# A SYSTEMATIC STUDY OF MASKED AUTOREGRESSIVE FLOWS FOR DENSITY ESTIMATION

*Sergio Hernan Garrido Mejia*

DTU Transport

## ABSTRACT

We studied Masked Autoregressive Flows (MAF) and how hyperparameters affect their performance. In order to uderstand MAFs we study Normalizing Flows (NF) and Masked Autoncoders (MADE). We compared this deep generative model for density estimation against Kernel Density Estimation (KDE) and Guassian Mixtures Model (GMM) using criminal data from Bogota. The performance of MAF is comparable to that of KDE and GMM. Further development of flow models is necessary to exploit their potential.

***Index Terms*—** Density estimation, Normalizing Flows, Masked Autoencoder

## 1. INTRODUCTION

Since their introduction in [1], Normalizing Flows (NF) have not caught the attention that other deep generative models (VAE, GANs, RBM) have caught. Recent advances and applications (such as WaveFlow for speech synethsis) of flow based models show how powerful they are compared to the available alternatives. I tried to do a systematic study of NF starting by understanding their theoretical base, empirical study and recent advances using them.

## 2. METHODOLOGY

In order to understand Masked Autoregressive Flows (MAF) we have to understand two basic concepts: Normalizing Flows (NF) and Masked AutoEncoders for Density Estimation (MADE).

### 2.1. Normalizing Flows

Normalizing Flows were formalized by Jimenez and Mohamed as "the transformation of probability density through a sequence of invertible mappings. By repeatedly applying the rule for change of variables, the initial density 'flows' through the sequence of invertible mappings". To further explain what this definition means, let's present the concept mathematically in a similar way as they did on the paper.

Let $x$ be a random variable distributed according to a distribution $p()$. If we apply an invertible function $f$ to the random variable, the resulting random variable $y = f(x)$ will have a distribution:

$$p(y) = p(f^{-1}(y))|det(J(f^{-1}(y))| \quad (1)$$

It is worth giving a slightly more detailed description of what equation (1) means. Recall that, if $y = f(x)$, then $x = f^{-1}(y)$ and the equation could be re written in the following way: $p(y) = p(x)|det(J((x))|$. In words, we are multiplying the original distribution of x by a positive constant. This positive constant has two key elements. First, the Jacobian, which is a local linear approximation of the function $f$ at point $x$ and the determinant which is the scaling factor of a matrix (or linear transformation). So the second term in equation (1) gives us a constant value which represents either an expansion or shrinking of the distribution $p()$ at point x. This guarantees that any transformation being done on the original random variable $x$ results in a random variable with a valid distribution. Note that estimating both the Jacobian of the inverse of a function and the determinant of the resulting matrix can be computationally expensive so we have to look for transformations which fulfill a series of conditions. First we want transformations that are bijective, second we want functions for which the Jacobian is easily estimated and Jacobians which have an easily estimated determinant.

There are two important points about NF.

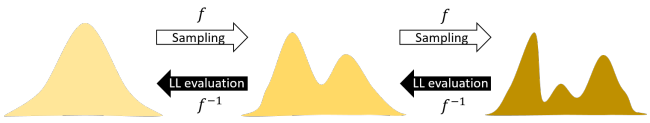


**Fig. 1**. Normalizing Flow representation

### 2.2. Masked Autoencoders for Density Estimation

Masked AutoEncoders for Density Estimation (MADE) were formalized in [2] as a way to do fast and efficient density es-

Thanks to Filipe Rodrigues for his comments and help during the process of this research project.

timation with an AutoEncoder (AE) architecture on a neural network. An AE neural network is one which takes some variables $x$ as input, transforms this input to a latent space (in a lower dimension than the original input space) and tries to output values as close as possible to the original input $\hat{x}$. In order to do density estimation with an AE, we need to assume the that the data behaves autorregresively, meaning that the distribution of $x_d$ is conditioned only on the values of $x_{d-1}$. Mathematically, $p(x) = \prod_{d=1}^{D} p(x_d|x_{<d})$. Since we are making this assumption on the data, we have to make sure that there are no weights from $x_d$ to $x_{d-1}$ $\forall d$. The way to achieve this is by masking (or zeroing out) such connections using a mask $M$ and instead of doing the "classic" affine transform for each hidden layer $a = h(xW + b)$ we would do $a = h(x(W \odot M) + b)$.

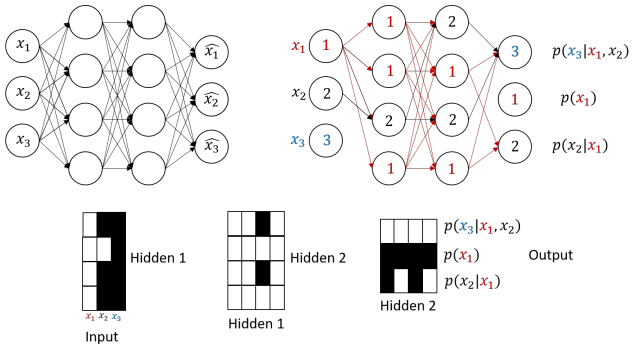


**Fig. 2**. MADE representation. 1) The neural network on the top left is a classic autoencoder, 2) The matrices on the bottom are masks generated randomly for the autoencoder on 1), 3) The neural network on the top right is a MADE network using the architecture on 1) and the masks on 2). Note that MADE is order agnostic and connectivity agnostic so that you can create ensembles of the density estimates on parallel.

Note that instead of outputting $p(x_d)$, MADEs could be used to output a parametrized version of any distribution, e.g., it could output the mean and the standard deviation of Gaussian or the weights, the mean and the standard deviation of a mixture of Gaussian. For the rest of the document, we will parametrize the conditionals with single gaussians: $p(x_i|x_{1:i-1}) = N(x_i|\mu_i, exp(\sigma_i)^2)$ where $\mu_i = f_{\mu_i}(x_{1:i-1})$ and $\sigma_i = f_{\sigma_i}(x_{1:i-1})$. Let $\epsilon_i = N(0,1)$, using the reparametrization trick [3] we can define $x_i = \epsilon_i exp(\sigma_i) + \mu_i$.

MADE are perfect transformations for normalizing flows. They are bijective transformations, they have an easily estimated Jacobian (the derivative of the likelihood function with respect to the input) and, by design, an easily estimated determinant of the Jacobian. The last property because of the autorregresive assumption, since it implies that $\frac{\partial f(x_{d-i})}{\partial x_d} = 0$ implying that the Jacobian is triangular and therefore the determinant is the product of its diagonal.

### 2.3. Masked Autorregresive Flows

We can combine the ideas from NF and MADE to create powerful density estimators. This was introduced in [4]. In section 2.2 we explained the suitability of Masked Autoencoders as transformations in Normalizing Flows, the main point of the Masked Autoregressive Flows (MAF) is to stack Autoencoders on top of each other to achieve better density estimations.

First, we define $x = f(\epsilon)$ as in section 2.2: $x_i = \epsilon_i exp(\sigma_i) + \mu_i$. Since $x$ are points in our data, we would like to retrieve the random numbers $\epsilon$ that generated our data. In order to do this, we need to estimate $\epsilon = f^{-1}(x)$. Inverting the function, $\epsilon_i = (x_i - \mu_i)exp(-\sigma_i)$. As explained before, due the autorregresive structure, the jacobian of $f^{-1}$ is triangular and the determinant is: $|det(\frac{\partial f^{-1}}{\partial x})| = exp(-\sum_i \sigma_i)$ where $\sigma_i = f_{\sigma_i}(x_{1:i-1})$ as defined before.

To make slightly clearer definition of the model, we use $i$ Masked Autoencoders to get vectors $\mu_i$ and $sigmai$. We use these to 1) invert $x$ until we get the random numbers $\epsilon$ that generated our points and 2) to keep track of $\sigma_i$ so we can keep track of the transformations and work with NF as in equation 1.

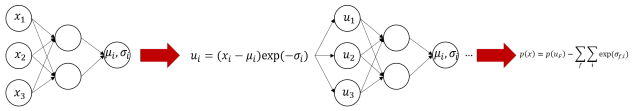


**Fig. 3**. MAF representation. Using iterative Masked Autoregressive networks as transformations into a basic distribution.

## 3. RESULTS

### 3.1. Data and experimental setting

The task to be performed is the density estimation of crime occurrences for a certain hour of the week in Bogota, Colombia. The variables available on the data are latitude, longitude, exact time of the crime (up to seconds) and type of crime. When training the model, we took into account only the day of the week and the 1-hour interval which the point belongs to. We trained the model using all the past crimes from the same day of the week and the same interval available, we further separated the training data into training and validation data using a 10% for validation data. We tested the model using the same day of the week and the same interval for the following four weeks. We train the model for two points in time Wednesday, 03. November 2004 11:00AM-12:00M for which there were 226 points available and Sunday, 03. November 2013 11:00AM-12:00M with 2424 points. We do this to test for the robustness of the model with smaller data sets.

We tested the model for a different number of flows,i.e., number of MADE stacks and different number of units per

flow, each flow had only one hidden layer. Furthermore, early stopping was used in order to try to avoid overfitting.

In crime modelling we are interested in the hit-rate or accuracy of our model, meaning the number of crimes we predict correctly. Since the accuracy of a density estimation is a function of how many points of the grid are chosen as hotspots or points were crimes occur, we compare the models using the Cumulative Accuracy Profile (CAP) measure. These is just a function from some parameter of the model (percentage of points of the grid chosen as hot-spots) to the hit-rate. This curve is similar to the Receiving Operating Curve (ROC). From this curve, we can extract both the Area Under the Curve (AUC) and crime prediction at certain grid coverage,e.g.,1%,5%,10%, etc. Particularly we are interested in the 10% of the coverage since this is the crime literature standard.

### 3.2. Experimental results

The MAF model was compared to a Parzen Window (otherwise known as Kernel Density Estimation) and a Gaussian Mixtures Model (GMM). We optimized the hyper-parameters of the three models using a grid search over the space, for the MAF model the hyper-parameter space contains number of flows, number of hidden layers on each flow, number of hidden units per layer on each flow, etc. but, as stated before, we reduce the hyper-parameter space to number of flows and number of hidden units in one hidden layer of the flow. For KDE we searched for the bandwidth and for GMM we searched for the number of generating Gaussians. The results for both small and large data set are exposed on table 3.2. We can see that for the small data set, the normalizing flows performs better for 10% of coverage than both models, it also has the second best AUC. For the large data set, the MAF performs worse than KDE but the same as GMM. In the large data set, the MAF has the worst performing AUC.

| Small data set (200 training points) | | | | | | |
|---|---|---|---|---|---|---|
| Model | AUC | 1% | 5% | 10% | 15% | 20% |
| MAF | 0.8261 | 0.0476 | 0.2380 | **0.5714** | 0.6190 | 0.6190 |
| KDE | 0.8152 | 0.0476 | 0.2857 | 0.4761 | 0.5714 | 0.6190 |
| GMM | **0.8295** | 0.0476 | 0.2857 | 0.4285 | 0.6666 | 0.7619 |
| Big dataset (2000 training points) | | | | | | |
| Model | AUC | 1% | 5% | 10% | 15% | 20% |
| MAF | 0.8806 | 0.125 | 0.3125 | 0.4375 | 0.625 | 0.875 |
| KDE | **0.9018** | 0.125 | 0.375 | **0.625** | 0.75 | 0.8125 |
| GMM | 0.8937 | 0.125 | 0.3125 | 0.4375 | 0.6875 | 0.9375 |

**Table 1**. Comparison of MAF, KDE and GMM using the Cumulative Accuracy Profile (CAP) using the percentage of points of the grid as the classifying parameter.

Note it is important to take these results with a grain of salt since only two intervals were tested. In order to get more reliable and and accurate results, at least a whole week should be tested, taking the final CAP as the average for every day.

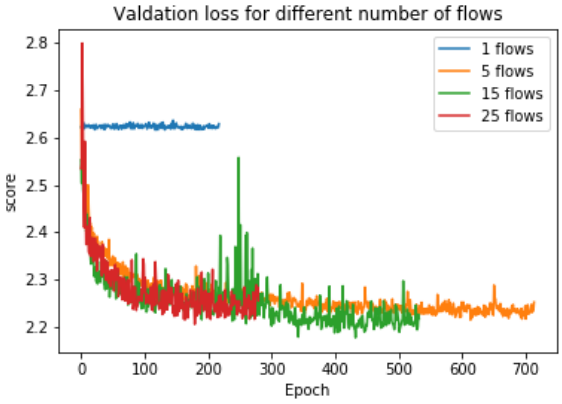


**Fig. 4**. Validation loss with different number of flows. Even though up to 100 flows were tested, they are not presented in this graph since its validation loss has high peaks which do not allow to see the behavior of the rest.

## 4. CONCLUSIONS AND FURTHER WORK

Some of the concluding remarks are the expected ones: a higher number of flows or hidden units increase the estimated density complexity and hence the fit of the training data. This is something to be very careful about because with complexity increases overfitting. Regularization techniques become indispensable when working with normalizing flows.

Training is highly unstable, same architecture can give extremely different results, since this stochasticity in these models has many sources:the minibatches for estimation, the order of the variables for the MADE, the initial weights of the MADE flows, etc. An important area of research on the NF literature could be to look to make these models stable.

One of the impressive conclusions of the use of these models is it's capabilities with a few amount of data points. As seen on table 3.2, MAF is the best performing among the three models on the small data set. Results, however, are not as good with high dimensionality of the data, working on scalable NF is a possibly interesting (and important area of research).

Finally, The estimations we made for this document were based on single intervals of time. Even though we can estimate the density estimation model for every interval for policy purposes, it would be interesting to do conditioning on these models on any variable (in our case, time) in order to have a much more robust model.

## 5. REFERENCES


[1] Danilo Jimenez Rezende and Shakir Mohamed, "Variational Inference with Normalizing Flows," *arXiv e-prints*, p. arXiv:1505.05770, May 2015.

[2] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle, "Made: Masked autoencoder for distribution estimation," in *Proceedings of the 32nd International Conference on Machine Learning*, Francis Bach and David Blei, Eds., Lille, France, 07–09 Jul 2015, vol. 37 of *Proceedings of Machine Learning Research*, pp. 881–889, PMLR.

[3] Diederik P Kingma and Max Welling, "Auto-Encoding Variational Bayes," *arXiv e-prints*, p. arXiv:1312.6114, Dec. 2013.

[4] George Papamakarios, Theo Pavlakou, and Iain Murray, "Masked Autoregressive Flow for Density Estimation," *arXiv e-prints*, p. arXiv:1705.07057, May 2017.

[5] Ilya Kostrikov, "Pytorch-flows," https://github.com/ikostrikov/pytorch-flows, 2018.


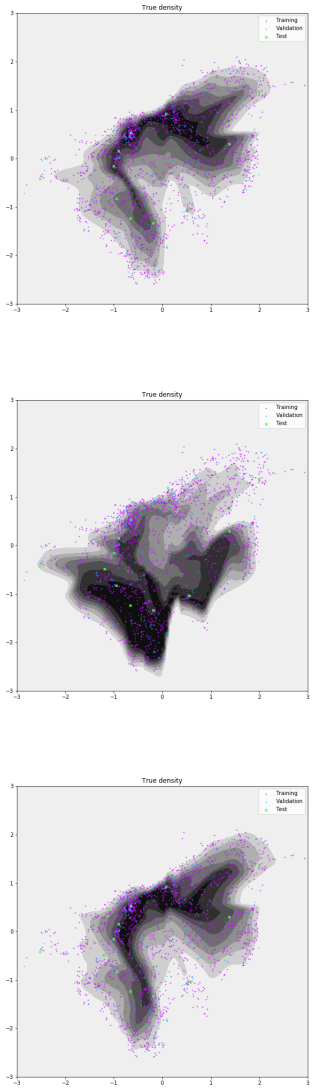


**Fig. 5**. Instability of training: different results for same architecture of 25 flows and 2048 hidden units on each flow.