

# Generative Modeling

Normalizing flows

Denis Derkach, Artem Ryzhikov, Maxim Artemev

Laboratory for methods of big data analysis

Spring 2021



# In this Lecture

- ▶ Normalizing flows problem statement.
- ▶ Block schemes for Normalizing flows.
- ▶ MAF/IAF duality

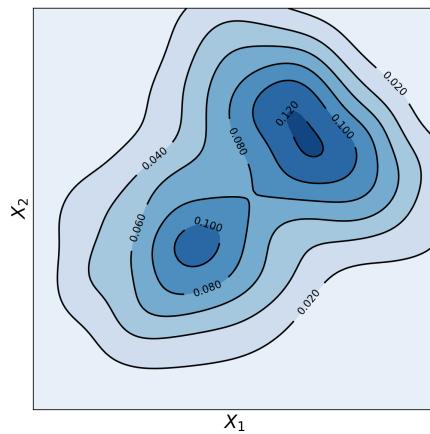
# Motivation



# Motivation

$$x_i \sim p_x(x)$$

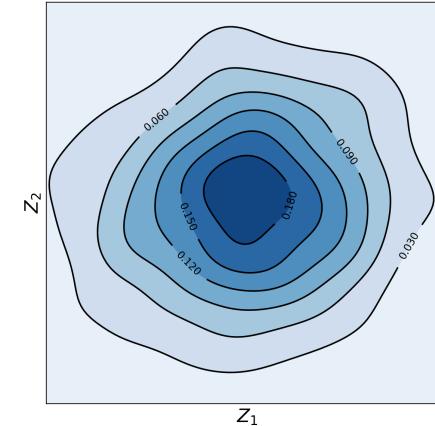
$$p_x(x) - ?$$



$$z_i \sim p_z(z)$$

$$p_z(z) - \text{known}$$

$$\begin{array}{c} z = f(x) \\ \longrightarrow \\ x = T(z) \end{array}$$



- ▶ Generative model:
  - Likelihood evaluation;
  - Sampling procedure.
- ▶ Use **deterministic map** to known distribution.

# Change of variables

For  $p_z(z)$  some pdf and  $z = \mathbf{f}(x)$  known then  $p_x(x) :$

$$p_x(x_i) = p_z(\mathbf{f}(x_i)) \left| \det \frac{\partial \mathbf{f}(x_i)}{\partial x_i} \right|,$$

Ratio of volume  $\partial z$  to  
new volume  $\partial x$

1<sup>st</sup> derivative matrix:

$$\frac{\partial \mathbf{f}(x_i)}{\partial x_i} = \begin{pmatrix} \frac{\partial \mathbf{f}(x_i)_1}{\partial x_{i1}} & \dots & \frac{\partial \mathbf{f}(x_i)_1}{\partial x_{in}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{f}(x_i)_m}{\partial x_{i1}} & \dots & \frac{\partial \mathbf{f}(x_i)_m}{\partial x_{in}} \end{pmatrix}.$$

# 1D example

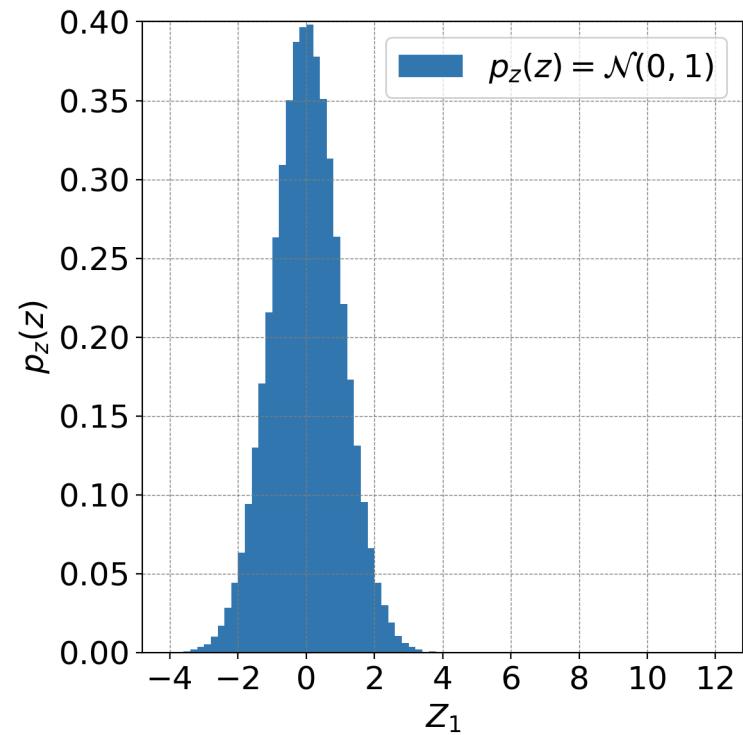


$$z = f(x) = 0.5x - 2.5$$



$$x_i \sim p_x(x)$$

$$p_x(x) - ?$$



$$z_i \sim p_z(z)$$

$$p_z(z) = \mathcal{N}(0, 1)$$

# 1D example

For function  $\mathbf{f}(x)$ :

$$z = \mathbf{f}(x) = 0.5x - 2.5$$

Jacobi matrix:

$$\frac{\partial \mathbf{f}(x_i)}{\partial x_i} = (0.5)$$

Jacobian:

$$\left| \det \frac{\partial \mathbf{f}(x_i)}{\partial x_i} \right| = 0.5$$

# 1D example

Change of variables formulas:

$$p_x(x_i) = \mathbf{p_z}(\mathbf{f}(x_i)) \left| \det \frac{\partial \mathbf{f}(x_i)}{\partial x_i} \right|,$$

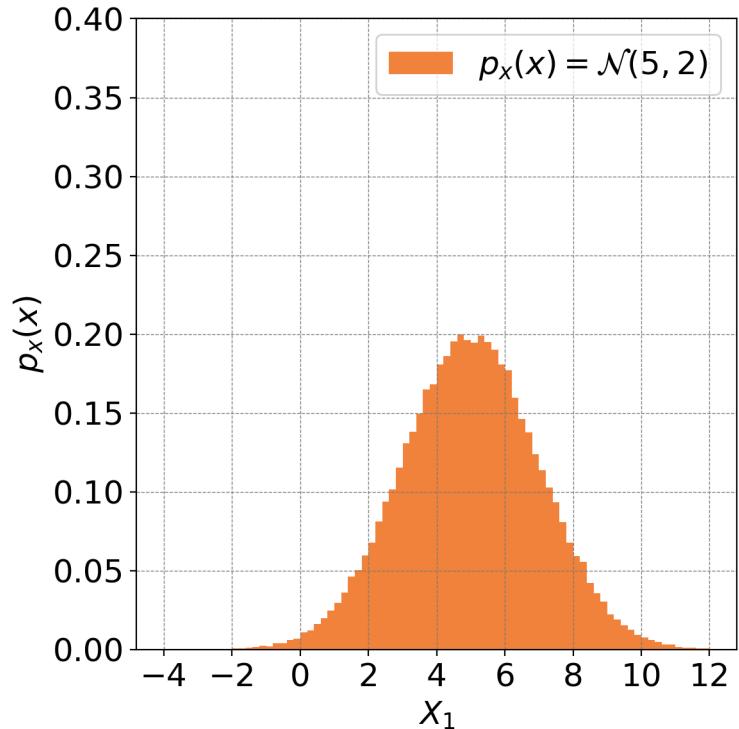
Using normal distribution definition:

$$\mathbf{p_z}(z_i) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(z_i)^2}{2}}$$

$$p_x(x_i) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(0.5x_i - 2.5)^2}{2}} * 0.5$$

$$= \frac{1}{2\sqrt{2\pi}} e^{-\frac{(x_i - 5)^2}{2*2^2}} = \mathcal{N}(5, 2)$$

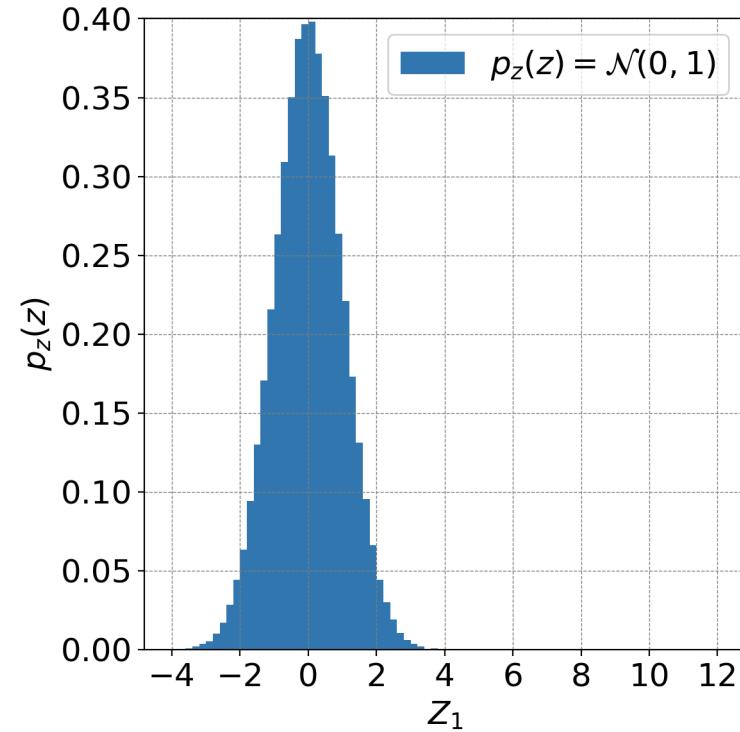
# 1D example



$$x_i \sim p_x(x)$$

$$p_x(x) = \mathcal{N}(5, 2)$$

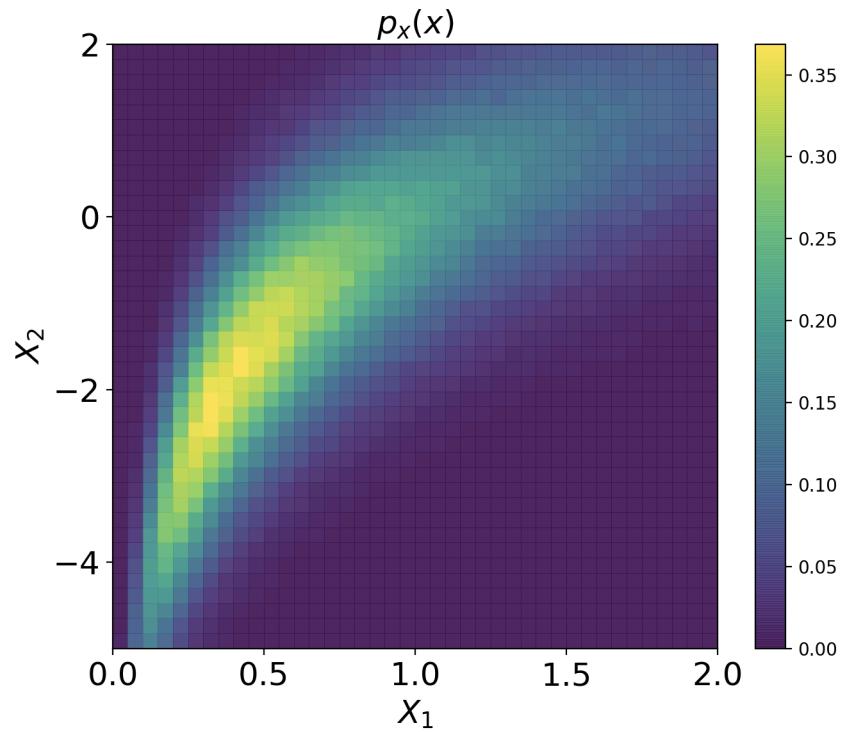
$$z = f(x) = 0.5x - 2.5$$



$$z_i \sim p_z(z)$$

$$p_z(z) = \mathcal{N}(0, 1)$$

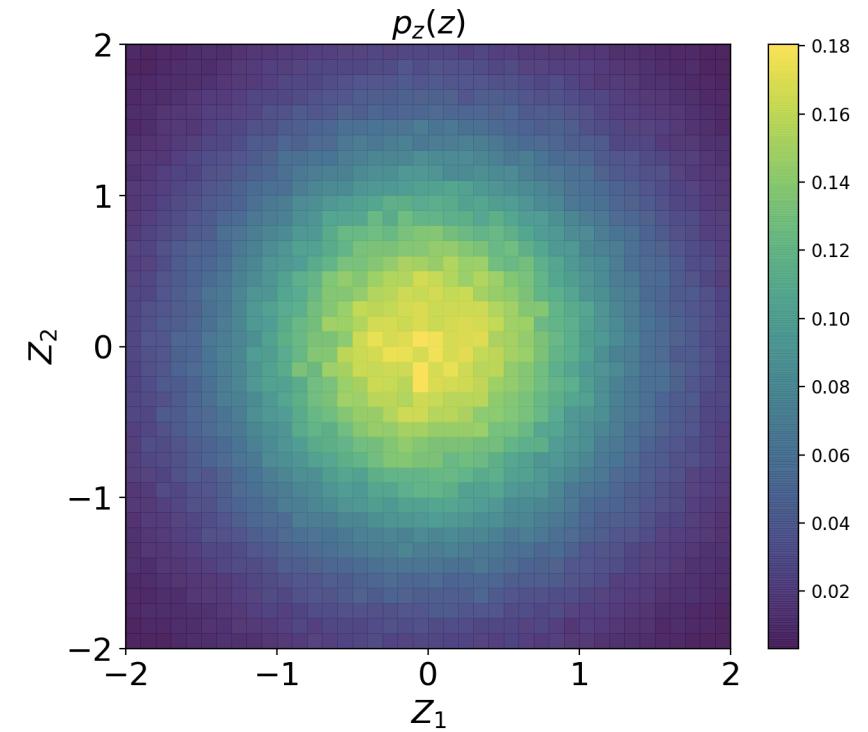
# 2D example



$$x_i \sim p_x(x)$$

$$p_x(x) - ?$$

$$\mathbf{f}(x) = \begin{cases} z_1 = \ln x_1 \\ z_2 = x_2 - 2 \ln x_1 \end{cases}$$



$$z_i \sim p_z(z)$$

$$p_z(z) = \frac{1}{2\pi} e^{-\frac{z_1^2+z_2^2}{2}}$$

# 2D example

For function:

$$z = \mathbf{f}(x) = \begin{cases} z_1 = \ln x_1 \\ z_2 = x_2 - 2 \ln x_1 \end{cases}$$

Linear derivatives are:

$$\frac{\partial \mathbf{f}(x)}{\partial x} = \begin{pmatrix} 1/x_1 & 0 \\ -2/x_{x1} & 1 \end{pmatrix}$$

Jacobian:

$$\left| \det \frac{\partial \mathbf{f}(x)}{\partial x} \right| = \frac{1}{x_1}$$

# 2D example

Change of variables:

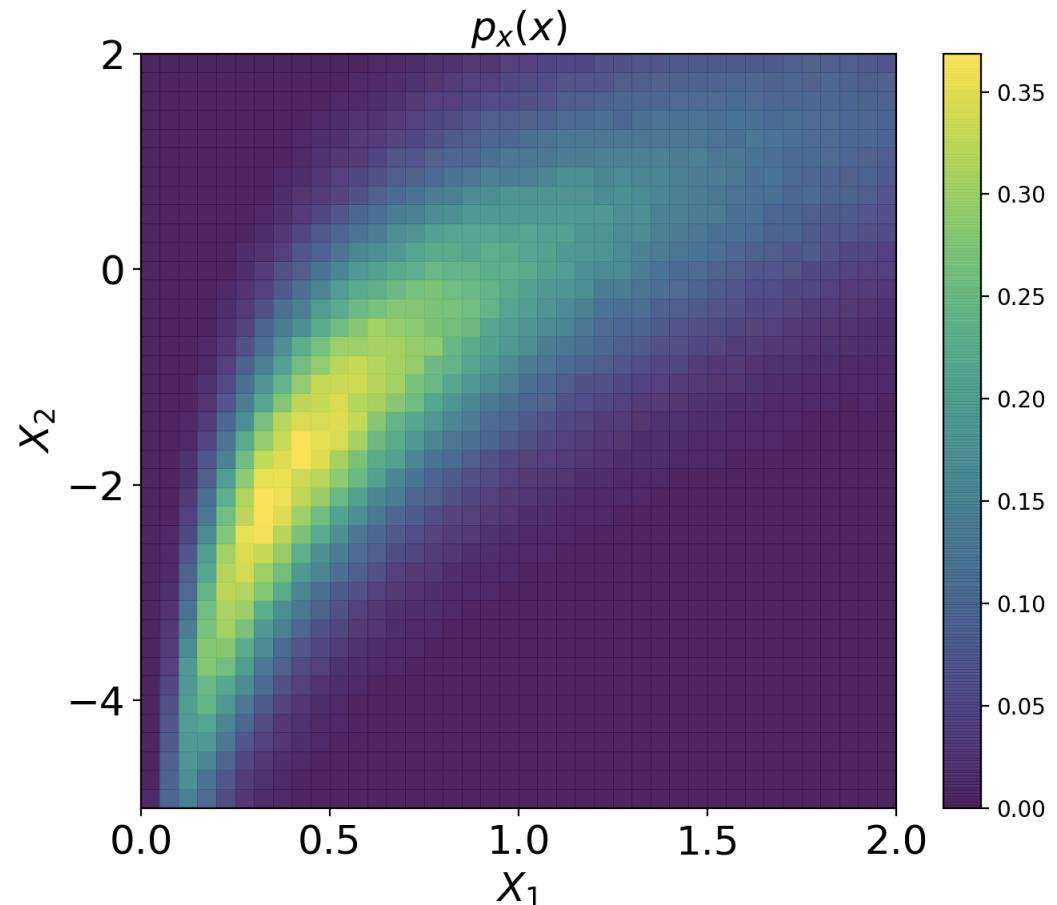
$$p_x(x_i) = \mathbf{p_z}(\mathbf{f}(x_i)) \left| \det \frac{\partial \mathbf{f}(x_i)}{\partial x_i} \right|,$$

Insert known shapes:

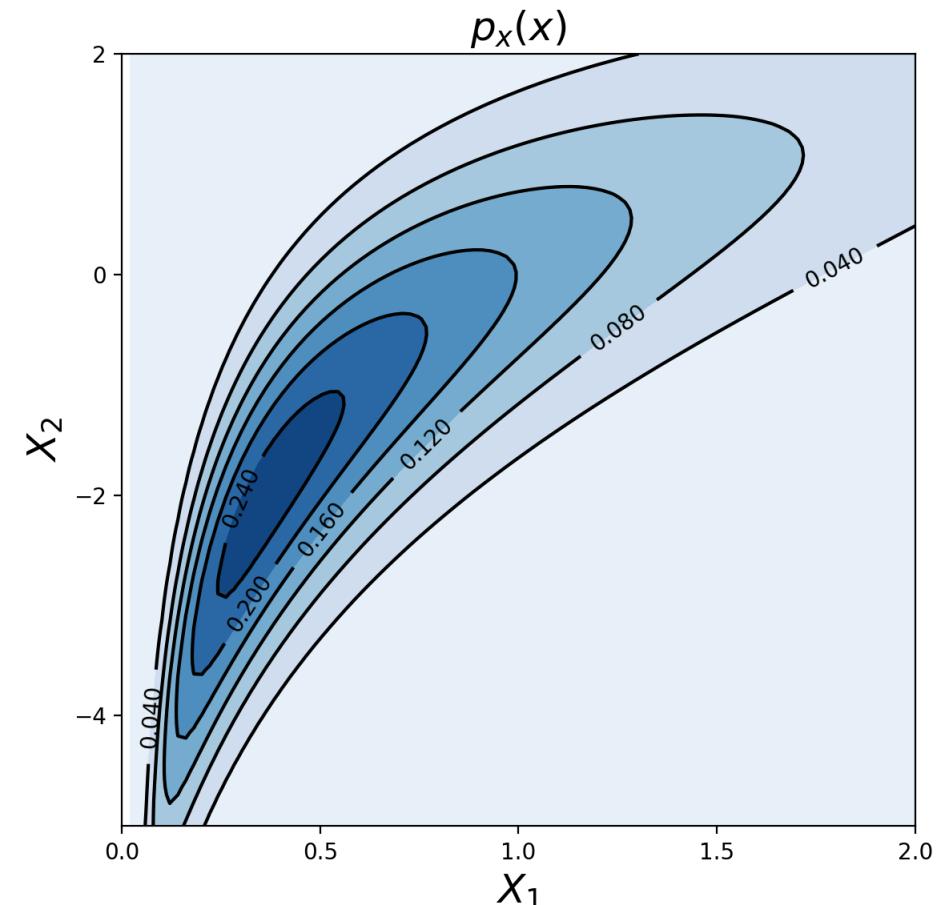
$$\mathbf{p_z}(z_i) = \frac{1}{2\pi} e^{-\frac{z_1^2+z_2^2}{2}}$$

$$p_x(x_i) = \frac{1}{2\pi} e^{-\frac{(\ln x_1)^2 + (x_2 - 2\ln x_1)^2}{2}} * \frac{1}{x_1}$$

# 2D example



Experiment



Theory

# Motivation recap

- ▶ Use **deterministic map** to known distribution:

$$\mathbf{z} = \mathbf{f}(\mathbf{x}).$$

- ▶ In this case, change of variables formula:

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

- ▶ Inverse map:

$$\mathbf{x} = \mathbf{T}(\mathbf{z}).$$

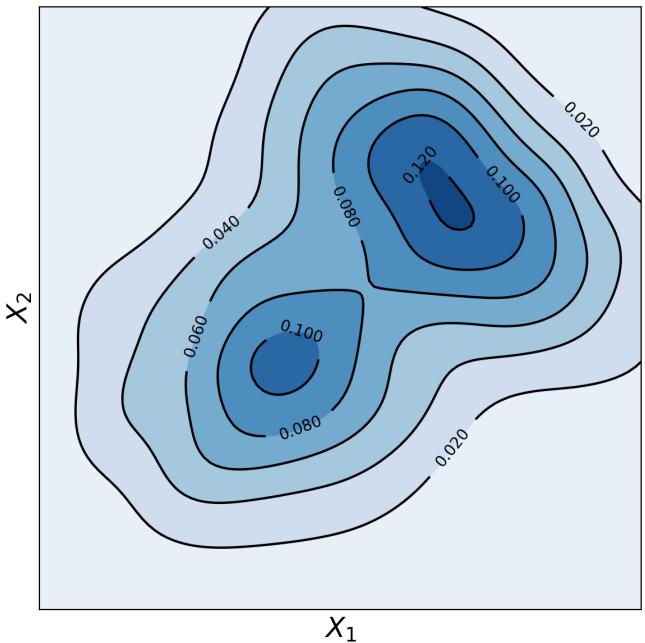
- ▶ In this case, change of variables formula:

$$p_{\mathbf{z}}(\mathbf{z}) = p_x(\mathbf{T}(\mathbf{z})) \left| \det \frac{\partial \mathbf{T}(\mathbf{z})}{\partial \mathbf{z}} \right|.$$

# Motivation

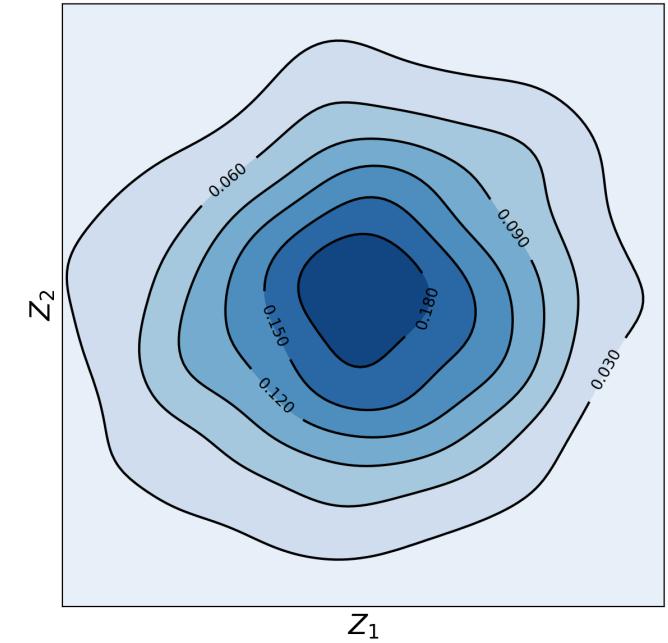
$$x_i \sim p_x(x)$$

$$p_x(x) - ?$$



$$z_i \sim p_z(z)$$

$$p_z(z) - \text{known}$$



Sampling  
←  
 $\boldsymbol{x} = \mathbf{T}(\mathbf{z})$   
Likelihood  
evaluation  
→  
 $\mathbf{z} = \mathbf{f}(\boldsymbol{x})$

**Can we have invertible function?**

# Motivation recap

- ▶ Use **deterministic map** to known distribution:

$$\mathbf{z} = \mathbf{f}(\mathbf{x}).$$

- ▶ In this case, change of variables formula:

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

- ▶ Inverse map:

$$\mathbf{x} = \mathbf{T}(\mathbf{z}) = \mathbf{f}^{-1}(\mathbf{z}).$$

- ▶ In this case, change of variables formula:

$$\mathbf{p}_z(\mathbf{z}) = p_x(f^{-1}(\mathbf{z})) \left| \det \frac{\partial \mathbf{f}^{-1}(\mathbf{z})}{\partial \mathbf{z}} \right| = p_x(f^{-1}(\mathbf{z})) \left| \det \frac{\partial \mathbf{f}(\mathbf{z})}{\partial \mathbf{z}} \right|^{-1}.$$

$f$ :

- ▶ Invertible.
- ▶ Differentiable.

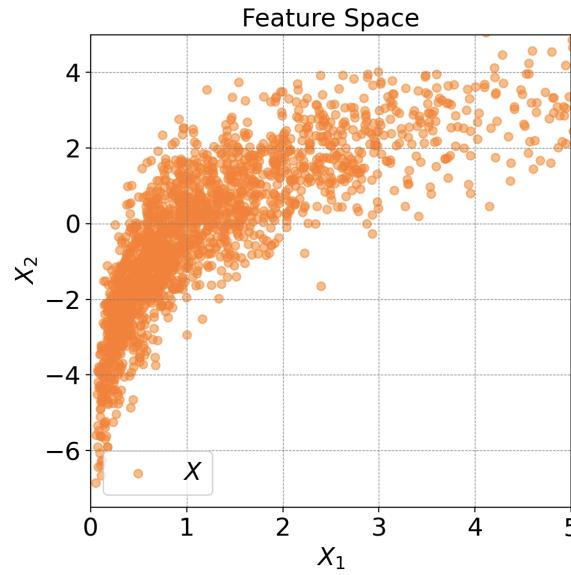
# Normalizing flows



# Problem Statement

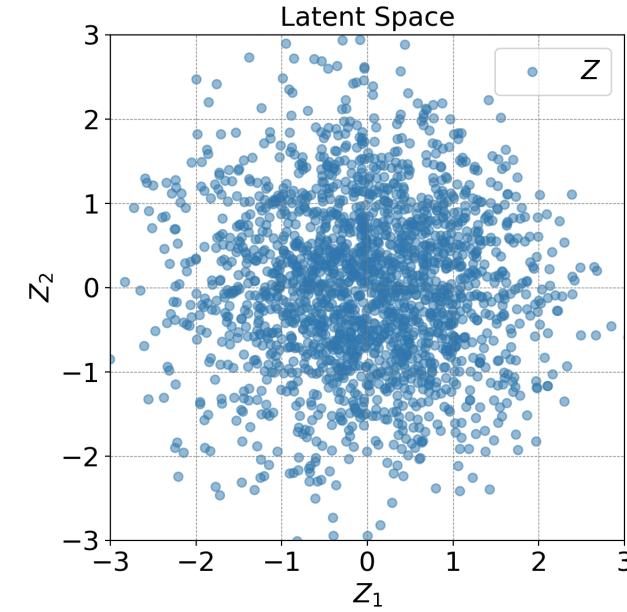
$$x_i \sim p_x(x) - \text{data}$$

$$p_x(x) - ?$$



$$z_i \sim p_z(z)$$

$$p_z(z) \text{ known}$$



$$z = f(x) - ?$$



$$x = f^{-1}(z) - ?$$

- **We have:** real objects  $\{x_i\}$
- **Task:** find invertible and differentiable  $f: z_i = f(x_i)$ , such that  $z_i \sim p_z(z)$ .  
For some known:  $p_z(z)$ .

# Loss function

Use log-likelihood:

$$\log \mathcal{L} = -\frac{1}{n} \sum_{i=1}^n \log p_x(x_i)$$

Change of variables:

$$p_x(x_i) = \mathbf{p_z}(\mathbf{f}(x_i)) \left| \det \frac{\partial \mathbf{f}(x_i)}{\partial x_i} \right|$$

Thus:

$$\log \mathcal{L} = -\frac{1}{n} \sum_{i=1}^n \left( \log \mathbf{p_z}(\mathbf{f}(x_i)) + \log \left| \det \frac{\partial \mathbf{f}(x_i)}{\partial x_i} \right| \right)$$

# Training Algorithm

**for** number of training iterations **do**:

- ▶ Sample  $m$  of real objects  $\{x_1, x_2, \dots, x_m\}$ .
- ▶ Calculate loss function:

$$L = -\frac{1}{m} \sum_{i=1}^m \left( \log \mathbf{p_z}(\mathbf{f}(x_i)) + \log \left| \det \frac{\partial \mathbf{f}(x_i)}{\partial x_i} \right| \right)$$

- ▶ Update parameters of the function  $z_i = \mathbf{f}(x_i)$ :

$$\theta_f = \theta_f - \nabla_{\theta_f} L$$

# Generation Algorithm

- ▶ Sample  $m$  objects  $\{z_1, z_2, \dots, z_m\}$ .
- ▶ Generate new objects using the learned function:

$$x_i = \mathbf{f}^{-1}(z_i)$$

# Planar flows

- ▶ Transformation:

$$x = f_{\theta}^{-1}(z) = z + uh(w^T z + b),$$

$\theta = (w \in \mathbb{R}^D, u \in \mathbb{R}^D, b \in \mathbb{R})$  – parameters,  $h(\cdot)$  – elementwise nonlinearity.

- ▶ Linear derivative matrix:

$$\frac{\partial f_{\theta}^{-1}}{\partial z} = I + uh'(w^T z + b)w^T.$$

- ▶ Jacobian:

$$\det \frac{\partial f_{\theta}^{-1}}{\partial z} = 1 + h'(w^T z + b)w^T u.$$

**Is it invertible? For  $h(\cdot) = \tanh(\cdot)$  need  $w^T u > -1$ .**

# Planar flows: grid behavior

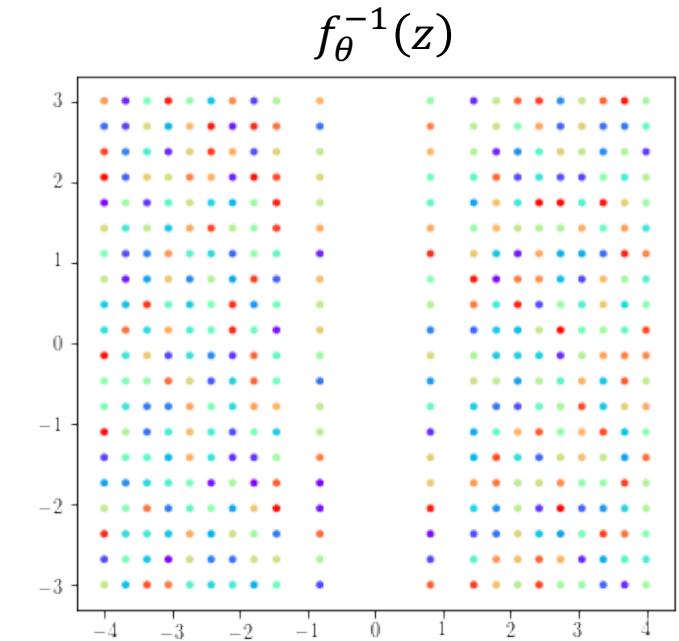
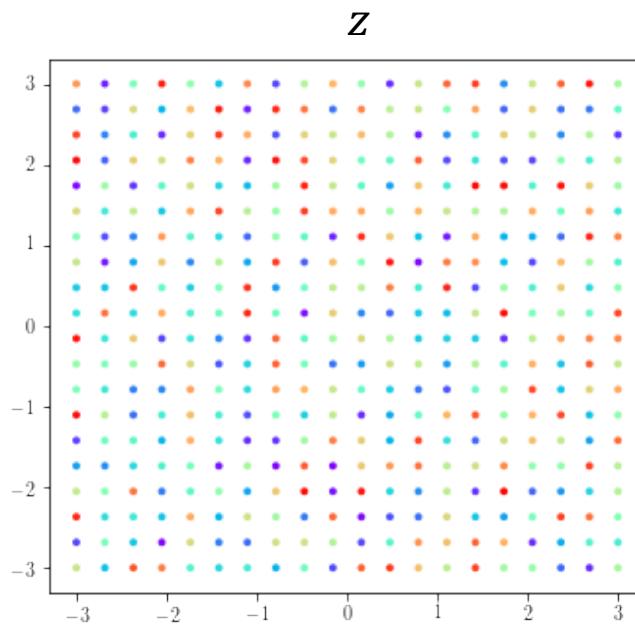
$$x = f_{\theta}^{-1}(z) = z + uh(w^T z + b)$$

$$w = [5; 0]^T$$

$$u = [1; 0]^T$$

$$b = 0$$

$$h(x) = \tanh(x)$$

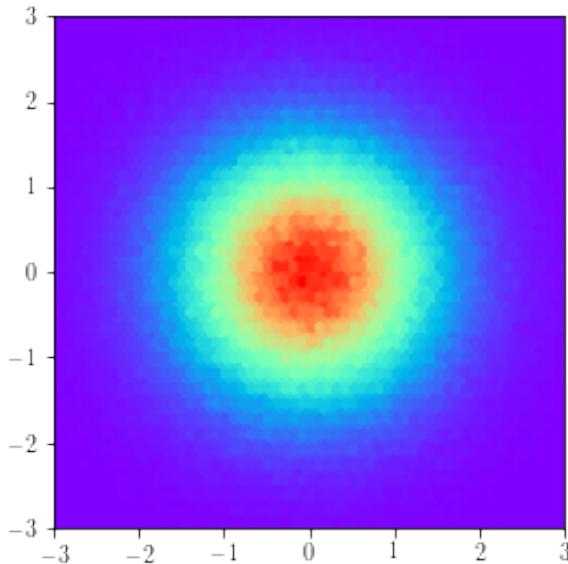


$$x = f_{\theta}^{-1}(z) = \begin{cases} z_1 + \tanh(5z_1) \\ z_2 \end{cases}$$

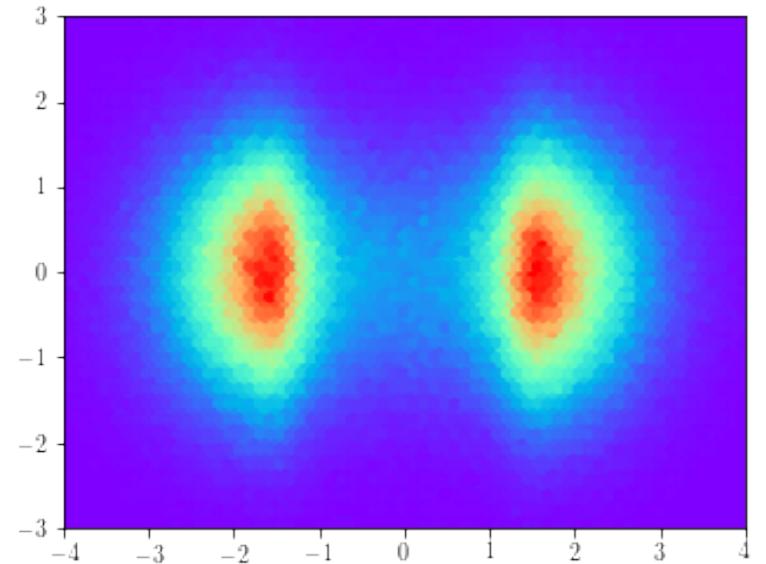
**Grid will be pushed away from zero.**

# Planar flows: one step

$$x = f_{\theta}^{-1}(z) = \begin{cases} z_1 + \tanh(5z_1) \\ z_2 \end{cases}$$



$$x = f^{-1}(z)$$

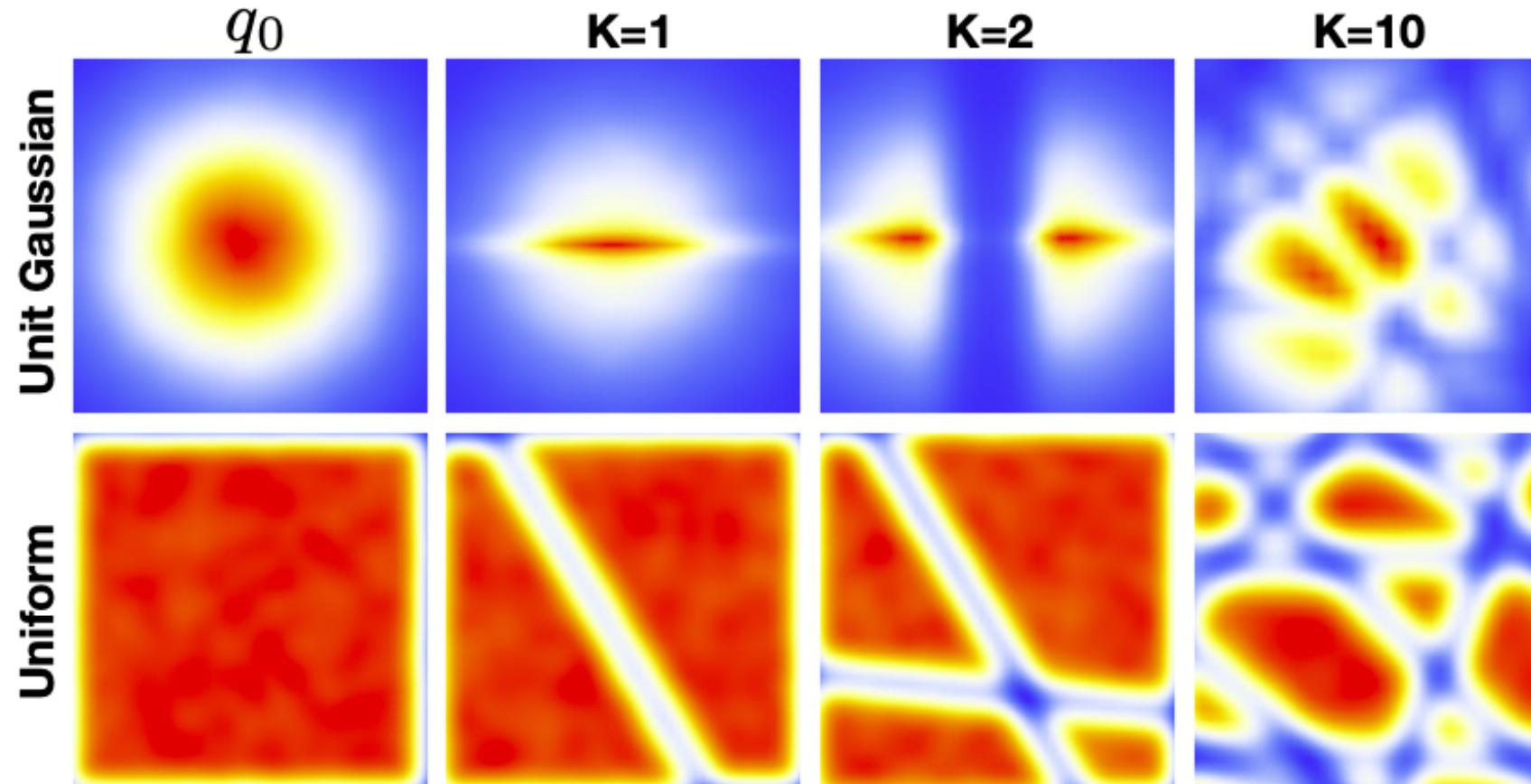


We can obtain a more complicated figure.

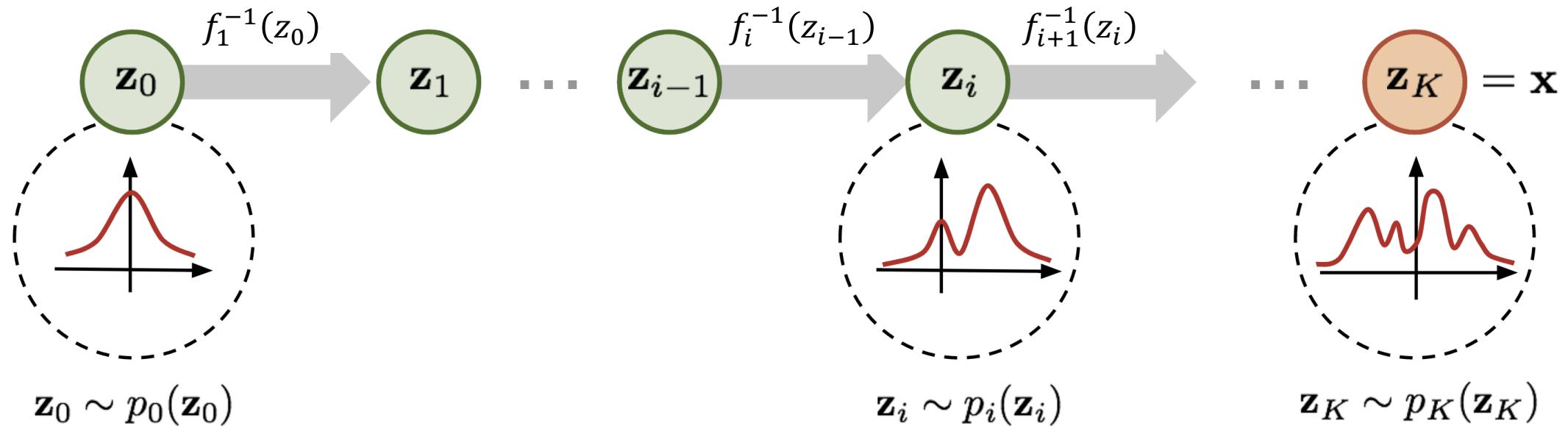
# Planar flows: more steps

$$x = f_{\theta}^{-1}(z) = z + uh(w^T z + b)$$

Change transformation parameters at each step K for any prior distribution.



# Stack more layers

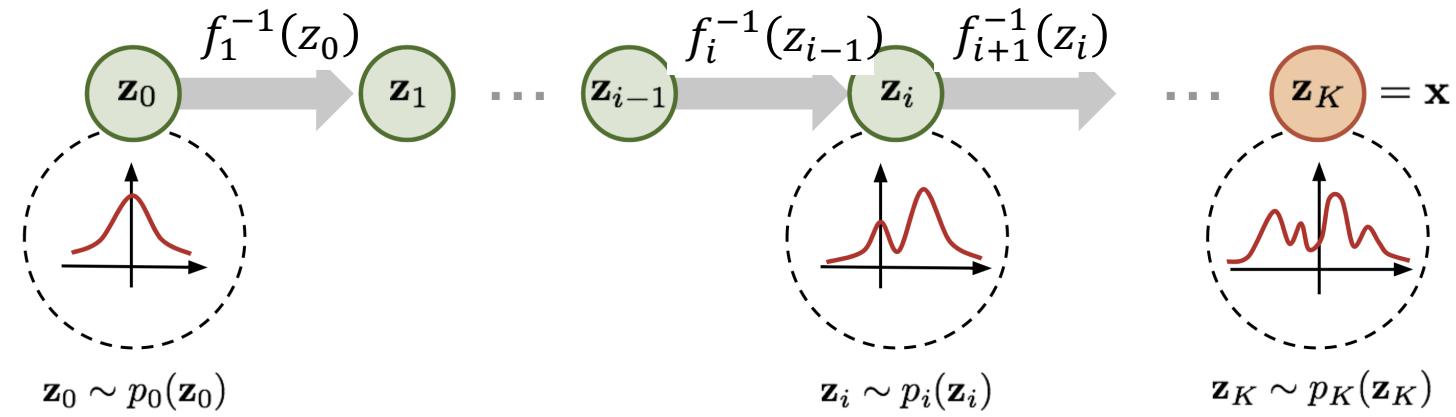


- ▶ We take several transformations consecutively.
- ▶ At each layer we have new function.

# Layers in Flows

Let  $z_0 = \mathbf{f}_1(z_1), z_1 = \mathbf{f}_2(x)$ .

Then:



$$p_x(x) = \mathbf{p}_{\mathbf{y}}(\mathbf{f}_2(x)) \left| \det \frac{\partial \mathbf{f}_2(x)}{\partial x} \right|$$

$$p_{z_1}(z_1) = \mathbf{p}_{z_0}(\mathbf{f}_1(z_1)) \left| \det \frac{\partial \mathbf{f}_1(z_1)}{\partial z_1} \right|$$

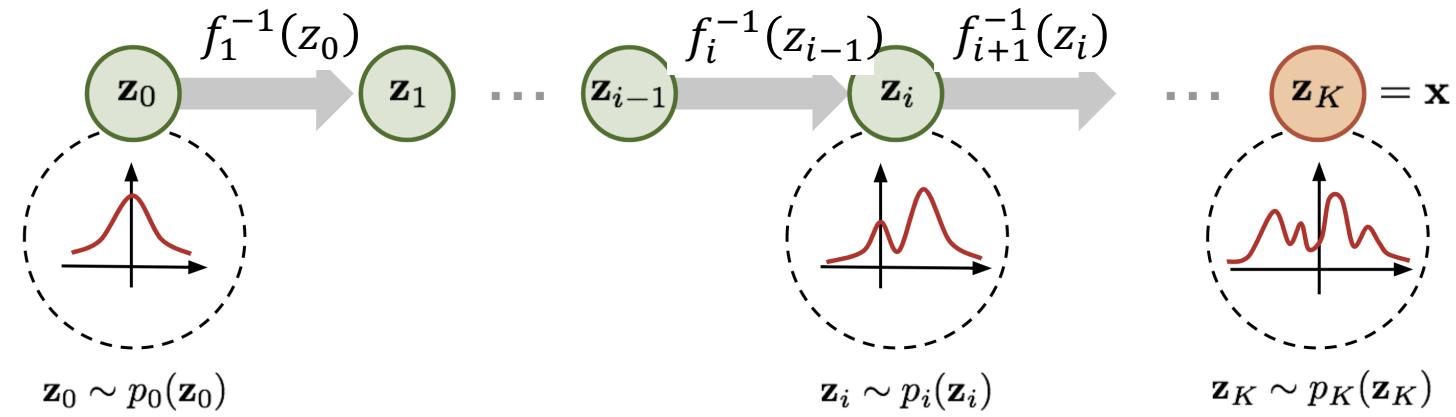
Which means:

$$p_x(x) = \mathbf{p}_{z_0}(\mathbf{f}_2(\mathbf{f}_1(x))) \left| \det \frac{\partial \mathbf{f}_1(x)}{\partial z_1} \right| \left| \det \frac{\partial \mathbf{f}_2(x)}{\partial x} \right|$$

# Layers in Flows

Let  $z_0 = \mathbf{f}_1(z_1), z_1 = \mathbf{f}_2(x)$ .

Then:



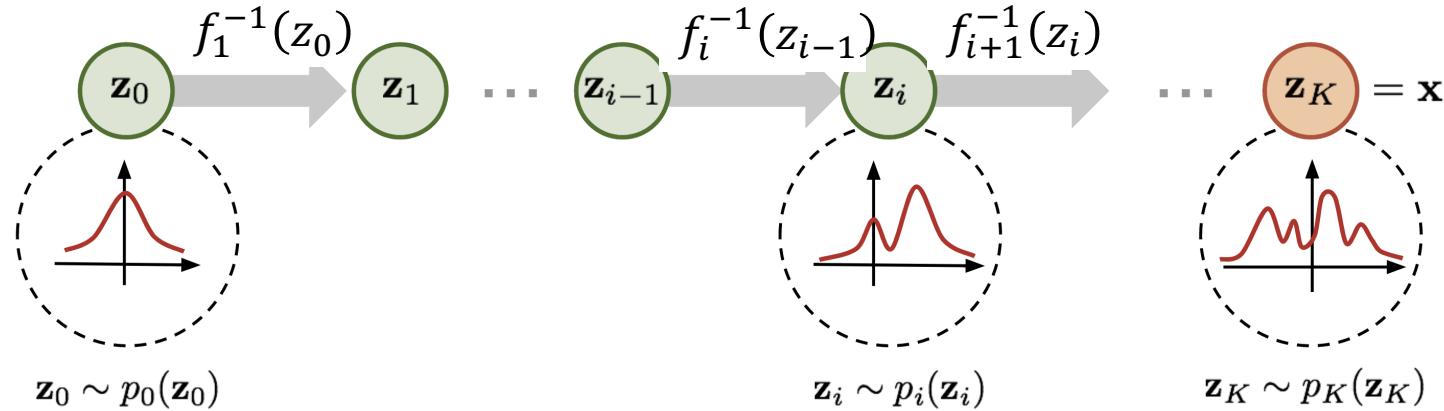
$$p_x(x) = \mathbf{p}_y(\mathbf{f}_2(x)) \left| \det \frac{\partial \mathbf{f}_2(x)}{\partial x} \right|$$

$$p_{z_1}(z_1) = \mathbf{p}_{z_0}(\mathbf{f}_1(z_1)) \left| \det \frac{\partial \mathbf{f}_1(z_1)}{\partial z_1} \right|$$

Which means:

$$p_x(x) = \mathbf{p}_{z_0}(\mathbf{f}_2(\mathbf{f}_1(x))) \left| \det \frac{\partial \mathbf{f}_1(x)}{\partial z_1} \right| \left| \det \frac{\partial \mathbf{f}_2(x)}{\partial x} \right|$$

# Flow discussion



$$p_x(x) = p_{\mathbf{z}_0}(\mathbf{f}_K(\dots(\mathbf{f}_1(x)))) \left| \det \frac{\partial \mathbf{f}_1(z_1)}{\partial z_1} \right| \dots \left| \det \frac{\partial \mathbf{f}_K(z_k)}{\partial z_k} \right|.$$

- ▶ It is possible to obtain  $p_x(x)$  consecutively changing observables.
- ▶ The overall transformation is invertible if individual layers are invertible.
- ▶ Dimensions of each observable is the same.
- ▶ Fit is performed using ML estimate.
- ▶ Need to calculate determinant.

# Jacobian Problems

$$\frac{\partial \mathbf{f}(x_i)}{\partial x_i} = \begin{pmatrix} \frac{\partial \mathbf{f}(x_i)_1}{\partial x_{i1}} & \dots & \frac{\partial \mathbf{f}(x_i)_1}{\partial x_{in}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{f}(x_i)_m}{\partial x_{i1}} & \dots & \frac{\partial \mathbf{f}(x_i)_m}{\partial x_{in}} \end{pmatrix}$$

- ▶  $d \times d$  determinant is **too expensive** to compute with  $\mathcal{O}(d^3)$  computations.
- ▶ **Idea: take only transformations with triangular matrix, in this case  $\mathcal{O}(d)$  computations.**
- ▶ **Bogachev theorem:** There always exists a unique (up to ordering) increasing triangular map that transforms a source density to a target density.

**We came to autoregressive models.**

# Block models



# Block matrix for Jacobian

$$\frac{\partial \mathbf{f}(x)}{\partial x} = \begin{pmatrix} \mathbb{I}_d & 0 \\ \frac{\partial z_{d+1:D}}{\partial x_{d+1:D}} & S \end{pmatrix}$$

- ▶ Randomly choose  $d$  such that we have two disjoint subsets of observables:  $z_{1:d}$  and  $z_{d+1:D}$ .
- ▶ Insert a block transform.
- ▶ Repeat for several layers.
- ▶ If needed insert scaling layers.
- ▶ Fit simultaneously.

# Non-linear Independent Components Estimation

$$z = \mathbf{f}(x) = \begin{cases} z_{1:d} = x_{1:d} \\ z_{d+1:D} = x_{d+1:D} - m(x_{1:d}) \end{cases}$$

- ▶  $m(x_{1:d})$  – **neural networks** with  $d$  inputs and  $D - d$  outputs;
- ▶ easy to invert;
- ▶ scaling layer  $x_i = s_i z_i$  can be added.
- ▶  $\det J = \prod_{i=1}^n s_i$ .
- ▶ Based on NADE autoregressive model.

# Non-linear Independent Components Estimation



(a) Model trained on MNIST



(b) Model trained on TFD

<https://arxiv.org/abs/1410.8516>

# Real-NVP

$$z = \mathbf{f}(x) = \begin{cases} z_{1:d} = x_{1:d} \\ z_{d+1:D} = x_{d+1:D} \odot \exp(\mathbf{s}(x_{1:d})) + \mathbf{t}(x_{1:d}) \end{cases}$$

- ▶  $\mathbf{s}(x_{1:d})$  и  $\mathbf{t}(x_{1:d})$  – **neural networks** with  $d$  inputs and  $D - d$  outputs.
- ▶ Invertible.
- ▶  $\det J_k = \exp \sum_{i=d+1}^D (\alpha_\theta(z_{1:d}))_i$  for k-th layer.
- ▶ Inspired by RNADE.

<https://arxiv.org/abs/1605.08803>

# R-NVP: results

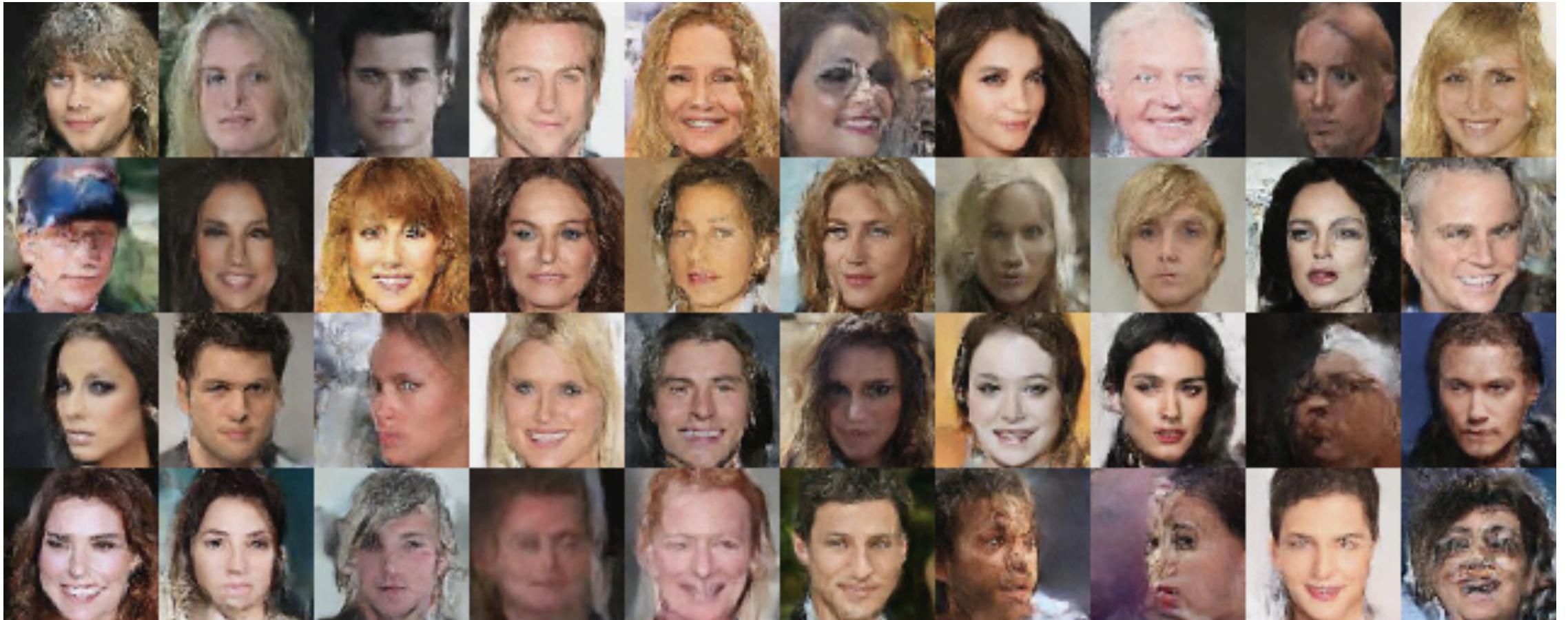


Рис.: [https://github.com/laurent-dinh/laurent-dinh.github.io/blob/master/img/real\\_nvp\\_fig/celeba\\_samples.png](https://github.com/laurent-dinh/laurent-dinh.github.io/blob/master/img/real_nvp_fig/celeba_samples.png)

# Discussion

- ▶ Normalizing flows based on block scheme have nice results.
- ▶ Block scheme allows for autoregressive model insertion.
- ▶ Fairly fast: need only one pass to get the sampling and likelihood evaluation, can run in parallel.
- ▶ Autoregressive quality is not conserved in multiple layers.

# Masked Autoregressive Flow (MAF)



# Masked Autoregressive Flow (MAF)

$$z = \mathbf{f}(x) = \begin{cases} z_1 = (x_1 - \mu_1) \exp(-s_1) \\ \dots \\ z_k = (x_k - \boldsymbol{\mu}_k(x_{1:k-1})) \odot \exp(-\boldsymbol{s}_k(x_{1:k-1})) \\ \dots \end{cases}$$

- ▶  $\boldsymbol{\mu}_d(x_{1:d-1})$  и  $\boldsymbol{s}_d(x_{1:d-1})$  – **neural networks.**
- ▶ MADE architecture inspired.
  - Fast in likelihood evaluation.
  - Slow ancestral sampling.

<https://arxiv.org/abs/1705.07057>

# Masked Autoregressive Flow (MAF): Jacobian

- ▶ Low-triangular matrix

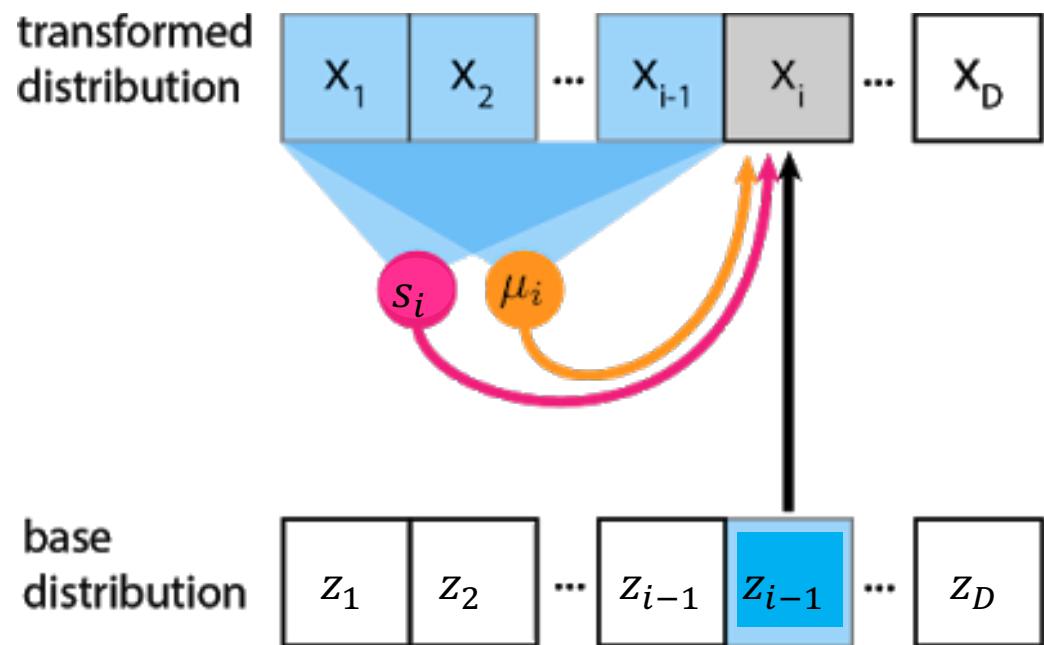
$$\frac{\partial \mathbf{f}(x)}{\partial x} = \begin{pmatrix} \exp(-\mathbf{s}_1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \frac{\partial z_D}{\partial x_1} & \cdots & \exp(-\mathbf{s}_D(x_{1:D-1})) \end{pmatrix}$$

- ▶ Jacobian:

$$\left| \det \frac{\partial \mathbf{f}(x)}{\partial x} \right| = \exp\left(-\sum_{j=1}^D \mathbf{s}_d(x_{1:d-1})\right)$$

# MAF sampling

$$x = \mathbf{f}^{-1}(z) = \begin{cases} x_1 = z_1 \exp(s_1) + \mu_1 \\ x_d = z_d \exp(\mathbf{s}_d(x_{1:d-1})) + \boldsymbol{\mu}_d(x_{1:d-1}) \end{cases}$$



► Forward pass:

$$z \sim N(0; 1)$$

$$x_1 = \exp(s_1) z_1 + \mu_1$$

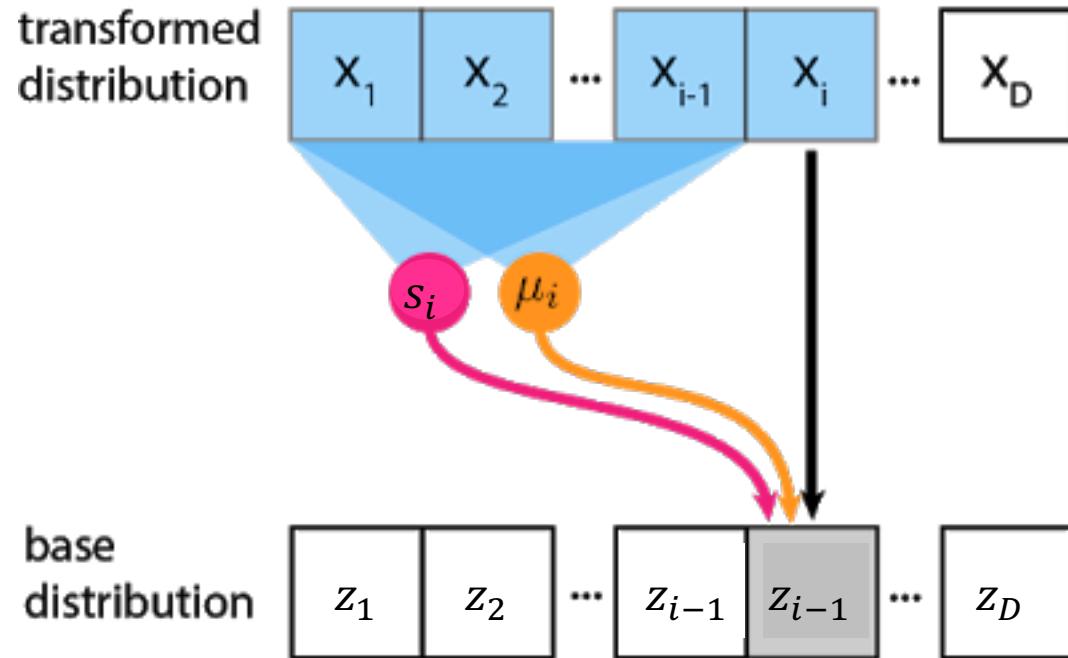
$$x_2 = \exp(s_2(x_1)) z_2 + \mu_2(x_1)$$

And so on.

**Consecutive and slow.**

# MAF Likelihood evaluation

$$z = \mathbf{f}(x) = \begin{cases} z_1 = (x_1 - \mu_1) \exp(-s_1) \\ z_k = (x_k - \boldsymbol{\mu}_k(x_{1:k-1})) \odot \exp(-\boldsymbol{s}_k(x_{1:k-1})) \end{cases}$$



► Inverse pass:

compute all  $\mu$  and  $s$ ;  
evaluate  $z$ .

Likelihood evaluation is fast and parallelizable.

Training is fast.

# MAF: results

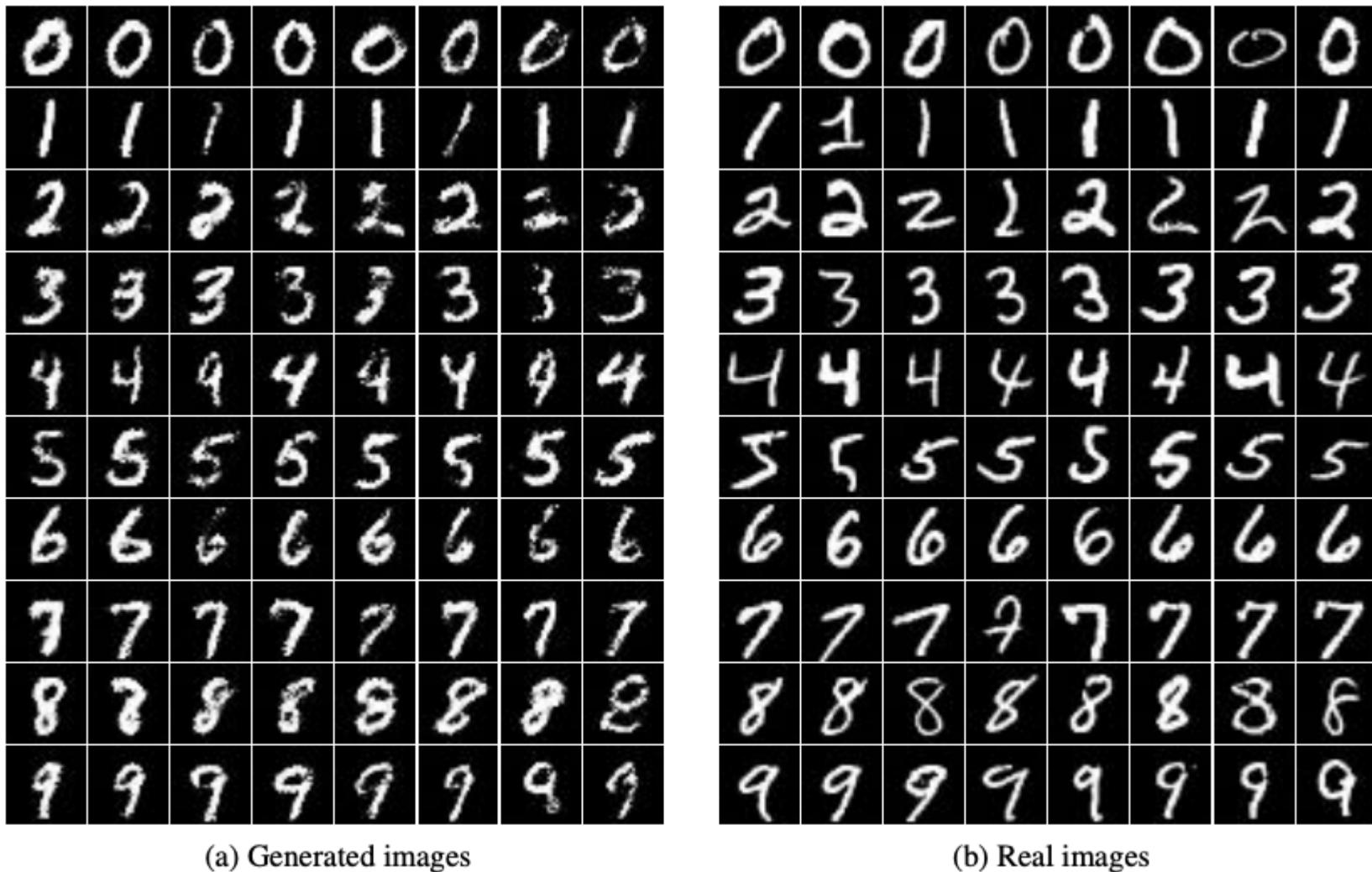


Figure 3: Class-conditional generated and real images from MNIST. Rows are different classes. Generated images are sorted by decreasing log likelihood from left to right.

# Inverse Autoregressive Flow (IAF)

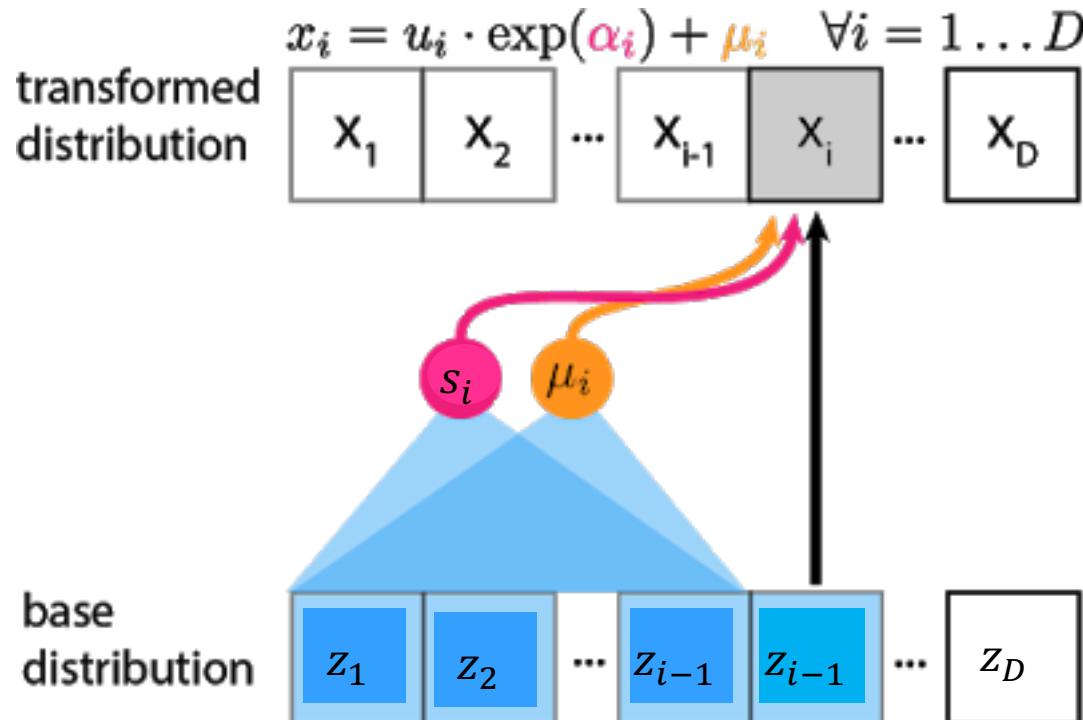
$$z = \mathbf{f}(x) = \begin{cases} z_1 = (x_1 - \mu_1) \exp(-s_1) \\ z_d = (x_d - \boldsymbol{\mu}_d(z_{1:d-1})) \exp(-\boldsymbol{s}_d(z_{1:d-1})) \end{cases}$$

- ▶  $\boldsymbol{\mu}_d(z_{1:d-1})$  и  $\boldsymbol{s}_d(z_{1:d-1})$  – **neural networks.**
- ▶ Similar to MAF but with inverse problems by construction:
  - Fast to sample.
  - Slow to evaluate likelihood.

<https://arxiv.org/abs/1606.04934>

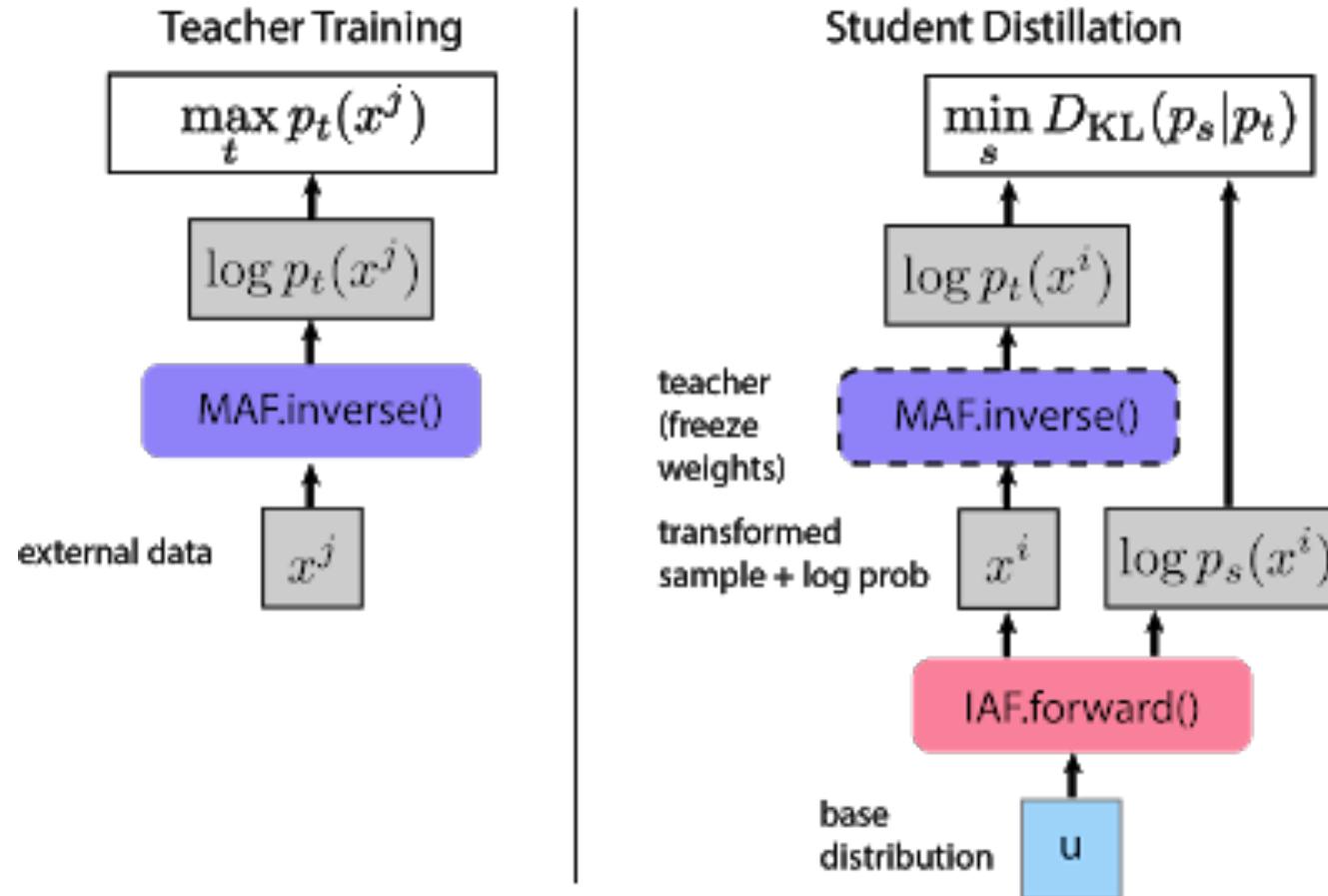
# Inverse Autoregressive Flow (IAF)

$$x = \mathbf{f}^{-1}(z) = \begin{cases} x_1 = z_1 \exp(s_1) + \mu_1 \\ x_d = z_d \exp(\mathbf{s}_d(z_{1:d-1})) + \mathbf{\mu}_d(z_{1:d-1}) \end{cases}$$



- ▶ Forward pass:  
compute all  $\mu$  and  $s$ ;  
evaluate  $x$ .
- Sampling** is fast and parallelizable.
- Training is slow.**

# Probability Density Distillation



- ▶ Two-part training with a teacher and student model.
- ▶ Teacher (MAF) trained first, then student (IAF) initialized.
- ▶ Student,  $s$ , is trained to match the teachers' distribution  $t$  using KL divergence:

$$KL(p_s || p_t) = H(p_s, p_t) - H(p_s).$$

Which is evaluated using MAF.

<https://arxiv.org/pdf/1711.10433.pdf>

# Conclusions

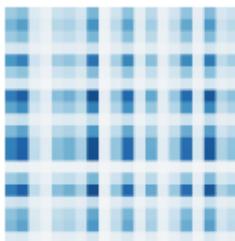


# Conclusions

## 1. Det Identities

Planar NF  
Sylvester NF  
...

Jacobian



(Low rank)

## 2. Coupling Blocks

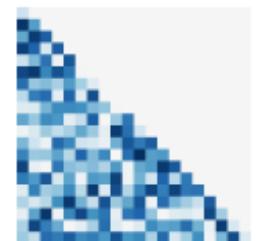
NICE  
Real NVP  
Glow  
...



(Lower triangular +  
structured)

## 3. Autoregressive

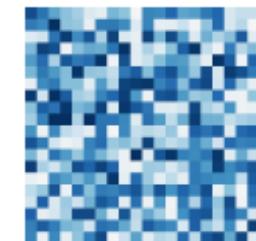
Inverse AF  
Neural AF  
Masked AF  
...



(Lower triangular)

## 4. Unbiased Estimation

FFJORD  
**Residual Flows**



(Arbitrary)