

# Deep Generative Models

## Lecture 5

Roman Isachenko

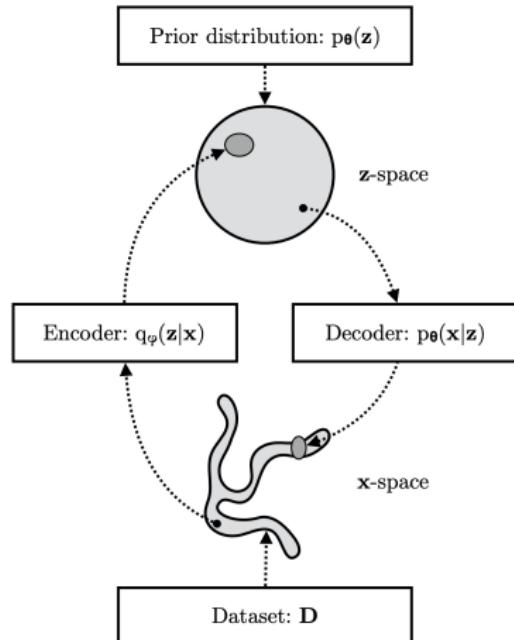
Moscow Institute of Physics and Technology

Autumn, 2021

# Recap of previous lecture

## Variational autoencoder (VAE)

- ▶ VAE learns stochastic mapping between  $\mathbf{x}$ -space, from  $\pi(\mathbf{x})$ , and a latent  $\mathbf{z}$ -space, with simple distribution.
- ▶ The generative model learns distribution  $p(\mathbf{x}, \mathbf{z}|\theta) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z}, \theta)$ , with a prior distribution  $p(\mathbf{z})$ , and a stochastic decoder  $p(\mathbf{x}|\mathbf{z}, \theta)$ .
- ▶ The stochastic encoder  $q(\mathbf{z}|\mathbf{x}, \phi)$  (inference model), approximates the true but intractable posterior  $p(\mathbf{z}|\mathbf{x}, \theta)$ .



# Recap of previous lecture

Likelihood-based models so far...

Autoregressive models

$$p(\mathbf{x}|\boldsymbol{\theta}) = \prod_{i=1}^m p(x_i|\mathbf{x}_{1:i-1}, \boldsymbol{\theta})$$

- ▶ tractable likelihood,
- ▶ no inferred latent factors.

Latent variable models

$$p(\mathbf{x}|\boldsymbol{\theta}) = \int p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}) d\mathbf{z}$$

- ▶ latent feature representation,
- ▶ intractable likelihood.

How to build a model with latent variables and tractable likelihood?

## Recap of previous lecture

### Change of variable theorem

Let  $\mathbf{x}$  be a random variable with density function  $p(\mathbf{x})$  and  $f : \mathbb{R}^m \rightarrow \mathbb{R}^m$  is a differentiable, invertible function (diffeomorphism). If  $\mathbf{z} = f(\mathbf{x})$ ,  $\mathbf{x} = f^{-1}(\mathbf{z}) = g(\mathbf{z})$ , then

$$p(\mathbf{x}) = p(\mathbf{z}) \left| \det \left( \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right) \right| = p(f(\mathbf{x})) \left| \det \left( \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right) \right|$$
$$p(\mathbf{z}) = p(\mathbf{x}) \left| \det \left( \frac{\partial \mathbf{x}}{\partial \mathbf{z}} \right) \right| = p(g(\mathbf{z})) \left| \det \left( \frac{\partial g(\mathbf{z})}{\partial \mathbf{z}} \right) \right|.$$

### Inverse function theorem

If function  $f$  is invertible and Jacobian is continuous and non-singular, then

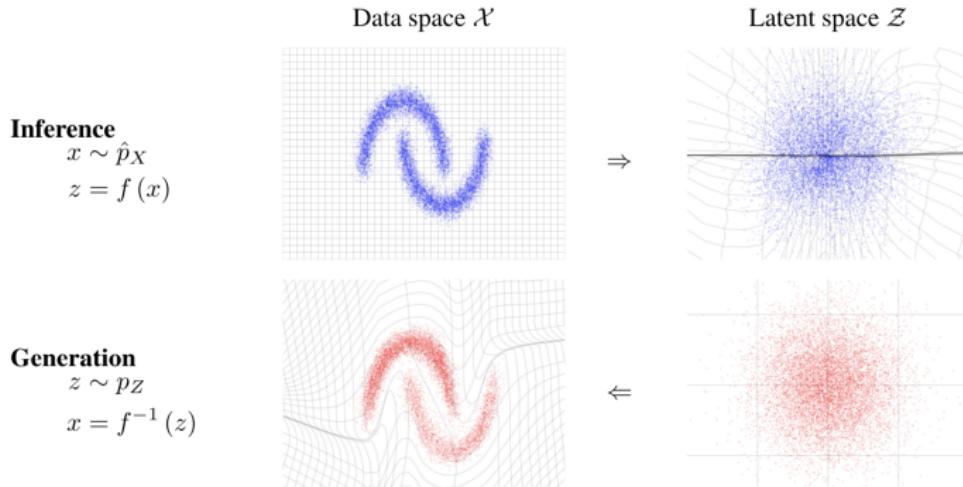
$$\left| \det \left( \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right) \right| = \left| \det \left( \frac{\partial g^{-1}(\mathbf{x})}{\partial \mathbf{x}} \right) \right| = \left| \det \left( \frac{\partial g(\mathbf{z})}{\partial \mathbf{z}} \right) \right|^{-1}$$

# Recap of previous lecture

## MLE problem for fitting flows

$$p(\mathbf{x}|\boldsymbol{\theta}) = p(\mathbf{z}) \left| \det \left( \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right) \right| = p(f(\mathbf{x}, \boldsymbol{\theta})) \left| \det \left( \frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \right) \right|$$

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(f(\mathbf{x}, \boldsymbol{\theta})) + \log \left| \det \left( \frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \right) \right| \rightarrow \max_{\boldsymbol{\theta}}$$



## Recap of previous lecture

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(f(\mathbf{x}, \boldsymbol{\theta})) + \log \left| \det \left( \frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \right) \right|$$

### Definition

Normalizing flow is a *differentiable, invertible* mapping from data  $\mathbf{x}$  to the noise  $\mathbf{z}$ .

- ▶ **Normalizing** means that the inverse flow takes samples from  $p(\mathbf{x})$  and normalizes them into samples from density  $p(\mathbf{z})$ .
- ▶ **Flow** refers to the trajectory followed by samples from  $p(\mathbf{z})$  as they are transformed by the sequence of transformations

$$\mathbf{z} = f_K \circ \cdots \circ f_1(\mathbf{x}); \quad \mathbf{x} = f_1^{-1} \circ \cdots \circ f_K^{-1}(\mathbf{z}) = g_1 \circ \cdots \circ g_K(\mathbf{z})$$

$$\begin{aligned} p(\mathbf{x}) &= p(f_K \circ \cdots \circ f_1(\mathbf{x})) \left| \det \left( \frac{\partial f_K \circ \cdots \circ f_1(\mathbf{x})}{\partial \mathbf{x}} \right) \right| = \\ &= p(f_K \circ \cdots \circ f_1(\mathbf{x})) \prod_{k=1}^K \left| \det \left( \frac{\partial \mathbf{f}_k}{\partial \mathbf{f}_{k-1}} \right) \right|. \end{aligned}$$

## Recap of previous lecture

### Forward KL for flow model

$$\log p(\mathbf{x}|\theta) = \log p(f(\mathbf{x}, \theta)) + \log \left| \det \left( \frac{\partial f(\mathbf{x}, \theta)}{\partial \mathbf{x}} \right) \right|$$

### Reverse KL for flow model

$$KL(p||\pi) = \mathbb{E}_{p(\mathbf{z})} \left[ \log p(\mathbf{z}) - \log \left| \det \left( \frac{\partial g(\mathbf{z}, \theta)}{\partial \mathbf{z}} \right) \right| - \log \pi(g(\mathbf{z}, \theta)) \right]$$

### Flow KL duality

$$\arg \min_{\theta} KL(\pi(\mathbf{x})||p(\mathbf{x}|\theta)) = \arg \min_{\theta} KL(p(\mathbf{z}|\theta)||p(\mathbf{z})).$$

- ▶  $p(\mathbf{z})$  is a base distribution;  $\pi(\mathbf{x})$  is a data distribution;
- ▶  $\mathbf{z} \sim p(\mathbf{z})$ ,  $\mathbf{x} = g(\mathbf{z}, \theta)$ ,  $\mathbf{x} \sim p(\mathbf{x}|\theta)$ ;
- ▶  $\mathbf{x} \sim \pi(\mathbf{x})$ ,  $\mathbf{z} = f(\mathbf{x}, \theta)$ ,  $\mathbf{z} \sim p(\mathbf{z}|\theta)$ ;

## Jacobian structure

### Flow log-likelihood

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(f(\mathbf{x}, \boldsymbol{\theta})) + \log \left| \det \left( \frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \right) \right|$$

The main challenge is a determinant of the Jacobian.

What is a det of Jacobian in the following cases?

1. Consider a linear layer  $\mathbf{z} = \mathbf{W}\mathbf{x}$ .
2. Let  $\mathbf{z}$  be a permutation of  $\mathbf{x}$ .
3. Let  $z_i$  depend only on  $\mathbf{x}_i$ .

$$\log \left| \det \left( \frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \right) \right| = \log \left| \prod_{i=1}^m f'_i(x_i, \boldsymbol{\theta}) \right| = \sum_{i=1}^m \log |f'_i(x_i, \boldsymbol{\theta})|.$$

4. Let  $z_i$  depend only on  $\mathbf{x}_{1:i}$  (autoregressive dependency).

# Residual Flows

## Matrix determinant lemma

$$\det(\mathbf{I}_m + \mathbf{V}\mathbf{W}^T) = \det(\mathbf{I}_d + \mathbf{W}^T\mathbf{V}), \quad \text{where } \mathbf{V}, \mathbf{W} \in \mathbb{R}^{m \times d}.$$

## Planar flow

$$\mathbf{x} = g(\mathbf{z}, \boldsymbol{\theta}) = \mathbf{z} + \mathbf{u} \sigma(\mathbf{w}^T \mathbf{z} + b).$$

Here  $\boldsymbol{\theta} = \{\mathbf{u}, \mathbf{w}, b\}$ ,  $\sigma(\cdot)$  is a smooth element-wise non-linearity.

$$\left| \det \left( \frac{\partial g(\mathbf{z}, \boldsymbol{\theta})}{\partial \mathbf{z}} \right) \right| = \left| \det (\mathbf{I} + \sigma'(\mathbf{w}^T \mathbf{z} + b) \mathbf{w} \mathbf{u}^T) \right| = \left| 1 + \sigma'(\mathbf{w}^T \mathbf{z} + b) \mathbf{w}^T \mathbf{u} \right|$$

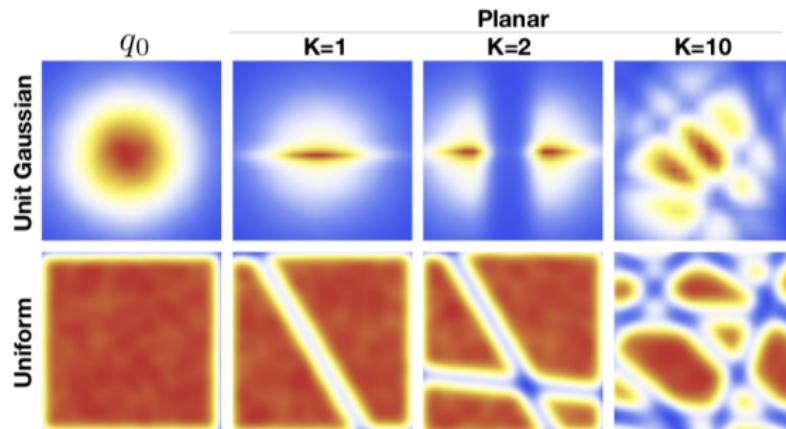
The transformation is invertible, for example, if

$$\sigma = \tanh; \quad \sigma'(\mathbf{w}^T \mathbf{z} + b) \mathbf{u}^T \mathbf{w} \geq -1.$$

# Residual Flows

## Expressiveness of planar flows

$$\mathbf{z}_K = g_1 \circ \cdots \circ g_K(\mathbf{z}); \quad g_k = g(\mathbf{z}_k, \theta_k) = \mathbf{z}_k + \mathbf{u}_k \sigma(\mathbf{w}_k^T \mathbf{z}_k + b_k).$$



Sylvester flow: planar flow extension

$$g(\mathbf{z}, \theta) = \mathbf{z} + \mathbf{V} \sigma(\mathbf{W}^T \mathbf{z} + \mathbf{b}).$$

---

Rezende D. J., Mohamed S. Variational Inference with Normalizing Flows, 2015  
Berg R. et al. Sylvester normalizing flows for variational inference, 2018

## Autoregressive flows

$$x_i = \tau(z_i, c(\mathbf{z}_{1:i-1})) \Leftrightarrow z_i = \tau^{-1}(x_i, c(\mathbf{z}_{1:i-1}))$$

- ▶  $\tau(\cdot, \cdot)$  – coupling law (invertible by first argument).
- ▶  $c(\cdot)$  – coupling function (do not need to be invertible, could be neural network).

### Coupling law $\tau(\cdot, \cdot)$

- ▶  $\tau(x, c) = x + c$  – additive;
- ▶  $\tau(x, c) = x \odot \exp c_1 + c_2$  – affine.

What is the Jacobian for the additive/affine coupling law?

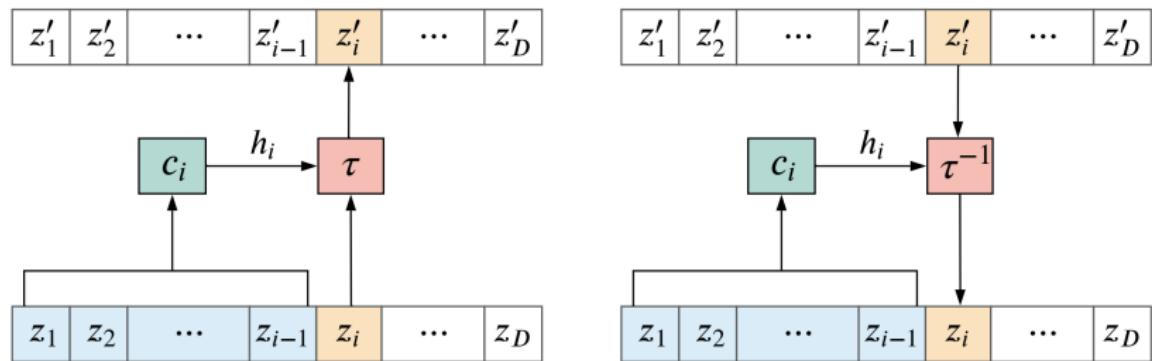
### Jacobian

$$\det \left( \frac{\partial \mathbf{x}}{\partial \mathbf{z}} \right) = \prod_{i=1}^m \frac{\partial x_i}{\partial z_i} = \prod_{i=1}^m \frac{\partial \tau(z_i, c(\mathbf{z}_{1:i-1}))}{\partial z_i}$$

# Autoregressive flows

## Forward and inverse transforms

$$x_i = \tau(z_i, c(\mathbf{z}_{1:i-1})) \Leftrightarrow z_i = \tau^{-1}(x_i, c(\mathbf{z}_{1:i-1}))$$



- ▶ Forward transform is **not sequential**.
- ▶ Inverse transform is **sequential**.

## Gaussian autoregressive model

Consider an autoregressive model

$$p(\mathbf{x}|\boldsymbol{\theta}) = \prod_{i=1}^m p(x_i|\mathbf{x}_{1:i-1}, \boldsymbol{\theta}), \quad p(x_i|\mathbf{x}_{1:i-1}, \boldsymbol{\theta}) = \mathcal{N}(\mu_i(\mathbf{x}_{1:i-1}), \sigma_i^2(\mathbf{x}_{1:i-1})).$$

Sampling: reparametrization trick

$$x_i = \sigma_i(\mathbf{x}_{1:i-1}) \cdot z_i + \mu_i(\mathbf{x}_{1:i-1}), \quad z_i \sim \mathcal{N}(0, 1).$$

Inverse transform

$$z_i = (x_i - \mu_i(\mathbf{x}_{1:i-1})) \cdot \frac{1}{\sigma_i(\mathbf{x}_{1:i-1})}.$$

We have got an autoregressive flow with base distribution  $\mathbf{z} = \mathcal{N}(0, 1)$  and affine coupling law  $\tau(x, \mu, \sigma) = x \odot \mu + \sigma$ .

## Gaussian autoregressive flow

$$\mathbf{x} = g(\mathbf{z}, \theta) \quad \Rightarrow \quad x_i = \sigma_i(\mathbf{x}_{1:i-1}) \cdot z_i + \mu_i(\mathbf{x}_{1:i-1}).$$

$$\mathbf{z} = f(\mathbf{x}, \theta) \quad \Rightarrow \quad z_i = (x_i - \mu_i(\mathbf{x}_{1:i-1})) \cdot \frac{1}{\sigma_i(\mathbf{x}_{1:i-1})}.$$

Generation function  $g(\mathbf{z}, \theta)$  is **sequential**. Inference function  $f(\mathbf{x}, \theta)$  is **not sequential**.

Forward KL for flow model

$$\log p(\mathbf{x}|\theta) = \log p(f(\mathbf{x}, \theta)) + \log \left| \det \left( \frac{\partial f(\mathbf{x}, \theta)}{\partial \mathbf{x}} \right) \right|$$

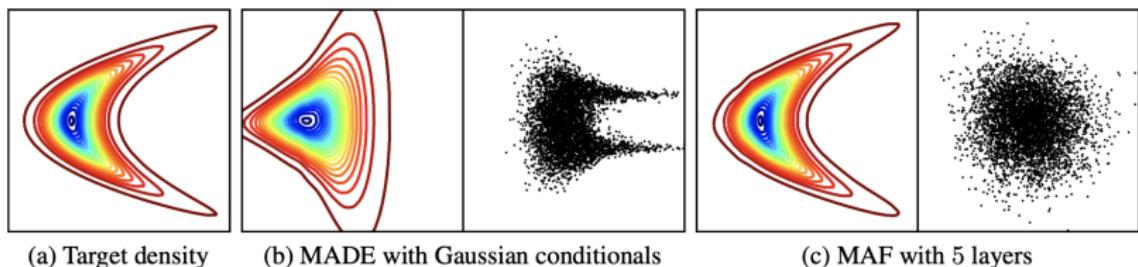
- ▶ We need to be able to compute  $f(\mathbf{x}, \theta)$  and its Jacobian.
- ▶ We need to be able to compute the density  $p(\mathbf{z})$ .
- ▶ We don't need to think about computing the function  $g(\mathbf{z}, \theta) = f^{-1}(\mathbf{z}, \theta)$  until we want to sample from the flow.

# Masked autoregressive flow (MAF)

## Gaussian autoregressive model

$$p(\mathbf{x}|\theta) = \prod_{i=1}^m p(x_i|\mathbf{x}_{1:i-1}, \theta) = \prod_{i=1}^m \mathcal{N}(x_i|\mu_i(\mathbf{x}_{1:i-1}), \sigma_i^2(\mathbf{x}_{1:i-1})).$$

We could use MADE (masked autoencoder) as a conditional model. The sampling order might be crucial.

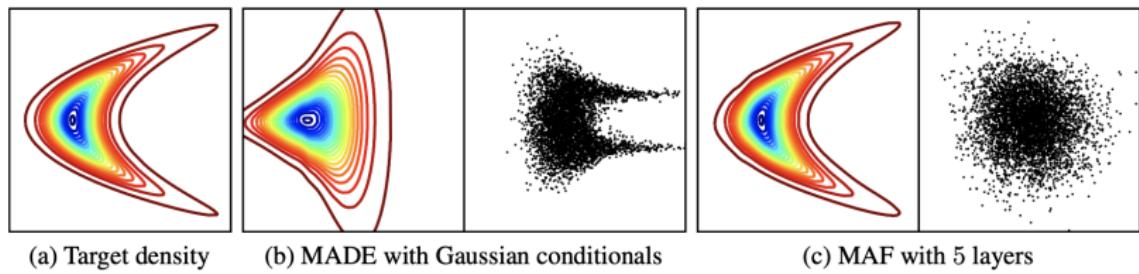


Samples from the base distribution could be an indicator of how good the flow was fitted.

# Masked autoregressive flow (MAF)

## Gaussian autoregressive model

$$p(\mathbf{x}|\theta) = \prod_{i=1}^m p(x_i|\mathbf{x}_{1:i-1}, \theta) = \prod_{i=1}^m \mathcal{N}(x_i|\mu_i(\mathbf{x}_{1:i-1}), \sigma_i^2(\mathbf{x}_{1:i-1})).$$



MAF is just a stacked MADE model with different ordering.

- ▶ Parallel density estimation.
- ▶ Sequential sampling.

## Inverse autoregressive flow (IAF)

Let's use the following reparametrization:  $\tilde{\sigma} = \frac{1}{\sigma}$ ;  $\tilde{\mu} = -\frac{\mu}{\sigma}$ .

### Gaussian autoregressive flow

$$x_i = \sigma_i(\mathbf{x}_{1:i-1}) \cdot z_i + \mu_i(\mathbf{x}_{1:i-1}) = (z_i - \tilde{\mu}_i(\mathbf{x}_{1:i-1})) \cdot \frac{1}{\tilde{\sigma}_i(\mathbf{x}_{1:i-1})}$$
$$z_i = (x_i - \mu_i(\mathbf{x}_{1:i-1})) \cdot \frac{1}{\sigma_i(\mathbf{x}_{1:i-1})} = \tilde{\sigma}_i(\mathbf{x}_{1:i-1}) \cdot x_i + \tilde{\mu}_i(\mathbf{x}_{1:i-1}).$$

Let's just swap  $\mathbf{z}$  and  $\mathbf{x}$ .

### Inverse autoregressive flow

$$\mathbf{x} = g(\mathbf{z}, \theta) \quad \Rightarrow \quad x_i = \tilde{\sigma}_i(\mathbf{z}_{1:i-1}) \cdot z_i + \tilde{\mu}_i(\mathbf{z}_{1:i-1})$$
$$\mathbf{z} = f(\mathbf{x}, \theta) \quad \Rightarrow \quad z_i = (x_i - \tilde{\mu}_i(\mathbf{z}_{1:i-1})) \cdot \frac{1}{\tilde{\sigma}_i(\mathbf{z}_{1:i-1})}.$$

# Inverse autoregressive flow (IAF)

Gaussian autoregressive flow:  $f(\mathbf{x}, \theta)$

$$x_i = \sigma_i(\mathbf{x}_{1:i-1}) \cdot z_i + \mu_i(\mathbf{x}_{1:i-1}).$$

Inverse transform:  $g(\mathbf{z}, \theta)$

$$z_i = (x_i - \mu_i(\mathbf{x}_{1:i-1})) \cdot \frac{1}{\sigma_i(\mathbf{x}_{1:i-1})};$$

$$z_i = \tilde{\sigma}_i(\mathbf{x}_{1:i-1}) \cdot x_i + \tilde{\mu}_i(\mathbf{x}_{1:i-1}).$$

Inverse autoregressive flow:  $f(\mathbf{x}, \theta)$

$$x_i = \tilde{\sigma}_i(\mathbf{z}_{1:i-1}) \cdot z_i + \tilde{\mu}_i(\mathbf{z}_{1:i-1}).$$

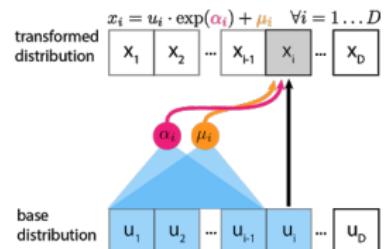
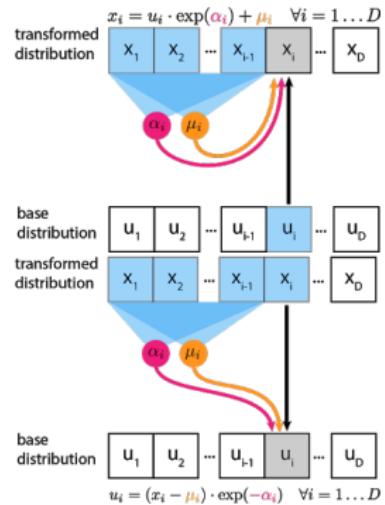


image credit: <https://blog.evjang.com/2018/01/nf2.html>

## Autoregressive flows

### Forward and inverse transforms in MAF

$$\mathbf{x} = g(\mathbf{z}, \theta) \quad \Rightarrow \quad x_i = \sigma_i(\mathbf{x}_{1:i-1}) \cdot z_i + \mu_i(\mathbf{x}_{1:i-1}).$$

$$\mathbf{z} = f(\mathbf{x}, \theta) \quad \Rightarrow \quad z_i = (x_i - \mu_i(\mathbf{x}_{1:i-1})) \cdot \frac{1}{\sigma_i(\mathbf{x}_{1:i-1})}.$$

- ▶ Sampling is sequential.
- ▶ Density estimation is parallel.

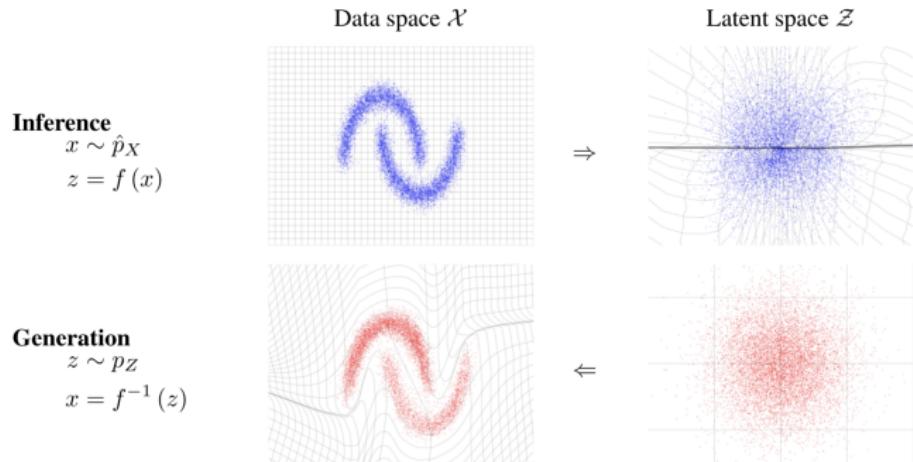
### Forward and inverse transforms in IAF

$$\mathbf{x} = g(\mathbf{z}, \theta) \quad \Rightarrow \quad x_i = \tilde{\sigma}_i(\mathbf{z}_{1:i-1}) \cdot z_i + \tilde{\mu}_i(\mathbf{z}_{1:i-1}).$$

$$\mathbf{z} = f(\mathbf{x}, \theta) \quad \Rightarrow \quad z_i = (x_i - \tilde{\mu}_i(\mathbf{z}_{1:i-1})) \cdot \frac{1}{\tilde{\sigma}_i(\mathbf{z}_{1:i-1})}.$$

- ▶ Sampling is parallel.
- ▶ Density estimation is sequential.

# Autoregressive flows



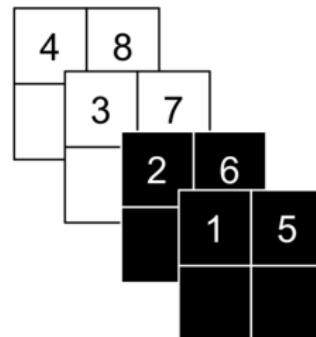
- ▶ MAF performs parallel inference that is useful for density estimation tasks (forward KL or MLE).
- ▶ IAF performs parallel generation that is useful for variational inference (reverse KL).

# RealNVP

## Coupling layer

$$\begin{cases} \mathbf{z}_{1:d} = \mathbf{x}_{1:d}; \\ \mathbf{z}_{d:m} = \tau(\mathbf{x}_{d:m}, c(\mathbf{x}_{1:d})); \end{cases} \Leftrightarrow \begin{cases} \mathbf{x}_{1:d} = \mathbf{z}_{1:d}; \\ \mathbf{x}_{d:m} = \tau^{-1}(\mathbf{z}_{d:m}, c(\mathbf{z}_{1:d})). \end{cases}$$

## Image partitioning



Masked convolutions are used to define ordering.

# RealNVP

## Affine coupling law

$$\begin{cases} \mathbf{z}_{1:d} = \mathbf{x}_{1:d}; \\ \mathbf{z}_{d:m} = \mathbf{x}_{d:m} \odot \exp(c_1(\mathbf{x}_{1:d}, \theta)) + c_2(\mathbf{x}_{1:d}, \theta). \end{cases}$$

$$\begin{cases} \mathbf{x}_{1:d} = \mathbf{z}_{1:d}; \\ \mathbf{x}_{d:m} = (\mathbf{z}_{d:m} - c_2(\mathbf{z}_{1:d}, \theta)) \odot \exp(-c_1(\mathbf{z}_{1:d}, \theta)). \end{cases}$$

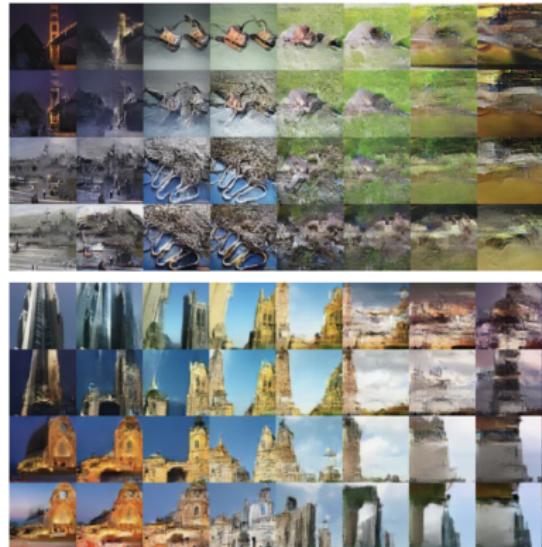
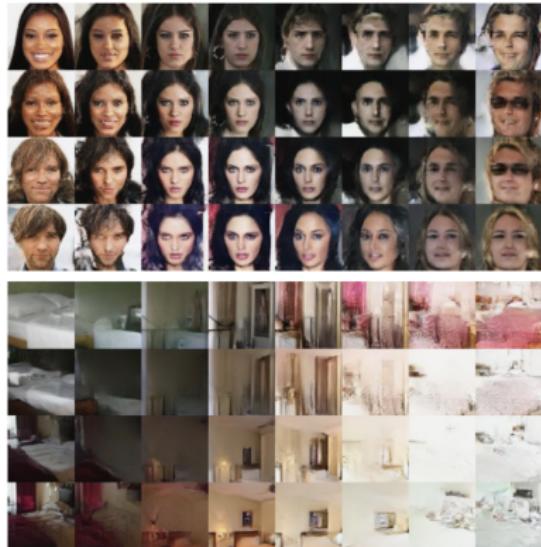
## Jacobian

$$\det\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right) = \det\begin{pmatrix} \mathbf{I}_d & \mathbf{0}_{d \times m-d} \\ \frac{\partial \mathbf{z}_{d:m}}{\partial \mathbf{x}_{1:d}} & \frac{\partial \mathbf{z}_{d:m}}{\partial \mathbf{x}_{d:m}} \end{pmatrix} = \prod_{i=1}^{m-d} \exp(c_1(\mathbf{x}_{1:d}, \theta)_i).$$

Non-Volume Preserving (the determinant of Jacobian  $\neq 0$ ).

# RealNVP

## Flow samples



# MAF vs IAF vs RealNVP

## MADE/MAF

$$\mathbf{x} = \sigma(\mathbf{z}) \odot \mathbf{z} + \mu(\mathbf{x}).$$

Estimating the density  $p(\mathbf{x}|\theta)$  - 1 pass, sampling -  $m$  passes.

## IAF

$$\mathbf{x} = \tilde{\sigma}(\mathbf{z}) \odot \mathbf{z} + \tilde{\mu}(\mathbf{z}).$$

Estimating the density  $p(\mathbf{x}|\theta)$  -  $m$  passes, sampling - 1 pass.

## RealNVP

$$\mathbf{x}_1 = \mathbf{z}_1;$$

$$\mathbf{x}_2 = \mathbf{z}_2 \odot \exp(c_1(\mathbf{z}_1, \theta)) + c_2(\mathbf{z}_1, \theta).$$

Estimating the density  $p(\mathbf{x}|\theta)$  - 1 pass, sampling - 1 pass.

# MAF vs IAF vs RealNVP

## RealNVP

$$\mathbf{x}_1 = \mathbf{z}_1;$$

$$\mathbf{x}_2 = \mathbf{z}_2 \odot \exp(c_1(\mathbf{z}_1, \theta)) + c_2(\mathbf{z}_1, \theta).$$

- ▶ Calculating the density  $p(\mathbf{x}|\theta)$  - 1 pass.
- ▶ Sampling - 1 pass.

RealNVP is a special case of MAF and IAF:

## MAF

$$\begin{cases} \mu_i = 0, \sigma_i = 1, i = 1, \dots, d; \\ \mu_i, \sigma_i - \text{functions of } \mathbf{x}_{1:d}, i = d + 1, \dots, m. \end{cases}$$

## IAF

$$\begin{cases} \tilde{\mu}_i = 0, \tilde{\sigma}_i = 1, i = 1, \dots, d; \\ \tilde{\mu}_i, \tilde{\sigma}_i - \text{functions of } \mathbf{z}_{1:d}, i = d + 1, \dots, m. \end{cases}$$

# Linear flows

## RealNVP

$$\begin{cases} \mathbf{z}_{1:d} = \mathbf{x}_{1:d}; \\ \mathbf{z}_{d:m} = \tau(\mathbf{x}_{d:m}, c(\mathbf{x}_{1:d})); \end{cases} \Leftrightarrow \begin{cases} \mathbf{x}_{1:d} = \mathbf{z}_{1:d}; \\ \mathbf{x}_{d:m} = \tau^{-1}(\mathbf{z}_{d:m}, c(\mathbf{z}_{1:d})). \end{cases}$$

- ▶ First step is a **split** operator which decouples a variable into 2 subparts:  $\mathbf{x}_1$  and  $\mathbf{x}_2$  (usually channel-wise).
- ▶ Stacking layers we should **permute** components.

$$\mathbf{z} = \mathbf{W}\mathbf{x}, \quad \mathbf{W} \in \mathbb{R}^{m \times m}$$

In general, we need  $O(m^3)$  to invert matrix.

## Invertibility

- ▶ Diagonal matrix  $O(m)$ .
- ▶ Triangular matrix  $O(m^2)$ .
- ▶ It is impossible to parametrize all invertible matrices.

# Linear flows

$$\mathbf{z} = \mathbf{W}\mathbf{x}, \quad \mathbf{W} \in \mathbb{R}^{m \times m}$$

## Matrix decompositions

- ▶ LU-decomposition

$$\mathbf{W} = \mathbf{P}\mathbf{L}\mathbf{U},$$

where  $\mathbf{P}$  is a permutation matrix,  $\mathbf{L}$  is lower triangular with positive diagonal,  $\mathbf{U}$  is upper triangular with positive diagonal.

- ▶ QR-decomposition

$$\mathbf{W} = \mathbf{Q}\mathbf{R},$$

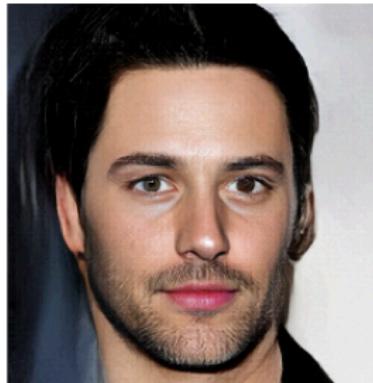
where  $\mathbf{Q}$  is an orthogonal matrix,  $\mathbf{R}$  is an upper triangular matrix with positive diagonal.

---

*Kingma D. P., Dhariwal P. Glow: Generative Flow with Invertible 1x1 Convolutions, 2018*

*Hoogeboom E., Van Den Berg R., and Welling M. Emerging convolutions for generative normalizing flows, 2019*

## Glow samples



## Summary

- ▶ Flow models use invertible transformation with tractable Jacobian.
- ▶ Planar and Sylvester flows are residual flows which use matrix determinant lemma.
- ▶ Autoregressive flows use autoregressive transform to make the Jacobian triangular.
- ▶ MAF/IAF is a special case of autoregressive flows.
- ▶ The RealNVP is an effective type of flow (special case of AR flows) that uses coupling layer.
- ▶ Linear flows try to parametrize set of invertible matrices via matrix decompositions.