Improving Variational Inference with Inverse Autoregressive Flow

MLMI4: Advanced Machine Learning

Word count: 4996
CRSid: H801P, H801L, H801M

University of Cambridge

April 4, 2021

1 Introduction

Variational Auto-encoders (VAE) ([4]) provide a popular framework for efficient, unsupervised training of Generative models. To train a Generative model $p_\theta(\boldsymbol{x})$, VAEs introduce a parametric Inference model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$, which approximates the true latent posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$. Parameters of the Inference and Generative models are then jointly optimised using an Evidence Lower Bound (ELBO) objective function, presented in Equation 1, which is a tractable lower bound of log marginal likelihood. The original work on VAEs ([4]) uses a simple isotropic multivariate Gaussian distribution for the inference model[1], which will often poorly approximate the true latent posterior. As a result, $D_{KL}\left(q(\mathbf{z} \mid \mathbf{x})||p(\mathbf{z} \mid \mathbf{x})\right)$ term could be large, limiting the tightness of the ELBO, and thus the accuracy of our parameter estimates in $p_\theta(\boldsymbol{x})$

$$\log p(\mathbf{x}) = \underbrace{\mathbb{E}_q[\log p(\mathbf{x} \mid \mathbf{z})] - D_{KL}\left(q(\mathbf{z} \mid \mathbf{x})||p(\mathbf{z})\right)}_{ELBO} + D_{KL}\left(q(\mathbf{z} \mid \mathbf{x})||p(\mathbf{z} \mid \mathbf{x})\right) \tag{1}$$

In this report we replicate the follow up work by [5] "Improving Variational Inference with Inverse Autoregressive Flow", which aims to improve tightness of the ELBO by increasing the flexibility of the Inference model. This is achieved by transforming a simple approximate posterior distribution $q_\phi(z|x)$ into a more flexible one via a series of invertible nonlinear autoregessive transforms ([2]) in a process called Normalizing Flow ([9]). With a more flexible approximate posterior, the KL divergence term between approximate and true posteriors can approach values closer to 0 during SGD optimisation, leading to a tighter lower bound and a higher $p(\boldsymbol{x})$.

[1] Authors note, however, that Gaussian distribution is chosen for simplicity and other distributions could be used instead.

This report is outlined as follows. In Section 2 we introduce ideas of generative modelling and Variational Bayes to motivate the work behind the standard VAE framework ([4]). In Section 3 we present the theory of how Inverse Autoregressive Flows ([5]) can achieve a more flexible latent posterior distribution. In Section 4, we replicate experiments conducted in the original paper and provide additional insight on merits of IAF-VAE framework. More specifically, we implement: (1) standard-VAEs, (2) simple IAF-VAE with MADE masking, and (3) bidirectional ResNet VAE with PixelCNN masking, and test these on MNIST and CIFAR-10 datasets. Additional details on architectures are presented in Appendix. Finally, we draw our conclusion in Section 5.

2 Background

2.1 Generative Modelling

Generative models are useful whenever one aims to understand the underlying process generating the data, wants to synthesise new data using the model, or conduct probabilistic inference about any of the observed variables. They are especially attractive because they learn in an unsupervised fashion which makes them applicable to a vast amount of unlabeled data. The goal of generative modeling can be formalised in Equation 2; we want to learn a model $p_\theta(\boldsymbol{x})$, parametrised by θ, such that it represents the true distribution of data $p^*(\boldsymbol{x})$.

$$p_\theta(\boldsymbol{x}) \approx p^*(\boldsymbol{x}) \tag{2}$$

Most of the time the data we want to generate has some underlying structure, some set of latent random variables which take part in the data generating process and define how density of $p^*(\boldsymbol{x})$ is distributed. For MNIST digit dataset, these latent variables could be: digit identity, tilt, stroke width etc. Assuming this structure we can factorise joint distribution over observed variables $\boldsymbol{x}$ and latent variables $\boldsymbol{z}$ as a directed Bayesian Network, presented in Equation 3. The likelihood function $p_\theta(\boldsymbol{x}|\boldsymbol{z})$, in VAE framework is defined by a neural network which, given its high expressiveness, can estimate any complex mapping from latent to observable space. The latent prior could itself be a complex distribution with a hierarchy of conditional dependencies between latent variables, for example in, MNIST the stroke and tilt can be both highly dependent on speed of writing. How to formulate $p_\theta(\boldsymbol{z})$ is not always straightforward and may require some inductive bias.

$$p_\theta(\boldsymbol{x}, \boldsymbol{z}) = p_\theta(\boldsymbol{x}|\boldsymbol{z})p_\theta(\boldsymbol{z}) \tag{3}$$

Fortunately, the standard VAE makes only weak assumptions about the latent space. It doesn't require prior knowledge about the structure or the exact number of latent variables. It assumes that $\boldsymbol{z} \sim N(\boldsymbol{0}, \boldsymbol{I})$ and that the exact number of latent variables and their dependencies can be learned by the early nonlinear layers of the decoder NN $p_\theta(\boldsymbol{x}|\boldsymbol{z})$, with later layers mapping the proper latent space to the observed variables.

2.2 Learning in Latent Variable Models

In order to learn $p_\theta(\boldsymbol{x})$ we need to optimise parameters θ of the generative model $p_\theta(\boldsymbol{x}|\boldsymbol{z})$. Usually this could be done using the maximum likelihood approach via gradient ascent,

which is equivalent to maximising the expectation of the model likelihood over the distribution $p_\theta(\boldsymbol{z})$, as presented in Equation 4.

$$\underset{\theta}{\operatorname{argmax}}\, p_\theta(\boldsymbol{x}) = \underset{\theta}{\operatorname{argmax}} \int p_\theta(\boldsymbol{x}|\boldsymbol{z}) p_\theta(\boldsymbol{z}) dz \tag{4}$$

However, $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ is a complex distribution so $p_\theta(\boldsymbol{x})$ has no closed form solution. Moreover, $\boldsymbol{z}$ is a continuous variable and conducting integration with respect to $\boldsymbol{z}$ for a complex $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ is computationally prohibitive. One way to overcome this problem would be to sample N data points from a prior $p_\theta(\boldsymbol{z})$ and use a Monte Carlo unbiased estimate of log marginal likelihood as presented in Equation 5. However, Monte Carlo estimate will have a high variance as most of the samples from $p(\boldsymbol{z})$ won't overlap with regions of high density of likelihood $p_\theta(\boldsymbol{x}|\boldsymbol{z})$, which is a complex function. As a result, the estimate of $\log p_\theta(\boldsymbol{x})$ will be based only on a fraction of samples and therefore, to obtain a confident estimate, the number of samples, N, would have to be very large.

$$\log p_\theta(\boldsymbol{x}) \approx \frac{1}{N} \log \sum_{n=1}^{N} p_\theta(\boldsymbol{x}|\boldsymbol{z}^{(n)}) \tag{5}$$

The closer the proposal distribution (the one we sample from) is to the true posterior, the lower the variance of the estimator. Therefore, we could turn towards MCMC-based estimators such as Hamiltonian MCMC which express log marginal likelihood as an expectation over the true posterior. Unfortunately, these methods are still computationally expensive and are often difficult to tune. Alternatively to sampling-based methods, we can utilise variational methods which formulate inference of latent samples as an optimisation problem. This method is the one used in VAE framework which we will explore in the following sub-section.

2.3 Variational Bayes in VAE framework

VAEs allow for efficient training of generative models thanks to two characteristics. First, like other variational methods, it optimises a tractable evidence lower bound (ELBO) of the log marginal likelihood. Second, it introduces an inference model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ to approximate true posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$. Since the parameters of the inference model are shared across data points, parameters ϕ and θ can be jointly optimised using efficient stochastic gradient descent of the ELBO objective function. These design decision allow for efficient learning and inference, not only in comparison to sampling-based methods, but also to other more traditional variational inference methods.

Derivation of ELBO is presented in Equation 6, it uses an alternative derivation to Jensen's inequality to highlight the gap between ELBO and marginal likelihood. In line 3, we have introduced a yet undefined distribution over latent variables $q_\phi(\boldsymbol{z})$. Traditional variational methods don't constrain the shape of this distribution, however in VAE, in order to improve tightness of ELBO, $q_\phi(\boldsymbol{z})$ is replaced with the inference model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$, as presented in the last line of Equation 6. This is a better approximation of true posterior and thus will allow us to sample $\boldsymbol{z}$ which are more likely to generate x. With a $q_\phi(\boldsymbol{z})$ that can efficiently approximate the true posterior distribution we can reduce the KL divergence between $q_\phi(\boldsymbol{z})$ and $p_\theta(\boldsymbol{z}|\boldsymbol{x})$. This will improve the tightness of ELBO which

will in turn allow us train more useful generative model (tighter ELBO can optimised to find a θ estimate which more closely corresponds to the true θ^*).

$$\begin{aligned}
\log p_\theta(\mathbf{x}) &= \mathbb{E}_{q(\mathbf{z})}\left[\log p_\theta(\mathbf{x})\right] \\
&= \mathbb{E}_{q(\mathbf{z})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{z}\mid\mathbf{x})}\right]\right] \\
&= \mathbb{E}_{q(\mathbf{z})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q(\mathbf{z})}\frac{q(\mathbf{z})}{p_\theta(\mathbf{z}\mid\mathbf{x})}\right]\right] \\
&= \underbrace{\mathbb{E}_{q(\mathbf{z})}\left[\log\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{q(\mathbf{z})}\right]\right]}_{ELBO} + \underbrace{\mathbb{E}_{q(\mathbf{z})}\left[\log\left[\frac{q(\mathbf{z})}{p_\theta(\mathbf{z}\mid\mathbf{x})}\right]\right]}_{\text{KL divergence}} \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}\mid\mathbf{x})}\right]\right] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{q_\phi(\mathbf{z}\mid\mathbf{x})}{p_\theta(\mathbf{z}\mid\mathbf{x})}\right]\right]
\end{aligned} \tag{6}$$

VAEs are efficient because they can jointly optimise ELBO with respect to both ϕ and θ. For each parameter update, we optimise ELBO by improving the estimate of θ, but also improve the tightness of ELBO by optimising ϕ. The learning process is efficient, because the mini-batch size in SGD update, doesn't have to be large. The Monte Carlo estimate of expectation over $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ has much lower variance than the naive Monte Carlo method which sampled over $p(\boldsymbol{z})$.

2.4 ELBO tightness

We have already established that tightness of ELBO is a crucial consideration for making optimisation of θ meaningful for the original objective function. It can be improved by either increasing flexibility of generative model $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ (which we do by employing a flexible NN), or by increasing flexibility of approximate posterior distribution. The original work on VAE uses an isotropic multivariate Gaussian distribution, which has limited representative power to match the often complex shape of the true posterior. Such choice is made because, in addition to flexibility, we also need the approximate posterior to be: (1) easy to sample from, (2) easy to compute, (3) differentiable, and (4) paralisable for high dimensional $\boldsymbol{z}$. These properties of $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ are necessary to make VAE training possible (differentiability) and efficient (reasons 1,2,4). Several methods for improving approximate latent posterior distribution were proposed, such as Auxiliary Latent Variables and Normalising Flows, in this report we explore and present the Inverse Autoregressive Flow method which improves flexibility of $q_\phi(\mathbf{z}\mid\mathbf{x})$, while keeping VAE training possible and efficient.

3 VAE with Inverse Autoregressive Flow

3.1 Motivation

The gap between ELBO and marginal likelihood is the KL divergence term presented in Equation 7. This effectively means that the error between training on the ELBO rather than on the likelihood is the KL divergence between the encoder and the inverse of the decoder.

$$\log(p_\theta(\boldsymbol{x})) - ELBO = D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z}|\boldsymbol{x})) \tag{7}$$

The intent in training the VAE on the ELBO is that the network will learn to minimise the divergence term and hence effectively train on the likelihood. This is however not the case in practice since the encoder is fixed to be an isotropic Gaussian. The inverse of the decoder, using Baye's rule, can be seen to result from the multiplication of the prior (generally an isotropic Gaussian) and the likelihood. This will not in the general case result in a simple unimodal distribution. For instance, for a VAE model trained on a binarised data set such as MNIST, the inverse of the decoder can be seen to be expressed by equation 8. In effect, this results in the multiplication of the isotropic Gaussian prior over the latent variables and a Bernoulli distribution over each dimension of the vector outputted by the decoder neural network (using the latent variables as inputs). This clearly does not reduce to an isotropic Gaussian. The encoder is hence prevented from training to minimise the KL divergence to 0. Furthermore, since this does not always result in a unimodal distribution, the minimum KL divergence between the encoder and the inverse decoder might be large due to the inability of a Gaussian to approximate multimodal distributions.

$$\begin{aligned} p_\theta(\boldsymbol{z}|\boldsymbol{x}) &\propto p_\theta(\boldsymbol{z}) p_\theta(\boldsymbol{x}|\boldsymbol{z}) \\ p_\theta(\boldsymbol{z}|\boldsymbol{x}) &\propto \mathcal{N}(\boldsymbol{z}; 0, I) \prod_j \rho_j(\boldsymbol{z})^{x_j} (1 - \rho_j(\boldsymbol{z}))^{1-x_j} \end{aligned} \tag{8}$$

This inability to train on a good lower-bound of the likelihood will lead to VAEs underperforming. Adding IAF to the model aims at tackling this issue by making the the encoder more flexible. This is done while keeping the main advantages of the standard encoder such as the ability to sample from it, compute probability densities and differentiate for training. It will furthermore be shown that using IAFs in particular will lead to an ability to scale to high dimensions of the latent variable without increasing the computational cost extensively.

3.2 Inverse Autoregressive Flow

The solution introduced here to make the encoder probability more flexible is to transform it through normalising flows. When a random sample z_0 is passed through a function, the probability distribution of the transformed sample z_1 can be shown to be given by Equation 9. Note that the resulting probability distribution is simply the sum of that of the sample and the determinant of the jacobian of the function.

$$log(q_\phi(\boldsymbol{z}_1|x)) = log(q_\phi(\boldsymbol{z}_0|x)) - log(|det(\frac{d\boldsymbol{z}_1}{d\boldsymbol{z}_0})|) \tag{9}$$

This process can then be extended to applying T transforms sequentially. The probability density of the output of the last sample z_T can be seen to be expressed by equation 10.

$$log(q_\phi(\boldsymbol{z}_T|x)) = log(q_\phi(\boldsymbol{z}_0|x)) - \sum_{t=1}^{T} log(|det(\frac{d\boldsymbol{z}_t}{d\boldsymbol{z}_{t-1}})|) \tag{10}$$

This means that if the jacobian of the T transforms applied is known, the pdf of the transformed sample can be calculated. This can however become tedious if the latent space becomes of high dimensions since it requires computing the determinants of a high dimensional jacobian matrix. This is addressed here by fixing the normalising flows to be inverse autoregressive transforms. Such transforms can be seen to be illustrated in

Figure 1. Each transform can be seen to sequentially multiply the sample by σ and add μ to generate the next sample. Both σ and μ are outputs of the autoregressive neural network that takes as input the previous sample and a context vector h. Building practical autoregressive neural networks is addressed in Section 3.3, here only their mathematical properties are discussed. Autoregressive networks are built such that the parameters of the transformation for the i^{th} dimension of the latent variable only depend on the $[0, i-1]$ inputs. This is expressed in Equation 11, with the superscript denoting the indexation in the latent vector z and the subscript denoting the number of IAF transforms.

$$z_t^j = \mu_t^j(z_{t-1}^{j-1}, ..., z_{t-1}^1, h) + \sigma_t^j(z_{t-1}^{j-1}, ..., z_{t-1}^1, h) z_{t-1}^j \tag{11}$$

This therefore leads to a jacobian for each of the autoregressive transforms to be of the form given in Equation 12.

$$\frac{dz_t}{dz_{t-1}} = \begin{bmatrix} \sigma_t^1 & 0 & 0 & 0 & 0 \\ \dots & \sigma_t^2 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \dots & \dots & \dots & \sigma_t^{k-1} & 0 \\ \dots & \dots & \dots & \dots & \sigma_t^K \end{bmatrix} \tag{12}$$

Since the autoregressive transforms have lower diagonal jacobian, the log determinant from Equation 10 can be simplified to the sum of the logs of the diagonal elements of the jacobian. This is shown in Equation 13, where the pdf of the transformed sample reduce to a simple to evaluate expression.

$$log(q_\phi(z_T|x)) = log(q_\phi(z_0|x)) - \sum_{t=1}^{T}\sum_{k=1}^{K} log(\sigma_t^k) \tag{13}$$

The key result here is that the pdf of the transformed sample can now be made significantly more complicated than the simple isotropic Gaussian from a VAE. More importantly, this is achieved while keeping all four advantages of sampling from a simple isotropic Gaussian, as described in Section 2.4: (1) easy to sample, (2) easy to compute, (3) differentiable, (4) paralelizable for high dimensional $\boldsymbol{z}$.

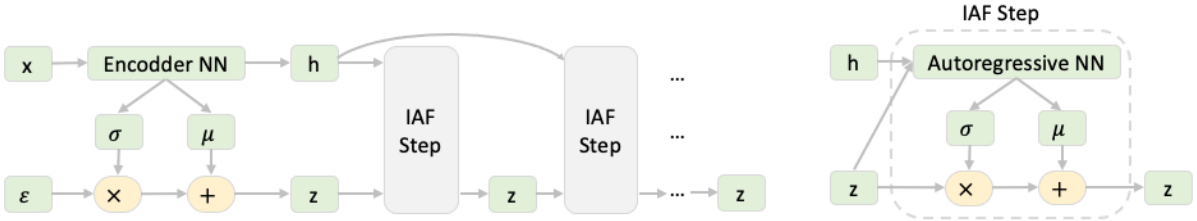


Figure 1: Schematic of inverse autoregressive transforms

Since the output distribution of samples from the encoder including IAF transform can be made highly complex, it can be concluded that optimal training will lead to a reduced KL divergence between the encoder and the inverse decoder. This should therefore tighten the gap between the ELBO and the likelihood of the data. Performance of well trained IAF-VAE should therefore be higher than that of VAE.

3.3 Building Autoregressive Neural Networks

In this Section we present methodology to build a practical inverse autoregressive network. The algorithm described here is a modified version of the one described in [2] to fit the requirements of the architecture described in [5]. The key requirements are to create a flexible neural network which is autoregressive with input $\boldsymbol{z}_{t-1}$ but not with input $\boldsymbol{h}$, outputting $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, as illustrated in Figure 2.

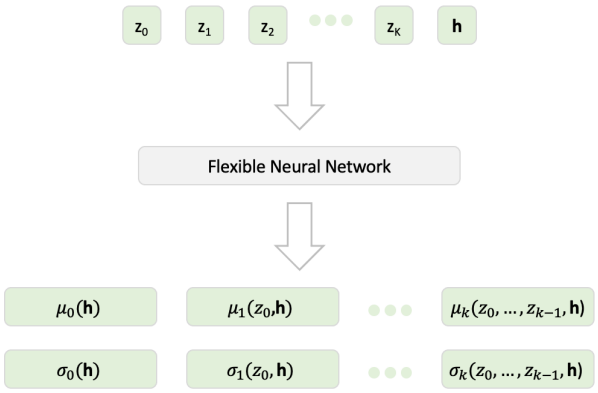


Figure 2: Requirements of autoregressive neural network for IAF transforms

The algorithm has 2 main steps. First, a fully connected network of N layers is created which takes $\boldsymbol{z}$ and $\boldsymbol{h}$ as inputs and output $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. Note that to output only positive $\boldsymbol{\sigma}$ is to take the exponential of the output of the neural network. Second, some of the connections of the fully connected layers are "masked", enforcing the autoregressive property (implemented by setting required weights to 0). For the second step, mask matrices can be built according to Algorithm 1[2]. Mask's shape will be the same as that of weight matrices', such that weights correspond to a mask value of 0 can be artificially forced to 0.

The psude-code in Algorithm 1 can be understood to first define the largest dependency of a neuron in the neural network. Connection are then cut accordingly to ensure this. In effect if a neuron is given a value of x, then it can only be connected to neurons that were given values of x or lower. The last layer is define starting from -1 since we require μ_x and σ_x to only depend on $z_{x-1}, \ldots, z_0$. Here the connections to input h were given a value of -1 since they are independent from $\boldsymbol{z}$. An example of a mask creation can be found in Figure 3. This illustrate how the connection from the first layer of the network can be cut or kept according to the drawing of the random sample $\boldsymbol{v}_1$.

[2]Note that $\boldsymbol{v}_n$ can be drawn randomly or according to some pattern (but needs to be integers between -1 and k-1 inclusive)

Algorithm 1 Creation of masks for autoregressive neural networks

initialise:
  $\boldsymbol{v}_0 = [0, 1, 2, \ldots, k, -1, \ldots, -1]$
  $\boldsymbol{v}_{N+1} = [-1, 0, 1, 2, \ldots, k-1]$
  $M = []$
for n in range(N-1) **do**
  draw $\boldsymbol{v}_n$ randomly of length of layer n
  $mask =$ array(len($\boldsymbol{v}_{n-1}$),len($\boldsymbol{v}_n$))
  for i in range(len($\boldsymbol{v}_{n-1}$)) **do**
    for j in range(len($\boldsymbol{v}_n$)) **do**
      if $\boldsymbol{v}_{n-1}(i) \geq \boldsymbol{v}_n(i)$ **then**
        $mask(i, j) = 1$
      else
        $mask(i, j) = 0$
      end if
    end for
  end for
  $M+ = [mask]$
end for
return M

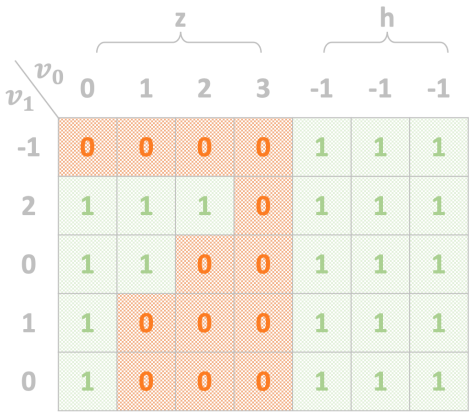


Figure 3: Example of a mask creation for the first layer of an autoregressive neural network using algorithm 1

4 Experiments

4.1 Overview

In our experiments we seek to both replicate the results from Kingma et al. [5], as well as providing further analysis. We test models on the binirised MNIST dataset ([10]) and CIFAR-10 dataset ([6]).

The MNIST digit dataset is relatively simple, and we utilise it to (1) provide visual illustrations of the inner workings of IAF-VAEs, (2) replicate results using the same model architecture as the original paper and (3) investigate whether having a learnt versus constant scaling factor output for the IAF transform has an effect.

The CIFAR-10 dataset contains more semantically complex data, which makes the task of learning a useful latent representation harder. The model used by Kingma et al. [5] for the CIFAR-10 dataset, referred to as a "ResNet VAE", is highly complex - utilising a bidirectional Inference network, with residual connections and PixelCNN masking ([7]) in the IAF step. A description of the ResNet VAE, along with illustrations of it's architecture are provided in Appendix 6.2. As this model extends the model used on MNIST in a multitude of ways, it is not obvious which parts are important, although Kingma et al. [5] do provide significant analysis in their Appendix. We were curious to see how important the ResNet VAE model was for performance, relative to the simpler model used in MNIST. Thus the CIFAR experiments section is broken up into sub-experiments (1) using the simple architecture from MNIST, and (2) using the ResNet VAE.

All code[3] for this replication was written from scratch in Pytorch - without any reliance on other code for high level modules. We implemented from scratch: MADE, PixelCNN, IAF, standard-VAEs, simple IAF-VAEs, and ResNet VAEs. Our code is more modular than that provided by Kingma et al. [5] - making it in our opinion, easier to interpret. Experimental results are reported for a single run - as although reporting aggregated statistics across many runs would have provided more accurate results, the computational costs were prohibitive. For many models we found that a more fine-grained training procedure was required than that used in the original paper. Specifically we added early-stopping to prevent overfitting, and learning-rate annealing to allow for fine-grained parameter tuning. Within each experiments sub-section we give broad descriptions of the specific models used, and provide detailed descriptions in the Appendix.

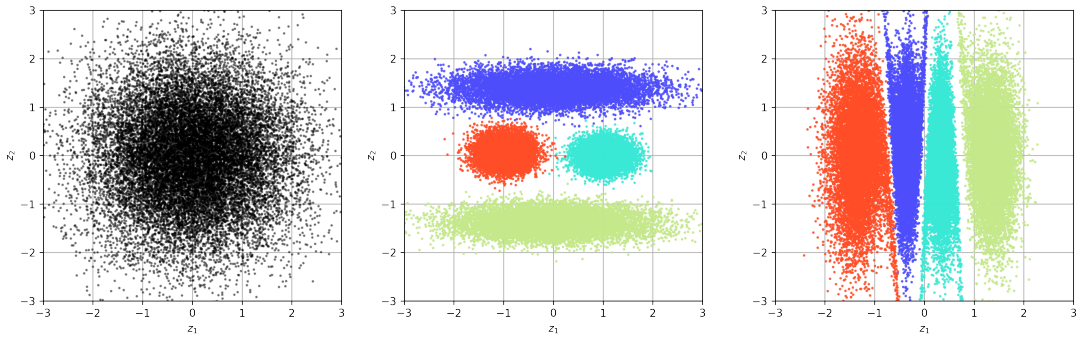
4.2 MNIST

Experiment #1: Visualising flexible posterior

As mentioned in the theory section, IAF allows for more flexibility in the approximate posterior, which decreases its KL divergence with both the true posterior and the latent prior. We confirm these theoretical properties of IAF with a visualisation of how IAF actually modifies the shape of the approximate posterior. Replicating the approach by Kingma et al. [5], we create a toy dataset containing only 4 images from MNIST, and

[3] All code for this report is available at https://github.com/lollcat/Autoencoders-deep-dive/tree/Pytorch

use it to train VAE and IAF-VAE models with a two dimensional latent space. Figure 4 visualises (a) the latent prior and (b) the latent posterior for a Guassian-VAE and (c) the latent posterior for IAF-VAE. Two observations can be made here, which support our theoretical expectations. First, IAF posterior is not constrained to elliptical shapes which allows it to better match true posterior and improve tightness of ELBO. Second, the closer $q(\boldsymbol{z}|\boldsymbol{x})$ resembles $p(\boldsymbol{z}|\boldsymbol{x})$ the better the density of approximate posterior will match that of a prior and thus reduce the regularisation term of ELBO. This follows from the expectation over the samples x fed into the encoder: $q(\boldsymbol{z}) = \int q(\boldsymbol{z}|\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}$, as $q(\boldsymbol{z}|\boldsymbol{x}) \to p(\boldsymbol{z}|\boldsymbol{x})$, then $q(\boldsymbol{z}) \to p(\boldsymbol{z})$.



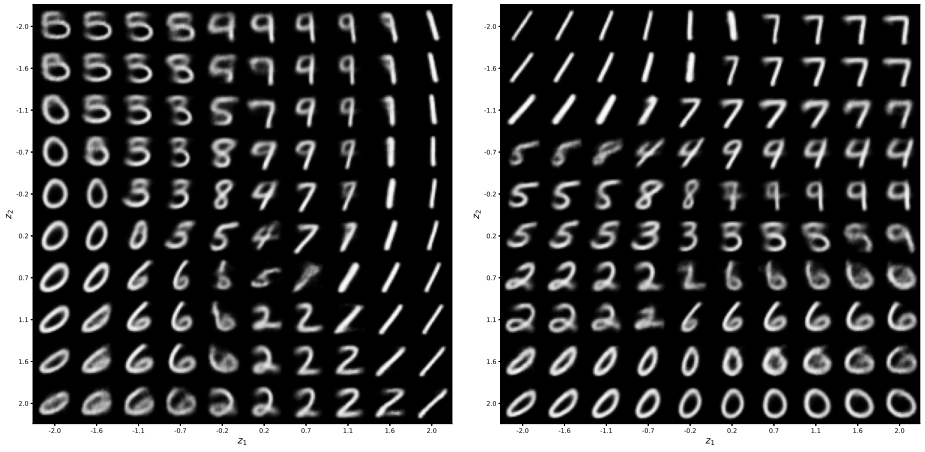
(a) Gaussian prior **(b)** Isotropic Guassian posterior **(c)** IAF modified posterior

Figure 4: Visualisation of prior and latent posteriors for VAE and VAE-IAF when trained on MNIST dataset with only 4 digits.

To better visualise the significance of low KL divergence between the approximate posterior and latent prior, in Figure 5 we plot digit reconstruction overlayed on 2D latent space (using full MNIST dataset). It shows how reconstructed digits vary as $\boldsymbol{z}$ latent sample changes along its 2 dimensions. We expect the guassian-VAE's reconstructed images to be more blurry, especially in those regions of 2D latent prior where approximate posterior has low density. Looking at Figure 5 this is not immediately obvious. Nevertheless, there are a few areas where the guassian-VAE hasn't determined which digit to draw, for example, at the top where 5 meets with 9, or below the center where latent representations of 4,6,1, and 2 meet together. We believe that with more semantically complex datasets, such as CIFAR-10, the significance of flexible approximate posterior will be much more tangible.

Experiment #2: Direct replication of results

We now seek to replicate the main results presented by Kingma et al. [5] on the binirised MNIST dataset (Salakhutdinov and Murray [10]), using identical model architectures to the original paper. We use the same symmetric encoder and decoder structure, and vary the number of IAF steps applied to the base encoder diagonal Gaussian output distribution. Each IAF step is composed of a 2-layer MADE autoregressive neural network. Multiple IAF transforms are stacked sequentially, with order reversing of the latent samples between steps. The encoder and decoder is composed of a sequence of 2-strided and 1-strided ResNet blocks, where the decoder utilises transposed-convolutions. For each



(a) VAE with gaussian posterior **(b)** VAE with 4 IAF steps

Figure 5: Comparison of digit reconstruction over 2D latent space.

pixel, the decoder outputs a probability corresponding to the probability that the pixel takes a value of 1, under the Bernoulli distribution. A more detailed description of the model architecture is given in Appendix 6.1

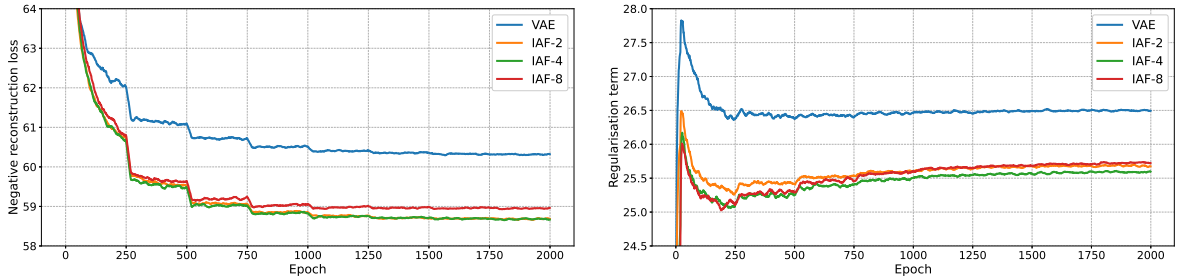
Table 1 compares the original results, and our replication. As expected, IAF-VAE has a tighter ELBO than the standard VAE and thus a higher marginal likelihood. Secondly, we see that more IAF steps result in a progressively tighter ELBO and a lower marginal likelihood - as the multiple IAF steps continue increasing the flexibility of the latent posterior. Our results do however differ from the original paper - which has clearly superior performance. This is most likely due to better hyperparameter settings, longer train time, and slight differences in implementation. Also, we don't see improved performance when using 8 IAF steps. This is most likely caused by higher sensitivity of deeper models to selection of hyperparameters and initialisation. We have also noticed some overfitting for the VAE with 8 IAFs step, which was interesting behavior as VAEs are typically more susceptible to under-fitting.

Table 1: Paper replication of VAE IAF MNIST results. Marginal likelihood estimated using importance sampling, using 128 samples (in both the original and our replication).

	Replication			**Original**		
Model	**VLB**	$\approx \log p(x)$	Δ	**VLB**	$\approx \log p(x)$	Δ
Diagonal covariance	86.80	83.22	3.58	84.08	81.08	3.00
IAF (Depth: 2, Width: 320)	84.40	81.58	2.82	82.02	79.77	2.25
IAF (Depth: 2, Width: 1920)	84.34	81.56	2.78	81.17	79.30	1.87
IAF (Depth: 4, Width: 1920)	84.22	81.49	2.73	80.93	79.17	1.76
IAF (Depth: 8, Width: 1920)	84.70	81.69	3.01	80.80	79.10	1.70

Figure 6 shows how (a): the negative reconstruction term and (b): the regularisation term, of ELBO (both of which we want to minimise) evolve during training. This helps to further visualise benefits IAF offers in minimisation of ELBO. It also shows that the regularisation term starts to increase after epoch 250. This is a direct effect of necessity to establish a trade-off between constraining approximate posterior to resemble latent prior and having flexibility in use of latent space to allow for easier, error free

reconstruction.



(a) Negative reconstruction loss

(b) Regularisation term

Figure 6: Comparison of training evolution for different VAE and VAE-IAF models.

Figure 7 compares performance of VAE and VAE with 4 IAF steps on reconstruction of 10 digits. This plot shows that both both models are able to reconstruct digits with high accuracy, with pixel wise errors present primarily at the parameter of digits. This suggests that Gaussian posterior may be sufficient for datasets which are semantically simple and application of IAF could prove more useful for datasets with more a complex true latent posterior.

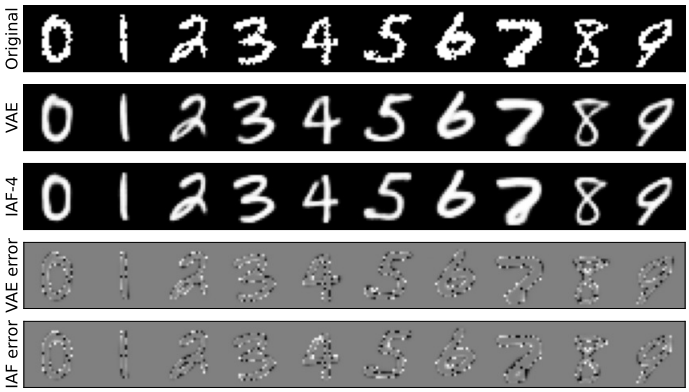


Figure 7: Images from top row to bottom: (1) original digits; (2, 3) digits reconstructed by VAE and VAE with 4 IAF steps; (4, 5) reconstruction pixel wise errors.

Experiment #3: Effect of learn-able versus constant sigma

Kingma et al. [5] provide an experiment where they utilise the IAF transforms with a constant rescaling term ($\sigma_t = 1$)[4] for their ResNet VAE (bidirectional inference VAE) on the CIFAR-10 dataset. Their results showed that the difference between having a learn-able versus constant σ was negligible. We found this to be a somewhat surprising - given that the rescaling contribution of σ to IAF seems to be important. We decided to perform the same test, but with the more simple model used for MNIST. The results of this experiment, given in Table 2 show that a learnt scaling factor yields better performance for the MNIST VAE models.

This may be a result of the different priors of the models: where the ResNet VAE model has a learnable prior, while the MNIST model's prior is fixed ($\mathrm{N}(0, 1)$). If z is output from an IAF step with constant σ, then the marginal $q_\phi(\boldsymbol{z})$, will almost certainly not be normalised $N(0, 1)$ - as the μ term has a complex dependency on other elements of z, and will contribute to the variance of $q_\phi(\boldsymbol{z})$ once we marginalise out over x. Thus for the MNIST IAF-VAE model σ is important, as the model needs precise control over the posterior in the latent space, to get it to fit well with the prior. In the case of the ResNet VAE used for CIFAR, this is less of an issue, as the more flexible prior $p_\theta(\boldsymbol{z})$ can be updated to be similar to whatever the marginal $q_\phi(\boldsymbol{z})$ is. To illustrate this one can imagine the task of choosing the best IAF transform for the 4 point toy problem (Figure 4), given the fixed prior. When the posterior z sample is close to the "boundary" between the distribution corresponding to the x point it was conditioned on and the distribution of one of the other points, the variance needs to be set to be very low to prevent distribution overlap for accurate reconstruction. However, with σ set to 1, the IAF would not be able to express this degree of control over the posterior. Thus in cases where the prior is fixed, the autoregressive neural network should output both μ and σ.

Table 2: Comparing the effects of learnt vs constant σ in IAF transform for MNIST dataset

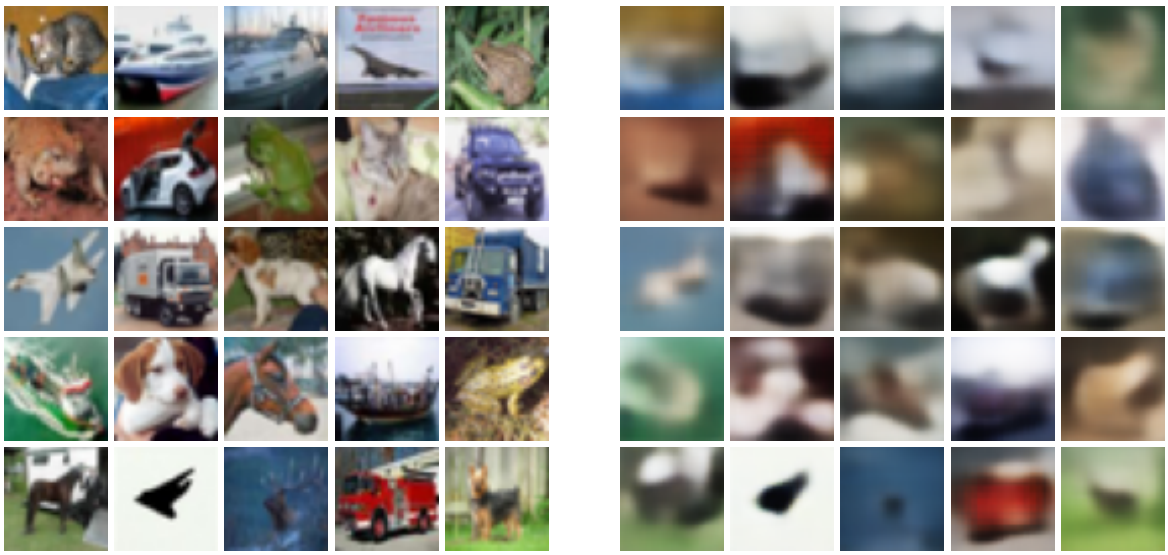
	Learnt σ			**Constant σ**		
Model	**VLB**	$\approx \log p(x)$	Δ	**VLB**	$\approx \log p(x)$	Δ
IAF (Depth: 2, Width: 320)	84.40	81.58	2.82	84.67	82.02	2.65
IAF (Depth: 2, Width: 1920)	84.34	81.56	2.78	84.99	82.27	2.72
IAF (Depth: 4, Width: 1920)	84.22	81.49	2.73	84.83	82.13	2.70
IAF (Depth: 8, Width: 1920)	84.70	81.69	3.01	85.18	82.42	2.76

4.3 CIFAR

Experiment #1 Simple baseline Model

For the baseline CIFAR results we use the same models and training regime as used in the MNIST experiments. However, for the output we use a discretized logisitc likelihood distribution, and the decoder therefore outputs a mean and variance for each pixel dimension ($32 \times 32 \times 3$), also used by [5]. Similarly to the MNIST section, we found the models required early stopping to prevent overfitting (especially for the 8-IAF-step VAE), and utilised the same learning rate schedule and early stopping criterion.

[4] σ_t is from the IAF transform equation $\mathbf{z}_t = \boldsymbol{\mu}_t + \boldsymbol{\sigma}_t \odot \mathbf{z}_{t-1}$.



(a) Original **(b)** Reconstruction

Figure 8: Reconstruction vs original for best performing model IAF (Depth:2, Width: 320)

Table 3 shows the results for the baseline CIFAR experiment. The performance from these experiments was quite noisy - making the results less clear[5]. Table 3 shows a less clear trend than MNIST, however generally the VAE with IAF steps does perform better than the Guassian VAE, with a better marginal likelihood. Comparing samples of the original images to their reconstruction (Figure 8), under the best performing baseline model we see that the reconstruction is able to learn the general colours and shapes from an image, but that the images are blurry. Overall the performance is far lower than the best model from the IAF-VAE paper, which had an average of 3.11 bits per dimension. This relatively poor performance is insightful, as it shows the importance of the improvements to the model that we explore in the sections below.

Table 3: Marginal likelihood estimated using importance sampling, using 128 samples.

Model	**>bits/dim**	**≈ bits/dim**	**Δ**
Diagonal covariance	6.460	6.431	0.029
IAF (Depth: 2, Width: 320)	6.452	6.424	0.028
IAF (Depth: 2, Width: 1920)	6.456	6.427	0.029
IAF (Depth: 4, Width: 1920)	6.452	6.424	0.028
IAF (Depth: 8, Width: 1920)	6.464	6.431	0.033

Experiment #2: Adding bidirectional inference

For our final experiment we aim to use the same "ResNet VAE" structure utilised for CIFAR in the IAF paper. Specifically we utilise a bidirectional architecture, containing multiple layers of latent variables, with interaction between "top down" and "bottom up

[5]Ideally, experiments would have be repeatedly run and then averaged, however, as noted previously, this required to much compute for this report, and thus only a single run's performance is given

processes". This differs from the MNIST model in that there are multiple layers of latent variables, and that the posterior latent variables gets to "interact" with the prior. We also utilise PixelCNN masking in the auto-regressive neural network instead of MADE. PixelCNN introduces a similar style of autoregressive dependency to that used in MADE, but in application to 2D images, using a masked convolution ([7]) (details of the network structure are presented in Appendix 6.2).

The motivation behind the Resnet VAE architecture is that it (1) introduces more complex dependencies between elements on the prior and (2) that by allowing "bottom up" and "top down" processes to interact, the inference model can be directly given knowledge of the state of the generative model at the same "layer" allowing the inference model to correct the generative model with the data conditioned likelihood.
We experiment with adding different numbers of stochastic layers (referred to as "rungs" given their inspiration from the ladder network), for models with a Guassian latent posterior, and with a latent posterior output from a single IAF step. As there are many design decisions for this model, we are certain that our implementation differs from the original in some aspects - however we seek to include all of the key major components.

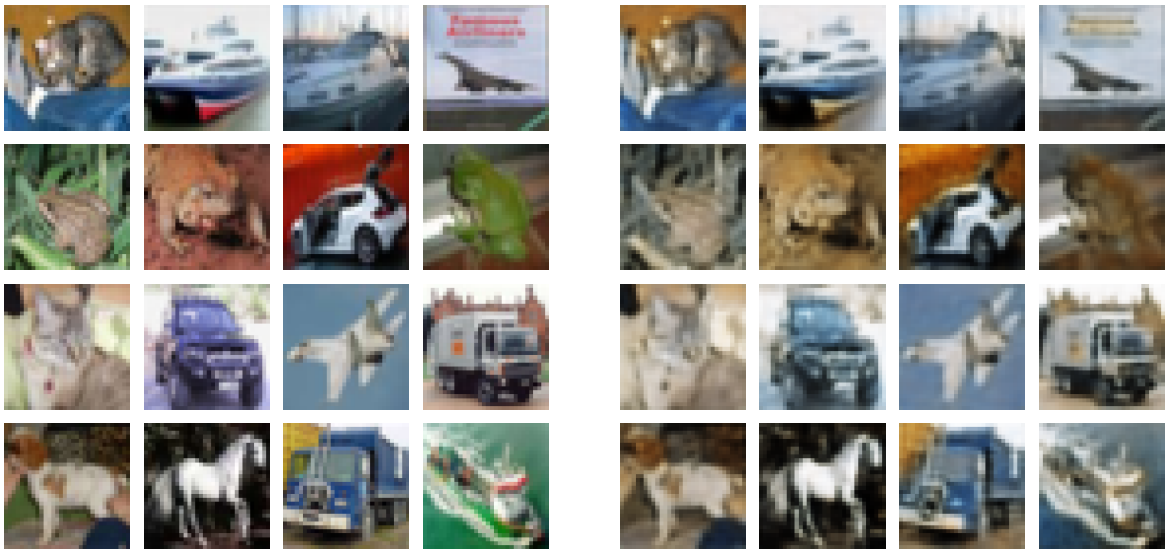
We found that this model architecture achieved more robust performance (less prone to overfit) than the architecture for MNIST. This is because the PixelCNN masked convolutional layer has a lower number of parameters than the MLP from MADE - making it less flexible, and less prone to overfitting. We confirmed this by training the same style model, but with MADE instead of PixelCNN style masking, and found it was more prone to overfitting. The increased robustness of the model means that it could be trained for longer, at a higher learning rate. In practice we only had enough time to train these models for 400 epoch - and suspect that we could obtain better performance with more training.

Table 4 shows that the Resnet VAE structure greatly improves performance relative to the baseline. In Figure 9 we see that the improved performance corresponds to a far clearer reconstruction than the baseline VAE model. Furthermore, Table 4 shows a strong performance increase with the number of rungs in the bidirectional inference network, and a sizeable increase in performance when using an IAF step to increase the flexibility of the latent posteriors. Our results are significantly worse than the best performing model in the original paper - which achieved a performance of **3.11** bits per dimension (which also utilised 20 rungs). This is most likely due to differences in design decisions and hyperparameter settings, and due to the far larger training time used in the original paper (1 million epoch).

The time required to generate a sample image from the prior (35.5 ms), and to complete a forward pass (82.6 ms) is very fast. This is similar to the 50 ms/image sampling time reported by Kingma et al. [5] for their 20 rung model. This is notably faster than other methods - for example Kingma et al. [5] report a sampling time of 52.0 s/image for the PixelCNN model ([7]).

Table 4: Results for CIFAR model with bidirectional inference and PixelCNN masked convolutional layer

Model	>bits/dim	≈ bits/dim	Δ
4 rungs Guassian	6.393	6.360	0.033
4 rungs IAF	6.081	6.050	0.031
8 rungs Guassian	5.840	5.808	0.032
8 rungs IAF	5.830	5.799	0.031
20 rungs Guassian	5.713	5.678	0.035
20 rungs IAF	5.518	5.485	0.033



(a) Original **(b)** Reconstruction

Figure 9: Reconstruction vs original for best performing model IAF with bidirectional inference and pixelCNN.

5 Conclusion

Across our experiments, we consistently saw an increased performance from VAEs that utilised IAF, with a IAF steps causing a tighter ELBO, and higher marginal likelihood $p_\theta(\boldsymbol{x})$. This replicates the key result of Kingma et al. [5], and confirms the utility of the approach.

The simple architecture used on the MNIST example, where the IAF step utilises the MADE network for the autoregressive neural network is relatively simple to implement - and we therefore recommend it as a useful addition to VAEs, that can easily improve performance. The ResNet VAE model is more difficult to implement than standard VAEs - with a more complex architecture that requires more design decisions, and results in a greater computational cost. The ResNet VAE model also has latent space with higher dimensionality, due to the multiple layers of latent variables. However the benefits introduced by this complex model, namely (1) the more complex, structured prior, and (2) interaction between the inference and generative steps are justified additions to the

VAE, that extend it's ability to learn generative models of complex data. Therefore we would recommend using the ResNet VAE model in situations where the true data generating process is complex, and where boosting performance, specifically in terms of marginal log likelihood is of utmost importance.

The literature on normalising flows has recently grown significantly - with a large host of variations available ([8]). The "comparative advantage" of IAF is that it allows a highly dependent structure to be introduced between elements of z, while allowing each element of z produced by the flow to be calculated quickly in parallel. The disadvantage of this type of flow is that they are more computationally expensive to invert - as this requires n passes of the autoregressive neural network, where n is the dimension of z. For VAEs one never has to invert the flow during training, and for high dimensional latent spaces, fast sample generation is important. Thus IAF is well suited to VAEs specifically.

References

[1] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. "Fast and accurate deep network learning by exponential linear units (elus)". In: *arXiv preprint arXiv:1511.07289* (2015).

[2] Mathieu Germain et al. *MADE: Masked Autoencoder for Distribution Estimation*. 2015. arXiv: 1502.03509 [cs.LG].

[3] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[4] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2014. arXiv: 1312.6114 [stat.ML].

[5] Diederik P. Kingma et al. *Improving Variational Inference with Inverse Autoregressive Flow*. 2017. arXiv: 1606.04934 [cs.LG].

[6] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[7] Aaron van den Oord et al. "Conditional image generation with pixelcnn decoders". In: *arXiv preprint arXiv:1606.05328* (2016).

[8] George Papamakarios et al. "Normalizing flows for probabilistic modeling and inference". In: *arXiv preprint arXiv:1912.02762* (2019).

[9] Danilo Jimenez Rezende and Shakir Mohamed. *Variational Inference with Normalizing Flows*. 2016. arXiv: 1505.05770 [stat.ML].

[10] Ruslan Salakhutdinov and Iain Murray. "On the quantitative analysis of deep belief networks". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 872–879.

[11] Tim Salimans and Diederik P Kingma. "Weight normalization: A simple reparameterization to accelerate training of deep neural networks". In: *arXiv preprint arXiv:1602.07868* (2016).

6 Appendix

6.1 MNIST model description

The encoder is composed of a sequence of ResNet ([3]) blocks, with three 2-strided Resnet blocks each with a 3×3 kernel-size, with [16, 32, 32] output-channels. Between each 2-strided Resnet block, there is 1-strided Resnet block with the same number of output-channels as the preceding 2-strided Resnet block. Finally the output from the final ResNet block is flattened into a $4 \times 4 \times 32 = 512$ dimensional vector, and passed to a fully-connected layer with 450 units, which then outputs the latent mean and log variance. The decoder has the symmetric structure to the encoder. It takes the latent samples as inputs, and starts with a hidden layer of 450 units, followed by hidden layer of 512 units which is reshaped into a $4 \times 4 \times 32$ matrix (of the same shape as the encoder ResNet output block). Which is then passed to a sequence of ResNet blocks, with 3 2-strided transposed convolutional layers, each with a 3×3 kernel-size, with [32, 16, 1] output-channels. Again, between each 2-strided transposed Resnet block, there is a 1-strided ResNet block, with the same number of output channels as the preceding ResNet block. The output of the decoder is of dimension 28×28, the same as the MNIST image dimension. Each output element corresponds to the probability that the pixel takes on a value of 0, under the Bernoulli distribution. For the 1-strided Resnet blocks we utilise identity residual connections, and for 2-strided Resnet blocks (where the input and output shape differ), we utilise the same convolution as the ResNet block, but with a 1 kernel-size. All fully-connected layers, and convectional layers are parameterized using Weight Normalisation ([11]). Exponential Linear Units ([1]) are used for all activation functions. To prevent under-fitting the encoder's output Gaussian distribution was re-parameterised to have initially low variance. The scaling factor, $\mathbf{s}$, output from Auto-regressive neural network, was re-parameterised to have an initial value nearby $+1.5$, as it was found to be beneficial in [5].

6.2 Resnet VAE model description

The ResNet VAE model uses a bidirectional inference structure, with the overall structure given in Figure 10, and the per layer structure given in Figure 11. The motivation behind this architecture is that it (1) introduces more complex dependencies between elements on the prior and (2) that by allowing "bottom up" and "top down" processes to interact, the inference model (encoder) can be directly given knowledge of the state of the generative model at the same "layer" allowing the inference model to correct the generative model with the data conditioned likelihood.

The relatively simple IAF-VAE used in MNIST, used a fixed prior defined by a unit Guassian. The ResNet VAE introduces a learnable prior, that includes more structuring, where the prior factorises autoregressively according to the ladder structure. This can be seen by example: if we have a ladder of three rungs, where we denote the latent variables corresponding to each level, z_1, z_2, z_3, then the prior is given by $p(z_1, z_2, z_3) = p(z_1|z_2, z_3) \times p(z_2|z_3) \times p(z_3)$, thus the prior includes a more complex dependency between different levels. Correspondingly each layer of the posterior distribution is conditioned on the "stochastic features" from the "higher up" layers, following a similar factorisation to the prior z_i $q(z_i|z_{>i}, x)$.

One final ingredient to the IAF-VAE is that we use PixelCNN style masking instead of MADE for the autoregressive neural network (Oord et al. [7]). PixelCNN masks, as shown in Figure 12, give the output of a convolution an autoregressive dependency, where pixels depend on the previous input pixels to the right, and above them (by masking pixels to the left and below them). We note that in Figure 11, the IAF step takes place inside the green block labelled "posterior sample".

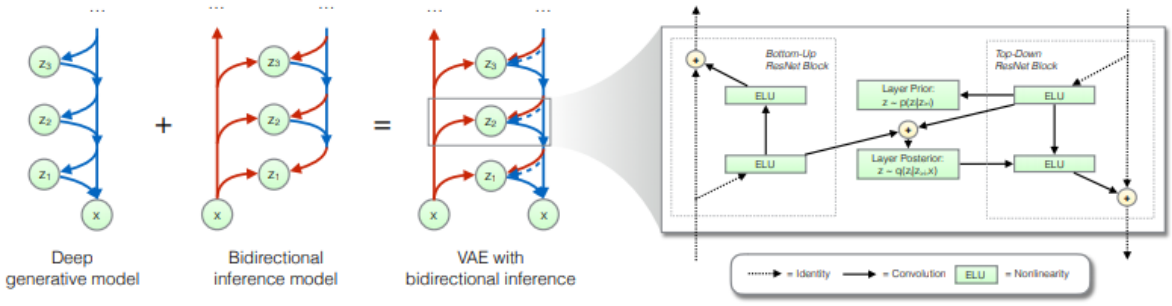


Figure 10: ResNet VAE macro structure. Image taken directly from Kingma et al. [5]

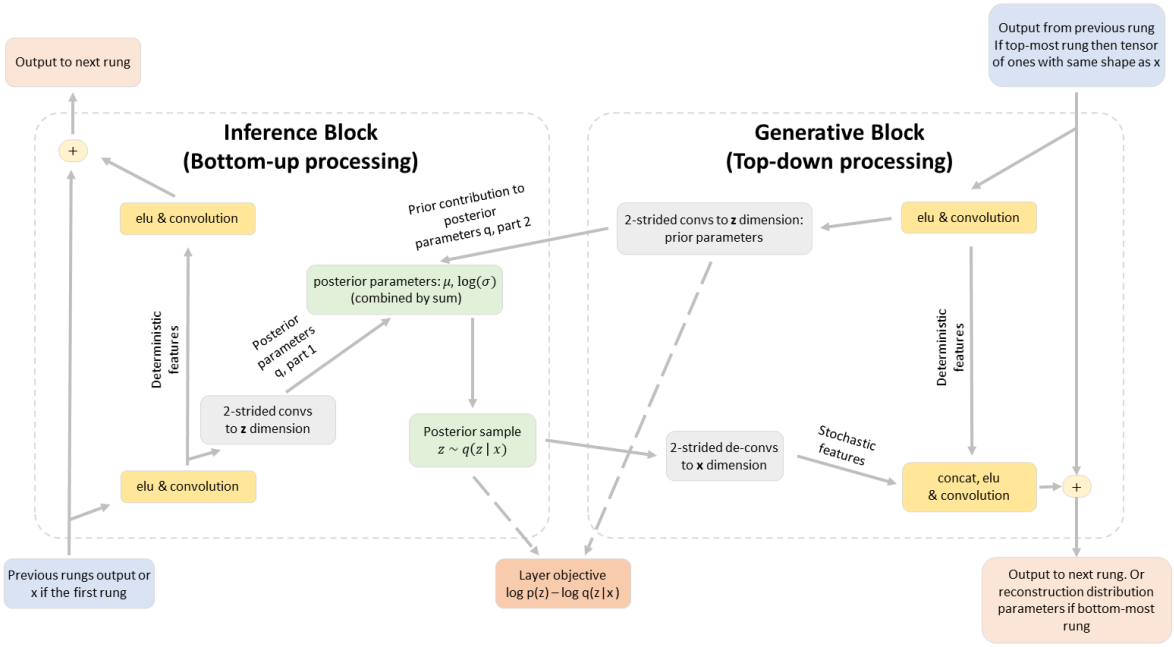


Figure 11: ResNet VAE micro computational flow schematic. Inspired by Kingma et al. [5]

Figure 12: PixelCNN mask. Type A is for the connection to the input to the autoregressive neural network. Type B is for the further hidden layers

Training bidirectional models with multiple stochastic layers is known to have issues with underfitting, where there is a stable local minima at $q(z|x) \approx p(z)$ ([5]). As per, Kingma et al. [5], we utilise the "Objective with Free bits" to prevent this, where we we modify the typical ELBO based training criterion to use a minimum amount of information per layer of stochastic variables, given by,

$$\tilde{\mathcal{L}}_{\lambda} = \mathbb{E}_{\mathbf{x}\sim\mathcal{M}}\left[\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x} \mid \mathbf{z})]\right] - \sum_{j=1}^{K} \text{maximum}\left(\lambda, \mathbb{E}_{\mathbf{x}\sim\mathcal{M}}\left[D_{KL}\left(q\left(\mathbf{z}_j \mid \mathbf{x}\right) \| p\left(\mathbf{z}_j\right)\right)\right]\right) \tag{14}$$

where j represents the grouping of stochastic variables (per layer), $\mathcal{M}$ is the minibatch over which the results are averaged over, and λ specifies the minimum nats of information per latent variable grouping. We utilise a value of $\lambda = 0.25$, as this was the default given in the code provided by Kingma et al. [5].