Normalizing Flows: Reproduction of results from Real-NVP normalizing flows

Tushar Kataria
Scientific Computing Institute ,University of Utah
Email: tushar.kataria@utah.edu*

***Abstract*—In this study, review of latest results on normalizing flows and reconstruction of results from Real-NVP[6] is planned and partially executed. Extensive study on variations of different parameters like number of concatenated normalizing flows, activation functions, number of samples for target distribution, noise in input samples, masks variability and complexity of normalizing flow models was done and observations made. Normalizing flows for Image data was not completed so it's set as future work. The github directory for the code is https://github.com/tushaarkataria/NormalizingFlowExperiments**

***Keywords*—Normalizing Flows, Probabilistic Modelling and Inference, Tutorial**

I. Introduction

Finding the exact probability distribution of the input data is an important problem in probabilistic modelling. Many of the famous techniques(Bayesian modelling, MAP, Boltzman machines etc) for finding exact distribution are tractable only for a few 100 dimensions or are really hard to solve for complex models. But with increasingly complex applications and huge data size there is a need for models which can work better for high dimensional data(images).

Deep generative modelling via GAN's [1] seems to give good answers for finding the distribution of training data, but it comes with it's own problems. First of all, the mapping between data to distribution and vice-versa is not invertible/bijective, so we don't have an explanation of why a particular sampled point in a distribution would relate to the image/data thus generated. The exact evaluation of sampled points is not possible. Other issues include mode collapse, posterior collapse due to training instability and also huge amount of data required for the networks to converge to a good solution.

Normalizing flows(NFs) [2], [3] [6] are another family of generative modelling which addresses some of the shortcomings in GAN modelling. NF's can model both sampling and density evaluation in an exact and efficient manner. Normalizing flow are based on the properties of transformation of a probability density which makes the modelling easier to understand and human interpret able. Because of the these advantages, NF's have already been used in many application such as image generation [4], image super resolution [5] [6] and many more.

But with all the advantages of normalizing flows, there are still issues. Although both sampling and density estimation is theoretically possible, it's not always track able for all types of normalizing flows. The other major disadvantage is that because of the nature of bijective transformations and invertibility, the dimentionality of data to which normalizing flows can be applied it still limited. GAN's can generate images of size 512 and 1024 but NF's images generation has not reached that dimentionality yet.

In this paper, section II will cover the basics of the Normalizing flows. Section III will cover the experiments done till now and Section IV will cover the Experiments planned for rest of the semester.

II. Section II : Normalizing Flows

Let us assume that we have a random variable $\mathbf{x}$ whose probability density is known and trackable, denoted by $\mathbf{p_x}(\mathbf{x})$. Let g be an invertible function which transforms the random variable $\mathbf{x}$ to $\mathbf{y}$.

$$\mathbf{y} = \mathbf{g}(\mathbf{x})$$

One can easily compute the probability density of $\mathbf{y}$ using the following equation

$$\mathbf{p_y}(y) = \mathbf{p_x}(f(y))|detD(f(\mathbf{y}))| \tag{1}$$

where $\mathbf{f}$ is the inverse of $\mathbf{g}$ and $D\mathbf{f}(y) = \frac{\partial \mathbf{f}}{\partial y}$ is the jacobian of $\mathbf{f}$. We can do the above for any set of $\mathbf{g}$ and $\mathbf{f}$ invertible functions. This is the main basis for normalizing flows modelling. One such function is called a flow, and because most of the time our base density of gaussian normal, we say that we are normalizing the data distribution, hence the term normalizing flows. One normalizing flow maybe enough for less complex data distribution but for high dimensional data, we generally need complex models.

We use another property of invertible functions, composition propertry. Which states that inverse of composition of the 2 invertible function is composition of inverse. Let g_1, g_2 be invertible function with f_1 and f_2 as inverses.

$$(g_1 \circ g_2)^{-1} = f_1 \circ \mathbf{f}_2$$

The determinant of jacobain is multiplication of all the jocobians.

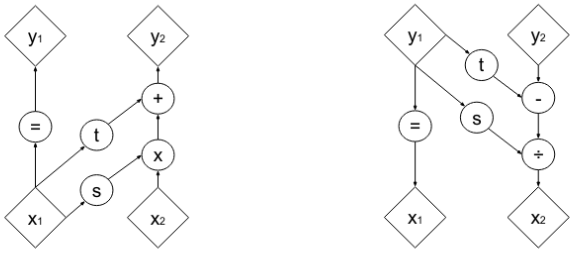
$$det\mathbf{Df}(f_1 \circ f_2) = det\mathbf{Df}_1(\mathbf{x}_1) * det\mathbf{Df}_2(\mathbf{x}_2)$$

We can extend this definition to any number of compositions. Making the normalizing flow model arbitrary complex which helps with modelling of a variety of complex datasets.

The one major issue with above equation **1** is inverse of matrix takes $O(n^3)$ operation which makes the above calculation really time consuming for higher dimensions. But most of the normalizing flows model side step that huge computation cost by designing the models where matrix inverse is not computed. Most of the models do this by creating the model in such a way that the jocobian matrix is a triangular matrix, making the determinant and inverse computating a set of multiplications and division.

A. Real NVP

This study will focus on recreating the results from Real-NVP [6]. Real NVP stands for real -valued non volume reserving transformation. These are a set of invertible, learnable transformation proposed in the paper. As explained in equation **1** above the transformation of the probability density is modelled by multiplication of 2 terms, function and it's Jacobian matrix. This paper presents a way to model these in a very simple and intuitive manner.



(a) Forward propagation (b) Inverse propagation

Figure 1: Computational Graphs for forward and inverse computation of the Real NVP model

1) Modeling of Real NVP :- Coupling Layers: Let **x**be an N dimensional vector for which we are writing a real-NVP normalizing flow. Let us split x in 2 set's of inputs $x_1 and x_2$ with x_1 containg first $d < N$ dimensions and x_2 containing the remaining ones. The forward propagation which is defined above is

$$y_1 = x_1$$
$$y_2 = x_2 \circ exp(s(x1)) + t(x_1)$$

Where s and t are scaling and translation functions. Due to such a definition of the transformation, the jacobian is

$$\frac{\partial y}{\partial x^T} = \begin{bmatrix} I_d & 0 \\ \frac{\partial y_2}{x_1} & diag(exp[s(x_1)]) \end{bmatrix}$$

As jacobian is a upper triangular matrix so we can easily computer the determinant is $\prod exp(s(x_1))$. If we take log on both side, for optimization of posterior probability we get jacobian as $\sum s(x_1)$.

III. Section III : Experiments Done and Analysis

Code was written separately for both 2-dimensional distribution matching and image data. My code for image data is not working correctly and still in debugging stage. Debugging of correct implementation was done with help from github directories [7] [8]

A. 2D Distribution Matching

A variety of experiments were done on 4 distributions with 2D data. distributions were two-moons, circles and 2 different spiral distributions. Results for two-moons are shown in 2, circles is shown in 3 and spiral densities are show in figures 4 5. The parameters of variations were, number of concatenated flows(ranging from 3 to 6), complexity of each flow from 2 intermediate layers to 6 intermediate layer, number of input samples target distribution provided, noise in input samples and activation function either relu, leaky relu or tanh. Analysis of the results is as flows

- **Number of Concatenated flows** Simple distributions like two moons and spiral one require less normalizing flow. But one flow is not enough to model all the transformation required for matching distributions. Even one flow with highly complex model was not able to do a good job for two-moons and failed miserably for spiral distribution. Number of concatenated normalizing flows is a hyper parameter and needs to be tuned for each distribution separately. But for 2-D distributions below, most of the time, 6 concatenated normalizing flows was optimal scenario.
- **Complexity of each flow** Increase the complexity of flows doesn't have a huge change in the performance of the model, concatenation of flows is more important.

- **Number of input samples target distribution** Increasing the number of samples for target distribution increased the time complexity but the model converged quickly and sometimes even with less number of concatenated flows(4 number of concatenated flows for two moons.)
- **Noise in input samples** More noise in the input samples, the more variations in the distribution, harder for the model to get a correct underlying distribution. But most of the time the final target distribution is matched to the noisy distribution. So if the noise if too high in input samples normalizing flow with not be able to get the underlying distribution no matter the model complexity.
- **Activation function** ReLU and Leaky ReLU performed in a similar fashion, but models with tanh activations didn't converge that well.
- **Mask variability** If the mask's of concatenated normalizing flows are covering only one dimension, then the model doesn't converge. Like for example if x_1 in Figure 1 is always taken for bypassing the modeling. Masks in the concatened normalizing flows need to change for good convergence. So, if the sequence of masks of the concatenated normalizing flows covers both the dimensions, the model generally converges.

The above model was implemented and tested for different base and target densities. Example outputs are given below in figures 2 3.

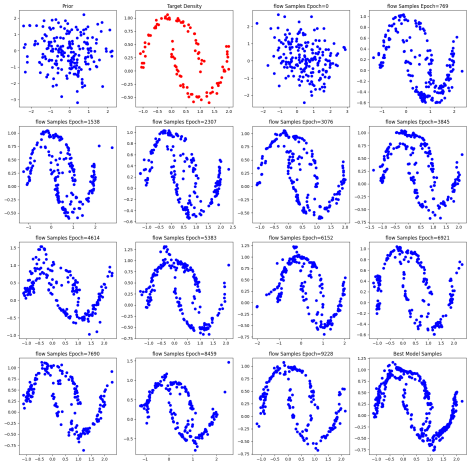


Figure 2: Two Moons example, densities samples at different iterations to see convergence. Number of flows used is 6

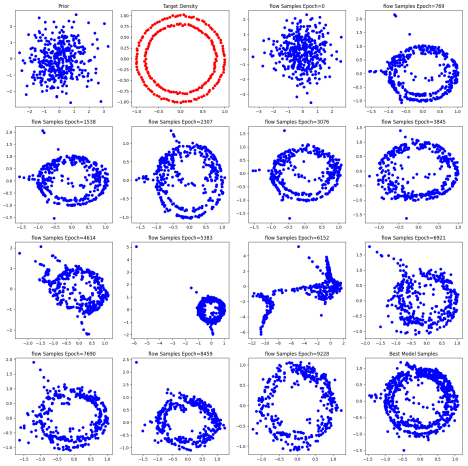


Figure 3: Circle example, densities samples at different iterations to see convergence, Number of flows used is 8

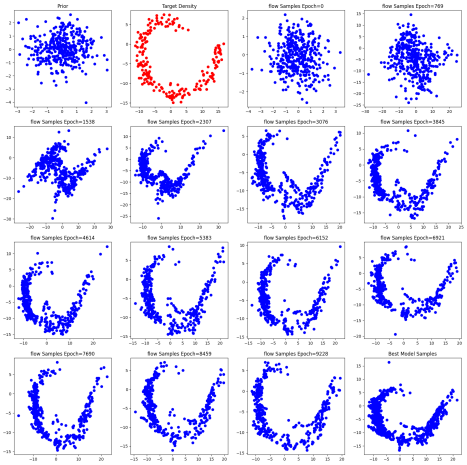


Figure 4: Spiral example, densities samples at different iterations to see convergence, Number of flows used is 6

IV. Future Work and Conclusion

It is clear from the experiments that one normalizing flow model is not enough even for simple distribution, concatenation of multiple normalizing flows is key to convergence and correct modeling of the target distribution.

Although I have much better understanding of the normalizing flows having read a fair few papers in the

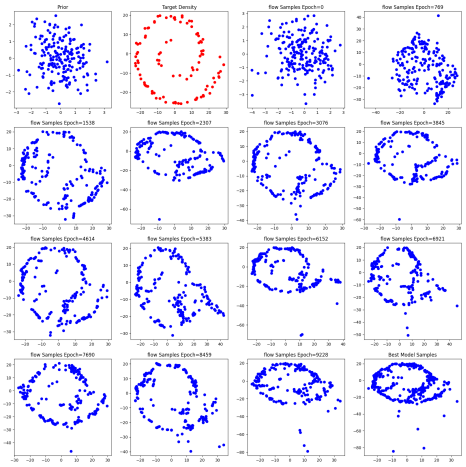


Figure 5: Spiral example 2, densities samples at different iterations to see convergence, Number of flows used is 8

domain. This project fell short of what i had planned to complete. I plan to continue this work for the summer so that i have a better understanding of how these model work and converge for images.

I have a better understanding of normalizing flows with all the variety of experiments executed but I hope to find new observation and new tricks in convergence of normalizing flow for image data.

References

[1] Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." Advances in neural information processing systems 27 (2014).

[2] Papamakarios, George, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. "Normalizing flows for probabilistic modeling and inference." Journal of Machine Learning Research 22, no. 57 (2021): 1-64.

[3] Kobyzev, Ivan, Simon JD Prince, and Marcus A. Brubaker. "Normalizing flows: An introduction and review of current methods." IEEE transactions on pattern analysis and machine intelligence 43, no. 11 (2020): 3964-3979.

[4] Ho, Jonathan, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. "Flow++: Improving flow-based generative models with variational dequantization and architecture design." In International Conference on Machine Learning, pp. 2722-2730. PMLR, 2019.

[5] Lugmayr, Andreas, Martin Danelljan, Luc Van Gool, and Radu Timofte. "Srflow: Learning the super-resolution space with normalizing flow." In European conference on computer vision, pp. 715-732. Springer, Cham, 2020.

[6] Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using real nvp." arXiv preprint arXiv:1605.08803 (2016).

[7] real-nvp coded by chrischute, https://github.com/chrischute/real-nvp

[8] real-nvp coded by fm2, https://github.com/fmu2/realNVP