Masked Autoregressive Flows with a Marginalising Flow Framework

Bachelor's Thesis in Artificial Intelligence by

Isaac Lee[1]
s1007848

Supervised by:
Luca Ambrogioni[2]

[1]Department of Artificial Intelligence, Radboud University
[2]Donders Institute for Brain Cognition and Behaviour, Radboud University

July 31, 2020

Abstract— **Invertible deep networks, or normalising flows, are the focus of an influx of new research in the field of machine learning. In this paper marginalising flows, which can be applied on top any normalising flow, will be implemented over the existing Masked Autoregressive Flow framework. The performance will be compared between different masked autoregressive models with and without marginalising flows. Additionally image generation will be analysed after training on CIFAR-10.**

I. Introduction

In the world of statistics and by extension, machine learning, it is vitally important to be able to mathematically describe the process through which data is produced. This can range from some 'distribution' as simple as a Gaussian, which has a known probability density calculation, to images of faces, which are too complex to describe with a formula. Being able to describe the density and efficiently sample from complex distributions is a goal of modern machine learning. Normalising flows, a type of generative model, have recently been used as a new method to solve this problem.

Normalising flows make up a subcategory of neural networks which possess specific properties. It is these properties that allow them to function as generative models. Typically a normalising flow learns to map data from a complex distribution to a more simple one. Therefore the flow acts as a singular transformation or series of individual transformations that are able to transform a point in a complex distribution to a point in a simple one. Normalising flows can function as generative models because the transformations they use are restricted such that they are not only invertible but also both the transformation and its inverse are differentiable.

This paper mainly explores the possible performance improvements of adding a marginalising flow to masked autoregressive flows [10]. Comparisons will be made between masked autoregressive flows with and without a marginalising component. Additionally, Planar flows will be briefly experimented with and analysed.

II. Related work

After the introduction of Planar and Radial flows by Rezende and Mohamed [13] for variational inference along with the NICE framework for density estimation by Dinh et al. [1], normalising flows have spiked in popularity. Many more advanced and complex frameworks have been created such as RealNVP [2], Glow [6] and Flow++ [4] for the purpose of accurate density estimation in high-dimensional images. Normalising flows have also been adapted from autoencoders, as is the case in Masked Autoregressive flows [10], upon recognition that autoencoders share the properties that normalising flows require.

More recently, there has been literature written to serve as an introduction to the concepts of normalising flows and the state of the field. [11][7].

III. Background

A. Normalising flows

A normalising flow is typically made up of multiple layers each performing a single transformation. The combination of the transformations of those layers is referred to as T. Equation 1 shows how T is made up of the transformations of each layer.

$$T = T_i \circ T_{i-1} \circ ... \circ T_2 \circ T_1 \tag{1}$$

For a normalising flow to be fully functional it is a requirement that T, and therefore all transformations that make up T, is invertible. In addition, both T and its inverse T^{-1} are required to have a tractable differentiation. This requirement is necessary to be be able to compute the determinant of the Jacobian. The Jacobian matrix is made up of the first order partial derivatives of the transformation function T. The absolute value of the determinant of this matrix represents the amount of expansion and shrinkage in the transformation. Satisfying these conditions allows us to describe the density of complex distributions via the combination of another distribution and a flow.

$$p_x(x) = p_u(T^{-1}x)|detJ_{T^{-1}}(x)| \tag{2}$$

Formula 2 describes the probability density of x, our complex density, in terms of u, our simple one, and the transformation T^{-1}. [11]
A normalising flow can be trained by sampling from the complex distribution, passing the sample through transformation T, and then computing a loss against a target distribution. The target distribution, $p_u(u)$, is typically taken to be a simple Gaussian.

Normalising flows have seen many uses, including density estimation, variational inference and image generation. In this paper we consider the performance of normalising flows in the context of density estimation and in the generation of images, specifically, in regard to implementing a marginalising flow framework to try and improve performance.

B. Marginalising flows

Marginalising flows are an adaptation of the generic normalising flow. It consists of an existing normalising flow model which has inputs that are augmented by an additional amount of dimensions. The additional dimensions are given as input to the original normalising flow in an attempt to capture any information that is lost when the original input data is passed through the normalising flow's transformations.

A simple example is to consider the squared function:

$$f(x) = x^2 \tag{3}$$

In this simplistic transformation the sign of the data is lost with no method of recovery. If an additional dimension is used the transformation can learn to capture the information to increase the likelihood of recovering the original input.

$$f([x]) = [x^2, sign(x)] \tag{4}$$

Equation 4 demonstrates how extending the dimensionality allow for more information to survive the transformation.

Marginalising flows are a novel framework that can be built on top of any existing normalising flow. This offers flexibility and allows it to be easily tested without having to

build your normalising flow again from the ground up. The size of the additional dimensions added is called epsilon and can be of any size.

C. Planar flows

A planar flow has transformation f given in the following equation.

$$f(z) = z + u \cdot h(w^T \cdot z + b) \tag{5}$$

Planar flows act to stretch and contract space in the direction perpendicular to the hyperplane $w^T \cdot z + b = 0$. In formula 5, u, w and b are learned parameters while h is a non linear function that is smooth. The hyperbolic tangent function is a common choice. The Jacobian determinant is calculated by means of the matrix determinant lemma. [13]

$$|detJ_f| = |1 + u^T \cdot (h'(w^T \cdot z + b) \cdot w)| \tag{6}$$

Where h' is the derivative of h, in this implementation, tanh.

$$h'(z) = 1 - tanh(z)^2 \tag{7}$$

In the field of normalising flows, planar flows can be considered basic. They have limited expressiveness when they are used alone, although their ability to approximate a more complex distribution can be increased by incrementing the number of planar flow layers so that the flow is made up of many transformation layers with each having their own parameters. Additionally, to note, planar flows have no analytical inverse although an inverse function can be guaranteed to exist by restricting the free parameters.

D. Masked Autoregressive flows

Masked Autoregressive Flows (MAFs) are made from alternating layers of two different transformations. The first transformation is batch normalisation and the second is a masked autoencoder.

1) *Batch normalisation:* This first layer normalises the data of each mini-batch to allow each layer to deal with similarly distributed data [5]. By calculating the mean and variance of each batch of input data you can normalise the input to make the mean and unit variance both approximately zero. Batch normalisation, as is required in a normalising flow, has both an algebraic easy inverse and a computable Jacobian.
2) *Masked autoencoder:* The second transformation is a masked autoencoder for density estimation, coined MADE [3]. MADE is an adaptation of autoencoders that can be efficiently used for density estimation. An autoregressive property is enforced to allow an autoencoder to be used to calculate probabilities on data.

$$\hat{x}_d = p(x_d | x_{<d}) \tag{8}$$

Equation 8 shows the autoregressive property. The output $\hat{x}_d$ must be the probability of x_d using only the previous values $x_{<d}$. This property is enforced by the use of masks, such a binary mask is multiplied element-wise with the weight matrix of the fully connected autoencoder and initialised to enforce the autoregressive property.

The MADE model also shares properties with normalising flows, such as the ability to be invertible and having a tractable Jacobian and thus can be easily used as part of a flow. A tractable Jacobian is possible because the autoregressive property causes the Jacobian matrix to be triangular and the determinant of a triangular matrix is the product of the diagonals.

Masked autoregressive flows combine the two transformations in blocks, made up of both, that can be stacked much like increasing the depth of other normalising flows. The stacking of these blocks has been shown to increase the flexibility of the autoencoder, e.g. giving it the ability to handle multimodal distributions [10].

While MAFs have an efficient forward pass, as it can be calculated with parallelisation on GPUs, the inverse is conversely sequential and does not benefit from offloading the calculations to a GPU. This means that training is fast but using the inverse to generate samples in the complex distribution is slow.

IV. Experiments

In the following section the experiments performed will be detailed. After which the results will be discussed.

All normalising flow and training implementations were done in Python with PyTorch [12].

A. Datasets

Three datasets have been used and trained on: MNIST [9], CIFAR-10 [8] and samples generated from a half moons distribution. MNIST is made up of handwritten digits. Each image has a single channel and consists of 28x28 greyscale pixels with a centered digit. This totals 784 input values. CIFAR-10 is made up of colour images of 10 different classes. Each image has 3 channels, RGB, and consists of 32x32 pixels with the subject of the image centered. The complexity of this dataset is far greater than MNIST and also has 3072 input values.

Both the MNIST and CIFAR datasets are the preprocessed versions from the MAF paper [10]. They are used to test the performance of generative modelling and density estimation.

A half moons dataset has also been used for testing planar flows. This is created by sampling 25000 values from sklearn's `make_moons` function with `noise=0.05`. Noise is added to add some randomness to the data.

All datasets have been reshaped to $[b, N]$ as input to the normalising flow models, where b is the mini-batch size and N is the number of individual data points. For example a mini-batch of 100 MNIST images is reshaped from $[100, 28, 28]$ to $[100, 784]$. This is done to simplify adding epsilon dimensions in the case that a marginalising flow is used.

All experimentation was done with a Gaussian as the base distribution and using the Adam optimiser with a learning rate of $5e-4$.

B. Planar flows

Planar flows were the first flow considered. They are simple, fast and easy to work with. After early testing, 8 layers of the planar flow transformation was decided on because it added enough flexibility while still allowing quick training.

Initial experiments were performed on the half moons dataset both without and with the marginalising flow framework. MNIST was used next to further evaluate the ability to learn complex densities. The performance of planar flows on MNIST was poor on all accounts. This led to further research and discovery of masked autoregressive flows which became the focus of my experimentation.

C. Masked autoregressive flows

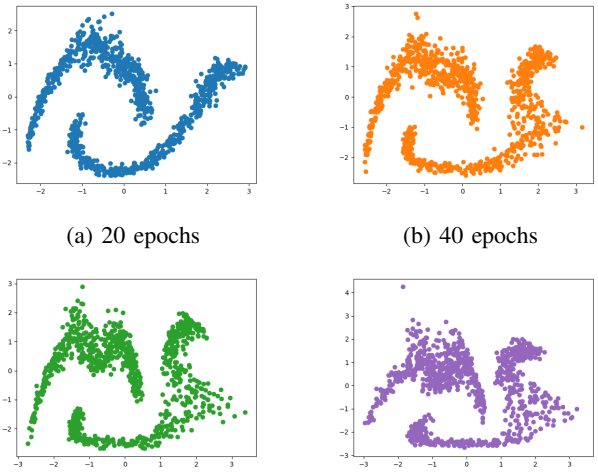
Masked autoregressive flows have been chosen for their performance on density estimation tasks and their simplistic architecture. MAFs offer comparative performance with Real NVP, that has many more layers and architecture complexity, and outperform it on density estimation tasks. [10] Masked autoregressive flows were first tested on MNIST to be able to evaluate comparative performance against planar flows. From there CIFAR-10 has been the focus of experimentation. Marginalising flows have also been briefly tested on MNIST with an epsilon of 256. The MAF configuration for MNIST was with two hidden layers of size 1024 for each MADE.

Regarding CIFAR-10 two different MAF configurations are used. The first uses three hidden layers for each MADE of size 1024 and the random mode for configuring masks. The second uses a single hidden layer for each MADE of size 3072 and the sequential mode. Both are made up of 5 blocks of alternating MADE and batch normalisation layers and use the ReLU activation function. A model with three hidden layers of size 1024 was chosen to contrast the model with a single hidden layer of size 3072. In addition, the marginalising model for the first instance does not have its hidden layer size augmented whereas in the case of the second model the single hidden layer size is increased to 3584 to match the input size. Each configuration is tested with and without a corresponding marginalising flow framework. An epsilon value of 512 has been tested and was chosen to be double the value from MNIST because of the increased number of inputs.

For all datasets used with masked autoregressive flows the gradients were clipped with a value of $5e-6$. This was necessary to avoid NaN's and aid in preventing artifacts from appearing in the data generation.

V. Discussion

This section analyses the results of my experimentation and shows results from the generative normalising flows. The negative log likelihood results of the experimentation with masked autoregressive flows and CIFAR-10 can be found in table I.



(a) 20 epochs (b) 40 epochs

(c) 60 epochs (d) 100 epochs

Fig. 1: 1000 half moon data samples passed through a planar flow after a varying amount of epochs.

A. Planar results

Figures 1 and 2 demonstrate the difference a marginalising flow can make on a simple distribution such as the half moon dataset with a planar flow. There is no generated data as planar flows have no analytical inverse that can be used to pass Gaussian data through the inverse transformation.

The performance of the flow can instead be analysed by how it is able to transform the half moon data to approximate the base Gaussian distribution. It is evident in the figures that the marginalising flow has a significant impact on the ability of this flow to do this. Fig. 1 can be seen to struggle in distorting the original data in contrast to fig. 2. It is suspected that the second model, with a marginalising flow, has captured the random noise of the appended epsilon dimensions to aid in transforming the data.

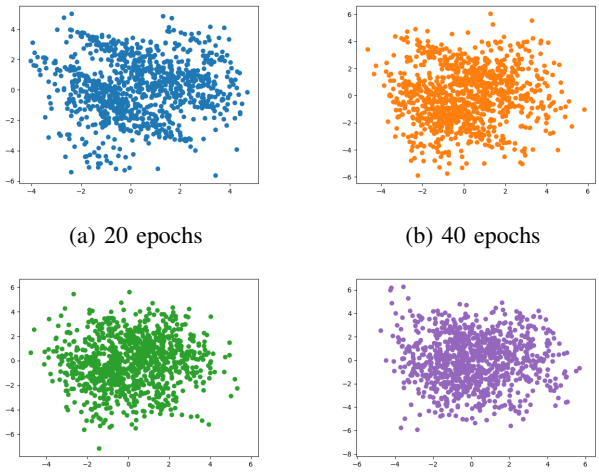
The model from the fig. 1 achieved a negative log-likelihood of -0.88 compared to the fig. 2's model with -7.27.

Both models were only trained for 100 epochs each although it is possible the performance would converge if trained for longer. Additionally, it should be said that it is impossible to know if the second model would achieve better performance in generation compared to the first model despite it being able to learn to approximate a Gaussian more accurately after 100 epochs.

The added complexity of MNIST over the half moons data is evident by the performance of the planar flow in fig. 3. The flow manages to create random noise around the edges of each image but not the center. The flow, for some reason, seems to leave the digit untouched perhaps due to its innate transformative properties.

B. MAF results

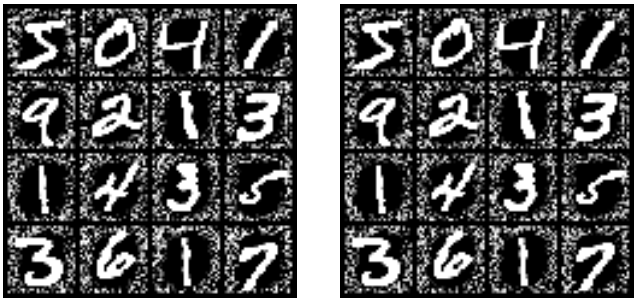
MNIST was initially experimented on to clarify the performance improvements over planar flows. This is evidenced in



(a) 20 epochs (b) 40 epochs

(c) 60 epochs (d) 100 epochs

Fig. 2: 1000 half moon data samples passed through a planar flow with a marginalising flow framework after a varying amount of epochs.



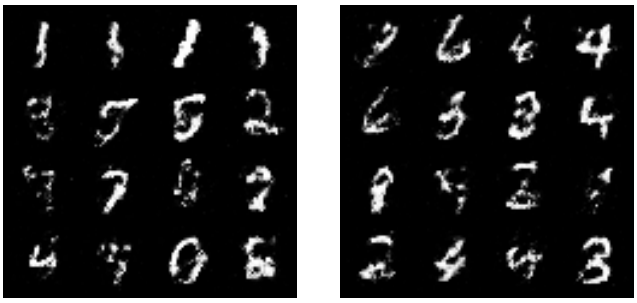
(a) Without (b) With

Fig. 3: 16 MNIST samples after being passed through a planar flow after 100 epochs of training with and without a marginalising framework.

fig. 4 where MAF performance can be compared with and without the marginalising framework. The experimentation with MNIST was brief and CIFAR-10 is the focus of this research.

The first MAF model's MADE layers each have three hidden layers of size 1024. The input size is 3072 and 3584 in the case of the marginalising flow being added. Results can be seen in fig. 5 which should be compared to the image generation results in fig. 6. The first model reached a negative log likelihood of -1272.4 while the second model with the added marginalising framework reached -624.1. As can be seen, both models show poor results in generating consistently plausible images that compare to the MAF paper. This contrasts with the more accurate images generated by the models with a large single layer. Both of these models also display tendencies to generate images that have very similar colour and structure as can be seen clearly in fig. 6 across all 4 instances.

It should be pointed out that all MAF models have artifacts



(a) Without (b) With

Fig. 4: Generated MNIST digits after 100 epochs of training with MAF with two hidden layers of size 1024, without and with a marginalising framework.

that crop up in the generated images that are defined by a complete take over of black and red pixels, and in some cases green/yellow. These artifacts become more common the longer the model has been trained and have already been significantly reduced by gradient clipping but not eliminated.

Figures 7 and 8 show the results of the MAF model trained with a large single hidden layer in each MADE, the first being without the marginalising flow and the second with. Both results have significant artifacts but the images that do not show comparable results to the original MAF paper. The first of these two models achieved a negative log likelihood of -3195.9 compared to the second which reached -3229.6. There is no significant difference between the results of these two models. It seems that the marginalising flow does not aid the normalising flow on such a complex distribution contrasting the impact it had on the half moons dataset with planar flows. In comparison to the previous set of MAF models, using the models with large single hidden layers offers a large performance improvement and is seen in both the negative log likelihood value and the generated images.

TABLE I: Negative log likelihood (lower is better) for density estimation on the masked autoregressive flow experiments.

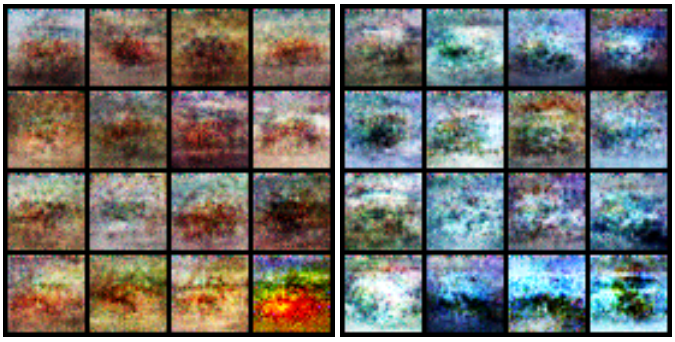
	CIFAR10	
	without MF	with MF
MAF 1024x3	-1272.4	-624.1
MAF 3072	-3195.9	-3229.6

C. Future work

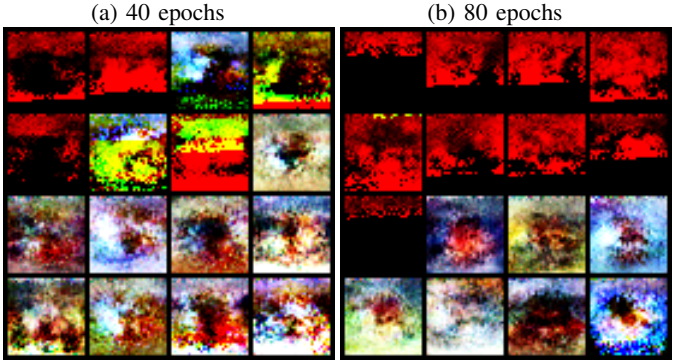
Much of the research and experimentation performed above can be easily extended and expanded.

Marginalising flows' impact on the simple planar flow can be examined to determine how the added dimensions play a part in transforming the distribution to a Gaussian one. In addition, the poor performance planar flows show in transforming the digits of MNIST could offer insights into how the flow can perform as the complexity of data increases.

Masked autoregressive flows have a lot of potential, not only for density estimation, but also for generative modelling.

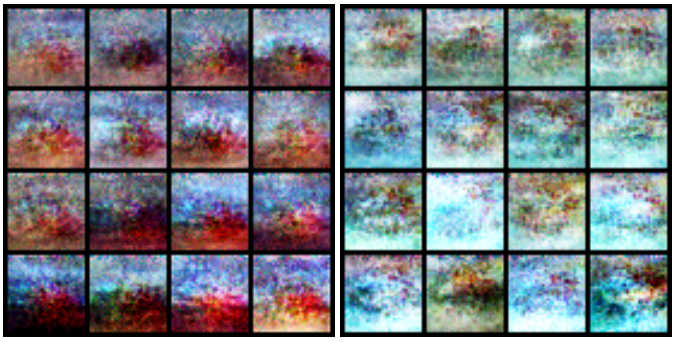


(a) 40 epochs (b) 80 epochs

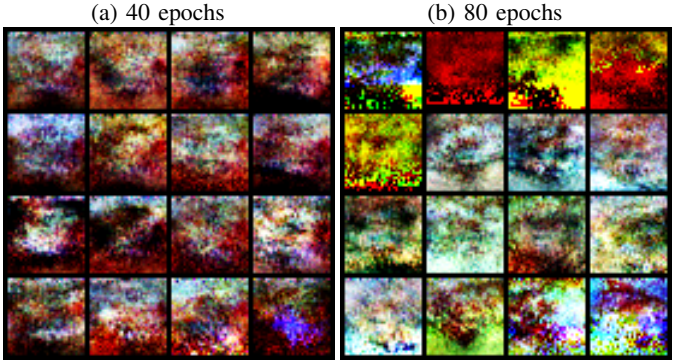


(c) 200 epochs (d) 400 epochs

Fig. 5: Results of image generation in the MAF model with three hidden layers of size 1024 and no marginalising flow.

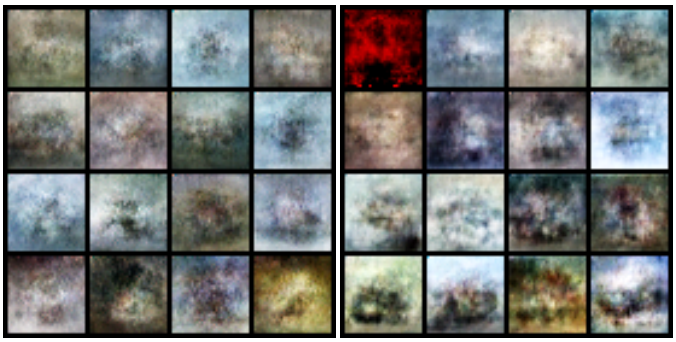


(a) 40 epochs (b) 80 epochs

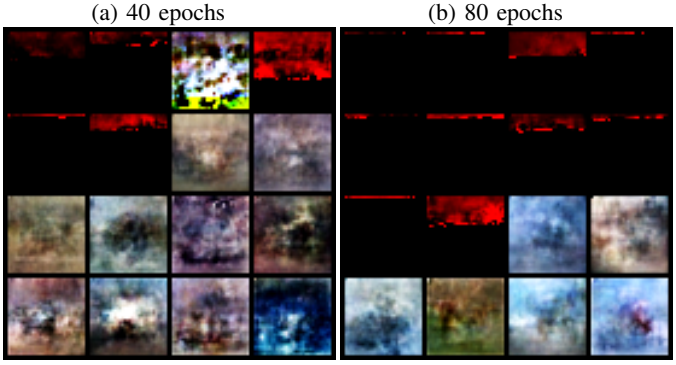


(c) 200 epochs (d) 400 epochs

Fig. 6: Results of image generation in the MAF model with three hidden layers of size 1024 and the marginalising flow framework.



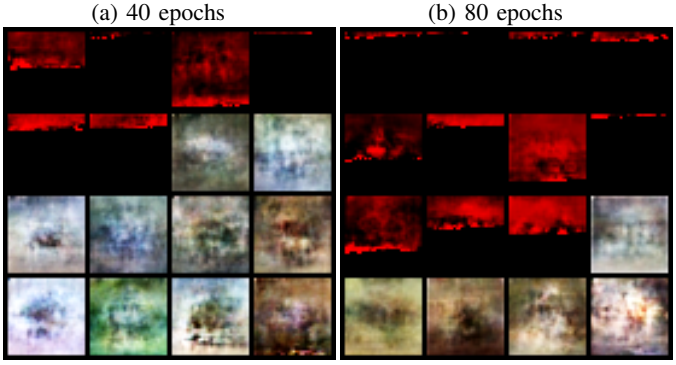
(a) 40 epochs (b) 80 epochs



(c) 200 epochs (d) 400 epochs

Fig. 7: Results of image generation in the MAF model with a single hidden layer of size 3072 and no marginalising flow framework.

(a) 40 epochs (b) 80 epochs



(c) 200 epochs (d) 400 epochs

Fig. 8: Results of image generation in the MAF model with a single hidden layer of size 3584 and the marginalising flow framework.

Additional flow layers such as 1x1 convolutions could be added alongside MADE and batch normalisation layers to help capture the structure of images which could lead to improved image generation.

Regarding marginalising flows, there is a wealth of research that could be performed to incorporate the framework on top of already existing flows or to build new flows around the concept of capturing lost data in auxiliary dimensions. The performance shown regarding planar flows with a simple distribution like half moons gives promise to the impact they could possibly have. With further GPU parallelisation and increased memory larger epsilon dimensions could be added for experiments on complex datasets, e.g. 3072 pixels and 3072 epsilon dimensions in the instance of a 32x32 RGB image.

VI. Conclusion

In this paper the impact of marginalising flows are investigated regarding simple flows and simple distributions as well as more complex flows and larger, complex data such as natural images. While they seem to have negligible effect on such complex instances there are promising results with regards to augmenting a simple normalising flow such as a planar flow in learning to transform data to a Gaussian. Marginalising flows are extremely flexible and can be built on top any normalising flow with minimal adjustment and there is great potential for further research into the impact they can have.

References

[1] Laurent Dinh, David Krueger, and Yoshua Bengio. *NICE: Non-linear Independent Components Estimation*. 2014. arXiv: 1410.8516 [cs.LG].

[2] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. *Density estimation using Real NVP*. 2016. arXiv: 1605.08803 [cs.LG].

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

[4] Jonathan Ho et al. *Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design*. 2019. arXiv: 1902.00275 [cs.LG].

[5] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 448–456. URL: http://proceedings.mlr.press/v37/ioffe15.html.

[6] Diederik P. Kingma and Prafulla Dhariwal. *Glow: Generative Flow with Invertible 1x1 Convolutions*. 2018. arXiv: 1807.03039 [stat.ML].

[7] Ivan Kobyzev, Simon Prince, and Marcus Brubaker. "Normalizing flows: An introduction and review of current methods". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

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

[9] Yann LeCun and Corinna Cortes. "MNIST handwritten digit database". In: (2010). URL: http://yann.lecun.com/exdb/mnist/.

[10] George Papamakarios, Theo Pavlakou, and Iain Murray. *Masked Autoregressive Flow for Density Estimation*. 2017. arXiv: 1705.07057 [stat.ML].

[11] George Papamakarios et al. *Normalizing Flows for Probabilistic Modeling and Inference*. 2019. arXiv: 1912.02762 [stat.ML].

[12] Adam Paszke et al. "Automatic differentiation in PyTorch". In: (2017).

[13] Danilo Jimenez Rezende and Shakir Mohamed. *Variational Inference with Normalizing Flows*. 2015. arXiv: 1505.05770 [stat.ML].