

Lecture 20: Normalizing Flows

CS480/680 Intro to Machine Learning

2023-3-23

Pascal Poupart
David R. Cheriton School of Computer Science



Normalizing Flows

- Neural networks to estimate probability density functions
 - Can estimate the likelihood of a data point
 - Generative models (i.e., generate data point from random embedding)
 - Invertible models (i.e., generate embedding corresponding to a data point)

Preview: GLOW (Generative fLOW)

Kingma, Dhariwal et al. (2018)



Figure 4: Random samples from the model, with temperature 0.7



Figure 5: Linear interpolation in latent space between real images

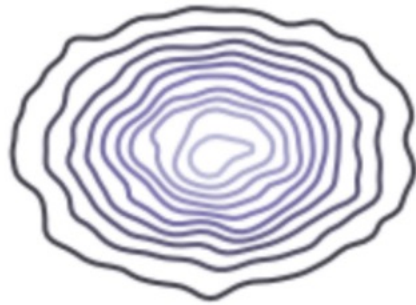
Recap

Variational Autoencoder

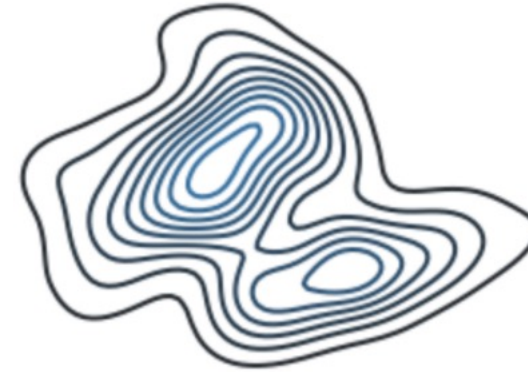
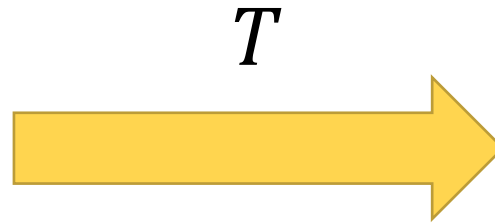
Generative Adversarial Network

Explicit density function

From Priyank Jaini



Source distribution
 $\mathbf{z} \sim p(\mathbf{z})$



Target distribution
 $\mathbf{x} \sim q(\mathbf{x})$

- Goal: learn a deterministic function T that transforms p into q

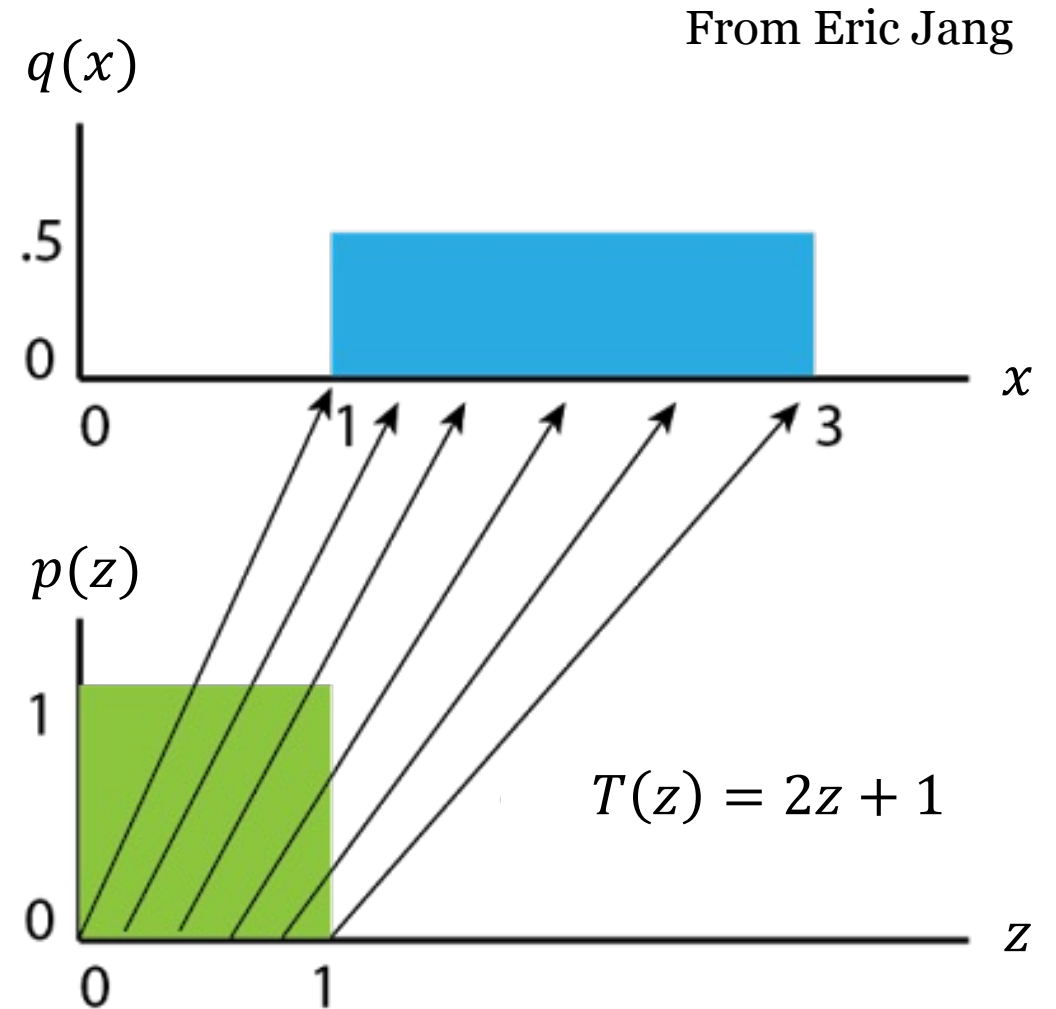
Conservation of Mass

- When we transform random variable Z into random variable $X = T(Z)$, the probability mass must be conserved

$$q(x)dx = p(z)dz$$

- Hence

$$\begin{aligned} q(x) &= p(z) \left| \frac{dz}{dx} \right| \\ &= p(z) \left| \frac{dx}{dz} \right|^{-1} \\ &= p(z) \left| \frac{dT(z)}{dz} \right|^{-1} \end{aligned}$$



Change of Variable

- Multivariable distributions $p(\mathbf{X})$ and $q(\mathbf{Z})$
 - \mathbf{X}, \mathbf{Z} : vectors of random variables
 - $T: \mathbf{Z} \rightarrow \mathbf{X}$ where $T_1(\mathbf{z}) = x_1, T_2(\mathbf{z}) = x_2, T_3(\mathbf{z}) = x_3$, etc.

- Change of variable formula:

$$p(\mathbf{x}) = p(\mathbf{z}) |\det(\nabla_{\mathbf{z}} T(\mathbf{z}))|^{-1}$$

where T is bijective and increasing

Learning

- Data
 - Let $\mathbf{Z} \sim N(\mathbf{0}, \mathbf{I})$ be random vectors
 - Let \mathbf{X} be data (i.e., images in a dataset)
- Find $T: Z \rightarrow X$ by maximum likelihood

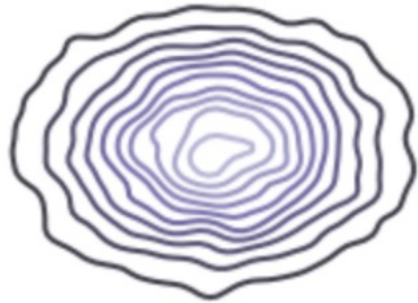
$$\begin{aligned}\operatorname{argmax}_T \prod_n P(\mathbf{z}^{(n)}) &= \operatorname{argmax}_T \sum_n \log P(\mathbf{z}^{(n)}) \\ &= \operatorname{argmax}_T \sum_n \log \left[P(\mathbf{x}^{(n)}) |\det(\nabla_{\mathbf{x}} T^{-1}(\mathbf{x}))|^{-1} \right] \\ &= \operatorname{argmax}_T \sum_n \log P(\mathbf{x}^{(n)}) - \log |\det(\nabla_{\mathbf{x}} T^{-1}(\mathbf{x}))|\end{aligned}$$

- NB: need T^{-1} and efficient way to compute $\det(\nabla_{\mathbf{x}} T^{-1}(\mathbf{x}))$

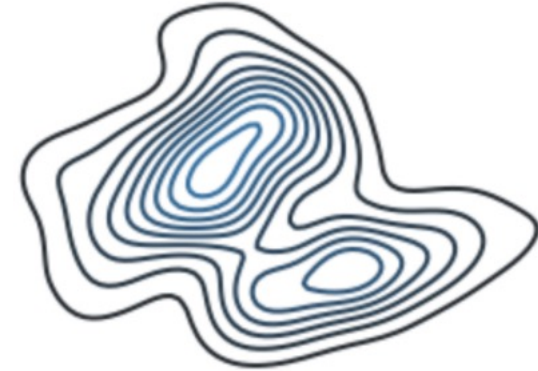
Triangular Maps

- Let T be such that $T_i(\mathbf{z}_{\leq i}) = x_i$
 - $x_1 = T_1(z_1)$
 - $x_2 = T_2(z_1, z_2)$
 - ...
 - $x_m = T_m(z_1, z_2, \dots, z_m)$
- $\det(\nabla T^{-1})$ is easy to compute
 - T lower triangular $\rightarrow T^{-1}$ lower triangular
 - $\det(\text{lower triangular } M) = \prod_i M_{ii}$
- **Theorem:** There always exists a **unique** increasing triangular map that transforms a source density into a target density

Framework



$$z \sim p(z)$$



$$x \sim q(x)$$

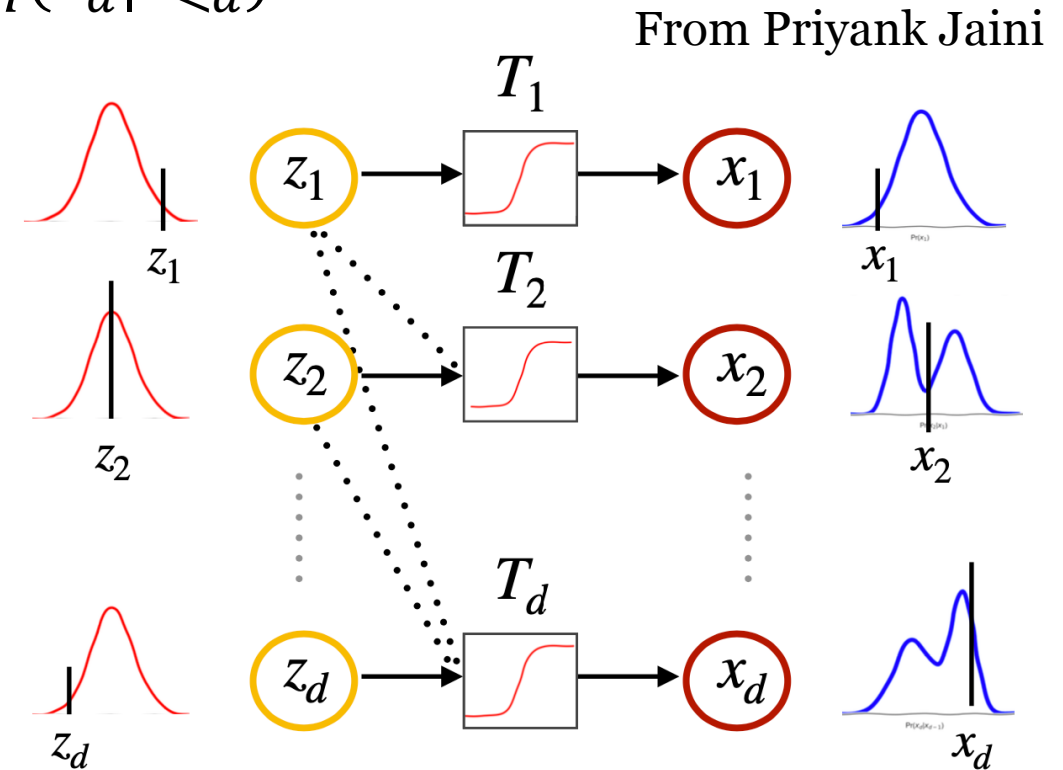
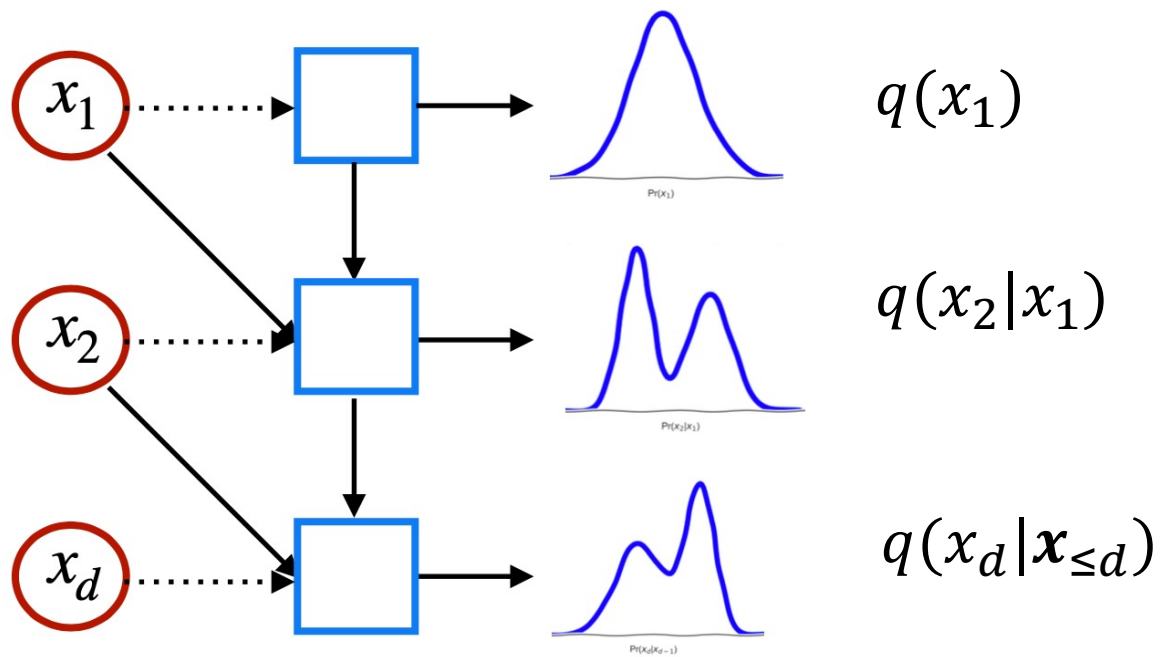
Find T by maximizing log likelihood

Set $P(\mathbf{z}) = N(0, I)$

$$\operatorname{argmax}_T \sum_n \log q(\mathbf{x}^{(n)}) = \operatorname{argmax}_T \sum_n \log p\left(T^{-1}(\mathbf{x}^{(n)})\right) - \sum_i \log \frac{\partial T_i^{-1}(\mathbf{x}^{(n)})}{\partial x_i}$$

Autoregressive Models

- Recall the chain rule: $q(\mathbf{x}) = q(x_1)q(x_2|x_1) \dots q(x_d|\mathbf{x}_{<d})$

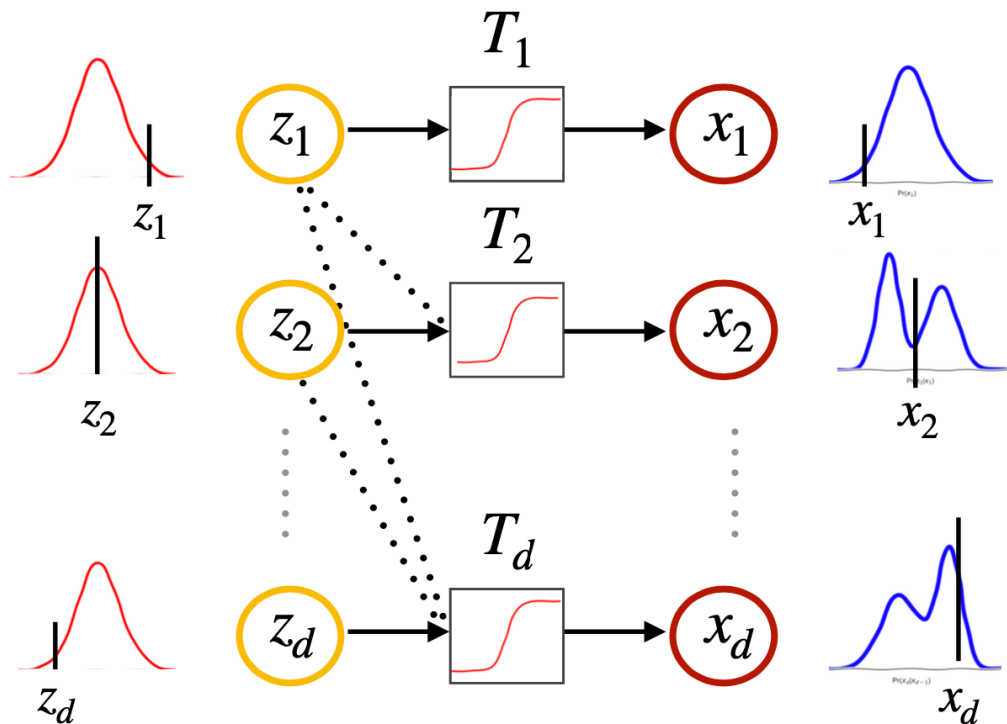


- Each mapping $x_i = T_i(\mathbf{z}_{\leq i})$ implicitly determines a corresponding $q(x_i|\mathbf{x}_{<i})$

Autoregressive Models with Gaussian Conditionals

- Recall the reparameterization trick

From Priyank Jaini



$$x_1 = T_1(z_1) = \sigma_1 z_1 + \mu_1$$

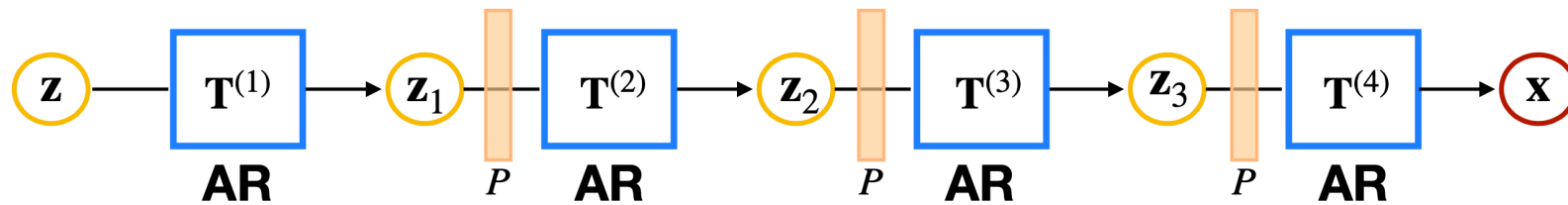
$$x_2 = T_2(z_1, z_2) = \sigma_2(z_1)z_2 + \mu_2(z_1)$$

$$x_d = T_d(z_1, \dots, z_d) = \sigma_d(\mathbf{z}_{<d})z_d + \mu_d(\mathbf{z}_{<d})$$

Masked Autoregressive Flows (MAFs)

Papamakarios et al., 2017

- Deep autoregressive flows with Gaussian conditionals



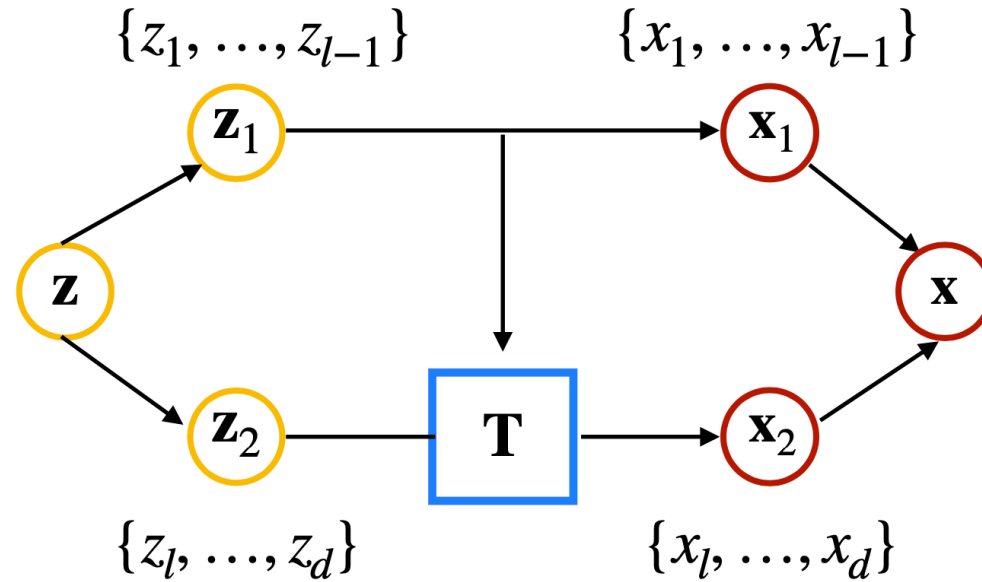
From Priyank Jaini

$$q(\mathbf{x}) = p(\mathbf{z}) |\det(\nabla T^{(1)})|^{-1} |\det(\nabla T^{(2)})|^{-1} |\det(\nabla T^{(3)})|^{-1} |\det(\nabla T^{(4)})|^{-1}$$

$$x_i = T_i(\mathbf{z}_{\leq i}) = \exp(\alpha_i(\mathbf{z}_{< i})) z_i + \mu_i(\mathbf{z}_{< i})$$

Real NVP

Dinh et al., 2017

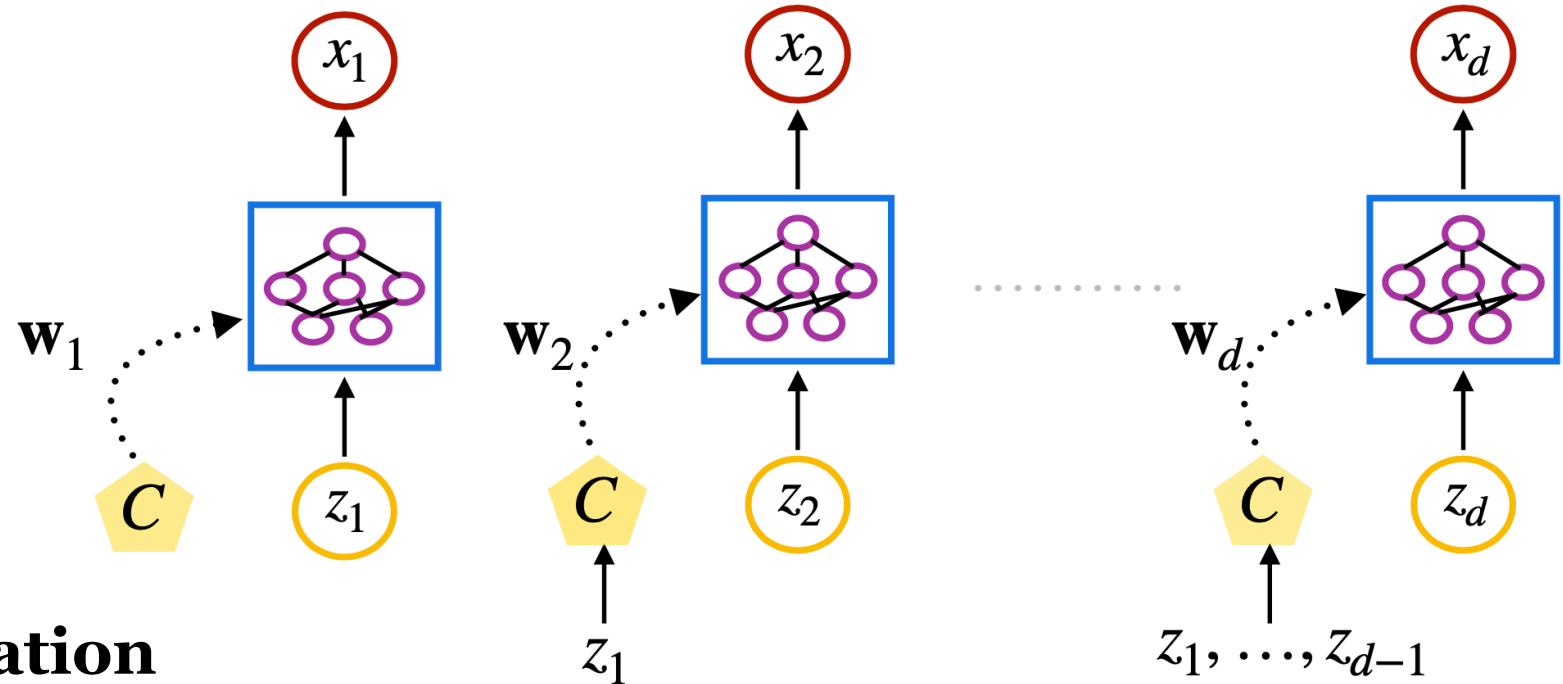


From Priyank Jaini

$$x_i = T_i(z_i, \mathbf{z}_{<i}) = \exp(\alpha_i(\mathbf{z}_{<i})\delta(i \geq l)) z_i + \mu_i(\mathbf{z}_{<i})\delta(i \geq l)$$

Neural Autoregressive Flows (NAFs)

- Strictly positive weights
- Strictly monotonic activation functions
- Hence: increasing map
- **Universal approximation**

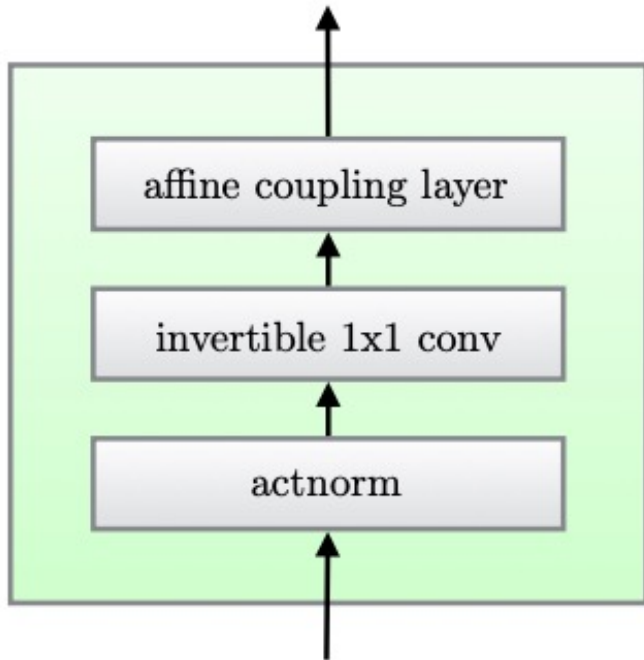


From Priyank Jaini

$$x_i = T_i(z_i, \mathbf{z}_{<i}) = DNN(z_i; w_i(\mathbf{z}_{<i}))$$

GLOW (Generative fLOW)

Kingma, Dhariwal et al. (2018)



Description	Function	Reverse Function
Actnorm. See Section 3.1.	$\forall i, j : \mathbf{y}_{i,j} = \mathbf{s} \odot \mathbf{x}_{i,j} + \mathbf{b}$	$\forall i, j : \mathbf{x}_{i,j} = (\mathbf{y}_{i,j} - \mathbf{b})/\mathbf{s}$
Invertible 1×1 convolution. $\mathbf{W} : [c \times c]$. See Section 3.2.	$\forall i, j : \mathbf{y}_{i,j} = \mathbf{W}\mathbf{x}_{i,j}$	$\forall i, j : \mathbf{x}_{i,j} = \mathbf{W}^{-1}\mathbf{y}_{i,j}$
Affine coupling layer. See Section 3.3 and (Dinh et al., 2014)	$\mathbf{x}_a, \mathbf{x}_b = \text{split}(\mathbf{x})$ $(\log \mathbf{s}, \mathbf{t}) = \text{NN}(\mathbf{x}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{y}_a = \mathbf{s} \odot \mathbf{x}_a + \mathbf{t}$ $\mathbf{y}_b = \mathbf{x}_b$ $\mathbf{y} = \text{concat}(\mathbf{y}_a, \mathbf{y}_b)$	$\mathbf{y}_a, \mathbf{y}_b = \text{split}(\mathbf{y})$ $(\log \mathbf{s}, \mathbf{t}) = \text{NN}(\mathbf{y}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{x}_a = (\mathbf{y}_a - \mathbf{t})/\mathbf{s}$ $\mathbf{x}_b = \mathbf{y}_b$ $\mathbf{x} = \text{concat}(\mathbf{x}_a, \mathbf{x}_b)$

GLOW (Generative fLOW)

Kingma, Dhariwal et al. (2018)



Figure 4: Random samples from the model, with temperature 0.7



Figure 5: Linear interpolation in latent space between real images