INFERENCE DISSECTION IN VARIATIONAL AUTOENCODERS

Anonymous authors
Paper under double-blind review

ABSTRACT

It can be difficult to diagnose whether a given variational autoencoder can benefit from more costly inference and if so, how to improve the inference. This paper introduces quantitative tests to answer these questions. We categorize the ways that inference can fail in VAEs in order to gain insight into how to improve it. These tests also illustrate how the choice of the approximate posterior influences the trained model.

1 INTRODUCTION

There has been significant work on improving variational autoencoders (VAEs) (Kingma & Welling, 2014; Rezende et al., 2014) through the development of expressive approximate posteriors (Rezende & Mohamed, 2015; Kingma et al., 2016; Ranganath et al., 2016; Tomczak & Welling, 2016; 2017). These works have shown that with more expressive approximate posteriors, the model learns a better distribution over the data.

In this paper, we analyze how inference is influenced by expressive approximate posteriors and subsequently how this affects the learned model. We begin by examining the sources that contribute to the mismatch between the true and approximate posterior. We refer to this mismatch as the inference gap. Moreover, we break down the gap into two components: the approximation gap and the amortization gap. The approximation gap comes from the inability of the approximate 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 datapoint independently. We refer the reader to Table 1 for detailed definitions.

To understand what roles these gaps play in inference, we train VAE models in a number of settings. We compare the gaps under different approximate posterior families, different encoder sizes on the MNIST and Fashion-MNIST (Xiao et al., 2017) datasets.

Our contributions are: a) we introduce the amortization and approximation gaps, b) we study in which settings each of them contribute more to the total inference gap, providing a tool to guide future improvements in VAE inference, and c) we quantitatively demonstrate that the true posterior can be influenced by the approximation through the joint optimization and the role warm-up plays.

2 BACKGROUND

2.1 VARIATIONAL AUTOENCODERS

Let \( x \) be the observed variable, \( z \) the latent variable, and \( p(x, z) \) be their joint distribution. Given a dataset \( X = \{x_1, x_2, ..., x_N\} \), we would like to maximize the marginal log-likelihood:

\[
\log p(X) = \sum_{i=1}^{N} \log p(x_i) = \sum_{i=1}^{N} \log \int p(x_i, z_i) dz_i.
\]  

(1)
Stacking these transformations remains tractable due to the determinant being decomposable:

\[ \text{det}(...) \]

From the above we can see that if \( q^*(z|z) = \arg\min_{q \in Q} \text{KL}(q(z|x)||p(z|x)) \) refers to the optimal approximation within a family \( Q \), i.e. \( q^*(z|x) \) refers to the optimal approximation within a family \( Q \).

In practice, the marginal log-likelihood is computationally intractable due to the integration over the latent variable \( z \). Instead, VAEs optimize the evidence lower bound (ELBO) of the marginal log-likelihood (Kingma & Welling, 2014; Rezende et al., 2014):

\[
\log p(x) = \mathbb{E}_{z \sim q(z|x)} \left[ \log \left( \frac{p(x,z)}{q(z|x)} \right) \right] + \text{KL}(q(z|x)||p(z|x))
\]

From the above we can see that if \( q(z|x) = p(z|x) \), then the lower bound is tight, which motivates approximations \( q(z|x) \) which have the capacity to model \( p(z|x) \).

The original VAE parameterizes the likelihood \( p(x|z) \) and inference distribution \( q(z|x) \) by deep neural nets. The overall model is trained by stochastically optimizing the ELBO using the reparametrization trick (Kingma & Welling, 2014).

### 2.2 Expressive Approximate Posteriors

There are a number of strategies for increasing the expressiveness of approximate posteriors, going beyond the original factorized-Gaussian. We briefly summarize normalizing flows and auxiliary variables.

#### 2.2.1 Normalizing Flows

Normalizing flow (Rezende & Mohamed, 2015) is a change of variables procedure for constructing complex distributions by transforming probability densities through a series of invertible mappings. Specifically, if we transform a random variable \( z_0 \) with distribution \( q_0(z) \), the resulting random variable \( z_T = T(z_0) \) has a distribution:

\[
q_T(z_T) = q_0(z_0) \left| \det \frac{\partial z_T}{\partial z_0} \right|^{-1}
\]

By successively applying these transformations, we can build arbitrarily complex distributions. Stacking these transformations remains tractable due to the determinant being decomposable: \( \text{det}(A \cdot B) = \text{det}(A)\text{det}(B) \). 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 its formula:

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

This is known as the law of the unconscious statistician (LOTUS). Using the change of variable and LOTUS, the lower bound can be written as:

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

The main constraint on these transformations is that the determinant of their Jacobian needs to be easily computable.
2.2.2 Auxiliary Variables

Deep generative models can be extended with auxiliary variables which leave the generative model unchanged but make the variational distribution more expressive. Just as hierarchical Bayesian models induce dependencies between data, hierarchical variational models can induce dependencies between latent variables. The addition of the auxiliary variable changes the lower bound to:

$$
\log(p(x)) \geq \mathbb{E}_{z \sim q(z|x)} \left[ \log \left( \frac{p(x, z)}{q(z|x)} \right) \right] \quad (7)
$$

$$
= \mathbb{E}_{q(z|x)} \left[ \log \left( \frac{p(x, z)}{q(z|x)} \right) - \text{KL} \left( q(v|z, x) \| r(v|z, x) \right) \right] \quad (8)
$$

where $r(v|z, x)$ is called the reverse model. From Eqn. 8 we see that this bound is looser than the regular ELBO, however the extra flexibility provided by the auxiliary variable can compensate and result in a higher lower bound. This idea has been employed in works such as auxiliary deep generative models (ADGM,[Maaløe et al.](2016)), hierarchical variational models (HVM,[Ranganath et al.](2016)) and Hamiltonian variational inference (HVI,[Salimans et al.](2015)).

2.3 Evaluation Metrics

Here we describe two options for estimating the marginal log-likelihood of a model: IWAE and AIS.

2.3.1 IWAE Bound

The bound used in [Burda et al.](2016) is a tighter lower bound than the VAE bound. 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)).

More specifically, if we take multiple samples from the $q$ distribution, we can compute a tighter lower bound to the marginal log likelihood:

$$
\log p(x) \geq \mathbb{E}_{z_1 \ldots z_k \sim q(z|x)} \left[ \log \left( \frac{1}{k} \sum_{i=1}^{k} \frac{p(x, z_i)}{q(z_i|x)} \right) \right] = L_{\text{IWAE}}[q]. \quad (9)
$$

This importance weighted bound was introduced along with the Importance Weighted Autoencoder ([Burda et al.](2016)), so we will refer to it as the IWAE bound. As shown by [Bachman & Precup](2015) and [Cremer et al.](2017), the IWAE bound can be seen as using the VAE bound but with an importance weighted $q$ distribution.

2.3.2 Annealed Importance Sampling

Annealed importance sampling (AIS,[Neal](2001),[Jarzynski](1997)) is another 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 $T_t(z'|z)$. 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 $T_t(z'|z)$ and the weight is updated according to:

$$
w_t = w_{t-1} \frac{f_1(z_{t-1})}{f_{t-1}(z_{t-1})}.
$$

This procedure produces weight $w_T$ such that $\mathbb{E} [w_T] = Z_T / Z_1$, 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 $p(x, z)$ and we are integrating with respect to $z$.

Typically, the intermediate distributions are simply defined to be geometric averages: $f_t(x) = f_1(x)^{1 - \beta_t} f_T(x)^{\beta_t}$, where $\beta_t$ is monotonically increasing with $\beta_1 = 0$ and $\beta_T = 1$. When $f_1(x) = p(z)$ and $f_T(x) = p(x, z)$, the intermediate distributions are: $f_t(x) = f_0(x) f_n(x)^{1 - \beta_t}$. 

3 Methods

3.1 Flexible Approximate Posterior

Our experimentation compares two families of approximate posteriors: the fully-factorized Gaussian (FFG) and a flexible flow (Flow). Our choice of flow is a combination of the Real NVP (Dinh et al., 2017) and auxiliary variables (Ranganath et al., 2016; Maaløe et al., 2016). Our model also resembles leap-frog dynamics applied in Hamiltonian Monte Carlo (HMC, Neal et al., 2011).

Let $z \in \mathbb{R}^n$ be the variable of interest and $v \in \mathbb{R}^n$ the auxiliary variable. Each flow step involves:

\begin{align}
  v' &= v \circ \sigma_1(z) + \mu_1(z) \\
  z' &= z \circ \sigma_2(v') + \mu_2(v')
\end{align}

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. The determinant of the combined transformation’s Jacobian, $|\det(Df)|$, can be easily evaluated. See section 6.2 in Appendix for a detailed derivation.

Thus, we can jointly train the generative and flow-based inference model by optimizing the bound:

$$
\log p(x) \geq \mathbb{E}_{z,v \sim q(z,v|x)} \left[ \log \left( \frac{p(x,z')r(v'|x,z')}{q(z|x,v)q(v|x)|\det(Df)|^{-1}} \right) \right] = \mathcal{L}_{flow}[q].
$$

Additionally, multiple such type of transformations can be stacked to improve expressiveness. We refer readers to the Appendix for details of our flow configuration adopted in the experimentation.

3.2 Evaluation Bounds

For our experiments, we test two different variational distributions $q$: the fully-factorized Gaussian $q_{FFG}$ or the flexible approximation $q_{Flow}$, described above. We use several bounds for evaluation. $\mathcal{L}_{VAE}[q]$ refers to the bound of Eqn. 3, with $q$ being either $q_{FFG}$ or $q_{Flow}$ depending on the experiment. $\mathcal{L}_{IWAE}[q]$ (or IW) is the bound defined at Eqn. 9. To compute $\mathcal{L}_{VAE}[q]$ and $\mathcal{L}_{IWAE}[q]$, we use 5000 samples. When computing the AIS bound, $\mathcal{L}_{AIS}$, we use Hamiltonian Monte Carlo (Neal, 2011) as our transition operator $T_t(z'|z)$. We use 100 samples, 10 HMC steps, and 500 intermediate distributions. The initial distribution is the prior, so that it is encoder-independent.

3.3 Gaps

We break down the inference gap into two parts:

Inference Gap = Approximation Gap + Amortization Gap

The definitions of Table 1 are general definitions for any lower bound and any variational distribution. In this paper, we focus on the VAE lower bound, $\mathcal{L}_{VAE}[q]$ (Eqn. 3). In the VAE case, we can translate the definitions to KL divergences by rearranging Eqn. 2:

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

Similarly, the approximation gap is: $\log p(x) - \mathcal{L}_{VAE}[q^*] = \text{KL}(q^*(z|x)||p(z|x))$, where $q^*(z|x) = \arg \min_{q \in \mathcal{Q}} \text{KL}(q(z|x)||p(z|x))$. In our experiments, to obtain $q^*$, we optimize the parameters of the variational distribution $q$ for each datapoint. Thus, for the FFG variational distribution, we optimize the mean and variance for each datapoint. For the Flow model, we optimize the parameters of the flow and auxiliary variable networks for each datapoint. We refer to the resulting distributions as Optimal FFG ($q^{\ast}_{FFG}$) and Optimal Flow ($q^{\ast}_{Flow}$). We approximate $\log p(x)$, denoted $\hat{\log p}(x)$, by $\max(\mathcal{L}_{AIS}, \mathcal{L}_{IWAE}[q^\ast])$.

4 Related Work

Model evaluation with AIS for decoder-based models was proposed by Wu et al. (2017). They validated the accuracy of the approach with Bidirectional Monte Carlo (BDMC, Grosse et al., 2015).
and demonstrated the advantage of using AIS over the IWAE bound for evaluation when the inference network overfits to the training data.

Much of the earlier work on variational inference focused on optimizing the variational parameters locally for each datapoint. Salakhutdinov & Larochelle (2010) perform such local optimization when learning deep Boltzmann machines. The original Stochastic Variational Inference scheme (SVI, Hoffman et al. (2013)) also specifies the variational parameters to be optimized locally in the inner loop. Krishnan et al. (2017) remark on two sources of error in variational learning with inference networks, and propose to optimize locally with 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. Yet, their work only dwells on reducing the amortization gap during training and does not touch on analyzing the approximation gap caused by using the Gaussian family.

Even though it is clear that failed inference would lead to a failed generative model, little quantitative assessment has been done showing the effect of the approximate posterior on the true posterior. Burda et al. (2016) visually demonstrate that trained with an importance-weighted approximate approximate, the resulting true posterior is more complex than those trained with fully-factorized Gaussian approximations. We extend this observation quantitatively in the setting of flow-based approximate inference in VAEs.

5 Experimental Results

5.1 Intuition through Visualization

To begin, we would like to gain some insight into the properties of inference in VAEs by visualizing different distributions in the latent space. To this end, we trained a VAE with a two-dimensional latent space and in Fig. 2 we show contour plots of various distributions. The first row contains contour plots of the true posteriors \( p(z|x) \) for four different training datapoints (columns). We’ve selected these four examples to highlight different inference phenomena. The amortized FFG row refers to the output of the recognition net, in this case, a fully-factorized Gaussian (FFG) approximation. Optimal FFG is the FFG that best fits this datapoint’s posterior. Optimal Flow is the optimal
Figure 3: 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.

5.2 THE INFERENCE GAP

How well is inference done in VAEs during training? Are we close to doing the optimal or is there much room for improvement? To answer this question, we quantitatively measure the inference gap: the gap between the true marginal log-likelihood and the lower bound. This amounts to measuring how well inference is being done during training. Since we cannot compute the exact marginal log-likelihood, we estimate it using the maximum of any of its lower bounds, described in 3.2.

Fig. 3a shows training curves for a FFG and Flow inference network as measured by the VAE, IWAE, and AIS bounds on the training and test set. The inference gap on the training set with the FFG model is 3.01 nats, whereas the Flow model is 2.71 nats. Accordingly, the plot shows that the training IWAE bound is much tighter for the Flow model compared to the FFG. Due to this lower inference gap during training, the Flow model achieves a higher AIS bound on the test set than the FFG model.

To demonstrate that strong inference can be achieved, even with FFG, we train a 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 show in Fig. 3b. Even with FFG, inference is so strong to the point that the AIS and IWAE bounds are overlapping. Yet, the model is overfitting as seen by the decreasing test set bounds.

5.2.1 ENCODER AND DECODER OVERFITTING

We will begin by explaining how we separate encoder from decoder overfitting. Decoder overfitting is the same as in the regular supervised learning scenario, where we compare the train and test error. To measure decoder overfitting independently from encoder overfitting, we use the AIS
bound since it is encoder-independent. Thus we can observe decoder overfitting through the AIS test training curve. In contrast, the encoder can only overfit in the sense that the recognition network becomes unsuitable for computing the marginal likelihood on the test set. Thus, encoder overfitting is computed by: $L_{AIS} - L_{IW}$ on the test set.

For the small dataset of Fig. 3b it clear that there is significant encoder and decoder overfitting. A model trained in this setting would benefit from regularization. For Fig. 3a, the model is not overfit and would benefit from more training. However, there is some encoder overfitting due to the gap between the AIS and IWAE bounds on the test set. Comparing the FFG and Flow models, it appears that the Flow does not have a large effect on encoder or decoder overfitting.

5.3 AMORTIZATION VS APPROXIMATION GAP

Here we will compare the impact on inference that amortization has compared to the posterior approximation. The evaluation is performed on a subset of 100 datapoints due to the computational cost of optimizing the local parameters for each datapoint. Table 2 are results from training on MNIST and Fashion MNIST. For MNIST, the amortization gap is slightly larger than the approximation gap on the FFG-trained model. For the flow model, although the inference gap is slightly smaller than the FFG model, a larger portion of its inference gap comes from the approximation gap. On Fashion MNIST, which is a more difficult dataset to model, the amortization gap is much larger than the approximation gap.

How does increasing the capacity of the encoder effect the gaps? We trained the same models with larger encoders. Table 3 are the results of this experiment. We see that the FFG model’s inference gap decreases and that decrease is mainly due to a reduction in the amortization gap. For the Flow model, the increased encoder size is less beneficial, but there is a decrease in the amortization gap.

5.4 INFLUENCE OF APPROXIMATE POSTERIOR ON TRUE POSTERIOR

We’ve seen that increasing the expressiveness of the approximation improves the marginal likelihood of the trained model, but to what amount does it alter the true posterior? Will a fully factorized Gaussian approximation cause the true posterior to be more FFG or is the true posterior mostly
fixed? Just as it is hard to evaluate a generative model by visually inspecting samples from the model, its hard to say how Gaussian the true posterior is by visual inspection. We can quantitatively determine how close the posterior is to a FFG distribution by comparing the Optimal FFG bound, $L_{VAE}[q^{*}_{FFG}]$, and the Optimal Flow bound, $L_{VAE}[q_{Flow}]$.

In Table 2 on MNIST, the Optimal Flow improves upon the Optimal FFG for the FFG trained model by 0.4 nats. In contrast, on the Flow trained model, the difference increases to 12.5 nats. 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. Although the generative model can learn to have a posterior that fits to the approximation, it seems that not having this constraint, ie. using a flexible approximate, results in better generative models.

### 5.4.1  ANNNEALING THE ENTROPY

Typical warm-up ([Bowman et al., 2015; Sønderby et al., 2016]) refers to annealing $KL(q(z|x)||p(z))$ during training. 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 during training. Rather than annealing the KL, we anneal the entropy of the approximate distribution $q$:

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

where $\lambda$ is annealed from 0 to 1 over training. This can be interpreted as maximum a posteriori (MAP) in the initial phase. Due to its similarity, we will also refer to this technique as warm-up.

We find that warm-up is very important for allowing the true posterior to be more complex. Table 4 are results from a model trained without the warm-up schedule. The difference between the Optimal Flow and Optimal FFG for the Flow-trained model is 1.5 nats. In contrast, when trained with warm-up in Table 2, the difference is 12.5 nats. Thus, the difference between the Optimal Flow and Optimal FFG is significantly smaller without warmup than with warmup. This suggests that, in addition to preventing the latent variable from degrading to the prior, warmup allows the true posterior to better utilize the flexibility of the expressive approximation, resulting in a better trained model.

<table>
<thead>
<tr>
<th></th>
<th>MNIST</th>
<th>Fashion MNIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_{VAE}[q^{*}_{Flow}]$</td>
<td>-90.89</td>
<td>-90.78</td>
</tr>
<tr>
<td>$L_{VAE}[q^{*}_{FFG}]$</td>
<td>-91.02</td>
<td>-92.34</td>
</tr>
<tr>
<td>$L_{VAE}[q]$</td>
<td>-94.31</td>
<td>-94.29</td>
</tr>
<tr>
<td>Approximation Gap</td>
<td>1.18</td>
<td>1.37</td>
</tr>
<tr>
<td>Amortization Gap</td>
<td>3.29</td>
<td>3.51</td>
</tr>
<tr>
<td>Inference Gap</td>
<td>4.47</td>
<td>4.88</td>
</tr>
</tbody>
</table>

Table 4: Without warm-up. The columns $q_{FFG}$ and $q_{Flow}$ refer to the variational distribution used for training the model. All numbers are in nats.

### 6  CONCLUSION

In this paper, we broke down the inference gap into two parts: the amortization and approximation gaps. We explored how these gaps were affected by different datasets, different posterior approximations, and different encoder capacities. We confirmed that increasing the capacity of the encoder, there is a reduction in amortization. These gaps could serve for diagnosing inference in VAEs. We also demonstrated the effect that the posterior approximation has on the true posterior by observing how well approximations fit to models trained with different variational distributions. We also improved our understanding of how warm-up helps model optimization, by noticing that warm-up allows the generative model to better utilize the flexibility of the expressive variational distribution. Future work includes evaluating other types of expressive approximations and more complex likelihood functions.
REFERENCES


APPENDIX

6.1 MODEL ARCHITECTURES AND TRAINING HYPERPARAMETERS

$D_X$ refers to the dimensionality of the input and $D_Z$ refers to the dimensionality of the latent space. For MNIST and Fashion MNIST, $D_X = 784$.

6.1.1 2D VISUALIZATION

The VAE model of Fig. 2 used a decoder $p(x|z)$ with architecture: $D_Z - 100 - D_X$. The encoder $q(z|x)$ was $D_X - 100 - 2D_Z$, where the latent space has dimensionality of 50. We used tanh activation functions and a batch size of 50. The model was trained for 3000 epochs with a learning rate of 0.0001.

6.1.2 MNIST

Here we describe the network architectures for the MNIST experiments. The decoder $p(x|z)$ is $D_Z - 200 - 200 - D_X$. The encoder $q(z|x)$ is $D_X - 200 - 200 - 2D_Z$, where the latent space has dimensionality of 50, so it outputs a mean and variance. For the large encoder experiment of Table 3, the encoder architecture is $D_X - 500 - 200 - 200 - 2D_Z$.

For flexible model, we use two flow steps where each step contains a number of other networks:

$q(v_0|x)$: $D_X - 200 - 2D_Z$
$q(z_0|v_0, x)$: $[D_X + D_Z] - 200 - 2D_Z$
$r(v_T|x)$: $[D_X + D_Z] - 200 - 2D_Z$

We use tanh for the activation functions and batch size of 100. We use the same learning rate schedule as (Burda et al., 2016). We use the warm-up schedule described in 5.4.1, where the annealing is over the first 100 epochs.

6.1.3 FASHION-MNIST

Fashion-MNIST consists of a training and test dataset with 60k and 10k datapoints respectively. We rescale the original dataset so that pixel values are within the greyscale range, i.e. $[0, 1]$. We then binarize the whole dataset statically, so that our Bernoulli likelihood model $p(x|z)$ is valid.

The VAE model used to trained on Fashion-MNIST has the same architecture as that used to train on MNIST. All neural networks chosen to parameterize the flow have hidden layers consisting of 100 units as opposed to 200 units as in the MNIST setting:

$q(v_0|x)$: $D_X - 100 - 2D_Z$
$q(z_0|v_0, x)$: $[D_X + D_Z] - 100 - 2D_Z$
$r(v_T|x)$: $[D_X + D_Z] - 100 - 2D_Z$

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 were trained with a linearly increasing annealing constant over the first 400 epochs of training.

The activation function were chosen to be the exponential linear unit (ELU, Clevert et al. (2015)), as we observed improved performance with it over tanh. We follow the same learning rate schedule...
and train for the same amount of epochs as described by Burda et al. (2016). All models were trained with the ADAM optimizer (Kingma & Ba, 2014).

6.2 Computation of the Determinant

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 therefore be easily computed:

\[
\text{det}(Df) = \text{det}(Df_1) \cdot \text{det}(Df_2) = \left( \prod_{i=1}^{n} \sigma_1(z)_i \right) \left( \prod_{j=1}^{n} \sigma_2(v')_j \right).
\]