Advanced Topics in Machine Learning
Group 12 Project Report

Reproducing results from "Variational Inference with Normalizing Flows" [Danilo Jimenez, Rezende Shakir Mohamed]

1032626, 1034125, 1034129, 1036969

University of Oxford

Chapter 1

Introduction

The idea of generative models is one of the most promising approaches to Machine Learning tasks. Many such models have been used in recent years to solve the problem of inference of a posterior distribution, given some expert knowledge and assumptions on the prior distribution and using observed data to adapt these. The main obstacle in attempting to generate the true posterior is its computational intractability. Consequently, approximation methods have been employed, aiming to make computation more efficient while at the same time achieving a close estimate of the true posterior.

The paper focusses on the method of variational inference. Essentially, this converts the inference problem to an optimization one where, given a family of posterior approximations, we look for the distribution that is the closest to the true one. To measure this proximity between approximations and the true posterior, we use the Kullback-Leibler divergence. There has been lots of research in the classes of such, efficiently calculatable, approximation families, such as mean-field approximations, but all of these prove to be of limited expressiveness, leading to solutions that are far from the true posterior. In fact, there is strong evidence showing that more expressive classes of posteriors would result in better performance of variational inference, which is what motivates the work of the paper.

To address the above issue, the authors propose constructing the approximations through a normalizing flow, which applies a sequence of invertible transformations to a simple initial density until it achieves a desired level of complexity. The main types of flows considered and used in the experiments are:

1. Invertible, linear-time flows, such as planar and radial flows.

2. Finite, volume-preserving flows, such as the Non-linear Independent Components Estimation (NICE).

The goal of both of the above is to provide a transformation to the original density that ends up being rich and faithful, while at the same time allowing for efficient computation of the determinant. The first kind of flow allows for computing the logdet-Jacobian in linear (in the number of dimensions) time, while the second one gives a Jacobian with a zero upper triangular part and therefore a determinant of

1. Thus, both can be efficiently implemented and used together with a Deep Latent Gaussian Model and an inference network to solve our problem.

The paper's experiments on the binarized MNIST dataset aim to demonstrate how variational inference with normalizing flows can outperform previous methods, compare the performance of different types of flows, and discuss the relation between the length of the flow and the quality of the resulting approximation.

In our report, we replicate the original contributions of the paper by running the same experiments on MNIST. Additionally, we consider another class of flows, that of invertible convolution flows, which use the properties of circulant matrices to achieve efficient computation of the log-determinant.

Finally, we provide an open-source implementation of the above in our Github repository: `https://github.com/ATML-Group-12/normalising_flows`

Chapter 2

Variational Inference

As stated in the introduction, Variational Inference is a method of performing Bayesian inference which has become increasingly popular over the last few years. The basic idea is to approximate the posterior distribution, typically by defining a family of distributions which can be parameterised, and then optimising the "Evidence Lower Bound" (ELBO) with respect to these parameters in order to find a distribution which closely approximates the true posterior. The ELBO can be thought of as a lower bound of the marginal likelihood of a probabilistic model.

2.1 Definition

Suppose we wish to find the posterior:

$$p_\theta(\mathbf{z}|\mathbf{x}) = \frac{p_\theta(\mathbf{z}, \mathbf{x})}{p_\theta(\mathbf{x})} \tag{2.1}$$

However, to calculate $p_\theta(\mathbf{x})$ requires the calculation of an integral which is in pratice intractable:

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z}) d\mathbf{z} \tag{2.2}$$

In variational inference, we aim to approximate the posterior with a distribution $q_\phi(\mathbf{z}|\mathbf{x})$ which is parameterised by ϕ, and the aim is to find ϕ such that $q_\phi(\mathbf{z}|\mathbf{x})$ approximates $p_\theta(\mathbf{z}|\mathbf{x})$ as closely as possible. The measure of closeness we choose to use is called the KL-divergence, defined as:

$$\mathbb{D}_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) = \int q_\phi(\mathbf{z}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} d\mathbf{z} \tag{2.3}$$

2.2 The Evidence Lower Bound

In order to minimise KL-divergence, we need to be able to calculate it. However, this is also intractable since the term $p_\theta(\mathbf{z}|\mathbf{x})$ is just our original posterior. But rearranging

we get:

$$\mathbb{D}_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{x})) = \mathbb{E}_q[\log q_\phi(\mathbf{z}|\mathbf{x})] - \mathbb{E}_q[\log p_\theta(\mathbf{z},\mathbf{x})] + \log p_\theta(\mathbf{x}) \tag{2.4}$$

Since KL-divergence is always non-negative, the quantity $\mathbb{E}_q[\log p_\theta(\mathbf{z}|\mathbf{x}) - \log q_\phi(\mathbf{z}|\mathbf{x})]$ is therefore a lower bound for the marginal log likelihood $\log p_\theta(\mathbf{z}|\mathbf{x})$. We hence define:

$$\text{Evidence Lower Bound (ELBO)} = \mathbb{E}_q[\log p_\theta(\mathbf{z}|\mathbf{x}) - \log q_\phi(\mathbf{z}|\mathbf{x})] \tag{2.5}$$

and seeking to maximise the ELBO both minimises the KL-divergence between $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{z}|\mathbf{x})$, and maximises the evidence $p_\theta(\mathbf{x})$.

So to perform variational inference, we use some method of optimisation to maximise the ELBO over parameters ϕ. This is normally done using stochastic gradient descent. However, this method relies heavily on a) efficiently computing the derivative of the log likelihood $\nabla_\phi \mathbb{E}_{q_\phi(z)}[\log p(\mathbf{x}|\mathbf{z})]$ and b) having a good family of approximate posterior distributions q_θ. Specifically we need to be able to capture a diverse range of distributions while also being computationally feasible. The authors of the paper aim to address this problem using normalising flows, which we discuss later on.

2.3 Stochastic Backpropagation

Stochastic Backpropagation is a method of computing the gradient of the expected log-likelihood which was used by the authors in their models. Since this step of variational inference is not a focus of the paper, we give only a brief overview.

Stochastic backpropagation involves two main steps. First, the latent variables are re-parameterised in terms of a known distribution using a differentiable transformation. For example we might transform a Gaussian distribution onto the standard normal distribution by using a location-scale transformation.

The second step is do backpropagation with respect to the parameters ϕ, which we can now do since with our re-parameterised distribution since it has a known derivative, using a Monte Carlo method that draws from the known base distribution:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f_\theta(z)] \Leftrightarrow \mathbb{E}_{\mathcal{N}(\epsilon|0,1)}[\nabla_\phi f_\theta(\mu + \sigma\epsilon)] \tag{2.6}$$

2.4 Deep Latent Gaussian Models

Deep Latent Gaussian Model (DLGMs) are a general class of graphical models which uses a hierarchy of Gaussian latent variables. To generate a sample from the model, we draw from a Gaussian distribution at the top (L'th) layer. Then the activation h_l of the lower layers is calculated as a non-linear transformation (typically an MLP) of the layer above, perturbed by some Gaussian noise.

Formally, the process of generating a sample from a DLGM is as follows:

$$\xi_l \sim \mathcal{N}(\xi_l|0, I), \quad l = 1, ..., L \tag{1}$$

$$\mathbf{h}_L = G_L \xi_L, \tag{2}$$

$$\mathbf{h}_l = T_l(\mathbf{h}_{l+1}) + \mathbf{G}_l \xi_l, \quad l = 1, ..., L-1 \tag{3}$$

$$\mathbf{v} \sim \pi(\mathbf{v}|T_0(\mathbf{h}_1)), \tag{4}$$

where the ξ_l are independent Gaussian variables, the T_l are MLPs and the G_l are matrices. At the bottom, we generate the sample from any distribution $\pi(\mathbf{v}|.)$, whose parameters are given by a transformation of the bottom latent layer of the model.

Chapter 3

Normalizing Flows

In the attempt to generate more complex families of posterior distributions, which should allow for better estimates of the true posterior, the paper introduces the idea of normalizing flows. A normalizing flow is a sequence of invertible transformations applied to a simple initial distribution (for example a unit gaussian), resulting in a complex distribution, whose density we can efficiently evaluate by inverting the flow and keeping track of the Jacobian of the composition of transformations, using the change of variable theorem.

3.1 Definition

More formally, consider a member of the flow's sequence of transformations to be an invertible, smooth mapping $f : \mathbb{R}^d \to \mathbb{R}^d$. Let $\mathbf{y} = f(\mathbf{z})$ be the outcome of applying the transformation to a random variable $\mathbf{z}$ with distribution $q(\mathbf{z})$. Then, using the change of variable theorem, the resulting density is:

$$q(\mathbf{y}) = q(\mathbf{z}) \left| \det \frac{\partial f^{-1}(\mathbf{y})}{\partial \mathbf{y}} \right| = q(\mathbf{z}) \left| \det \frac{\partial f(\mathbf{z})}{\partial \mathbf{z}} \right|^{-1} \tag{3.1}$$

where the second equality follows from the inverse function theorem and the definition of $\mathbf{y}$.

We can then proceed to build more complicated densities by systematically composing multiple transformations such as the one above, since the composition of invertible transformations remains an invertible transformation. To simplify notation, we define the log-density $\ln q_K$ resulting from applying a sequence of K transformations $f_1, f_2, ..., f_K$ to a random variable $\mathbf{z_0}$ with an initial distribution q_0 as:

$$\mathbf{z_K} = f_K \circ ... \circ f_2 \circ f_1(\mathbf{z_0}) \tag{3.2}$$

$$\ln q_K(\mathbf{z_K}) = \ln q_0(\mathbf{z_0}) - \sum_{k=1}^{K} \ln \left| \det \frac{\partial f_k}{\partial \mathbf{z_{k-1}}} \right| \tag{3.3}$$

where the first equation represents the sequence of random variables generated by the flow. The normalizing flow is then defined as the path of the successive distributions q_k.

3.2 Law of the Unconscious Statistician

A useful property of these transformations is what is known as the law of the unconscious statistician (LOTUS). The LOTUS refers to the fact that we can evaluate expectations with respect to the transformed density q_K without explicitly knowing it, by expressing any such expectation as:

$$\mathbb{E}_{q_K} = \mathbb{E}_{q_0}[h(f_K \circ ... \circ f_2 \circ f_1(\mathbf{z_0}))] \tag{3.4}$$

This is an expectation over the known density q_0, which does not require computation of the logdet-Jacobian terms when $h(\mathbf{z})$ does not depend on q_K.

3.3 Invertible Linear-time Transformations

Notice that a naive choice of these transformations would lead to a $O(d^3)$ complexity to compute the determinant of the Jacobian. Therefore, the classes of flows studied in our report are all based on the idea of having an efficient way to calculate this determinant. We begin by investigating two types of linear flows, namely planar and radial flows.

3.3.1 Planar Flows

Consider the following class of transformations:

$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u}h(\mathbf{w}^\mathsf{T}\mathbf{z} + b) \tag{3.5}$$

where $\lambda = \{\mathbf{w} \in \mathbb{R}^D, \mathbf{u} \in \mathbb{R}^D, b \in \mathbb{R}\}$ are free parameters and $h(\cdot)$ is a smooth, element-wise non-linearity. We can use the matrix determinant lemma to calculate the logdet-Jacobian term in linear time, yielding:

$$\left|\det \frac{\partial f}{\partial \mathbf{z}}\right| = \left|1 + \mathbf{u}^\mathsf{T}\psi(\mathbf{z})\right| \tag{3.6}$$

where $\psi(\mathbf{z}) = h'(\mathbf{w}^\mathsf{T}\mathbf{z} + b)\mathbf{w}$ and h' is the derivative of the function h. Using this, we can now substitute in the formula for $\ln q_K$ to get the following closed form for planar flows:

$$\ln q_K(\mathbf{z_K}) = \ln q_0(\mathbf{z}) - \sum_{k=1}^{K} \ln \left|1 + \mathbf{u_k}^\mathsf{T}\psi_k(\mathbf{z_{k-1}})\right| \tag{3.7}$$

The name planar comes from the fact that the above transformation applies a series of contractions and expansions in the direction perpendicular to the $\mathbf{w}^\mathsf{T}\mathbf{z} + b = 0$ hyperplane to the initial density q_0.

3.3.2 Radial Flows

The other family of invertible linear-time transformations studied is defined by the following equation:

$$f(\mathbf{z}) = \mathbf{z} + \beta h(\alpha, r)(\mathbf{z} - \mathbf{z_0}) \tag{3.8}$$

where $r = |\mathbf{z} - \mathbf{z_0}|$, $h(\alpha, r) = 1/(\alpha + r)$, and the set of parameters is $\lambda = \{\mathbf{z_0} \in \mathbb{R}^D, \alpha \in \mathbb{R}^+, \beta \in \mathbb{R}\}$. The time-complexity of the computation of the determinant of this class is also linear:

$$\left|\det \frac{\partial f}{\partial \mathbf{z}}\right| = [1 + \beta h(\alpha, r)]^{d-1}[1 + \beta h(\alpha, r) + \beta h'(\alpha, r)] \tag{3.9}$$

This transformation results in radial contractions and expansions around the reference point $\mathbf{z_0}$, hence the name radial flows.

Chapter 4

Experiments & Results

We detail the experiments from the paper, our implementation, and the results.

4.1 Experiments

The paper included two experiments. The first was to assess the representative power of normalizing flows, where the models were asked to fit 4 different 2D distributions with unnormallized densities $p(x) \propto exp(-U(x))$ which is defiend in the original paper.

4.1.1 NICE

Using the idea of flows, the authors reviewed other work in the field as flows. One of the main comparisons was the coupling layers in the NICE. A coupling layer f is a neural network layer with easy to compute inverse and a trivial Jacobian. For an input vector $\mathbf{z} \in \mathbb{R}^D$, we have

$$f(\mathbf{z}) = (\mathbf{z}_A, g(\mathbf{z}_B, h(\mathbf{z}_A))) \tag{4.1}$$

$$f^{-1}(\mathbf{z}) = (\mathbf{z}_A, g^{-1}(\mathbf{z}_B, h(\mathbf{z}_A))) \tag{4.2}$$

where (A, B) is a partition of $\{1, 2, \ldots, D\}$, h is an arbitrary function with input size $|A|$, and g is a coupling law, a function that is invertible for the first argument given the second. In the paper, h is a neural network and $g(a, b) = a + b$, so the Jacobian is just the identity matrix. Since the determinant of the Jacobian is 1, the authors of (variation paper) classify it as a volume-preserving flow. The authors of the NICE do point out this issue, and introduce a diagonal scaling layer as the last layer of their models. This was not mentioned in the original paper. Moreover, the authors introduced variants NICE-ortho and NICE-perm instead of the original directly, where random orthogonal and permutation matrices are applied to $\mathbf{z}$ before the partition.

4.2 Implementation

We followed any specifications from the paper as closely as possible, but do note there were parts which are ambiguous. Notably, the term "parameter update" was used for both experiments. While this generally refers to the number of times a parameter in the model is updated, the models for the second part had more than 500000 parameters. Moreover, while the choice of models and optimizers were clear, the actual architecture of the model was not specified well, which made replicating its results non-trivial.

For the first experiment, a "model" took in a vector from $\mathbb{R}^2$ and returned a multivariate normal distribution independent of the input. The transformations were then applied to this distribution. For NICE, their choice of architecture was unclear, and we have chosen h to be a ReLU network of 4 layers with hidden dimensions of 2. Each model was train for 500000 parameter updates, so they were each trained for different number of iterations. The types of transforms used are radial flow, planar flow, NICE-ortho and NICE-perm. We also introduce NICE-ortho-diag and NICE-perm-diag, which are NICE-ortho and NICE-perm but with diagonal scaling applied as an additional transform.

For the second experiment, we were only able to run and obtain we created encoders and decoders for distributions, allowing one or more transformations in between. The encoder

For implementing radial flow, we assume that $\beta \geq -\alpha$ and were able to obtain an explicit inverse. We parameterized $\hat{\alpha} = \exp(\alpha)$ to enforce $\alpha > 0$ and $\hat{\beta} = -\alpha + \exp(\beta)$ as per the appendix of the original paper.

4.3 Results

After 50000 parameter updates, the variational bounds are show in Figure 4.1. While the range of values is close to those from the paper, the shapes are visibly inconsistent. We suspect this is due to the "parameter updates" being iterations or epochs. We do note that among NICE-based models, those with diagonal scaling do perform better than their counterparts.

Due to time and computationale resource constraints, we were unable to finish training for the second experiment.

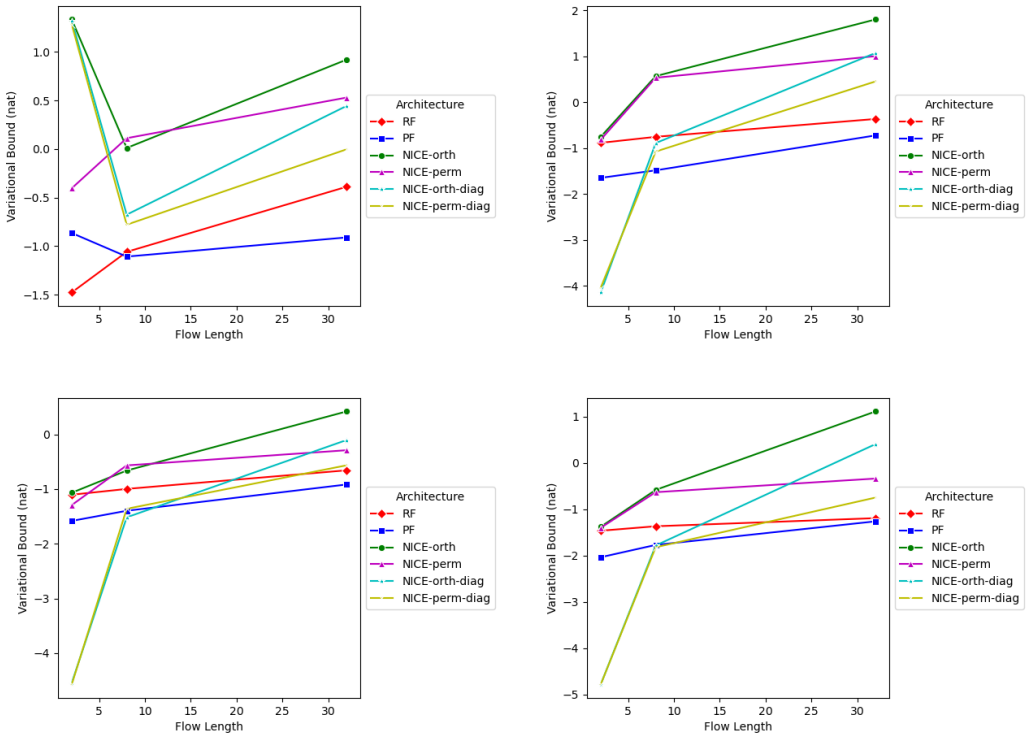


Figure 4.1: Variational bounds of the models with respect to the 4 distributions. Read from top to bottom, left to right.

Chapter 5

Extension - Invertible Convolution Flows

As an extension to the paper, we implement another type of flow, from "Invertible Convolutional Flows" [**?**]. In particular, we have chosen to implement a Circular Convolutional Flow. As far as we can tell there are no available implementations of this paper available at the time of writing.

5.1 Circular Convolution Flows

In standard flow-based models, a complex probability density is built up by transforming a simple base density, such as a standard normal, by a chain of smooth bijections. However, using a complex transformation to define a normalised density requires computing the Jacobian determinant of the transformation, which is impractical for general transformations by neural nets. There has been much research on designing families of transformations which are guaranteed to have nice Jacobian forms, such as [**?**] which considers transformations whose Jacobian is a perturbation of a diagonal matrix. The authors of Invertible Convolutional Flows have used transformations whose determinants are *circulant matrices*, which will allow computation of the log determinant, as well as the convolution and its inverse, to be calculated in time $\mathcal{O}(N \log N)$ by making use of Discrete Fourier Transforms (DTFs).

5.1.1 Definition

A *circulant matrix* is a matrix $\mathbf{C}$ such that the rows and columns are cyclic permutations of the first row/column, ie

$$\mathbf{C}_{l,m} = \mathbf{C}_{1,(l-m) \bmod N} \tag{5.1}$$

Define the circular convolution as

$$\mathbf{y} := \mathbf{w} \circledast \mathbf{x} \text{ where } \mathbf{y}(i) := \sum_{n=0}^{N-1} \mathbf{x}(n)\mathbf{w}(i-n)_{mod\, N} \tag{5.2}$$

Then the circular convolution can be expressed as a matrix-vector multiplication $\mathbf{y} = \mathbf{C}_\mathbf{w}\mathbf{x}$ where $\mathbf{C}_\mathbf{w}$ is a circulant matrix with $\mathbf{w}$ as its first row. This means that the Jacobian of the map is just $\mathbf{C}_\mathbf{w}$, and we can calculate the log determinant jacobian as follows [**?**]:

$$\log|\det \mathbf{J}_\mathbf{y}| = \sum_{n=0}^{N-1} \log|\mathbf{w}_\mathcal{F}(n)| \tag{5.3}$$

where $\mathbf{w}_\mathcal{F}$ is the DFT of $\mathbf{w}$.

Moreover, by continuing to work in the Fourier space, we can also compute the convolution and the inverse in $\mathcal{O}(N \log N)$ by:

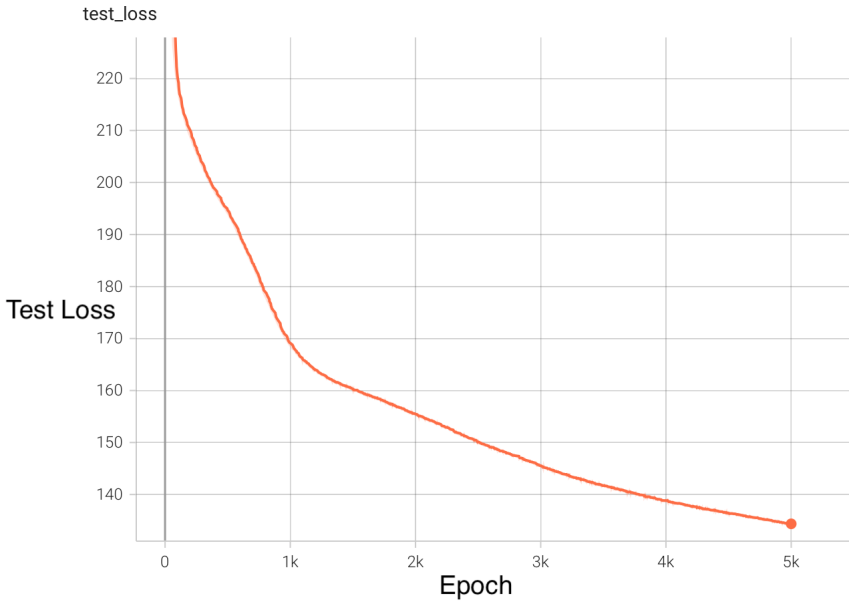
$$\mathbf{y}_\mathcal{F}(n) = \mathbf{w}_\mathcal{F}(n)\mathbf{x}_\mathcal{F}(n) \tag{5.4}$$

$$\mathbf{x}_\mathcal{F}(n) = \mathbf{w}_\mathcal{F}^{-\mathbf{1}}(n)\mathbf{y}_\mathcal{F}(n) \tag{5.5}$$

5.2 Our Experiment

The circular convolutional flows was tested on the MNIST dataset using a flow length of 10 only. This was mainly due to resource limitations, training the model with larger flow lengths proved infeasible in the time we had.

5.3 Results



After 5000 epochs of training, the train and test loss were both still decreasing, so additional training time is likely to have produced an even better model. Also due to this limitation, it is difficult to tell whether the circular convolutional flows would have produced a better model than the normalising flows in our main experiments.

Chapter 6

Conclusion

Overall, the paper [**?**] provided a very interesting introduction to the idea of Normalizing Flows, and as a team we appreciated diving deep into this field we were previously not familiar with. Although in multiple parts of the paper some of the details on how they ran the experiments and made related choices were missing or unclear, we ended up replicating the main experiments and achieving similar results.

The main issue we faced was lack of computational resources to train our model, since all experiments required running the training function for hours, and in some cases this ended up proving infeasible (for example, this was the reason we did not have time to extend our experiments to other datasets). Thankfully, some of the referenced papers included useful details on the implementation as well as the training choices, which saved us an important amount of time we would have had to spend on tuning various hyperparameters.

6.1 Further Extensions

Time constraints, in combination with the limited computational resources we possessed, forced us to focus this project on a single extension, that of circular convolution flows. However, in our reading prior to completing the project, we came across a variety of other directions in which we could extend the paper's results. What follows is a summary of some of these ideas, grouped by the part of the paper which they would have to modify.

6.1.1 Different Types of Flows

In our project we considered the two different types of linear flows in section 2.3, as well as NICE in section 2.4. However, there has been lots of ongoing research on flows in recent years, so a possible extension would be to experiment with these and compare the results to the ones from the paper.

Infinitesimal (Continuous) Flows

To begin with, one can consider what happens if we allow for the length of the flows to tend to infinity. Such flows are called infinitesimal flows. Even though they are mentioned in [**?**], there is no related experiment discussed, and they serve mostly as a theoretical guarantee for the correctness of the normalizing flows framework. However, these can be modeled as partial differential equations and studied in more depth (see [**?**]).

Autoregressive Flows

Another emerging category of flows is that of autoregressive flows [**?**], where each dimension of a vector variable is being conditioned on the previous dimensions. Some instances of autoregressive flows in literature are Masked Autoregressive Flows (MAF)[**?**] and Inverse Autoregressive Flows (IAF) [**?**].

6.1.2 Additional Datasets

The paper provided comparison and extensive experimentation only on the MNIST dataset, so an obvious extension would be to use different datasets. CIFAR-10 is already discussed in the paper, so given more computational resources we would have liked to build models on that and compared them. Having completed this, it would also make sense to proceed with more complex image datasets of higher data dimensionality, such as ImageNet32 and ImageNet64.

6.1.3 Alternatives to the loss function

Finally, we could have trained our models with different loss functions. Most work has been done in loss functions optimized by minimization of the Kullback-Leibler divergence, similarly what we did in this report. However, there is work on different approaches, see [**?**] and [**?**].