EPiC Series in Computing

Volume 79, 2021, Pages 82–90

Proceedings of ISCA 34th International Conference on Computer Applications in Industry and Engineering

Mixtures of Normalizing Flows

Sebastian Ciobanu

Faculty of Computer Science
Alexandru Ioan Cuza University, Iași, Romania
aciobanusebi@gmail.com

Abstract

Normalizing flows fall into the category of deep generative models. They explicitly model a probability density function. As a result, such a model can learn probabilistic distributions beyond the Gaussian one. Clustering is one of the main unsupervised machine learning tasks and the most common probabilistic approach to solve a clustering problem is via Gaussian mixture models. Although there are a few approaches for constructing mixtures of normalizing flows in the literature, we propose a direct approach and use the masked autoregressive flow as the normalizing flow. We show the results obtained on 2D datasets and then on images. The results contain density plots or tables with clustering metrics in order to quantify the quality of the obtained clusters. Although on images we usually obtain worse results than other classic models, the 2D results show that more expressive mixtures of distributions (than the Gaussian mixture models) can be learned indeed. The code which implements this method can be found at https://github.com/aciobanusebi/nf-mixture.

1 Introduction

Machine learning can be roughly split into two categories: supervised learning and unsupervised learning. The latter one covers a wide range of tasks: density estimation, clustering, dimensionality reduction, anomaly detection, etc. The first two tasks will constitute the main focus of this paper. There are classic approaches to density estimation like simply using a multivariate Gaussian/normal distribution [13, sec. 3.2] and fitting the data using the maximum likelihood principle (MLE) [13, sec. 4.2]. Besides a Gaussian distribution, a mixture of distributions can be formed: Gaussian mixture model (GMM) [13, sec. 3.5.1]. The mixture models can be successfully applied in clustering: after fitting the mixture parameters on the given data using MLE, one can compute for each instance i and cluster c the posterior probability for i to be in c, given the instance i—i.e. $P(i \in c|i)$.

Deep learning has gained more and more popularity in recent years. New probabilistic machine learning algorithms were also developed due to deep learning. The new era of generative models spans the following categories: autoregressive models, variational autoencoders (VAEs) [13, sec. 20.3.5], normalizing flows (NFs) [13, sec. 19.3.6.3], and generative adversarial networks (GANs) [13, sec. 19.3.6.2]. The first and the third categories of algorithms model explicitly the probability density function. We will discuss only the normalizing flows, by creating a mixture of these models.

The structure of this paper is as follows:

1. Section 2 contains related work;
2. in Section 3 we include information on our method, methodology, and results;
3. in Section 4 we present the conclusion and the ideas for future work.

2 Related Work

The literature provides a few examples of mixtures of normalizing flows, but no construction of such mixtures is done straightforwardly as we will do.

In [4], a mixture of NFs is created, by partitioning the $\mathbb{R}^d$ space into disjoint subsets and by combining real and discrete latent variables. If a VAE is used and the distribution of the observed variables given the latent ones is a mixture of NFs, then this describes the approach in [16], which applies the mixture in the context of point cloud generation and reconstruction. The mixture of NFs in [15] differs from the straightforward one by introducing variational inference when maximizing the likelihood function. Handling discrete variables in probabilistic models can be problematic and a possible solution is to use the Gumbel-Softmax relaxation; the authors in [10] created a more advantageous solution by using a discrete mixture of NFs.

3 Method and Experiments

3.1 Method

We propose a modality to directly build a mixture of NFs.

In general, a mixture of distributions can be defined as follows:

$$p(x; \text{parameters}) = \pi_1 p_1(x; \text{parameters}_1) + \cdots + \pi_k p_k(x; \text{parameters}_k),$$

where $\sum_{i=1}^{k} \pi_i = 1$, $\pi \in [0, 1]$, p_i are probability/density functions, and k is the number of distributions—or clusters in the context of clustering.

In general, to define a normalizing flow the following are mandatory: observable variables X, latent variables Z, and a deterministic invertible mapping such that $X = f_{\text{parameters}}(Z)$. The probability/density $p(x)$ can be computed via the change of variables formula:

$$p_X(x; \text{parameters}) = p_Z(f^{-1}_{\text{parameters}}(x)) \left| \det \left(\frac{\partial f^{-1}_{\text{parameters}}(x)}{\partial x} \right) \right| .$$

Usually, $Z \sim \mathcal{N}(\mu, \Sigma)$, where μ and Σ are the parameters of a multivariate normal distribution. An example of normalizing flow is given by the masked autoregressive flow (MAF) [14]. We chose to work with a MAF model because it is more expressive than other NFs—e.g. Real NVP [3]—; it is also known that its sampling process takes a lot of time, but the likelihood evaluation time is short, which is what we need in our context—i.e. we do not sample, but we compute likelihoods.

Specifically, we work with the following mixture model:

$$p(x; \text{parameters}) = \pi_1 \text{MAF}_1(x; \text{parameters}_1) + \cdots + \pi_k \text{MAF}_k(x; \text{parameters}_k),$$

where $\sum_{i=1}^{k} \pi_i = 1$, $\pi \in [0,1]$, MAF_i are probability/density functions, and k is the number of distributions—or clusters in the context of clustering. For each MAF_i, there is a different corresponding masked autoencoder (MADE) [6] and a different corresponding Z distribution:

$$Z \sim \mathcal{N}\left(\mu = \begin{pmatrix} \text{sample from } \mathcal{U}(0,1) \\ \text{sample from } \mathcal{U}(0,1) \\ \vdots \\ \text{sample from } \mathcal{U}(0,1) \end{pmatrix}, \Sigma = \begin{pmatrix} 0.1 & 0 & \dots & 0 \\ 0 & 0.1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0.1 \end{pmatrix}\right),$$

where $\mathcal{U}(0,1)$ represents the continuous uniform distribution on the interval $[0,1]$. The variances were heuristically set to 0.1 because every dataset is pre-processed such that all the values in the data are in the interval $[0,1]$.

3.2 Experiments

We work with the following dataset types:

- toy 2D datasets
  - circles
  - moons
  - pinwheel
  - smile
  - two bananas
- image datasets
  - MNIST [11]: 70000 images, $28 \times 28 = 784$ pixels, 10 classes
  - MNIST5: MNIST only with the digits 0, 1, 2, 3, 4: 35735 images, $28 \times 28 = 784$ pixels, 5 classes
  - Fashion MNIST (F-MNIST) [18]: 70000 images, $28 \times 28 = 784$ pixels, 10 classes
  - CIFAR-10 [9]: 60000 images, $32 \times 32 \times 3 = 3072$ pixels, 10 classes.

Every dataset is pre-processed such that for each attribute the minimum value is 0, and the maximum value is 1. For the image datasets, we also applied our model to the 100 dimensions returned by the Principal Component Analysis (PCA) [13, sec. 20.1] algorithm.

All the masked autoencoders are different, but they have the same neural network architecture—a multilayer perceptron (MLP) [13, sec. 13.2]—per dataset type:

- toy 2D datasets: 1 layer with 10 units, tanh activation function, Glorot uniform initialization
- image datasets: 1 layer with 1024 units, tanh activation function, Glorot uniform initialization.

We maximize the likelihood or, equivalently, minimize the negative log-likelihood, by using an acceleration of the gradient descent algorithm: Adam [8] with the parameters set to the default values in TensorFlow [1]. The implementation was written using the TensorFlow probability library [2]. The $\pi_1, \dots, \pi_k$ variables are initialized in a uniform manner: $\pi_1^{\text{init}} = \frac{1}{k}, \dots, \pi_k^{\text{init}} = \frac{1}{k}$.

Other training parameters differ among the two categories:

- toy 2D datasets: 2000 epochs; batch size of 4000; shuffle buffer size of 4000
- image datasets: 10 epochs; batch size of 4000 (MNIST), 3500 (F-MNIST), 1000 (CIFAR10), and 1000 (100-dimensional version of MNIST, F-MNIST, CIFAR10 via PCA); shuffle buffer size of 1024

We compare our model with the following algorithms:

- random clustering
- k-means [13, sec. 21.3]
- Expectation-Maximization (EM) [13, sec. 8.7.2] for GMMs
- for MNIST5, we also report the results in [15].

We ran each algorithm 10 times and retained only the best run in terms of likelihood maximization or, equivalently, of loss minimization. For this best run, we compute the following clustering evaluation metrics:

- purity [13, sec. 21.1.1.1],
- adjusted Rand index (ARI) [7],
- Normalized Mutual Information (NMI) [5],
- unsupervised cluster accuracy (ACC) [12].

3.3 Results

3.3.1 Toy 2D datasets

The results are presented in Figures 1–2. Figure 1 shows the results of applying our algorithm from three perspectives: by labeling the points with the predicted labels, by drawing the decision boundaries between the clusters, and by density plots. On the circles dataset the results are poor. The best results are obtained on the pinwheel and the two bananas datasets. One remarkable observation is that the density plots highlight that the mixture tries to cover the data with more flexibility than a normal distribution would do this—i.e. the shapes of the distributions are not ellipses as in the case of the normal distribution. Figure 2 strengthens this idea by showing other non-optimal density plots.

3.3.2 Image datasets

The results are presented in Tables 1–4. Table 1 presents the clustering metric values on the image datasets. The mixture of NFs attains the best score in no situation. Nevertheless, it is significantly better than the random model and in at least three situations it attains scores approximately equal to the ones returned by the EM/GMM. Table 2 is taken from [15] and it regards only the MNIST5 dataset. We wanted to compare our model to theirs and these were the quantitative measures we could extract from their paper. As a result, Tables 3–4 come into play. Without using PCA, we obtain only 3 clusters instead of 5—in fact, we obtain 2 empty clusters—: cluster 1 mainly contains images of digit 1, cluster 2 mainly contains images of digit 4, and the other 3 digits are mainly in cluster 0. When using PCA, somehow the same pattern occurs: two clusters mainly contain only one category of digits; the difference is that now there are no empty clusters. From this perspective, Table 2 is in accordance with Tables 3–4 up to a certain level.

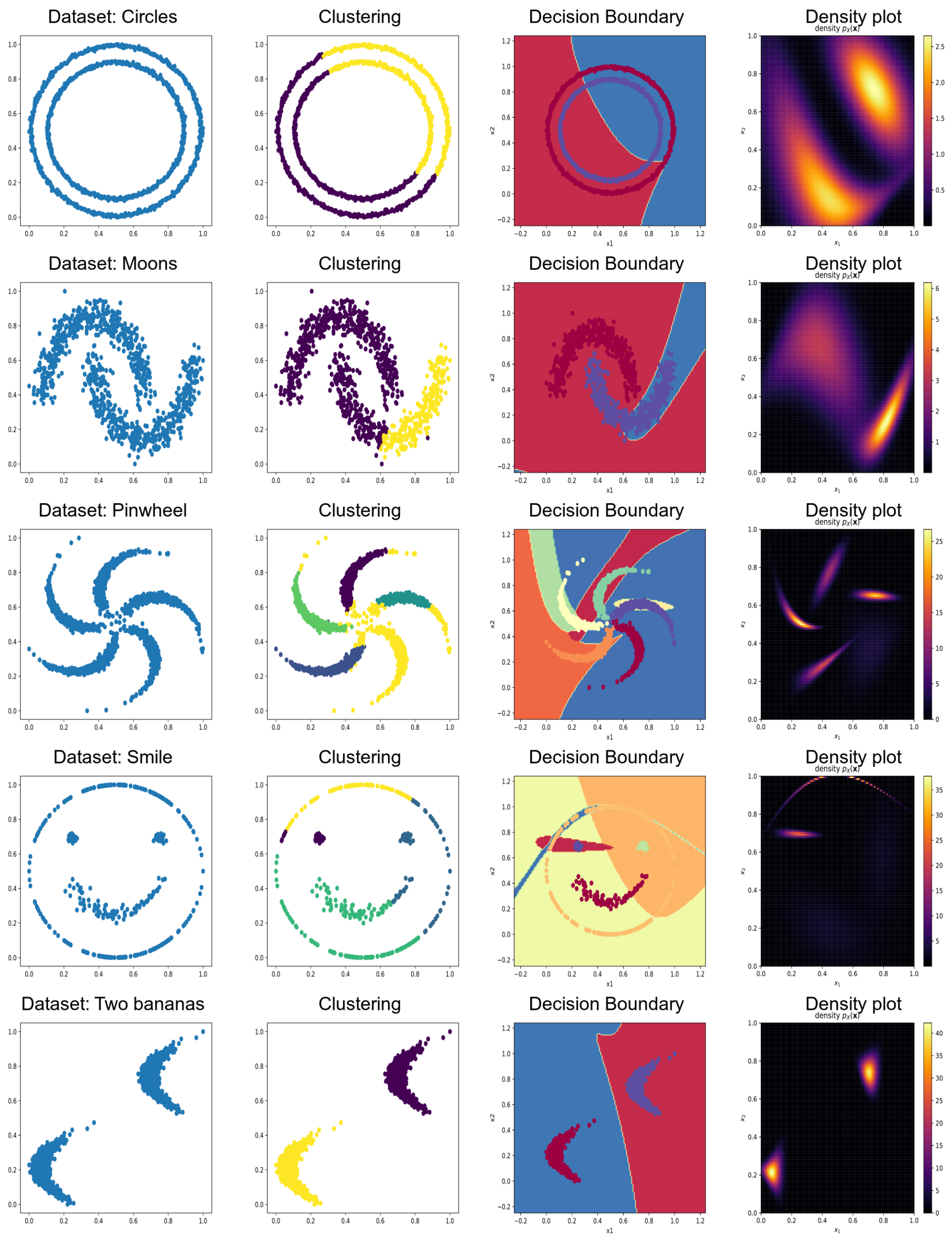

Figure 1: The results on the toy 2D datasets. Four columns: the dataset, the best clustering result in 10 runs in terms of likelihood maximization, the decision boundary created when clustering via the mixture using the best model—the points are colored using the real labels—, the density plot (using the code in [17]) of the mixture using the best model—low densities are darker and high density values are lighter

Metric	Data	Random	k-means	EM/GMM	Mixture of NFs
purity$\uparrow$	MNIST	0.1133	**0.58035**	0.4102	0.2828
	MNIST5	0.2204	**0.8810**	$\underline{0.5678}$	$\underline{0.5663}$
	F-MNIST	0.1065	0.5546	**0.5587**	0.4626
	CIFAR10	0.1063	**0.2212**	0.2065	0.1
	MNIST$_{\text{PCA}}$	0.1132	**0.5807**	0.5799	0.3440
	MNIST5$_{\text{PCA}}$	0.2204	**0.8810**	0.8650	0.6779
	F-MNIST$_{\text{PCA}}$	0.1054	**0.5757**	0.5529	0.3358
	CIFAR10$_{\text{PCA}}$	0.1057	0.2212	**0.298**	0.1830
ARI$\uparrow$	MNIST	0	**0.3646**	0.1948	0.1518
	MNIST5	0	**0.7325**	$\underline{0.2639}$	$\underline{0.3965}$
	F-MNIST	0	0.3481	**0.3728**	0.3174
	CIFAR10	0	**0.0417**	0.0226	0
	MNIST$_{\text{PCA}}$	0	**0.3618**	0.3449	0.1870
	MNIST5$_{\text{PCA}}$	0	**0.7325**	0.6773	0.4643
	F-MNIST$_{\text{PCA}}$	0	**0.3741**	0.3711	0.1757
	CIFAR10$_{\text{PCA}}$	0	0.0417	**0.0940**	0.0333
NMI$\uparrow$	MNIST	0.0002	**0.4845**	0.3285	0.2901
	MNIST5	0	**0.7099**	0.3513	0.5624
	F-MNIST	0.0002	0.5119	**0.5365**	0.4924
	CIFAR10	0.0003	**0.0793**	0.0569	0
	MNIST$_{\text{PCA}}$	0.0002	0.4831	**0.5322**	0.3095
	MNIST5$_{\text{PCA}}$	0.0001	0.7099	**0.7357**	0.5161
	F-MNIST$_{\text{PCA}}$	0.0002	0.5124	**0.5726**	0.3515
	CIFAR10$_{\text{PCA}}$	0.0003	0.0792	**0.1619**	0.0567
ACC$\uparrow$	MNIST	0.1059	**0.542**	0.3766	0.2828
	MNIST5	0.2028	**0.8810**	$\underline{0.5678}$	$\underline{0.5663}$
	F-MNIST	0.1061	0.4740	**0.5396**	0.462
	CIFAR10	0.1055	**0.2062**	0.1915	0.1
	MNIST$_{\text{PCA}}$	0.1048	**0.5432**	0.5371	0.3412
	MNIST5$_{\text{PCA}}$	0.2048	**0.8810**	0.8650	0.6779
	F-MNIST$_{\text{PCA}}$	0.1045	**0.5399**	0.5027	0.3266
	CIFAR10$_{\text{PCA}}$	0.1055	0.2060	**0.2873**	0.1822

Table 1: Quantitative comparisons of the clustering algorithms on MNIST, MNIST5, F-MNIST, and CIFAR10. Each algorithm was run 10 times, and only the best run in terms of likelihood maximization was reported. For each row, the highest value was highlighted in bold. We underlined the cases where the NF mixture's metric surpasses or is almost equal to the EM/GMM metric.

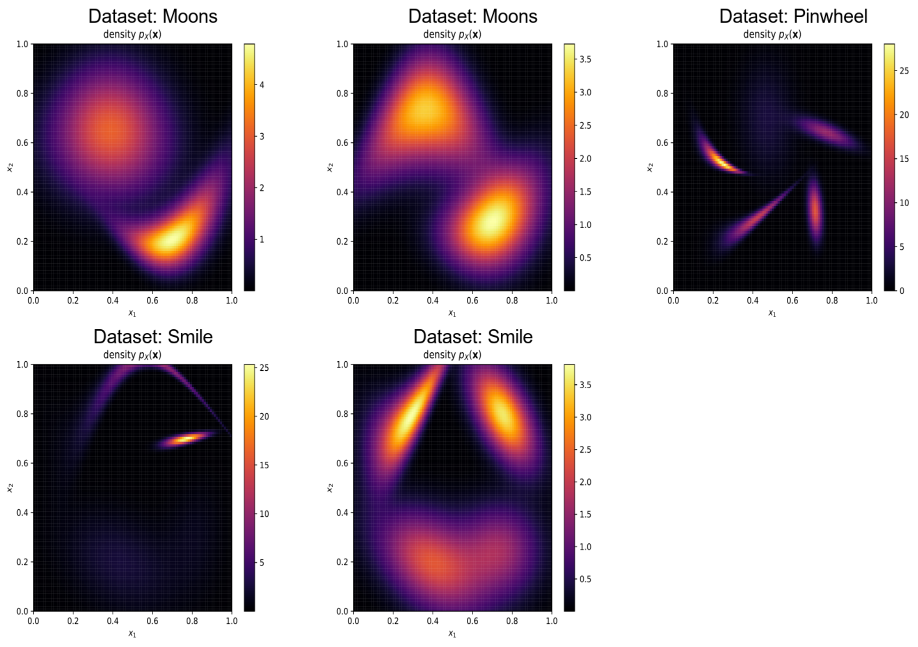

Figure 2: Across the toy 2D datasets: selected density plots from the 10 runs which were not the best in terms of likelihood maximization, but which express that the component distributions differ from the Gaussian distribution

True label\Cluster index	cluster 0	cluster 1	cluster 2	cluster 3	cluster 4
digit 0	0.000602	0.012432	0.002807	**0.982555**	0.001604
digit 1	0.002139	0.020146	**0.977001**	0.000178	0.000535
digit 2	0.000802	**0.952276**	0.011630	0.007219	0.028073
digit 3	0.001558	**0.479455**	0.300682	0.004284	0.214021
digit 4	**0.646166**	0.347273	0.005125	0.001435	0.000000

Table 2: Normalized contingency table for the clustering in [15] on the MNIST5 dataset. Directly taken from [15]. We wrote in bold the maximum value from each row.

True label\Cluster index	cluster 0	cluster 1	cluster 2	cluster 3	cluster 4
digit 0	**0.9818**	0	0.0181	-	-
digit 1	0.0712	**0.8852**	0.0435	-	-
digit 2	**0.9457**	0.0002	0.0539	-	-
digit 3	**0.9413**	0.0042	0.0544	-	-
digit 4	0.0489	0.0002	**0.9507**	-	-

Table 3: Normalized contingency table for our best clustering on the MNIST5 dataset. We wrote in bold the maximum value from each row.

True label\Cluster index	cluster 0	cluster 1	cluster 2	cluster 3	cluster 4
digit 0	0.0111	0	**0.9210**	0.0078	0.0599
digit 1	**0.9841**	0.0044	0.0002	0.0096	0.0015
digit 2	0.2184	0.0364	0.0908	**0.4035**	0.2506
digit 3	0.4178	0.0102	0.0194	0.0222	**0.5301**
digit 4	0.0932	0.0043	0.0118	**0.8900**	0.0004

Table 4: Normalized contingency table for our best clustering on the MNIST5$_{\text{PCA}}$ dataset. We wrote in bold the maximum value from each row.

4 Conclusion and Future Work

We combined three ideas from the unsupervised machine learning field: density estimation, clustering, and deep generative models. We proposed a straightforward mixture of NFs—a mixture of MAFs, specifically— to be used in the context of clustering. The experiments were on 2D datasets and on image datasets. Density plots and clustering metrics were presented. The results show that indeed the mixture of NAFs is more expressive than a GMM, but in the context of images, the results are not necessarily encouraging, although a hyperparameter tweak should be further explored.

Besides the hyperparameter search, other future work can include replacing MAFs with other NFs to see how this influences the results. Moreover, one can replace the PCA pre-processing with the features given by starting to feed the images into a pre-trained neural network and stopping at a specific layer. Furthermore, other initialization techniques for μ—like the centroids after fitting a k-means model—can be used instead of the uniformly distributed μ means in the interval $[0, 1]$.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] Joshua V Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A Saurous. Tensorflow distributions. *arXiv preprint arXiv:1711.10604*, 2017.

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

[4] Laurent Dinh, Jascha Sohl-Dickstein, Hugo Larochelle, and Razvan Pascanu. A RAD approach to deep mixture models. *arXiv preprint arXiv:1903.07714*, 2019.

[5] Pablo A Estévez, Michel Tesmer, Claudio A Perez, and Jacek M Zurada. Normalized mutual information feature selection. *IEEE Transactions on neural networks*, 20(2):189–201, 2009.

[6] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. MADE: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889. PMLR, 2015.

[7] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[10] Tomasz Kuśmierczyk and Arto Klami. Reliable categorical variational inference with mixture of discrete normalizing flows. *arXiv preprint arXiv:2006.15568*, 2020.

[11] Yann LeCun, Corinna Cortes, and CJ Burges. MNIST handwritten digit database, 2010.

[12] Erxue Min, Xifeng Guo, Qiang Liu, Gen Zhang, Jianjing Cui, and Jun Long. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, 6:39501–39514, 2018.

[13] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2021.

[14] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *arXiv preprint arXiv:1705.07057*, 2017.

[15] Guilherme GP Pires and Mário AT Figueiredo. Variational mixture of normalizing flows. *arXiv preprint arXiv:2009.00585*, 2020.

[16] Janis Postels, Mengya Liu, Riccardo Spezialetti, Luc Van Gool, and Federico Tombari. Go with the flows: Mixtures of normalizing flows for point cloud generation and reconstruction. *arXiv preprint arXiv:2106.03135*, 2021.

[17] Louis C Tiao. Building Probability Distributions with the TensorFlow Probability Bijector API. *tiao.io*, 2018.

[18] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.