Improved Inference in Variational Auto-Encoders using real-NVP

Abhinav Agrawal
Roll No. 14011

Archit Sharma
Roll No. 14129

Anubhav Shrivastava
Roll No. 14114

1 Problem Description and Motivation

Generative Modelling is one of the core problems in machine learning. The advances in deep learning have brought about a revolution, increasing our capacity to model high dimensional data. Classically, the aim is to provide the estimate of probability distribution of the data, given finite training examples from the same. The data belongs to different modalities, like natural images, videos etc. A more relaxed objective, which is usually pursued in the domain of deep learning, is to get novel and realistic generations from the underlying probability distributions.

Generative models, in general, are widely applicable. At the core of it, generative modelling represents our ability to manipulate high dimensional spaces and extract meaningful representations out of it. This representations can be useful in a host of other task depending upon the domain. Solving generative tasks are fundamentally harder. For example, having controllable generation of images is fundamentally harder than classifying images (even for humans!). Thus, the representations learnt in generative modelling have been found useful in other tasks, especially when the training data available for that task is low. This is essentially why generative modelling finds great applicability in semi-supervised learning tasks or tasks with missing data. The list of specific applications of generative modelling is too long to list over here, especially considering how generally applicable it is. In Vision, generative modelling helps with various tasks such as image-to-image translation, superresolution etc. In Reinforcement Learning, generative models are often used to ”predict future states”, which helps in determining the utility of the current state. Similar applications exist in Language and Audio as well. Thus, good generative models are quintessential to the progress of machine learning in general.

One of the key aspects in generative modelling is the inference. Inference mechanisms are different for different models. For our project, we look at Variational Auto-Encoders, and try to improve the inference network using Real NVP transformations.

2 Literature Survey

[] provides a good survey of deep generative models. In context of Deep Generative Modelling, the the following two pieces of work have been really influential:

- *Generative Adversarial Networks (GANs)*: GAN [10] models implicitly learn the distribution over training data by playing a “game” between two networks. The ‘generator’ network takes random noise as input and outputs samples, usually images, whereas the ‘discriminator’ network tries to distinguish between generated samples and samples from the training distribution. As is clear, the generator and discriminator are adversaries. As both the generator and discriminator are trained, both become better at their respective tasks until a saddle point is reached. To fool the discriminator, the generator has to generate samples ‘similar’ to the training data, which gives us a generative model for the training data without explicitly specifying a training objective for it. The idea of adversarial training has gained a lot of attention, and a lot of work has gone into making GAN training stable and avoiding problems such as mode collapse.

- *Variational Autoencoders (VAEs)*: VAEs [11] fall under the larger class of Deep Latent Gaussian Models (DLGMs). While DLGMs have been known for a while, VAEs have made them really successful. Inference in DLGMs requires variational parameters for every training example. Every new datapoint requires an iterative process for inference, which makes them unsuitable for scaling up. VAEs introduced the idea of using global recognition networks which do not need to be iterated for every new datapoint. Combining the Variational Bayes objective with the reparametrization trick, VAEs can be trained in a fashion which is very similar to the classical Autoencoders (hence the name!). The difference is that autoencoders are only trained for reconstruction, while VAEs are trained to be generative models. Efficient inference and wide applicability (beyond images!) have made VAEs very successful.

These works inspired a lot of successful variants. There are a lot of other generative models which are really successful as well, like PixelRNN/PixelCNN, NADE, NICE, RealNVP, GLOs and others. We describe some of the paper we looked at while looking for a concrete problem to work on.

Real NVP [7] provides a deep invertible generator which can be trained by maximizing likelihood of training images tractably. They use real-valued non-volume preserving (real NVP) transformations. The structure of their model such that the Jacobian of the transformation is a triangular matrix, which implies the determinant of the Jacobian is simply the product of the diagonal elements. The invertibility of the deep network provides an exciting prospect with respect to achieving exact inference. However, their formulation imposes the constraint that the dimensionality of representation will be the same as that for the output.

Normalizing Flows [13] improves the inference networks in VAEs. They convert simple posteriors like $\mathcal{N}(\mu, \Sigma)$, as is the case in VAEs usually, into more complex posteriors by using a sequence of invertible transformations. The invertible transformations are cheap to compute, and have tractably computable determinant of the jacobian. Normalizing flows shows great results, and the idea has been picked up by Inverse Autoregressive Flows [12], which model the transformations as autoregressive models. We discuss Normalizing flows in our model construction.

Bayesian GAN [1], which as the name suggests, looks to model the posterior over the network parameters using MCMC techniques. There are two proposed benefits inherent in this approach: By obtaining samples from posterior, the paper counters the “mode collapse” problem (MCMC techniques would promote exploration of different modes of the posterior). The other benefit is that the training is considerably simplified, doing away with stability tricks (minibatch discrimination, feature matching) usually required for GAN training. The Bayesian model can be naturally extended into the semi-supervised domain, as has been shown in the paper. Although the paper presents a very novel framework, the results are not competitive with the contemporary papers (neither in generation quality, nor in semi-supervised learning).

Adversarially Learned Inference [6] and Adversarial Feature Learning [5] provide an inference mechanism compatible with adversarial training. In essence, both these papers add another ‘encoder’ network which is trained jointly with the ‘generator/decoder’ network. Both the networks try to fool the discriminator, which tries to differentiate between the *joint* samples from the networks. Authors in [6] show that this training procedure corresponds to matching the two joint distribution (modelled by the encoder and decoder networks). They provide competitive performance on various semi-supervised learning tasks, which justifies the presence of an inference network.

Variational Approaches for AE-GAN [2] tried to combine the benefits of VAEs and GANs. The method uses variational inference as the building tool for learning, but replaces the intractable likelihood with a synthetic likelihood. Further, the unknown posterior distribution has been replaced by an implicit distribution. Discriminators have been used to learn both, the synthetic likelihood and the implicit posterior. Ultimately, the shortcomings of both VAEs and GANs have been overcome by defining a unified loss function. A similar approach has been described in [3].

Another interesting approach towards generative modelling was proposed in ”Optimizing the Latent Space of Generative Networks” [9], where the Generator network was trained without a discriminator by correcting the noise inputs in a regularized fashion. It achieves comparable performance with visually appealing image generation. Although, the framework presents several advantages, it lacks a inference mechanism like the original GAN models, an extension of which seems non-trivial.

3 Approach and Novelty

After the literature survey, we focused our attention on the ideas developed in [7]. This was primarily motivated by their ability to provide exact and efficient likelihood computation with immense flexibility in the way transformations are carried out. Although authors in [7] answered many questions with their formulation, their approach does not allow for a change of dimensions between the variables while transforming, an idea is not well received as the images reach more real life dimensions. This warranted a framework which can exploit the flexibility of [7] while still manage to keep dimensions of latent representations under control. To keep such a framework ubiquitous we considered the model proposed in [13], where authors present a variety of flows that can be applied within their general framework 1, exploiting the archaic change of variables formula. Among the transformations proposed, the one used is a planar flow, which applies non-linear transformations as shown in 2. A departure from these seemingly simple transformation was thus warranted, but difficult to conceive since the efficient computation of Jacobian matrix is also a necessary condition. This is setting seems almost perfect for [7], and hence we undertake the task of doing an empirical study of how [7] fairs as a flow transformation.

$$\begin{aligned} q(z') &= q(z)\Big|det\Big(\frac{\partial f^{-1}}{\partial z'}\Big)\Big| \\ &= q(z)\Big|det\Big(\frac{\partial f}{\partial z}\Big)\Big|^{-1} \\ z_K &= f_K \bigcirc ... \bigcirc f_2 \bigcirc f_1(z_0) \\ \ln q_K(z_K) &= \ln q_0(z_K) - \Sigma_{k=1}^{K} \ln \Big|det\frac{\partial f_k}{\partial z_{k-1}}\Big| \end{aligned} \tag{1}$$

$$\begin{aligned} f(z) &= z + \mathbf{u}h(w^T x + b) \\ \psi(z) &= h\text{'}(w^T z + b)\mathbf{w} \\ \Big|det\frac{\partial f}{\partial z}\Big| &= |det(\mathbf{I} + \mathbf{u}^T\psi(z)^T)| \\ &= |1 + \mathbf{u}^T\psi(z)| \\ \ln q_K(z_K) &= \ln q_0(z_K) - \Sigma_{k=1}^{K} \ln|1 + \mathbf{u}_k^T\psi_k(z_{k-1})| \end{aligned} \tag{2}$$

$$\begin{aligned} p_X(x) &= p_Z\big(f(x)\big)\Big|det\Big(\frac{\partial f(x)}{\partial x}\Big)\Big| \\ y_{1:d} &= x_{1:d} \\ y_{d+1:D} &= x_{d+1:D} \odot exp\big(s(x_{1:d})\big) + t(x_{1:d}) \\ [x_{1:d} &= y_{1:d} \\ x_{d+1:D} &= \Big(y_{d+1:D} - t(y_{1:d})\Big) \odot exp\Big(- s(y_{1:d})\Big) \end{aligned} \tag{3}$$

$$\begin{aligned}
F(x) &= \mathbb{E}_{q_\phi(z|x)}[\log q_\phi(z|x) - \log p(x,z)] \\
&= \mathbb{E}_{q_0(z_0)}[\log q_K(z_K) - \log p(x, z_K)] \\
F(x)_{NF} &= \mathbb{E}_{q_0(z_0)}[\log q_0(z_0)] - \mathbb{E}_{q_0(z_0)}[\log p(x, z_K)] \\
&- \mathbb{E}_{q_0(z_0)}\left[\Sigma_{k=1}^{K} \ln|1 + \mathbf{u}_k^T \psi_k(z_{k-1})|\right] \\
F(x)_{rNVP} &= \mathbb{E}_{q_0(z_0)}[\log q_0(z_0)] - \mathbb{E}_{q_0(z_0)}[\log p(x, z_K)] \\
&- \mathbb{E}_{q_0(z_0)}\left[\Sigma_{k=1}^{K} (s_{1,k}(b \odot z_{k-1}))\right]
\end{aligned} \tag{4}$$

The non-volume preserving transformations proposed in coupling layers3, leave a part of the variable unchanged. This was remedied by implementing these in an alternating fashion which not only makes efficient use of the variables, but also preserves the efficient Jacobian computations. The formulations corresponding to the non-volume preserving flows are presented here4. We currently implement the arbitrary transformations within the coupling layer through a MLP for now but this can be easily extended to convolutional networks. This is left as a future work. The decoder and encoder network are the standard convolutional network blocks.

4 Description of Softwares

All our model were implemented in Tensorflow [14]. Tensorflow is an automatic differentiation library, which has become really popular for Deep Learning models. Using the library, one can define the forward propagation in arbitrary computational graphs, and the library will compute the backpropagation itself. It also optimizes the graph for faster computation GPU, and provides various implementations of popular optimizers like RMSProp, Adam etc.

5 Experiments

We experiment on Binarized MNIST dataset [15], which is a derived from the famous MNIST dataset. The dataset standardizes the binarization of MNIST images.

Table 1: **NF**: Normalizing Flows, **rNVP**: Real NVP. k denotes the number of transformations. Note: Lower scores are better!

Models	-log $p(x|z)$
rNVP (k = 2)	60.57
rNVP (k = 5)	75.56
rNVP (k = 10)	75.01
rNVP (k = 20)	81.37
NF (k = 4)	65.5
NF (k =10)	68.9
NF (k = 20)	75.4
NF (k = 40)	83.4

As said in the table caption, k represents the number of transformations carried. For fair comparison, a real-NVP with k transformations should be compared with $2 * k$ normalizing flows transformations. This is because in each real-NVP transformation, we use two coupling layers.

The results need some explanation. Ideally, increasing k should give better results (lower negative log likelihoods). However, since the dataset is simple and the aim is to test the validity of the approach, we were only training the model for a fixed number of epochs. While lower number of transformations were trained to appropriate levels, the same cannot be said for higher number of transformations. Also, Binarized MNIST is a simple problem. Increasing the k might be causing the models to ovefit. The key result, therefore, is that real-NVP for $k = 2$ transformations achieves

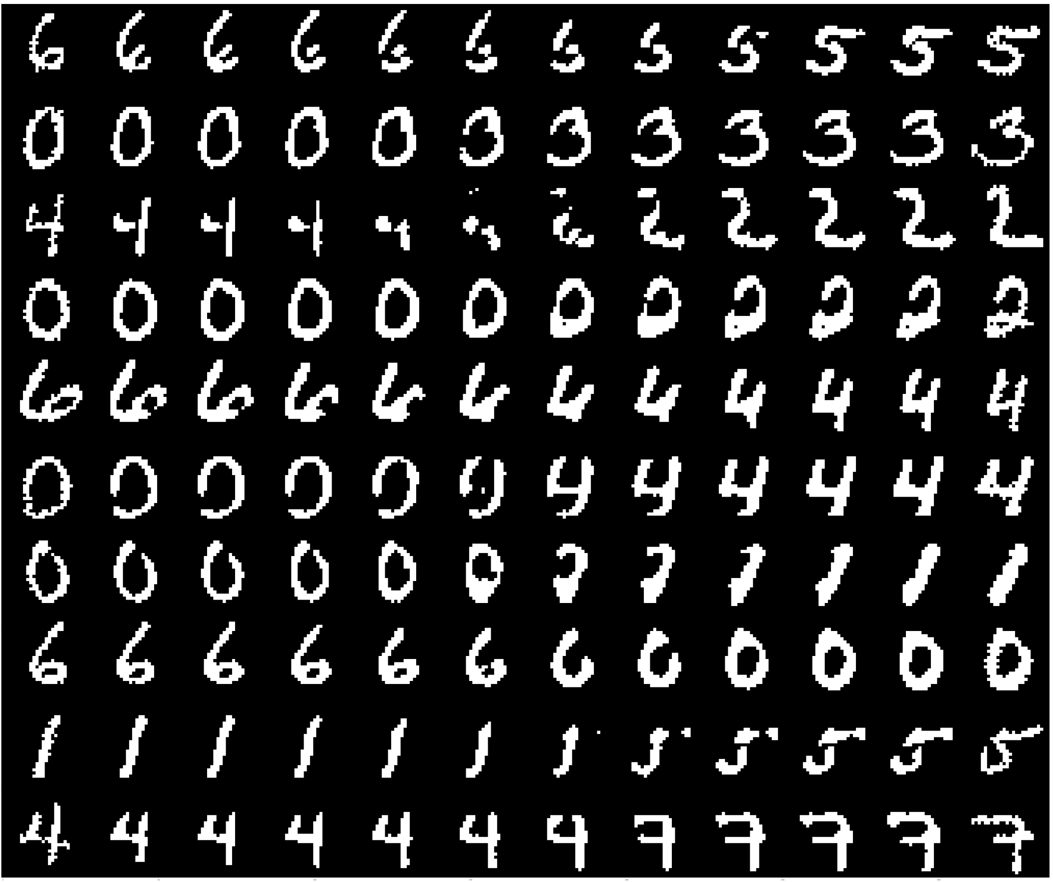

Figure 1: Interpolation of latent space from VAE

lower negative log likelihoods as compared to Normalizing flows with $k = 4$. This result serves as a motivation for the rest of the approach.

6 Things We Learnt

We learnt a lot from this project:

- We surveyed a lot of papers before deciding on a problem. This gave us a wholesome view of the field of generative modelling (though not complete yet!)
- We came up a lot of problems and possible approaches during the course of this project. However, (to our dismay sometimes!), these approaches had been tried out. Nonetheless, these approaches were more well-thought out and experimented with than we began with, thus giving us valuable lessons in research.
- We became more familiar with implementing deep learning models, going from an idea to implementation quickly. We also gained more proficiency in Tensorflow.

7 Future Work

The results from the framework look promising. The future work includes the following:

- The standard metric to compute upon is the marginal likelihood, that is $\log p(x)$. This is usually computed as follows $p(x) = \int p(x|z)p(z)dz = \int p(x|z)\frac{p(z)}{q(z|x)}q(z|x)dz \approx$

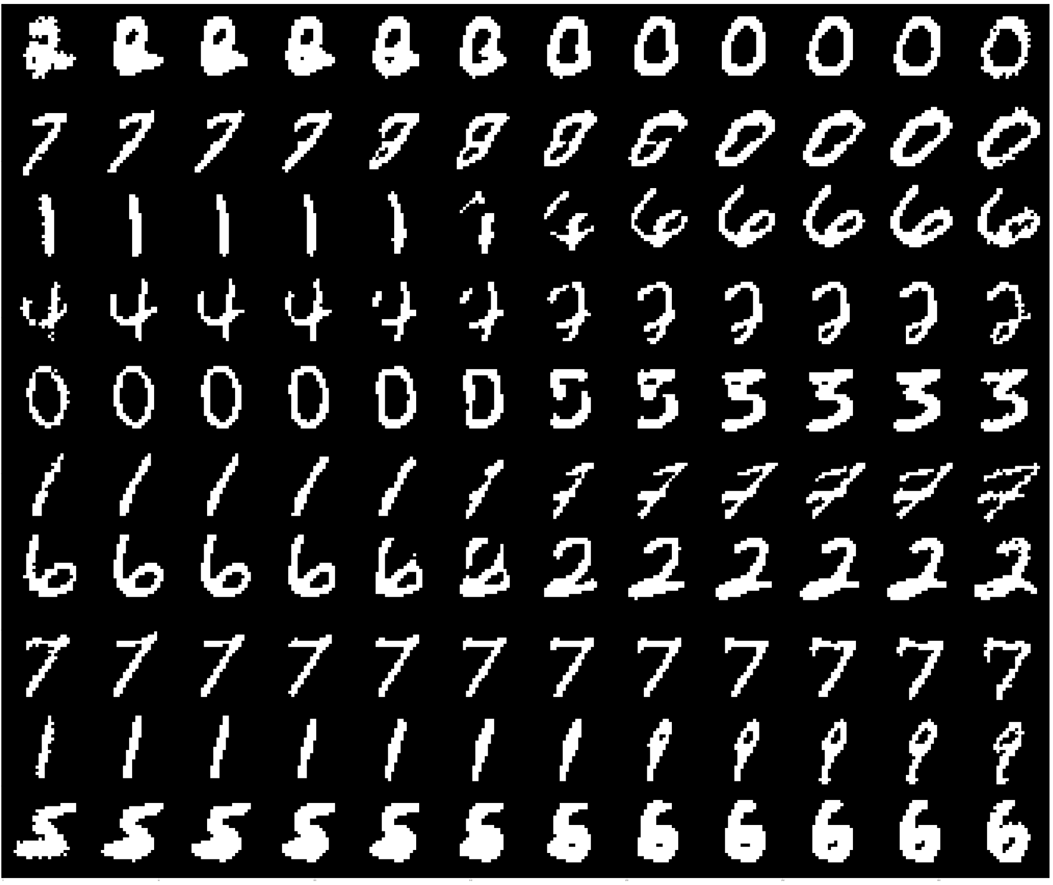

Figure 2: Interpolation of latent space from VAE with Real NVP transformations

$\frac{1}{L}\sum_{l=1}^{L} p(x|z^{(l)})\frac{p(z^{(l)})}{q(z^{(l)}|x)}$ where $z^{(l)} \sim q(z|x)$. We have implemented this metric for all models, but training and compiling results for this metric would require more time.

- Test our framework on other popular datasets such as CIFAR10, SVHM, CelebA.
- Formulate a fully convolutional network, where the real-NVP transformations are implemented as convolutions as well. Currently, the encoder and decoder are convolutional, but real-NVP transformations are done via fully connected layers. This will only be reasonable for higher resolution images.

References

[1] Yunus Saatchi and Andrew Gordon Wilson. ”Bayesian GAN” *arXiv preprint arXiv:1705.09558 (2017).*

[2] Rosca, Mihaela, et al. ”Variational Approaches for Auto-Encoding Generative Adversarial Networks.” *arXiv preprint arXiv:1706.04987 (2017).*

[3] Mescheder, Lars, Sebastian Nowozin, and Andreas Geiger. ”Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks.” *arXiv preprint arXiv:1701.04722 (2017).*

[4] Danihelka, Ivo, et al. ”Comparison of Maximum Likelihood and GAN-based training of Real NVPs.” *arXiv preprint arXiv:1705.05263 (2017).*

[5] Donahue, Jeff, Philipp Krhenbhl, and Trevor Darrell. ”Adversarial feature learning.” *arXiv preprint arXiv:1605.09782 (2016).*

[6] Dumoulin, Vincent, et al. ”Adversarially learned inference.” *arXiv preprint arXiv:1606.00704 (2016).*

[7] Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio. ”Density estimation using Real NVP.” *arXiv preprint arXiv:1605.08803 (2016).*

[8] Arjovsky, Martin, Soumith Chintala, and Lon Bottou. ”Wasserstein gan.” *arXiv preprint arXiv:1701.07875 (2017).*

[9] Bojanowski, Piotr, et al. ”Optimizing the Latent Space of Generative Networks.” *arXiv preprint arXiv:1707.05776 (2017).*

[10] Goodfellow, Ian and Pouget-Abadie, Jean and Mirza, Mehdi and Xu, Bing and Warde-Farley, David and Ozair, Sherjil and Courville, Aaron and Bengio, Yoshua. ”Generative adversarial nets” *arXiv preprint arXiv:1406.2661(2014).*

[11] Kingma, Diederik P., and Max Welling. ”Auto-encoding variational bayes.” *arXiv preprint arXiv:1312.6114 (2013).*

[12] Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, Max Welling ”Improving Variational Inference with Inverse Autoregressive Flow” *arXiv preprint arXiv:1606.04934(2016).*

[13] Danilo Jimenez Rezende, Shakir Mohamed ”Variational Inference with Normalizing Flows” *arXiv preprint arXiv:1505.05770 (2015).*

[14] Abadi, Martn, et al. ”Tensorflow: Large-scale machine learning on heterogeneous distributed systems.” *arXiv preprint arXiv:1603.04467 (2016)*

[15] Benigno Uria, Iain Murray, Hugo Larochelle ”A Deep and Tractable Density Estimator” *arXiv preprint arXiv:1310.1757 (2014)*

[16] Ivo Danihelka, Balaji Lakshminarayanan, Benigno Uria, Daan Wierstra, Peter Dayan ”Comparison of Maximum Likelihood and GAN-based training of Real NVPs” *arXiv preprint arXiv:1705.05263(2017)*