

Approximate Inference in Variational Autoencoders

Chris Cremer



Department of Computer Science
University of Toronto

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy

© Copyright 2020 by Chris Cremer

Abstract

Approximate Inference in Variational Autoencoders

Chris Cremer

Doctor of Philosophy

Graduate Department of Computer Science

University of Toronto

2020

A deep latent variable model is a powerful tool for modelling complex distributions. However, in order to train this model, we must perform approximate inference of the latent variable. A variational autoencoder (VAE) is a framework for learning both the generative and inference models for a latent variable model. This thesis provides novel analyses, applications, and interpretations of approximate inference in VAEs.

This thesis reviews many of the recent developments made to improve VAEs. One such improvement is the importance-weighted autoencoder. The standard interpretation of importance-weighted autoencoders is that they maximize a tighter lower bound on the marginal likelihood than the standard evidence lower bound. The first contribution of this thesis is to provide an alternative interpretation: that it optimizes the standard variational lower bound, but using a stochastic importance-weighted variational distribution.

In order to improve approximate inference in VAEs, it is important to understand what causes inference to be suboptimal. Two important factors that determine the quality of approximate inference in VAEs are: a) whether the variational distribution is expressive enough to match the true posterior and b) the ability of the recognition network to produce good variational parameters for each datapoint. This thesis contributes to our understanding of approximate inference in VAEs by using these factors to diagnose suboptimal inference across a number of settings.

Finally, we investigate the task of using a latent variable to learn a joint distribution over two data modalities: images and text. The contribution of this work is to develop a model for this task and analyze how to perform approximate inference in this model. Specifically, we identify a problem arising from the mismatch between the posteriors of each modality and we demonstrate how the problem can be largely addressed by modelling the aggregate of the image posteriors.

Acknowledgements

Firstly, I'd like to express immense gratitude towards my supervisors, Quaid Morris and David Duvenaud. Quaid has advised me for over 5 years and he's been a constant source of guidance, patience, and encouragement. David has been a great mentor and I've tremendously enjoyed and benefited from our lengthy discussions regarding new research ideas. I'm grateful to both of my supervisors for the support they've given me over the years.

To my collaborators, Xuechen Li and Nate Kushman, working with you was a joy. I'd like to thank my supervisory committee members, Radford Neal and Roger Grosse, as well as my final defence committee members, Rich Turner and David Fleet, for their time and valuable feedback.

Finally, I'd like to thank my family and friends for their unending encouragement and support over the years.

Contents

1	Introduction	1
1.1	Before the VAE	2
1.2	Motivation	3
1.3	Outline and Contributions	4
2	Background	6
2.1	Likelihood-Based Generative Models	6
2.1.1	Variational Autoencoders	7
2.1.2	Normalizing Flow Models	11
2.1.3	Auto-Regressive Models	15
2.2	Beyond Standard VAEs	16
2.2.1	Importance Weighted Autoencoders	17
2.2.2	Flow Posteriors	18
2.2.3	Auxiliary Variables	19
2.2.4	Hamiltonian Variational Inference	20
2.2.5	Hierarchical Latent Variable	21
2.2.6	More Expressive Priors	22
2.2.7	More Expressive Decoders	23
2.2.8	Summary of the Lower Bounds	24
3	Reinterpreting Importance-Weighted Autoencoders	26
3.1	Background	26
3.2	Defining the implicit distribution \tilde{q}_{IW}	28

3.2.1	\tilde{q}_{IW} approaches the true posterior	28
3.2.2	Recovering the IWAE bound from the VAE bound	30
3.2.3	Expected importance weighted distribution q_{EW}	30
3.2.4	Proof that q_{EW} is closer to the true posterior than q	31
3.3	Visualizing the nonparameteric approximate posterior	32
3.4	Resampling for prediction	32
3.5	Discussion	33
3.6	Detailed Derivations	35
3.6.1	Detailed derivation of the equivalence of VAE and IWAE bound . . .	35
3.6.2	Proof that q_{EW} is a normalized distribution	36
4	Inference Suboptimality in Variational Autoencoders	37
4.1	Introduction	38
4.2	Background	39
4.2.1	Inference in Variational Autoencoders	39
4.3	Methods	40
4.3.1	Approximation Gap and Amortization Gap	40
4.3.2	Flexible Approximate Posteriors	41
4.3.3	Marginal Log-Likelihood Estimation and Evidence Lower Bounds . .	42
4.3.4	Local Optimization of the Approximate Distribution	43
4.3.5	Validation of Bounds	43
4.4	Related Work	44
4.5	Experimental Results	44
4.5.1	Intuition through Visualization	44
4.5.2	Amortization vs Approximation Gap	46
4.5.3	Influence of Flows on the Amortization Gap	48
4.5.4	Influence of Approximate Posterior on True Posterior	49
4.5.5	Generalization of Amortized Inference on Held-Out Data	50
4.5.6	Annealing the Entropy	52
4.6	Conclusion	53

5	Aggregate Posteriors and the Vision-Language VAE	54
5.1	Introduction	55
5.2	Vision-Language VAE	58
5.2.1	Inference with Only Images	58
5.2.2	Conditional Generation	62
5.3	Learning the Aggregate Posterior	64
5.4	Related Work	65
5.5	Results	67
5.5.1	Comparison of $q_{Agg}(z y)$ and $q_{True}(z y)$	69
5.5.2	Comparison of VLVAE and JMVAE	70
5.6	Conclusion	71
6	Discussion	72
6.1	Summary of Contributions	72
6.2	Comparison of Likelihood-Based Models	73
6.3	Future Directions	75
6.4	Conclusion	77
	Bibliography	78
A	Supplementary Material	89
A.1	Chapter 3 Supplementary Material	89
A.1.1	Proof that $\mathcal{L}_{VAE}[q_{EW}]$ is an upper bound of $\mathcal{L}_{IWAE}[q]$	89
A.2	Chapter 4 Supplementary Material	91
A.2.1	Generalization of Amortized Inference on Held-Out Data	93
A.2.2	Influence of Flows On Amortization Gap Experiment	94
A.2.3	Computation of the Determinant for Flow	94
A.2.4	Annealed Importance Sampling	95
A.2.5	Extra MNIST Inference Gaps	96
A.3	Chapter 5 Supplementary Material	97
A.3.1	Model Architecture	97

A.3.2	Dataset Details	99
A.3.3	Training Details	99
A.3.4	2D Visualization Details	100
A.3.5	Classifier Details	100
A.3.6	Aggregate Posterior Prior Decomposition	102
A.3.7	Nearest Training Images	103

List of Tables

2.1	Summary of Lower Bounds	24
4.1	Summary of Gap Terms	39
4.2	Inference Gaps	46
4.3	Larger encoder results	47
4.4	Influence of expressive approximations	48
4.5	Increased decoder capacity gaps	50
4.6	Gaps of models trained without entropy annealing	52
5.1	Approximations of joint VAE models	66
5.2	Comparison of the aggregate of the image posteriors	69
5.3	Comparison of JMVAE and VLVAE objectives	71

List of Figures

2.1	VAE Graphical Model	8
2.2	Diagram of two coupling layer transformations	13
2.3	Demonstration of a 2D affine coupling layer transformations	14
2.4	Graphical model of a VAE with an auxiliary variable	19
2.5	Hierarchical latent variable models	21
2.6	Decoder Graphical Models	23
3.1	Visualization of 1D \tilde{q}_{IW} and q_{EW} distributions.	29
3.2	2D q_{EW} Approximations	32
3.3	Reconstructions of MNIST samples from $q(z x)$ and q_{EW}	33
4.1	Inference Gaps Breakdown	38
4.2	True Posterior and Approximate Distributions of a VAE with 2D latent space	45
4.3	Inference gaps over epochs	51
5.1	Graphical model of a Vision-Language VAE	55
5.2	2D Visualization of the distribution mismatch	56
5.3	Example datapoints from Two Object CLEVR and CelebA	57
5.4	Inference with both modalities vs inference with images only	59
5.5	Decoding distribution of text from a VLVAE	60
5.6	Sample from a VLVAE model on CelebA	62
5.7	Conditional samples from approximation of true posterior.	63
5.8	Conditional samples from $q_{Agg}(z y)$	68
5.9	Comparison of $q_{Agg}(z y)$ and $q_{True}(z y)$ over training	70

6.1	Venn diagram of different combinations of the likelihood-based models	74
A.1	Gaps over epochs using the AF model	93
A.2	Training curves for a FFG and a Flow inference model on MNIST	96
A.3	Nearest training images	103

Chapter 1

Introduction

Modelling a distribution over complex, high dimensional datasets, such as images and text, remains a challenging and important problem in machine learning. Substantial progress has been made on this problem through the development of expressive deep generative models. Generative models are models that learn a distribution over data and are capable of generating samples from their distribution. Generative models are capable of addressing many tasks, such as outlier detection, inferring the values of missing data, and quantifying the uncertainty of a model's prediction.

The class of generative models that this thesis will focus on are those that maximize the probability of the data under the model, referred to as likelihood-based generative models. Among these models are latent variable models, where we model the data by assuming the data are generated from a hidden (latent) variable. This modelling choice allows us to model complex distributions but it also requires that we integrate over the value of the latent variable.

Variational inference is a method used to train deep latent variable models by performing approximate inference using a variational distribution. Due to the inference being approximate, the model does not optimize the exact likelihood; instead it optimizes a lower bound of the data log-likelihood. Inference can be amortized by training a neural network to output the variational distribution for a given datapoint. This neural network is often referred to as the inference network, recognition network, and/or the encoder network.

In order to learn the encoder network efficiently, we need to backpropagate gradients

through the stochastic latent variable layer. When using some specific continuous variational distributions, such as the Gaussian distribution, we can use the reparameterization trick (Williams, 1992; Kingma and Welling, 2014; Rezende et al., 2014) to obtain low variance gradients of the variational lower bound with respect to the encoder parameters. This approach to learning and inference in deep latent variable models, which uses an encoder network and the reparameterization trick, is called a Variational Autoencoder (VAE, Kingma and Welling (2014); Rezende et al. (2014)).

1.1 Before the VAE

One of the key components that distinguishes the VAE approach from other approaches to learning latent variable models is the use of the reparameterization trick. Prior to the VAE, the REINFORCE (Williams, 1992) gradient estimator was a technique to obtain unbiased gradients of the lower bound objective. However, when the REINFORCE estimator is used to train large models, the variance of this estimator is often too high to learn efficiently. The use of control variates for variational inference (Blei et al., 2012; Ranganath et al., 2013) is a technique that can be used to help reduce the variance of gradient estimators. The work of Gregor et al. (2014) is an example of a binary latent variable model with an autoencoder structure being trained with REINFORCE and control variates.

The wake-sleep algorithm (Dayan et al., 1995; Hinton et al., 1995) was another technique for training deep directed graphical models as well as the recognition model. In the wake phase, the generative model is updated using latent variable samples from the recognition model to optimize the variational lower bound. In the sleep phase, a ‘dream’ sample is generated from the model by sampling the prior and then the decoder, and the inference network is trained to infer the latent variable for that generated datapoint. The sleep phase minimizes the KL divergence of the true and approximate posterior under the generative model data distribution, whereas the wake phase optimizes the bound under the true data distribution. Thus one downside of the wake-sleep algorithm is that the sleep phase gradients are biased. However, one advantage of the algorithm is that it is applicable to discrete variables, unlike the reparameterization trick. Furthermore, the wake-sleep algorithm was

improved with the reweighted wake-sleep algorithm (Bornschein and Bengio, 2015), which is still a competitive technique today for discrete variable models (Le et al., 2019).

Another related method to VAEs is the denoising autoencoder (DAE) (Vincent et al., 2010; Bengio et al., 2013), which corrupts the input with noise and trains the encoder-decoder to reduce the reconstruction error. The model can then be sampled using Langevin or Metropolis-Hastings MCMC. Unlike the DAE, the VAE objective is directly derived using the variational approach and sampling the VAE is easy via ancestral sampling.

1.2 Motivation

While standard VAEs can model quite complex distributions, there are still many directions for improvement. One avenue for improvement of VAEs is through the tightening of their lower bound objective. This can be achieved through the use of importance sampling. A question that arises when using an importance weighted objective is how does it affect the model’s implicit approximate posterior? Another question is how do we visualize and sample this implicit distribution? These are questions that this thesis will address.

Recently, there have been many works proposing VAE modifications with the goal of improving approximate inference. However, even with these improvements, amortized inference in VAEs continues to be suboptimal. This raises many important questions, such as: What are the main factors causing inference to be suboptimal? Is the distribution that we’re trying to model too complex? Is it too difficult for a single neural network to infer the latent variable associated with each observation? Answering these questions is another aim of this thesis.

Typically, latent variables are used in settings where each data dimension belongs to a single modality (data type), e.g. pixels of an image. What if we’d like to use a latent variable to model data from two data modalities? For example, how would inference be different if we modelled images and their captions with a single latent variable? How does the choice of approximate inference affect the model? How should we perform inference if we want to infer one modality given the other? This thesis will also explore these questions.

As can be seen, there are many areas to explore relating to approximate inference in VAEs.

The purpose of this thesis is to answer these questions in order to gain a better understanding of approximate inference in VAEs.

1.3 Outline and Contributions

Chapter 2 begins by reviewing three powerful likelihood-based generative models: latent variable models, auto-regressive models, and flow models. These model types are often used in conjunction to improve the generative model. Chapter 2 then goes into more detail about the VAE and describes many of the extensions that have been made to the standard VAE model.

In Chapter 3, we investigate the importance-weighted autoencoder (IWAE) and explore how its inference differs from the standard VAE. The standard interpretation of IWAE is that it maximizes a tighter lower bound on the marginal likelihood than the standard evidence lower bound. We give an alternate interpretation of this procedure: that it optimizes the standard variational lower bound, but using a stochastic importance-weighted variational distribution. The contribution of this section is to derive this result, present a tighter lower bound, visualize the implicit importance-weighted distribution and provide the algorithms for plotting and sampling from this distribution.

In Chapter 4, we aim to understand what makes approximate inference in variational autoencoders suboptimal. Two important factors that determine the quality of approximate inference in VAEs are: a) whether the variational distribution is expressive enough to match the true posterior and b) the ability of the recognition network to produce good variational parameters for each datapoint. Thus in this chapter, we evaluate the divergence between the approximate posterior and true posterior in terms of the divergence caused by the limited expressiveness of the variational distribution and the divergence caused by the limited capacity of the recognition network. By measuring these divergences we make a number of observations. We find that the divergence from the true posterior is often due to imperfect recognition networks, rather than the limited complexity of the approximating distribution. We also show that this is due partly to the generator learning to accommodate the choice of approximation. Furthermore, we show that the parameters used to increase the expressiveness

of the approximation play a role in generalizing inference rather than simply improving the complexity of the approximation. These observations demonstrate that our analysis is a useful tool for diagnosing suboptimal inference.

In Chapter 5, we study how to model the joint distribution over two data modalities using a latent variable. Specifically, we introduce the Vision-Language VAE to model the joint distribution over images and text. We begin by investigating how inferring the latent variable using only images affects the resulting model. Next, we explore a common task for these multimodal models: to perform conditional generation. For instance, generating an image conditioned on text. One approach to this task is to sample the posterior of the text then generate the image given the latent variable. However, we find that a problem with this approach is that the posterior of the text does not match the aggregate of the image posteriors corresponding to that text. The result is that the generated images are either of poor quality or don't match the text. In this chapter, we highlight the importance of learning aggregate posteriors when faced with these types of distribution mismatches.

Finally, given that there are a number of choices for likelihood-based generative models, how do we know which is best for a given problem? How can the various generative models be combined? Chapter 6 discusses some of the advantages and disadvantages of each model type and points to future directions for improvement. A thorough understanding of the limitations of these models will help guide future improvements.

Chapter 2

Background

This background section will first cover likelihood-based generative models (VAEs, auto-regressive models, flows) because these model types will be used throughout the thesis and it will be helpful to have an understanding of how VAEs fit into the wider generative modelling picture. Next, the background section will go deeper into the developments that have been made in VAEs. These developments include improvements to inference (importance weighting, flow posteriors, auxiliary variables) as well as model improvements (hierarchical latent variables, improved priors, improved decoders).

2.1 Likelihood-Based Generative Models

Likelihood-based generative models are models that learn to model a distribution by maximizing the probability the data x with respect to (wrt) the model parameters θ : $\max_{\theta} p_{\theta}(x)$. In other words, these models maximize the likelihood of the parameters given the data. To date, the most common, powerful likelihood-based generative models include latent variable models, auto-regressive models, and flow (change of variable) models.

A brief introduction to each of the models is the following: Latent variable models can be seen as an infinite mixture model over the data by integrating over the distribution of the latent variable. Auto-regressive models involve partitioning the data dimensions and learning the distribution of each dimension conditioned on the previous dimensions. Finally, flow models involve transforming a simple distribution into a complex distribution using invertible

transformations. These three model types can be summarized by the following equations, which describe the probability of the data under each model:

$$\text{Latent variable: } p(x) = \int p(x|z)p(z)dz$$

$$\text{Auto-regressive: } p(x) = p(x_0) \prod_{d=1}^D p(x_d|x_{<d})$$

$$\text{Flow: } p(x) = p_Z(f(x)) \left| \frac{\partial f(x)}{\partial x} \right|$$

where $x_{<d}$ of the auto-regressive equation represents all previous dimensions before dimension d . These three equations illustrate the core modelling assumptions mentioned above for each of the models.

This thesis is mainly concerned with latent variable models; however, as we'll see, these models are often used in conjunction. In the following sections, I will go into more depth regarding these model types. In section 6.2, I will examine the advantages and disadvantages of each of the models.

2.1.1 Variational Autoencoders

Deep latent variable models are a powerful and popular approach to modelling complex, high dimensional distributions. Consider a general probabilistic model $p_\theta(x, z)$ of data x , latent variables z , and model parameters θ . The generative process of a typical latent variable model is shown in Fig. 2.1a. Based on the graphical model, the factorization of the joint distribution $p_\theta(x, z)$ is $p_\theta(x|z)p(z)$. The plate notation used to represent the number of datapoints N in 2.1a will be omitted in future graphical models for simplicity.

Learning to maximize the data likelihood, $p_\theta(x)$, under this model involves integrating out z : $p_\theta(x) = \int_z p_\theta(x, z)dz$. Given that this integral is intractable, it is approximated via sampling from an approximation to the true posterior distribution. Through the variational inference approach, posterior inference is performed by learning a parameterized distribution $q_\phi(z|x)$ which approximates the true posterior $p(z|x)$.

The variational autoencoder (VAE, Kingma and Welling (2014); Rezende et al. (2014)) is a general framework for learning latent variable models with variational inference. Generally, VAEs use neural networks for the generative model as well as the inference model. We will

refer to the union of the generative model and inference model as the VAE model. This section will cover three important aspects of VAEs: the evidence lower bound, amortized inference and the reparameterization trick.

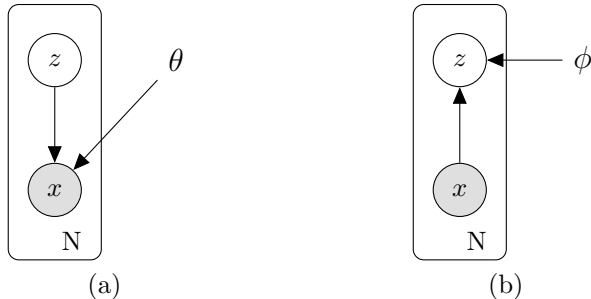


Figure 2.1: (a) Generative graphical model and (b) inference graphical model of a VAE.

Variational Inference and the Evidence Lower Bound (ELBO)

Learning in a latent variable model involves maximizing the likelihood of the parameters given the data: $p_\theta(x) = \int_z p_\theta(x, z) dz$. This integral is intractable for the model class that we consider: neural net parameterization of $p_\theta(x|z)$. Thus, the integral is approximated via Monte Carlo sampling. Variational inference learns a distribution $q(z|x)$ that approximates the true posterior $p(z|x)$. This variational distribution can be used for importance sampling to approximate the intractable integral.

Since $q(z|x)$ is only an approximation to $p(z|x)$, VAEs do not maximize the exact marginal log likelihood of the parameters. Instead, they maximize a lower bound of the marginal log likelihood, often referred to as the evidence lower bound (ELBO), which can be derived using Jensen's inequality:

$$\begin{aligned}
 \log p_\theta(x) &= \log \int p_\theta(x|z)p(z)dz \\
 &= \log \left(\mathbb{E}_{q_\phi(z|x)} \left[\frac{p_\theta(x|z)p(z)}{q_\phi(z|x)} \right] \right) \\
 &\geq \mathbb{E}_{q_\phi(z|x)} \left[\log \left(\frac{p_\theta(x|z)p(z)}{q_\phi(z|x)} \right) \right] \\
 &= \mathcal{L}_{VAE}[q].
 \end{aligned} \tag{2.1}$$

We use $\mathcal{L}_{VAE}[q]$ to refer to the ELBO of the VAE using variational distribution q . The expectation in Eqn. 2.1 cannot be computed analytically, thus it is approximated by Monte Carlo sampling.

We can also use an alternative derivation of the ELBO to highlight the KL divergence between the true and approximate posteriors:

$$\log p_\theta(x) = \mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x)] \quad (2.2)$$

$$= \mathbb{E}_{q_\phi(z|x)} \left[\log \left(\frac{q_\phi(z|x) p(z|x)}{q_\phi(z|x) p(z|x)} p_\theta(x) \right) \right] \quad (2.3)$$

$$= \mathbb{E}_{q_\phi(z|x)} \left[\log \left(\frac{p_\theta(x, z)}{q_\phi(z|x)} \right) + \log \left(\frac{q_\phi(z|x)}{p(z|x)} \right) \right] \quad (2.4)$$

$$= \mathbb{E}_{q_\phi(z|x)} \left[\log \left(\frac{p_\theta(x|z)p(z)}{q_\phi(z|x)} \right) \right] + \mathbb{E}_{q_\phi(z|x)} \left[\log \left(\frac{q_\phi(z|x)}{p(z|x)} \right) \right] \quad (2.5)$$

$$= \mathbb{E}_{q_\phi(z|x)} \left[\log \left(\frac{p_\theta(x|z)p(z)}{q_\phi(z|x)} \right) \right] + KL(q_\phi(z|x) || p(z|x)) \quad (2.6)$$

$$\geq \mathbb{E}_{q_\phi(z|x)} \left[\log \left(\frac{p_\theta(x|z)p(z)}{q_\phi(z|x)} \right) \right] \quad (2.7)$$

$$= \mathcal{L}_{VAE}[q].$$

where the inequality follows from the removal of the non-negative KL term. From Eqn. 2.6, we see that the closer the approximation $q_\phi(z|x)$ is to the true posterior $p(z|x)$, the tighter the bound will be. If $q_\phi(z|x) = p(z|x)$, then the objective is no longer a lower bound:

$$\mathbb{E}_{p(z|x)} \left[\log \left(\frac{p_\theta(x|z)p(z)}{p(z|x)} \right) \right] = \mathbb{E}_{p(z|x)} [\log (p_\theta(x))] = \log p_\theta(x).$$

To summarize, the ELBO objective is a stochastic lower bound: it is stochastic due to the finite samples that we use to approximate the expectation and it is a lower bound due to the mismatch between the approximate and true posteriors.

Amortized Inference

In order to make inference efficient, VAEs learn to perform inference using a neural network. Rather than having inference parameters ϕ_i for each datapoint x_i (Hoffman et al., 2013), VAEs utilize a network with a single set of parameters ϕ to perform inference for the whole

dataset:

$$\begin{aligned} \text{Amortized } \phi &: \mathbb{E}_{q_\phi(z|x_i)} \left[\log \left(\frac{p(x_i, z)}{q_\phi(z|x_i)} \right) \right] \\ \text{Non-amortized } \phi_i &: \mathbb{E}_{q_{\phi_i}(z|x_i)} \left[\log \left(\frac{p(x_i, z)}{q_{\phi_i}(z|x_i)} \right) \right] \end{aligned}$$

Amortized inference is reflected in the graphical model representation of the inference process shown in Fig. 2.1b. The network that outputs the parameters of q is often called an encoder, but it is also known as a recognition network or inference network. Given that the encoder is shared by all datapoints, we can say that the encoder amortizes inference over the dataset. The hope is that the encoder can also learn to generalize inference to new datapoints as well.

In a standard VAE, the approximate posterior that the encoder outputs is a factorized Gaussian. That is, the approximate posterior distribution is defined as $q_\phi(z|x) = N(z; \mu_\phi, \Sigma_\phi)$, where μ_ϕ and Σ_ϕ are functions of x and the covariance matrix Σ_ϕ is diagonal.

Reparameterization Trick

In order to learn efficiently in VAEs, we require low variance gradient estimators of the ELBO. This can be achieved by reparameterizing the sample $z \sim q(z|x)$ as: $z = z(\epsilon, \phi) = f_\phi(x, \epsilon)$, where $\epsilon \sim p(\epsilon)$ and $f_\phi(x, \epsilon)$ is a deterministic function of the noise variable ϵ . For Gaussian distributions, the transformation is: $z = \sigma\epsilon + \mu$, $\epsilon \sim N(0, 1)$, where σ is the standard deviation and μ is the mean of the distribution. With this transformation, the gradient of the lower bound from Eqn. 2.8 can be reformulated by pushing the gradient operator inside the expectation:

$$\begin{aligned} \nabla_\phi \mathbb{E}_{q_\phi(z|x)} \left[\log \left(\frac{p_\theta(x, z)}{q_\phi(z|x)} \right) \right] &= \nabla_\phi \mathbb{E}_{\epsilon \sim N(0,1)} \left[\log \left(\frac{p_\theta(x, z(\epsilon, \phi))}{q(z(\epsilon, \phi)|x)} \right) \right] \\ &= \mathbb{E}_{\epsilon \sim N(0,1)} \left[\nabla_\phi \log \left(\frac{p_\theta(x, z(\epsilon, \phi))}{q(z(\epsilon, \phi)|x)} \right) \right] \end{aligned}$$

This gradient estimation approach has been termed the reparameterization trick (Kingma and Welling, 2014) and it was also introduced in Williams (1992) and Rezende et al. (2014). Without this trick, the variance of the gradients is typically much larger. For instance, when learning over discrete latent variable distributions, the reparameterization trick is no longer

applicable and thus we must turn to the REINFORCE (Williams, 1992) gradient estimator , which is derived as follows for the ELBO:

$$\begin{aligned}
 \nabla_{\phi} \mathbb{E}_{q_{\phi}(z|x)} \left[\log \left(\frac{p(x, z)}{q_{\phi}(z|x)} \right) \right] &= \nabla_{\phi} \int q_{\phi}(z|x) \log \left(\frac{p(x, z)}{q_{\phi}(z|x)} \right) dz \\
 &= \int \nabla_{\phi} \left(q_{\phi}(z|x) \log \left(\frac{p(x, z)}{q_{\phi}(z|x)} \right) \right) dz \\
 &= \int (\nabla_{\phi} q_{\phi}(z|x)) \log \left(\frac{p(x, z)}{q_{\phi}(z|x)} \right) + q_{\phi}(z|x) \nabla_{\phi} \log \left(\frac{p(x, z)}{q_{\phi}(z|x)} \right) dz \\
 &= \int q_{\phi}(z|x) (\nabla_{\phi} \log q_{\phi}(z|x)) \log \left(\frac{p(x, z)}{q_{\phi}(z|x)} \right) + q_{\phi}(z|x) \nabla_{\phi} \log \left(\frac{p(x, z)}{q_{\phi}(z|x)} \right) dz \\
 &= \mathbb{E}_{q_{\phi}(z|x)} \left[\log \left(\frac{p(x, z)}{q_{\phi}(z|x)} \right) \nabla_{\phi} \log q_{\phi}(z|x) - \nabla_{\phi} \log q_{\phi}(z|x) \right] \\
 &= \mathbb{E}_{q_{\phi}(z|x)} \left[\left(\log \left(\frac{p(x, z)}{q_{\phi}(z|x)} \right) - 1 \right) \nabla_{\phi} \log q_{\phi}(z|x) \right] \tag{2.8}
 \end{aligned}$$

This REINFORCE gradient estimator often has high variance which results in slow model learning. Therefore, the reparameterization trick is an important component that allows for efficient learning in VAEs.

2.1.2 Normalizing Flow Models

Normalizing flows are based on the change of variable formula:

$$p(x) = p_Z(f(x)) \left| \det \frac{\partial f(x)}{\partial x} \right| \tag{2.9}$$

$$\text{where } z = f(x) \text{ and } x = f^{-1}(z)$$

In the above equation, $\left| \det \frac{\partial f(x)}{\partial x} \right|$ refers to the absolute value of the determinant of the Jacobian of the invertible transformation f and p_Z is a simple distribution such as $\mathcal{N}(0, 1)$. The idea behind normalizing flows is to transform x with an invertible function f and compute the likelihood of $f(x)$ under a simpler distribution $p_Z(f(x))$. To correct for the expansion and contraction of the space caused by the transformation, we multiply by the determinant of the Jacobian of the transformation $\left| \det \frac{\partial f(x)}{\partial x} \right|$. Since p_Z is a Normal distribution, we are normalizing the data, resulting in normalizing flows. Normalizing flows is a tool for

constructing complex distributions by transforming probability densities through a series of invertible mappings. By successively applying these transformations $f_1(f_2(\dots f_n(x)))$, we can build arbitrarily complex distributions.

Normalizing flows require that the determinant of the Jacobian be efficiently computed. This has been achieved through different approaches such as identities involving the determinant, coupling layer architectures, and other special invertible models. These approaches will be discussed in more detail in the following sections.

Determinant Identities

Some approaches to normalizing flows involve taking advantage of identities involving the determinant in order to compute the determinant of the Jacobian efficiently. For instance, in [Rezende and Mohamed \(2015\)](#), they introduce the planar flow: $f(x) = x + uh(w^T x + b)$, where $w \in \mathbb{R}^D$, $u \in \mathbb{R}^D$, and $b \in \mathbb{R}$ are free parameters and h is a smooth element-wise non-linearity. Using the matrix determinant lemma, the determinant of the Jacobian of the planar flow can be computed efficiently. This approach has been extended in [van den Berg et al. \(2018\)](#), where they use Sylvester’s determinant identity.

Coupling Layers

Another popular approach to normalizing flows is the use of coupling layers. Coupling layers partition the input in two then transform one partition given the other. More specifically, given a D dimensional input x , and a partitioning of the dimensions into $x_{1:d}$ and $x_{d+1:D}$, an affine coupling layer is given by:

$$y_{1:d} = x_{1:d} \tag{2.10}$$

$$y_{d+1:D} = x_{d+1:D} \odot \exp(s(x_{1:d})) + t(x_{1:d}), \tag{2.11}$$

where s and t are expressive functions (neural nets) and \odot is the Hadamard (element-wise) product. See Fig. 2.2 for a visual representation of this flow. Inverting this transformation is

straightforward:

$$x_{1:d} = y_{1:d} \quad (2.12)$$

$$x_{d+1:D} = \frac{y_{d+1:D} - t(y_{1:d})}{\exp(s(y_{1:d}))} \quad (2.13)$$

The Jacobian of this transformation is lower triangular with ones and $\exp(s(x_{1:d}))$ on the diagonal. The determinant of a diagonal matrix is the product of the diagonal, making the determinant of the Jacobian of coupling layers very efficient to compute. The specific type of dimension partitioning used depends on the task. For images, the partitioning often exploits the local correlation structure of images: spatial checkerboard patterns, and channel-wise masking.

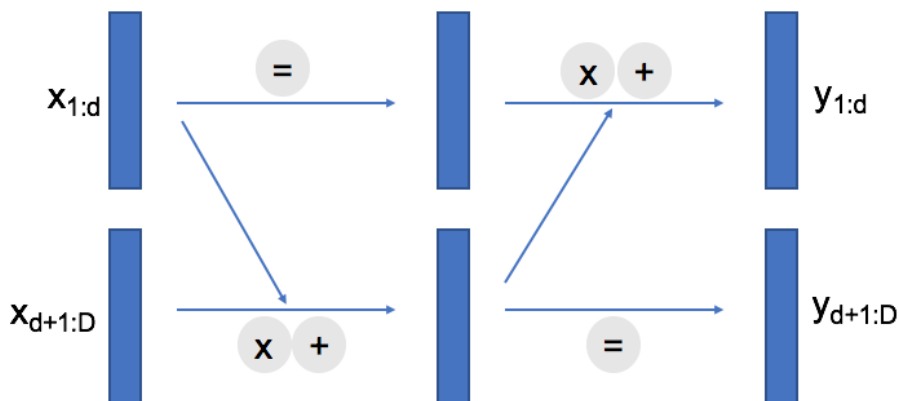


Figure 2.2: Diagram of two coupling layer transformations. The input is partitioned into two parts and one half is transformed given the other, resulting in an efficiently invertible transformation. The multiply and addition symbols represent the affine transformation of Eqn. 2.11.

One of the first uses of coupling layers in normalizing flows was in Non-linear Independent Component Estimation (NICE, [Dinh et al. \(2015\)](#)). Their coupling layers used only additive transformations (ie. no multiplication in Eqn. 2.11), which resulted in volume-preserving transformations. This work was improved upon with affine transformations (Eqn. 2.11) in real-valued non-volume preserving transformations (Real NVP, [Dinh et al. \(2017\)](#)). See Fig. 2.3 for an example of four Real NVP ([Dinh et al., 2017](#)) flows transforming a Gaussian distribution into a target distribution. This figure illustrates how flexible these simple transformations can be.

In Real NVP (Dinh et al., 2017), they propose a flow that, prior to each flow step, reverses the ordering of the channels. This was improved upon in Glow (Kingma and Dhariwal, 2018) where they replace this fixed permutation with a learned invertible 1×1 convolution. Another direction of improvement is to replace the affine transformation of the coupling layers with more complex functions, such as piecewise-polynomial transformations (Müller et al., 2019; Durkan et al., 2019a,b).

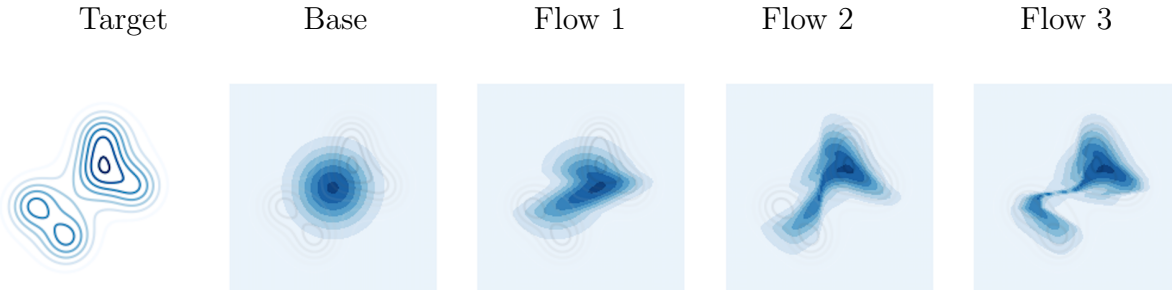


Figure 2.3: Demonstration of a 2D affine coupling layer transformations. The left distribution is the target and the following images are a sequence of flows transforming a Gaussian base distribution.

Other Invertible Models

Most flows are constrained to special architectures to assure invertibility. The approach of FFJORD (Grathwohl et al., 2019) is to specify the transformation by an ordinary differential equation (ODE). FFJORD uses a continuous version of the change of variable formula where the determinant is replaced by a trace. This approach allows for unrestricted neural network architectures but requires an ODE solver and uses a stochastic estimator to approximate the trace. The approach of iResnet (Behrmann et al., 2019) is to enforce a constraint on the Lipschitz constant of the transformation. In MintNet (Song et al., 2019), they achieve efficient invertibility by restricting their weight matrices to be triangular.

Flow as a Special Case of Latent Variable Model

Flows can be seen as a special case of a latent variable model with the following equation:

$$p(x) = \int_z \delta(x - f^{-1}(z)) \left| \det \frac{\partial f(x)}{\partial x} \right| p(z)$$

The delta function term and the det-Jac term can be interpreted as $p(x|z)$ of the latent variable

model. The main distinction between flow and latent variable models is the invertibility of the flow transformation.

2.1.3 Auto-Regressive Models

Auto-Regressive models are based on the factorized decomposition of the joint distribution:

$$p(x) = p(x_1) \prod_{i=2}^D p(x_i | x_{1:i-1}).$$

The conditional dependence of $p(x_i | x_{1:i-1})$ is often modelled using a sequential model such as RNNs, CNNs, and more recently, attention models (Bahdanau et al., 2014). The distribution of $p(x_i | x_{1:i-1})$ is often a simple distribution such as a Gaussian or Categorical distribution. Auto-regressive models have seen success on many data types. For text data, attention models based on the Transformer (Vaswani et al., 2017) have become increasingly popular (Devlin et al., 2018; Peters et al., 2018). For raw audio, the WaveNet model (van den Oord et al., 2016a) used CNNs to model the sequential dependencies. Auto-regressive models have also seen strong success on data types that aren't inherently sequential, such as images. PixelRNN (van den Oord et al., 2016b) and PixelCNN (van den Oord et al., 2016c) modeled pixels sequentially and achieved state-of-the-art performance when introduced. This was further improved with attention-based auto-regressive models such as the Image Transformer (Parmar et al., 2018) and PixelSnail (Chen et al., 2018). The rational behind the improved results is that the effective receptive field of the attention models is larger than those of the CNN-based models.

One limitation of auto-regressive models is that they must sample sequentially, rather than in parallel. Because of this, it can be quite slow to sample, especially for fine grained audio or large images. The follow-up work to Wavenet, Parallel Wavenet (van den Oord et al., 2018) performed what they called density distillation, which transforms noise into samples, so that samples can be generated in parallel.

Auto-Regressive Models as Normalizing Flows

Interestingly, when an auto-regressive model has Gaussian conditional distributions, then

it can be interpreted as a normalizing flow. This observation was first made in Kingma et al. (2016) and further elaborated in Papamakarios et al. (2017). To understand this interpretation, consider an auto-regressive model with Gaussian conditional distributions:

$$p(x_i|x_{1:i-1}) = \mathcal{N}(x_i|\mu_i, \sigma_i^2) \quad \text{where} \quad \mu_i = f_{\mu_i}(x_{1:i-1}) \quad \text{and} \quad \sigma_i = f_{\sigma_i}(x_{1:i-1}) \quad (2.14)$$

To sample from this model, we must generate each dimension sequentially, so the sample is defined recursively as follows:

$$x_i = z_i\sigma_i + \mu_i \quad \text{where} \quad z_i \sim \mathcal{N}(0, 1) \quad (2.15)$$

The Jacobian $\frac{\partial x}{\partial z}$ of this transformation is triangular due to the auto-regressive dependency. To invert this transformation (recover z from x), we simply need to compute:

$$z_i = \frac{x_i - \mu_i}{\sigma_i} \quad (2.16)$$

Thus, given the invertibility of the auto-regressive transformation, we can interpret it as a flow transformation with the change of variable formula (Eqn. 2.9). The Masked Autoregressive Flow from Papamakarios et al. (2017) takes advantage of this interpretation by stacking auto-regressive transformations, just like one would with flow models.

2.2 Beyond Standard VAEs

A VAE consists of three important pieces: 1) the prior distribution, 2) the variational distribution, and 3) the decoder distribution. A standard VAE typically involves a $N(0, I)$ prior, an encoder that outputs a factorized Gaussian distribution, and a decoder that outputs a factorized distribution over the data. In this section, I will review some current techniques for improving all three aspects of VAEs. Many of the improvements involve changing the posterior approximation (variational distribution), which results in a change to the standard lower bound. At the end of this section, there will be a table of the various lower bounds, which will help to summarize a number of posterior improvements.

2.2.1 Importance Weighted Autoencoders

The Importance Weighted Autoencoder (IWAE, [Burda et al. \(2016\)](#)) is a generative model which shares the VAE architecture, but uses importance weighting in order to train with a tighter lower bound of the data log-likelihood. More specifically, the IWAE lower bound takes multiple samples from the q distribution and computes the following bound:

$$\log p(x) \geq \mathbb{E}_{z_1 \dots z_k \sim q(z|x)} \left[\log \left(\frac{1}{k} \sum_{i=1}^k \frac{p(x, z_i)}{q(z_i|x)} \right) \right]. \quad (2.17)$$

As can be seen from the equation above, when $k = 1$, then we recover the VAE lower bound. Importantly, the importance weighted lower bound approaches $\log p(x)$ as k goes to infinity. Consequently, this lower bound is often used with a large k in order to evaluate latent variable models ([Kingma et al., 2016](#); [Sønderby et al., 2016](#)).

To better understand the IWAE objective, as shown in [Burda et al. \(2016\)](#), it can be informative to derive the gradient update of the IWAE bound with respect to the parameters of p and q :

$$\begin{aligned} \nabla \log \left(\frac{1}{k} \sum_{i=1}^k \frac{p(x, z_i)}{q(z_i|x)} \right) &= \left(\frac{1}{k} \sum_{i=1}^k \frac{p(x, z_i)}{q(z_i|x)} \right)^{-1} \nabla \left(\frac{1}{k} \sum_{i=1}^k \frac{p(x, z_i)}{q(z_i|x)} \right) \\ &= \left(\frac{1}{k} \sum_{i=1}^k \frac{p(x, z_i)}{q(z_i|x)} \right)^{-1} \frac{1}{k} \sum_{i=1}^k \frac{p(x, z_i)}{q(z_i|x)} \nabla \log \frac{p(x, z_i)}{q(z_i|x)} \\ &= \frac{1}{k} \sum_{i=1}^k \tilde{w}_i \nabla \log \frac{p(x, z_i)}{q(z_i|x)} \end{aligned} \quad (2.18)$$

where $\tilde{w}_i = \frac{w_i}{\frac{1}{k} \sum_{i=1}^k w_i}$ and $w_i = \frac{p(x, z_i)}{q(z_i|x)}$. The gradient $\nabla \log \frac{p(x, z_i)}{q(z_i|x)}$ is the gradient of a regular VAE. Thus we can see that the IWAE gradient is similar to the regular VAE objective but where each sample is weighed by the normalized importance weights.

When training with this tighter lower bound, [Burda et al. \(2016\)](#) show that the deep latent variable model achieves a higher data log-likelihood compared to the single sample VAE objective. Furthermore, [Burda et al. \(2016\)](#) observed that the IWAE bound increased the number of active dimensions in the latent space. Reducing inactive latent dimensions is important because they cause the model to only use a fraction of its full capacity.

2.2.2 Flow Posteriors

As discussed in the section on variational inference (2.1.1), we can achieve a tighter lower bound when the approximate posterior is closer to the true posterior. Thus, to reduce the distribution mismatch, it is beneficial to use flexible approximate posterior distributions. As seen in the Flow section (2.1.2), we can create complex distributions through the use of normalizing flows. More specifically, if we transform a random variable z_0 with a simple base distribution $q_0(z)$, the resulting random variable $z_T = T(z_0)$ has a distribution:

$$q_T(z_T) = q_0(z_0) \left| \det \frac{\partial z_T}{\partial z_0} \right|^{-1} \quad (2.19)$$

By successively applying these transformations, we can build arbitrarily complex distributions which can better approximate the true posterior. An important property of these transformations is that we can take expectations with respect to the transformed density $q_T(z_T)$ without explicitly knowing $q_T(z_T)$. The expectation can be written as:

$$\mathbb{E}_{q_T}[h(z_T)] = \mathbb{E}_{q_0}[h(f_T(f_{T-1}(\dots f_1(z_0)))))] \quad (2.20)$$

This is known as the law of the unconscious statistician (LOTUS). Using the change of variable equation (Eqn. 2.19) and LOTUS (Eqn. 2.20), the lower bound of the VAE with a flow posterior is the following:

$$\log p(x) \geq \mathbb{E}_{z_0 \sim q_0(z|x)} \left[\log \left(\frac{p(x, z_T)}{q_0(z_0|x) \prod_{t=1}^T \left| \det \frac{\partial z_t}{\partial z_{t-1}} \right|^{-1}} \right) \right]. \quad (2.21)$$

The first application of normalizing flows for inference in VAEs was the work of [Rezende and Mohamed \(2015\)](#). Since then, there has been many other flows introduced to improve inference in VAEs ([Kingma et al., 2016](#); [Tomczak and Welling, 2016](#); [van den Berg et al., 2018](#); [Huang et al., 2018](#); [Grathwohl et al., 2019](#); [Durkan et al., 2019b](#)).

Flows for Density Estimation vs Variational Inference

It is interesting to note that some flows are suited for density estimation while others are suited for variational inference. For density estimation, the flow needs to efficiently compute

the likelihood of an arbitrary datapoint from the dataset. Whereas for variational inference, the flow needs to efficiently compute the probability density of a sample from the model. This distinction is highlighted in the work of Kingma et al. (2016), where they introduce the Inverse Autoregressive Flow (IAF). As discussed in section 2.1.3, Autoregressive Flows (AF) can efficiently compute the likelihood of a datapoint, but to generate a sample they require sequential generation of each dimension, which is computationally inefficient. In AFs, the model transforms input x to noise z . In contrast, IAFs use an autoregressive flow (Germain et al., 2015) to transform noise z into input space x . The benefit of this is that sampling the model is now very efficient since all dimensions of z can be sampled and transformed into x in parallel, which makes it very well suited for variational inference.

2.2.3 Auxiliary Variables

Another technique that can be used to improve inference in deep latent variable models is the addition of auxiliary variables. See Fig. 2.4 for a graphical model of a VAE augmented with an auxiliary variable model. As the figure shows, the generative process from z to x is unchanged, whereas the inference of z is now dependent on auxiliary variable v .

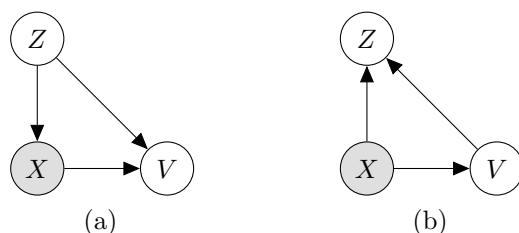


Figure 2.4: Graphical model of a) the generative process and b) the inference model of a VAE with an auxiliary variable.

Just as hierarchical Bayesian models induce dependencies between data, hierarchical variational models can induce dependencies between latent variables. They can capture structure of correlated variables because they turn the posterior into a mixture distribution:

$$q(z|x) = \int q(z|x, v)q(v|x)dv \quad (2.22)$$

The addition of the auxiliary variable changes the lower bound to:

$$\log p(x) \geq \mathbb{E}_{z,v \sim q(z,v|x)} \left[\log \left(\frac{p(x,z)r(v|x,z)}{q(z|x,v)q(v|x)} \right) \right] \quad (2.23)$$

where $r(v|x,z)$ is called the reverse model. This idea has been employed in auxiliary deep generative models (ADGM, [Maaløe et al. \(2016\)](#)) and hierarchical variational models (HVM, [Ranganath et al. \(2016\)](#)). In order to increase the flexibility of the distributions over the auxiliary variables, these models often take advantage of normalizing flows. In [Ranganath et al. \(2016\)](#), flows are used to increase the complexity of the reverse model $r(v|x,v)$ and in MNF ([Louizos and Welling, 2017](#)), normalizing flows transform the auxiliary variable of a variational Bayesian neural net.

2.2.4 Hamiltonian Variational Inference

Another important method for approximate inference is Markov Chain Monte Carlo (MCMC). The advantage of MCMC over variational inference (VI) is that it is nonparametric and asymptotically exact, whereas the advantage of VI is that it optimizes an explicit objective and it is often faster. Thus there has been work to combine these methods in order to improve the approximation to the posterior.

For instance, in Hamiltonian Variational Inference (HVI, [Salimans et al. \(2015\)](#)) they combine the MCMC sampler of Hamiltonian Monte Carlo (HMC, [Neal \(2011\)](#)) with VI. In HMC, an auxiliary variable v , referred to as the momentum, is randomly initialized and then the method simulates the dynamics of the Hamiltonian defined by the latent variable and the momentum: $H(z,v) = 0.5v^T v - \log p(x,z)$, where z and v are iteratively updated using the leapfrog integrator:

$$\begin{aligned} v_{\frac{\epsilon}{2}} &= v_0 - \frac{\epsilon}{2} \frac{\partial}{\partial z} \log p(x,z) \\ z_T &= z_0 + \epsilon v_{\frac{\epsilon}{2}} \\ v_T &= v_{\frac{\epsilon}{2}} - \frac{\epsilon}{2} \frac{\partial}{\partial z} \log p(x,z) \end{aligned}$$

Since the leap frog steps are invertible, HVI can be interpreted as a flow on an augmented

space. Thus HVI is a combination of an auxiliary variable model and a flow model. Also, the flow is volume-preserving because its log det-Jacobian is zero. Thus, the lower bound that HVI maximizes is:

$$\log(p(x)) \geq \mathbb{E}_{z_0, v_0 \sim q(z, v|x)} \left[\log \left(\frac{p(x, z_T) r(v_T|x, z_T)}{q(z_0|x) q(v_0|x, z_0)} \right) \right]. \quad (2.24)$$

Augmenting VI with HMC dynamics can be an effective way of exploring the posterior distribution because the samples are guided by the gradient of the exact log posterior. However, a disadvantage of HVI is that for every step, we need to compute the gradient of $\log p(x, z)$, which can be computationally expensive.

2.2.5 Hierarchical Latent Variable

Typical VAEs possess a single multi-dimensional latent variable z , however, to increase the expressiveness of the model, VAEs can be augmented with a hierarchy of latent variables. There are various options for the conditioning of the hierarchical variables, as shown in Fig. 2.5a and Fig. 2.5b. The preferred hierarchical architecture will depend on the given task.

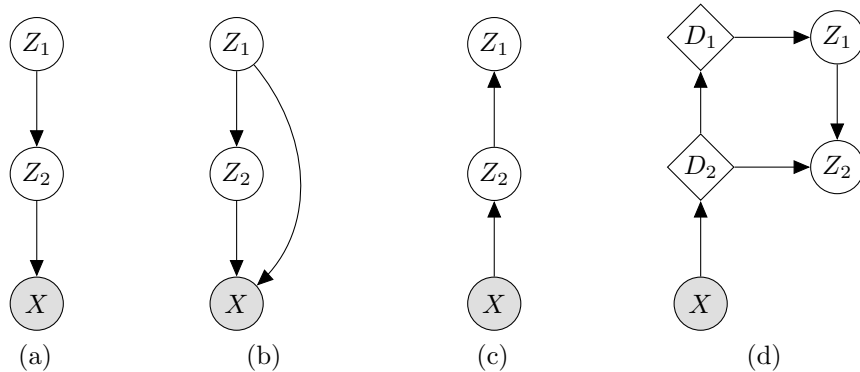


Figure 2.5: Hierarchical latent variable models. a) and b) are hierarchical models with differing dependencies. c) and d) are two different inference models. d) is the inference model used by Ladder VAEs.

For Fig. 2.5a, the joint distribution of the model is now given by:

$$p(x, z_{1:L}) = p(x|z)p(z_1) \prod_{i=2}^L p(z_i|z_{1:i-1})$$

and if we substitute this joint probability into the variational lower bound, the result is:

$$\log p(x) \geq \mathbb{E}_{z_{1:L} \sim q(z_{1:L}|x)} \left[\log \left(\frac{p(x|z_{1:L})p(z_1) \prod_{i=2}^L p(z_i|z_{1:i-1})}{q(z_{1:L}|x)} \right) \right]. \quad (2.25)$$

Augmenting the model with multiple latent variables is an approach taken in [Burda et al. \(2016\)](#), [Sønderby et al. \(2016\)](#), [Razavi et al. \(2019\)](#), and [Maaløe et al. \(2019\)](#). This work has shown that extra layers of stochastic variables improve the models density estimation.

Modifying the model with extra latent variables is often accompanied by modifications in the inference process. For instance, [Sønderby et al. \(2016\)](#) introduce the ladder VAE which incorporates a top-down dependency structure both in the inference and generative models, as shown in Fig. 2.5d. This is in contrast to a more basic inference model shown in Fig. 2.5c. With this modification, the approximate posterior distribution benefits from the merging of information between the top-down prior from the generative distribution and the inference model.

2.2.6 More Expressive Priors

Typical VAEs often use a Normal distribution with zero mean and diagonal unit variance for the latent variable prior distribution $p(z)$. This prior is used due to its simplicity, however it is quite restrictive. Just as limited approximate posteriors impede learning, limited priors make it more challenging for the model to learn the data distribution. Ideally, the prior distribution of the latent variable $p(z)$ should match the marginal distribution of the data posteriors: $\mathbb{E}_{p_D(x)} [p(z|x)]$, where $p_D(x)$ is the data distribution. Matching these distributions is desired because it achieves the maximum likelihood for the parameters given the datapoints.

The problem of the mismatch between the prior and the marginal posterior (also known as the aggregate posterior) has been remarked in a number of works ([Makhzani et al., 2015](#); [Huang et al., 2017](#); [Rosca et al., 2018](#)). There have been a several approaches to addressing this problem. In [Makhzani et al. \(2015\)](#), they use an adversarial loss to fit the aggregate of the posteriors to the prior. Other works have shown that there are benefits to learning the prior rather than using a fixed prior. For instance, in [Tomczak and Welling \(2018\)](#), they propose the VampPrior for $p(z)$. The VampPrior consists of a mixture distribution with components

given by variational posteriors conditioned on learnable pseudo-inputs. The approach taken by [Chen et al. \(2016\)](#) and [Huang et al. \(2017\)](#) is to use an autoregressive flow as the prior. [Chen et al. \(2016\)](#) show that using a latent code transformed by an autoregressive flow is equivalent to using inverse autoregressive flow (IAF) as an approximate posterior. In [Razavi et al. \(2019\)](#), they use an autoregressive model, PixelCNN, as their prior over the latent variable. These works demonstrate the VAE can be significantly improved when using learned priors.

2.2.7 More Expressive Decoders

The decoder distribution of a standard VAE is a factorized distribution over the dimensions. In other words, the dimensions are conditionally independent given the latent variable. This distribution is represented in Fig. 2.6a.

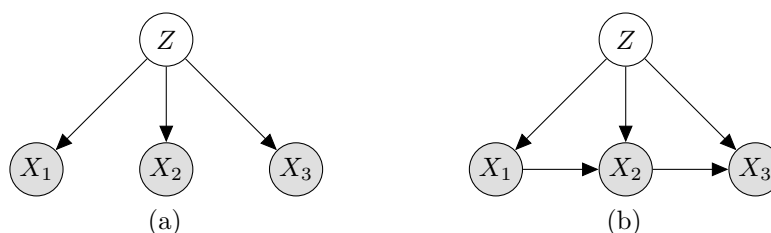


Figure 2.6: Graphical model of a) the standard factorized decoder distribution and b) the auto-regressive decoder.

To improve the modelling capacity of the decoder, some works have utilized auto-regressive decoders. As shown in Fig. 2.6b, this decoder no longer assumes the dimensions are independent given the latent variable. In [Chen et al. \(2016\)](#) and [Gulrajani et al. \(2016\)](#), they use PixelCNNs ([van den Oord et al., 2016c](#)) as decoders. As discussed in the auto-regressive model section (2.1.3), PixelCNNs are CNNs designed to have an auto-regressive structure (each pixel is conditioned on previous pixels). This auto-regressive decoder provides much more flexibility to the output distribution $p(x|z)$ in comparison to an output distribution with independent dimensions. The results of [Chen et al. \(2016\)](#) and [Gulrajani et al. \(2016\)](#) show that auto-regressive decoders have improved performance over typical decoders.

However, in [Bowman et al. \(2016\)](#), they observed that combining these auto-regressive

models and latent variable models in this manner lead to the auto-regressive part of the model explaining most of the structure in the data, while the latent variables has little use. In [Chen et al. \(2016\)](#), they propose the Variational Lossy Autoencoder (VLAE), which limits the auto-regressive decoder so that the latent variable has more of the variance in the data to model. For instance, they construct a specific factorization of the auto-regressive distribution such that it has a small local receptive field, which results in the latent variable capturing the global representation, rather than the detailed texture of the image.

2.2.8 Summary of the Lower Bounds

Model	Lower Bound	Inference	Generative
VAE	$\mathbb{E}_{z \sim q(z x)} \left[\log \left(\frac{p(x,z)}{q(z x)} \right) \right]$ (Eqn. 2.8)		
IWAE	$\mathbb{E}_{z_1 \dots z_k \sim q(z x)} \left[\log \left(\frac{1}{k} \sum_{i=1}^k \frac{p(x, z_i)}{q(z_i x)} \right) \right]$ (Eqn. 2.17)	✓	
NF	$\mathbb{E}_{z_0 \sim q(z_0 x)} \left[\log \left(\frac{p(x, z_T)}{q(z_0 x) \prod_{t=1}^T \left \det \frac{\partial z_t}{\partial z_{t-1}} \right ^{-1}} \right) \right]$ (Eqn. 2.21)	✓	
Aux	$\mathbb{E}_{z, v \sim q(z, v x)} \left[\log \left(\frac{p(x, z) r(v x, z)}{q(z x, v) q(v x)} \right) \right]$ (Eqn. 2.23)	✓	
HVI	$\mathbb{E}_{z_0, v_0 \sim q(z, v x)} \left[\log \left(\frac{p(x, z_T) r(v_T x, z_T)}{q(z_0 x, v_0) q(v_0 x)} \right) \right]$ (Eqn. 2.24)	✓	
Hierarchical	$\mathbb{E}_{z_{1:L} \sim q(z_{1:L} x)} \left[\log \left(\frac{p(x z_{1:L}) p(z_1) \prod_{l=2}^{L-1} p(z_l z_{l < i})}{q(z_{1:L} x)} \right) \right]$ (Eqn. 2.25)		✓
Prior+	$\mathbb{E}_{z \sim q(z x)} \left[\log \left(\frac{p(x z) p^+(z)}{q(z x)} \right) \right]$ (Sec. 2.2.6)		✓
Generator+	$\mathbb{E}_{z \sim q(z x)} \left[\log \left(\frac{p^+(x z) p(z)}{q(z x)} \right) \right]$ (Sec. 2.2.7)		✓

Table 2.1: Summary of the lower bounds from this background section. The Inference column refers to improvements to the inference model and the Generative column refers to improvements to the generative model.

The previous sections covered the lower bound objectives for a number of VAE variants. In order to contrast them more easily, Table 2.1 summarizes the lower bounds from this background section.

These improvements to the VAE can be sorted into two groups: improved inference and improved generative model. Inference can be improved through IWAE, normalizing flows, auto-regressive models, and auxiliary variables. The generative model can be improved at the level of the prior or the decoder and it can be augmented with a hierarchy of latent variables.

Chapter 3

Reinterpreting Importance-Weighted Autoencoders

The standard interpretation of importance-weighted autoencoders is that they maximize a tighter lower bound on the marginal likelihood than the standard evidence lower bound. We give an alternate interpretation of this procedure: that it optimizes the standard variational lower bound, but using a more complex distribution. The contribution of this section is to derive this result, present a tighter lower bound, visualize the implicit importance-weighted distribution and provide the algorithms for plotting and sampling from this distribution.

This chapter comes from the work done in [Cremer et al. \(2017\)](#), which was a collaboration between myself, Quaid Morris and David Duvenaud. In that work, I produced the figures and most of the writing was done by David and myself.

3.1 Background

As introduced in Section 2.2.1, the importance-weighted autoencoder (IWAE; [Burda et al. \(2016\)](#)) is a variational inference strategy capable of producing arbitrarily tight evidence lower bounds. The improved tightness of the bound leads to improved models and it is also often used as an evaluation metric. IWAE maximizes the following multi-sample evidence lower

bound (ELBO):

$$\log p(x) \geq E_{z_1 \dots z_k \sim q(z|x)} \left[\log \left(\frac{1}{k} \sum_{i=1}^k \frac{p(x, z_i)}{q(z_i|x)} \right) \right] = \mathcal{L}_{IWAE}[q] \quad (\text{IWAE ELBO})$$

which is a tighter lower bound than the ELBO maximized by the variational autoencoder (VAE; Kingma and Welling (2014)):

$$\log p(x) \geq E_{z_1 \dots z_k \sim q(z|x)} \left[\frac{1}{k} \sum_{i=1}^k \log \left(\frac{p(x, z_i)}{q(z_i|x)} \right) \right] = \mathcal{L}_{VAE}[q]. \quad (\text{VAE ELBO})$$

Here we've written the VAE bound as a multisample lower bound to compare it to the IWAE bound. To better understand the difference between these two bounds, we will examine their gradients. As the derivation in Section 2.2.1 showed, the following equations are the gradients of the multi-sample VAE ELBO and the IWAE ELBO, respectively:

$$\nabla_{\Theta} \mathcal{L}_{VAE}[q] = E_{z_1 \dots z_k \sim q(z|x)} \left[\sum_{i=1}^k \frac{1}{k} \nabla_{\Theta} \log \left(\frac{p(x, z_i)}{q(z_i|x)} \right) \right] \quad (3.1)$$

$$\nabla_{\Theta} \mathcal{L}_{IWAE}[q] = E_{z_1 \dots z_k \sim q(z|x)} \left[\sum_{i=1}^k \tilde{w}_i \nabla_{\Theta} \log \left(\frac{p(x, z_i)}{q(z_i|x)} \right) \right] \quad (3.2)$$

where

$$\tilde{w}_i = \frac{\frac{p(x, z_i)}{q(z_i|x)}}{\sum_{j=1}^k \frac{p(x, z_j)}{q(z_j|x)}}.$$

From equations (3.1) and (3.2), we see that the gradient of the VAE ELBO evenly weights the samples, whereas the IWAE gradient weights the samples based on their relative importance \tilde{w}_i .

In this work, we would like to know what approximate posterior distribution is IWAE effectively sampling from? In other words, since the VAE ELBO is a lower bound due to the divergence between the approximate $q(z|x)$ and the true posterior $p(z|x)$:

$$\log p(x) = \mathcal{L}_{VAE}[q] + KL(q(z|x)||p(z|x))$$

then what is q_{IW} in

$$\log p(x) = \mathcal{L}_{IWAE}[q] + KL(q_{IW} || p(z|x))?$$

3.2 Defining the implicit distribution \tilde{q}_{IW}

In this section, we present the implicit distribution that arises from importance sampling from a distribution p using q as a proposal distribution. Given a batch of samples $z_2 \dots z_k$ from $q(z|x)$, the following is the unnormalized importance-weighted distribution:

$$\tilde{q}_{IW}(z|x, z_{2:k}) = \frac{\frac{p(x,z)}{q(z|x)}}{\frac{1}{k} \sum_{j=1}^k \frac{p(x,z_j)}{q(z_j|x)}} q(z|x) = \frac{p(x,z)}{\frac{1}{k} \left(\frac{p(x,z)}{q(z|x)} + \sum_{j=2}^k \frac{p(x,z_j)}{q(z_j|x)} \right)} \quad (3.3)$$

The batch of samples is indexed starting from z_2 because the z that is being evaluated is considered z_1 . Here are some properties of the approximate IWAE posterior:

- When $k = 1$, $\tilde{q}_{IW}(z|x, z_{2:k})$ equals $q(z|x)$.
- When $k > 1$, the form of $\tilde{q}_{IW}(z|x, z_{2:k})$ depends on the true posterior $p(z|x)$.
- As $k \rightarrow \infty$, $\mathbb{E}_{z_2 \dots z_k} [\tilde{q}_{IW}(z|x, z_{2:k})]$ approaches the true posterior $p(z|x)$ pointwise.

As a demonstration, see Fig. 3.1 for a visualization of a 1D \tilde{q}_{IW} with different batches of $z_2 \dots z_k$. In this figure, the blue $p(z)$ distribution is the distribution that the green $q(z)$ distribution is attempting to approximate. The three instances of \tilde{q}_{IW} have different z samples from $q(z)$. We can see that they are unnormalized and that they are moving closer to the target $p(z)$ compared to the base distribution $q(z)$.

3.2.1 \tilde{q}_{IW} approaches the true posterior

Why does $\mathbb{E}_{z_2 \dots z_k} [\tilde{q}_{IW}(z|x, z_{2:k})]$ approach the true posterior $p(z|x)$ as $k \rightarrow \infty$? Recall that the marginal likelihood can be approximated by importance sampling:

$$p(x) = E_{q(z|x)} \left[\frac{p(x,z)}{q(z|x)} \right] \approx \frac{1}{k} \sum_i^k \frac{p(x, z_i)}{q(z_i|x)} \quad (3.4)$$

where z_i is sampled from $q(z|x)$. Eqn. 3.4 is in the denominator of \tilde{q}_{IW} , meaning that it is approximating $p(x)$. If $q(z|x)$ has positive probability for all regions where $p(z|x)$ is positive, then it follows from the strong law of large numbers that, as k approaches infinity, \tilde{q}_{IW} converges to the true posterior $p(z|x)$ almost surely. This interpretation becomes clearer when we factor out the true posterior from \tilde{q}_{IW} :

$$\tilde{q}_{IW}(z|x, z_{2:k}) = \frac{p(x)}{\frac{1}{k} \left(\frac{p(x,z)}{q(z|x)} + \sum_{j=2}^k \frac{p(x,z_j)}{q(z_j|x)} \right)} p(z|x) \quad (3.5)$$

We see that the closer the denominator becomes to $p(x)$, the closer \tilde{q}_{IW} is to the true posterior.

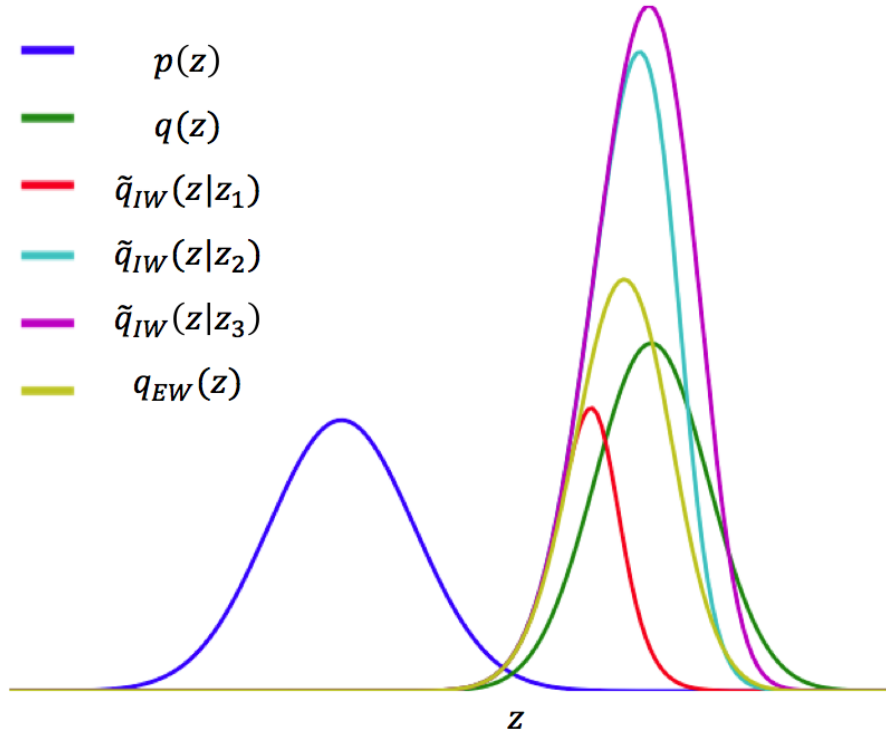


Figure 3.1: Visualization of 1D \tilde{q}_{IW} and q_{EW} distributions. The blue $p(z)$ distribution is the distribution that the green $q(z)$ distribution is attempting to approximate. Both $p(z)$ and $q(z)$ are normalized. The three instances of \tilde{q}_{IW} have different z samples from $q(z)$ and we can see that they are unnormalized. The yellow $q_{EW}(z)$ (defined in section 3.2.3) is the expectation over 30 \tilde{q}_{IW} distributions and we can see that it is a normalized distribution. These distributions were plotted using Algorithm 2.

3.2.2 Recovering the IWAE bound from the VAE bound

Here we show that the IWAE ELBO is equivalent to the VAE ELBO with \tilde{q}_{IW} (Eqn. 3.3), where \tilde{q}_{IW} is a more flexible, unnormalized distribution, implicitly defined by importance sampling. This equivalence requires an expectation over the batches of $z_2 \dots z_k$ for the VAE bound in order to define \tilde{q}_{IW} . If we replace $q(z|x)$ with $\tilde{q}_{IW}(z|x, z_{2:k})$ and take an expectation over $z_2 \dots z_k$, then we recover the IWAE ELBO:

$$\begin{aligned} \mathbb{E}_{z_2 \dots z_k \sim q(z|x)} [\mathcal{L}_{VAE}[\tilde{q}_{IW}(z|z_{2:k})]] &= \mathbb{E}_{z_2 \dots z_k \sim q(z|x)} \left[\int_z \tilde{q}_{IW}(z|z_{2:k}) \log \left(\frac{p(x, z)}{\tilde{q}_{IW}(z|x, z_{2:k})} \right) dz \right] \\ &= \mathbb{E}_{z_2 \dots z_k \sim q(z|x)} \left[\int_z \tilde{q}_{IW}(z|z_{2:k}) \log \left(\frac{1}{k} \sum_{i=1}^k \frac{p(x, z_i)}{q(z_i|x)} \right) dz \right] \\ &= \mathbb{E}_{z_1 \dots z_k \sim q(z|x)} \left[\log \left(\frac{1}{k} \sum_{i=1}^k \frac{p(x, z_i)}{q(z_i|x)} \right) \right] = \mathcal{L}_{IWAE}[q] \end{aligned}$$

For a more detailed derivation, see section 3.6.1. Note that we are abusing the VAE lower bound notation because this implies an expectation over an unnormalized distribution. Consequently, we replace the expectation with an equivalent integral.

3.2.3 Expected importance weighted distribution q_{EW}

Given that the previous section took an expectation over $z_2 \dots z_k$ for $\mathcal{L}_{VAE}[\tilde{q}_{IW}(z|z_{2:k})]$, it is interesting to ask what is the distribution that we're sampling in expectation? Accordingly, the expected importance-weighted distribution $q_{EW}(z|x)$ is a distribution given by:

$$q_{EW}(z|x) = E_{z_2 \dots z_k \sim q(z|x)} [\tilde{q}_{IW}(z|x, z_{2:k})] = E_{z_2 \dots z_k \sim q(z|x)} \left[\frac{p(x, z)}{\frac{1}{k} \left(\frac{p(x, z)}{q(z|x)} + \sum_{j=2}^k \frac{p(x, z_j)}{q(z_j|x)} \right)} \right] \quad (3.6)$$

Unlike q_{IW} , q_{EW} is a normalized distribution; we show this in section 3.6.2. Interestingly, if we could use q_{EW} in the VAE ELBO, $\mathcal{L}_{VAE}[q_{EW}]$, then the result would be in an upper bound of $\mathcal{L}_{IWAE}[q]$. Unfortunately, the expectation in q_{EW} is intractable. See Section A.1.1 in the Appendix for the proof of $\mathcal{L}_{VAE}[q_{EW}] \geq \mathcal{L}_{IWAE}[q]$, provided by Christian Naesseth, which is a special case of the proof in Naesseth et al. (2018).

Algorithm 1 Sampling $q_{EW}(z|x)$

```

1:  $k \leftarrow$  number of importance samples
2: for  $i$  in  $1 \dots k$  do
3:    $z_i \sim q(z|x)$ 
4:    $w_i = \frac{p(x, z_i)}{q(z_i|x)}$ 
5: Each  $\tilde{w}_i = w_i / \sum_{i=1}^k w_i$ 
6:  $j \sim \text{Categorical}(\tilde{\mathbf{w}})$ 
7: Return  $z_j$ 
    
```

Algorithm 2 Plotting $q_{EW}(z|x)$

```

1:  $k \leftarrow$  number of importance samples
2:  $S \leftarrow$  number of function samples
3:  $L \leftarrow$  locations to plot
4:  $\hat{f} = \text{zeros}(|L|)$ 
5: for  $s$  in  $1 \dots S$  do
6:    $z_2 \dots z_k \sim q(z|x)$ 
7:    $\hat{p}(x) = \sum_{i=2}^k \frac{p(x, z_i)}{q(z_i|x)}$ 
8:   for  $z$  in  $L$  do
9:      $\hat{f}[z] += \frac{p(x, z)}{\frac{1}{k} (\frac{p(x, z)}{q(z|x)} + \hat{p}(x))}$ 
10: Return  $\hat{f}/S$ 
    
```

3.2.4 Proof that q_{EW} is closer to the true posterior than q

The proof of Section A.1.1 tells us that $\mathcal{L}_{IWAE}(q) \leq \mathcal{L}_{VAE}(q_{EW})$. That is, the IWAE ELBO with the base q is a lower bound to the VAE ELBO with the importance weighted q_{EW} . Due to Jensen's inequality and as shown in [Burda et al. \(2016\)](#), we know that the IWAE ELBO is an upper bound of the VAE ELBO: $L_{IWAE}(q) \geq L_{VAE}(q)$. Furthermore, the log marginal likelihood can be factorized into: $\log(p(x)) = L_{VAE}(q) + KL(q||p)$, and rearranged to: $KL(q||p) = \log(p(x)) - L_{VAE}(q)$. Following the observations above, we have:

$$KL(q_{EW}||p) = \log(p(x)) - L_{VAE}[q_{EW}] \quad (3.7)$$

$$\leq \log(p(x)) - L_{IWAE}[q] \quad (3.8)$$

$$\leq \log(p(x)) - L_{VAE}[q] = KL(q||p) \quad (3.9)$$

Thus, $KL(q_{EW}||p) \leq KL(q||p)$, meaning q_{EW} is closer to the true posterior than q in terms of KL divergence.

3.3 Visualizing the nonparameteric approximate posterior

The IWAE approximating distribution is nonparametric in the sense that, as the true posterior grows more complex, so does the shape of \tilde{q}_{IW} and q_{EW} . This makes plotting these distributions challenging. A kernel-density-estimation approach could be used, but requires many samples. Thankfully, equations (3.3) and (3.6) give us a simple and fast way to approximately plot \tilde{q}_{IW} and q_{EW} without introducing artifacts due to kernel density smoothing. In Algorithm 2, we provide the steps necessary for plotting q_{EW} . To plot \tilde{q}_{IW} , we use the same algorithm but with a single function sample, i.e. $S=1$.

With this algorithm we can visualize how \tilde{q}_{IW} looks with different z samples. As was shown in Fig. 3.1, we visualize different \tilde{q}_{IW} distributions and q_{EW} in 1D. Similarly, Fig. 3.2 utilizes Algorithm 2 to visualize q_{EW} on a 2D distribution approximation problem. The base distribution q is a Gaussian. As we increase the number of samples k and keep the base distribution fixed, we see that the approximation approaches the true distribution.

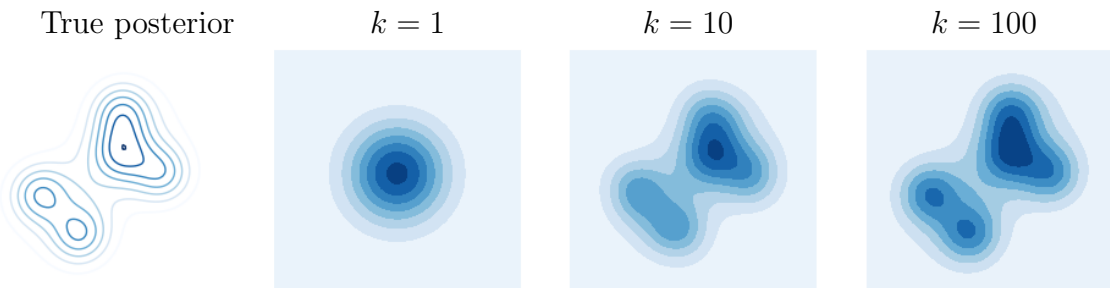


Figure 3.2: q_{EW} approximations to a complex true distribution. As k grows, this approximation approaches the true distribution.

3.4 Resampling for prediction

When training a VAE with the IWAE ELBO, we sample the q distribution and the objective implicitly weights the samples (as seen in Eqn. 3.2). After training, in order to sample from the implicit q distribution, we need to explicitly reweight samples from q . In other words, we trained the model with $q_{EW}(z|x)$, thus we need to sample $q_{EW}(z|x)$ in order to sample

the approximate posterior of the model properly. The procedure to sample from $q_{EW}(z|x)$ is shown in Algorithm 1. It is equivalent to sampling-importance-resampling (SIR).

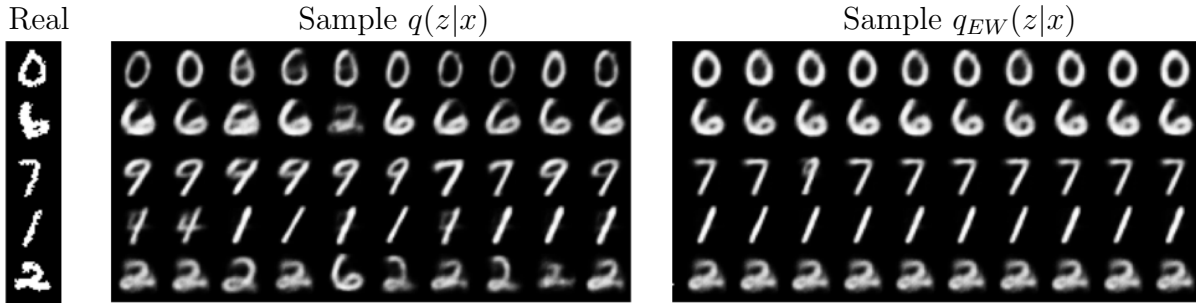


Figure 3.3: Reconstructions of MNIST samples from $q(z|x)$ and q_{EW} . The model was trained by maximizing the IWAE ELBO with $K=50$ and 2 latent dimensions. The reconstructions from $q(z|x)$ are greatly improved with the sampling-resampling step of q_{EW} .

In figure 3.3, we demonstrate the need to sample from $q_{EW}(z|x)$ rather than the base distribution $q(z|x)$ for reconstructing MNIST digits. We trained the model to maximize the IWAE ELBO with $K=50$ and 2 latent dimensions, similar to Appendix C in [Burda et al. \(2016\)](#). When we sample from $q(z|x)$ and reconstruct the samples, we see a number of anomalies. However, if we perform the sampling-resampling step (Alg. 1), then the reconstructions are much more accurate. The intuition here is that we trained the model with q_{EW} with $K = 50$ then sampled from $q(z|x)$ (q_{EW} with $K = 1$), which can be very different distributions, as seen in Fig. 3.2.

3.5 Discussion

[Bachman and Precup \(2015\)](#) also showed that the IWAE objective is equivalent to stochastic variational inference with a proposal distribution corrected towards the true posterior via normalized importance sampling. We build on this idea by further examining \tilde{q}_{IW} and by providing visualizations to help better grasp the interpretation. This work also informs readers on how to properly plot and sample from the approximate posterior of the model trained with IWAE.

Our work covered a number of marginal log-likelihood lower bounds, so to summarize our

observations, the following is the ordering of lower bounds given specific proposal distributions,

$$\log p(x) \geq L_{VAE}[q_{EW}] \geq \mathbb{E}_{z_2, \dots, z_k \sim q(z|x)} [L_{VAE}[\tilde{q}_{IW}(z|z_{2:k})]] = L_{IWAE}[q] \geq L_{VAE}[q]$$

In light of this, IWAE can be related to approaches which increase the complexity of the approximate distribution q , such as Normalizing Flows (Rezende and Mohamed, 2015), Variational Boosting (Miller et al., 2016) or Hamiltonian variational inference (Salimans et al., 2015) and others mentioned in section 2.2. With this interpretation in mind, we can possibly generalize \tilde{q}_{IW} to be applicable to other divergence measures. An interesting avenue of future work is the comparison of IW-based variational families with alpha-divergences or operator variational objectives.

This work also has connections to multi-sample sequential VAE models (Naesseth et al., 2018; Maddison et al., 2017; Le et al., 2018). In order to avoid particle degeneracy as time increases, these models re-weight and sample their particles at intermediate timesteps. In other words, they sample their intermediate importance weighted distribution following Alg. 1, so that low weight particles are not expanded.

Our work also nicely complements work that analyzed the variance of the gradient of the inference network. In Rainforth et al. (2018), they show that using the IWAE bound can be detrimental to learning because it reduces the signal-to-noise ratio of the gradient estimator for the inference network. From the perspective of our work, this occurs because with large k , the approximate posterior q_{EW} approaches the true posterior, thus reducing the need to fit the base distribution $q(z|x)$ to the true posterior $p(z|x)$. Further work (Tucker et al., 2019) has demonstrated that this can be remedied by a second application of the reparameterization trick, resulting in a reduction in the variance of the gradients.

3.6 Detailed Derivations

3.6.1 Detailed derivation of the equivalence of VAE and IWAE bound

Here we show that the expectation over $z_2 \dots z_k$ of the VAE lower bound with the unnormalized importance-weighted distribution \tilde{q}_{IW} , $\mathcal{L}_{VAE}[\tilde{q}_{IW}(z|z_{2:k})]$, is equivalent to the IWAE bound with the original q distribution, $\mathcal{L}_{IWAE}[q]$.

$$\mathbb{E}_{z_2 \dots z_k \sim q(z|x)} [\mathcal{L}_{VAE}[\tilde{q}_{IW}(z|z_{2:k})]] = \mathbb{E}_{z_2 \dots z_k \sim q(z|x)} \left[\int_z \tilde{q}_{IW}(z|x, z_{2:k}) \log \left(\frac{p(x, z)}{\tilde{q}_{IW}(z|x, z_{2:k})} \right) dz \right] \quad (3.10)$$

$$= \mathbb{E}_{z_2 \dots z_k \sim q(z|x)} \left[\int_z \tilde{q}_{IW}(z|x, z_{2:k}) \log \left(\frac{p(x, z)}{\frac{1}{k} \sum_{i=1}^k \frac{p(x, z_i)}{q(z_i|x)}} \right) dz \right] \quad (3.11)$$

$$= \mathbb{E}_{z_2 \dots z_k \sim q(z|x)} \left[\int_z \tilde{q}_{IW}(z|x, z_{2:k}) \log \left(\frac{1}{k} \sum_{i=1}^k \frac{p(x, z_i)}{q(z_i|x)} \right) dz \right] \quad (3.12)$$

$$= \mathbb{E}_{z_2 \dots z_k \sim q(z|x)} \left[\int_z k \frac{\frac{p(x, z)}{q(z|x)}}{\sum_{j=1}^k \frac{p(x, z_j)}{q(z_j|x)}} q(z|x) \log \left(\frac{1}{k} \sum_{i=1}^k \frac{p(x, z_i)}{q(z_i|x)} \right) dz \right] \quad (3.13)$$

$$= \mathbb{E}_{z_2 \dots z_k \sim q(z|x)} \left[\int_{z_1} k \frac{\frac{p(x, z_1)}{q(z_1|x)}}{\sum_{j=1}^k \frac{p(x, z_j)}{q(z_j|x)}} q(z_1|x) \log \left(\frac{1}{k} \sum_{i=1}^k \frac{p(x, z_i)}{q(z_i|x)} \right) dz \right] \quad (3.14)$$

$$= \mathbb{E}_{z_1 \dots z_k \sim q(z|x)} \left[k \frac{\frac{p(x, z_1)}{q(z_1|x)}}{\sum_{j=1}^k \frac{p(x, z_j)}{q(z_j|x)}} \log \left(\frac{1}{k} \sum_{i=1}^k \frac{p(x, z_i)}{q(z_i|x)} \right) \right] \quad (3.15)$$

$$= \mathbb{E}_{z_1 \dots z_k \sim q(z|x)} \left[\frac{\sum_{j=1}^k \frac{p(x, z_j)}{q(z_j|x)}}{\sum_{j=1}^k \frac{p(x, z_j)}{q(z_j|x)}} \log \left(\frac{1}{k} \sum_{i=1}^k \frac{p(x, z_i)}{q(z_i|x)} \right) \right] \quad (3.16)$$

$$= \mathbb{E}_{z_1 \dots z_k \sim q(z|x)} \left[\log \left(\frac{1}{k} \sum_{i=1}^k \frac{p(x, z_i)}{q(z_i|x)} \right) \right] \quad (3.17)$$

$$= \mathcal{L}_{IWAE}[q]$$

(3.14): Change of notation $z = z_1$.

(3.16): z_i has the same expectation as z_1 so we can replace k with the sum of k terms.

3.6.2 Proof that q_{EW} is a normalized distribution

$$\int_z q_{EW}(z|x) dz = \int_z E_{z_2 \dots z_k \sim q(z|x)} [\tilde{q}_{IW}(z|x, z_{2:k})] dz \quad (3.18)$$

$$= \int_z E_{z_2 \dots z_k \sim q(z|x)} \left[\frac{p(x, z)}{\frac{1}{k} \left(\frac{p(x, z)}{q(z|x)} + \sum_{j=2}^k \frac{p(x, z_j)}{q(z_j|x)} \right)} \right] dz \quad (3.19)$$

$$= \int_z \frac{q(z|x)}{q(z|x)} E_{z_2 \dots z_k \sim q(z|x)} \left[\frac{p(x, z)}{\frac{1}{k} \left(\frac{p(x, z)}{q(z|x)} + \sum_{j=2}^k \frac{p(x, z_j)}{q(z_j|x)} \right)} \right] dz \quad (3.20)$$

$$= E_{z \sim q(z|x)} E_{z_2 \dots z_k \sim q(z|x)} \left[\frac{\frac{p(x, z)}{q(z|x)}}{\frac{1}{k} \left(\frac{p(x, z)}{q(z|x)} + \sum_{j=2}^k \frac{p(x, z_j)}{q(z_j|x)} \right)} \right] \quad (3.21)$$

$$= E_{z_1 \dots z_k \sim q(z|x)} \left[\frac{\frac{p(x, z_1)}{q(z_1|x)}}{\frac{1}{k} \left(\sum_{j=1}^k \frac{p(x, z_j)}{q(z_j|x)} \right)} \right] \quad (3.22)$$

$$= k \cdot E_{z_1 \dots z_k \sim q(z|x)} \left[\frac{\frac{p(x, z_1)}{q(z_1|x)}}{\sum_{j=1}^k \frac{p(x, z_j)}{q(z_j|x)}} \right] \quad (3.23)$$

$$= \sum_{i=1}^k E_{z_1 \dots z_k \sim q(z|x)} \left[\frac{\frac{p(x, z_i)}{q(z_i|x)}}{\sum_{j=1}^k \frac{p(x, z_j)}{q(z_j|x)}} \right] \quad (3.24)$$

$$= E_{z_1 \dots z_k \sim q(z|x)} \left[\frac{\sum_{i=1}^k \frac{p(x, z_i)}{q(z_i|x)}}{\sum_{j=1}^k \frac{p(x, z_j)}{q(z_j|x)}} \right] \quad (3.25)$$

$$= E_{z_1 \dots z_k \sim q(z|x)} [1] \quad (3.26)$$

$$= 1$$

(3.22): Change of notation $z = z_1$.

(3.24): z_i has the same expectation as z_1 so we can replace k with the sum of k terms.

(3.25): Linearity of expectation.

Chapter 4

Inference Suboptimality in Variational Autoencoders

Amortized inference allows latent-variable models trained via variational learning to scale to large datasets. The quality of approximate inference is determined by two factors: a) the capacity of the variational distribution to match the true posterior and b) the ability of the recognition network to produce good variational parameters for each datapoint. In this chapter, we examine approximate inference in variational autoencoders in terms of these factors. We find that divergence from the true posterior is often due to imperfect recognition networks, rather than the limited complexity of the approximating distribution. We also show that this is due partly to the generator learning to accommodate the choice of approximation. Furthermore, we show that the parameters used to increase the expressiveness of the approximation play a role in generalizing inference rather than simply improving the complexity of the approximation.

This chapter comes from the work done in [Cremer et al. \(2018\)](#), which was a collaboration between myself, Xuechen Li, and David Duvenaud. The experiments were performed by Xuechen and myself, and the writing of the paper was done by all collaborators.

4.1 Introduction

This chapter analyzes inference suboptimality, which is the mismatch between the true and approximate posterior. More specifically, we are interested in understanding what factors cause the gap between the marginal log-likelihood and the evidence lower bound (ELBO) in variational autoencoders VAEs. We refer to this as the *inference gap*. Moreover, we break down the inference gap into two components: the *approximation gap* and the *amortization gap*.

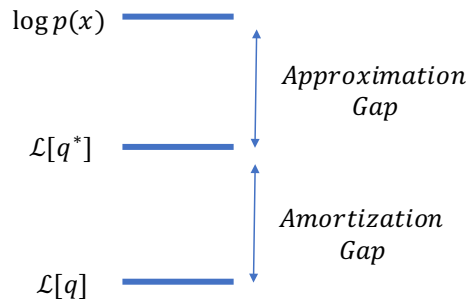


Figure 4.1: Two complementary gaps in inference

The approximation gap comes from the inability of the variational distribution family to exactly match the true posterior. The amortization gap refers to the difference caused by amortizing the variational parameters over the entire training set, instead of optimizing for each training example individually. We refer the reader to Table 4.1 for the definitions of the gaps and to Fig. 4.1 for a simple illustration of the gaps. In Fig. 4.1, $\mathcal{L}[q]$ refers to the ELBO evaluated using an amortized distribution q , as is typical of VAE training. In contrast, $\mathcal{L}[q^*]$ is the ELBO evaluated using the optimal approximation within its variational family.

There has been significant work on improving variational inference in VAEs through the development of expressive approximate posteriors (Rezende and Mohamed, 2015; Kingma et al., 2016; Ranganath et al., 2016; Tomczak and Welling, 2016, 2017). These works have shown that with more expressive approximate posteriors, the model learns a better distribution over the data. Our study aims to gain a better understanding of the relationship between expressive approximations and improved generative models.

Our experiments investigate how the choice of encoder, posterior approximation, decoder,

Term	Definition	KL Formulation
Inference Gap	$\log p(x) - \mathcal{L}[q]$	$\text{KL}(q(z x) p(z x))$
Approximation Gap	$\log p(x) - \mathcal{L}[q^*]$	$\text{KL}(q^*(z x) p(z x))$
Amortization Gap	$\mathcal{L}[q^*] - \mathcal{L}[q]$	$\text{KL}(q(z x) p(z x)) - \text{KL}(q^*(z x) p(z x))$

Table 4.1: Summary of Gap Terms. The middle column refers to the general case where our variational objective is a lower bound on the marginal log-likelihood. The right most column demonstrates the specific case in VAEs. $q^*(z|x)$ refers to the optimal approximation within a family \mathcal{Q} , i.e. $q^*(z|x) = \arg \min_{q \in \mathcal{Q}} \text{KL}(q(z|x)||p(z|x))$.

and optimization affect the approximation and amortization gaps. We train VAE models in a number of settings on the MNIST (LeCun et al., 1998), Fashion-MNIST (Xiao et al., 2017), and CIFAR-10 Krizhevsky and Hinton (2009) datasets.

Our contributions are: a) we investigate inference suboptimality in terms of the approximation and amortization gaps, providing insight to guide future improvements in VAE inference, b) we quantitatively demonstrate that the learned generative model accommodates the choice of approximation, and c) we demonstrate that parameterized functions that improve the expressiveness of the approximation play a significant role in reducing amortization error.

4.2 Background

4.2.1 Inference in Variational Autoencoders

This section will quickly review and highlight some background material from section 2.1.1. Let x be the observed variable, z the latent variable, and $p(x, z)$ be their joint distribution. Given a dataset $X = \{x_1, x_2, \dots, x_N\}$, we would like to maximize the marginal log-likelihood with respect to the model parameters θ :

$$\log p_{\theta}(X) = \sum_{i=1}^N \log p_{\theta}(x_i) = \sum_{i=1}^N \log \int p_{\theta}(x_i, z_i) dz_i.$$

In practice, the marginal log-likelihood is computationally intractable due to the integration over the latent variable z . Instead, VAEs introduce an inference network $q_{\phi}(z|x)$ to approximate the true posterior $p(z|x)$ and optimize the ELBO with respect to model parameters θ

and inference network parameters ϕ :

$$\log p(x) = \mathbb{E}_{q_\phi(z|x)} \left[\log \left(\frac{p_\theta(x, z)}{q_\phi(z|x)} \right) \right] + \text{KL}(q_\phi(z|x) || p_\theta(z|x)) \quad (4.1)$$

$$\geq \mathbb{E}_{q_\phi(z|x)} \left[\log \left(\frac{p_\theta(x, z)}{q_\phi(z|x)} \right) \right] = \mathcal{L}_{\text{VAE}}[q]. \quad (4.2)$$

From the above equation, we see that the ELBO is tight when $q_\phi(z|x) = p_\theta(z|x)$. The choice of $q_\phi(z|x)$ is often a fully factorized Gaussian distribution (q_{FFG}) for its simplicity and efficiency. By utilizing the inference network (also referred to as encoder or recognition network), VAEs amortize inference over the entire dataset.

There are a number of strategies for increasing the expressiveness of approximate posteriors, going beyond the original factorized-Gaussian. In this chapter we will experimenting with normalizing flows and auxiliary variables. See section 2.2.2 and section 2.2.3 for a review of these techniques.

4.3 Methods

4.3.1 Approximation Gap and Amortization Gap

The inference gap \mathcal{G} is the difference between the marginal log-likelihood $\log p(x)$ and a lower bound $\mathcal{L}[q]$. Given the distribution in the family that maximizes the bound, $q^*(z|x) = \arg \max_{q \in \mathcal{Q}} \mathcal{L}[q]$, the inference gap decomposes as the sum of approximation and amortization gaps:

$$\mathcal{G} = \log p(x) - \mathcal{L}[q] = \underbrace{\log p(x) - \mathcal{L}[q^*]}_{\text{Approximation}} + \underbrace{\mathcal{L}[q^*] - \mathcal{L}[q]}_{\text{Amortization}}.$$

For VAEs, we can translate the gaps to KL divergences by rearranging Eqn. (4.1):

$$\mathcal{G}_{\text{VAE}} = \underbrace{\text{KL}(q^*(z|x) || p(z|x))}_{\text{Approximation}} + \underbrace{\text{KL}(q(z|x) || p(z|x)) - \text{KL}(q^*(z|x) || p(z|x))}_{\text{Amortization}}. \quad (4.3)$$

4.3.2 Flexible Approximate Posteriors

Our experiments involve expressive approximations which use flow transformations and auxiliary variables. The flow transformation that we employ is of the same type as the transformations of Real NVP (Dinh et al., 2017). We partition the latent variable z into two, z_1 and z_2 , then perform the following transformations:

$$z'_1 = z_1 \circ \sigma_1(z_2) + \mu_1(z_2) \tag{4.4}$$

$$z'_2 = z_2 \circ \sigma_2(z'_1) + \mu_2(z'_1) \tag{4.5}$$

where $\sigma_1, \sigma_2, \mu_1, \mu_2 : \mathbb{R}^n \rightarrow \mathbb{R}^n$ are differentiable mappings parameterized by neural nets and \circ takes the Hadamard or element-wise product. We partition the latent variable by simply indexing the elements of the first half and the second half. The determinant of the combined transformation’s Jacobian, $|\det(\frac{\partial z'}{\partial z})|$, can be easily evaluated. See section A.2.3 of the supplementary material for a derivation. The lower bound of this approximation is the same flow bound as Eqn. (2.21). We refer to this approximation as q_{Flow} .

We also experiment with an approximation that combines flow transformations and auxiliary variables. Let $z \in \mathbb{R}^n$ be the variable of interest and $v \in \mathbb{R}^n$ the auxiliary variable. The flow is the same as equations (4.4) and (4.5), where z_1 is replaced with z and z_2 with v . We refer to this approximate distribution as q_{AF} , where AF stands for auxiliary flow. We train this model by optimizing the following bound:

$$\begin{aligned} & \mathbb{E}_{q_0(z,v|x)} \left[\log \left(\frac{p(x, z_T)r(v_T|x, z_T)}{q_T(z_T, v_T|x) \left| \det \left(\frac{\partial z_t v_t}{\partial z_{t-1} v_{t-1}} \right) \right|^{-1}} \right) \right] \\ & = \mathcal{L}[q_{AF}]. \end{aligned} \tag{4.6}$$

We refer readers to Section A.2 in the Supplementary material for specific details of the flow configuration adopted in the experiments.

4.3.3 Marginal Log-Likelihood Estimation and Evidence Lower Bounds

In this section, we describe the estimates we use to compute the bounds of the inference gaps: $\log p(x)$, $\mathcal{L}[q^*]$, and $\mathcal{L}[q]$. We use two bounds to estimate the marginal log-likelihood, $\log p(x)$: IWAE (Burda et al., 2016) and AIS (Neal, 2001).

As explained in section 2.2.1, the IWAE bound takes multiple importance weighted samples from the variational q distribution resulting in a tighter lower bound than the VAE bound. The IWAE bound is computed as:

$$\begin{aligned} \log p(x) &\geq \mathbb{E}_{z_1, \dots, z_k \sim q(z|x)} \left[\log \left(\frac{1}{k} \sum_{i=1}^k \frac{p(x, z_i)}{q(z_i|x)} \right) \right] \\ &= \mathcal{L}_{\text{IWAE}}[q]. \end{aligned} \tag{4.7}$$

As the number of importance samples approaches infinity, the bound approaches the marginal log-likelihood. It is often used as an evaluation metric for generative models (Burda et al., 2016; Kingma et al., 2016). AIS is potentially an even tighter lower bound. AIS weights samples from distributions which are sequentially annealed from an initial proposal distribution to the true posterior. See Section A.2.4 in the Supplementary material for further details regarding AIS. To compute the AIS bound, we use 100 chains, each with 10000 intermediate distributions, where each transition consists of one HMC trajectory with 10 leapfrog steps. The initial distribution for AIS is the prior, so that it is encoder-independent.

We estimate the marginal log-likelihood by independently computing our tightest lower bounds then take the maximum of the two:

$$\log \hat{p}(x) = \max(\mathcal{L}_{\text{AIS}}, \mathcal{L}_{\text{IWAE}}).$$

The $\mathcal{L}[q^*]$ and $\mathcal{L}[q]$ bounds are the standard ELBOs, \mathcal{L}_{VAE} , from Eqn. (4.2), computed with either the amortized q or the optimal q^* (see below). When computing \mathcal{L}_{VAE} and $\mathcal{L}_{\text{IWAE}}$, we use 5000 samples.

4.3.4 Local Optimization of the Approximate Distribution

To compute $\mathcal{L}_{\text{VAE}}[q^*]$, we optimize the parameters of the variational distribution for every datapoint. For the local optimization of the fully factorized approximate posterior (q_{FFG}), we initialize the mean and variance as the prior, i.e. $\mathcal{N}(0, I)$. We optimize the mean and variance using the Adam optimizer with a learning rate of 10^{-3} . To determine convergence, after every 100 optimization steps, we compute the average of the previous 100 ELBO values and compare it to the best achieved average. If it does not improve for 10 consecutive iterations then the optimization is terminated. For the more complex approximate posteriors (q_{Flow} and q_{AF}), the same process is used to optimize all of its parameters. All neural nets for the flow were initialized with a variant of the Xavier initialization (Glorot and Bengio, 2010). We use 100 Monte Carlo samples when computing the ELBO to reduce variance.

4.3.5 Validation of Bounds

The soundness of our empirical analysis depends on the reliability of the marginal log-likelihood estimator. For general importance sampling based estimators, the sample variance of the normalized importance weights can serve as an indicator of accuracy (Geweke, 1989; Neal, 2001). This quantitative measure, however, can also be unreliable, e.g. when the proposal misses an important mode of the target distribution (Neal, 2001).

In this work, we follow Wu et al. (2017) to empirically validate our AIS estimates with Bidirectional Monte Carlo (BDMC, Grosse et al. (2015, 2016)). In addition to a lower bound provided by AIS, BDMC runs AIS chains backward from exact posterior samples to obtain an upper bound on the marginal log-likelihood. It should be noted that BDMC relies on the assumption that the distribution of the simulated data from the model roughly matches that of the real data. This is because the backward chain initializes from exact posterior samples (Grosse et al., 2015).

For the MNIST and Fashion datasets, BDMC gives a gap within 0.1 nat for a linear schedule AIS with 10^4 intermediate distributions and 100 importance samples on 10^3 simulated datapoints. For 3-BIT CIFAR, the same AIS setting gives a gap within 1 nat with the sigmoidial annealing schedule (Grosse et al., 2015) on 100 simulated datapoints. Loosely

speaking, this should give us confidence in how well our AIS lower bounds reflect the marginal log-likelihood computed on the real data.

4.4 Related Work

Much of the earlier work on variational inference focused on optimizing the variational parameters locally for each datapoint, e.g. the original Stochastic Variational Inference scheme (SVI, Hoffman et al. (2013)). To scale inference to large datasets, most related works utilize inference networks to amortize the cost of inference over the entire dataset. Our work analyses the error that these inference networks introduce.

Most relevant to our work is the recent work of Krishnan et al. (2018), which explicitly remarks on two sources of error in variational learning with inference networks, and proposes to optimize approximate inference locally from an initialization output by the inference network. They show improved training on high-dimensional, sparse data with the hybrid method, claiming that local optimization reduces the negative effects of random initialization in the inference network early on in training. Thus their work focuses on reducing the amortization gap early on in training. Similar to this idea, Hoffman (2017) proposes to perform approximate inference during model training with MCMC at an initialization given by a variational distribution. Our work provides a means of explaining these improvements in terms of the sources of inference suboptimality that they reduce.

4.5 Experimental Results

4.5.1 Intuition through Visualization

To begin, we would like to gain an intuitive visualization of the gaps presented in Section 4.3.1. To this end, we trained a VAE with a factorized Gaussian approximation and a two-dimensional latent space on MNIST. In Fig. 4.2 we show contour plots of various distributions in the latent space. The first row contains contour plots of the true posteriors $p(z|x)$ for four different training datapoints (columns A, B, C, D). See section A.2 for details regarding plotting $p(z|x)$. We selected four datapoint examples to highlight different inference

phenomena. The amortized factorized Gaussian refers to the output of the recognition net, in this case, a factorized Gaussian approximation. Optimized Factorized Gaussian is a Gaussian that we’ve optimized to fit the posterior of the datapoint. Similarly, Optimized Flow is a flow distribution that has been fit to the posterior distribution for each datapoint.

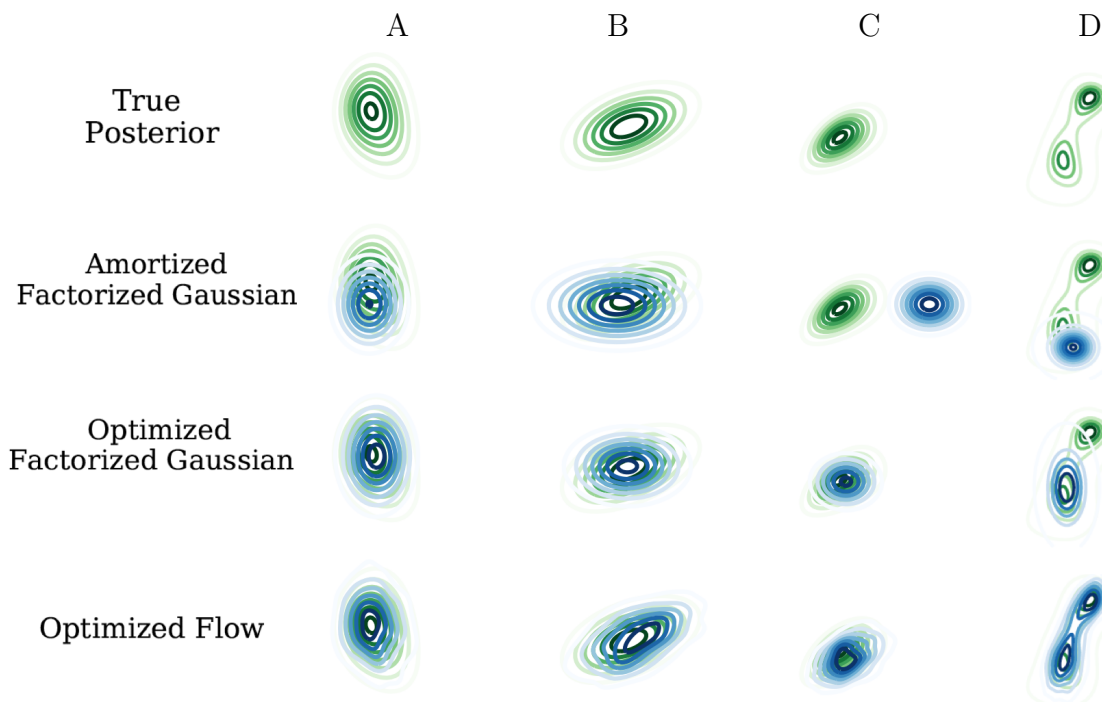


Figure 4.2: True Posterior and Approximate Distributions of a VAE with 2D latent space. The columns represent four different datapoints. The green distributions are the true posterior distributions, highlighting the mismatch with the blue approximations. Amortized: Variational parameters learned over the entire dataset. Optimized: Variational parameters optimized for each individual datapoint. Flow: Using a flexible approximate distribution, q_{AF} , defined in Eqn. 4.6.

In Fig. 4.2, Posterior A is an example of a distribution where a factorized Gaussian can fit relatively well. Posterior B is an example of a posterior with dependence between dimensions, demonstrating the limitation of having a factorized approximation. Posterior C highlights a shortcoming of performing amortization with a limited-capacity recognition network, where the amortized factorized Gaussian shares little support with the true posterior. In this case, the amortization gap would be large and the approximation gap would be low. Posterior D is a bi-modal distribution which demonstrates a scenario where the approximation gap would be relatively high and it demonstrates the ability of the flexible approximation to fit to complex distributions, in contrast to the simple Gaussian approximation. These observations raise the

	MNIST		Fashion-MNIST		3-BIT CIFAR	
	q_{FFG}	q_{AF}	q_{FFG}	q_{AF}	q_{FFG}	q_{AF}
$\log \hat{p}(x)$	-89.80	-88.94	-97.47	-97.41	-816.9	-820.56
$\mathcal{L}_{VAE}[q_{AF}^*]$	-90.80	-90.38	-98.92	-99.10	-820.19	-822.16
$\mathcal{L}_{VAE}[q_{FFG}^*]$	-91.23	-113.54	-100.53	-132.46	-831.65	-861.62
$\mathcal{L}_{VAE}[q]$	-92.57	-91.79	-104.75	-103.76	-869.12	-864.28
Approximation Gap	1.43	1.44	3.06	1.69	14.75	1.60
Amortization Gap	1.34	1.41	4.22	4.66	37.47	42.12
Inference Gap	2.77	2.85	7.28	6.35	52.22	43.72

Table 4.2: Inference Gaps. The columns q_{FFG} and q_{AF} refer to the variational distribution used for training the model. These lower bounds are computed on the training set and are in units of nats.

following question: in more typical VAEs, is the amortization of inference the leading cause of the distribution mismatch, or is it the limited expressiveness of the approximation?

4.5.2 Amortization vs Approximation Gap

In this section, we compare how much the approximation and amortization gaps each contribute to the total inference gap. The VAE models for the experiments on MNIST and Fashion-MNIST have the same architecture and we use similar models as is used in [Burda et al. \(2016\)](#). The encoder has two hidden layers with 200 units each. The generator is the reverse of the encoder. For the CIFAR experiments, the inference network consists of three 4 by 4 convolution layers, then a fully-connected layer outputs the 50-dimensional mean and log-variance of the latent variable. Similarly, the generator is the reverse of the encoder but where we use deconvolutional layers instead of convolutions. The activation function is chosen to be the exponential linear unit (ELU, [Clevert et al. \(2016\)](#)), as we observe improved performance compared to tanh. The latent space has 50 dimensions. We follow the same learning rate schedule and train for the same amount of epochs as described by [Burda et al. \(2016\)](#). All models are trained with a batch-size of 100 with ADAM. For more architectural details, see section A.2.

Table 4.2 are results of inference on the training set of MNIST, Fashion-MNIST and 3-BIT CIFAR (a binarized version of CIFAR-10, see Section A.2 for details). For each dataset, we trained models with two different approximate posterior distributions: a fully-

factorized Gaussian, q_{FFG} , and the flexible distribution, q_{AF} . Due to the computational cost of optimizing the local parameters for each datapoint, our evaluation is performed on a subset of 1000 datapoints for MNIST and Fashion-MNIST and a subset of 100 datapoints for 3-BIT CIFAR.

For MNIST, we see that the amortization and approximation gaps each account for nearly half of the inference gap. On the more difficult Fashion-MNIST dataset, the amortization gap is larger than the approximation gap. For CIFAR, we see that the amortization gap is much more significant compared to the approximation gap. Thus, for the three datasets and model architectures that we consider, the amortization gap is likely to be the more prominent cause of inference suboptimality, especially when the dataset becomes more challenging to model. This indicates that improvements in inference will likely be a result of reducing amortization error, rather than approximation errors.

Increased Encoder Capacity

With these results in mind, would simply increasing the capacity of the encoder improve the amortization gap? We examined this by training the MNIST and Fashion-MNIST models from above but with larger encoders. In the large encoder setting, we change the number of hidden units for the inference network to be 500, instead of 200 in the standard model. Table 4.3 are the results of this experiment. Comparing to Table 4.2, we see that, for both datasets and both variational distributions, using a larger encoder results in the inference gap decreasing and the decrease is mainly due to a reduction in the amortization gap.

	MNIST		Fashion-MNIST	
	q_{FFG}	q_{AF}	q_{FFG}	q_{AF}
$\log \hat{p}(x)$	-89.61	-88.99	-95.99	-96.18
$\mathcal{L}_{VAE}[q_{AF}^*]$	-90.65	-90.44	-97.40	-97.91
$\mathcal{L}_{VAE}[q_{FFG}^*]$	-91.07	-108.71	-99.64	-129.70
$\mathcal{L}_{VAE}[q]$	-92.18	-91.19	-102.73	-101.67
Approximation Gap	1.46	1.45	3.65	1.73
Amortization Gap	1.11	0.75	3.09	3.76
Inference Gap	2.56	2.20	6.74	5.49

Table 4.3: Larger Encoder results. The columns q_{FFG} and q_{AF} refer to the variational distribution used for training the model. The lower bounds are computed on the training set and are in units of nats.

4.5.3 Influence of Flows on the Amortization Gap

The common reasoning for increasing the expressiveness of the approximate posterior is to minimize the difference between the true and approximate distributions, i.e. reduce the approximation gap. However, given that the expressive approximation is often accompanied by many additional parameters, we would like to know how much influence it has on the amortization error.

To investigate this, we trained a VAE on MNIST, discarded the encoder, then retrained encoders with different approximate distributions on the fixed decoder. We fixed the decoder so that the true posterior is constant for all the retrained encoders. The initial encoder was a two-layer MLP with a factorized Gaussian distribution. In order to emphasize a large amortization gap, the retrained encoders had no hidden layers (ie. just linear transformations). For the retrained encoders, we tested three approximate distributions: fully factorized Gaussian (q_{FFG}), auxiliary flow (q_{AF}), and Flow (q_{Flow}). See Section 4.3.2 for the details of these distributions.

Variational Family	q_{FFG}	q_{AF}	q_{Flow}
$\log \hat{p}(x)$	-84.70	-84.70	-84.70
$\mathcal{L}_{VAE}[q^*]$	-86.61	-85.48	-85.13
$\mathcal{L}_{VAE}[q]$	-129.83	-98.58	-97.99
Approximation Gap	1.91	0.78	0.43
Amortization Gap	43.22	13.10	12.86
Inference Gap	45.13	13.88	13.29

Table 4.4: Influence of expressive approximations on the amortization gap. The parameters used to increase the flexibility of the approximate distribution also reduce the amortization gap.

The inference gaps of the retrained encoders on the training set are shown in Table 4.4. As expected, we observe that the small encoder with q_{FFG} has a very large amortization gap. However, when we use q_{AF} or q_{Flow} as the approximate distribution, we see the approximation gap decrease, but more importantly, there is a significant decrease in the amortization gap. This indicates that the parameters used for increasing the complexity of the approximation also play a large role in diminishing the amortization error.

These results are expected given that the parameterization of the Flow distribution can be

interpreted as an instance of the RevNet (Gomez et al., 2017) which has demonstrated that Real-NVP transformations (Dinh et al., 2017) can model complex functions similar to typical MLPs. Thus the flow transformations we employ should also be expected to increase the expressiveness of the approximation while also increasing the capacity of the encoder. The implication of this observation is that models which improve the flexibility of their variational approximation, and attribute their improved results to the increased expressiveness, may have actually been due to the reduction in amortization error.

4.5.4 Influence of Approximate Posterior on True Posterior

To what extent does the posterior approximation affect the learned model? Turner and Sahani (2011) studied the biases in parameter learning induced by the variational approximation when learning via variational Expectation-Maximization. Similarly, we ask whether a factorized Gaussian approximation causes the true posterior to be more like a factorized Gaussian? Burda et al. (2016) visually demonstrate that when trained with an importance-weighted approximate posterior, the resulting true posterior is more complex than those trained with factorized Gaussian approximations. Just as it is hard to evaluate a generative model by visually inspecting samples, it is hard to judge how “Gaussian” the true posterior is by visual inspection. We can quantitatively determine how close the posterior is to a fully-factorized Gaussian (FFG) by comparing the marginal log-likelihood estimate $\log \hat{p}(x)$ and the Optimal FFG bound $\mathcal{L}_{\text{VAE}}[q_{\text{FFG}}^*]$. This is equivalent to estimating the KL divergence between the optimal Gaussian and the true posterior, $\text{KL}(q^*(z|x)||p(z|x))$.

In Table 4.2, the results on MNIST for the FFG trained model show that $\text{KL}(q^*(z|x)||p(z|x))$ is nearly the same for both q_{FFG}^* and q_{AF}^* . In contrast, on the model trained with q_{AF} , $\text{KL}(q^*(z|x)||p(z|x))$ is much larger for q_{FFG}^* than q_{AF}^* . This suggests that the true posterior of a FFG-trained model is closer to FFG than the true posterior of the Flow-trained model. The same observation can be made on the Fashion-MNIST dataset. This implies that the decoder can learn to have a true posterior that fits better to the approximation.

These observations justify our results of Section 4.5.2. which showed that the amortization error is often the main cause of inference suboptimality. One reason for this is that the generator accommodates the choice of approximation, thus reducing the approximation error.

Increased Decoder Capacity

Given that we have seen that the generator can accommodate the choice of approximation, our next question is whether a generator with more capacity increases its ability to fit to the approximation. To this end, we trained VAEs with decoders of different sizes and measured the approximation gaps on the training set. Specifically, we trained decoders with 0, 2, and 4 hidden layers on MNIST. The encoder of each model was the same size: 2 hidden layers. See Table 4.5 for the results. We see that as the capacity of the decoder increases, the approximation gap decreases. This result suggests that the more flexible the generator is, the less flexible the approximate distribution needs to be to ensure accurate inference.

Generator Hidden Layers	0	2	4
$\log \hat{p}(x)$	-100.52	-84.78	-82.19
$\mathcal{L}_{\text{VAE}}[q_{FG}^*]$	-104.42	-86.61	-83.82
Approximation Gap	3.90	1.83	1.63

Table 4.5: Increased decoder capacity reduces the approximation gap.

4.5.5 Generalization of Amortized Inference on Held-Out Data

How well does amortized inference generalize at test time? We address this question by visualizing the gaps on training and validation datapoints across the training epochs. In Fig. 4.3, the models are trained on 50000 binarized Fashion-MNIST datapoints and the gaps are computed on a subset of a 100 training and validation datapoints. The top and bottom boundaries of the blue region represent $\log \hat{p}(x)$ and $\mathcal{L}[q^*]$. The bottom boundary of the orange region represents $\mathcal{L}[q]$. In other words, the blue region is the approximation gap and the orange is the amortization gap.

In Fig. 4.3, the ‘Gaussian Approximate Posterior’ model (top left) is a VAE with latent size 20 and the encoder and decoder both have two hidden layers each consisting of 200 hidden units. The ‘Flow Approximate Posterior’ model (top right) augments the previous model with a q_{Flow} variational distribution. ‘Larger Decoder’ and ‘Larger Encoder’ models have factorized Gaussian approximate posteriors and increase the number of hidden layers to three and the number of units in each layer to 500.

Firstly, we observe that for all models, the approximation gap on the training and

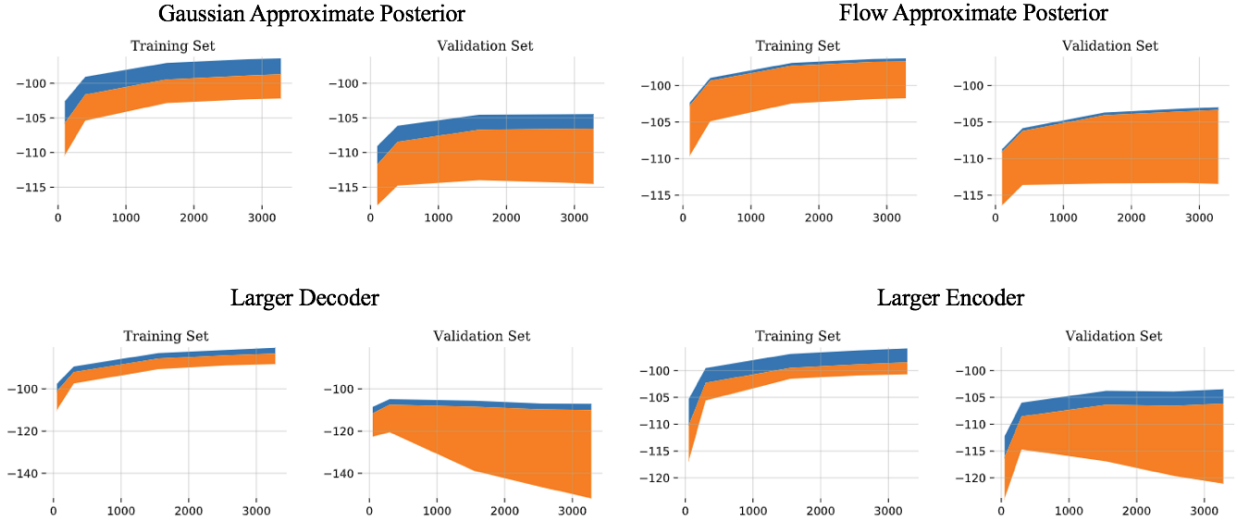


Figure 4.3: Inference gaps over epochs trained on binarized Fashion-MNIST. Blue is the approximation gap. Orange is the amortization gap. Standard is a VAE with FFG approximation. Flow is a VAE with a Flow approximation.

validation sets are roughly equivalent. This indicates that the true posteriors of the held-out data are similar to that of the training data. Secondly, we note that for all models, the encoder overfits more than the decoder. These observations resonate with the encoder overfitting findings by Wu et al. (2017).

How does increasing decoder capacity affect inference on held-out data? We know from Section 4.5.4 that increasing generator capacity results in a posterior that better fits the approximation making posterior inference easier. Furthermore, the Larger Decoder plot of Fig. 4.3 shows that increasing generator capacity causes the model to be more prone to overfitting. Thus, there is a tradeoff between ease of inference and decoder overfitting.

Encoder Capacity and Approximation Expressiveness

We have seen in Sections 4.5.2 and 4.5.3 that expressive approximations as well as increasing encoder capacity can lead to a reduction in the amortization gap. This leads us to the following question: when should we increase encoder capacity versus increasing the expressiveness of the approximation? We answer this question in terms of how well each model can generalize its efficient inference (recognition network and variational distribution) to held-out data.

In Fig. 4.3, we see that the Flow model and the Larger Encoder model achieve similar

	MNIST		Fashion-MNIST	
	q_{FFG}	q_{AF}	q_{FFG}	q_{AF}
$\log \hat{p}(x)$	-89.82	-89.52	-102.56	-102.88
$\mathcal{L}_{\text{VAE}}[q_{AF}^*]$	-90.96	-90.45	-103.73	-104.02
$\mathcal{L}_{\text{VAE}}[q_{FFG}^*]$	-90.84	-92.25	-103.85	-105.80
$\mathcal{L}_{\text{VAE}}[q]$	-92.33	-91.75	-106.90	-107.01
Approximation Gap	1.02	0.93	1.29	1.14
Amortization Gap	1.49	1.30	3.05	2.29
Inference Gap	2.51	2.23	4.34	4.13

Table 4.6: Models trained without entropy annealing. The columns q_{FFG} and q_{AF} refer to the variational distribution used for training the model. The lower bounds are computed on the training set and are in units of nats.

$\log \hat{p}(x)$ on the validation set at the end of training. However, we see that the $\mathcal{L}[q]$ bound of the Larger Encoder model is significantly lower than the $\mathcal{L}[q]$ bound of the Flow model due to the encoder overfitting to the training data. Although they both model the data nearly equally well, the recognition net of the Larger Encoder model is no longer suitable to perform inference on the held-out data due to overfitting. Thus a potential rationale for utilizing expressive approximations is that they improve generalization to held-out data in comparison to increasing the encoder capacity.

We highlight that, in many scenarios, efficient test time inference is not required and consequently, encoder overfitting is not an issue, since we can use non-efficient encoder-independent methods to estimate $\log p(x)$, such as AIS, IWAE with local optimization, or potentially retraining the encoder on the held-out data. In contrast, when efficient test time inference is required, encoder generalization is important and expressive approximations are likely advantageous.

4.5.6 Annealing the Entropy

Typical warm-up (Bowman et al., 2016; Sønderby et al., 2016) refers to annealing the $\text{KL}(q(z|x)||p(z))$ term of the VAE objective so that we begin by only optimizing $p(x|z)$. This can also be interpreted as performing maximum likelihood estimation (MLE) early on during training. This optimization technique is known to help prevent the latent variable from degrading to the prior (Burda et al., 2016; Sønderby et al., 2016). We employ a similar

annealing scheme during training by annealing the entropy of the approximate distribution:

$$\mathbb{E}_{z \sim q(z|x)} [\log p(x, z) - \lambda \log q(z|x)],$$

where λ is annealed from 0 to 1 over training. This can be interpreted as *maximum a posteriori* (MAP) in the initial phase of training.

We find that warm-up techniques, such as annealing the entropy, are important for allowing the true posterior to be more complex. Table 4.6 are results from a model trained without the entropy annealing schedule. Comparing these results to Table 4.2, we observe that the difference between $\mathcal{L}_{\text{VAE}}[q_{FG}^*]$ and $\mathcal{L}_{\text{VAE}}[q_{AF}^*]$ is significantly smaller without entropy annealing. This indicates that the true posterior is more Gaussian when entropy annealing is not used. This suggests that, in addition to preventing the latent variable from degrading to the prior, entropy annealing allows the true posterior to better utilize the flexibility of the expressive approximation.

4.6 Conclusion

This chapter investigated how encoder capacity, approximation choice, decoder capacity, and model optimization influence inference suboptimality in terms of the empirical approximation and amortization gaps. We discovered that the amortization gap can be a leading source to inference suboptimality and that the generator can reduce the approximation gap by learning a true posterior that fits to the choice of approximation. We showed that the parameters used to increase the expressiveness of the approximation play a role in generalizing inference rather than simply improving the complexity of the approximation. We confirmed that increasing the capacity of the encoder reduces the amortization error. Additionally, we demonstrated that optimization techniques, such as entropy annealing, help the generative model to better utilize the flexibility of expressive variational distributions. Analyzing these gaps can be useful for guiding improvements in VAEs. Future work includes evaluating other types of expressive approximations, more complex likelihood functions, and datasets.

Chapter 5

Aggregate Posteriors and the Vision-Language VAE

In the previous chapters, we investigated inference in VAE models which generated a single data modality: images. What if we wanted to generate data from two data modalities using a single latent variable? In this chapter, we study joint latent variable models of two modalities: images and text (Vision-Language VAE). A common task for these multimodal models is to perform conditional generation; for instance, generating an image conditioned on text. This can be achieved by sampling the posterior of the text then generating the image given the latent variable. However, we find that a problem with this approach is that the posterior of the text does not match the aggregate of the image posteriors corresponding to that text. The result is that the generated images are either of poor quality or don't match the text. A similar problem is also encountered in the mismatch between the prior and the marginal aggregate posterior. In this chapter, we highlight the importance of learning aggregate posteriors when faced with these types of distribution mismatches. We demonstrate this on modified versions of the CLEVR and CelebA datasets.

This chapter comes from the work done in [Cremer and Kushman \(2018\)](#), which was a collaboration between myself and Nate Kushman. In that work, I produced the results and both of us wrote the paper.

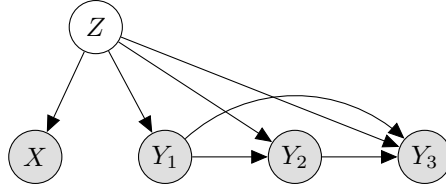


Figure 5.1: Graphical model of a Vision-Language VAE, where X represents images, Y represents text, and Z represents the latent variable. We use three words (Y_1, Y_2, Y_3) to represent the auto-regressive text decoder, but in practice the model can generalize to any number of words.

5.1 Introduction

In this chapter, we are interested in learning distributions over multiple data modalities. We can form a joint distribution over the modalities by introducing a latent variable. By marginalizing out the latent variable, we aim to maximize the following joint probability:

$$p(x, y) = \int p(x, y, z) dz \quad (5.1)$$

In this work, we will be exploring latent variable models of images and text. We combine both modalities by utilizing a single latent variable for both the text and images to share. We refer to our VAE that generates both images and text as a Vision-Language VAE (VLVAE). This model includes two decoders: one for images and one for text, where the text decoder is auto-regressive. See Fig. 5.1 for the graphical model of a VLVAE, where X represents images, Y represents text, and Z represents the latent variable. The text Y is represented as a sequence of words (not characters) and the vocabulary consists of a set of words for the specific task.

With a joint model such as the above VLVAE, we can perform a number of tasks, such as generation and density estimation of: images $p(x)$, text $p(y)$, image-text pairs $p(x, y)$, and conditional generation and density estimation of images given text $p(x|y)$ and text given images $p(y|x)$. This chapter explores a number of these tasks.

Through our experiments, we made an important observation regarding the inference for conditional generation. We found that, in practice, there is a mismatch between the text posterior and the aggregate of the image posteriors corresponding to the same text. In other

words, the problem is that

$$p(z|y) \neq \mathbb{E}_{p_D(x|y)} [p(z|x)],$$

where $p_D(x|y)$ is the data distribution of images x for a given text y . The reason that this is a significant problem is that if we sample the posterior for some text $p(z|y)$, then generate images with $p(x|z)$, these images will potentially be either of poor quality or unrelated to y .

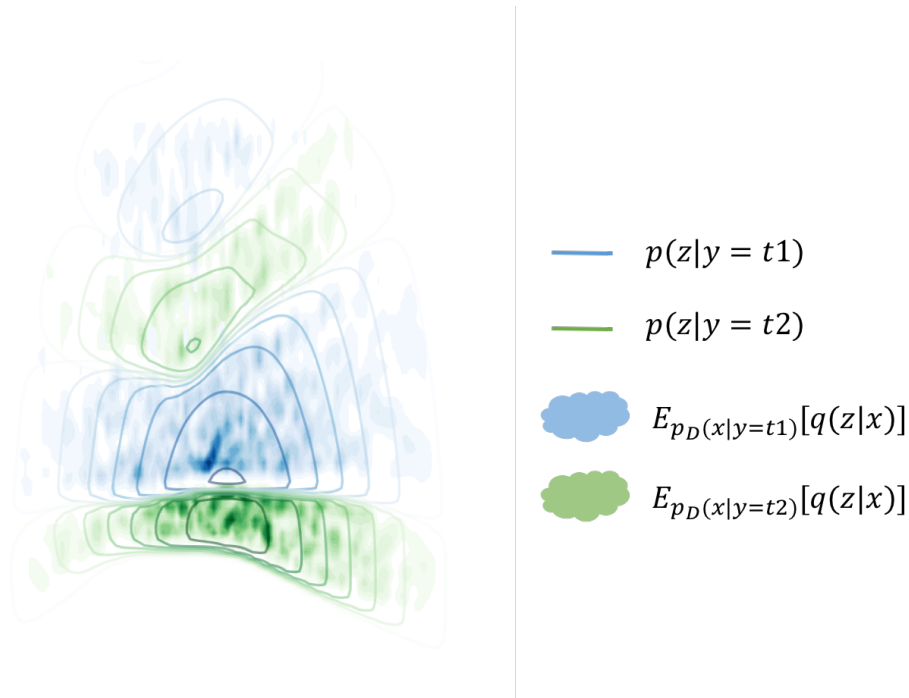


Figure 5.2: 2D Visualization of the distribution mismatch between the posterior for a text statement and the aggregate posterior of the corresponding images. The lines show the level sets of the text posterior, $p(z|y)$, for two different text statements, one in blue and the other in green. The shading represents the approximate aggregate posterior of the images corresponding to the text, $\mathbb{E}_{p_D(x|y)} [q(z|x)]$.

Fig. 5.2 demonstrates this mismatch by visualizing the posterior of two different texts, $t1$ and $t2$, and the aggregate of the image approximate posteriors corresponding to those texts. The VLVAE model was trained with a latent size of 2 dimensions so that it could be visualized. The contour lines represent level sets of density for the text posteriors. The shading areas represent the aggregate of the image posteriors corresponding to those texts. We can see that these distributions are not equivalent and that there are many holes in the aggregate posterior, which means that if we sample $p(z|y)$, then we could be sampling a z that has low probability under the aggregate posterior, resulting in images which do not

match the given text or images of poor quality. See section A.3.4 for more details regarding Fig. 5.2.

This problem is very similar to the prior mismatch problem: $p(z) \neq \mathbb{E}_{p_D(x)} [p(z|x)]$. There have been steps towards improving this problem by either learning the prior during training (Tomczak and Welling, 2018) or learning more flexible posteriors (Rezende and Mohamed, 2015; Makhzani et al., 2015; van den Berg et al., 2018; Takahashi et al., 2018). In this chapter, we highlight this distribution mismatch problem in multimodal latent variable models and explore the benefit of learning aggregate posteriors with flow distributions. We also relate this approach to other multimodal VAE objectives, such as JMVAE (Suzuki et al., 2016) and TELBO (Vedantam et al., 2018).

Datasets

We perform experiments on two datasets. The first dataset has images of two objects and a corresponding text which describes the image. We call this dataset Two Object CLEVR, because its a modified version of the CLEVR dataset (Johnson et al., 2017). The images have dimensions: [112,112,3]. The text consists of nine words, where the first four words describe one object and the last four words describe the second object. The middle word describes the spatial relationship of the two objects (ex: right, left, front, back). The second dataset is the CelebA dataset (Liu et al., 2015), which contains images of faces and a sequence of words describing the face. See Fig. 5.3 for an example from each dataset and see section A.3.2 for more details regarding the datasets. We chose to use these simple datasets in order to quickly test and iterate new models as well as to be able to more easily understand the mistakes made by the models.

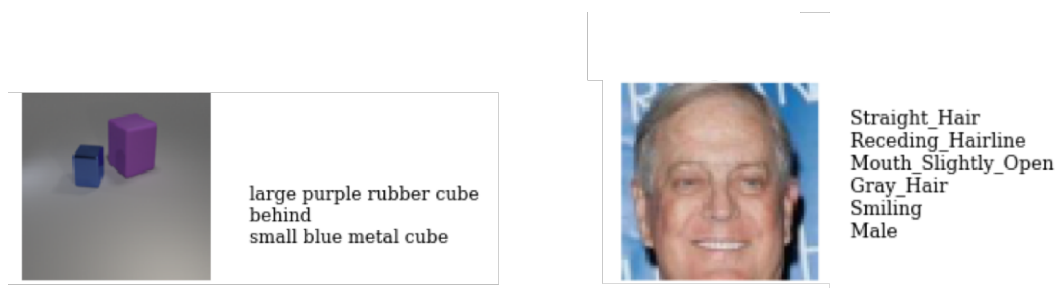


Figure 5.3: Example datapoints from Two Object CLEVR (left) and CelebA (right).

5.2 Vision-Language VAE

In this work, we learn the VLVAE model of Fig. 5.1 by maximizing the following lower bound of the joint log-likelihood:

$$\log p(x, y) \geq \mathbb{E}_{q(z|x)} \left[\log \frac{p(x|z)p(y|z)p(z)}{q(z|x)} \right] \quad (5.2)$$

$$= \mathbb{E}_{q(z|x)} \left[\log \frac{p(x|z)p(z)}{q(z|x)} \right] + \mathbb{E}_{q(z|x)} [\log p(y|z)] \quad (5.3)$$

Its important to note that the text decoder $p(y|z)$ is auto-regressive: each generated word is conditioned on the previously generated words and the latent variable z . We chose this because of the success that auto-regressive models have seen on text data. Notice also that we perform inference using only images $q(z|x)$. From Eqn. 5.3 we see that the objective is the standard ELBO for the images and an image-encoder text-decoder loss for the text. In the following sections, we will explore the significance of this objective.

Model Architecture Overview

There are three main components to the VLVAE architecture: image encoder $q(z|x)$, image decoder $p(x|z)$, and text decoder $p(y|z)$. The image encoder has a ResNet architecture and it outputs the parameters of the variational distribution. For simplicity, we use a factorized Gaussian for $q(z|x)$, thus the encoder outputs the means and variances for each dimension of the latent variable z . The latent variable has 50 dimensions and the prior $p(z)$ is a $N(0, I)$ distribution. The image decoder has the inverse architecture of the image encoder and it outputs a Beta distribution for $p(x|z)$. The text decoder is an RNN with GRU units, which outputs a Categorical distribution for each word. See section A.3.1 for more details about the model and its implementation.

5.2.1 Inference with Only Images

Here we will explain how we train the model using only images for inference and how this affects the model. By optimizing the ELBO of Eqn. 5.2, we are minimizing the following KL

divergence:

$$\text{KL}(q_\phi(z|x)||p(z|x,y)), \quad (5.4)$$

which is the KL divergence between the approximate posterior and the true posterior. If the true posterior $p(z|x,y)$ depends on y , then $q_\phi(z|x)$ will obviously be suboptimal for modelling the posterior, since it does not depend on y . However, if the model learns not to store information specific to the text in the latent variable, then $p(z|x,y) = p(z|x)$ since the posterior is conditionally independent of the text given the image. If this is the case, then $q_\phi(z|x)$ will have the potential to accurately model the true posterior.

Thus, by training the model with inference from images only, the model is incentivized to have the true posterior be independent of text-specific information. How can $p(z|x,y)$ become more independent of y ? It can do this by modelling the variability in y through the auto-regressive text decoder instead of using the latent variable.

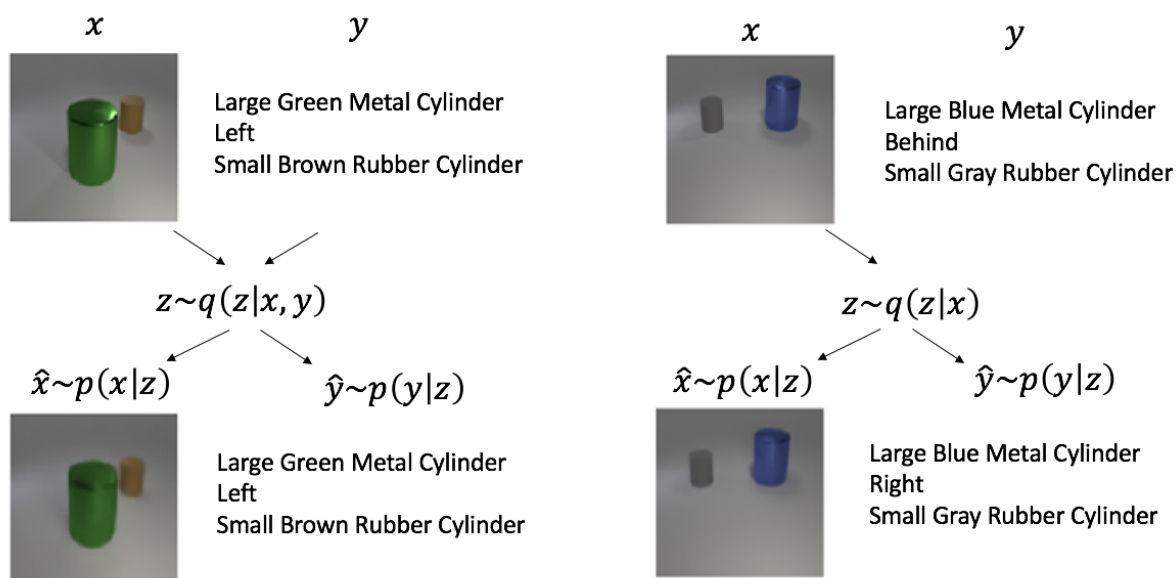


Figure 5.4: Comparison between inference with both modalities $q(z|x,y)$ (left) and inference with images only $q(z|x)$ (right). The model trained with $q(z|x,y)$ learns to store text-specific information in the latent variable, whereas the model trained with $q(z|x)$ uses the auto-regressive text decoder to model the variance in the text.

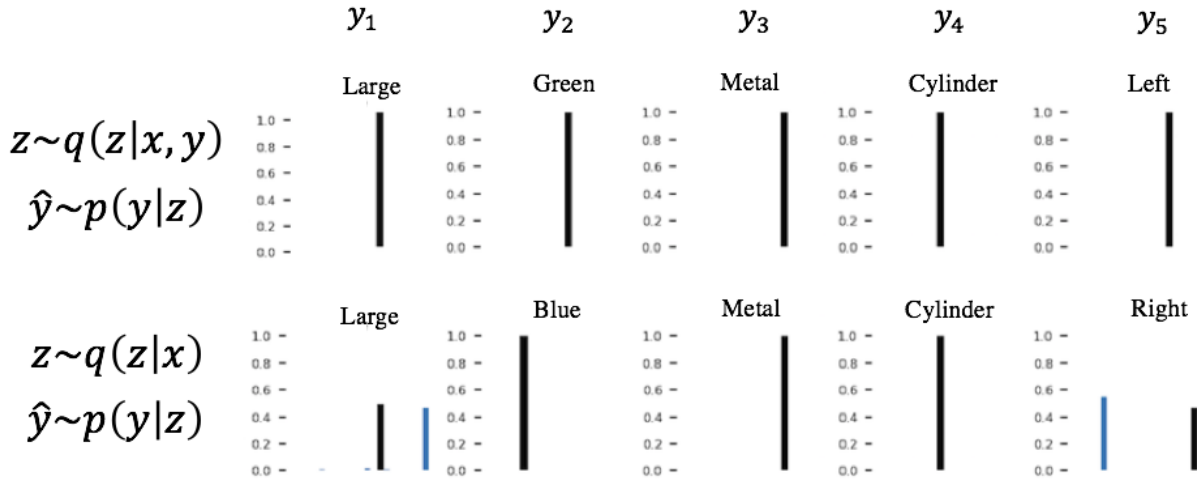


Figure 5.5: Decoding distribution of the first five words of the text from Fig. 5.4. Each word has a histogram over the entire vocabulary of the text (19 words, described in section A.3.2). Top row is from a model trained with $q(z|x, y)$ and bottom row is from a model trained with $q(z|x)$. Black bar indicates sampled word whereas blue bar indicates non-sampled words.

Comparison of image-only and image-text inference

To see the difference between inference with images only $q(z|x)$ and inference with both modalities $q(z|x, y)$, we trained two VLVAEs using each type of inference. Fig. 5.4 shows image and text samples from each model. On the left is a model trained with $q(z|x, y)$. We can see that the text reconstruction is identical to the text input. In contrast, the model trained with $q(z|x)$ (right) changes the word ‘Behind’ to ‘Right’ when reconstructing, which are both valid relational words for this image. This is an indication that the text-specific information is not contained in the latent variable and is instead modelled with the auto-regressive text decoder.

To better understand the changes in text reconstruction, we can observe the histograms over the vocabulary for each word sampled from the model. The top row of Fig. 5.5 shows the histograms for the first five words sampled from the $q(z|x, y)$ trained model, and we see that there is no uncertainty in these distributions. In comparison, for the $q(z|x)$ trained model there is uncertainty in both the first word and the relational word. The uncertainty in the first word comes from the ambiguity of which object is being referred to first. The uncertainty in the relational word is due to there being two valid words describing the relation

between the objects.

These comparisons show how changing the information given for inference can cause changes in model behaviour: in this case, the auto-regressive text decoder models the uncertainty in the word sequence instead of using the latent variable.

See Fig. 5.6 for another demonstration of the text distribution produced by the RNN decoder when the model is trained with image-only inference. This figure shows a model trained on the CelebA dataset, where there is substantial uncertainty in the ordering of words given an image. This uncertainty is reflected in the histograms the RNN outputs for each word.

Why perform inference with only images?

There are a few reasons we chose to use only images for approximate inference. Firstly, training with image-only inference reduces the ambiguity on how the text-specific information is modelled. If we used $q(z|x, y)$ to train the model, then both the latent variable and the RNN could model the text. However, when using image-only inference, it forces the auto-regressive text decoder to model the text-specific information. The text-specific information includes the syntax and the ordering of the words. This is similar in spirit to the work of [Chen et al. \(2016\)](#), where they hinder the auto-regressive decoder such that it can only model local image features (ex: fine details) so that the latent variable represents global features (ex: image label). In our case, we limit the information provided to the inference network so that the auto-regressive decoder must model what the latent variable misses.

Another reason is that it's one fewer network that needs to be trained. We only need to train $q(z|x)$ and $q(z|y)$ in order to perform inference for images and text. If we performed inference with $q(z|x, y)$ then that would be an extra network to train.

A final reason to use inference with only images is to help 'entangle' the shared concepts in the image and text. For example, the blue pixels in the image and the word 'blue' represent the same information, yet if we perform inference with $q(z|x, y)$, they could end up encoded separately within the latent space. However, when using $q(z|x)$ for inference, the concepts are forced to share the same representation.

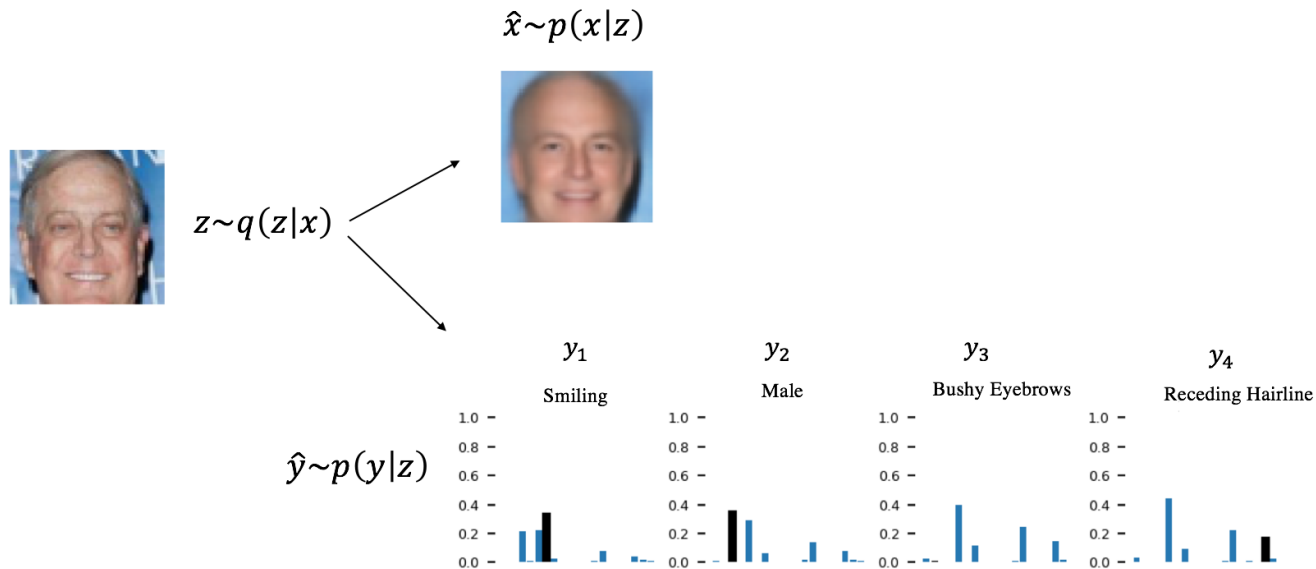


Figure 5.6: Sample from a VLVAE model on CelebA. The image on the left is an image from the dataset. Using this image, we sample from the approximate distribution $q(z|x)$, then generate an image \hat{x} and text \hat{y} . The blue bars are the text decoding histogram for each word. The black bar indicates the sampled word which is also printed above the histogram. We can see that since the latent space does not contain information specific to the text, there is uncertainty in the ordering of the attributes.

5.2.2 Conditional Generation

To generate images conditioned on text, we must learn to perform inference with text. To perform approximate inference with only text, we can learn $q_\phi(z|y)$ such that it approximates the true posterior $p(z|y)$. This is achieved by optimizing the following lower bound:

$$\log p(y) \geq \mathbb{E}_{q_\phi(z|y)} \left[\log \left(\frac{p(y|z)p(z)}{q_\phi(z|y)} \right) \right] \quad (5.5)$$

Importantly, when optimizing this lower bound, optimization is done wrt ϕ only, not the model parameters of $p(y|z)$, which are trained with Eqn. 5.2.

Now that we've trained $q(z|x)$, $q(z|y)$, $p(x|z)$, and $p(y|z)$, with Eqns. 5.2 and 5.5 we can generate conditionally. For instance, to generate images given text, we can infer and sample $z \sim q(z|y)$, then generate images with $p(x|z)$.

See Fig. 5.7 for example image samples conditioned on text. As can be seen, the images

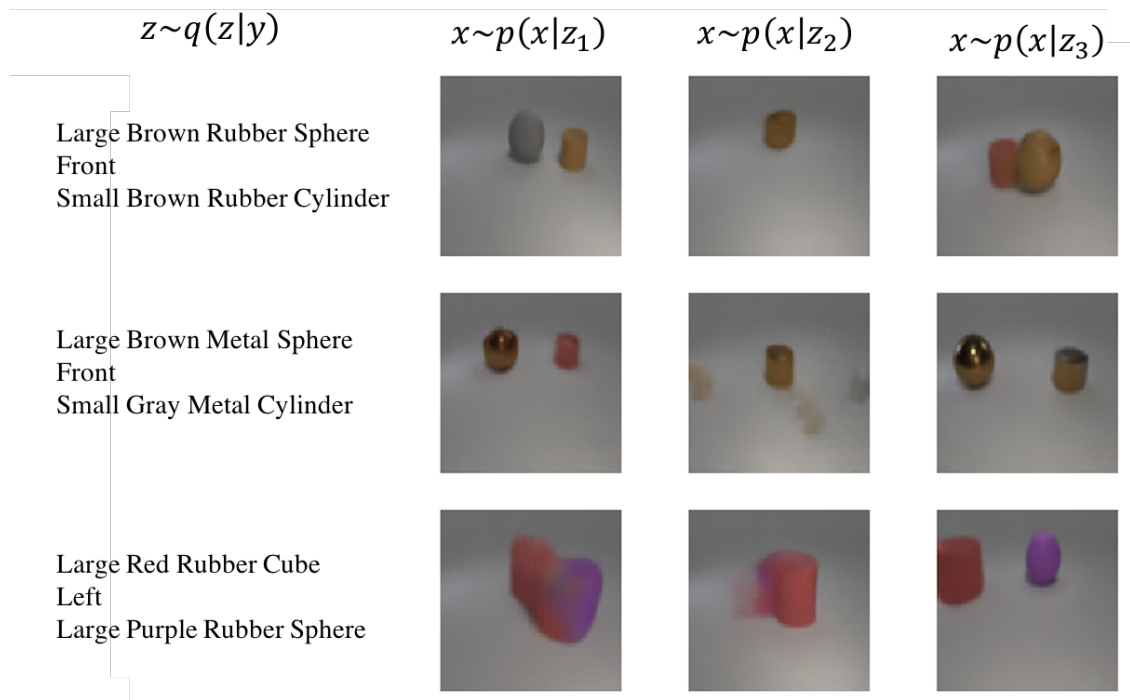


Figure 5.7: Example conditional samples $p(x|y)$ where $q(z|y)$ is trained to approximate the true posterior with Eqn. 5.5. The images are conditioned on the text on the left. There are three latent variable samples (z_1, z_2, z_3) per text. As can be seen, the images are of poor quality, which motivates the use of learning the aggregate posterior.

generated from latent variable samples from $q_\phi(z|y)$ are of poor quality. We hypothesize that this problem is due to the mismatch between the aggregate of the image posteriors and the text posterior. One source of this mismatch comes from using multiple decoders to generate the multiple modalities. Since the decoders share the same latent variable and the decoders do not generalize in the same way, this causes their posteriors to differ resulting in this distribution mismatch problem.

The text inference network $q(z|y)$ is trained with Eqn. 5.5, thus it is minimizing the following divergence:

$$\text{KL}(q_\phi(z|y) || p_\theta(z|y)), \quad (5.6)$$

which is the approximate posterior trying to match the true posterior of the text. Instead, we think it should approximate the aggregate of the image posteriors for that text. In the next section, we will investigate this problem.

5.3 Learning the Aggregate Posterior

Rather than optimizing $q_\phi(z|y)$ to approximate the true posterior $p(z|y)$, we can train $q_\phi(z|y)$ to approximate the aggregate of the image posteriors corresponding to the same y : $\mathbb{E}_{p_D(x|y)} [p(z|x)]$, where $p_D(x|y)$ is the dataset of images x with the corresponding text y . Since we don't have access to the true posterior $p(z|x)$, we will instead model the aggregate of the image *approximate* posteriors:

$$q^y(z) = \mathbb{E}_{p_D(x|y)} [q(z|x)] = \frac{1}{N} \sum_{i=1}^N q(z|x_i), x_i \sim p_D(x|y) \quad (5.7)$$

We can learn to approximate $q^y(z)$ with $q_\phi(z|y)$ by minimizing the following loss wrt ϕ :

$$L_{Agg} = \mathbb{E}_{p_D(x,y)} [KL(q(z|x)||q_\phi(z|y))] \quad (5.8)$$

To see that this objective does in fact approximate $q^y(z)$ with $q_\phi(z|y)$, we will decompose the loss into two divergences by introducing $q^y(z)$:

$$\begin{aligned} \mathbb{E}_{p_D(x|y)} [KL(q(z|x)||q(z|y))] &= \mathbb{E}_{p_D(x|y)} \left[\mathbb{E}_{q(z|x)} \left[\log \left(\frac{q(z|x)}{q(z|y)} \right) \right] \right] \\ &= \mathbb{E}_{p_D(x|y)} \left[\mathbb{E}_{q(z|x)} \left[\log \left(\frac{q(z|x)q^y(z)}{q(z|y)q^y(z)} \right) \right] \right] \\ &= \mathbb{E}_{p_D(x|y)} \left[\mathbb{E}_{q(z|x)} \left[\log \left(\frac{q(z|x)}{q^y(z)} \right) + \log \left(\frac{q^y(z)}{q(z|y)} \right) \right] \right] \\ &= \mathbb{E}_{p_D(x|y)} [KL(q(z|x)||q^y(z))] + \mathbb{E}_{p_D(x|y)q(z|x)} \left[\log \left(\frac{q^y(z)}{q(z|y)} \right) \right] \\ &= \mathbb{E}_{p_D(x|y)} [KL(q(z|x)||q^y(z))] + \mathbb{E}_{p_D(y)q^y(z)q(x|z)} \left[\log \left(\frac{q^y(z)}{q(z|y)} \right) \right] \\ &= \mathbb{E}_{p_D(x|y)} \left[KL(q(z|x)||q^y(z)) + \mathbb{E}_{q^y(z)} \left[\log \left(\frac{q^y(z)}{q(z|y)} \right) \right] \right] \\ &= \mathbb{E}_{p_D(x|y)} [KL(q(z|x)||q^y(z)) + KL(q^y(z)||q(z|y))] \end{aligned}$$

This derivation shows us that minimizing the KL divergence of Eqn. 5.8 is equivalent to minimizing 1) the KL between the approximate posterior of the image and the the aggregate posterior and 2) the KL between the aggregate posterior and the approximate posterior of

the text:

$$\mathbb{E}_{p_D(x|y)} \left[\underbrace{KL(q(z|x)||q_\phi(z|y))}_{\text{Posterior x to Posterior y}} \right] = \mathbb{E}_{p_D(x|y)} \left[\underbrace{KL(q(z|x)||q^y(z))}_{\text{Posterior x to Aggregate y}} + \underbrace{KL(q^y(z)||q_\phi(z|y))}_{\text{Aggregate y to Posterior y}} \right] \quad (5.9)$$

Consequently, minimizing the loss L_{Agg} wrt ϕ effectively minimizes $KL(q^y(z)||q_\phi(z|y))$, since ϕ appears only in the second term of the RHS of Eqn. 5.9. Also, the gradient of Eqn. 5.8 is equivalent to the gradient of the cross-entropy: $H(q(z|x), q_\phi(z|y))$, since $q(z|x)$ is not being optimized, just sampled from.

5.4 Related Work

Distribution Mismatch

The problem of the mismatch between the prior and the aggregate posterior has been remarked in a number of works (Makhzani et al., 2015; Huang et al., 2017; Rosca et al., 2018). There have been a several approaches to addressing this problem. In Makhzani et al. (2015), they use an adversarial loss to fit the aggregate posterior to the prior. In Tomczak and Welling (2018), they learn a mixture of posteriors as the prior. Similar to our work, in Huang et al. (2017) they learn the prior with a flow distribution. In contrast to these works, our work addresses this distribution mismatch in joint VAEs. This mismatch problem can be more severe in this setting because there are multiple decoders sharing the same latent variable which means the decoders can have differences in how they generalize to unseen latent variables, leading to posteriors that are different.

We gain insight into the behaviour of the objective of Eqn. 5.8 through our derivation of its decomposition (Eqn. 5.9). This derivation is similar to the ones done in Hoffman and Johnson (2016) and Vedantam et al. (2018), however they use the indexes of the datapoints in their derivation and they don't express the final decomposition in terms of KL divergences like ours.

Name	$q(z y)$ Approximates	$q(z x)$ Approximates
BiVCCA Wang et al. (2016)	$p(z x, y)$	$p(z x)$
JMVAE Suzuki et al. (2016)	$\mathbb{E}_{p_D(x,y)} [q(z x)]$	$p(z x) + q(z y)$
TELBO Vedantam et al. (2018)	$p(z y)$	$p(z x)$
VLVAE Cremer and Kushman (2018)	$\mathbb{E}_{p_D(x,y)} [q(z x)]$	$p(z x)$

Table 5.1: Approximations of joint VAE models. If we applied the objectives of each joint VAE model to the VLVAE model, these would be the distributions that $q(z|y)$ and $q(z|x)$ approximate.

Joint VAE Models

There have been a number of works that model joint distributions over multiple modalities using a latent variable. Although these works do not model the same modalities and may have different architectures, it is illuminating to compare their modelling assumptions.

In Wang et al. (2016), they present deep variational canonical correlation analysis (VCCA). Depending on the downstream task their objective may vary, however to train their inference networks for both modalities they optimize the joint probability using the modalities individually (BiVCCA objective):

$$\mathbb{E}_{q(z|x)} \left[\log \left(\frac{p(x|z)p(y|z)p(z)}{q(z|x)} \right) \right] + \mathbb{E}_{q(z|y)} \left[\log \left(\frac{p(x|z)p(y|z)p(z)}{q(z|y)} \right) \right]$$

In the case of inference with just the text, this is not ideal because it needs to infer the posterior of both the image and text but it does not have access to the image information.

Suzuki et al. (2016) present the joint multimodal VAE (JMVAE). The JMVAE objective for learning $q(z|y)$ applied to our model would be very similar to our Eqn. 5.8:

$$\mathbb{E}_{q(z|x)} \left[\log \frac{q(z|x)}{q(z|y)} \right],$$

However, the main difference is that JMVAE optimizes both $q(z|x)$ and $q(z|y)$ with the above loss. From our decomposition in Eqn. 5.9, we can see that this would pull $q(z|x)$ towards the aggregate distribution. In contrast, we only optimize $q(z|y)$ with Eqn. 5.8, so that $q(z|x)$ learns to approximate the true posterior only. Our objective isolates the optimization of

$q(z|y)$ from the rest of the model, so that we can learn the aggregate only. The JMVAE objective also optimizes $q(z|x)$ with Eqn. 5.2, thus it approximates the true posterior as well.

In Vedantam et al. (2018), they optimize a triple ELBO (TELBO) objective, which is the combination of the ELBOs for the joint probability and the the marginal probability of each modality. The result of this objective is that each approximate distribution tries to match the corresponding true posterior. Thus for $q(z|y)$, the TELBO optimizes Eqn. 5.5:

$$\mathbb{E}_{q_\phi(z|y)} \left[\log \left(\frac{p(y|z)p(z)}{q_\phi(z|y)} \right) \right].$$

In Vedantam et al. (2018), they show that the correctness of the samples from the TELBO-optimized model is sometimes lower than the JMVAE-optimized model. Our analysis of the aggregate posterior helps elucidate their results: their correctness is lower due to the mismatch between the aggregate posterior and the true posterior.

If we assume that each of the objectives that we’ve addressed uses the same model as the VLVAE, then table 5.1 is a summary of the distributions that each encoder is approximating. For instance, the TELBO approximates the true posterior for each modality. Whereas, JMVAE and VLVAE approximate the aggregate with $q(z|y)$, but differ on what $q(z|x)$ approximates.

Some works employ other approaches to match the distributions of the multiple modalities. In Chaudhury et al. (2017) they minimize the distance between the encodings of each modality. In Hsu and Glass (2018) they disentangle by separation, which means they maximize the similarity of the encodings from different modalities and minimize the similarity between different samples, ie. the multiview contrastive loss (Hermann and Blunsom, 2014).

5.5 Results

Conditional Sampling of $q_{Agg}(z|y)$

We will refer to $q_\phi(z|y)$ optimized using Eqn. 5.8 as $q_{Agg}(z|y)$ since it models the aggregate posterior instead of the true posterior of the text. In our experiments, we use a flow distribution for $q_{Agg}(z|y)$. The flow transformation that we employ is similar to the transformations of

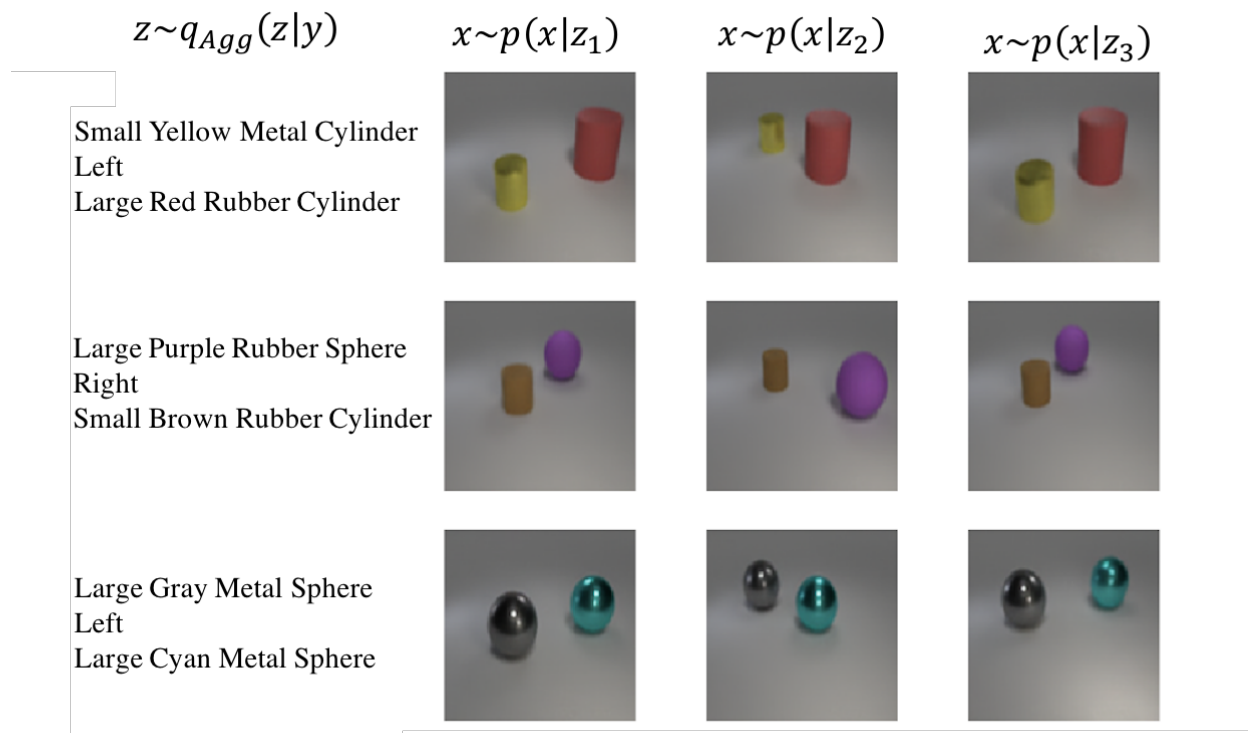


Figure 5.8: Conditional samples from $q_{Agg}(z|y)$. Column 1: Text provided to the text inference network. Columns 2,3,4: Image decoding of three samples z_0, z_1, z_2 from $q_{Agg}(z|y)$. In comparison to Fig. 5.7, these samples from the aggregate posterior are of much higher visual quality.

Real NVP (Dinh et al., 2017), however our flows are conditioned on text. The $q_{Agg}(z|y)$ distribution has six flow transformations conditioned on text stacked together. We use the same architecture as was used in section 5.2.2.

In Fig. 5.8, we show conditional image samples from this distribution. In contrast to Fig. 5.7, which approximated the true posterior of the text, we can see that the images are of much higher quality. This is a qualitative result demonstrating the benefit of modelling the aggregate posterior. However, it’s important to note that not all samples from $q_{Agg}(z|y)$ look as good as Fig. 5.8, but that on the whole, images from $q_{Agg}(z|y)$ look better than $q_{True}(z|y)$. The following section will compare the distributions more quantitatively.

Measuring Conditional Likelihood and Image Correctness

We compare the approximate posteriors in terms of two metrics: correctness and conditional likelihood. Following Vedantam et al. (2018), the correctness is the fraction of attributes for each generated image that match those specified in the text’s description. To compute

	Two Objects		CelebA	
	$q_{True}(z y)$	$q_{Agg}(z y)$	$q_{True}(z y)$	$q_{Agg}(z y)$
Correct (%)	57.29 ± 4.56	78.09 ± 0.05	80.17 ± 0.25	82.92 ± 0.04
$\log \hat{p}(x y)$	-6999.38 ± 817.45	1472.51 ± 0.23	-1710.59 ± 174.27	-178.24 ± 0.12

Table 5.2: Comparison of the aggregate of the image posteriors for a given text $q_{Agg}(z|y)$ and the approximate posterior for the text $q_{True}(z|y)$ on the validation set. Three runs are used to compute the mean and standard deviation for each value. Learning the aggregate generally helps improve the correctness of samples and conditional likelihood.

correctness, we use a classifier, which was trained independently from the model, to predict the attributes for a given image. Thus, in order to achieve high correctness, the model must generate images of sufficient quality for the classifier to distinguish as well as generate images which match the text that it’s conditioned on. We also estimate the conditional likelihood of held-out images under $q_\phi(z|y)$ by computing the following lower bound:

$$\log p(x|y) \geq \mathbb{E}_{q(z|x)} \left[\log \left(\frac{p(x|z)q_\phi(z|y)}{q(z|x)} \right) \right] \quad (5.10)$$

Thus, $q_\phi(z|y)$ will have lower conditional likelihood if it is not modelling the full diversity of images conditioned on the text.

5.5.1 Comparison of $q_{Agg}(z|y)$ and $q_{True}(z|y)$

We use $q_{True}(z|y)$ to refer to $q_\phi(z|y)$ trained to approximate the true posterior by optimizing Eqn 5.5. Similarly, we use $q_{Agg}(z|y)$ to refer to $q_\phi(z|y)$ trained to approximate the aggregate posterior by optimizing Eqn 5.9. For this experiment, we first train a VLVAE by optimizing the objective of Eqn. 5.2, then subsequently train $q_\phi(z|y)$.

Table 5.2 shows the comparison of $q_{True}(z|y)$ and $q_{Agg}(z|y)$ in terms of correctness and conditional likelihood on the validation set. We see that learning $q_{Agg}(z|y)$ improves the correctness of the generated images on both datasets. The large differences in $\log p(x|y)$ suggest that the true posterior $p(z|y)$ and the aggregate posterior $\mathbb{E}_{p(x,y)} [q(z|x)]$ are quite different distributions.

Figure 5.9 shows the correctness of image and text samples from each distribution during training. As expected, for $q_{True}(z|y)$, the text samples match the original text well, whereas

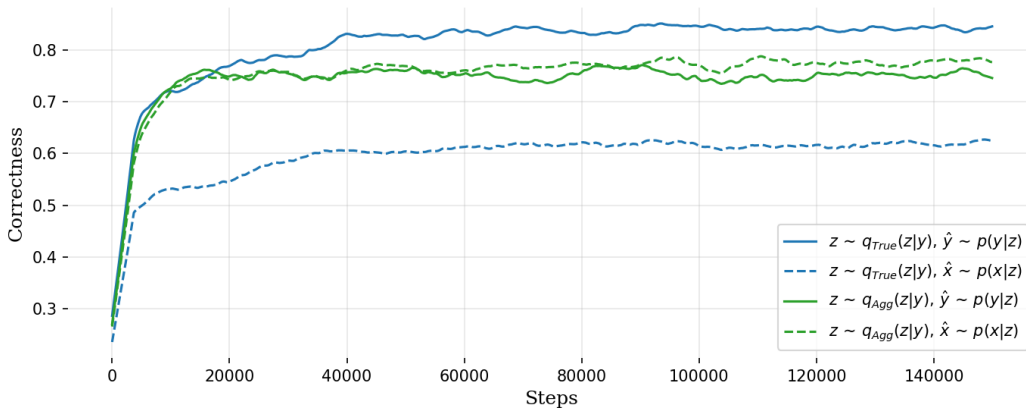


Figure 5.9: Comparison of $q_{Agg}(z|y)$ and $q_{True}(z|y)$ over training steps. Correctness refers to how well the generated sample (\hat{y} or \hat{x}) matches the text y that it is conditioned on. Blue lines are samples from $q_{True}(z|y)$ and green lines are from $q_{Agg}(z|y)$. Solid lines are text samples and dashed lines are image samples.

the image samples have lower correctness. For $q_{Agg}(z|y)$, the correctness of both the images and text are similar and its above the correctness of the images from $q_{True}(z|y)$.

In Fig. A.3 of the supplementary material, we show the nearest images in terms of Euclidean distance in the training set to the generated images from $q_{Agg}(z|y)$. We see that the generated images are all different from the training set, indicating that $q_{Agg}(z|y)$ is not simply overfitting to the training set and is able to generalize to new text.

5.5.2 Comparison of VLVAE and JMVAE

From section 5.4, we’ve seen that the objectives of the VLVAE and JMVAE are quite similar but differ due to JMVAE optimizing both $q(z|x)$ and $q(z|y)$ with Eqn. 5.8, whereas the VLVAE objective optimizes only $q(z|y)$. Given this difference, we’d like to see how it affects the resulting model. One concern with the VLVAE objective is that it optimizes $q(z|x)$ isolated from $q(z|y)$, thus the concern is that the aggregate of $q(z|x)$ may become too complex for $q(z|y)$ to model. We will measure the lower bound of $\log p(x|y)$ to examine this concern since it measures how well we are modelling the aggregate. We will also measure the lower bound of $\log p(x, y)$ to see how well each objective models the joint distribution.

We trained two models as described in section 5.2 with the VLVAE and JMVAE objectives and reported the lower bounds in Table 5.3. We see that the VLVAE objective achieves a

	Two Objects		CelebA	
	JMVAE	VLVAE	JMVAE	VLVAE
$\log \hat{p}(x, y)$	1049.71 ± 5.92	1134.94 ± 2.52	-1561.21 ± 24.52	-906.14 ± 8.92
$\log \hat{p}(x y)$	1468.87 ± 1.66	1472.06 ± 0.58	-773.95 ± 22.77	-178.42 ± 6.83

Table 5.3: Comparison of JMVAE and VLVAE objectives. Experiments are on the Two Object CLEVR dataset and CelebA. We approximate the likelihoods by measuring lower bounds. Bounds are computed on the validation set and three runs are used to compute the mean and standard deviation for each value.

higher lower bound for both $\log p(x, y)$ and $\log p(x|y)$, which suggests that optimizing $q(z|y)$ in isolation from the optimization of $q(z|x)$ is beneficial for the model.

5.6 Conclusion

In this work, we developed a latent variable model for text and images. We explored the benefits of training a model using inference from only images. We also showed that, due to a distribution mismatch, it is beneficial to learn the aggregate posterior. Through our decomposition of the objectives, we have improved our understanding of how to learn the aggregate posterior. This change in inference leads to improvements in the likelihoods and the quality of the generations from the model.

One direction of future work involves improving the inductive bias of the generative model. For instance, the current naive model is trained on images with two objects because the model could not generate the original CLEVR (Johnson et al., 2017) dataset, which has many objects per image. The improved inductive bias will likely arise from transfer learning, where the model learns to easily reason about the notion of objects.

Chapter 6

Discussion

In this section, I summarize the contributions of my thesis, I compare the tradeoffs between likelihood-based models, and discuss some open questions that remain to be addressed.

6.1 Summary of Contributions

The central theme of this thesis is the analysis of approximate inference in VAEs. Through this analysis, we learned about the many obstacles facing inference in VAEs and we gained insight into how to overcome them. My analysis of approximate inference in VAEs was composed of three parts. In Chapter 3, I demonstrated how the tighter bound objective of IWAE is equivalent to improved inference and I provided the necessary algorithms to properly use the distribution defined by this new interpretation of IWAE. In Chapter 4, I showed how we can break down and estimate the factors contributing to suboptimal inference and I performed an analysis of these factors over different datasets and model designs so that we can better understand how to improve approximate inference in VAEs. In Chapter 5, I developed a model for modelling the joint distribution over text and images and demonstrated the benefit of learning the aggregate posterior to deal with the mismatch of the posteriors of each modality.

6.2 Comparison of Likelihood-Based Models

When should a latent variable model be used over other density estimators? Although this thesis has focused on the analysis of VAEs, it is important to understand when using a VAE is appropriate. In particular, I'll compare the pros and cons of the three main likelihood-based models: VAEs, auto-regressive models, and flows. Although the choice of model is highly dependent on the specific task, this section will give a small snapshot of the current general state of these models and the details that should be considered when choosing a model.

Data Likelihood

Currently, auto-regressive models achieve the highest data likelihood on popular image datasets (van den Oord et al., 2016b,c; Chen et al., 2018; Parmar et al., 2018) and are the method of choice for text applications (Vaswani et al., 2017; Devlin et al., 2018). There are a number of possible reasons for its superior performance relative to VAEs and flows.

Firstly, for the popular image and texts datasets, its possible that auto-regressive models may have superior inductive biases for these types of data compared to VAEs and flows. Furthermore, most flow models are defined on continuous data, in contrast to discrete data, such as pixel values. Thus for image datasets, the data must be dequantized prior to passing through the flow. The dequantization results in the optimization of a lower bound to the data log-likelihood and improvements to the dequantization can lead to improvements in data log-likelihood (Ho et al., 2019). In contrast, auto-regressive models and VAEs can have output distributions over discrete data such as a a discretized logistic mixture likelihood (Salimans et al., 2017). However, there has been work on flows that work directly on discrete data (Tran et al., 2019; Hoogeboom et al., 2019). In addition to strong data likelihood, auto-regressive models can compute and optimize the likelihood exactly. In contrast, VAEs optimize a lower bound which may contribute to the model having lower likelihood over the data.

Sample Generation

The time required to generate a sample from the model may be important in real-time applications. For example, generation efficiency is important in a robotic task that requires generating possible future rollouts of video frames for decision making. Generally, auto-

regressive models generate each data dimension sequentially. Thus the generation time scales with the number of data dimensions, which may be very large for image and videos. There has been work to speed up auto-regressive generation by caching computation (Ramachandran et al., 2017). Another line of work attempts to distill the density of an auto-regressive model so that data generation can be done in parallel (van den Oord et al., 2018).

In contrast, VAEs with feedforward or convolutional decoders can generate samples quickly since the computation does not scale directly with the data dimensions. For flow models, some models can generate data in one pass (Dinh et al., 2017) while others require iterative methods to invert the forward transformation (Behrmann et al., 2019).

It is important to note that the likelihood-based models don't need to be used independently. As we've seen throughout the thesis, they can be combined. Other works have also combined latent variable models with auto-regressive decoders (Chen et al., 2016), or flow posteriors (Rezende and Mohamed, 2015), and flow models have been combined with auto-regressive models (Kumar et al., 2019). See Fig. 6.1 for a Venn diagram illustrating some of the combinations between latent variable models, auto-regressive models (AR) and flow models. FlowAR refers to a flow distribution where the base distribution is auto-regressive.

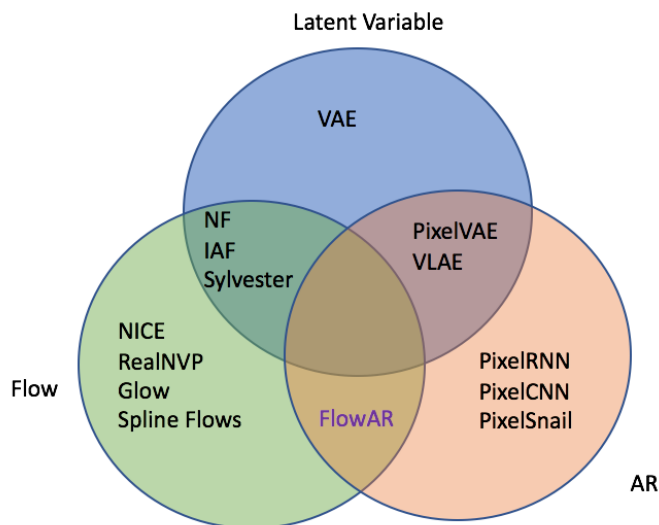


Figure 6.1: Venn diagram of different combinations of the likelihood-based models.

6.3 Future Directions

Distribution Underfitting: Sample Quality and Density Estimation

A criticism of the samples from VAEs relative to Generative Adversarial Networks (GANs) is that the images are blurry. One reason for this is that maximum likelihood is equivalent to minimizing the forward-KL between the model distribution and the data distribution:

$$\begin{aligned}\nabla_{\theta}\mathbb{E}_{p^*(x)}[\log p_{\theta}(x)] &= -\nabla_{\theta}\mathbb{E}_{p^*(x)}\left[\log\frac{p^*(x)}{p_{\theta}(x)}\right] \\ &= -\nabla_{\theta}\text{KL}(p^*(x)||p_{\theta}(x))\end{aligned}$$

The forward-KL is known to be mode-covering (Kristiadi, 2016), whereas GANs tend to be mode-seeking (Sajjadi et al., 2018). The result is that each model underfits the data distribution differently: VAEs tend to model more of the data diversity but with lower quality, whereas GANs have less diversity (mode dropping) but higher quality samples.

An open question is whether, given enough data and model capacity, will maximum likelihood models have similar quality images to GANs, such as Karras et al. (2019a,b)? Furthermore, how does underfitting affect the generalization of the density estimation of VAEs? Are the out-of-distribution samples that have higher probability (Nalisnick et al., 2019; Fetaya et al., 2019) due to simply underfitting the data distribution and the inductive biases of the model? Would more data solve these issues? Generalization in high dimensions is likely more challenging than in lower dimensions, thus improvements to the inductive biases of the model might be required.

Model Optimization

VAE learning is often faced with optimization challenges. For instance, mode collapse is a optimization problem encountered during VAE training where the approximate posterior is close to the prior distribution for a subset of the latent variable dimensions. VAE models are especially prone to this when the decoder is an auto-regressive model, because the decoder can model the data distribution without the need of the latent variable. However, even with simple decoders, this problem can arise. There are a number of techniques used to address

this problem. For instance, as shown in [Burda et al. \(2016\)](#), using the IWAE bound can reduce the number of latent variables that go unused. Another approach is to modify the ELBO either with free-bits ([Kingma et al., 2016](#)) or annealing the weight of the latent cost $KL(q(z|x)||p(z))$ from 0 to 1 during training ([Bowman et al., 2016](#); [Sønderby et al., 2016](#)).

Conditional generative models are often difficult to train as explained in [Fetaya et al. \(2019\)](#). VAEs are especially prone to optimization issues when training conditional generative models because the latent variable tends to include the same information contained in the conditioning variable, which leads the model to ignore the conditioning variable. These optimization difficulties can lead to suboptimal models and thus improving VAE optimization is still an open problem.

Discrete Latent Variables

Inferring latent variables with discrete structure is often desired for a number of reasons, including, for example, interpretability or to enforce specific structure on the latent variable. However, when taking expectations over discrete variables, we no longer have access to the reparameterization gradient and thus we must resort to other gradient estimators. Most unbiased gradient estimation approaches have revolved around the REINFORCE estimator approach:

$$\nabla_{\theta} \mathbb{E}_{p(b|\theta)} [f(b)] = \mathbb{E}_{p(b|\theta)} [f(b) \nabla_{\theta} \log p(b|\theta)]$$

These gradient estimators often suffer from high variance leading to slow learning. Using a control variate is one technique to reduce the variance. Recently, [Tucker et al. \(2017\)](#) took advantage of the Gumbel distribution to create a control variate that is conditioned on a continuous relaxation of the discrete variable while remaining unbiased. This idea was further generalized in [Grathwohl et al. \(2018\)](#), obtaining the RELAX gradient estimator.

The following is a comparison of a number of unbiased gradient estimators:

$$\begin{aligned}\hat{g}_{\text{REINFORCE}} &= f(b)\nabla_{\theta} \log p(b|\theta) \\ \hat{g}_{\text{RE+Baseline}} &= [f(b) - c] \nabla_{\theta} \log p(b|\theta) \\ \hat{g}_{\text{RELAX}} &= [f(b) - c_{\phi}(\tilde{z})] \nabla_{\theta} \log p(b|\theta) + \nabla_{\theta} c_{\phi}(z) - \nabla_{\theta} c_{\phi}(\tilde{z}) \\ \hat{g}_{\text{SIMPLAX}} &= [f(b) - c_{\phi}(z)] \nabla_{\theta} \log p(z|\theta) + \nabla_{\theta} c_{\phi}(z)\end{aligned}$$

where $p(b|\theta)$ is a Categorical distribution, $p(z|\theta)$ is a Gumbel distribution, $z \sim p(z|\theta)$, $b = \text{argmax}(z)$, and $\tilde{z} \sim p(\tilde{z}|b, \theta)$. The Simplax estimator is an estimator that has not been thoroughly investigated, but it is of interest due to its simplicity relative to RELAX.

Even with these variance reduction techniques, there is still considerable room for improvement (Le et al., 2019). With improved estimators, we will be able to more easily learn highly structured latent variables.

6.4 Conclusion

In this thesis, I examined inference in VAEs in a number of ways. I demonstrated how the tighter bound objective of IWAE is equivalent to improved inference. Furthermore, I investigated whether typical VAEs should be improved by reducing the errors arising from simple approximate distributions or errors associated with amortized inference. Finally, I developed a latent variable model over images and text and demonstrated the benefit of learning aggregate posteriors. I hope these analyses will lead to further improvements in generative modelling.

Bibliography

- Bachman, P. and Precup, D. (2015). Training Deep Generative Models: Variations on a Theme. In *NIPS Approximate Inference Workshop*.
- Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *International Conference on Learning Representations*.
- Behrmann, J., Grathwohl, W., Chen, R. T. Q., Duvenaud, D., and Jacobsen, J.-H. (2019). Invertible Residual Networks. *International Conference on Machine Learning*.
- Bengio, Y., Yao, L., Alain, G., and Vincent, P. (2013). Generalized denoising auto-encoders as generative models. *Advances in Neural Information Processing Systems*.
- Blei, D., Jordan, M., and Paisley, J. (2012). Variational bayesian inference with stochastic search. *International Conference on Machine Learning*.
- Bornschein, J. and Bengio, Y. (2015). Reweighted wake-sleep. *International Conference on Learning Representations*.
- Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. (2016). Generating Sentences from a Continuous Space. *Conference on Computational Natural Language Learning*.
- Burda, Y., Grosse, R., and Salakhutdinov, R. (2016). Importance weighted autoencoders. In *International Conference On Learning Representations*.
- Chaudhury, S., Dasgupta, S., Munawar, A., Salam Khan, M. A., and Tachibana, R. (2017). Conditional generation of multi-modal data using constrained embedding space mapping. *ICML Workshop on Implicit Models*.

- Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., and Abbeel, P. (2016). Variational Lossy Autoencoder. *International Conference on Learning Representations*.
- Chen, X., Mishra, N., Rohaninejad, M., and Abbeel, P. (2018). PixelSNAIL: An Improved Autoregressive Generative Model. *International Conference on Machine Learning*.
- Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2016). Fast and accurate deep network learning by exponential linear units (elus). *International Conference on Learning Representations*.
- Cremer, C. and Kushman, N. (2018). On the Importance of Learning Aggregate Posteriors in Multimodal Variational Autoencoders. In *Symposium on Advances in Approximate Bayesian Inference*.
- Cremer, C., Li, X., and Duvenaud, D. (2018). Inference Suboptimality in Variational Autoencoders. In *International Conference on Machine Learning*.
- Cremer, C., Morris, Q., and Duvenaud, D. (2017). Reinterpreting Importance-Weighted Autoencoders. In *ICLR Workshop*.
- Dayan, P., Hinton, G., Neal, R., and Zemel, R. (1995). The Helmholtz machine. *Neural computation*.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Association for Computational Linguistics*.
- Dinh, L., Krueger, D., and Bengio, Y. (2015). NICE: Non-linear Independent Components Estimation. *International Conference on Learning Representations*.
- Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2017). Density estimation using Real NVP. *International Conference on Learning Representations*.
- Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. (2019a). Cubic-Spline Flows. *ICML Workshop on Invertible Neural Networks and Normalizing Flows*.

- Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. (2019b). Neural Spline Flows. *Conference on Neural Information Processing Systems*.
- Fetaya, E., Jacobsen, J.-H., and Zemel, R. (2019). Conditional Generative Models are not Robust. *arXiv e-prints*, page arXiv:1906.01171.
- Germain, M., Gregor, K., Murray, I., and Larochelle, H. (2015). MADE: Masked Autoencoder for Distribution Estimation. In *International Conference on Machine Learning*.
- Geweke, J. (1989). Bayesian inference in econometric models using monte carlo integration. *Econometrica: Journal of the Econometric Society*, pages 1317–1339.
- Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256.
- Gomez, A. N., Ren, M., Urtasun, R., and Grosse, R. B. (2017). The reversible residual network: Backpropagation without storing activations. In *Advances in Neural Information Processing Systems*, pages 2211–2221.
- Grathwohl, W., Chen, R. T. Q., Bettencourt, J., Sutskever, I., and Duvenaud, D. (2019). FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models. *International Conference on Learning Representations*.
- Grathwohl, W., Choi, D., Wu, Y., Roeder, G., and Duvenaud, D. (2018). Backpropagation through the Void: Optimizing control variates for black-box gradient estimation. *International Conference on Learning Representations*.
- Gregor, K., Danihelka, I., Mnih, A., Blundell, C., and Wierstra, D. (2014). Deep AutoRegressive Networks. *International Conference on Machine Learning*.
- Grosse, R., Ghahramani, Z., and Adams, R. P. (2015). Sandwiching the marginal likelihood using bidirectional monte carlo. *arXiv preprint*, page arXiv:1511.02543.

- Grosse, R. B., Ancha, S., and Roy, D. M. (2016). Measuring the reliability of mcmc inference with bidirectional monte carlo. In *Advances in Neural Information Processing Systems*, pages 2451–2459.
- Gulrajani, I., Kumar, K., Ahmed, F., Taïga, A. A., Visin, F., Vázquez, D., and Courville, A. C. (2016). Pixelvae: A latent variable model for natural images. *CoRR*.
- Hermann, K. M. and Blunsom, P. (2014). Multilingual Distributed Representations without Word Alignment. *International Conference on Learning Representations*.
- Hinton, G., Dayan, P., Frey, B., and Neal, R. (1995). The “wake-sleep” algorithm for unsupervised neural networks. *Science*.
- Ho, J., Chen, X., Srinivas, A., Duan, Y., and Abbeel, P. (2019). Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design. *International Conference on Machine Learning*.
- Hoffman, M. and Johnson, M. (2016). Elbo surgery: yet another way to carve up the variational evidence lower bound. In *NIPS Workshop on Advances in Approximate Bayesian Inference*.
- Hoffman, M. D. (2017). Learning deep latent gaussian models with markov chain monte carlo. In *International Conference on Machine Learning*, pages 1510–1519.
- Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347.
- Hoogeboom, E., Peters, J. W. T., van den Berg, R., and Welling, M. (2019). Integer Discrete Flows and Lossless Compression. *Conference on Neural Information Processing Systems*.
- Hsu, W.-N. and Glass, J. (2018). Disentangling by Partitioning: A Representation Learning Framework for Multimodal Sensory Data. *ArXiv e-prints*.
- Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. (2018). Neural Autoregressive Flows. *International Conference on Machine Learning*.

- Huang, C.-W., Touati, A., Dinh, L., Drozdal, M., Havaei, M., Charlin, L., and Courville, A. (2017). Learnable Explicit Density for Continuous Latent Space and Variational Inference. *ICML workshop on Principle Approaches to Deep Learning*.
- Jarzynski, C. (1997). Nonequilibrium equality for free energy differences. *Physical Review Letters*, 78(14):2690.
- Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L., and Girshick, R. (2017). CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. *Conference on Computer Vision and Pattern Recognition*.
- Karras, T., Laine, S., and Aila, T. (2019a). A Style-Based Generator Architecture for Generative Adversarial Networks. *Conference on Computer Vision and Pattern Recognition*.
- Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. (2019b). Analyzing and Improving the Image Quality of StyleGAN. *arXiv e-prints*, page arXiv:1912.04958.
- Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.
- Kingma, D. P. and Dhariwal, P. (2018). Glow: Generative Flow with Invertible 1x1 Convolutions. *Conference on Neural Information Processing Systems*.
- Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016). Improving Variational Inference with Inverse Autoregressive Flow. In *Conference on Neural Information Processing Systems*.
- Kingma, D. P. and Welling, M. (2014). Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*.
- Krishnan, R. G., Liang, D., and Hoffman, M. (2018). On the challenges of learning with inference networks on sparse, high-dimensional data. In *International Conference on Artificial Intelligence and Statistics*.
- Kristiadi, A. (2016). Blog Post. <https://wiseodd.github.io/techblog/2016/12/21/forward-reverse-kl/>.

- Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. *University of Toronto*.
- Kumar, M., Babaeizadeh, M., Erhan, D., Finn, C., Levine, S., Dinh, L., and Kingma, D. (2019). VideoFlow: A Flow-Based Generative Model for Video. *ICML Workshop on Invertible Neural Networks and Normalizing Flows*.
- Larochelle, H. and Bengio, Y. (2008). Classification using discriminative restricted boltzmann machines. In *Proceedings of the 25th international conference on Machine learning*, pages 536–543. ACM.
- Le, T. A., Igl, M., Jin, T., Rainforth, T., and Wood, F. (2018). Auto-Encoding Sequential Monte Carlo. In *International Conference on Learning Representations*.
- Le, T. A., Kosiosek, A. R., Siddharth, N., Whye Teh, Y., and Wood, F. (2019). Revisiting Reweighted Wake-Sleep for Models with Stochastic Control Flow. *Uncertainty in Artificial Intelligence*.
- LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*.
- Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. *In Intl. Conf. on Computer Vision*.
- Louizos, C. and Welling, M. (2017). Multiplicative Normalizing Flows for Variational Bayesian Neural Networks. In *International Conference on Machine Learning*.
- Maaløe, L., Fraccaro, M., Liévin, V., and Winther, O. (2019). BIVA: A Very Deep Hierarchy of Latent Variables for Generative Modeling. *Conference on Neural Information Processing Systems*.
- Maaløe, L., Sønderby, C., Sønderby, S., and Winther, O. (2016). Auxiliary Deep Generative Models. In *International Conference on Machine Learning*.

- Maddison, C. J., Lawson, D., Tucker, G., Heess, N., Norouzi, M., Mnih, A., Doucet, A., and Whye Teh, Y. (2017). Filtering Variational Objectives. In *Conference on Neural Information Processing Systems*.
- Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. (2015). Adversarial Autoencoders. *arXiv e-prints*, page arXiv:1511.05644.
- Miller, A. C., Foti, N., and Adams, R. P. (2016). Variational Boosting: Iteratively Refining Posterior Approximations. *Advances in Approximate Bayesian Inference, NIPS Workshop*.
- Müller, T., McWilliams, B., Rousselle, F., Gross, M., and Novák, J. (2019). Neural Importance Sampling. *ACM Transactions on Graphics*.
- Naesseth, C. A., Linderman, S. W., Ranganath, R., and Blei, D. M. (2018). Variational Sequential Monte Carlo. In *International Conference on Artificial Intelligence and Statistics*.
- Nalisnick, E., Matsukawa, A., Whye Teh, Y., Gorur, D., and Lakshminarayanan, B. (2019). Do Deep Generative Models Know What They Don't Know? *International Conference on Learning Representations*.
- Neal, R. (2001). Annealed importance sampling. *Statistics and Computing*.
- Neal, R. (2011). MCMC using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*.
- Papamakarios, G., Pavlakou, T., and Murray, I. (2017). Masked Autoregressive Flow for Density Estimation. *Conference on Neural Information Processing Systems*.
- Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., and Tran, D. (2018). Image Transformer. *International Conference on Machine Learning*.
- Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *North American Chapter of the Association for Computational Linguistics*.

- Rainforth, T., Kosiorek, A. R., Le, T. A., Maddison, C. J., Igl, M., Wood, F., and Whye Teh, Y. (2018). Tighter Variational Bounds are Not Necessarily Better. *International Conference on Machine Learning*.
- Ramachandran, P., Le Paine, T., Khorrami, P., Babaeizadeh, M., Chang, S., Zhang, Y., Hasegawa-Johnson, M. A., Campbell, R. H., and Huang, T. S. (2017). Fast Generation for Convolutional Autoregressive Models. *International Conference on Learning Representations*.
- Ranganath, R., Gerrish, S., and Blei, D. M. (2013). Black Box Variational Inference. *arXiv e-prints*, page arXiv:1401.0118.
- Ranganath, R., Tran, D., and Blei, D. M. (2016). Hierarchical Variational Models. In *International Conference on Machine Learning*.
- Razavi, A., van den Oord, A., and Vinyals, O. (2019). Generating Diverse High-Fidelity Images with VQ-VAE-2. *arXiv e-prints*, page arXiv:1906.00446.
- Rezende, D. and Mohamed, S. (2015). Variational Inference with Normalizing Flows. In *International Conference on Machine Learning*.
- Rezende, D., Mohamed, S., and Wierstra, D. (2014). Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *International Conference on Machine Learning*.
- Rosca, M., Lakshminarayanan, B., and Mohamed, S. (2018). Distribution Matching in Variational Inference. *arXiv e-prints*, page arXiv:1802.06847.
- Sajjadi, M. S. M., Bachem, O., Lucic, M., Bousquet, O., and Gelly, S. (2018). Assessing Generative Models via Precision and Recall. *Conference on Neural Information Processing Systems*.
- Salakhutdinov, R. and Murray, I. (2008). On the quantitative analysis of deep belief networks. In *Proceedings of the 25th international conference on Machine learning*, pages 872–879. ACM.

- Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. (2017). PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications. *International Conference on Learning Representations*.
- Salimans, T., Kingma, D. P., and Welling, M. (2015). Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*.
- Sønderby, C., Raiko, T., Maaløe, L., Kaae Sønderby, S., and Winther, O. (2016). Ladder Variational Autoencoders. *Conference on Neural Information Processing Systems*.
- Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. (2016). Ladder variational autoencoders. In *Advances in Neural Information Processing Systems*, pages 3738–3746.
- Song, Y., Meng, C., and Ermon, S. (2019). MintNet: Building Invertible Neural Networks with Masked Convolutions. *Conference on Neural Information Processing Systems*.
- Suzuki, M., Nakayama, K., and Matsuo, Y. (2016). Joint Multimodal Learning with Deep Generative Models. *International Conference on Learning Representations Workshop*.
- Takahashi, H., Iwata, T., Yamanaka, Y., Yamada, M., and Yagi, S. (2018). Variational Autoencoder with Implicit Optimal Priors. *ArXiv e-prints*.
- Tomczak, J. M. and Welling, M. (2016). Improving Variational Auto-Encoders using Householder Flow. *NIPS Bayesian Deep Learning Workshop*.
- Tomczak, J. M. and Welling, M. (2017). Improving Variational Auto-Encoders using convex combination linear Inverse Autoregressive Flow. *ArXiv e-prints*.
- Tomczak, J. M. and Welling, M. (2018). VAE with a VampPrior. *International Conference on Artificial Intelligence and Statistics*.
- Tran, D., Vafa, K., Krishna Agrawal, K., Dinh, L., and Poole, B. (2019). Discrete Flows: Invertible Generative Models of Discrete Data. *Conference on Neural Information Processing Systems*.

- Tucker, G., Lawson, D., Gu, S., and Maddison, C. J. (2019). Doubly Reparameterized Gradient Estimators for Monte Carlo Objectives. *International Conference on Learning Representations*.
- Tucker, G., Mnih, A., Maddison, C. J., Lawson, D., and Sohl-Dickstein, J. (2017). REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models. *Conference on Neural Information Processing Systems*.
- Turner, R. and Sahani, M. (2011). Two problems with variational expectation maximisation for time-series models. In Barber, D., Cemgil, T., and Chiappa, S. (eds.), *Bayesian Time series models*, Chapter 5, pp. 109–130. Cambridge University Press.
- van den Berg, R., Hasenclever, L., Tomczak, J. M., and Welling, M. (2018). Sylvester Normalizing Flows for Variational Inference. *Uncertainty in Artificial Intelligence*.
- van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016a). WaveNet: A Generative Model for Raw Audio. *arXiv e-prints*, page arXiv:1609.03499.
- van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. (2016b). Pixel Recurrent Neural Networks. *International Conference on Machine Learning*.
- van den Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. (2016c). Conditional Image Generation with PixelCNN Decoders. *Conference on Neural Information Processing Systems*.
- van den Oord, A., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., van den Driessche, G., Lockhart, E., Cobo, L. C., Stimberg, F., Casagrande, N., Grewe, D., Noury, S., Dieleman, S., Elsen, E., Kalchbrenner, N., Zen, H., Graves, A., King, H., Walters, T., Belov, D., and Hassabis, D. (2018). Parallel WaveNet: Fast High-Fidelity Speech Synthesis. *International Conference on Machine Learning*.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention Is All You Need. *Conference on Neural Information Processing Systems*.

- Vedantam, R., Fischer, I., Huang, J., and Murphy, K. (2018). Generative Models of Visually Grounded Imagination. *International Conference on Learning Representations*.
- Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*.
- Wang, W., Yan, X., Lee, H., and Livescu, K. (2016). Deep Variational Canonical Correlation Analysis. *arXiv e-prints*, page arXiv:1610.03454.
- Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256.
- Wu, Y., Burda, Y., Salakhutdinov, R., and Grosse, R. (2017). On the Quantitative Analysis of Decoder-Based Generative Models. In *International Conference on Learning Representations*.
- Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. github.com/zalando-research/fashion-mnist.
- Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *International Conference on Computer Vision*.

Appendix A

Supplementary Material

A.1 Chapter 3 Supplementary Material

A.1.1 Proof that $\mathcal{L}_{VAE}[q_{EW}]$ is an upper bound of $\mathcal{L}_{IWA E}[q]$

Proof adapted from a proof provided from personal communication with Christian Naesseth.

This proof is used in Section 3.2.3.

$$\text{Let } \hat{p}(x|z_{1:k}) = \frac{1}{k} \left(\frac{p(x,z)}{q(z|x)} + \sum_{j=2}^k \frac{p(x,z_j)}{q(z_j|x)} \right)$$

$$\mathcal{L}_{VAE}[q_{EW}] = E_{z \sim q_{EW}} \left[\log \left(\frac{p(x,z)}{q_{EW}(z|x)} \right) \right] \quad (\text{A.1})$$

$$= E_{z \sim q_{EW}} \left[\log \left(\frac{p(x,z)}{E_{q(z_{2:k}|x)} \left[\frac{p(x,z)}{\hat{p}(x|z_{1:k})} \right]} \right) \right] \quad (\text{A.2})$$

$$= E_{z \sim q_{EW}} \left[\log \left(\frac{1}{E_{q(z_{2:k}|x)} \left[\frac{1}{\hat{p}(x|z_{1:k})} \right]} \right) \right] \quad (\text{A.3})$$

$$= E_{z \sim q_{EW}} \left[-\log \left(E_{q(z_{2:k}|x)} \left[\hat{p}(x|z_{1:k})^{-1} \right] \right) \right] \quad (\text{A.4})$$

$$= - \int_z p(x,z) E_{q(z_{2:k}|x)} \left[\hat{p}(x|z_{1:k})^{-1} \right] \log \left(E_{q(z_{2:k}|x)} \left[\hat{p}(x|z_{1:k})^{-1} \right] \right) dz \quad (\text{A.5})$$

$$\geq - \int_z p(x,z) E_{q(z_{2:k}|x)} \left[\hat{p}(x|z_{1:k})^{-1} \log \left(\hat{p}(x|z_{1:k})^{-1} \right) \right] dz \quad (\text{A.6})$$

$$= - \int_z p(x,z) \int_{z_{2:k}} q(z_{2:k}|x) \hat{p}(x|z_{1:k})^{-1} \log \left(\hat{p}(x|z_{1:k})^{-1} \right) dz \quad (\text{A.7})$$

$$= - \int_{z_{1:k}} \frac{q(z_1|x)}{q(z_1|x)} p(x,z_1) q(z_{2:k}|x) \hat{p}(x|z_{1:k})^{-1} \log \left(\hat{p}(x|z_{1:k})^{-1} \right) dz \quad (\text{A.8})$$

$$= - \int_{z_{1:k}} \frac{p(x, z_1)}{q(z_1|x)} q(z_{1:k}|x) \hat{p}(x|z_{1:k})^{-1} \log(\hat{p}(x|z_{1:k})^{-1}) dz \quad (\text{A.9})$$

$$= \int_{z_{1:k}} \frac{\frac{p(x, z_1)}{q(z_1|x)}}{\hat{p}(x|z_{1:k})} q(z_{1:k}|x) \log(\hat{p}(x|z_{1:k})) dz \quad (\text{A.10})$$

$$= k \int_{z_{1:k}} \frac{\frac{p(x, z_1)}{q(z_1|x)}}{\sum_{j=1}^k \frac{p(x, z_j)}{q(z_j|x)}} q(z_{1:k}|x) \log\left(\frac{1}{k} \sum_{j=1}^k \frac{p(x, z_j)}{q(z_j|x)}\right) dz \quad (\text{A.11})$$

$$= \sum_{i=1}^k \int_{z_{1:k}} \frac{\frac{p(x, z_i)}{q(z_i|x)}}{\sum_{j=1}^k \frac{p(x, z_j)}{q(z_j|x)}} q(z_{1:k}|x) \log\left(\frac{1}{k} \sum_{j=1}^k \frac{p(x, z_j)}{q(z_j|x)}\right) dz \quad (\text{A.12})$$

$$= \int_{z_{1:k}} \frac{\sum_{i=1}^k \frac{p(x, z_i)}{q(z_i|x)}}{\sum_{j=1}^k \frac{p(x, z_j)}{q(z_j|x)}} q(z_{1:k}|x) \log\left(\frac{1}{k} \sum_{j=1}^k \frac{p(x, z_j)}{q(z_j|x)}\right) dz \quad (\text{A.13})$$

$$= \int_{z_{1:k}} q(z_{1:k}|x) \log\left(\frac{1}{k} \sum_{j=1}^k \frac{p(x, z_j)}{q(z_j|x)}\right) dz \quad (\text{A.14})$$

$$= E_{q(z_{1:k})} \left[\log\left(\frac{1}{k} \sum_{j=1}^k \frac{p(x, z_j)}{q(z_j|x)}\right) \right] \quad (\text{A.15})$$

$$= \mathcal{L}_{IWAEL}[q] \quad (\text{A.16})$$

(A.6): Given that $f(A) = -A \log A$ is concave for $A > 0$, and $f(E[x]) \geq E[f(x)]$, then $f(E[x]) = -E[x] \log E[x] \geq E[-x \log x]$.

(A.8): Change of notation $z = z_1$.

(A.12): z_i has the same expectation as z_1 so we can replace k with the sum of k terms.

A.2 Chapter 4 Supplementary Material

2D Visualization

The VAE model of Fig. 4.2 uses a decoder $p(x|z)$ with architecture: 2 – 100 – 784, and an encoder $q(z|x)$ with architecture: 784 – 100 – 4. We use tanh activations and a batch size of 50. The model is trained for 3000 epochs with a learning rate of 10^{-4} using the ADAM optimizer (Kingma and Ba, 2015).

To plot the true posteriors $p(z|x) = \frac{p(x,z)}{p(x)}$ in Fig. 4.2, we compute $p(x|z)p(z)$ in a grid in latent space, then normalize by $p(x) = \int_z p(x,z)dz \approx \sum p(x,z)\Delta z$, where the sum is over the grid and Δz is the spacing of the grid.

MNIST & Fashion-MNIST

Both MNIST and Fashion-MNIST consist of a training and test set with 60000 and 10000 datapoints respectively, where each datapoint is a 28x28 grey-scale image. We rescale the original images so that pixel values are within the range $[0, 1]$. For MNIST, We use the statically binarized version described by Larochelle and Bengio (2008). We also binarize Fashion-MNIST *statically*. For both datasets, we adopt the Bernoulli likelihood for the generator.

The VAE models for MNIST and Fashion-MNIST experiments have the same architecture. The encoder has two hidden layers with 200 units each. The activation function is chosen to be the exponential linear unit (ELU, Clevert et al. (2016)), as we observe improved performance compared to tanh. The latent space has 50 dimensions. The generator is the reverse of the encoder. We follow the same learning rate schedule and train for the same amount of epochs as described by Burda et al. (2016). All models are trained with the a batch-size of 100 with ADAM.

In the large encoder setting, we change the number of hidden units for the inference network to be 500, instead of 200. The warm-up models are trained with a linear schedule over the first 400 epochs according to Section 4.5.6.

The auxiliary variable of requires a couple distributions: $q(v_0|z_0)$ and $r(v_T|z_T)$. These distributions are both factorized Gaussians which are parameterized by MLP’s with two

hidden layers, 100 units each, with ELU activations.

The flow transformation $q(z_{t+1}, v_{t+1}|z_t, v_t)$ involves functions σ_1 , σ_2 , μ_1 , and μ_2 from Eqn. 4.4 and 4.5. These also have two hidden layers with 100 units each and ELU units.

3-BIT CIFAR

CIFAR-10 consists of a training and test dataset with 50000 and 10000 datapoints respectively, where each datapoint is a 32×32 RGB image. We rescale individual pixel values to be in the range $[0, 1]$. We then statically binarize the scaled pixel values by setting individual pixel values of channels to 1 if the rescaled value is greater than 0.5 and 0 otherwise. In this manner, we can model the observation with a factorized Bernoulli likelihood. We call this binarized CIFAR-10 dataset as *3-BIT CIFAR*, since 3 bits are required to encode each pixel, where 1 bit is needed for each of the channels. We acknowledge that such binarization scheme may reduce the complexity of the original problem, since originally 24 bits were required to encode a single pixel. Nevertheless, the 3-bit CIFAR dataset is still much more challenging compared MNIST and Fashion. This is because 784 bits are required to encode one MNIST/Fashion image, whereas for one 3-bit CIFAR image, 3072 bits are required. Most notably, we were able to validate our AIS estimates using BDMC with the simplified dataset. This, however, was not achievable in any reasonable amount of time with the original CIFAR-10 dataset.

For the latent variable, we use a 50-dimensional factorized Gaussian for $q(z|x)$. For all neural networks, ELU is chosen to be the activation function. The inference network consists of three 4 by 4 convolution layers with stride 2, batch-norm, and 64, 128, 256 channels respectively. Then a fully-connected layer outputs the 50-dimensional mean and log-variance of the latent variable. Similarly, the generator consists of a fully-connected layer outputting 256 by 2 by 2 tensors. Then three deconvolutional layers each with 4 by 4 filters, stride 2, batch-norm, and 128, 64, and 3 channels respectively. For the model with expressive inference, we use three normalizing flow steps, where the parametric functions in the flow and auxiliary variable distribution also take in a hidden layer of the encoder.

We use a learning rate of 10^{-3} . Warm-up is applied with a linear schedule over the first 50 epochs. All models are trained with a batch-size of 100 with ADAM. Early-stopping is applied based on the performance computed with the IWAE bound (k=1000) on the held-out

set of 5000 examples from the original training set.

A.2.1 Generalization of Amortized Inference on Held-Out Data

This section provides details and extra experiments for section 4.5.5. The models are trained with batch size 50 and latent dimension size of 20. The rest of the hyperparameters are equivalent to Section A.2.

Architecture of q_{Flow} : The flow transformation involves functions σ_1 , σ_2 , μ_1 , and μ_2 from Eqn. 4.4 and 4.5. Each function is an MLP with a 50 unit hidden layer and ELU activations. We apply this flow transformation twice.

We performed the same experiments of Fig. 4.3 with the q_{AF} approximate posterior. For this model, the transformations are the same as q_{Flow} , but rather than partitioning the latent variable, we introduce an auxiliary variable. The auxiliary variable also requires a reverse model $r(v|z)$ which is a factorized Gaussian parameterized by an MLP with a 50 unit hidden layer and ELU activations. Fig. A.1 are the plots for the q_{AF} model.

Comparing q_{AF} in Fig. A.1 to q_{Flow} in Fig. 4.3, we see that the q_{AF} has a larger approximation gap. This increase is likely due to the $\text{KL}(q(v|z, x)||r(v|x, z))$ term of the auxiliary variable lower bound from 2.2.3. This motivates also using expressive approximations for the reverse model $r(v|z)$.

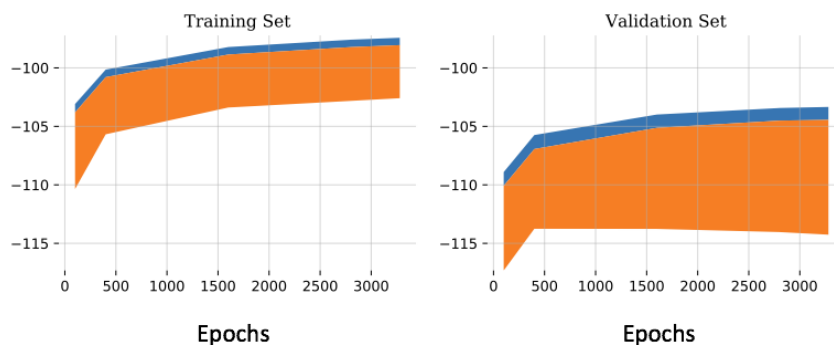


Figure A.1: Approximation Gap (blue) and Amortization Gap (orange) over epochs of the AF (auxiliary flow) model.

A.2.2 Influence of Flows On Amortization Gap Experiment

The aim of this experiment is to show that the parameters used for increasing the expressiveness of the approximation also contribute to reducing the amortization error. To show this, we train a VAE on MNIST, discard the encoder, then retrain two encoders on the fixed decoder: one with a factorized Gaussian distribution and the other with a parameterized 'flow' distribution. We use fixed decoder so that the true posterior is constant for both encoders. See 4.5.3 for the results and below for the architecture details.

The architecture of the decoder is: $D_Z - 200 - 200 - D_X$. The architecture of the encoder used to train the decoder is $D_X - 200 - 200 - 2D_Z$. The approximate distribution $q(z|x)$ is a factorized Gaussian.

Next, we describe the encoders which were trained on the fixed trained decoder. In order to highlight a large amortization gap, we employed a very small encoder architecture: $D_X - 2D_Z$. This encoder has no hidden layers, which greatly impoverishes its ability and results in a large amortization gap.

We compare two approximate distributions $q(z|x)$. Firstly, we experiment with the typical fully factorized Gaussian (FFG). The second is what we call a flow distribution. Specifically, we use the transformations of [Dinh et al. \(2017\)](#). We also include an auxiliary variable so we don't need to select how to divide the latent space for the transformations. The approximate distribution over the latent z and auxiliary variable v factorizes as: $q(z, v|x) = q(z|x)q(v)$. The $q(v)$ distribution is simply a $N(0,1)$ distribution. Since we're using a auxiliary variable, we also require the $r(v|z)$ distribution which we parameterize as $r(v|z): [D_Z] - 50 - 50 - 2D_Z$. The flow transformation is the same as in Section 4.3.2, which we apply twice.

A.2.3 Computation of the Determinant for Flow

The overall mapping f that performs $(z, v) \mapsto (z', v')$ is the composition of two sheer mappings f_1 and f_2 that respectively perform $(z, v) \mapsto (z, v')$ and $(z, v') \mapsto (z', v')$. Since the Jacobian of either one of the sheer mappings is diagonal, the determinant of the composed transformation's

Jacobian Df can be easily computed:

$$\begin{aligned}\det(Df) &= \det(Df_1)\det(Df_2) \\ &= \left(\prod_{i=1}^n \sigma_1(z)_i\right) \left(\prod_{j=1}^n \sigma_2(v'_j)\right).\end{aligned}$$

A.2.4 Annealed Importance Sampling

Annealed importance sampling (AIS, Neal (2001); Jarzynski (1997)) is a means of computing a lower bound to the marginal log-likelihood. Similarly to the importance weighted bound, AIS must sample a proposal distribution $f_1(z)$ and compute the density of these samples, however, AIS then transforms the samples through a sequence of reversible transitions $\mathcal{T}_t(z'|z)$. The transitions anneal the proposal distribution to the desired distribution $f_T(z)$.

Specifically, AIS samples an initial state $z_1 \sim f_1(z)$ and sets an initial weight $w_1 = 1$. For the following annealing steps, z_t is sampled from $\mathcal{T}_t(z'|z)$ and the weight is updated according to:

$$w_t = w_{t-1} \frac{f_t(z_{t-1})}{f_{t-1}(z_{t-1})}.$$

This procedure produces weight w_T such that $\mathbb{E}[w_T] = \mathcal{Z}_T/\mathcal{Z}_1$, where Z_T and Z_1 are the normalizing constants of $f_T(z)$ and $f_1(z)$ respectively. This pertains to estimating the marginal likelihood when the target distribution is $p(x, z)$ when we integrate with respect to z .

Typically, the intermediate distributions are simply defined to be geometric averages: $f_t(z) = f_1(z)^{1-\beta_t} f_T(z)^{\beta_t}$, where β_t is monotonically increasing with $\beta_1 = 0$ and $\beta_T = 1$. When $f_1(z) = p(z)$ and $f_T(z) = p(x, z)$, the intermediate distributions are: $f_i(x) = p(z)p(x|z)^{\beta_i}$.

Model evaluation with AIS appears early on in the setting of deep belief networks (Salakhutdinov and Murray, 2008). AIS for decoder-based models was also used by Wu et al. (2017).

A.2.5 Extra MNIST Inference Gaps

To demonstrate that a very small inference gap can be achieved, even with a limited approximation such as a factorized Gaussian, we train the model on a small dataset. In this experiment, our training set consists of 1000 datapoints randomly chosen from the original MNIST training set. The training curves on this small dataset are shown in Fig. A.2. Even with a factorized Gaussian distribution, the inference gap is very small: the AIS and IWAE bounds are overlapping and the VAE is just slightly below. Yet, the model is overfitting as seen by the decreasing test set bounds.

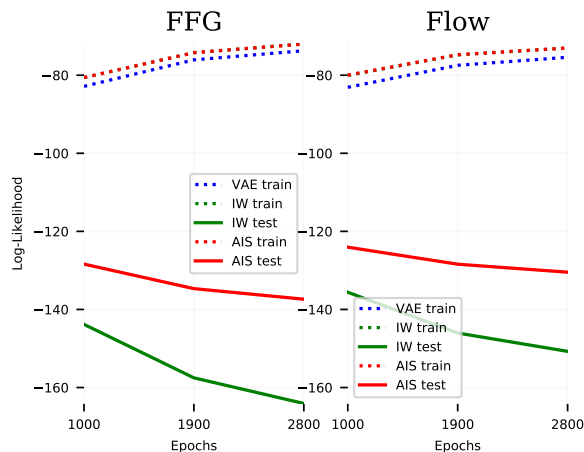


Figure A.2: Training curves for a FFG and a Flow inference model on MNIST. AIS provides the tightest lower bound and is independent of encoder overfitting. There is little difference between FFG and Flow models trained on the 1000 datapoints since inference is nearly equivalent.

A.3 Chapter 5 Supplementary Material

A.3.1 Model Architecture

Overview of model: The model will involve different components depending of which modalities it uses for inference and which modalities to generate. For instance, in Eqn. 5.2, the model performs inference with only images and then generates images and text. In this case, the model begins by encoding the image into a vector. If we were performing inference with both modalities, both modalities would be encoded and the vectors would be concatenated. Next, the encoding is passed to another network which outputs the parameters of a Gaussian distribution over the latent space. The distribution is then sampled and the images and text are decoded. To decode the latent vector samples, the samples begin by being decoded by a network which is shared by both modalities. Then this vector is passed to an image decoder and text decoder. In the following, we will provide more detail of each of those components.

Image encoder: We use networks similar to the ones used in CycleGAN (Zhu et al., 2017). Broadly, we use a 3-block Resnet with instance norm then flatten the representation and output a 200 dimensional vector. We build off this implementation:

<https://github.com/aitorzip/PyTorch-CycleGAN/blob/master/models.py>.

Text encoder: The text is encoded with a GRU with a hidden state size of 200. The final hidden state is used as the text encoding

Shared encoder network: The encodings of each modality are concatenated then passed through three fully-connected layers with batchnorm and a residual connection. It outputs the means and variances of the Gaussian for the latent variable z .

Shared decoder network: The latent variable z is passed through five fully-connected layers with two residual connections and batchnorm. The resulting vector has dimensionality of 200.

Image decoder: The vector from the shared decoder network is linearly transformed into a 1000 dimensional vector then reshaped into 2D representation with 10 channels. Similar to the image encoder, we use a 3-block Resnet with instance norm to output the parameters of the image likelihood distribution. See Section A.3.1 for details of the image likelihood.

Text decoder: The text decoder is a GRU with a 200 dimensional hidden state. Each hidden state is dependent on the previous word, previous hidden state, and the shared decoder network output. The hidden state for each word is put through three fully-connected layers with a residual connection, batchnorm, and dropout, and outputs the distribution over the vocabulary.

Flow Distributions

In Chapter 5, we use flows to model the distributions which are conditioned on text $q_\phi(z|y)$. The flow transformation that we employ is similar to the transformations of Real NVP (Dinh et al., 2017). We partition the latent variable z into two, z_1 and z_2 , then perform the following transformations:

$$z'_1 = z_1 \circ \sigma_1(z_2) + \mu_1(z_2) \tag{A.17}$$

$$z'_2 = z_2 \circ \sigma_2(z'_1) + \mu_2(z'_1) \tag{A.18}$$

where $\sigma_1, \sigma_2, \mu_1, \mu_2 : \mathbb{R}^n \rightarrow \mathbb{R}^n$ are differentiable mappings parameterized by neural nets and \circ takes the Hadamard or element-wise product. We partition the latent variable by simply indexing the elements of the first half and the second half. The determinant of the combined transformation's Jacobian is $|\det(\frac{\partial z'}{\partial z})| = \left(\prod_{i=1}^n \sigma_1(z)_i\right) \left(\prod_{j=1}^n \sigma_2(v')_j\right)$. We employ two of these flows for each sample.

Image Likelihood Distribution

The images are preprocessed to be continuous values between 0 and 1. For our image likelihood $p(x|z)$, we use a constrained Beta distribution. Specifically, the image decoder outputs the α parameter of the Beta distribution and we set the β parameter to $1 - \alpha$, thus the distribution is constrained to α and β values which sum to 1. We then scale both α and β by a factor of 100 in order to adjust the variance of the distribution. We chose a Beta distribution because it is a distribution over continuous values between 0 and 1, unlike other commonly used distributions such as Gaussian or Laplace.

A.3.2 Dataset Details

Two Object CLEVR

We modified the CLEVR dataset (Johnson et al., 2017) so that the image contains only two objects and the text is a description of the objects as well as the relation between them. We call this dataset the Two Object dataset. The images have dimensions: [112,112,3]. The text consists of nine words and the format of the text is as following: $O_1^{size}, O_1^{colour}, O_1^{material}, O_1^{shape}, R, O_2^{size}, O_2^{colour}, O_2^{material}, O_2^{shape}$, where O_1 and O_2 are used to refer to each object in the image and R refers to the relational word. We will now list the vocabulary for each attribute. O^{size} : small, large. O^{colour} : cyan, red, gray, blue, yellow, purple, brown, green. $O^{material}$: rubber, metal. O^{shape} : sphere, cylinder, cube. For the relational word, the possible words are: right, left, front, behind. Thus, there is a total of $2*8*2*3*4=384$ possible different texts. The dataset consists of 100k image-text pairs, where 90k are used for training and 10k for validation.

CelebA

We use the CelebA dataset (Liu et al., 2015), where we’ve converted the attributes into a sequence of shuffled words. The images have dimensions: [64,64,3]. The max length of the text is 9 words, but the median length is 4 words. Following Vedantam et al. (2018), we use a subset of the attributes. The vocabulary encompasses 20 different tokens: Bushy Eyebrows, Male, Female, Mouth Slightly Open, Smiling, Bald, Bangs, Black Hair, Blond Hair, Brown Hair, Eyeglasses, Gray Hair, Heavy Makeup, Mustache, Pale Skin, Receding Hairline, Straight Hair, Wavy Hair, Wearing Hat, and ‘-’ to indicate blank word. There are a total of 202599 image-text pairs, and 180k are used for training and 22598 are used for validation.

A.3.3 Training Details

We trained the model for 400k steps with the ADAM optimizer with learning rate $4 * 10^{-4}$. We used a batch size of 20. The size of the latent variable was 50 dimensions. We employed warmup to the objective for the first 20k steps. More specifically, we annealed the weight of

the KL term of the objective from 0 to 1 over the first 20k steps.

Likelihood Term Weights

Given that the dimensionality of the modalities widely differ, the evidence lower bound objective does not align with the real importance of each modality. For instance, we have images of $112 \times 112 \times 3 = 37632$ dimensions, compared to text with around 9 dimensions. Thus there is much greater weight put on the image compared to the text. To compensate, we down-weight the likelihood term of the images $p(x|z)$ and we up-weight the text likelihood $p(y|z)$ in the objective during training. For the experiments of section 5.5, on the Two Object CLEVR dataset we multiply $p(x|z)$ by .02 and $p(y|z)$ by 200. On the CelebA dataset, we multiply $p(x|z)$ by .1 and $p(y|z)$ by 100. We tried a number of different values for these hyperparameters, however for the range of hyperparameters we tried, our results were consistent for the comparisons we made.

A.3.4 2D Visualization Details

For Fig. 5.2, we trained a VLVAE model with a latent size of 2 dimensions. It was trained on a simplified version of the CLEVR dataset, where each image contains only a single object and where the object only varies in its colour and size. More specifically, the object in the image can only be blue or red and large or small. Once trained, to plot the true posterior for the text $p(z|y)$, we computed $p(y|x)p(z)$ for each z in the grid then normalize. To plot the aggregate distributions, we encoded the images that correspond to the text y then normalized the mixture of the image approximate posteriors.

A.3.5 Classifier Details

To measure the correctness of the samples from the model, we train classifiers to predict the text corresponding to a given image. The architecture of the classifiers is nearly the same as the image encoder. For the correctness of the conditional samples of Table 5.2, we compare generated images to the given real text.

For the two object CLEVR dataset, there is ambiguity regarding the object ordering in

the text as well as the chosen relational word (right, left, front, back). Due to this, we pass the relational word to the classifier so that the output text is deterministic. The correctness is simply the fraction of matching words between the sample from the model and the classifier prediction. For the CelebA dataset, since the word ordering is random, the correctness measure ignores ordering. In this case, the classifier takes as input an image and outputs the probability of each word being associated with that image. Given some text, the correctness of the text is computed based on the fraction of the words that have a higher than 50% probability of being from that image (based on the classifier).

A.3.6 Aggregate Posterior Prior Decomposition

The decomposition of Eqn. 5.9 can be done for the prior as well. The following is the ELBO:

$$\mathcal{L} = \mathbb{E}_{p_D(x)} \left[\underbrace{\mathbb{E}_{q(z|x)} [\log p(x|z)]}_{\text{Reconstruction}} - \underbrace{KL(q(z|x)||p(z))}_{\text{Posterior to Prior}} \right] \quad (\text{A.19})$$

As is shown in Hoffman and Johnson (2016), we can rewrite the KL term above in terms of the aggregate posterior $q(z) = \mathbb{E}_{p_D(x)} [q(z|x)]$:

$$\begin{aligned} \mathbb{E}_{p_D(x)} [KL(q(z|x)||p(z))] &= \mathbb{E}_{p_D(x)} \left[\mathbb{E}_{q(z|x)} \left[\log \left(\frac{q(z|x)}{p(z)} \right) \right] \right] \\ &= \mathbb{E}_{p_D(x)} \left[\mathbb{E}_{q(z|x)} \left[\log \left(\frac{q(z|x)}{p(z)} * \frac{q(z)}{q(z)} \right) \right] \right] \\ &= \mathbb{E}_{p_D(x)} \left[\mathbb{E}_{q(z|x)} \left[\log \left(\frac{q(z|x)}{q(z)} \right) + \log \left(\frac{q(z)}{p(z)} \right) \right] \right] \\ &= \mathbb{E}_{p_D(x)} [KL(q(z|x)||q(z))] + \mathbb{E}_{p_D(x)q(z|x)} \left[\log \left(\frac{q(z)}{p(z)} \right) \right] \\ &= \mathbb{E}_{p_D(x)} [KL(q(z|x)||q(z)) + KL(q(z)||p(z))] \end{aligned}$$

So the lower bound \mathcal{L} can be written as:

$$\mathcal{L} = \mathbb{E}_{p_D(x)} \left[\underbrace{\mathbb{E}_{q(z|x)} [\log p(x|z)]}_{\text{Reconstruction}} - \underbrace{KL(q(z|x)||q(z))}_{\text{Posterior to Aggregate}} - \underbrace{KL(q(z)||p(z))}_{\text{Aggregate to Prior}} \right] \quad (\text{A.20})$$

A.3.7 Nearest Training Images

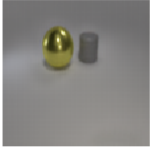
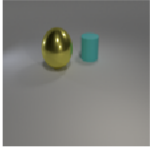
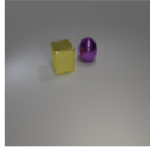
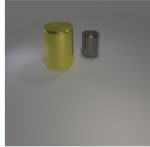
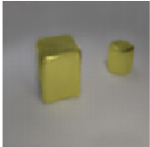
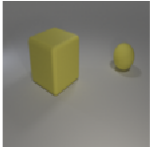
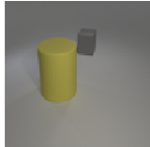
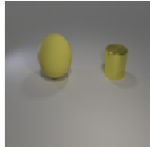
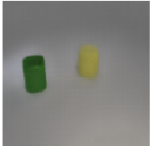
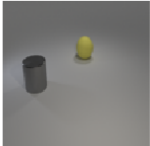
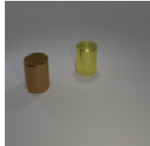
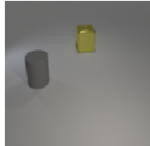

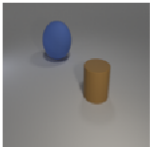
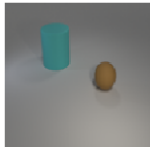
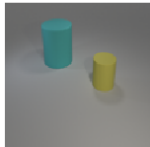

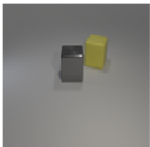

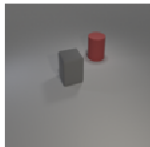



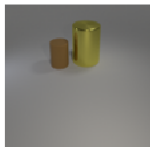
Conditioning Text	Generated Image	Closest Training Image 1	Closest Training Image 2	Closest Training Image 3
large yellow metal sphere left small gray rubber cylinder				
large yellow metal cube left small yellow metal cube				
small yellow rubber cube right small green metal cylinder				
small brown rubber sphere behind large blue rubber cube				
small gray rubber cylinder front small brown rubber sphere				
small brown metal cylinder behind large yellow metal cube				

Figure A.3: Nearest images in the training set to the generated conditional images based on Euclidean distance. The generated images are all different than the training images, indicating that the model has not overfit to the training set.