# PixelCNN as a Single-Layer Flow

**Didrik Nielsen**
Technical University of Denmark
didni@dtu.dk

**Ole Winther**
Technical University of Denmark
University of Copenhagen
olwi@dtu.dk

## Abstract

Flow models have recently made great progress at modelling sensor data such as images, audio, etc., but are still outperformed by autoregressive models. We present a novel interpretation of several existing autoregressive models, including WaveNet, PixelCNN and PixelCNN++, as single-layer flow models defined through an invertible transformation between uniform noise and data samples. These models thus admit a latent representation of data. To demonstrate the usefulness of this interpretation, we explore the latent space of PixelCNN and PixelCNN++ by inspecting latent image representations and performing latent space interpolations.

## 1 Introduction

Sensor data like images and audio arise from physical sensors where continuous analog signals are converted into digital signals. *Quantization* is used to convert continuous signals to a finite set of discrete values such as $\{0, 1, 2, ..., 255\}$. Signals are typically stored in a quantized format. *Dequantization* refers to the process of converting the discrete, quantized values back to continuous values. Typically, a quantized value $q \in \{0, 1, 2, ..., 255\}$ is converted to a dequantized value $x \in [0, 1]$ by adding uniform noise and scaling, i.e.

$$x = \frac{q + u}{256} \quad \text{where} \quad u \sim \text{Unif}(0, 1). \tag{1}$$

**Flow models** such as RealNVP [3] and Glow [5] model $\boldsymbol{x} \in [0, 1]^D$ using a continuous distribution defined using an invertible transformation $f$ of a latent variable $\boldsymbol{z}$ from a base distribution $p(\boldsymbol{z})$, i.e.

$$\boldsymbol{x} = f^{-1}(\boldsymbol{z}) \quad \text{where} \quad \boldsymbol{z} \sim p(\boldsymbol{z}) \tag{2}$$

such that the density can be computed as

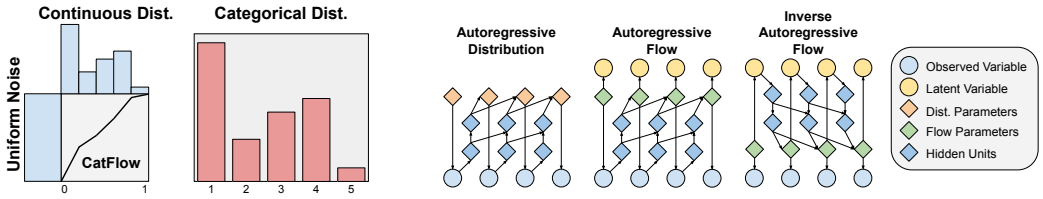
$$p(\boldsymbol{x}) = p(\boldsymbol{z}) \left| \det \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}} \right| = p(f(\boldsymbol{x})) \left| \det \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} \right|. \tag{3}$$

**Autoregressive models** such as PixelCNN and related models [14, 13, 12, 4, 11, 10, 1, 7, 2], on the other hand, model $\boldsymbol{q} \in \{0, 1, 2, ..., 255\}^D$ directly with a discrete distribution, i.e.

$$p(\boldsymbol{q}) = \prod_{d=1}^{D} p(q_d | \boldsymbol{q}_{1:d-1}). \tag{4}$$

We will show that autoregressive models relying on discrete conditional distributions can equivalently be viewed as invertible transformations between uniform noise and data samples.

4th workshop on Bayesian Deep Learning (NeurIPS 2019), Vancouver, Canada.




(a) (b)

Figure 1: Illustration of the similarity between **(a)** the categorical distribution and the continuous analogue defined by CatFlows and **(b)** autoregressive distributions, autoregressive flows and inverse autoregressive flows, exemplified using a causal convolutional autoregressive network [12].

## 2 PixelCNN as a Flow

**PixelCNN** and related models [14, 13, 12, 4, 7, 2] model quantized sensor data $\boldsymbol{q}$ such as audio, images and video using an autoregressive network to parameterize a Categorical distribution, i.e.

$$p(\boldsymbol{q}) = \prod_{d=1}^{D} \text{Cat}\left(q_d | \boldsymbol{q}_{1:d-1}\right) = \prod_{d=1}^{D} \prod_{k=1}^{K} \pi_{dk}(\boldsymbol{q}_{1:d-1})^{\mathbb{I}(q_d=k)}. \tag{5}$$

**A flow equivalent** can be developed using what we will term *CatFlows*, a piecewise linear transformation [8] which gives rise to a continuous analogue of the categorical distribution.

Consider a piecewise linear transformation between the points $\{(x_k, z_k)\}_{k=0}^{K}$, where

$$0 \equiv x_0 < x_1 < x_2 < ... < x_K \equiv 1 \tag{6}$$

$$0 \equiv z_0 < z_1 < z_2 < ... < z_K \equiv 1. \tag{7}$$

We define a *CatFlow* as the case where $x_k = \frac{k}{K}$ and $z_k = \sum_{l=1}^{k} \pi_l$ with $\sum_{l=1}^{K} \pi_l = 1$.
For $x_{k-1} \leq x < x_k$ and $z_{k-1} \leq z < z_k$, a CatFlow $f$ and its inverse $f^{-1}$ can be written as

$$z = f(x) = \text{CatFlow}(x|\boldsymbol{\pi}) = z_{k-1} + \pi_k \frac{x - x_{k-1}}{x_k - x_{k-1}} \tag{8}$$

$$x = f^{-1}(z) = \text{CatFlow}^{-1}(z|\boldsymbol{\pi}) = x_{k-1} + \frac{1}{K} \frac{z - z_{k-1}}{z_k - z_{k-1}}. \tag{9}$$

By transforming uniform noise $z$ using a CatFlow, i.e. $x = \text{CatFlow}^{-1}(z|\boldsymbol{\pi})$ with $z \sim \text{Unif}(z|0, 1)$, we obtain a density

$$p(x|\boldsymbol{\pi}) = K \prod_{k=1}^{K} \pi_k^{\mathbb{I}(x \in [\frac{k-1}{K}, \frac{k}{K}))} \tag{10}$$

which is a continuous analogue of the categorical distribution. See App. A for more details. Consequently, specifying an autoregressive flow [6, 9] with CatFlow elementwise transformations that transform uniform noise to dequantized data samples $\boldsymbol{x} \in [0, 1]^D$ yields equivalent models to specifying autoregressive distributions with categorical conditionals for quantized data samples $\boldsymbol{q} \in \{0, 1, ..., 255\}^D$. This similarity is illustrated in Fig. 1.

**PixelCNN++** and related models [11, 10, 1] also use an autoregressive network to model quantized sensor data. However, instead of using the Categorical distribution for each sub-pixel, they rely on a multivariate discretized mixture of logistics (DMOL) distribution for each pixel. Due to space constraints, we refer the reader to App. B for more details on the multivariate DMOL distribution.

**A flow equivalent** can be developed by 1) rewriting the multivariate DMOL as an autoregressive distribution, 2) developing a univariate DMOL flow and 3) developing the multivariate DMOL flow as an autoregressive flow with univariate DMOL flows as the elementwise transformations. Due to space constraints, we refer the reader to App. C for more details.

With the resulting multivariate DMOL flow, PixelCNN++ can equivalently be viewed as a *nested autoregressive flow*. That is, the PixelCNN++ network, which is autoregressive over the spatial dimensions, outputs parameters for an autoregressive flow along the channel dimension where the elementwise transformations are given by univariate DMOL flows.

## 3 Experiments

PixelCNN, PixelCNN++ and related models are typically viewed as purely autoregressive models which have no noise/latent space associated with them. Our interpretation of these models as single-layer flows opens up for exploration of these existing models latent space. To illustrate this possibility, we trained PixelCNN [14] and PixelCNN++ [11] parameterizing CatFlows and multivariate DMOL flows on CIFAR-10. The models obtain 3.14 and 2.93 bits/dim, respectively, on the test set, closely matching the results of [14] and [11] at 3.14 and 2.92 bits/dim.

### 3.1 Latent Space Inspection

If our model were to exactly match the true data distribution, sampling $\boldsymbol{z} \sim \prod_{d=1}^{D} \mathrm{Unif}(z_d|0,1)$ and transforming $\boldsymbol{x} = f^{-1}(\boldsymbol{z})$ would yield samples from the true distribution. This fact is commonly used as a qualitative check of generative models, where generated samples are compared to true samples.

Another approach we could take, which in some sense takes an inverse view, is to take true samples $\boldsymbol{x}$ and encode them to $\boldsymbol{z} = f(\boldsymbol{x})$. If the model matches the true distribution, $\boldsymbol{z}$ should correspond exactly to uniform noise. Here we can both visually inspect latent images $\boldsymbol{z}$, which typically should have no discernible patterns left, and perform other quantitative or qualitative checks for uniformity.

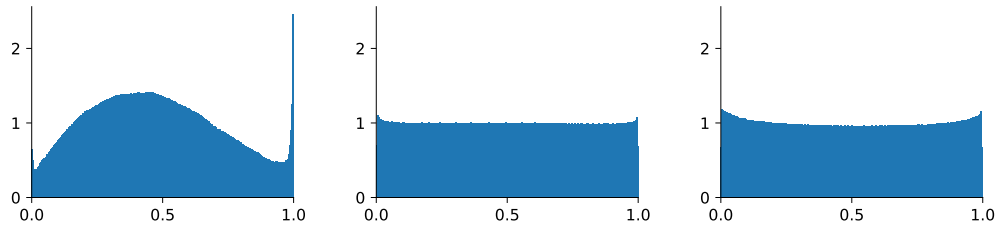
To test this, we encode the CIFAR-10 test set using our flow versions of PixelCNN and PixelCNN++. Some encoded images are shown in Figure 3. In addition to visual checks, we would like to test the latent $\boldsymbol{z}$ for uniformity. A necessary, but not sufficient, condition for $\boldsymbol{z}$ to be jointly uniform is that the marginals are uniform. A plot of the marginal distribution of all sub-pixels is shown in Figure 2. A Kolmogorov–Smirnov test for uniformity of the marginals gives a p-value of 0 (smaller than machine epsilon) for both models, i.e. a strong reject of the null hypothesis that the samples are uniform. This suggests that, perhaps not surprisingly, there is still room for improvement.

### 3.2 Latent Space Interpolation

Next, we demonstrate that we can interpolate in the latent space to obtain image samples between two existing images. To do this, we first transform two real images $\boldsymbol{x}^{(0)}$ and $\boldsymbol{x}^{(1)}$ to the latent space and obtain $\boldsymbol{z}^{(0)}$ and $\boldsymbol{z}^{(1)}$. Linearly interpolating in this space does not yield uniform samples. Empirically, we found this to often give blurry, single-color interpolations. Instead, we first further transformed the latent images $\boldsymbol{z}^{(0)} \rightarrow \boldsymbol{y}^{(0)}$ and $\boldsymbol{z}^{(1)} \rightarrow \boldsymbol{y}^{(1)}$ using the inverse Gaussian CDF for each dimension. As this is an invertible transformation, we can equivalently consider the base distribution as the isotropic Gaussian. Subsequently, we interpolated according to

$$\boldsymbol{y}^{(w)} = \frac{1}{\sqrt{w^2 + (1-w)^2}} \left[ w\boldsymbol{y}^{(0)} + (1-w)\boldsymbol{y}^{(1)} \right], \quad \text{for } 0 \leq w \leq 1. \tag{11}$$

This yields a path of equally probable samples under the base distribution, i.e. $\boldsymbol{y}^{(w)} \sim \mathrm{N}(0,1)$ for $\boldsymbol{y}^{(0)}, \boldsymbol{y}^{(1)} \sim \mathrm{N}(0,1)$. Finally, the intermediate noise variables $\boldsymbol{y}^{(w)}$ are transformed back to intermediate samples $\boldsymbol{x}^{(w)}$. Some examples of interpolations are shown in Figure 4.




(a) Data $\boldsymbol{x}$. (b) Encoded $\boldsymbol{z}$ using PixelCNN. (c) Encoded $\boldsymbol{z}$ using PixelCNN++.

Figure 2: Marginal distribution of all sub-pixels for the CIFAR-10 test images. We observe that the encoded marginals have a density lower than 1 *at* the edges and higher than 1 *near* the edges.

Figure 3: CIFAR-10 test set images and the latent encoding from flow versions of a PixelCNN (top row) and PixelCNN++ (bottom row). We see that certain patterns, such as edges, at times leave footprints in the latent images. While this effect is visible for both models, it is sometimes more pronounced for PixelCNN than PixelCNN++.

Figure 4: Latent space interpolations between pairs of CIFAR-10 test set images using flow versions of PixelCNN (odd rows) and PixelCNN++ (even rows). These models are known for capturing local correlations well, but typically struggle with long-range dependencies. This is reflected in several of the interpolated images, which tend to lack global coherence.

**Acknowledgments**

We thank the NVIDIA Corporation with the donation of Titan X GPUs.

## References


[1] Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. Pixelsnail: An improved autoregressive generative model. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 863–871, 2018.

[2] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509, 2019.

[3] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[4] Nal Kalchbrenner, Aäron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1771–1779, 2017.

[5] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 10236–10245, 2018.

[6] Diederik P. Kingma, Tim Salimans, Rafal Józefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improving variational autoencoders with inverse autoregressive flow. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4736–4744, 2016.

[7] Jacob Menick and Nal Kalchbrenner. Generating high fidelity images with subscale pixel networks and multidimensional upscaling. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[8] Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. Neural importance sampling. *CoRR*, abs/1808.03856, 2018.

[9] George Papamakarios, Iain Murray, and Theo Pavlakou. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 2338–2347, 2017.

[10] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 4052–4061, 2018.

[11] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[12] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016.

[13] Aäron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Koray Kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4790–4798, 2016.

[14] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1747–1756, 2016.

Table 1: Comparison of categorical distribution and distribution arising from transforming uniform noise using CatFlows. This can be seen as a continuous analogue of the categorical distribution.

| **Property** | **Categorical Distribution** | **Uniform+CatFlow Distribution** |
|---|---|---|
| Support | $x \in \{1, 2, ..., K\}$ | $x \in [0, 1]$ |
| Probability mass/density $p(x\|\boldsymbol{\pi})$ | $\prod_{k=1}^{K} \pi_k^{\mathbb{I}(x=k)}$ | $K \prod_{k=1}^{K} \pi_k^{\mathbb{I}(x \in [\frac{k-1}{K}, \frac{k}{K}))}$ |
| Gradients $\nabla_{\boldsymbol{\theta}} \log p(x\|\boldsymbol{\pi}(\boldsymbol{\theta}))$ | $\sum_{k=1}^{K} \mathbb{I}(x=k) \nabla_{\boldsymbol{\theta}} \log \pi_k(\boldsymbol{\theta})$ | $\sum_{k=1}^{K} \mathbb{I}(x \in [\frac{k-1}{K}, \frac{k}{K})) \nabla_{\boldsymbol{\theta}} \log \pi_k(\boldsymbol{\theta})$ |

# A On the Connections Between CatFlows and the Categorical Distribution

By transforming uniform noise using CatFlows, i.e.

$$z \sim \text{Unif}(z|0, 1) \tag{12}$$

$$x = \text{CatFlow}^{-1}(z|\boldsymbol{\pi}), \tag{13}$$

we obtain a density of the form

$$p(x|\boldsymbol{\pi}) = K \prod_{k=1}^{K} \pi_k^{\mathbb{I}(x \in [\frac{k-1}{K}, \frac{k}{K}))}. \tag{14}$$

As summarized in Table 1 this distribution can be considered a continuous analogue of the categorical distribution that only differ by a constant scaling factor (of $K$). Note that the scaling factor arises from the continuous distribution having to integrate to one over the $[0, 1]$ range (with an average bin height of 1) rather than sum to one over $\{1, 2, ..., K\}$ (with an average bin height of $1/K$).

# B The Discretized Mixture of Logistics Distribution

As described in Section 2.1 of [11], the discretized mixture of logistics (DMOL) distribution for $x \in \{1, 2, ..., K\}$ is given by

$$p(x) = \text{DMOL}(x|\boldsymbol{w}, \boldsymbol{\mu}, \boldsymbol{s}) = \begin{cases} \sum_{m=1}^{M} w_m \left[\sigma\left(\frac{\frac{1}{256} - \mu_m}{s_m}\right)\right], & \text{for } x = 1 \\ \sum_{m=1}^{M} w_m \left[\sigma\left(\frac{\frac{x}{256} - \mu_m}{s_m}\right) - \sigma\left(\frac{\frac{x-1}{256} - \mu_m}{s_m}\right)\right], & \text{for } x \in \{2, ..., 255\} \\ \sum_{m=1}^{M} w_m \left[1 - \sigma\left(\frac{\frac{255}{256} - \mu_m}{s_m}\right)\right], & \text{for } x = 256. \end{cases} \tag{15}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{s}$ are the location and scale parameters for each mixture, while $\boldsymbol{w}$ are the mixture weights with $\sum_m w_m = 1$ and $w_m \geq 0$ for $m = 1, 2, ..., M$.

We refer to this as the *univariate DMOL distribution* and note that it also can be written as

$$p(x) = \text{DMOL}(x|\boldsymbol{w}, \boldsymbol{\mu}, \boldsymbol{s}) \tag{16}$$

$$= \sum_{m=1}^{M} w_m \text{DL}(x|\mu_m, s_m), \tag{17}$$

where *DL* is the discretized logistic distribution. The *multivariate DMOL distribution*, as described in Section 2.2 of [11], can then be obtained as

$$\begin{aligned} p(\boldsymbol{x}) &= \text{MultiDMOL}(\boldsymbol{x}|\boldsymbol{w}, \boldsymbol{\mu}, \boldsymbol{s}, \boldsymbol{r}) \\ &= \sum_{m=1}^{M} w_m \text{DL}(x_3|\mu_{3,m}(x_1, x_2, \boldsymbol{r}_m), s_{3,m}) \\ &\qquad \text{DL}(x_2|\mu_{2,m}(x_1, \boldsymbol{r}_m), s_{2,m}) \text{DL}(x_1|\mu_{1,m}, s_{1,m}), \end{aligned} \tag{18}$$

where

$$\begin{aligned} \mu_{1,m} &= \mu_{1,m} \\ \mu_{2,m}(x_1, \boldsymbol{r}_m) &= \mu_{2,m} + r_{1,m} x_1 \\ \mu_{3,m}(x_1, x_2, \boldsymbol{r}_m) &= \mu_{3,m} + r_{2,m} x_1 + r_{3,m} x_2. \end{aligned} \tag{19}$$

## C The Discretized Mixture of Logistics Flow

We will develop a flow equivalent of the multivariate DMOL distribution in 3 parts:

1. Develop a univariate DMOL flow.
2. Rewrite the multivariate DMOL distribution as an autoregressive distribution.
3. Obtain the multivariate DMOL flow as an autoregressive flow with univariate DMOL flows as elementwise transformations.

### C.1 The Univariate DMOL Flow

A flow equivalent of the univariate DMOL distribution in Eq. 15 can be developed as a CatFlow with the bin probabilities $\boldsymbol{\pi}$ parameterized by a mixture of logistics function. That is, we can let the bin locations be parameterized by a function $h$, i.e.

$$z_k = \begin{cases} 0, & \text{for } k = 0 \\ h(x_k), & \text{for } k \in \{1, 2, ..., K-1\} \\ 1, & \text{for } k = K. \end{cases} \tag{20}$$

such that the bin probabilities are given by $\pi_k = h(x_k) - h(x_{k-1})$. By choosing the function $h$ as

$$h(x) = \sum_{m=1}^{M} w_m \left[ \sigma \left( \frac{x - \mu_m}{s_m} \right) \right], \tag{21}$$

we obtain a *univariate DMOL flow* which corresponds to a CatFlow with $\boldsymbol{\pi}$ given by

$$\pi_k = h(x_k) - h(x_{k-1}) = \begin{cases} \sum_{m=1}^{M} w_m \left[ \sigma \left( \frac{\frac{1}{K} - \mu_m}{s_m} \right) \right], & \text{for } k = 1 \\ \sum_{m=1}^{M} w_m \left[ \sigma \left( \frac{\frac{k}{K} - \mu_m}{s_m} \right) - \sigma \left( \frac{\frac{k-1}{K} - \mu_m}{s_m} \right) \right], & \text{for } k \in \{2, ..., K-1\} \\ \sum_{m=1}^{M} w_m \left[ 1 - \sigma \left( \frac{\frac{K-1}{K} - \mu_m}{s_m} \right) \right], & \text{for } k = K \end{cases} \tag{22}$$

where $\boldsymbol{w}$ correspond to the mixture weights and $\boldsymbol{\mu}$ and $\boldsymbol{s}$ are location and scale parameters. The forward and inverse transformations are as in Eq. 8 and 9, but with $\boldsymbol{\pi}$ given by Eq. 22. Thus, transforming uniform noise with this flow, one obtains a continuous analogue of the univariate DMOL distribution in Eq. 15.

### C.2 Rewriting the Multivariate DMOL Distribution

In order to develop a flow equivalent of the multivariate DMOL distribution, we first rewrite it as an autoregressive mixture distribution. In the three-dimensional case, we can do this as

$$p(\boldsymbol{x}) = \sum_{m=1}^{M} w_m p_m(x_3|x_2, x_1) p_m(x_2|x_1) p_m(x_1) \tag{23}$$

$$= \left[ \frac{\sum_{m=1}^{M} w_m p_m(x_3|x_2, x_1) p_m(x_2|x_1) p_m(x_1)}{\sum_{m=1}^{M} w_m p_m(x_2|x_1) p_m(x_1)} \right] \cdot \left[ \frac{\sum_{m=1}^{M} w_m p_m(x_2|x_1) p_m(x_1)}{\sum_{m=1}^{M} w_m p_m(x_1)} \right] \cdot \left[ \sum_{m=1}^{M} w_m p_m(x_1) \right] \tag{24}$$

$$= \left[ \sum_{m=1}^{M} w_{3,m} p_m(x_3|x_2, x_1) \right] \left[ \sum_{m=1}^{M} w_{2,m} p_m(x_2|x_1) \right] \left[ \sum_{m=1}^{M} w_{1,m} p_m(x_1) \right] \tag{25}$$

where

$$
\begin{aligned}
w_{1,m} &= w_m \\
w_{2,m} &= \frac{w_m p_m(x_1)}{\sum_{m'=1}^{M} w_{m'} p_{m'}(x_1)} \\
w_{3,m} &= \frac{w_m p_m(x_2|x_1) p_m(x_1)}{\sum_{m'=1}^{M} w_{m'} p_{m'}(x_2|x_1) p_{m'}(x_1)}.
\end{aligned}
\tag{26}
$$

Note that in this form, the multivariate DMOL is simply an autoregressive distribution with univariate DMOL conditionals, i.e.

$$
\begin{aligned}
\text{MultiDMOL}(\boldsymbol{x}|\boldsymbol{w}, \boldsymbol{\mu}, \boldsymbol{s}, \boldsymbol{r}) = \; & \text{DMOL}(x_3|\boldsymbol{w}(x_2, x_1), \boldsymbol{\mu}(x_2, x_1, \boldsymbol{r}), \boldsymbol{s}) \\
& \cdot \text{DMOL}(x_2|\boldsymbol{w}(x_1), \boldsymbol{\mu}(x_1, \boldsymbol{r}), \boldsymbol{s}) \\
& \cdot \text{DMOL}(x_1|\boldsymbol{w}, \boldsymbol{\mu}, \boldsymbol{s}).
\end{aligned}
\tag{27}
$$

### C.3 The Multivariate DMOL Flow

As can be seen in Eq. 27, the multivariate DMOL distribution can be seen as an autoregressive distribution where each conditional distribution is a univariate DMOL distribution. Using the univariate DMOL flows developed in App. C.1 and their relation to the univariate DMOL distribution, we obtain the multivariate DMOL flow simply as an autoregressive flow where each elementwise transformation is a univariate DMOL flow with mixture weights $\boldsymbol{w}$ given by Eq. 26 and the mixture means given by Eq. 19.

Transforming uniform noise using the multivariate DMOL flow,

$$
\boldsymbol{z} \sim \prod_{d=1}^{3} \text{Unif}(z_d|0, 1) \tag{28}
$$

$$
\boldsymbol{x} = \text{MultiDMOLFlow}^{-1}(\boldsymbol{z}|\boldsymbol{w}, \boldsymbol{\mu}, \boldsymbol{s}, \boldsymbol{r}) \tag{29}
$$

we obtain a continuous analogue of the multivariate DMOL distribution.