latent sample from the prior $x' \sim p_\theta(x|z')$, where $z' \sim p(z)$, may be blurry or invalid. A common solution is to train another model, e.g. a VAE [Dai and Wipf, 2019] or a PixelCNN [Van Oord et al., 2016, Van Den Oord et al., 2017] to approximate $q_\phi(z)$. In our case, we instead directly sample from $q_\phi(z)$ rather than $p(z)$ to generate samples in Equation 4.25, which can be done by first sampling $x' \sim p_d(x)$ (from the training dataset) and then sample $z' \sim q_\phi(z|x = x')$. This scheme still results in a consistent training objective since $q_{\phi^*}(z) = p(z)$ for the optimal posterior $q_{\phi^*}(z|x)$. We use Adam with $lr = 1\times10^{-5}$ and train the reverse half-asleep inference for 100 epochs on the training data and train the optimal inference strategy for 500 epochs on the test data, see Figure 4.4c for the result. We find the proposed reverse half-asleep training approach (with sampling from $q_\phi(z)$) consistently improves the generalization performance of the amortized posterior. Conversely, our experiments revealed that when training the encoder using samples generated from the prior $p(z)$, the resulting BPDs consistently exceed 5.0 across all tested epochs. This suggests that the quality of samples from the generator plays a pivotal role in the efficacy of the proposed reverse-half-asleep inference method.

Although the proposed method is still worse than the optimal inference, it has the advantage of not requiring encoder fine-tuning on test data during test time. This is particularly critical for applications prioritizing test-time inference speed, such as lossless compression, which we will discuss in Section 4.5.

4.4.1 Comparisons with Regularization Methods

Recent work [Shu et al., 2018] proposed to alleviate overfitting of amortized inference by optimizing a linear combination between the traditional amortized inference



(a) Binary MNIST (b) Grey MNIST (c) CIFAR10

Figure 4.4: We present test BPD comparisons across three inference methods: amortized inference (VAE), the optimal inference strategy, and reverse half-asleep inference, applied to three datasets. The x-axis indicates the number of training epochs. For the MNIST experiments, as shown in Figures a and b, all samples are derived from the predefined prior $p(z)$. In contrast, for the CIFAR experiment, samples are produced using the aggregate posterior $q_\phi(z)$. Notably, when employing $p(z)$ as the prior in CIFAR, the BPD values for the Reverse-Half-Asleep method consistently exceed 5. To maintain clarity, these results are omitted from Figure c.

(Equation 4.6) and a denoising objective

$$\alpha \int \mathrm{KL}(q_\phi(z|x+\varepsilon)||p_\theta(z|x))p(\varepsilon)d\varepsilon + (1-\alpha)\mathrm{KL}(q_\phi(z|x)||p_\theta(z|x)), \tag{4.29}$$

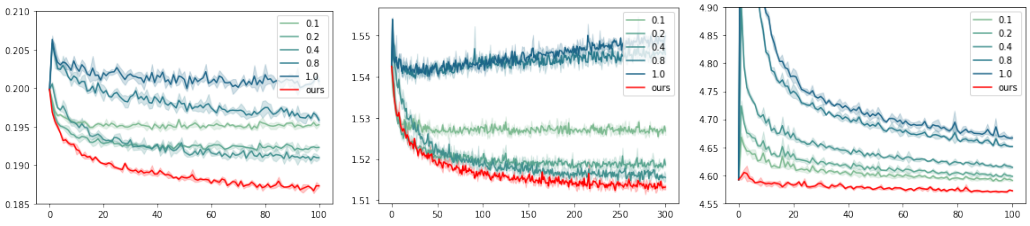
where $p(\varepsilon) = \mathcal{N}(0, \sigma^2 I)$. We compare this regularizer to our method by training the amortized posterior of VAEs for an additional 100, 300 and 100 epochs on Binary, Grey MNSIT and CIFAR respectively. For the denoising regularizer, we use the same linear combination weight $\alpha = 0.5$ as that used in Equation 4.27 and vary $\sigma \in \{0.1, 0.2, 0.4, 0.6, 0.8, 1.0\}$, see Table 4.1 for the comparisons. For MNIST, we find $\sigma \in \{0.1, 0.2, 0.4\}$ improves the generalization but larger noise levels hurt the performance. For CIFAR10, only $\sigma = 0.1$ can slightly improve the generalization by 0.001 BPD. In contrast, our method consistently achieves better generalization performance without tuning any hyper-parameters, see Figure 4.5 for the test BPD (evaluated every training epoch, the mean/std are calculated with 3 random seeds). Compared to the denoising approach, one limitation of our method is the requirement of model samples, which is more computationally expansive during training.

Since the decoder is shared and fixed in all comparisons, better test ELBO indicates the predicted $q_\phi(z|x')$ is closer to the true posterior $p_\theta(z|x')$ under the KL divergence (see Equation 4.3, higher ELBO with fixed θ indicates $\mathrm{KL}(q_\phi(z|x)||p_\theta(z|x))$ is smaller). Therefore, the proposed method can also benefit a range of tasks that

require accurate prediction of the posterior on the test data. In Appendix C.2 and C.3, we demonstrate our method can provide better proposal distributions for the Importance Weighted Auto-Encoder (IWAE) [Burda et al., 2015] and also improve the representation learning performance for down-stream classification tasks.

Table 4.1: Test BPD comparisons with Denoising Regularizer [Shu et al., 2018].

Methods	VAE	$\sigma = 0.1$	$\sigma = 0.2$	$\sigma = 0.4$	$\sigma = 0.8$	$\sigma = 1.0$	Ours
Binary MNIST	0.200	0.195	0.192	0.191	0.196	0.201	**0.187**
Grey MNIST	1.543	1.527	1.519	1.515	1.545	1.550	**1.513**
CIFAR10	4.592	4.591	4.598	4.614	4.651	4.667	**4.572**



(a) Binary MNIST (b) Grey MNIST (c) CIFAR 10

Figure 4.5: Test BPD evaluated after every training epoch. We find, compared to the denoising regularizer, the proposed amortized inference training scheme consistently achieves better generalization performance in all tasks.

4.5 Application of Lossless Compression

Lossless compression is an important application of VAEs where generalization plays a key role in the compression rate. Given a trained VAE, a practical compressor can be efficiently implemented using the Bits Back algorithm [Hinton and Van Camp, 1993, Townsend et al., 2019] with the ANS coder [Duda, 2013], see the next section for a brief introduction.

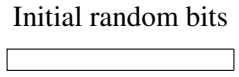
4.5.1 Introduction of VAE-based Lossless Compression

Given a discrete Latent VAE model specified by the probability mass functions (PMFs) $\{p_\theta(x|z), q_\phi(z|x), p(z)\}$ and a target data x' to compress. A naive strategy is to first generate a sample $z' \sim q_\phi(z|x')$ and then encode x' with $p_\theta(x|z')$. We also encode z' with distribution $\log p(z)$, so the total code length is then We also encode

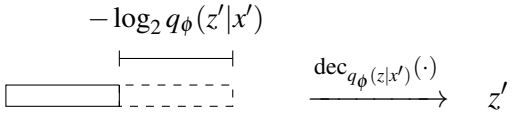
z' with distribution $\log p(z)$, so the total code length is then

$$-\log_2 p_\theta(x'|z') - \log_2 p(z'), \tag{4.30}$$

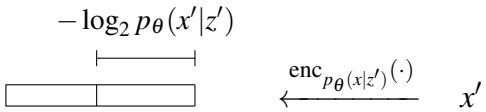
which is larger than the optimal code length $-\log_2 p(x')$ by $-\log_2 p_\theta(z'|x')$ bits. To achieve the optimal code length, a key observation is that the sampling process $z' \sim q_\phi(z|x')$ can be done by decoding random bits using the distribution $q_\phi(z|x')$. Specifically, we assume that we can access a message that already contains random bits, which we visualize as the following figure[4].



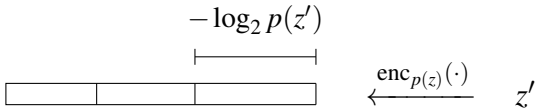
In the encoding stage, we first sample z' form $q_\phi(z|x')$ by decoding random bits with distribution $q_\phi(z'|x')$, so the message length decreased by length $-\log_2 q_\phi(z'|x')$.



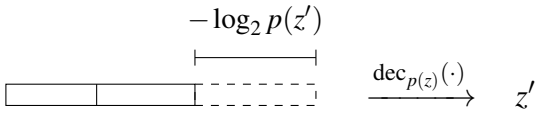
We then encode x' with distribution $p_\theta(x'|z')$, so the message is increased by length $-\log_2 p_\theta(x'|z')$.



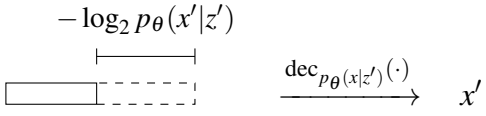
Finally, we encode z' with distribution $p(z)$ and the message is increased bylength $-\log p(z')$.



In the decoding stage, we first decode z' using $p(z)$.

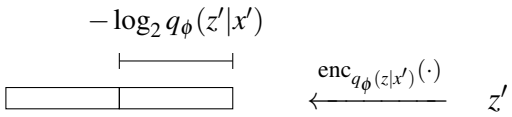


We then decode x' with distribution $p_\theta(x|z')$.



Finally, we encode the random bits 'back' to the stack to recover the initial message.

[4]The visualization is based on [Townsend et al., 2019].



Therefore, the 'net' message length to compress data x' with a VAE is

$$-\log_2 p_\theta(x|z') - \log_2 p(z') + \log q_\phi(z'|x'), \tag{4.31}$$

which is a one-sample estimation of the ELBO and is optimal (equal to $-\log_2 p_\theta(x')$) when the amortized variational posterior is equal to the true posterior $q_\phi(z|x') = p_\theta(z|x')$.

This scheme and also be extended to continuous latent z with negligible cost by quantizing the PDF $p(z)$ and $q_\phi(z)$ into PMF to conduct the compression. See [Townsend et al., 2019] for details. This 'Bits Back' coding method was first introduced as a thought experiment in [Wallace, 1990, Hinton and Van Camp, 1993] and was later implemented by [Frey and Hinton, 1996] with an AC coder. Recently, [Townsend et al., 2019] proposed to implement the Bits Back with ANS [Townsend et al., 2020] and a VAE model, which allows great improvement of both the compression rate and the computational efficiency. We refer the reader to [Townsend et al., 2019] for other practical considerations and implementation details.

4.5.2 Improving the Generalization of VAE-based Compression

In Algorithm 1, we summarize the Bits Back procedure with amortized inference to compress/decompress a test data point x' to a stack that contains bit string messages. The resulting code length for data x' is approximately equal to the negative ELBO

$$-\log_2 p_\theta(x'|z') - \log_2 p(z') + \log_2 q_\phi(z'|x'). \tag{4.32}$$

We have shown that $q_\phi(z|x)$ may overfit the training data, degrading compression performance. To improve the compression BPD, the optimal inference strategy can also be applied in the Bits Back algorithm. In the compression stage, we can

Algorithm 1 Bits Back with Amortized Inference.

Comp./decomp. stages share $\{p_\theta(x|z), q_\phi(z|x), p(z)\}$.

Compression

Draw sample $z' \sim q_\phi(z|x')$ from the stack.
Encode $x' \sim p_\theta(x|z')$ onto the stack.
Encode $z' \sim p(z)$ onto the stack.

Decompression

Decode $z' \sim p(z)$ from the stack.
Decode $x' \sim p_\theta(x|z')$ from the stack.
Encode $z' \sim q_\phi(z|x')$ onto the stack.

train ϕ by

$$\phi^* = \arg\max_{\phi} \text{ELBO}(x', \theta, \phi). \tag{4.33}$$

When the $q_\phi(z|x')$ is parameterized to be a Gaussian, we can just take ϕ to be the mean and standard deviation $\mathcal{N}(\phi_\mu, \phi_\sigma^2)$, which only contains two training parameters. In the decompression stage, we observe that the compressed data x' is recovered before the $q_\phi(z|x')$ is used to encode z'. Therefore, we can also train the $q_\phi(z|x')$ using the recovered x' to maximize the test ELBO. If the optimization procedure is the same as that used in the compression stage, we will get the same $q_{\phi^*}(z|x')$. In practice, we need to pre-specify the number of gradient descent steps K. When K is large, we recover the optimal inference strategy and the code length is approximately

$$-\log_2 p_\theta(x'|z') - \log_2 p(z') + \log_2 q_{\phi^*}(z'|x'). \tag{4.34}$$

This observation was initially introduced in the paper [Hinton and Van Camp, 1993] and then applied in the context of both lossy [Yang et al., 2020] and lossless compression [Ruan et al., 2021]. Furthermore, by varying the optimization steps K in the optimal inference, we can trade-off between the speed and the compression rate. This is valuable for practical applications with different speed/rate requirements. See Algorithm 2 for a summary of the Bits Back algorithm with K-step optimal inference.

Algorithm 2 Bits Back with K-step Optimal Inference

Comp./decomp. stages share $\{p_\theta(x|z), q_\phi(z|x), p(z)\}$ and the optimization procedure of Equation 4.33.

Compression

Take K gradient steps $\phi \to \phi^K$ with Equation 4.33.
Draw sample $z' \sim q_{\phi^K}(z|x')$ from the stack.
Encode $x' \sim p_\theta(x|z')$ onto the stack.
Encode $z' \sim p(z)$ onto the stack.

Decompression

Decode $z' \sim p(z)$ from the stack.
Decode $x' \sim p_\theta(x|z')$ from the stack.
Take K gradient steps $\phi \to \phi^K$ with Equation 4.33.
Encode $z' \sim q_{\phi^K}(z|x')$ onto the stack.

Although the optimal inference strategy can be used in lossless compression, it requires extra run-time for training at the compression stages. In contrast, our proposed reverse half-asleep inference scheme can improve the compression rate *without* sacrificing any speed. Additionally, our method can also provide a better initialization for the optimal inference strategy to allow a better trade-off between compression rate and speed.

We implement Bits Back with ANS [Duda, 2013] and compare the compression among four inference methods:

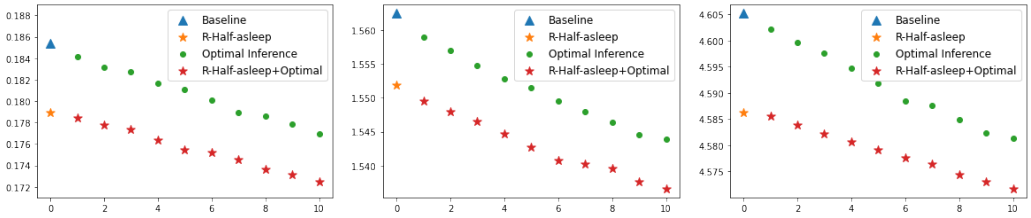
1. Baseline: This is the classic VAE-based compression introduced by [Townsend et al., 2019]. For binary and grey MNIST, both the encoder and decoder contain 2 fully connected layers with 500 hidden units and latent dimension 10. The observation distributions are Bernoulli and discretized Logistic distribution respectively. For CIFAR10, we use fully convolutional ResNets [He et al., 2016] with 3 residual blocks in the encoder/decoder, latent dimension 128 and discretized Logistic distribution with channel-wise linear autoregressive[Salimans et al., 2017] as the observation distribution. We train both the amortized posterior and the decoder by maximizing the ELBO (Equation 4.2) using Adam with $lr = 3\times10^{-4}$ for 100, 100 and 500 epochs (for Binary MNIST, Grey MNIST and CIFAR10 respectively), and then apply Algorithm 1 to conduct compression.

2. Reversed Half-asleep: we do amortized inference using Equation 4.27 for

100 and 300 epochs with Adam optimizer ($lr = 3\times10^{-4}$) for binary and grey MNIST respectively, and $lr = 1\times10^{-5}$ for 100 epochs for CIFAR10. Other training details are the same as the baseline method.

3. Optimal Inference: we take the amortized posterior (encoder) and decoder from the baseline and apply the K-step optimal inference strategy described in Algorithm 2 to do compression. We use Adam optimizer and vary the K from 1 to 10 to achieve a trade-off curve between compression rate and speed. We actively choose the highest learning rate that can make the BPD consistently improve with the increment of K: $lr = 5\times10^{-3}$ for binary and grey MNIST and $lr = 1\times10^{-3}$ for CIFAR10.

4. Reversed Half-asleep + Optimal Inference: We take the encoder in method 2 and decoder from the baseline and conduct K-step optimal inference. All other training details are as per method 3.



(a) Binary MNIST (b) Grey MNIST (c) CIFAR10

Figure 4.6: We plot the comparisons for different methods. The y-axis is the BPD and the x-axis represents the K gradient steps in the optimal inference. The baseline and our R-Half-sleep can be seen as special cases of optimal inference with $K = 0$. We find that given a fixed computational budget, our method achieves a lower BPD than one using traditional amortized inference training.

In Figure 4.6, we plot test BPD comparisons for the different methods outlined. We can see if optimization is not allowed at compression time, the use of our reverse-half-asleep method achieves a better compression rate with no additional computational cost. If we allow K-step optimization during compression, for a given computational budget, the amortized posterior initialized using our reverse-half-asleep method also achieves lower BPD, which leads to a better trade-off between the time and compression rate. Table 4.2 also reports the average time improvements of our method to compress a single MNIST and CIFAR10 image respectively, which

	Baseline	Ours	K=7
BPD	0.185	0.179	0.179
Com. Time	0.006	0.006	0.013
Dec. Time	0.006	0.006	0.013
Time Cost	-	0%	116.7%

(a) MNIST

	Baseline	Ours	K=8
BPD	4.602	4.585	4.585
Com. Time	0.27	0.27	0.38
Dec. Time	0.26	0.26	0.38
Time Cost	-	0%	46.2%

(b) CIFAR10

Table 4.2: Compression (Com.) and decompression (Dec.) time comparison. We show that to achieve the same BPD as our method, the K-step optimal inference strategy that initializes the amortized posterior needs $K = 7$ (binary MNIST) and $K = 8$ (CIFAR10) steps for each test datapoint, which will cost an additional 116.7% and 46.2% of time respectively during compression.

shows the effectiveness of our method.

4.6 Related Work

A different perspective on generative models' generalization is proposed in paper [Zhao et al., 2018] where the generalization is evaluated by testing if the model can generate novel combinations of features. However, the generalization defined in our work is purely measured by the test likelihood, which is a different perspective and more relevant for the application of lossless compression.

Recent work [Zhang et al., 2021] first studies the likelihood-based generalization for lossless compression. They focus on the test and train data that are from different distributions whereas we assume they follow the same distribution. Additionally, their model has a tractable likelihood and relates to the *generative model-related generalization*, whereas we focus on *inference-related generalization* in VAEs.

Previous work [Cremer et al., 2018] studied the *amortization gap* in amortized inference, which is caused by using $q_{\phi^*}(z|x_n)$ to generate posteriors for each input x_n rather than learning a posterior $q_n^*(z)$ for x_n individually. This gap can be alleviated using a larger capacity encoder network. This amortization gap is fundamentally different from the *inference generalization gap* we discuss in this work since the latter focuses solely on test time generalization but the former problem also exists at training time.

Recent work [Ruan et al., 2021] proposes a compression scheme based on the IWAE [Burda et al., 2015] bound, which is tighter than the ELBO and thus

improves the compression rate. However, this method has to compress/decompress multiple latent samples, which requires extra time and computational cost. On the other hand, we focus on improving the ELBO-based compression that only needs to compress one single latent sample. Nevertheless, similar to the K-step optimal inference strategy, our amortized training objective can also be used in the IWAE-based method, which gives a better proposal distribution for importance sampling, see Appendix C.2 for a demonstration.

Paper [Cemgil et al., 2020] considers the following data generation procedure $x_1 \sim p_d(x)$, $z_1 \sim p_\theta(z|x_1)$, $x_2 \sim p_\theta(x|z_1)$ and propose to enforce latent consistency between $q_\phi(z|x_1)$ and $q_\phi(z|x_2)$ for paired data (x_1, x_2) to encourage the robustness of the learned representation. This procedure is close to the self-supervised contrasting learning method [Chen et al., 2020] where the augmented data is the reconstruction of the training data using the VAE model. In our method, we want to encourage the sample from the model $x' \sim \int p_\theta(x|z)p(z)dz$ to have high ELBO under the model (Equation 4.25) to improve the generalization of the amortized inference and no paired data is required in our procedure. Therefore, both motivations and methodologies are different from our method.

4.7 Conclusions

In this chapter, we've explored the generalization gap of VAEs, delineating how both the amortized inference (encoder) and the generative model (decoder) influence it. Our findings illuminate the considerable impact of the amortized inference network on VAE generalization. To address this, we introduced a novel variational inference method, which, as evidenced by the lossless compression application, offers enhanced generalization.

Looking ahead, there's an exciting opportunity to delve deeper into the generalization attributes of the decoder model, with the aim of further optimizing VAE performance.

For this chapter, our discussion has revolved around scenarios where both training and testing data come from an identical distribution. As we transition to

the next chapter, our lens will shift towards the out-of-distribution (OOD) setting. Here, testing data deviates from the training data's distribution. We will investigate strategies to improve model generalization in this OOD context, ultimately enhancing applications such as OOD lossless compression and OOD detection.

Chapter 5

Out-of-distribution Generalization of Probabilistic Image Modelling

Out-of-distribution (OOD) detection and lossless compression constitute two problems that can be solved by the training of probabilistic models on a first dataset with subsequent likelihood evaluation on a second dataset, where data distributions differ. By defining the generalization of probabilistic models in terms of likelihood we show that, in the case of image models, the OOD generalization ability is dominated by local features. This motivates our proposal of a *Local Autoregressive* model that exclusively models local image features towards improving OOD performance. We apply the proposed model to OOD detection tasks and achieve state-of-the-art unsupervised OOD detection performance *without* the introduction of additional data. Additionally, we employ our model to build a new lossless image compressor: NeLLoC (Neural Local Lossless Compressor) and report state-of-the-art compression rates and model size.

5.1 Introduction

Probabilistic modelling has achieved great success in the modelling of images. Likelihood based models, *e.g.* Variational Auto-Encoders (VAE) [Kingma and Welling, 2014], Flow [Kingma and Dhariwal, 2018], Pixel CNN [Van Oord et al., 2016, Salimans et al., 2017] are shown to successfully generate high-quality images, in addition to estimating underlying densities.

The goal of probabilistic modelling is to use a model p_θ to approximate the unknown data distribution p_d using the training data $\{x_1,\dots,x_N\} \sim p_d$. A common method to learn parameters θ is to minimize some divergence between p_d and p_θ, for example, a popular choice is the KL divergence

$$\mathrm{KL}(p_d||p_\theta) = -\mathrm{H}(p_d) - \int \log p_\theta(x) p_d(x)\,\mathrm{d}x, \tag{5.1}$$

where the entropy of the data distribution is a constant. Since we only have access to finite p_d data samples $x_1,\dots,x_N$, the second term is typically approximated using a Monte Carlo estimation

$$\mathrm{KL}(p_d||p_\theta) \approx -\frac{1}{N}\sum_{n=1}^{N} \log p_\theta(x_n) - const.. \tag{5.2}$$

Minimizing the KL divergence is therefore equivalent to Maximum Likelihood Estimation. A typical evaluation criterion for the learned models is the test likelihood $\frac{1}{M}\sum_{m=1}^{M} \log p_\theta(x_m)$, with test set $\{x_1,\dots,x_M\} \sim p_d$. We refer to this evaluation as *in-distribution (ID) generalization*, since both training and test data are sampled from the same distribution p_d. However, in this work, we are interested in *out-of-distribution (OOD) generalization* such that the test data are drawn from p_o, where $p_o \neq p_d$. To motivate our study of this topic, we firstly introduce two applications: lossless compression and OOD detection, that both can make use of the OOD generalization property.

5.1.1 Model-based Lossless Compression

Lossless compression has a strong connection to probabilistic models [MacKay, 2003]. Let $\{x_1,\dots,x_M\}$ be test data to compress, where x_m is sampled from some underlying distribution with probability mass function (pmf) p_d. If p_d is known, we can design a compression scheme to compress each data x_m to a bit string with length approximately $-\log_2 p_d(x_m)$[1]. As $M \to \infty$, the averaged compression length will approach the entropy of the distribution p_d, that is; $\frac{1}{M}\sum_{m=1}^{M} -\log_2 p_d(x_m) \to \mathrm{H}(p_d)$

[1] For a compression method like Arithmetic Coding [Witten et al., 1987], the message length is always within two bits of $-\log_2 p_d(x_m)$ [MacKay, 2003], also see Section 5.4.1.

where $\mathrm{H}(\cdot)$ denotes entropy. In this case, the compression length is *optimal* under Shannon's source coding theorem [Shannon, 2001], *i.e.* we cannot find another compression scheme with compression length less than $\mathrm{H}(p_d)$. In practice, however, the true data distribution p_d is unknown and we must use a model $p_\theta \approx p_d$ to build the lossless compressor. Recent work successfully apply deep generative models such as VAEs [Townsend et al., 2019, 2020, Kingma et al., 2019], Flow [Hoogeboom et al., 2019, Berg et al., 2020] to conduct lossless compression. We note that the underlying models are designed to focus on test data that follows the same distribution as the training data, resulting in test compression rates that depend on the ID generalization ability of the model.

However, in practical compression applications, the distribution of the incoming test data is unknown, and is usually different from the training distribution p_d: $\mathscr{X}^o_{test} = \{x'_1, \ldots, x'_M\}$ where $x' \sim p_o \neq p_d$. To achieve good compression performance, a lossless compressor model should still be able to assign *high* likelihood for these OOD data. This practical consideration motivates **encouragement of the OOD generalization ability** of the model.

Empirical results [Townsend et al., 2020, Hoogeboom et al., 2019] have shown that we can still use model p_θ (trained to approximate p_d) in order to compress test data $\mathscr{X}^o_{test}$ with reasonable compression rates. However, the phenomenon that these models can generalize to OOD data lacks intuition and key components, affecting this generalization ability, remain underexplored. Consideration of recent advances in likelihood-based OOD detection next allows us to further investigate these questions and lead to our proposal of a new model that can encourage OOD generalization.

5.1.2 Likelihood-based OOD Detection

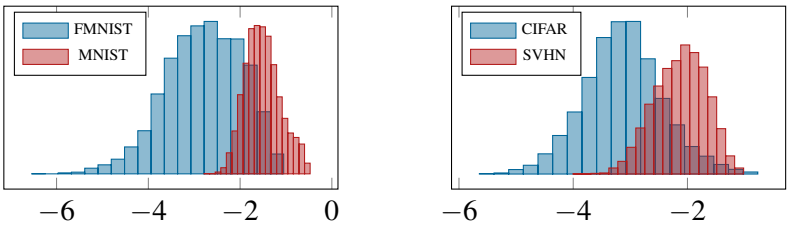
Given a set of unlabeled data, sampled from p_d, and a test data x' then the goal of OOD detection is to distinguish whether or not x' originates from p_d. A natural approach [Bishop, 1994] involves fitting a model p_θ to approximate p_d and treat sample x' as in-distribution if its (log) likelihood is larger than a threshold; $\log p_\theta(x') > \varepsilon$. Therefore, a good OOD detector model should assign *low* likelihood to the OOD data. In contrast to lossless compression, this motivates **discouragement of the**

OOD generalization ability of the model.

Surprisingly, recent work [Nalisnick et al., 2019a] report results showing that popular deep generative models, including *e.g*. VAE [Kingma and Welling, 2014], Flow [Kingma and Dhariwal, 2018] and PixelCNN [Salimans et al., 2017], can assign OOD data *higher* density values than in-distribution samples, where such OOD data may contain differing semantics, *c.f*. the samples used for maximum likelihood training. We demonstrate this phenomenon using PixelCNN models, trained on Fashion MNIST (CIFAR10) and tested using MNIST (SVHN). Figure 5.1 provides histograms of model evaluation using negative bits-per-dimension (BPD), that is; the $\log_2$ likelihood normalized by data sample dimension (larger negative BPD corresponds to larger likelihood). We corroborate previous work and observe that tested models assign a higher likelihood to the OOD data, in both cases. It's noteworthy to mention a peculiar observation: this phenomenon is not symmetrical. For instance, models trained on MNIST and SVHN typically yield a reduced average likelihood when evaluated on Fashion MNIST and CIFAR10, see [Nalisnick et al., 2019a] for an example. This counterintuitive phenomenon suggests that likelihood-based approaches may not make for a good OOD image detection criterion, yet encouragingly also illustrates that a probabilistic model, trained using one dataset, may be employed to compress data originating from a different distribution with a potentially higher compression rate. This intuition builds a connection between OOD detection and lossless compression. Inspired by this link, we next investigate the underlying latent causes of image model generalizability, towards improving both lossless compression and OOD detection.

5.2 OOD Generalizations of Image Models

Previous work studies the potential causes of the surprising OOD detection phenomenon: OOD data may have a higher model likelihood than ID data. For example, [Nalisnick et al., 2019b] used a typical set to reason about the source of the effect, while the work of [Ren et al., 2019] argues that likelihoods are significantly affected by image background statistics or by the size and smoothness of the back-



(a) Fashion vs MNIST (b) CIFAR10 vs SVHN

Figure 5.1: Left: log-likelihood of FashionMNIST, MNIST test data using a full PixelCNN model, trained on FashionMNIST training set. Right: log-likelihood of CIFAR10, SVHN test data using a PixelCNN model, trained on CIFAR10 training set. The x-axis indicates the value of the log-likelihood (negative BPD), the y-axis provides data sample counts.

ground [Krusinga et al., 2019]. In this work, we alternatively consider a recent hypothesis proposed by [Schirrmeister et al., 2020] (also implicitly discussed in [Kirichenko et al., 2020] for a flow-based model): *low-level local features, learned by (CNN-based) probabilistic models, are common to all images and dominate the likelihood.* From the perspective of OOD generalization, we can restate the hypotheses as: 1. Models **can** generalize to OOD images as local features are shared between image distributions, and 2. Models can generalize **well** to OOD images since local features dominate the likelihood.

In the work of [Schirrmeister et al., 2020], the authors investigated their original hypothesis by studying the differences between individual pixel values and neighbourhood mean values and additionally considered the correlation between models trained on small image patches and trained, alternatively, on full images. To further investigate this hypothesis, we rather propose to *directly model the in-distribution dataset, using only local feature information*. If the hypothesis is true, then the proposed *local* model alone should generalize well to OOD images. By contrasting such an approach with a standard *full* model, that considers both local and non-local features, we are also able to study the contribution that local features make to the full model likelihood. We first discuss how to build a local model for the image distribution and then use the proposed model to study generalization on OOD datasets.

5.2.1 Local Model Design

Autoregressive models have proven popular for modeling image datasets and common instantiations include PixelCNN [Van Oord et al., 2016, Salimans et al., 2017], PixelRNN [Van Oord et al., 2016] and Transformer based models [Child et al., 2019]. Assuming data dimension D, the Autoregressive model $p_f(x)$ can be written as

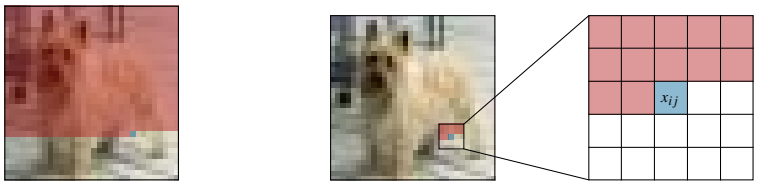
$$p_f(x) = p(x_{[1]}) \prod_{d=2}^{D} p(x_{[d]}|x_{[1:d-1]}), \tag{5.3}$$

informally we refer to this type of model as a "full model" since it can capture all dependencies between each dimension (pixel). Similarly, we can define a local autoregressive model $p_l(x)$ where pixel x_{ij}, at image row i column j, depends on previous pixels with fixed horizon h:

$$p_l(x) = \prod_{ij} p(x_{[ij]}|x_{[i-h:i-1,j-h:j+h]}, x_{[i,j-h:j-1]}), \tag{5.4}$$

with zero-padding used in cases where i or j are smaller than h. Figure 5.2b illustrates the resulting local autoregressive model dependency relationships. We implement this model using a masked CNN [Van Oord et al., 2016], with kernel size $k{=}2{\times}h{+}1$ in our first network layer, to mask out future pixel dependency. A full PixelCNN model would then proceed to stack multiple masked CNN layers, where increasing kernel depth affords receptive field increases. In contrast, we employ masked CNN with $1{\times}1$ convolutions in subsequent layers. Such $1{\times}1$ convolutions can model the correlation between channels, as in [Kingma and Dhariwal, 2018], and additionally prevent our local model from obtaining information from pixels outwith the local feature region, defined by h. Pixel dependencies are therefore defined solely using the kernel size of the first masked CNN layer, allowing for easy control over model's local feature size. We note that the proposed local autoregressive model can also be implemented using alternative backbones *e.g.* Pixel RNN [Van Oord et al., 2016] or Transformers [Child et al., 2019]. We plot local model samples in Figure 5.4. Unlike full autoregressive models [Van Oord et al., 2016, Salimans et al., 2017], which can be used to generate semantically coherent samples, we find the samples from the

local model are locally consistent yet have no clear semantic meaning.



(a) Full Autoregressive Model (b) Local Autoregressive Model

Figure 5.2: (a) full autoregressive model pixel dependencies; the distribution of the current pixel (blue) depends on all *previous* pixels (red); (b) local autoregressive model dependencies, with $h = 2$. The distribution of x_{ij} (blue) depends on only the pixels in a local region (red).

5.2.2 Local Model Generalization

To investigate the generalization ability of our local autoregressive model, we fit the model to Fashion MNIST (grayscale) and CIFAR10 (color) training datasets and test using in-distribution (ID) images (respective dataset test images) and additional out-of-distribution (OOD) datasets: MNIST, KMNIST (grayscale) and SVHN, CelebA[2] (color). Both models use the discretized mixture of logistic distributions [Salimans et al., 2017] with 10 mixtures for the predictive distribution and a ResNet architecture [Van Oord et al., 2016, He et al., 2016]. We use a local horizon length h=3 (kernel size k=7) for both grayscale and color image data. We compare our local autoregressive model to a standard full autoregressive model (*i.e.* a standard PixelCNN), with additional network architecture and training details found in Appendix D.1. Tables 5.1, 5.2 report comparisons in terms of BPD (where lower values

Table 5.1: Test BPD (Trained on Fashion MNIST)

Test Dataset	Full	Local
Fashion MNIST (ID)	2.78	2.89
MNIST (OOD)	1.50	1.49
KMNIST (OOD)	2.48	2.44

Table 5.2: Test BPD (Trained on CIFAR10)

Test Dataset	Full	Local
CIFAR10 (ID)	3.12	3.25
SVHN (OOD)	2.13	2.13
CelebA (OOD)	3.33	3.35

entail higher likelihood) for Fasion MNIST and CIFAR10, respectively. We observe that for in-distribution (ID) data, the full model has better generalization ability *c.f.*

[2]We down-sample the original CelebA to 32×32×3, see Appendix A.7 for details.

Table 5.3: Generalization of local model with different horizon sizes. The model is trained on FashionMNIST dataset.

Method	h=1	h=2	h=3	h=4	h=5
(ID) Fashion	3.17	2.93	2.89	2.88	2.88
(OOD) MNIST	1.54	**1.48**	1.49	1.50	1.51
(OOD) KMNIST	2.54	**2.43**	2.44	2.46	2.47

the local model (0.11 and 0.13 BPD, respectively). This is unsurprising as training and test data originate from the same distribution; both local and non-local features, as learned by the full model, help ID generalization. For OOD data, we observe that the local model has generalization ability similar to the full model, exhibiting very small empirical gaps (only ≈ 0.02 BPD on average), showing that the local model alone can generalize well to OOD distributions. We thus verify the hypothesis considered at the start of Section 5.2.

For simple datasets containing gray-scale images, the PixelCNN model is flexible enough to capture both local and global features. We notice that, in Table 5.1, our local model exhibits even better OOD generalization than the full model. This drives us to further study the role of non-local features for generalization. When the local horizon size increases, the model will be able to learn features with greater non-locality. We thus vary the local horizon size to study generalization ability under this property, see Table 5.3. We find the model has poor generalization performance when local features are too small and increasing the horizon size helps ID generalization but decreases the OOD generalization. A consistent phenomenon is observed when considering color images, see Appendix D.2.1. We can thus conclude: **non-local features are not shared between images distributions, overfitting to non-local features will hurt generalization**. These hypotheses indicate two opposing strategies for the considered tasks:

OOD detection: distributions are distinguished by dataset-unique features, thus building **non-local** models, able to discount common local features, improves the detector distinguishability power.

Lossless compression: OOD generalization is possible due to the sharing of

Table 5.4: The generalization ability of local models with increasing horizon size. All models have residual block number r=1 and are trained on CIFAR10. The reported BPDs have standard deviation 0.02 across multiple random seeds.

Method	h=2	h=3	h=4	h=5
(ID) CIFAR	3.38	3.28	3.26	3.25
(OOD) SVHN	2.21	2.16	2.15	2.15
(OOD) CelebA	4.08	4.07	4.07	4.07

local features between distributions. Employing only a local model can encourage OOD generalization, by preventing the model from over-fitting to dataset-unique features, specific to the training distribution. Sections 5.3 and 5.4 will further demonstrate how contrasting modeling strategies can benefit these tasks.

5.3 OOD Detection with Non-Local Model

As was discussed in Section 5.1.2, local image features are observed to be largely common across the real-world image distribution and can be treated as a domain-prior. Therefore, in order to detect whether or not an image is out-of-distribution, we can stipulate a non-local model able to discount local features of the image distribution; denoted here $p_{nl}(x)$. It is however not easy to build such a non-local model directly, since the concept of "non-local" lacks a mathematically rigorous definition. However, we propose that a non-local model can be considered to be the complement of a local model, from a respective full model. In the following section, we therefore propose to use a product of experts to indirectly define $p_{nl}(x)$, and demonstrate how this may be used for OOD detection.

5.3.1 Product of Experts and Non-Local Model

As demonstrated in Section 5.2.2, the full model $p_f(x)$ and the local model $p_l(x)$ can be easily built for the image distribution, *e.g.* a full autoregressive model and a local autoregressive model. We further assume the full model allows the following decomposition:

$$p_f(x) = \frac{p_l(x)p_{nl}(x)}{Z}, \tag{5.5}$$

where $Z = \int p_l(x)p_{nl}(x)\,\mathrm{d}x$ is the normalizing constant. This formulation can also be thought of as a product of experts (PoE) model [Hinton, 2002] with two experts; p_l (local expert) and p_{nl} (non-local expert). An interesting property of the PoE model is that if a data sample x' has high full model probability $p_f(x')$, it should possess high probability mass in *each* of the expert components[3]. Therefore, the PoE model assumption is consistent with our image modelling intuition: a valid image requires both valid local features (*e.g.* stroke, texture, local consistency) and valid non-local features (semantics).

By our model assumption, the density function of the non-local model can be formally defined and is proportional to the likelihood ratio:

$$p_{nl}(x) \propto \frac{p_f(x)}{p_l(x)} \equiv \hat{p}_{nl}(x), \tag{5.6}$$

where $\hat{p}_{nl}(x)$ denotes the unnormalized density. For the OOD detection task, we require only $\hat{p}_{nl}(x)$ in order to provide a score classifying whether or not test data x is OOD and therefore *do not require to estimate the normalization constant* Z. We also note that as we increase the local horizon length for p_l, the local model will converge to a full model $p_l \to p_f$, and $\hat{p}_{nl}(x) = 1$ becomes a constant and inadequate for OOD detection. This further suggests the importance of using a local model. Figure 5.3 shows histograms of $\hat{p}_{nl}(\cdot)$ for both ID and OOD test datasets. We observe that the majority of ID test data obtains higher likelihood than OOD data, illustrating the effectiveness of non-local models.

5.3.2 Connections to Related Methods

We highlight that the score $\hat{p}_{nl}(x)$ that we use to conduct OOD detection allows a principled likelihood interpretation: **the unnormalized likelihood of a non-local model.** We believe this to be the first time that the likelihood of a non-local model

[3]This PoE property differs from a Mixture of Experts (MoE) model that linearly combines experts. If data x' has high probability in the local model (*e.g.* $p_l(x') = 0.9$) but low probability in the non-local model (*e.g.* $p_{nl}(x') = 0.1$), the probability in the full PoE model $p_f(x') \propto 0.9 * 0.1$ is also *small*. On the contrary, if we assume p_f is a MoE: $p_f = \frac{1}{2}p_l + \frac{1}{2}p_{nl}$, then $p_l(x') = 0.9$ and $p_{nl}(x') = 0.1$ results in a *high* full MoE model $p_f(x')$ value *i.e.* $p_f(x') = 0.5 * 0.9 + 0.5 * 0.1 = 0.5$. We refer to [Hinton, 2002] for additional PoE model details.

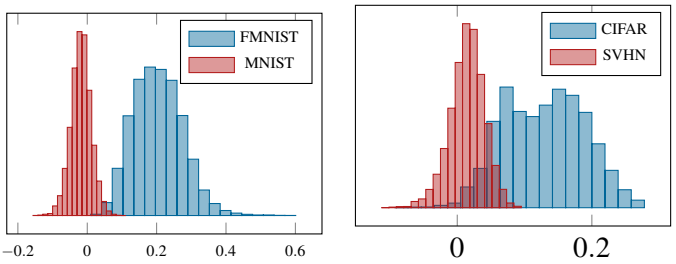


Figure 5.3: Unnormalized log likelihood of the non-local model $\hat{p}_{nl}(x)=\frac{p_f(x)}{p_l(x)}$ for both ID test datasets (FashionMNIST, CIFAR10) and OOD datasets (MNIST, SVHN). ID test datasets obtain significantly higher likelihoods, on average, in each case.

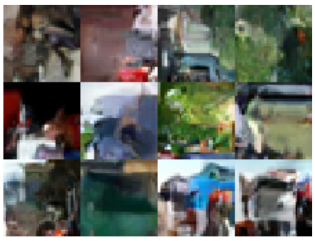


Figure 5.4: Samples from the local autoregressive model, see Appendix D.2.2 for details.

is considered in the literature. However, other likelihood ratio variants have been previously explored for OOD detection. We briefly discuss related work and highlight where our method differs from relevant literature.

In [Ren et al., 2019], it is assumed that each data sample x can be factorized as $x = \{x_b, x_s\}$, where x_b is a background component, characterized by population level background statistics: p_b. Further, x_s then constitutes a semantic component, characterized by patterns belonging specifically to the in-distribution data: p_s. A full model, fitted on the original data, (*e.g.* a flow-based model) can then be factorized as $p_f(x) = p_f(x_b, x_s) = p_b(x_b)p_s(x_s)$ and the semantic model can correspondingly be defined as a ratio: $p_s(x) = \frac{p_f(x)}{p_b(x)}$, where $p_f(x)$ is a full model. In order to estimate $p_b(x_b)$, the authors of [Ren et al., 2019] design a perturbation scheme and construct samples from $p_b(x_b)$ by adding random perturbation to the input data. A further full generative model is then fitted to the samples towards estimating p_b. In our method, both $p_l(x)$ and $p_{nl}(x)$ constitute distributions of the same sample space (that of the original image x) whereas p_s and p_b in [Ren et al., 2019] form distributions in different sample spaces (that of x_s and x_b, respectively). Additionally, in comparison with our local and non-local experts factorization of the model distribution, their decomposition of an image into 'background' and 'semantic' parts may not be suitable for all image content and the construction of samples from $p_b(x_b)$ (adding random perturbation) lacks principled explanation. In [Serrà et al., 2019, Schirrmeister et al., 2020], the score for OOD detection is defined as $s(x) = \frac{p_f(x)}{p_c(x)}$ where p_f

is a full model and p_c is a complexity measure containing an image domain prior. In practice, p_c is estimated using a traditional compressor (*e.g.* PNG or FLIF), or a model trained on an additional, larger dataset in an attempt to capture general domain information [Schirrmeister et al., 2020]. In comparison, our method does not require the introduction of new datasets and our explicit local feature model, p_l, can be considered more transparent than PNG of FLIF. Additionally, our likelihood ratio can be explained as the *(unnormalized) likelihood of the non-local model* for the in-distribution dataset, whereas the score described by [Schirrmeister et al., 2020, Serrà et al., 2019] does not offer a likelihood interpretation. In Section 5.4, we discuss how the proposed local model may be utilized to build a lossless compressor, further highlighting the connection between our OOD detection framework and traditional lossless compressors (*e.g.* PNG or FLIF). In Table 5.6, we report experimental results showing that a lossless compressor based on our model significantly improves compression rates *c.f.* PNG and FLIF, further suggesting the benefits of the introduced OOD detection method.

5.3.3 Experiments

We conduct OOD detection experiments using four different dataset-pairs that are considered challenging [Nalisnick et al., 2019a]: Fashion MNIST (ID) *vs.* MNSIT (OOD); Fashion MNIST (ID) *vs.* OMNIGLOT (OOD); CIFAR10 (ID) *vs.* SVHN (OOD); CIFAR10 (ID) *vs.* CelebA (OOD). We actively select not to include dataset pairs such as CIFAR10 *vs.* CIFAR100 or CIFAR10 *vs.* ImageNet since these contain duplicate classes and cannot be treated as strictly disjoint (or OOD) datasets [Serrà et al., 2019]. Additional experimental details are provided in Appendix D.1. In Table 5.5 we report the 'area under the receiver operating characteristic curve' (AUROC), a common measure for the OOD detection task [Hendrycks and Gimpel, 2016]. We compare against methods, some of which require additional label information[4], or datasets. Our method achieves state-of-the-art performance in most cases

[4]In principle methods that require labels correspond to *classification*, task-dependent OOD detection, which may be considered fundamentally different from task-independent OOD detection (with access to only image space information), see [Ahmed and Courville, 2020] for details. We compare against both classes of method, for completeness.

Table 5.5: OOD detection AUROC comparisons. Higher values indicate better performance, results are rounded to three decimal places. Results are reported in each case directly using the original references except in the cases of ODIN [Ren et al., 2019, Lee et al., 2018] and VIB [Choi et al., 2018]. Results for the typicality test are from [Serrà et al., 2019], corresponding to batches of two samples of the same type. (a) The Mahalanobis method requires knowledge of the validation data (OOD distribution). (b) A full PixelCNN (see Appendix D.1) is trained on the ID dataset and its likelihood evaluations are then used to calculate AUROC.

ID dataset:	FashionMNIST		CIFAR10	
OOD dataset:	MNIST	Omniglot	SVHN	CelebA
Using Labels				
ODIN [Liang et al., 2017]	0.697	-	0.966	-
VIB [Alemi et al., 2018]	0.941	0.943	0.528	0.735
Mahalanobis[a] [Hendrycks et al., 2018]	0.986	-	0.991	-
Gram-Matrix [Sastry and Oore, 2019]	-	-	0.995	-
Using Additional Datasets				
Outlier Exposure [Hendrycks et al., 2018]	-	-	0.758	0.615
Glow/Tiny-Glow [Schirrmeister et al., 2020]	-	-	0.939	-
PCNN/Tiny-PCNN [Schirrmeister et al., 2020]	-	-	0.944	-
Not Using Additional Information				
WAIC (ensemble) [Choi et al., 2018]	0.766	0.796	**1.000**	-
Glow/PNG [Schirrmeister et al., 2020]	-	-	0.754	-
PCNN/PNG [Schirrmeister et al., 2020]	-	-	0.823	-
Likelihood Ratio in [Ren et al., 2019]	0.997	-	0.912	-
MSMA KD Tree [Mahmood et al., 2020]	0.693	-	0.991	-
S using Glow/FLIF [Serrà et al., 2019]	0.998	**1.000**	0.950	0.736
S using PCNN/FLIF [Serrà et al., 2019]	0.967	**1.000**	0.929	0.535
Full PixelCNN likelihood[b]	0.074	0.361	0.113	0.602
Our method	**1.000**	**1.000**	0.969	**0.949**

without requiring additional information. We observed that the model-ensemble methods, WAIC [Choi et al., 2018] and MSMA [Mahmood et al., 2020] can achieve higher AUROC in the experiments involving color images yet are significantly outperformed by our approach in the case of grayscale data. We thus evidence that our simple method is consistently more reliable than alternative approaches and that our score function allows a principled likelihood interpretation.

5.4 Lossless Compression with Local Model

Recent deep generative model based compressors [Townsend et al., 2019, Berg et al., 2020, Kingma et al., 2019, Ho et al., 2019] are designed under the assumption that data to be compressed originates from the same distribution (source) as model training data. However, in practical scenarios, test images may come from a diverse set of categories or domains and training images may be comparatively limited [MacKay, 2003]. Obtaining a single method capable of offering strong compression performance on data from different sources remains an open problem and related study involves consideration of "universal" compression methods [MacKay, 2003]. Based on our previous intuitions relating to generalization ability; to build such a "universal" compressor in the image domain, we believe a promising route involves leveraging models that only depend on common local features, shared between differing image distributions. We thus propose a new "universal" lossless image compressor: NeLLoC (Neural Local Lossless Compressor), built upon the proposed local autoregressive model and the concept of Arithmetic Coding [Witten et al., 1987]. In comparison with alternative recent deep generative model-based compressors, we find that NeLLoC has competitive compression rates on a diverse set of data, yet requires significantly smaller model sizes which in turn reduces storage space and computation costs. We further note that due to our design choices, and in contrast to alternatives, NeLLoC can compress images of *arbitrary* size.

In the remaining parts of this section, we first discuss NeLLoC model structure and then provide important resulting properties of the method.

5.4.1 NeLLoC Model

Our NeLLoC model uses the same network backbone as that of our OOD detection experiment (Section 5.3): a Masked CNN with kernel size $k = 2 \times h + 1$ (h is the horizon size) in the first layer and followed by several residual blocks with 1×1 convolution, see Appendix D.1 for the network architecture and training details. To realize the predictive distribution for each pixel, we propose to use a discretized Logistic-Uniform mixture distribution, which we now introduce.

Discretized Logistic-Uniform Mixture Distribution The discretized logistic

mixture distribution, proposed by [Salimans et al., 2017], has shown promising results for the task of modeling color images. In order to ensure numerical stability, the original implementation can provide only an (accurate) approximation and therefore cannot be used for our task of lossless compression, which requires exact numerical evaluation. We, therefore, propose to use the discretized Logistic-Uniform Mixture distribution, which mixes the original discretized logistic mixture distribution with a discrete uniform distribution:

$$x \sim (1-\alpha)\left(\sum_{i=1}^{K} \pi_i \text{Logistic}(\mu_i, s_i)\right) + \alpha \text{U}(0,\ldots,255), \tag{5.7}$$

where $\text{U}(0,\ldots,255)$ is the discrete uniform distribution over the support $\{0,\ldots,255\}$. The proposed mixture distribution can explicitly avoid numerical issues and its pmf and cdf can be easily evaluated without requiring approximation. We can then use this cdf evaluation in relation to Arithmetic Coding. In practice; we set $\alpha = 10^{-4}$ to balance numerical stability and model flexibility. We use $K = 10$ (mixture components) for all models in the compression task.

5.4.2 Properties of NeLLoC

Universal Image Compressor As discussed previously, the motivation for designing NeLLoC is to realize an image compressor that is applicable (generalizable) to images originating from differing distributions. Towards this, NeLLoC conducts compression depending on local features which are shown to constitute a domain prior for all images. We next discuss other important aspects towards making NeLLoC practical when considering real applications.

Arbitrary Image Size Common generative models, *e.g.* VAE or Flow, can only model image distributions with fixed dimension. Therefore, lossless compressors based on such models [Townsend et al., 2019, Hoogeboom et al., 2019, Berg et al., 2020, Ho et al., 2019, Kingma et al., 2019] can only compress images of fixed size. Recently, HiLLoC [Townsend et al., 2020] explore fully convolutional networks, capable of accommodating variable size input images, yet still requires even height

Table 5.6: Lossless Compression Comparisons. We compare against traditional image compression and neural network-based models. For neural models, we report results where models are trained on CIFAR10 or ImageNet32 and tested on other ID or OOD test datasets. We use † to represent the best ID generalization and ⋆ to represent the best OOD generalization. (a) We down-sample CelebA to 32×32, see Appendix D.1.2. (b) The BPD 3.15 reported in [Townsend et al., 2020] is tested on 2000 random samples from the full ImageNet test set, whereas we test HiLLoC on the whole test set with 49032 images. The reported BPD of NeLLoC has a standard deviation ∼0.02 across multiple random seeds.

Method	Size(MB)	CIFAR	SVHN	CelebA[a]	IN32	IN64	IN
Generic							
PNG [Boutell and Lane, 1997]	N/A	5.87	5.68	6.62	6.58	5.71	5.12
WebP [Lian and Shilei, 2012]	N/A	4.61	3.04	4.68	4.68	4.64	3.66
JPEG2000 [Rabbani, 2002]	N/A	5.56	4.10	5.70	5.60	5.10	3.74
FLIF [Sneyers and Wuille, 2016]	N/A	4.19	2.93	4.44	4.52	4.19	3.51
Train/test on one distribution		ID			ID	ID	
LBB [Ho et al., 2019]	-	**3.12**†	-	-	3.88	3.70	-
IDF++[Berg et al., 2020]	-	3.26	-	-	4.12	4.81	-
Trained on CIFAR		ID	OOD	OOD	OOD	OOD	OOD
IDF [Hoogeboom et al., 2019]	223.0	3.34	-	-	4.18	3.90	-
Bit-Swap [Kingma et al., 2019]	44.7	3.78	2.55	3.82	5.37	-	-
HiLLoC [Townsend et al., 2020]	156.4	3.32	2.29	3.54	4.89	4.46	3.42
L3C [Mentzer et al., 2019]	19.11	3.39	3.17	4.44	4.97	4.77	4.88
NeLLoC ($r=0$)	**0.49**	3.38	2.23	3.44	4.20	3.86	3.30
NeLLoC ($r=1$)	1.34	3.28	2.16	3.37	4.07	3.74	3.25
NeLLoC ($r=3$)	2.75	3.25	**2.13***	**3.35***	**4.02***	**3.69***	**3.24***
Trained on ImgNet32		OOD	OOD	OOD	ID	OOD	OOD
IDF [Hoogeboom et al., 2019]	223.0	3.60	-	-	4.18	3.94	-
Bit-Swap [Kingma et al., 2019]	44.9	3.97	3.00	3.87	4.23	-	-
HiLLoC [Townsend et al., 2020]	156.4	3.56	2.35	3.52	4.20	3.89	**3.25**$^{\star b}$
L3C [Mentzer et al., 2019]	19.11	4.34	3.21	4.27	4.55	4.30	4.34
NeLLoC ($r=0$)	**0.49**	3.64	2.38	3.54	3.93	3.63	3.37
NeLLoC ($r=1$)	1.34	3.56	2.26	3.47	3.85	3.55	3.31
NeLLoC ($r=3$)	2.75	**3.51***	**2.21**	**3.43***	**3.82**†	**3.53***	3.29

and width[e]. L3C [Mentzer et al., 2019], based on a multi-scale autoencoder, can compress large images yet also requires height and width to be even. NeLLoC is able to compress images with arbitrary size based on an alternative and simple intuition:

[e]For images with odd height or width, padding is required.

we only model the conditional distribution, based on local neighbouring pixels. We thus do not model the distribution of the entire image and can therefore, in contrast to HiLLoC and L3C, compress arbitrary image sizes without padding requirements.

To validate method properties, we compare the compression performance of NeLLoC with both traditional image compressors and recently proposed generative model based compressors. We train NeLLoC with horizon length $h = 3$ on two (training) datasets: CIFAR10 (32×32) and ImageNet32 (32×32) and test on the previously introduced test sets, including both ID and OOD data. We also test on ImageNet64 with size 64×64 and a full ImageNet[f] test set (with average size 500×374). Table 5.6 provides details of the comparison. We find NeLLoC achieves better BPD in a majority of cases. Exceptions include LBB [Ho et al., 2019] having better ID generalization for CIFAR and HiLLoC [Townsend et al., 2020] having better OOD generalization in the full ImageNet, when the model is trained on ImageNet32.

Small Model Size In comparison with traditional codecs such as PNG or FLIF, one major limitation of current deep generative model based compressors is that they require generation and storage of models of very large size. For example, HiLLoC [Townsend et al., 2020] contains 24 stochastic hidden layers, resulting in a capacity and parameter size of 156 MegaBytes (MB) using 32-bit floating point model weights. This poses practical challenges relating to both storage and transmission of such models to real, often resource-limited, edge devices. Since NeLLoC only models the local region, a small network of three Residual blocks with 1×1 convolutions (except the first layer) is enough to achieve state-of-the-art compression performance with parameter size 2.75 MB. We also investigate the compression task under NeLLoC with 1 and 0 residual blocks, which have parameter sizes of 1.34 and 0.49 MB respectively, and yet still observe respectable performance. We report model size comparisons in Table 5.6. In principle, NeLLoC can also be combined with other resource-saving techniques such as weight quantization [Bird et al., 2020], which we consider a promising line of future investigation.

[f]Images with height or width greater than 1000 are removed, resulting in a total of 49032 test images.

5.5 Parallel Decoding of NeLLoC

In contrast to full autoregressive models, where pixels must be decoded sequentially, there exist pixels that can be independently decoded in a local autoregressive model. Figure 5.5 gives the topological order of parallel decoding, for a 5×5 image, using a local autoregressive model with $h = 1$. The number in each pixel indicates the time at which the pixel can be decoded. For example, the two red pixels marked with time 6 can be decoded in parallel since they are independent under the model.

In general, for an image with size $D \times D$, on a machine with $\lfloor \frac{D+h}{h+1} \rfloor$ parallel processing units, the total decoding time $T = D + (D-1) \times (h+1)$. Since h is a small constant $h \ll D$, the decoding time scales with $\mathcal{O}(D)$, which is a significant improvement over the $\mathcal{O}(D^2)$ of full autoregressive models. In the example in Figure 5.5, for a 5×5 image with dependency length $h = 1$, the decoding time is $T = 13$ whereas in a full autoregressive model, $T = 25$. In the next section, we discuss how to implement the parallelization scheme in practice.

1	2	3	4	5
3	4	5	6	7
5	6	7	8	9
7	8	9	10	11
9	10	11	12	13

Figure 5.5: Topological decoding order. Pixels with the same number are parallel decoded.

We observe that for fixed h, the positions of pixels decoded at each time step do not change. We can thus pre-compute the topological ordering and save the locations the pixels computed at each time step. At each time step in the decoding stage, we just load the saved positions of the independent pixels to be decoded and collect the image patches on which they depend into a batch. For example, for the pixel x_{33} in Figure 5.6a, the relevant patch is a 3×3 square marked in red, the redundant pixels $\{x_{33}, x_{34}, x_{42}, x_{43}, x_{44}\}$ will be masked out in the local autoregressive model. One can also take a rectangle patch that contains $\{x_{22}, x_{23}, x_{24}, x_{32}, x_{33}, x_{34}\}$ to reduce the computation and $\{x_{33}, x_{34}\}$ will be masked out in this case. We pad the patches with 0 for the pixels near the boundary, to make sure all the patches have the same size. The parallel decoding procedure is summarized in Algorithm 3.

Algorithm 3 Parallel Decoding Procedure of Local Autoregressive Models

1: **for** t = 1 to T **do**
2: Load the positions of independent pixels to be decoded at time t.
3: Gather the relevant patches based on the loaded positions to form a batch.
4: In parallel, compute the predictive distributions for those pixels using the batch.
5: Decode the pixel values using the predictive distributions.

5.5.1 Sheared Local Autoregressive Model

We notice that in Algorithm 3, the conditionally independent pixels in each step are located in nonadjacent positions. For example, the dependent areas (green) of the two red pixels in Figure 5.5 are not aligned in memory. This requires extra indexing time when reading/writing their values. To alleviate the speed limitation, we propose to transform the model such that the conditionally independent pixels are aligned. Specifically, for a local autoregressive model with dependency horizon h, we shear both the model and image, with offset $o = h + 1$. Figure 5.6 shows an example where the local autoregressive model with $h = 1$ (Figure 5.6a) is sheared with offset $o = 2$. We observe that in the sheared model, the conditionally independent pixels are aligned in memory, allowing significantly faster parallel reading/writing of those pixels. The sheared model has length $L = D + (D-1) \times o$ for a $D \times D$ images, which is equal to the decoding steps T in pNeLLoC since $o = h + 1$. Therefore, the inference time scales with $O(D)$ on parallel processing units.

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}
x_{31}	x_{32}	x_{33}	x_{34}	x_{35}
x_{41}	x_{42}	x_{43}	x_{44}	x_{45}
x_{51}	x_{52}	x_{53}	x_{54}	x_{55}

(a) Local model with $h = 1$

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	0	0	0	0	0	0	0	0
0	0	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	0	0	0	0	0	0
0	0	0	0	x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	0	0	0	0
0	0	0	0	0	0	x_{41}	x_{42}	x_{43}	x_{44}	x_{45}	0	0
0	0	0	0	0	0	0	0	x_{51}	x_{52}	x_{53}	x_{54}	x_{55}

(b) Sheared model with offset $o = 2$

Figure 5.6: Pixel dependency after the shear operation. The red pixels in the same column in the sheared image (b) are conditionally independent given the green pixels and are aligned in memory.

As discussed in Section 2, the pixel dependency structure of the local autore-

Algorithm 4 Parallel Decoding Procedure of Sheared Local Autoregressive Models

1: Shear the image based on the dependency horizon.
2: **for** t = 1 to T **do**
3: In parallel, compute the predictive distributions for those pixels in each column.
4: Decode the pixel values using the predictive distributions.
5: Undo the shear operation for the decoded image.

gressive model only depends on the first convolution kernel. Therefore, to shear the model, we only need to shear the first convolution kernel. Figure 5.7 visualizes the sheared convolutional kernel for two local autoregressive models with $h = 1$ and $h = 2$. After shearing the model, we also need to shear the images to conduct compression and decompression, see Algorithm 4 for a summary of the decoding procedure. We refer to compression with the sheared model as **Shear**ed **Lo**cal Lossless **C**ompression (ShearLoC).

w_{11}	w_{12}	w_{13}
w_{21}	0	0

(a) NeLLoC ($h = 1$)

w_{11}	w_{12}	w_{13}
0	0	w_{21}

(b) ShearLoC ($h = 1$)

w_{11}	w_{12}	w_{13}	w_{14}	w_{15}
w_{21}	w_{22}	w_{23}	w_{24}	w_{25}
w_{31}	w_{32}	0	0	0

(c) NeLLoC ($h = 2$)

w_{11}	w_{12}	w_{13}	w_{14}	w_{15}	0	0	0
0	0	0	w_{21}	w_{22}	w_{23}	w_{24}	w_{25}
0	0	0	0	0	0	w_{31}	w_{32}

(d) ShearLoC ($h = 2$)

Figure 5.7: Convolution kernel weights in the first layer of the local models and the corresponding sheared local models. We show two examples with dependency horizons: $h = 1$ (a,b) and $h = 2$ (c,d).

5.5.2 Demonstrations

We use the three pre-trained (on CIFAR10) models (with 0, 1 and 3 ResNet blocks) for all the experiments. During encoding, since all pixels are observed, the statistics of all the pixels can be computed in parallel. However, we have found that on CPU the computations may not be deterministic when using different batch sizes during the encoding and decoding. Therefore, we instead use the an identical inference procedure in both the encoding and decoding stages. Other details can be found in the provided repository. All the experiments are conducted on a MacBook Air (2020) with M1 chip and 8GB memory, the results are averaged over 10 images from the

ImageNet [Deng et al., 2009] dataset.

We compare parallel NeLLoC (pNeLLoC) and ShearLoC with the original sequential NeLLoC (sNeLLoC) implementation. Since all algorithms use the same underlying pre-trained model, they have the same compression BPD. Table 5.7a shows the decompression time comparison using three models on images with side length 32. We find pNeLLoC is 2x faster than sNeLLoC with the 0 ResNet block model, and the improvement increases for larger models. Compared to pNeLLoC, ShearLoC achieves a further speed improvement, with a more significant advantage in larger models.

We also compare the decompression time on square images with increased side lengths: $[32, 64, 128, 1024]$[g]. Table 5.7b shows the improvement percentage from using pNeLLoC grows when we increase the size of the test images. This is consistent with the theoretical argument that the proposed parallelization scheme improves the computation complexity from $O(D^2) \to O(D)$ on parallel units. Similarly, additional improvements can be achieved when using ShearLoC, which shares the same complexity with pNeLLoC but has more efficient memory access.

Table 5.7: Decoding time (s) comparisons. We show the improvement of pNeLLoC (in green) and ShearLoC (in red) comparing to using sNeLLoC.

(a) Different model sizes, the image has size 32×32.

Res. Num.	0	1	3
BPD	3.39	3.32	3.29
sNeLLoC	0.460 (-)	0.578 (-)	0.774 (-)
pNeLLoC	0.223 (2.06x)	0.277 (2.09x)	0.335 (2.31x)
ShearLoC	0.218 (2.11x)	0.222 (2.60x)	0.245 (3.16x)

(b) Different image sizes, the model has 0 ResNet blocks.

Side len.	32	64	128	1024
BPD	3.39	3.05	2.93	2.22
sNeLLoC	0.460 (-)	1.879 (-)	7.574 (-)	475.9 (-)
pNeLLoC	0.223 (2.06x)	0.757 (2.48x)	2.217 (3.42x)	100.0 (4.58x)
ShearLoC	0.218 (2.11x)	0.612 (3.07x)	1.683 (4.60x)	73.00 (6.52x)

[g]The 1024×1024 images are provided in the repository, the corresponding result is averaged over 3 images.

5.6 Conclusions

In this work, we propose the local autoregressive model to study OOD generalization in probabilistic image modelling and establish an intriguing connection between two diverse applications: OOD detection and lossless compression. We verify and then leveraged a hypothesis regarding local features and generalization ability in order to design approaches towards solving these tasks.

One major challege of the proposed techniques is to with our proposed methods lies in selecting the appropriate size for the local window, a crucial hyper-parameter. In practice, local window size can be decided on a validation dataset and then subsequently applied to in the test dataset. However, comprehending the relation between the local window size and the OOD generation behavior for images necessitates an understanding of the manifold image distribution. Given the complexity of this task, we earmark it for future research directions.

For parallel decoding using NeLLoC, several methods have been proposed to improve the sampling runtime in autoregressive models. For example, [Reed et al., 2017, Razavi et al., 2019] explore the multi-scale structure in the image domain, and design models that allow parallel generation of pixels in higher resolution samples conditioned on low-resolution samples. In contrast to previous works, the parallel method proposed in this paper is specially designed for local autoregressive models, with the flexibility to handle images of arbitrary size. Local autoregressive models can also be combined with latent variable models to generate semantically-coherent images [Gulrajani et al., 2017b, Zhang et al., 2022]. In this case, the proposed parallelization schemes can be also used to improve the sampling efficiency, which we leave to future work.

End of Part II

As we conclude the second part of the thesis, we've traversed the landscape of generalization properties of the generative models. In Chapter 5, we discussed the generalization attributes of VAE, introducing a method aimed at amplifying the in-distribution generalization performance for lossless compression tasks through improved amortized inference. Chapter 6 pivoted to the out-of-distribution (OOD) generalization for image modeling. Within this context, we crafted both a local and a non-local model, each designed to bolster performance for OOD lossless compression and OOD detection respectively.

In the upcoming concluding chapter, we'll weave the threads of our discussions together, highlighting the interrelation of the various topics and how they coalesce into a unified narrative. We'll reflect on the implications of our findings, discuss the limitations inherent to our current approach, and chart possible trajectories for future research in this domain.

Chapter 6

Conclusions

In this chapter, we offer a comprehensive reflection on our research journey, highlighting its core contributions, potential constraints, and broader implications within the evolving landscape of machine learning and generative modeling. Through introspection and contextualization, we provide readers with a holistic understanding of our work's impact and the promising pathways it paves for future exploration.

6.1 Summary of the Thesis

Improving the training and generalization have always been the most important topics in classic machine learning research. The thesis delves into these two critical aspects respectively related to probabilistic models to enhance their effectiveness and reliability. Firstly, it focuses on identifying the failure modes of training-specific data distribution with common statistical divergences and proposes principled healing techniques to address these failures, thereby improving the training of the probabilistic models. Secondly, we study the generalization ability of likelihood-based generative models and discuss the generalization in both in-distribution and out-of-distribution settings. This aspect has significant implications for applications such as lossless compression and out-of-distribution (OOD) detection.

Our exploration into the nuances of machine learning revolves around the interconnected themes of training and generalization. Addressing training failures using statistical divergences ensures our probabilistic models are built on a solid foundation. Furthermore, a well-trained model's true test is in its ability to generalize,

especially in unpredictable out-of-distribution scenarios. This thesis bridges these two pivotal aspects: the initial act of robust training and the subsequent challenge of reliable generalization. In tandem, they outline our comprehensive effort to craft probabilistic models that excel both in theory and practice.

6.2 Limitations and Future Work

Throughout this thesis, we have delved deep into various facets of probabilistic modeling and machine learning. Our endeavors have unveiled innovative methodologies and insightful revelations. Yet, in the spirit of comprehensive research, it's imperative to discuss the limitations of our work and chart out potential future research directions.

Spread Divergence (Chapter 1): Our introduction of the spread divergence addresses certain shortcomings of the KL divergence. However, the choice of the noise distribution might not always be in perfect alignment with specific data distributions or characteristics. Potential future exploration in this area includes diving deeper into the wide spectrum of spread noise and its inherent properties, pinpointing the most suitable noise types for distinct tasks, and leveraging the principles of spread divergence to enhance the training of different models in addition to the implicit latent variable model.

Mixture Fisher Divergence (Chapter 2): While our findings in this chapter provide a promising foundation, they come with various challenges when applying the method for high-dimensional distributions. Determining an effective $m(x)$ for vast datasets poses a significant hurdle, and the post-training estimation of $Z(\theta)$ for expansive distributions remains intricate. Looking ahead, there is a need to devise efficient algorithms for high-dimensional computations, strategize methods to identify a suitable $m(x)$ for vast datasets, and find innovative methods to precisely estimate $Z(\theta)$ for intricate distributions.

Generalization Gap of VAEs (Chapter 3): Our exploration was centered predominantly around the inference network, leaving the nuanced generalization characteristics of the decoder relatively uncharted. A prospective area of focus is

delving deeper into the generalization attributes of the decoder to enhance the overall VAE performance.

Local/Non-local Image Models (Chapter 4): A salient limitation we encountered here is the critical yet challenging task of choosing the optimal local window size. The correlation between this local window size and out-of-distribution generalization remains enigmatic. Future studies might venture into exploring the manifold image distribution intricacies, aiming to decipher the subtle relation between local window size and OOD generalization.

In conclusion, our research has marked significant milestones, but it's evident that there are several frontiers yet to be explored. The limitations and potential avenues highlighted here provide a roadmap for subsequent scholarly endeavors in the domain.

6.3 Implications for the State of the Art

Recent advancements in Stable Diffusion [Rombach et al., 2022] and ChaptGPT [Vaswani et al., 2017, Radford et al., 2019, Brown et al., 2020] have revolutionized generative modeling applications and research, particularly in text-to-image generation and text generation. This section delineates how our thesis contributes to a deeper understanding and potential enhancement of state-of-the-art applications in these domains.

Diffusion models [Song and Ermon, 2019, Ho et al., 2020] have been exceptionally performant in image generation. One can interpret the diffusion model as an energy-based model trained by denoising score matching (DSM) with varied noise levels [Vincent, 2011, Song and Ermon, 2019]. As explored in Section 2.6.1, DSM's central idea is to introduce convolutional noise to the data, subsequently instructing an EBM to adapt to the noised data distribution. While minor noise introduction may have a negligible impact on the visual generation of natural images, the DSM may fall short in tasks demanding precision and noise-free data modeling. An intriguing insight derived from the spread divergence is the potential to introduce equivalent noise amounts to both a pristine and a noisy model. The noisy model can then be

adjusted to align with the noisy data distribution, indirectly ensuring the pristine model adapts to the clean distribution. Extending this concept to enhance diffusion model training remains an exciting area for future work. Consequently, theories and methods developed within the spread divergence paradigm can potentially be applied to the diffusion model. This offers prospects for exploring its attributes further and enhancing its configuration, potentially leading to more proficient and precise image generation. Such advancements hold relevance for numerous applications spanning computer vision, graphics, and multimedia.

Autoregressive modeling forms the cornerstone of GPT-style models [Vaswani et al., 2017, Radford et al., 2019, Brown et al., 2020]. These models have exhibited remarkable generalization in language tasks, underscoring the enigmatic nature of generalization within language models. A recent talk [Sutskever, Year of publication] and a paper [Delétang et al., 2023] both endeavor to unpack the generalization properties of language models, viewing them through a lossless compression lens. This viewpoint aligns with the themes presented in the second part of our thesis. By leveraging both local and global observations, as suggested in our research, we might gain insights into the generalization of syntactic and semantic attributes in expansive language models, marking a promising avenue for future exploration.

In concluding this section, it's evident that our research presented in this thesis not only aligns with, but also strengthens the contemporary advances in generative modeling, setting the groundwork for potential future breakthroughs in both theoretical and practical domains.

Bibliography

Hyvärinen Aapo. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

Felix V Agakov and David Barber. An auxiliary variational method. In *International Conference on Neural Information Processing*, pages 561–566. Springer, 2004.

Faruk Ahmed and Aaron Courville. Detecting semantic anomalies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3154–3162, 2020.

Alexander A Alemi, Ian Fischer, and Joshua V Dillon. Uncertainty in the variational information bottleneck. *arXiv preprint arXiv:1807.00906*, 2018.

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.

Andreas Anastasiou, Alessandro Barp, François-Xavier Briol, Bruno Ebner, Robert E Gaunt, Fatemeh Ghaderinezhad, Jackson Gorham, Arthur Gretton, Christophe Ley, Qiang Liu, et al. Stein's method meets statistics: A review of some recent developments. *arXiv preprint arXiv:2105.03481*, 2021.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

David Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

Alessandro Barp, Francois-Xavier Briol, Andrew Duncan, Mark Girolami, and Lester Mackey. Minimum stein discrepancy estimators. *Advances in Neural Information Processing Systems*, 32, 2019.

Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, pages 573–582. PMLR, 2019.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Rianne van den Berg, Alexey A Gritsenko, Mostafa Dehghani, Casper Kaae Sønderby, and Tim Salimans. Idf++: Analyzing and improving integer discrete flows for lossless compression. *arXiv preprint arXiv:2006.12459*, 2020.

Alain Berlinet and Christine Thomas-Agnan. *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2011.

Olivier Bermond and Jean-François Cardoso. Approximate likelihood for noisy mixtures. In *Proc. ICA*, volume 99, pages 325–330. Citeseer, 1999.

David Berthelot, Thomas Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

David Beymer and Tomaso Poggio. Image representations for visual learning. *Science*, 272(5270):1905–1909, 1996.

Thomas Bird, Friso H Kingma, and David Barber. Reducing the computational cost of deep generative models with binary neural networks. *arXiv preprint arXiv:2010.13476*, 2020.

Christopher M Bishop. Novelty detection and neural network validation. *IEE Proceedings-Vision, Image and Signal processing*, 141(4):217–222, 1994.

Christopher M Bishop. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518): 859–877, 2017.

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–e57, 2006.

Thomas Boutell and T Lane. Png (portable network graphics) specification version 1.0. *Network Working Group*, pages 1–102, 1997.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

George Casella and Roger L Berger. *Statistical inference*. Cengage Learning, 2021.

A Taylan Cemgil, Sumedh Ghaisas, Krishnamurthy Dvijotham, Sven Gowal, and Pushmeet Kohli. Autoencoding variational autoencoder. *Neural Information Processing Systems*, 2020.

Edward Challis and David Barber. Affine independent variational inference. *Advances in Neural Information Processing Systems*, 25:2186–2194, 2012.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Hyunsun Choi, Eric Jang, and Alexander A Alemi. Waic, but why? generative ensembles for robust anomaly detection. *arXiv preprint arXiv:1810.01392*, 2018.

Kacper Chwialkowski, Heiko Strathmann, and Arthur Gretton. A kernel test of goodness of fit. In *International conference on machine learning*, pages 2606–2615. PMLR, 2016.

Rob Cornish, Anthony Caterini, George Deligiannidis, and Arnaud Doucet. Relaxing bijectivity constraints with continuously indexed normalising flows. In *International conference on machine learning*, pages 2133–2143. PMLR, 2020.

Harald Cramér. *Mathematical methods of statistics*, volume 26. Princeton university press, 1999.

Chris Cremer, Xuechen Li, and David Duvenaud. Inference suboptimality in variational autoencoders. In *International Conference on Machine Learning*, pages 1078–1086. PMLR, 2018.

Imre Csiszár. Information-type Measures of Difference of Probability Distributions and Indirect Observation. *studia scientiarum Mathematicarum Hungarica*, 2: 229–318, 1967.

Imre Csiszár. A Class of Measures of Informativity of Observation Channels. *Periodica Mathematica Hungarica*, 2(1-4):191–213, 1972.

Bin Dai and David Wipf. Diagnosing and enhancing vae models. *arXiv preprint arXiv:1903.05789*, 2019.

Francesco D'Angelo and Vincent Fortuin. Annealed stein variational gradient descent. *arXiv preprint arXiv:2101.09815*, 2021.

Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, et al. Language modeling is compression. *arXiv preprint arXiv:2309.10668*, 2023.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum Likelihood from Incomplete Data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. *Probability: Theory and Examples*, volume 49. Cambridge university press, 2019.

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.

Sever S Dragomir. Some General Divergence Measures for Probability Distributions. *Acta Mathematica Hungarica*, 109(4):331–345, Nov 2005. ISSN 1588-2632. doi: 10.1007/s10474-005-0251-6.

Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. *Advances in Neural Information Processing Systems*, 32, 2019.

Jarek Duda. Asymmetric numeral systems: entropy coding combining speed of huffman coding with compression rate of arithmetic coding. *arXiv preprint arXiv:1311.2540*, 2013.

Gintare Karolina Dziugaite, Daniel M Roy, and Zoubin Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. *arXiv preprint arXiv:1505.03906*, 2015.

William Fedus, Mihaela Rosca, Balaji Lakshminarayanan, Andrew M Dai, Shakir Mohamed, and Ian Goodfellow. Many Paths to Equilibrium: GANs Do Not Need to Decrease a Divergence at Every Step. *arXiv:1710.08446*, 2017.

Thomas S Ferguson. An Inconsistent Maximum Likelihood Estimate. *Journal of the American Statistical Association*, 77(380):831–834, 1982.

Gerald B Folland. *Advanced calculus*. Pearson, 2001.

Brendan J Frey and Geoffrey E Hinton. Free energy coding. In *Proceedings of Data Compression Conference-DCC'96*, pages 73–81. IEEE, 1996.

Thomas Furmston and David Barber. Solving Deterministic Policy (PO)MPDs using Expectation-Maximisation and Antifreeze. In *First international workshop on learning and data mining for robotics (LEMIR)*, pages 56–70, 2009. In conjunction with ECML/PKDD-2009.

Matthias Gelbrich. On a Formula for the L2 Wasserstein Metric Between Measures on Euclidean and Hilbert Spaces. *Mathematische Nachrichten*, 147(1):185–203, 1990.

Sebastien Gerchinovitz, Pierre Ménard, and Gilles Stoltz. Fano's Inequality for Random Variables. *arXiv*, 2018. doi: arXiv:1702.05985v2.

Wenbo Gong and Yingzhen Li. Interpreting diffusion score matching using normalizing flow. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, June 2021.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

Jackson Gorham, Andrew B Duncan, Sebastian J Vollmer, and Lester Mackey. Measuring sample quality with diffusions. *The Annals of Applied Probability*, 29 (5):2884–2928, 2019.

Alex Graves, Rupesh Kumar Srivastava, Timothy Atkinson, and Faustino Gomez. Bayesian flow networks. *arXiv preprint arXiv:2308.07037*, 2023.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A Kernel Two-Sample Test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017a.

Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taiga, Francesco Visin, David Vazquez, and Aaron Courville. Pixelvae: A latent variable model for natural images. *International Conference on Learning Representations*, 2017b.

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.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*, 2018.

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. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13, 1993.

Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214): 1158–1161, 1995.

Jonathan Ho, Evan Lohn, and Pieter Abbeel. Compression with flows via local bits-back coding. *arXiv preprint arXiv:1905.08500*, 2019.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Emiel Hoogeboom, Jorn WT Peters, Rianne van den Berg, and Max Welling. Integer discrete flows and lossless compression. *arXiv preprint arXiv:1905.07376*, 2019.

Aapo Hyvärinen. Some extensions of score matching. *Computational statistics & data analysis*, 51(5):2499–2512, 2007.

Daniel Im Im, Sungjin Ahn, Roland Memisevic, and Yoshua Bengio. Denoising Criterion for Variational Auto-Encoding Framework. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167*, 2015.

Oliver Johnson. *Information theory and the central limit theorem*. World Scientific, 2004.

Alexia Jolicoeur-Martineau, Rémi Piché-Taillefer, Ioannis Mitliagkas, and Remi Tachet des Combes. Adversarial score matching and improved sampling for image generation. In *International Conference on Learning Representations*, 2020.

Olav Kallenberg. *Foundations of Modern Probability*. Springer Science & Business Media, 2006.

Leonid V Kantorovich. Mathematical methods of organizing and planning production. *Management science*, 6(4):366–422, 1960.

Taesup Kim and Yoshua Bengio. Deep directed generative models with energy-based probability estimation. *arXiv preprint arXiv:1606.03439*, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*, 2018.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations*, 2014.

Friso Kingma, Pieter Abbeel, and Jonathan Ho. Bit-swap: Recursive bits-back coding for lossless compression with hierarchical latent variables. In *International Conference on Machine Learning*, pages 3408–3417. PMLR, 2019.

Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Why normalizing flows fail to detect out-of-distribution data. *arXiv preprint arXiv:2006.08545*, 2020.

Naveen Kodali, Jacob Abernethy, James Hays, and Zsolt Kira. On Convergence and Stability of GANs. *arXiv:1705.07215*, 2017.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Ryen Krusinga, Sohil Shah, Matthias Zwicker, Tom Goldstein, and David Jacobs. Understanding the (un) interpretability of natural image distributions using generative models. *arXiv preprint arXiv:1901.01499*, 2019.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and Fujie Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems*, pages 7167–7177, 2018.

Erich Leo Lehmann. *Elements of Large-sample Theory*. Springer Science & Business Media, 2004.

Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. MMD GAN: Towards Deeper Understanding of Moment Matching Network. In *Advances in Neural Information Processing Systems*, pages 2203–2213, 2017.

Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In *International conference on machine learning*, pages 1718–1727. PMLR, 2015.

Li Lian and Wei Shilei. Webp: A new image compression format based on vp8 encoding. *Microcontrollers & Embedded Systems*, 3, 2012.

Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

Qiang Liu, Jason Lee, and Michael Jordan. A kernelized stein discrepancy for goodness-of-fit tests. In *International conference on machine learning*, pages 276–284. PMLR, 2016.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Gabriel Loaiza-Ganem, Brendan Leigh Ross, Jesse C Cresswell, and Anthony L Caterini. Diagnosing and fixing manifold overfitting in deep generative models. *arXiv preprint arXiv:2204.07172*, 2022.

Gabriel Loaiza-Ganem, Brendan Leigh Ross, Luhuan Wu, John Patrick Cunningham, Jesse C Cresswell, and Anthony L Caterini. Denoising deep generative models. In *Proceedings on*, pages 41–50. PMLR, 2023.

Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are GANs Created Equal? A Large-Scale Study. In *Advances in Neural Information Processing Systems*, pages 700–709, 2018.

Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary deep generative models. In *International conference on machine learning*, pages 1445–1453. PMLR, 2016.

David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Ahsan Mahmood, Junier Oliva, and Martin Styner. Multiscale score matching for out-of-distribution detection. *arXiv preprint arXiv:2010.13132*, 2020.

Xudong Mao, Qing Li, Haoran Xie, Raymond YK 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, 2017.

Takuo Matsubara, Jeremias Knoblauch, François-Xavier Briol, and Chris Oates. Generalised Bayesian inference for discrete intractable likelihood. *arXiv:2206.08420*, 2022.

Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Practical full resolution learned lossless image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10629–10638, 2019.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks. *arXiv:1802.05957*, 2018.

Shakir Mohamed and Balaji Lakshminarayanan. Learning in Implicit Generative Models. *arXiv*, 2016. doi: arXiv:1610.03483.

Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, Bernhard Schölkopf, et al. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends® in Machine Learning*, 10(1-2):1–141, 2017.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? *International Conference on Learning Representations*, 2019a.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, and Balaji Lakshminarayanan. Detecting out-of-distribution inputs to deep generative models using a test for typicality. *arXiv preprint arXiv:1906.02994*, 5, 2019b.

Jiquan Ngiam, Zhenghao Chen, Pang W Koh, and Andrew Y Ng. Learning deep energy models. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 1105–1112, 2011.

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*, 2016.

Gabriel Peyré and Marco Cuturi. Computational Optimal Transport. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

Majid Rabbani. Jpeg2000: Image compression fundamentals, standards and practice. *Journal of Electronic Imaging*, 11(2):286, 2002.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Aaditya Ramdas, Sashank Jakkam Reddi, Barnabás Póczos, Aarti Singh, and Larry Wasserman. On the Decreasing Power of Kernel and Distance Based Nonparametric Hypothesis Tests in High Dimensions. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Suman Ravuri, Shakir Mohamed, Mihaela Rosca, and Oriol Vinyals. Learning Implicit Generative Models with the Method of Learned Moments. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 4311–4320, 2018. URL http://proceedings.mlr.press/v80/ravuri18a.html.

Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *Advances in neural information processing systems*, pages 14866–14876, 2019.

Scott Reed, Aäron Oord, Nal Kalchbrenner, Sergio Gómez Colmenarejo, Ziyu Wang, Yutian Chen, Dan Belov, and Nando Freitas. Parallel multiscale autoregressive density estimation. In *International Conference on Machine Learning*. PMLR, 2017.

Jie Ren, Peter J Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark Depristo, Joshua Dillon, and Balaji Lakshminarayanan. Likelihood ratios for out-of-distribution detection. In *Advances in Neural Information Processing Systems*, pages 14707–14718, 2019.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and variational inference in deep latent gaussian models. In *International Conference on Machine Learning*, volume 2, page 2. Citeseer, 2014.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Pro-*

ceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 10684–10695, 2022.

Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing Training of Generative Adversarial Networks through Regularization . *arXiv:1705.09367*, 2017.

Yangjun Ruan, Karen Ullrich, Daniel Severo, James Townsend, Ashish Khisti, Arnaud Doucet, Alireza Makhzani, and Chris J Maddison. Improving lossless compression rates via monte carlo bits-back coding. *arXiv preprint arXiv:2102.11086*, 2021.

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

Chandramouli Shama Sastry and Sageev Oore. Detecting out-of-distribution examples with in-distribution examples and gram matrices. *arXiv preprint arXiv:1912.12510*, 2019.

Robin Tibor Schirrmeister, Yuxuan Zhou, Tonio Ball, and Dan Zhang. Understanding anomaly detection with deep invertible networks through hierarchies of distributions and features. *arXiv preprint arXiv:2006.10848*, 2020.

Joan Serrà, David Álvarez, Vicenç Gómez, Olga Slizovskaia, José F Núñez, and Jordi Luque. Input complexity and out-of-distribution detection with likelihood-based generative models. *arXiv preprint arXiv:1909.11480*, 2019.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 5(1):3–55, 2001.

Rui Shu, Hung H Bui, Shengjia Zhao, Mykel J Kochenderfer, and Stefano Ermon. Amortized inference regularization. *Neural Information Processing Systems*, 2018.

Alex Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. A hilbert space embedding for distributions. In *International conference on algorithmic learning theory*, pages 13–31. Springer, 2007.

Jon Sneyers and Pieter Wuille. Flif: Free lossless image format based on maniac compression. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 66–70. IEEE, 2016.

Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised Map Inference for Image Super-Resolution. *arXiv:1610.04490*, 2016.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, 2019.

Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33:12438–12448, 2020.

Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

Bharath K Sriperumbudur, Kenji Fukumizu, and Gert RG Lanckriet. Universality, Characteristic Kernels and RKHS Embedding of Measures. *J. Mach. Learn. Res.*, 12:2389–2410, July 2011. ISSN 1532-4435.

Bharath K Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, and Gert RG Lanckriet. On the Empirical Estimation of Integral Probability Metrics. *Electronic Journal of Statistics*, 6:1550–1599, 2012.

Jan Stanczuk, Christian Etmann, Lisa Maria Kreusser, and Carola-Bibiane Schönlieb. Wasserstein gans work because they fail (to approximate the wasserstein distance). *arXiv preprint arXiv:2103.01678*, 2021.

Ilya Sutskever. An observation on generalization, Year of publication. URL https://www.youtube.com/watch?v=AKMuA_TVz3A. YouTube video.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

Terence Tao. An Introduction to Measure Theory. 2011.

Terence Tao. Analysis ii, texts and readings in mathematics, 2015.

Michael E Tipping and Christopher M Bishop. Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society, Series B*, 21/3:611–622, January 1999.

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein Auto-Encoders. *arXiv:1711.01558*, 2017.

James Townsend. A tutorial on the range variant of asymmetric numeral systems. *arXiv preprint arXiv:2001.09186*, 2020.

James Townsend, Tom Bird, and David Barber. Practical lossless compression with latent variables using bits back coding. *arXiv preprint arXiv:1901.04866*, 2019.

James Townsend, Thomas Bird, Julius Kunze, and David Barber. Hilloc: Lossless image compression with hierarchical latent variable models. *International Conference on Learning Representations*, 2020.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pages 1747–1756. PMLR, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Cédric Villani. *Optimal transport: old and new*, volume 338. Springer, 2009.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Abraham Wald. Note on the Consistency of the Maximum Likelihood Estimate. *The Annals of Mathematical Statistics*, 20(4):595–601, 1949.

Chris S Wallace. Classification by minimum-message-length inference. In *International Conference on Computing and Information*, pages 72–81. Springer, 1990.

Yixin Wang and David M Blei. Frequentist consistency of variational bayes. *Journal of the American Statistical Association*, 114(527):1147–1161, 2019.

Li Wenliang and Heishiro Kanagawa. Blindness of score-based methods to isolated components and mixing proportions. *arXiv preprint arXiv:2008.10087*, 2020.

Li Wenliang, Danica J Sutherland, Heiko Strathmann, and Arthur Gretton. Learning

deep kernels for exponential family densities. In *International Conference on Machine Learning*, pages 6737–6746. PMLR, 2019.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

Ian H Witten, Radford M Neal, and John G Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

Yuhuai Wu, Yuri Burda, Ruslan Salakhutdinov, and Roger Grosse. On the quantitative analysis of decoder-based generative models. *arXiv preprint :1611.04273*, 2016.

Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A theory of generative convnet. In *International Conference on Machine Learning*, pages 2635–2644. PMLR, 2016.

Yibo Yang, Robert Bamler, and Stephan Mandt. Improving inference for neural image compression. *arXiv preprint arXiv:2006.04240*, 2020.

Lantao Yu, Yang Song, Jiaming Song, and Stefano Ermon. Training deep energy-based models with f-divergence minimization. In *International Conference on Machine Learning*, pages 10957–10967. PMLR, 2020.

Shuangfei Zhai, Yu Cheng, Rogerio Feris, and Zhongfei Zhang. Generative adversarial networks as variational training of energy based models. *arXiv preprint arXiv:1611.01799*, 2016.

Mingtian Zhang, Thomas Bird, Raza Habib, Tianlin Xu, and David Barber. Variational f-divergence minimization. *arXiv preprint arXiv:1907.11891*, 2019a.

Mingtian Zhang, Thomas Bird, Raza Habib, Tianlin Xu, and David Barber. Variational f-divergence minimization. *arXiv preprint arXiv:1907.11891*, 2019b.

Mingtian Zhang, Andi Zhang, and Steven McDonagh. On the out-of-distribution generalization of probabilistic image modelling. In *Advances in Neural Information Processing Systems*, 2021.

Mingtian Zhang, Tim Z Xiao, Brooks Paige, and David Barber. Improving vae-based representation learning. *arXiv preprint arXiv:2205.14539*, 2022.

Mingtian Zhang, Alex Hawkins-Hooker, Brooks Paige, and David Barber. Moment matching denoising gibbs sampling. *Advances in Neural Information Processing Systems*, 2023a.

Mingtian Zhang, Yitong Sun, Chen Zhang, and Steven Mcdonagh. Spread flows for manifold modelling. In *International Conference on Artificial Intelligence and Statistics*, pages 11435–11456. PMLR, 2023b.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*, 2017.

Shengjia Zhao, Hongyu Ren, Arianna Yuan, Jiaming Song, Noah Goodman, and Stefano Ermon. Bias and generalization in deep generative models: An empirical study. *arXiv preprint arXiv:1811.03259*, 2018.

Appendix A

Appendix of Chapter 2

A.1 Annealing the Noise

In section(2.1.1) we discussed the common approach to first adding noise to a model $\mathbb{Q}$ in order to define a proper density and then using maximum likelihood to fit that 'noised model' to data. We can use standard Woodberry identities to rewrite the expected log-likelihood equation 2.8 as

$$-\frac{\theta_p^2}{\sigma^2}+\frac{\left(\theta_p^\top\theta_q\right)^2}{\sigma^2\left(\sigma^2+\theta_q^2\right)}-\log\left(1+\frac{\theta_q^2}{\sigma^2}\right)-D\log\sigma^2. \tag{A.1}$$

where D is the dimension of θ_p.

Differentiating wrt θ_q, we note that the optimal solution is given when

$$\theta_q=\gamma\theta_p, \tag{A.2}$$

for scalar γ. Plugging this form back into equation A.1 we find that the optimum is obtained when

$$\theta_q=\sqrt{\frac{\theta_p^2-\sigma^2}{\theta_p^2}}\theta_p. \tag{A.3}$$

For finite Gaussian noise $\sigma^2>0$ the resulting estimator for the toy model in section(2.1.1) is therefore not consistent.

A natural question is what would happen if one uses a numerical optimisation of equation A.1 but anneals the noise σ^2 to zero during the optimisation process?

As σ^2 tends to zero, the expression equation A.1 blows up. This means that a naive approach to annealing σ^2 towards zero whilst using a standard optimisation technique is unlikely to result in θ_q converging to θ_p. However, if one considers removing the additive constant $D\log\sigma^2$ and multiplying the remaining objective by σ^2, the resulting quantity

$$\frac{\left(\theta_p^\mathsf{T}\theta_q\right)^2}{\left(\sigma^2+\theta_q^2\right)} - \sigma^2\log\left(1+\frac{\theta_q^2}{\sigma^2}\right), \tag{A.4}$$

is well-behaved as $\sigma^2 \to 0$, as plotted in figure(A.1).

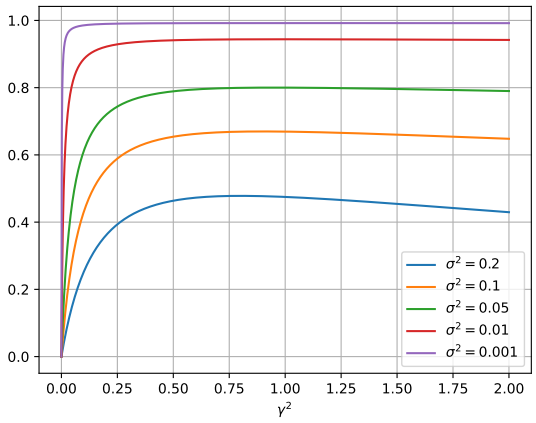


Figure A.1: The (modified) expected log likelihood equation A.4 when adding noise σ^2 to the model only and for unit length true data generating parameter $\theta_p^2 = 1$. The x-axis is the value γ^2 assuming that the optimal θ_q is of the form $\theta_q = \gamma\theta_p$. As we see, as $\sigma^2 \to 0$ the scaled objective becomes flat around the optimum point $\gamma^2 = 1$.

Nevertheless, the objective equation A.4 becomes flat (with respect to θ_q) around the optimum as $\sigma^2 \to 0$. In figure(A.1) we plot the scaling behaviour of the objective equation A.4, assuming $\theta_q = \gamma\theta_p$, showing how it becomes flat with respect to γ as σ^2 is annealed towards zero. This means that a standard first-order numerical optimisation approach, even for this modified objective, will result in a 'critical slowing down' phenomenon, leading to θ_q not updating. This might be fixable by taking the curvature of the objective into consideration.

However, addressing all the above issues requires an understanding of the small σ^2 behaviour of the original objective; dealing with arbitrarily large constants,

arbitrarily large scaling and loss of curvature. In general, such insight is unlikely to be available for any given implicit generative model. Thus, we are doubtful that it will be possible to find an annealing schedule and associated general numerical optimisation procedure that will result in a consistent estimator.

A.2 Noise Requirements for Discrete Distributions

Our main interest is to define a new divergence in situations where the original divergence $\mathrm{D}(p||q)$ is itself not defined. For discrete variables $x \in \{1,\ldots,n\}$, $y \in \{1,\ldots,n\}$, the spread $P_{ij} = p(y=i|x=j)$ must be a distribution; $\sum_i P_{ij} = 1$, $P_{ij} \geq 0$, and

$$\tilde{p}_i \equiv \sum_j P_{ij} p_j = \sum_j P_{ij} q_j \equiv \tilde{q}_i \quad \forall i \tag{A.5}$$

$$\Rightarrow \quad p_j = q_j \quad \forall j, \tag{A.6}$$

which is equivalent to the requirement that the matrix P is invertible. In addition, for the Spread Divergence to exist in the case of f-divergences, $\tilde{p}$ and $\tilde{q}$ must have the same support. This requirement is guaranteed if

$$\sum_j P_{ij} p_j > 0, \quad \sum_j P_{ij} q_j > 0 \quad \forall i, \tag{A.7}$$

which is satisfied if $P_{ij}>0$. Therefore, in general, there is a space of spread distributions $p(y|x)$ that define a valid Spread Divergence in the discrete case.

A.3 Spread Noise Makes Distributions More Similar

The data processing inequality for f-divergences [Gerchinovitz et al., 2018] states that $\mathrm{D}_f(\tilde{p}(y)||\tilde{q}(y)) \leq \mathrm{D}_f(p(x)||q(x))$. For completeness, we provide here an elementary proof of this result. We consider the following joint distributions with densities

$$q(y,x) = p(y|x)q(x), \qquad p(y,x) = p(y|x)p(x), \tag{A.8}$$

whose marginals are the spreaded distributions

$$\tilde{p}(y) = \int p(y|x)p(x)\,\mathrm{d}x, \qquad \tilde{q}(y) = \int p(y|x)q(x)\,\mathrm{d}x. \tag{A.9}$$

The divergence between the two joint distributions is

$$\mathrm{D}_f(p(y,x)||q(y,x)) = \int q(y,x) f\left(\frac{p(y|x)p(x)}{p(y|x)q(x)}\right)\mathrm{d}x\,\mathrm{d}y = \mathrm{D}_f(p(x)||q(x))\,. \tag{A.10}$$

More generally, the f-divergence between two marginal distributions is no larger than the f-divergence between the joint [Zhang et al., 2019a]. To see this, consider

$$\begin{aligned}
\mathrm{D}_f(p(u,v)||q(u,v)) &= \int q(u) \int q(v|u) f\left(\frac{p(u,v)}{q(u,v)}\right)\mathrm{d}y du && \text{(A.11)}\\
&\geq \int q(u) f\left(\int q(v|u)\frac{p(u,v)}{q(v|u)q(u)} dv\right) du && \text{(A.12)}\\
&= \int q(u) f\left(\frac{p(u)}{q(u)}\right) du = \mathrm{D}_f(p(u)||q(u))\,. && \text{(A.13)}
\end{aligned}$$

Hence,

$$\widetilde{\mathrm{D}}_f(q(x)||p(x)) \equiv \mathrm{D}_f(\tilde{p}(y)||\tilde{q}(y)) \leq \mathrm{D}_f(p(y,x)||q(y,x)) = \mathrm{D}_f(p(x)||q(x))\,. \tag{A.14}$$

Intuitively, spreading two distributions increases their overlap, reducing the divergence. When the distributions $\mathbb{P}$ and $\mathbb{Q}$ are absolutely continuous and their densities p and q have the same support, the spread f-divergence is always a lower bound of f-divergence. When the densities do not have the same support or are not well defined, then $\mathrm{D}_f(\mathbb{P}||\mathbb{Q})$ is not well-defined.

A.4 Mixture Divergence

We motivated the Spread Divergence between distribution $\mathbb{P}$ and $\mathbb{Q}$ by the requirement to produce a divergence that satisfying $\widetilde{\mathrm{D}}(\mathbb{P}||\mathbb{Q}) = 0 \Rightarrow \mathbb{P} = \mathbb{Q}$, where the original $\mathrm{D}(\mathbb{P}||\mathbb{Q})$ does not exist. We briefly discuss the case that $\mathbb{P}$ and $\mathbb{Q}$ are absolutely continuous but their density functions p and q have different supports, so

f-divergence $\mathrm{D}_f(\mathbb{P}||\mathbb{Q}) = \mathrm{D}(p||q)$ is still not defined. For example, $\mathbb{P}$ and $\mathbb{Q}$ can be two uniform distributions with different supports. We mention here an alternative divergence that also can be used , namely a mixture divergence, and discuss why we focus on the Spread Divergence thereafter. Specifically, we can define a mixture model with density $\tilde{p}(x)$ of the original distribution and a 'noise' distribution with density function $n(x)$:

$$\tilde{p}(x) = \alpha p(x) + (1-\alpha)n(x) \tag{A.15}$$

for $0 < \alpha < 1$. Provided $n(x)$ is non-zero, then $\tilde{p}(x)$ has support everywhere. Similarly, we can define

$$\tilde{q}(x) = \alpha q(x) + (1-\alpha)n(x). \tag{A.16}$$

As with the Spread Divergence formulation presented previously, this will usually enable us to define a divergence $\mathrm{D}(\tilde{p}||\tilde{q})$ when $\operatorname{supp} p \neq \operatorname{supp} q$. Furthermore, provided the divergence between $\tilde{p}$ and $\tilde{q}$ is zero, then the two distributions $\tilde{p}$ and $\tilde{q}$ match, as do the original distributions p and q since

$$\tilde{p}(x) = \tilde{q}(x) \Leftrightarrow \alpha p(x) + (1-\alpha)n(x) = \alpha q(x) + (1-\alpha)n(x) \Leftrightarrow p(x) = q(x). \tag{A.17}$$

Therefore, creating a mixture model in this way also allows us to define a divergence between absolutely continuous distributions that otherwise would not have an appropriate divergence. However, in contrast to the Spread Divergence formulation, we cannot use this approach for distributions that are not absolutely continuous, which for many applications of interest cannot be achieved. As a simple example, consider generalised densities $p(x) = \delta(x-\mu_p)$, $q(x) = \delta\left(x-\mu_q\right)$ with

$$\tilde{p}(x) = \alpha\delta(x-\mu_p) + (1-\alpha)n(x), \qquad \tilde{q}(x) = \alpha\delta\left(x-\mu_q\right) + (1-\alpha)n(x). \tag{A.18}$$

In this case, the divergence $\mathrm{D}(\tilde{p}(x)||\tilde{q}(x))$ is not defined since neither $\tilde{p}(x)$ nor $\tilde{q}(x)$ is a valid probability density. A similar issue arises in training implicit generative models in which a value cannot be feasibly computed for $\tilde{p}$ or $\tilde{q}$; see section(2.5.3). Hence, for implicit models in, we cannot feasibly assign a value to this mixture

divergence. As such it appears to have only limited value in training continuous variable models.

One can combine the spread and the mixture approaches to produce a more general affine divergence

$$\tilde{p}(y) = \alpha \int p(y|x)p(x)\,\mathrm{d}x + (1-\alpha)n(y), \tag{A.19}$$

for spread $p(y|x)$ and (generalised) density $p(x)$. It follows for this case that $\mathrm{D}(\tilde{p}||\tilde{q}) = 0 \Leftrightarrow \mathbb{P} = \mathbb{Q}$; however, the benefit of the mixture noise over the spread noise is not clear. Our central interest in this work is to train implicit models and, as such, we focus interest only on the first 'spread' term $\int_x p(y|x)p(x)$ in equation A.19 and leave the study of the potential additional benefits of including a mixture component $n(y)$ for future work.

A.5 Statistical Properties of Spread MLE

A.5.1 Existence of Spread MLE

In some situations there may not exist a Maximum Likelihood Estimator (MLE) for $p(x|\theta)$, but there can exist a MLE for the spread model $p(y|\theta) = \int p(y|x)p(x|\theta)\,\mathrm{d}x$. For example, suppose that $X \sim \mathcal{N}(\mu, \sigma^2)$ $(\mu, 0 < \sigma^2 < \infty)$. So $\theta = (\mu, \sigma^2) \in \mathbb{R} \times \mathbb{R}^+$. Assume we only have one data point x. Then the log-likelihood function is $L(x;\theta) \propto -\log\sigma - \frac{1}{2\sigma^2}(x-\mu)^2$. Maximising with respect to μ, we have $\mu = x$ and the log-likelihood becomes unbounded as $\sigma^2 \to 0$. In this sense, the MLE for (μ, σ^2) does not exist, see Casella and Berger [2021] for more discussions.

In contrast, we can check whether the MLE for $p(y|\theta)$ exists. We assume Gaussian spread noise with fixed variance σ_f^2. Since we only have one data point x, the spread data distribution becomes $p(y|x) = \mathcal{N}(y|x, \sigma_f^2)$, and the model is $p(y|\theta) = \mathcal{N}(y|\mu, \sigma^2 + \sigma_f^2)$. We can sample N points from the spread model, so the spread log likelihood function is (neglecting constants) $L(y_1, \ldots, y_N; \theta) = -\frac{N}{2}\log(\sigma^2 + \sigma_f^2) - \frac{1}{2(\sigma^2+\sigma_f^2)}\sum_{i=1}^N (y_i - \mu)^2$. The MLE solution for μ is $\mu = \frac{1}{N}\sum_{i=1}^N y_i$; the MLE solution for σ^2 is $\sigma^2 = \frac{1}{N}\sum_i (y_i - \mu)^2 - \sigma_f^2$, which has bounded spread likelihood

value. Note that in the limit of a large number of spread samples $N \to \infty$, the MLE $\sigma^2 = \frac{1}{N}(y_i - \mu)^2 \to \sigma_f^2$ tends to 0. Throughout, however, the (scaled by N) log likelihood remains bounded.

A.5.2 Consistency

Consistency of an estimator is an important property that guarantees the validity of the resulting estimate at convergence as the number of data points tends to infinity. In what follows, we outline the sufficient conditions for a consistent MLE estimator, before addressing the question of whether using spread MLE is also consistent and under what conditions.

A.5.2.1 Consistency for MLE

Sufficient conditions for the MLE being consistent and converging to the *global* maximum are given in Wald [1949]. However, they are usually difficult to check even for some standard distributions. The sufficient conditions for MLE being consistent and converging to a *local* maxima are given in Cramér [1999] and are more straight forward to check:

C1. (Identifiable): $p(x|\theta_1) = p(x|\theta_2) \to \theta_1 = \theta_2$.

C2. The parameter space Θ is an open interval $(\underline{\theta}, \bar{\theta})$, $\Theta : -\infty \leq \underline{\theta} < \theta < \bar{\theta} \leq \infty$.

C3. $p(x|\theta)$ is continuous in θ and differentiable with respect to θ for all x.

C4. The set $A = \{x : p_\theta(x) > 0\}$ is independent of θ.

Let $X_1, X_2, \ldots$ be $i.i.d$ with density $p(x|\theta_0)$ $(\theta \in \Theta)$ satisfying conditions C1–C4, then there exists a sequence $\hat{\theta}_n = \hat{\theta}_n(X_1, ..., X_n)$ of local maxima of the likelihood function $L(\theta_0) = \prod_{i=1}^{n} p(x_i|\theta_0)$ which is consistent:

$$\hat{\theta} \xrightarrow{p} \theta_0 \quad \text{for all } \theta \in \Theta.$$

The proof can be found in Lehmann [2004] or Cramér [1999].

A.5.2.2 Consistency of spread MLE

We provide the necessary conditions for Spread MLE being consistent.

C1. (Identifiable): $p(x|\theta)$ is identifiable. From section(2.2.1) it follows immediately that $p(y|\theta_1) = p(y|\theta_2) \rightarrow p(x|\theta_1) = p(x|\theta_2) \rightarrow \theta_1 = \theta_2$, where the final implication follows from the assumption that $p(x|\theta)$ is identifiable. Hence if $p(x|\theta)$ is identifiable, so is $p(y|\theta)$.

C2. The parameter space Θ is an open interval $(\underline{\theta}, \bar{\theta})$, $\Theta : -\infty \leq \underline{\theta} < \theta < \bar{\theta} \leq \infty$. This condition is unchanged for $p(y|\theta)$.

C3. On $p(y|\theta)$, we require the same condition on $p(x|\theta)$ as in MLE; $p(y|\theta)$ is continuous in θ and differentiable with respect to θ for all y.

C4. For spread noise $p(y|x)$ who has full support on $\mathbb{R}^d$ (for example Gaussian noise), $p(y|\theta)$ is greater than zero everywhere and hence the original condition C4 is automatically guaranteed.

The conditions that guarantee consistency for spread MLE are weaker for the spread model $p(y|\theta)$ than for the standard model $p(x|\theta)$, since C4 is automatically satisfied. [Ferguson, 1982] gives an example for which MLE exists but is not consistent by violating condition C4, whereas spread MLE can be used to obtain a consistent estimator.

A.5.3 Asymptotic Efficiency

A key desirable property of any estimator is that it is efficient. The Cramer-Rao bound places a lower bound on the variance of any unbiased estimator and an efficient estimator must reach this minimal value in the limit of a large amount of data. Under certain conditions (see below) the Maximum Likelihood Estimator attains this minimal variance value meaning that there is no better estimator possible (in the limit of a large amount of data). This is one of the reasons that the maximum likelihood is a cherished criterion.

A.5.3.1 Asymptotic Efficiency for MLE

Building upon conditions C1-C4, additional conditions on $p(x|\theta)$ are required to show MLE is asymptotical efficient:

C5. For all x in its support, the density $p_\theta(x)$ is three times differentiable with respect to θ and the third derivative is continuous.

C6. The derivatives of the integral $\int p_\theta(x)\,\mathrm{d}x$ respect to θ can be obtained by differentiating under the integral sign, that is: $\nabla_\theta \int p_\theta(x)\,\mathrm{d}x = \int \partial_\theta p_\theta(x)\,\mathrm{d}x$.

C7. There exists a positive number $c(\theta_0)$ and a function $M_{\theta_0}(x)$ such that

$$\left|\frac{\partial^3}{\partial\theta^3}\log p_\theta(x)\right| \leq M_{\theta_0}(x) \quad \text{for all } x \in A, |\theta-\theta_0| < c(\theta_0),$$

where A is the support set of x and $\mathbb{E}_{\theta_0}\left[M_{\theta_0}(x)\right] < \infty$.

Let $X_1,...,X_n$ be *i.i.d* with density $p_\theta(x)$ and satisfy conditions C1-C7, then any consistent sequence $\hat{\theta} = \hat{\theta}_n(X_1,...,X_n)$ of roots of the likelihood equation satisfies

$$\sqrt{n}(\hat{\theta}-\theta_0) \xrightarrow{d} \mathcal{N}\left(0, F(\theta_0)^{-1}\right), \tag{A.20}$$

where $F^{-1}(\theta_0)$ is the inverse of Fisher information matrix (also called Cramér-Rao Lower Bound, which is a lower bound on variance of any unbiased estimators). The conditions and proof can be found in [Lehmann, 2004].

A.5.3.2 Asymptotic Efficiency for MLE

As with MLE above, we require further conditions on $p(y|\theta)$ for ensuring spread MLE is asymptotically efficient:

C5. On $p(y|\theta)$, we require the same condition as applied to $p(x|\theta)$ in the MLE case; for all y in its support, the density $p_\theta(y)$ is three times differentiable with respect to θ and the third derivative is continuous.

C6. For spread noise $p(y|x)$, which has full support on $\mathbb{R}^d$ (for example Gaussian

noise), the support of y is independent of θ. Leibniz's rule[a] allows us to differentiate under the integral: $\nabla_\theta \int p_\theta(y)\,\mathrm{d}y = \int \partial_\theta p_\theta(y)\,\mathrm{d}y$, so this condition is automatically satisfied.

C7. On $p(y|\theta)$, we require the same condition as applied to $p(x|\theta)$ in the MLE case; There exist positive number $c(\theta_0)$ and a function $M_{\theta_0}(y)$ such that

$$\left|\frac{\partial^3}{\partial\theta^3}\log p_\theta(y)\right| \leq M_{\theta_0}(y) \quad \text{for all } y \in A, |\theta-\theta_0| < c(\theta_0),$$

where A is the support set of y and $\mathbb{E}_{\theta_0}\left[M_{\theta_0}(y)\right] < \infty$.

Thus the conditions that guarantee asymptotic efficiency for the spread model $p(y|\theta)$ are weaker than for the standard model $p(x|\theta)$, since C4 and C6 are automatically satisfied.

A.6 MNIST Experiment

We first scaled the MNIST data to lie in $[0,1]$. We use Laplace spread noise with $\sigma = 0.3$ and Gaussian spread noise with $\sigma = 0.3$ for the δ-VAE case. Both the encoder and the decoder networks contain 3 feed-forward layers, each layer has 400 units and use ReLu activation functions. The latent dimension is $Z = 64$. The variational inference network $q_\phi(z|y) = \mathcal{N}(z|\mu_\phi(y), \sigma_\phi^2 I_Z)$ has a similar structure for the mean network $\mu_\phi(y)$. For fixed spread δ-VAE , learning was done using the Adam [Kingma and Ba, 2014] optimizer with learning rate $5e^{-4}$ for 200 epochs. For δ-VAE with learned spread (learned covariance), we interleave 2 covariance training epochs with 10 model training epochs (using the Adam optimizer with learning rate $5e^{-5}$).

A.7 CelebA Experiment

We pre-processed CelebA images by first taking 140x140 centre crops and then resizing to 64x64. Pixel values were then rescaled to lie in $[0,1]$. For the learned

[a] Leibniz's rule tells us: $\frac{d}{d\theta}\int_{a(\theta)}^{b(\theta)} p(x,\theta)\,\mathrm{d}x = \int_{a(\theta)}^{b(\theta)} \partial_\theta p(x,\theta)\,\mathrm{d}x + p(b(\theta),\theta)\frac{d}{d\theta}b(\theta) - p(a(\theta),\theta)\frac{d}{d\theta}a(\theta)$, so if $a(\theta)$ and $b(\theta)$ are independent of θ, then $\frac{d}{d\theta}\int_a^b p(x,\theta)\,\mathrm{d}x = \int_a^b \partial_\theta p(x,\theta)\,\mathrm{d}x$.

spread we use Gaussian noise with a learned injective function ResNet $f_\psi(\cdot) = I(\cdot) + g_\psi(\cdot)$, where $g_\psi(\cdot)$ is a one layer convolutional neural net with kernel size 3×3, with stride length 1. We use spectral normalization [Miyato et al., 2018] to satisfy the Lipschitz constraint. That is, we replace the weight matrix w of the convolution kernel by $w_{SN}(w) := c \times w/\sigma(w)$, where $\sigma(w)$ is the spectral norm of w and $c \in (0,1)$. This guarantees that f_ψ is invertible - see Behrmann et al. [2019].

The encoder and decoder are 4-layer convolutional neural networks with batch norm [Ioffe and Szegedy, 2015]. Both networks use a fully convolutional architecture with 5x5 convolutional filters with stride length 2 in both vertical and horizontal directions, except the last deconvolution layer where we use stride length 1. Conv_k represents a convolution with k filters and DeConv_k represents a deconvolution with k filters, BN for the batch normalization [Ioffe and Szegedy, 2015], Relu for the rectified linear units, and FC_k for the fully connected layer mapping to $\mathbb{R}^k$.

$$
\begin{aligned}
x \in \mathbb{R}^{64\times64\times3} &\to \text{injective} f(\cdot) \in \mathbf{R}^{64\times64\times3} \\
&\to \text{Conv}_{128} \to \text{BN} \to \text{Relu} \\
&\to \text{Conv}_{256} \to \text{BN} \to \text{Relu} \\
&\to \text{Conv}_{512} \to \text{BN} \to \text{Relu} \\
&\to \text{Conv}_{1024} \to \text{BN} \to \text{Relu} \to \text{FC}_{100}
\end{aligned}
$$

$$
\begin{aligned}
z \in \mathbb{R}^{100} &\to \text{FC}_{10\times10\times1024} \\
&\to \text{DeConv}_{512} \to \text{BN} \to \text{Relu} \\
&\to \text{DeConv}_{256} \to \text{BN} \to \text{Relu} \\
&\to \text{DeConv}_{128} \to \text{BN} \to \text{Relu} \to \text{DeConv}_3 \to \textit{sigmoid}(\cdot) \\
&\to \text{injective} f(\cdot) \in \mathbb{R}^{64\times64\times3}
\end{aligned}
$$

We use batch size 100 and latent dimension 100 in all CelabA experiments. For the δ-VAE with fixed spread, we use the fixed Gaussian noise with 0 mean and

$(0.5)^2 I$ covariance. We train the model for 500 epochs using Adam optimizer with learning rate $1e^{-4}$. We decay the learning rate with scaling factor 0.9 every 100000 iterations.

For the δ-VAE with learned spread we first train a δ-VAE with fixed $f(x) = x$ and fixed Gaussian noise with 0 mean and $(0.5)^2 I$ diagonal covariance for 300 epochs. We decay the learning with scaling factor 0.9 every 100000 iterations. We start iterative training by doing one step inner maximisation over the Spread Divergence parameters ψ using Adam optimizer with learning rate $1e^{-5}$ and one step minimization over the model parameter's (θ, ϕ) using Adam optimizer for additional 200 epochs. We can share the first 300 epochs between the two models. When we sample form two models, we first sample from a 100 dimensional standard Gaussian distribution $z \sim \mathcal{N}(0, I)$ and use the same latent code z to get samples from both δ-VAE with fixed and learned spread, so we can easily compare the sample quality between two models.

Appendix B

Appendix of Chapter 3

B.1 Derivations and Proofs

B.1.1 Derivation of Equation 3.3

Let two differentiable densities g_1 and g_2 have disjoint supports $\mathscr{X}_1 \cap \mathscr{X}_2 = \emptyset$ and

$$p(x) = \alpha_p g_1(x) + (1-\alpha_p) g_2(x), \quad q(x) = \alpha_q g_1(x) + (1-\alpha_q) g_2(x). \tag{B.1}$$

The FD between p and q can be written as

$$\mathrm{FD}(p||q) = \frac{\alpha_p}{2} \int_{\mathscr{X}_1} g_1(x) ||s_p(x) - s_q(x)||_2^2 \, \mathrm{d}x + \frac{1-\alpha_p}{2} \int_{\mathscr{X}_2} g_2(x) ||s_p(x) - s_q(x)||_2^2 \, \mathrm{d}x. \tag{B.2}$$

Since g_1 and g_2 has disjoint support, so g_2 will be a zero function on the support of g_1, so $g_2(x') = \nabla_x g_2(x') = 0$ for $x' \in \mathscr{X}_1$. We then have

$$s_p(x') = \frac{\alpha_p \nabla g_1(x') + \cancel{(1-\alpha_p) \nabla g_2(x')}}{\alpha_p g_1(x') + \cancel{(1-\alpha) g_2(x')}} = \frac{\alpha_p \nabla_x g_1(x')}{\alpha_p g_1(x')} = s_{g_1}(x'), \tag{B.3}$$

and

$$s_q(x') = \frac{\alpha_q \nabla g_1(x') + \cancel{(1-\alpha_q) \nabla g_2(x')}}{\alpha_q g_1(x') + \cancel{(1-\alpha) g_2(x')}} = \frac{\alpha_q \nabla_x g_1(x')}{\alpha_q g_1(x')} = s_{g_1}(x'), \tag{B.4}$$

Similarly, for $x' \in \mathscr{X}_2$ we have $s_p(x') = s_q(x') = s_{g_2}(x')$. Therefore, the FD is equivalent to

$$\mathrm{FD}(p||q) = \frac{\alpha_p}{2} \int_{\mathscr{X}_1} g_1(x) ||s_{g_1}(x) - s_{g_1}(x)||_2^2 \mathrm{d}x + \frac{1-\alpha_p}{2} \int_{\mathscr{X}_2} g_2(x) ||s_{g_2}(x) - s_{g_2}(x)||_2^2 \mathrm{d}x = 0, \quad \text{(B.5)}$$

which is independent of α_q.

B.1.2 Proof of Theorem 2

The following two lemmas can be found in Folland [2001, Corollary 2.41 and Theorem 2.42]. For completeness, we also provide simplified proofs.

Lemma 5. *Suppose $f : \mathscr{X} \to \mathbb{R}$ is differentiable on an open convex set $\mathscr{X} \subseteq \mathbb{R}^d$ and $\nabla_x f(x) = 0$ for all $x \in \mathscr{X}$, then f is a constant on $\mathscr{X}$.*

Proof. For any two points $x_1, x_2 \in \mathscr{X}$, we denote the the line segment that connects a, b as L_{x_1,x_2}. Since $\mathscr{X}$ is a convex set, then $L_{x_1,x_2} \subseteq \mathscr{X}$. By the Mean Value Theorem (see Folland [2001, Theorem 2.39]), there exists a point $x_3 \in L_{x_1,x_2}$ such that $f(x_2) - f(x_1) = \nabla_x f(x_3)(x_2 - x_1)$. Since $x_3 \in S$, so $\nabla_x f(x_3) = 0$ thus $f(x_2) = f(x_1)$. Therefore, f has to be a constant function. □

Lemma 6. *Suppose $f : \mathscr{X} \to \mathbb{R}$ is differentiable on a connected open set $\mathscr{X} \subseteq \mathbb{R}^d$ and $\nabla_x f(x) = 0$ for all $x \in \mathscr{X}$, then f is a constant on $\mathscr{X}$.*

Proof. For any point $a \in \mathscr{X}$, we define $\mathscr{X}_1 = \{x \in \mathscr{X} : f(x) = f(a)\}$ and $\mathscr{X}_2 = \{x \in \mathscr{X} : f(x) \neq f(a)\}$, so $\mathscr{X} = \mathscr{X}_1 \cup \mathscr{X}_2$ by construction. For every $x \in \mathscr{X}_1$, there is a ball $B \in S$ centred at x. Since B is convex, we have $B \in \mathscr{X}_1$ by Lemma 5. Therefore, every point $x \in \mathscr{X}_1$ is an interior point of $\mathscr{X}_1$, so $\mathscr{X}_1$ is an open set. The image of $\mathscr{X}_2$ under f: $\mathbb{R} \setminus \{f(a)\}$ is an open set, so $\mathscr{X}_2$ is a open set since f is a continuous function (see Folland [2001, Theorem 1.33]). We thus have both $\mathscr{X}_1$ and $\mathscr{X}_2$ are open sets and $\mathscr{X}_1$ is non-empty (it contains a). Since any connected space cannot be written as an union of two disjoint non-empty sets (see Tao [2015, Definition 2.4.1]), so $\mathscr{X} = \mathscr{X}_1 \cup \mathscr{X}_2$ indicates $\mathscr{X}_2 = \emptyset$. Therefore, f is a constant function. □

We can then prove the Theorem 2. For two a.c. distributions that are supported on a connected space $\mathscr{X} \subseteq \mathbb{R}^d$ with differentiable density p and q. Then $\mathrm{FD}(p||q) = 0 \Leftrightarrow \nabla_x \log p(x) = \nabla_x \log q(x)$ for $x \in S$. We define function $f(x) = \log p(x) - \log q(x)$, so $f(x)$ differentiable on $\mathscr{X}$ and $\nabla_x f(x) = 0$. By Lemma 6, we have f as a constant function (we denote as c) so we have $p = q\exp(c)$. Since p and q are densities, we have $\int q(x)\exp(c)\,\mathrm{d}x = 1 \Leftrightarrow c = 0$. Therefore, $\mathrm{FD}(p||q) = 0 \Leftrightarrow p = q$.

B.1.3 Proof of Theorem 3

Since we can always represents a distribution with disjoint support set as a mixture distribution with components supported on several connected subsets, we can then prove the theorem by Proposition 1.

Proposition 1 (FD is ill-defined on disconnected sets)**.** *Let a set of a.c. distributions have differentiable densities* $\{g_1, \cdots, g_K\}$ *with mutual disjoint (disconnected) support sets* $\{\mathscr{X}_1, \cdots, \mathscr{X}_K\}$*:* $\mathscr{X}_i \bigcap \mathscr{X}_j = \emptyset$ *for any* $i \neq j$ *and each support* $\mathscr{X}_i$ *is connected. Let two densities* $p = \sum_k \alpha_p^k g_k$ *and* $q = \sum_k \alpha_q^k g_k$ *with positive coefficients* $\sum_k \alpha_p^k = 1$ *and* $\sum_{k=1} \alpha_q^k = 1$*. Then* $\mathrm{FD}(p||q) = 0 \Leftrightarrow \alpha_p^k = \alpha_q^k e^{c_k}$*, where* $\{c_1, \cdots, c_K\}$ *is a set of constants with constraints* $\sum_k e^{c_k} = 1$*.*

We can decompose $\mathrm{FD}(p||q) = \frac{1}{2}\sum_{k=1}^K \alpha_p^k \int_{\mathscr{X}_k} g_k(x)||s_p(x) - s_q(x)||_2^2\,\mathrm{d}x$. Since α_p^k and g_k are positive, $\mathrm{FD}(p||q) = 0 \Rightarrow \int_{\mathscr{X}_k} g_k(x)||s_p(x) - s_q(x)||_2^2\,\mathrm{d}x = 0$ for any k, so $\nabla_x \log p(x) = \nabla_x \log q(x)$ for $x \in \bigcup_{k=1}^K \mathscr{X}_k$. Since $\mathscr{X}_k$ is connected, by Lemma 6, we have for $x \in \mathscr{X}_k$, $\log p(x) - \log q(x) = c_k \Leftrightarrow p(x) = q(x)e^{c_k} \Leftrightarrow \alpha_p^k g_k(x) = \alpha_q^k g_k(x) e^{c_k} \Leftrightarrow \alpha_p^k = \alpha_q^k e^{c_k}$, where $\{c_1, \cdots, c_K\}$ is a set of constants. Since $\sum_k \alpha_p^k = \sum_k \alpha_q^k e^{c_k} = 1$ and $\sum_k \alpha_q^k = 1$, we then have the constrain $\sum_k e^{c_k} = 1$.

B.1.4 Kernelized Stein Discrepancy Extensions

For two a.c. distributions p and q, the Kernelized Stein Discrepancy Liu et al. [2016], Chwialkowski et al. [2016] can be defined as (see [Liu et al., 2016, Definition 3.2])

$$\mathrm{KSD}(p||q) = \mathbb{E}_{x,x' \sim p}\left[(s_p(x) - s_q(x))k(x,x')(s_p(x') - s_q(x'))\right], \tag{B.6}$$

where k is an integrally strictly positive kernel (see [Liu et al., 2016, Definition 3.1]) and x, x' are i.i.d. samples from $p(x)$. The $\mathrm{KSD}(p||q) = 0$ if and only if $s_p = s_q$ (see Liu et al. [2016], Chwialkowski et al. [2016]). Therefore, when p and q are supported on a connected open set, by Lemma 6, we have $\mathrm{KSD}(p||q) = 0 \Leftrightarrow s_p = s_q \Leftrightarrow p = q$. When p and q are supported on a disconnected space, we have $\mathrm{KSD}(p||q) = 0 \not\Rightarrow p = q$. This is because the KSD can be upper bounded by a (positively) scaled FD [Liu et al., 2016, Theorem 5.1]:

$$|\mathrm{KSD}(p||q)| \leq \sqrt{\mathbb{E}_{x,x' \sim p}[k(x,x')^2]} \times \mathrm{FD}(p||q), \tag{B.7}$$

we then have $\mathrm{FD}(p||q) = 0 \Rightarrow \mathrm{KSD}(p||q) = 0$. When p and q are supported on a disconnected space, we have $\mathrm{FD}(p||q) = 0 \not\Rightarrow p = q$ (Theorem 3), so $\mathrm{KSD}(p||q) = 0 \not\Rightarrow p = q$.

B.1.5 Proof of Theorem 4

Since the support of m as $\mathscr{X}_m = \mathbb{R}^d$ then $\tilde{p}$ and $\tilde{q}$ have the same support $\mathscr{X} = \mathbb{R}^d$. For the score functions, we also have

$$\int_{\mathscr{X}} ||s_{\tilde{p}}(x)||_2^2 \tilde{p}(x)\,\mathrm{d}x = \int_{\mathscr{X}} \|\nabla_x \log(\beta p(x) + (1-\beta)m(x))\|_2^2 \tilde{p}(x)\,\mathrm{d}x \tag{B.8}$$

$$= \int_{\mathscr{X}} \left\| \frac{\beta \nabla_x p(x) + (1-\beta)\nabla_x m(x)}{\beta p(x) + (1-\beta)m(x)} \right\|_2^2 \tilde{p}(x)\,\mathrm{d}x \tag{B.9}$$

$$\leq \int_{\mathscr{X}} \left\| \frac{\beta \nabla_x p(x)}{\beta p(x) + (1-\beta)m(x)} \right\|_2^2 \tilde{p}(x)\,\mathrm{d}x + \int_{\mathscr{X}} \left\| \frac{(1-\beta)\nabla_x m(x)}{\beta p(x) + (1-\beta)m(x)} \right\|_2^2 \tilde{p}(x)\,\mathrm{d}x \tag{B.10}$$

$$\leq \int_{\mathscr{X}} ||s_p||_2^2 \tilde{p}(x)\,\mathrm{d}x + \int_{\mathscr{X}} ||s_m||_2^2 \tilde{p}(x)\,\mathrm{d}x \leq \int_{\mathscr{X}} ||s_p||_2^2 p(x)\,\mathrm{d}x + \int_{\mathscr{X}} ||s_m||_2^2 p(x)\,\mathrm{d}x < \infty, \tag{B.11}$$

so $s_{\tilde{p}} \in L^2(\tilde{p})$ and similarly $s_{\tilde{q}} \in L^2(\tilde{p})$. Therefore, the FD between $\tilde{p}$ and $\tilde{q}$ is a valid divergence i.e. $\mathrm{FD}(\tilde{p}||\tilde{q}) = 0 \Leftrightarrow \tilde{p} = \tilde{q} \Leftrightarrow \beta p(x) + (1-\beta)m(x) = \beta q(x) + (1-\beta)m(x) \Leftrightarrow p(x) = q(x)$, thus $\mathrm{MFD}(p||q) = 0 \Leftrightarrow \mathrm{FD}(\tilde{p}||\tilde{q}) = 0 \Leftrightarrow p = q$.

B.2 Experiment Details

For both experiments, we sample 100k data from p_d as our training datasets. The energy network $f_\theta(x)$ is a 3-layer feedforward network with 200 hidden units and swish activation functions Ramachandran et al. [2017]. We train the model for 30k iterations with the Adam optimizer Kingma and Ba [2014] and batch-size 300. For the numerical integration we use Simpson's rule provided in the package Virtanen et al. [2020]. We use a Monte-Carlo approximation to estimate the KL divergence evaluations $\widehat{\mathrm{KL}}(p_d(x)||q_\theta(x)) = \frac{1}{K}\sum_{k=1}^{K} \log p_d(x_k) - \log p_\theta(x_k)$, where we use $K = 10000$.

Appendix C

Appendix of Chapter 5

C.1 Model-based Lossless Compression

Lossless compression aims to create an invertible mapping from real-world data (e.g. image, audio, video) to binary strings with the lengths of the strings as short as possible. We then briefly introduce how to design an optimal lossless compressor in practice.

C.1.1 Information Theory of Lossless Compression

Let X be a discrete random variable that taking values from a finite countable set $\mathscr{X}$ and has a probability mass function (PMF) $p : \mathscr{X} \to \mathbb{R}$ such that $\forall x \in \mathscr{X}$, $p(x) > 0$ and $\sum_x p(x) = 1$.

Definition 1. The *Shannon information content* of a sample $x \sim p(x)$ is defined as

$$h_p(x) \equiv -\log_2 p(x). \tag{C.1}$$

Definition 2. The *Shannon Entropy* of a distribution p is defined as

$$H(p) \equiv -\sum_x p(x) \log_2 p(x). \tag{C.2}$$

We then give the informal statement of the *Shannon Source Coding Theorem* [Shannon, 2001], the detailed statement and the proof can be found in Chapter 4 of [MacKay, 2003].

Theorem 7 (Shannon's Source Coding Theorem (informal))**.** *Given N i.i.d samples form the data generation distribution with PMF $p_d(x)$ can be losslessly compressed into more than $NH(p_d)$ bits when $N \to \infty$. Conversely, they cannot be losslessly compressed into fewer than $NH(p_d)$ bits.*

To obtain a 'near-optimal' lossless compression scheme in practice, one strategy is to compress each data $x \sim p_d(x)$ into a binary string with length equal to $h_{p_d}(x) + \varepsilon$, where $h(x)$ is the *Shannon information content* and ε represents a small coding overhead. Therefore, given N *i.i.d* samples $\{x_1, \cdots, x_N\} \sim p_d(x)$, the averaged compression length is

$$-\frac{1}{N}\sum_{n=1}^{N} \log_2 p_d(x_n) + \varepsilon \xrightarrow{N\to+\infty} -\sum_x p_d(x)\log_2 p_d(x) + \varepsilon = H(p_d) + \varepsilon, \quad \text{(C.3)}$$

which is close to optimal when ε is small.

Different coders are proposed to make the overhead ε for different types of data. For multi-dimensional data, there exits two methods that can provide us 'near-optimal' lossless compression: Arithmetic Coding (AC) [Witten et al., 1987] and Asymmetric Numeral System (ANS) [Duda, 2013], we recommend Chapter 6 of [MacKay, 2003] and [Townsend, 2020] for detailed introductions of the two methods respectively. Weuse the ANS coder in this paper since it has a faster speed comparing to AC. For simplicity, we abstract an ANS coder as an invertible function $\text{enc}_p(\cdot)$ that maps a given data $x' \in \mathscr{X}$ to a binary string message m' with length $\text{len}(m') = -\log_2 p(x') + \varepsilon$, where ε is a negligible coding overhead. We also denote the decoding function as $\text{dec}_p(\cdot) = \text{enc}_p^{-1}(\cdot)$ and have $\text{dec}_p(m') \to x'$.

We have introduced how to optimally compress the data when we know the true data generation distribution $p_d(x)$. However, the distribution $p_d(x)$ is usually unknown in practice, we would like to learn a model $p_\theta(x)$ to approximate the underlying data distribution $p(x)$ and then use the learned model $p_\theta(x)$ to conduct lossless compression. In this case, the *averaged* data compression length for

$\{x_1, \cdots, x_N\} \sim p_d(x)$ is (ignoring the coding overhead ε):

$$-\frac{1}{N}\sum_{n=1}^{N}\log_2 p_\theta(x_n) \xrightarrow{N\to+\infty} -\sum_x p(x)\log_2 p_\theta(x_n). \tag{C.4}$$

The difference between the model compression length and the optimal compression length is

$$-\frac{1}{N}\sum_{n=1}^{N}\Big(\log_2 p_\theta(x_n) - \log_2 p_d(x_n)\Big) \xrightarrow{N\to+\infty} \mathrm{KL}(p_d(x)||p_\theta(x))\,. \tag{C.5}$$

C.2 Tightness of the ELBO and IWAE Improvement

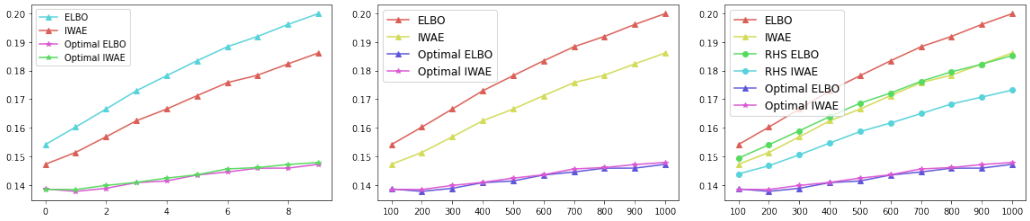
In this section, we want to verify the tightness of the ELBO as a lower bound of the log-likelihood. Consider the likelihood for a single data point x', we have

$$\log p_\theta(x') \geq \langle \log p_\theta(x'|z)\rangle_{q_\phi(z|x')} - \mathrm{KL}(q_\phi(z|x')||p(z)) \equiv \mathrm{ELBO}(x',\theta,\phi). \tag{C.6}$$

To evaluate $\log p_\theta(x')$, we can use an importance weighted estimation (IWAE Burda et al. [2015]), which can be rewritten as

$$\log p_\theta(x') = \log\Big\langle \frac{p_\theta(x'|z)p(z)}{q_\phi(z|x)}\Big\rangle_{q_\phi(z|x)} \approx \log\frac{1}{K}\sum_{k=1}^{K}\frac{p_\theta(x'|z_k)p(z_k)}{q_\phi(z_k|x')} \equiv \mathrm{IWAE}_k(x',\theta,\phi), \tag{C.7}$$

where $z_k \sim q_\phi(z|x')$. The accuracy of the importance sampling heavily depends on the proposal distribution $q_\phi(z|x')$ and will be poor if $q_\phi(z|x')$ underestimates the high density region of $p_\theta(z|x)$ Burda et al. [2015]. For the ELBO with optimal inference, we can assume the approximate posterior is close to the true posterior, so if the lower bound is tight, we will observe that the ELBO is approximately equal to the IWAE. In Figure C.1 we compare the ELBO and IWAE using classic amortized inference and optimal inference respectively (we use $k = 10$ in all cases). We find that the IWAE can improve the ELBO for the traditional amortized inference and is approximately equivalent to the ELBO using the optimal inference strategy. Therefore, we can conclude that the ELBO with the optimal inference strategy is tight to $\log p_\theta(x)$.



(a) Tightness (b) IWAE Improvement (c) All Comparisons

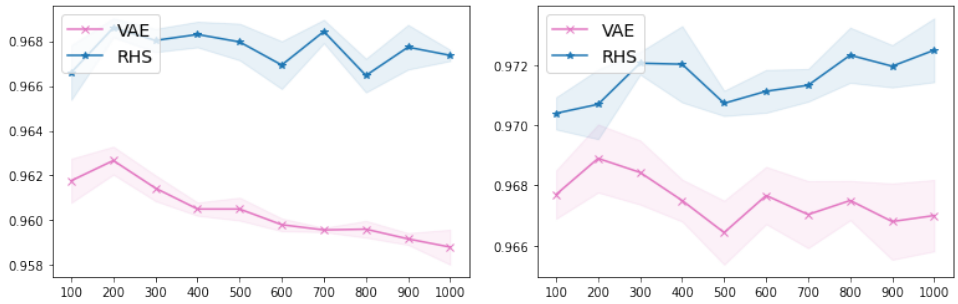
Figure C.1: IWAE comparisons on Binary MNIST. The x-axis indicates the training epoch and the y-axis is the Bits-per-dimension, which corresponds to the negative ELBO or IWAE with log 2 base and normalized by data dimension, lower is better. In Figure a, we see that IWAE improves the ELBO when using classic amortized inference but is approximately equal to the ELBO when using optimal inference, which indicates the bound is tight. In Figure b, we compare the IWAE with classic amortized inference, optimal inference and the the proposed reverse half-asleep (RHS) inference. Here we find the proposed method can also improve the classic IWAE estimation without training on the test data. In Figure 3, we plot the ELBO and IWAE for all three amortized inference methods.

We also estimate the IWAE using the proposal posterior learned by the proposed reverse half-asleep inference and find that our method can also improve the IWAE result, see Figure C.1 for details. This is intuitive since our method can provide a better proposal distribution for importance sampling.

C.3 Amortized Posterior for Classification

In Section 4, we discussed that the proposed reverse half-asleep method can improve the posterior prediction for the test data. One direct application is to use the learned amortized posterior $q_\phi(z|x)$ for down-stream tasks, e.g. image classification, where the samples $z' \sim q_\phi(z|x')$ can be treated as the 'stochastic representation' Zhang et al. [2022], Bengio et al. [2013] of the given data point x'. Given a labeled dataset $\{(x_1, y_1), \cdots, (x_N, y_N)\}$ and a trained amortized posterior (encoder) $q_\phi(z|x)$, we can then train a classifier $p_\eta(y|z)$ that maps from the latent space z to the label y. After training the classifier, for a given test set of unlabelled data $\{x'_1, \cdots, x'_M\}$, the predictive distribution can be written as $p(y|x) = \int p_\eta(y|z) q_\phi(z|x) dz$ and can be approximated by Monte-Carlo: $p(y|x) \approx \frac{1}{K} \sum_{k=1}^{K} p(y|z'_k)$, where $z'_k \sim q_\phi(z|x)$. We train a classifier with 2 layer feed-forward neural network with hidden size 200, RelU activation and dropout with rate 0.1 on two datasets: binary MNIST and grey

MNIST. The models are trained for 10 epochs with Adam optimizer and learning rate 3×10^{-4}. During training, we randomly sample one z' for each data point x and we use $k = 100$ in the testing stage to estimate the predictive distribution. Figure C.2 shows the comparisons between the posterior trained by the classic amortized inference and the proposed reverse half-asleep method respectively. We can see our method consistently improves the classification accuracy performance.

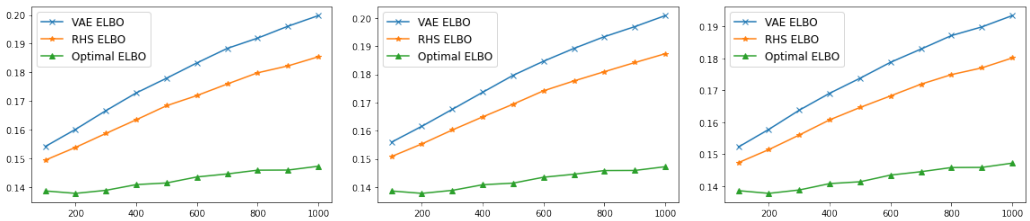


(a) Binary MNIST (b) Grey MNIST

Figure C.2: Representation Learning for Down-Stream Classification. We train the VAE for 1000 epochs and evaluate the classification accuracy (y-axis, higher is better) on the down-stream classification task every 100 epochs (x-axis). The results are averaged over 3 random seeds and we also plot the standard deviation.

C.4 Effects of the Latent Space Dimensionality

We study the effect of the latent dimension size on the generalization of the amortized inference. We use the VAE described in Section 4 with different latent size $[16, 64, 128]$ on Binary MNIST, see Figure C.3 for the result. We find the overfitting of amortized inference happens in all cases regardless of the latent size. We also apply the proposed reverse half-asleep training method to the saved model every 100 epoch and found our method can consistently improve the generalization performance.



(a) Dim$(z) = 16$ (b) Dim$(z) = 64$ (c) Dim$(z) = 128$

Figure C.3: Effects of different latent dimension. The y-axis is the BPD and x-axis is the training epochs. We find the amortized inference generalization gap exits in all cases.

Appendix D

Appendix of Chapter 6

D.1 Experiments Details

D.1.1 Compute resources

All models were trained using an NVDIA GeForce RTX 2080 Ti and an NVDIA GeForce V100 GPU.

D.1.2 Prepossessing of CelebA

We first perform a central crop with edge length 150px and then resize to $32\times32\times3$. We select the first 10000 images as our CelebA test set.

D.1.3 Model Architecture

This section provides additional details concerning the model building blocks used in our experiments. All models share a similar PixelCNN Van Oord et al. [2016] architecture, which contains a masked CNN as the first layer and multiple Residual blocks as subsequent layers, we next provide further details on both components.

Masked CNN structure is proposed in Van Oord et al. [2016]. For our local model with dependency horizon h, one kernel of the CNN has size $k \times k$ with $k = 2 \times h + 1$. The masked CNN contains masks to zero out the input of the future pixels. There are two types of Masked CNN, which we refer to as mask A (zero out the current pixel) and mask B (allow connections from a color to itself), see Van Oord et al. [2016] for further details. The first layer of our model utilizes mask A and the residual blocks use mask B.

Residual Block Each residual Block Van Oord et al. [2016] contains the following structure. We use MaskedCNN$_B$ to denote a Masked CNN using mask B.

Algorithm 5 Residual Block

Input: x_{input}
$h = \text{MaskedCNN}_B(x_{input})$
$h = \text{ReLU}(h)$
$h = \text{MaskedCNN}_B(h)$
$h = \text{ReLU}(h)$
$h = \text{MaskedCNN}_B(h)$
$h = \text{ReLU}(h)$
Return : $x_{input} + h$

Pixel CNN Our full Pixel CNN and local pixel CNN shares the same backbone. The difference between the two models is that the full model has kernel size 3×3 for the second masked CNN layer in the Residual Block and, in contrast, the local model uses a kernel size of 1×1 for each of the masked CNN layers, in the Residual block. Crucially, this difference results in the receptive field of the full model increasing when stacking multiple Residual blocks whereas the receptive field of the local model does not increase. A Pixel CNN with N residual blocks has the following structure:

Algorithm 6 Pixel CNN

Input: x_{input}
$h = \text{MaskedCNN}_A(x_{input})$
$h = \text{ReLU}(h)$
for i from 1 to N:
$h = \text{ResBlock}_i(h)$
$h = \text{MaskedCNN}_B(h)$
$h = \text{ReLU}(h)$
$h = \text{MaskedCNN}_B(h)$
Return : h

D.1.3.1 OOD detection

Gray images For gray images, our full Pixel CNN model has five residual blocks and channel size 256, except the final layer which has 30 channels. The kernel size is 7 for the first Pixel CNN and the kernel size is $[1\times1, 3\times3, 1\times1]$ for three masked

CNNs in the residual blocks. Our local Pixel CNN model has one residual block and channel size 256, except the final layer which has 30 channels. The kernel size is 7 for the first Pixel CNN and the kernel size is $[1\times1, 1\times1, 1\times1]$ for three masked CNNs in the residual blocks. We use the discretized mixture of logistic distribution Salimans et al. [2017] with 10 mixture components. The models are trained using the Adam optimizer Kingma and Ba [2014] with learning rate 3×10^{-4} and batch size 100 for 100 epochs.

Color images For color images, our full Pixel CNN model has 10 residual blocks and channels size 256, except the final layer which has 100 channels. The kernel size is 9 for the first masked CNN and the kernel size is $[1\times1, 3\times3, 1\times1]$ for three masked CNNs in the residual blocks. Our local Pixel CNN model has 10 residual blocks and channels size 256, except for the final layer which has 100 channels. The kernel size is 7 for the first Pixel CNN and the kernel size is $[1\times1, 1\times1, 1\times1]$ for three masked CNNs in the residual blocks. We use the discretized mixture of logistic distribution Salimans et al. [2017] with 10 mixture components. The models are trained using the Adam optimizer Kingma and Ba [2014] with learning rate 3×10^{-4} and batch size 100 for 1000 epochs.

D.1.3.2 Lossless compression

The NeLLoC is based on a Pixel CNN. Since it is a local model, the kernel size is 1×1 for all the kernels in the Residual block. Additional details regarding the first layer kernel size and number of residual blocks are found in the main text, Section 4. All models are trained using the Adam optimizer Kingma and Ba [2014] with a learning rate of 3×10^{-4} and batch size 100 for both the CIFAR dataset (1000 epochs) and the ImageNet32 dataset (400 epochs).

D.2 Local Model

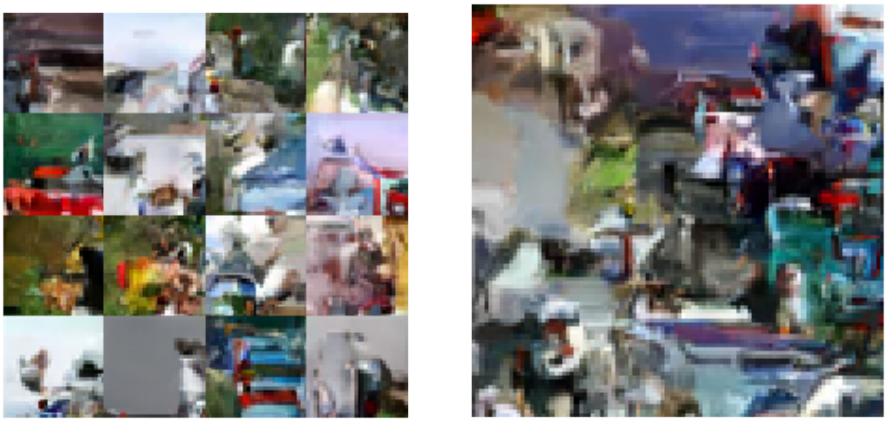
D.2.1 Effect of Horizon Size for Color Images

In Section 5.2.2 of the main manuscript we show that, for a simple gray-scale dataset, the model can overfit to non-local features that are specific to the training distribution and thus degrade OOD generalization performance. For a more complex training

distribution, *e.g.* CIFAR, we find that a model with limited capacity is less susceptible to overfit to non-local features. However, as observable in Table 5.4, when the local horizon size increases, the ID generalization continues to improve, whereas the OOD generalization remains stable. This is consistent with our hypothesis: local features are not shared between distributions and cannot significantly aid OOD generalization. We conjecture that, with the use of a more flexible base model *e.g.* PixelCNN++ Salimans et al. [2017], over-fitting to the non-local features will occur and thus result in familiar degradation of OOD generalization abilities.

D.2.2 Samples from Local Model

We show samples from a local model with h=3, trained on CIFAR 32×32×3, in Figure D.1. It can be observed in Figure D.1a that the samples are locally consistent yet images do not possess much in the way of recognizable and meaningful global semantics. Figure D.1b shows an example image of size 100×100×3. This is made possible since the local model does not require sampled images to have a fixed size, *i.e.* size is not required to be consistent with the training data.



(a) 16 samples with size 32×32×3 (b) One sample with size 100×100×3

Figure D.1: Samples from a local autoregressive model.