Probabilistic AI 2022
PhD Course Assignment

Konstantina Nikolaidou, Luleå University of Technology

1 Task 1: Normalizing Flows

This report presents three implementations of Normalizing Flows, Planar Flows [1], Real-NVP Flows [2], and Continuous Normalizing Flows (CNF) [3]. To evaluate these Flows, three datasets are used, provided by the assignment, Two Moons, Two Blobs, and Boomerang. The code can be found in the following link: https://github.com/koninik/dt8122-Normalizing-Flows.

1.1 Planar Flows and Real-NVP

Planar Flows are considered as a simple type of Normalizing Flows where a hyperplane is specified to modify the distribution around it with several contractions and expansions. We implement the followed transformation, as defined in the original paper:

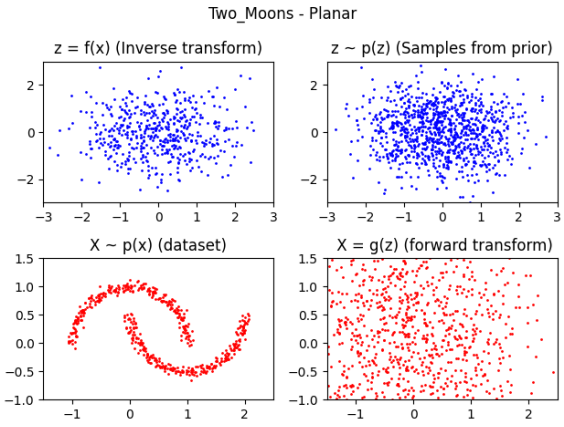
$$f(z) = z + u \tanh(w^T z + b) \tag{1}$$

with w, u, b representing learnable parameters and tanh as the non-linearity function, represented as h in the paper. In order for $f(z)$ to be invertible the condition $w^T u \geq -1$ is used.

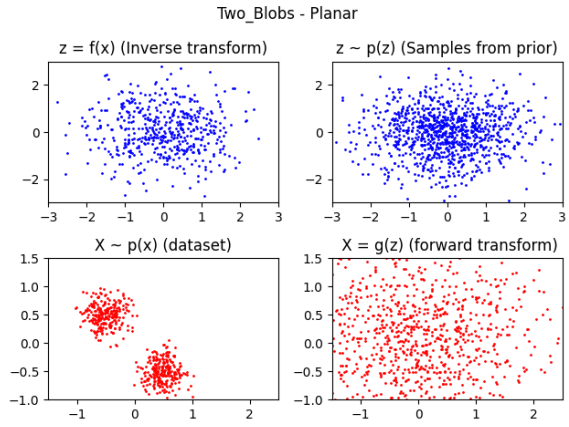
For our implementation, we use 32 Planar transformations and we use RMSprop optimizer as suggested in the paper. Besides that, the implementation relies on the equation mentioned (1) and then the invertibility condition. The resulted plots of the Planar Flow for the three datasets are presented in Figure 1. As we can see from the plots, the results are not the desirable ones. Although the forward function works nicely and we can achieve a transformation from the complex distribution to the prior normal one, the inverse is not possible. This is mainly because of the problem that Planar Flows face due to difficulty of invertibility. Another problem we faced, is that we could not approximate the target distributions, so the resulted loss was not ideal. To tackle these issues, one could use a function with a known inverse and approximate the data with several techniques. Unfortunately, there was limited time to investigate these aspects and make the Planar Flow work properly.

The main characteristic of Real-NVP Flows is the use of Affine Coupling Bijections. The characteristic of these flows is that a variable is splitted, and half of it is undergoing an element-wise affine transformation multiplied with some constant, which is a neural network s that scales it, and then another neural network t used as an offset, that come from the other variables. In our implementation, both networks consist of one input and one hidden layer, followed by GELU non-linearity function [4], and then the output layer. For s network the output layer is followed by a tanh. We use Adam optimizer [5] with a learning rate of 10^{-4}. During sampling, we choose 1K random samples from the prior normal distribution. The resulted plots of the Real-NVP Flow using the three datasets are presented in Figure 2. In Figure 2, we can see that there is some small confusion, especially for the two moons and two blobs datasets, where the two clusters are met. This could be because we used a higher number of samples (1K) in the inverse function (prior to target) compared to the datasets that contain 500 samples. Nevertheless, the results seem satisfactory and the implementation seemed much easier than the Planar Flow, without any issue appearing in the forward and inverse functions. The performance could be increased with some investigation in the parameter values and mostly in the network architectures (layers, activations, etc.), but unfortunately, due to lack of time, we limited the work in one case.

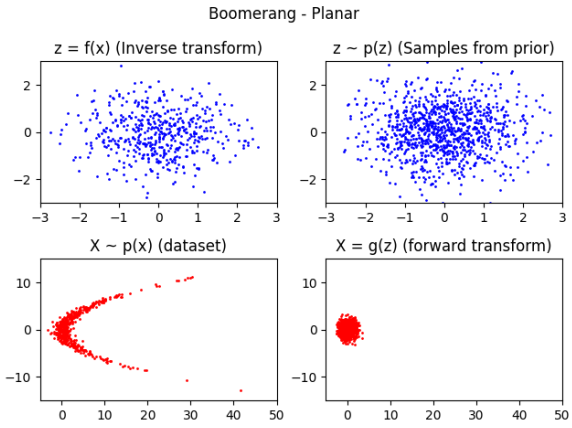
The main difference between Planar and Real-NVP Flows is the invertibility. Although the used restriction assures the invertibility of the Planar Flow function, the inverse function is not known or cannot be computed easily, thus it has to be approximated. On the other hand, the coupling layers



(a) Planar - Two Moons



(b) Planar - Two Blobs



(c) Planar - Boomerang

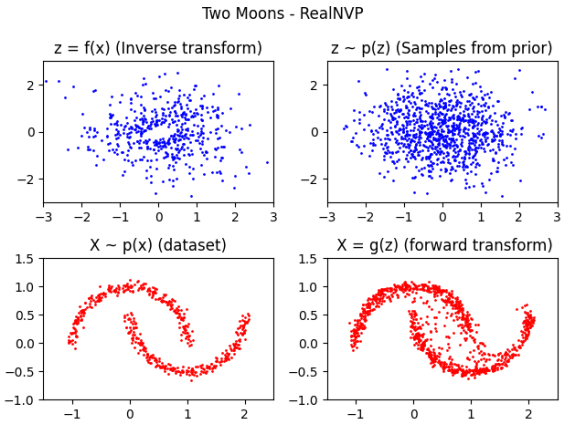
Figure 1: Plots for the Planar Flow method on the three given datasets.

used in Real-NVP help overcome this issue. Thus, Planar Flows lack in expressivity due to the density estimation issue. Furthermore, the affine coupling layers in Real-NVP guarantee invertibility and tractability of the Jacobian. Planar Flows are also more suitable to handle low-dimensional random variables.

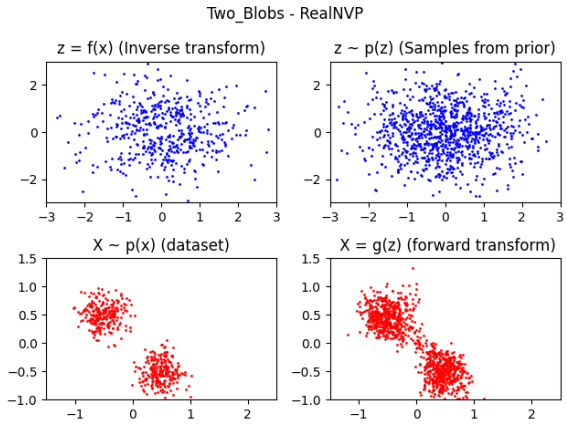
2 Task 2: Continuous Normalizing Flows

Continuous Normalizing Flows (CNF) aim to transform a simple distribution into a complex one with the use of an Ordinary Differential Equation (ODE) Flow instead of a sequence of transformations. As mentioned in [6], the ODE described the evolution of the flow across the different timesteps. The difference with the previous flows is that, instead of computing the log-likelihood for the change of variables, here only the trace of the Jacobian should be computed, which is more efficient. Furthermore, the adjoint method that is used as a continuous-time backpropagation reduces the computation and memory cost.

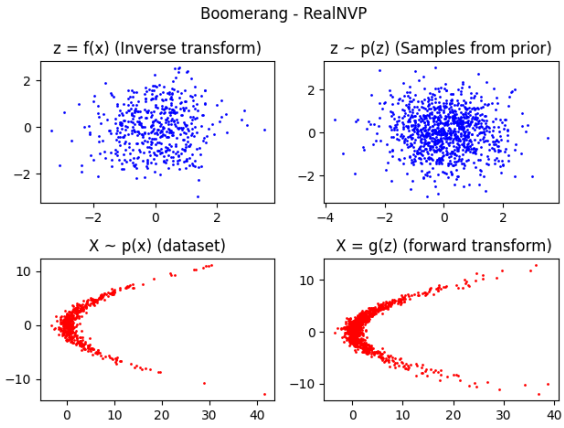
We train for 2000 iterations using a Adam [5] as our optimizer with a learning rate of 10^{-3}. For the transform from the prior normal distribution we choose 2000 points to map into the target distributions. The resulted plots are presented in Figures 3 and 4. In Figure 3, in every subfigure, one can see the dataset distribution (lower left), the prior normal base distribution (upper right), and the forward transform from the prior base to the target distribution (lower right). Due to the use of different functions for the forward and inverse of the flow, the plots of the transformation from the data to the prior base can be seen in Figure 4. As seen in Figure 4, the forward function worked, but for some uknown reason it was stopping in the second timestep. In subfigures (a) and (b), we can already



(a) Real NVP - Two Moons



(b) Real NVP - Two Blobs



(c) Real NVP - Boomerang

Figure 2: Plots for the Real-NVP flow method on the three given datasets.

see how the original data distribution is ruined and tries to transform into a normal distribution. For the boomerang dataset we present both the first (c) and the second timestep (d), as the beginning of the transformation is not clear just by looking at subfigure (d). From the comparison of the two subfigures ((c) and (d)) it is clear that the data points are starting to gather in (0, 0) implying the transformation into the prior. Unfortunately, there was not more time to fix the issue and show the plots of the last timestep.

References

[1] D. J. Rezende and S. Mohamed, "Variational Inference with Normalizing Flows," 2015.

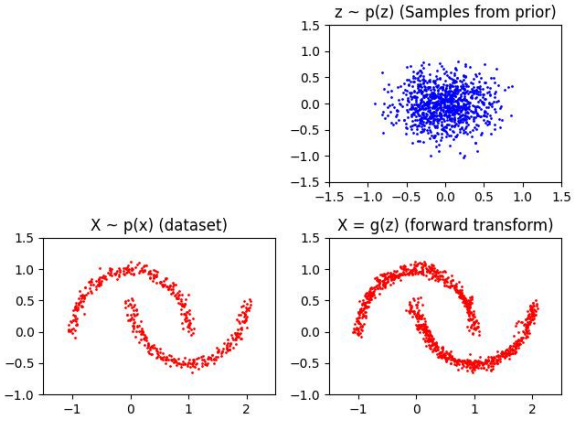
[2] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using Real NVP," 2016.

[3] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural Ordinary Differential Equations," 2018.

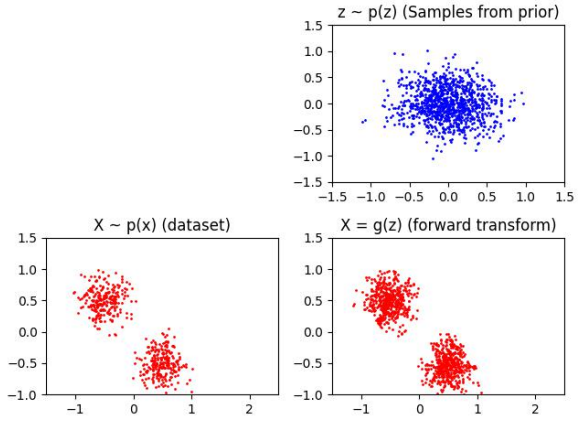
[4] D. Hendrycks and K. Gimpel, "Gaussian Error Linear Units (GELUs)," 2016.

[5] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," 2014.

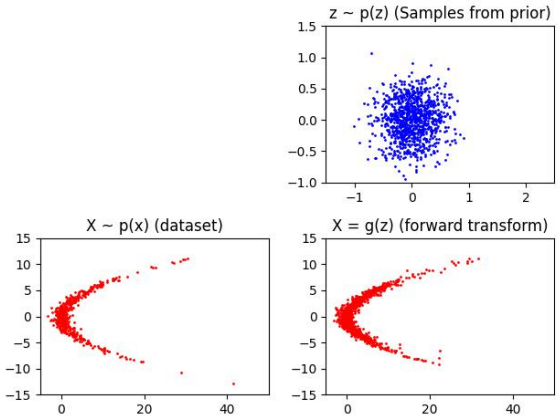
[6] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing Flows for Probabilistic Modeling and Inference," 2019.



(a) CNF - Two Moons

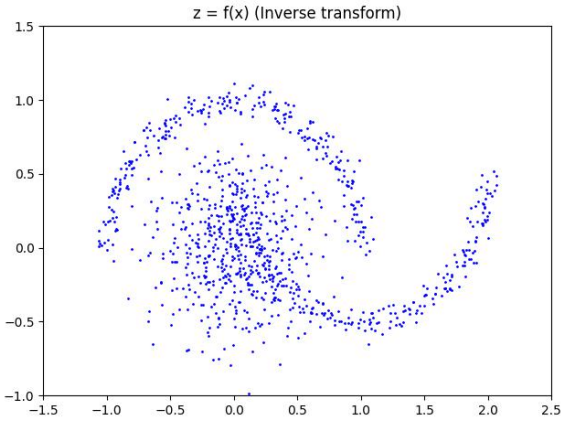


(b) CNF - Two Blobs

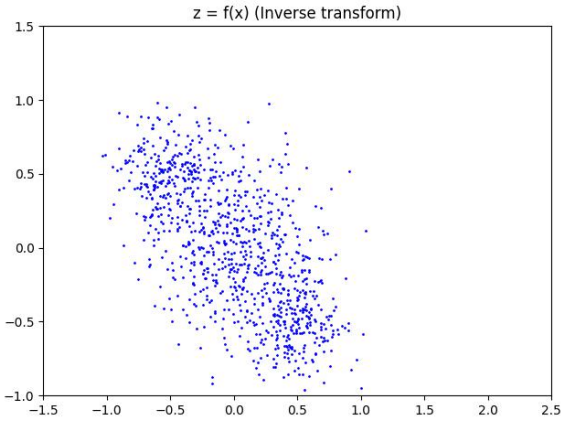


(c) CNF - Boomerang

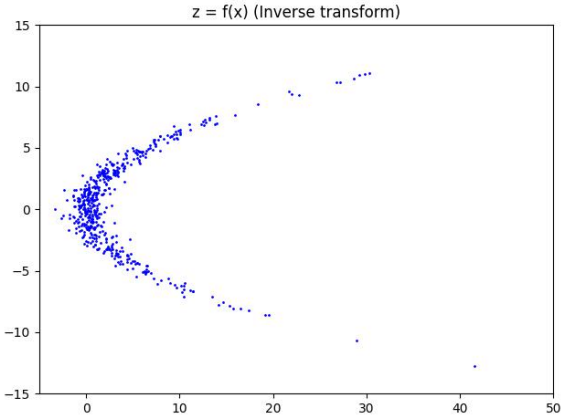
Figure 3: Plots for the CNF Flow method on the three given datasets. The plot of the transformation from the target distribution to the normal base distribution is presented in Figure 4.



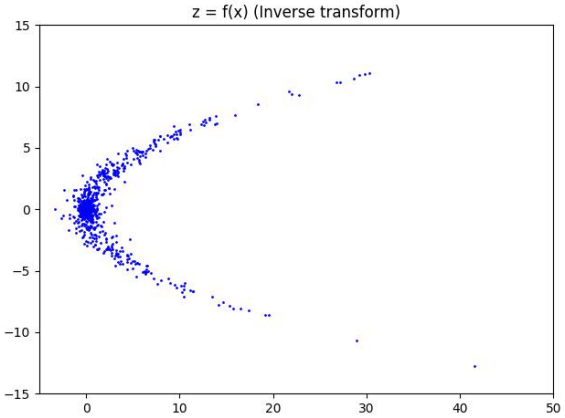
(a) CNF - Two Moons



(b) CNF - Two Blobs



(c) CNF - Boomerang - 1st step



(d) CNF - Boomerang - 2nd step

Figure 4: Transformations from the target distribution to the normal base distribution for the three datasets. All plots, except (c), present the 2nd timestep of the transformation from the data distribution to the normal base distribution. Plot (c) shows the 1st timestep of the transformation which is basically the original data distribution.