Lecture 5: Normalizing Flows

Deep Generative Models

Sajjad Amini

Department of Electrical Engineering
Sharif University of Technology

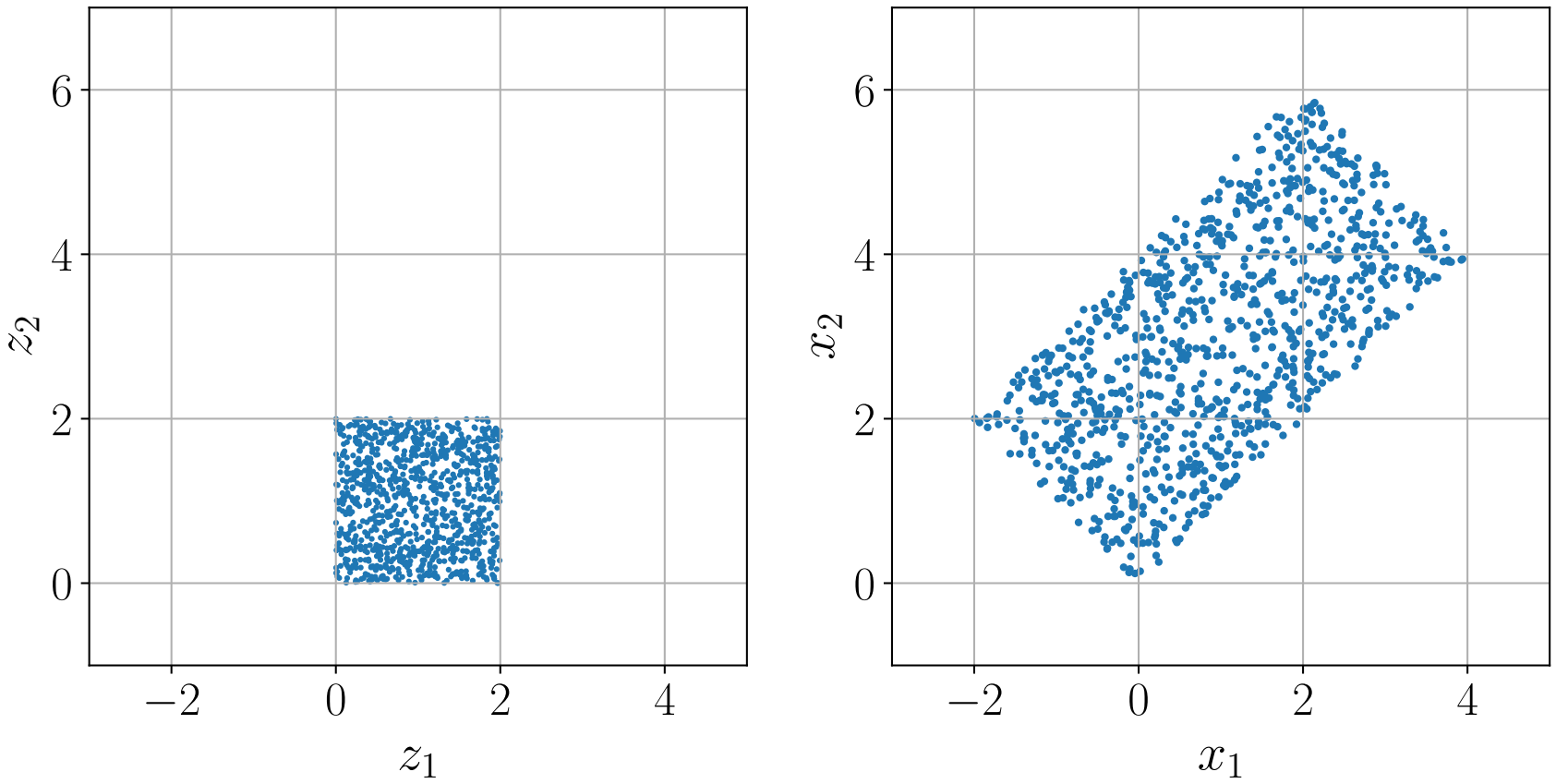
Contents

1. Intuition
2. Change of Variable Theorem
3. Normalizing Flow Models
4. Learning
5. Nonlinear Independent Component Estimation
6. Real-valued Non-volume Preserving Transformations
7. Masked Autoregressive Flows
8. Inverse Autoregressive Flows
9. Conclusion

Section 1

Intuition

Intuition

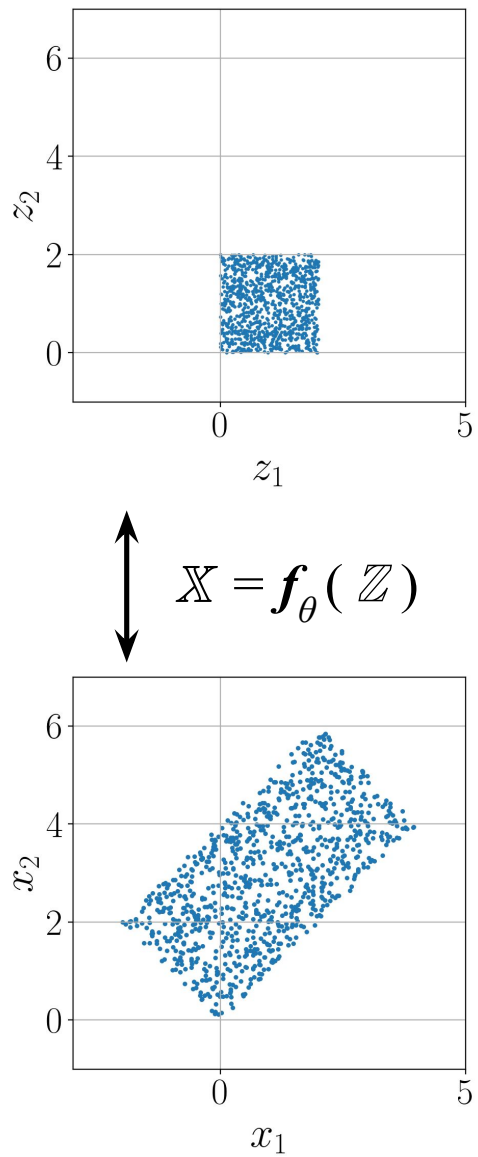


(a) Base random variable $p(\mathbb{Z})$

(b) Data Samples $p_{\text{data}}(\mathbb{X})$

Figure: The scenario: We have access to random variable $\mathbb{Z}$ with $p(\mathbb{Z})$ distribution while our target distribution is $p_{\text{data}}(\mathbb{X})$

General Idea



General Idea

To achieve the target distribution using the base one, we follow these steps:

- Design a parametric transfromation $\boldsymbol{f}_\theta(\cdot)$
- Calculate the $p_\theta(\boldsymbol{x})$
- Maximize the dataset likelihood to find the transformation parameters $\boldsymbol{\theta}$

☞ But, how can we find $p_\theta(\boldsymbol{x})$?

Change of Variable

Assume $\boldsymbol{z} \sim p(\mathbb{Z})$ and $\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{z})$. Given that the transformation $\boldsymbol{f}(\boldsymbol{z})$ is invertible, you can calculate the probability distribution $p(\mathbb{X})$.

Section 2

Change of Variable Theorem

Transformation [1]

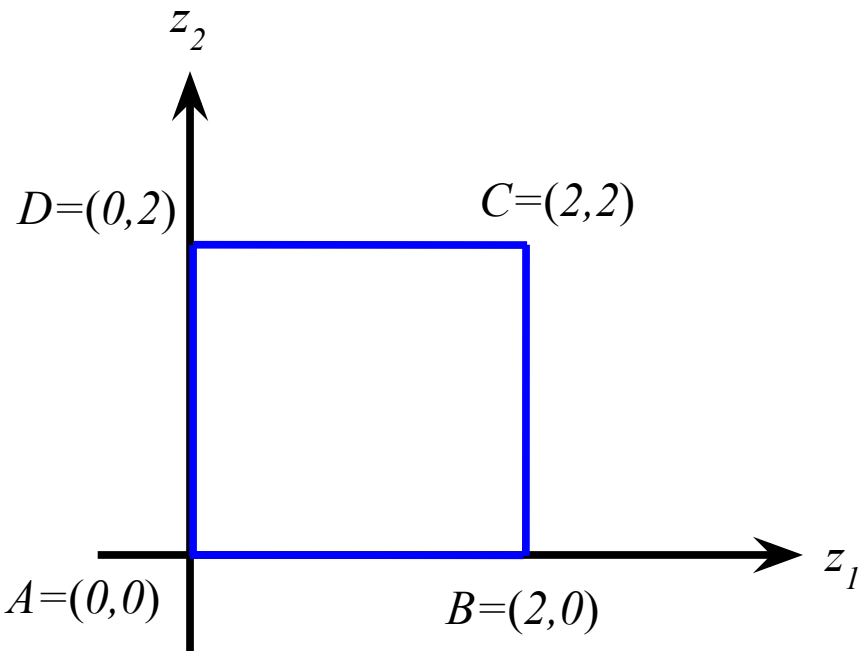


Figure: Base random variable

Transformation [1]

$$\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{z}) \Rightarrow \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2z_1 - z_2 \\ 2z_1 + z_2 \end{pmatrix}$$

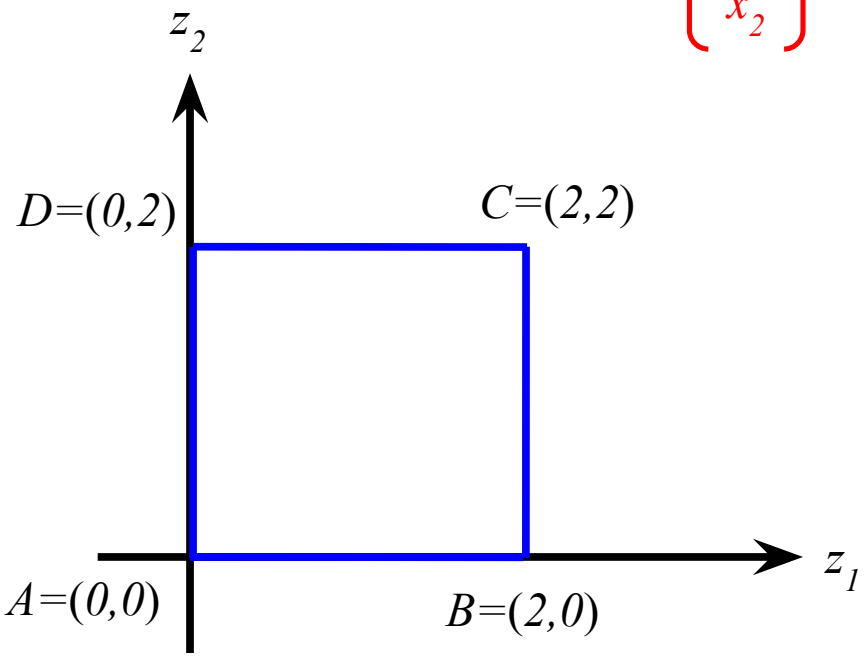


Figure: Transformation

Transformation [1]

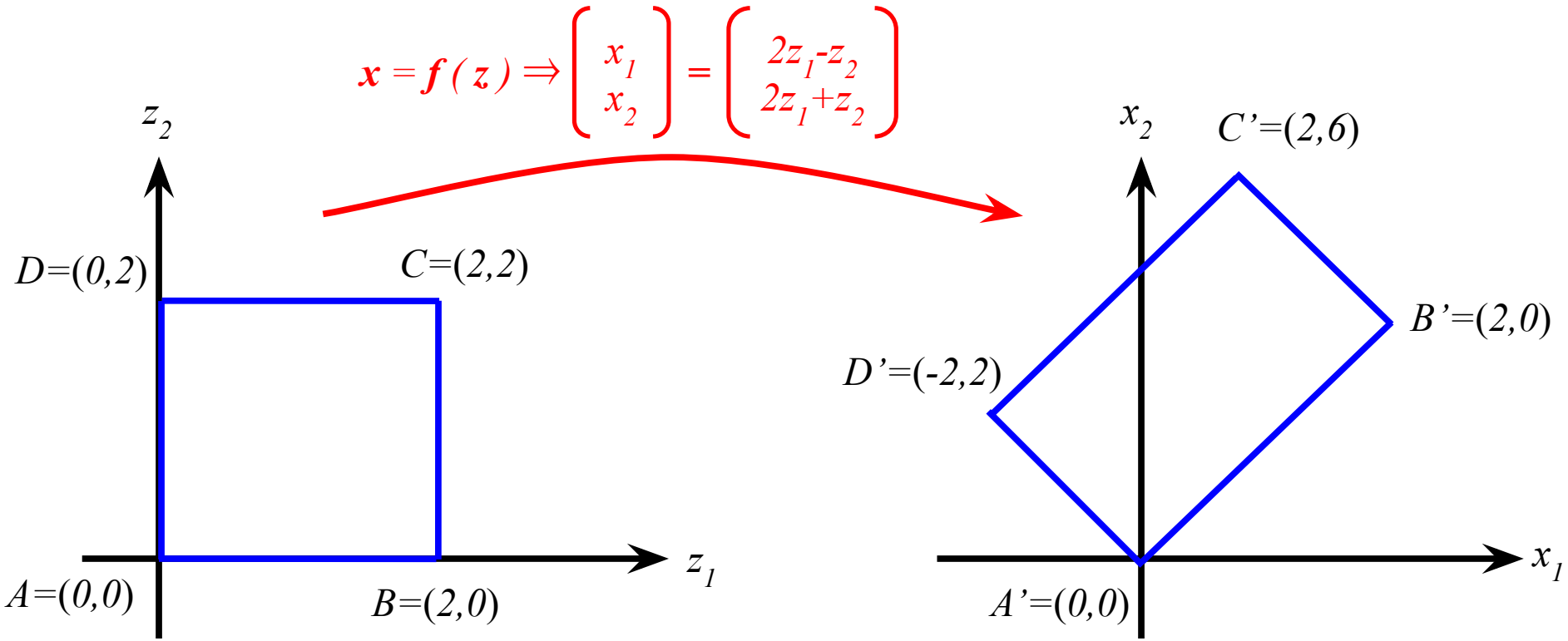


Figure: Transformed random variable

Correspondence

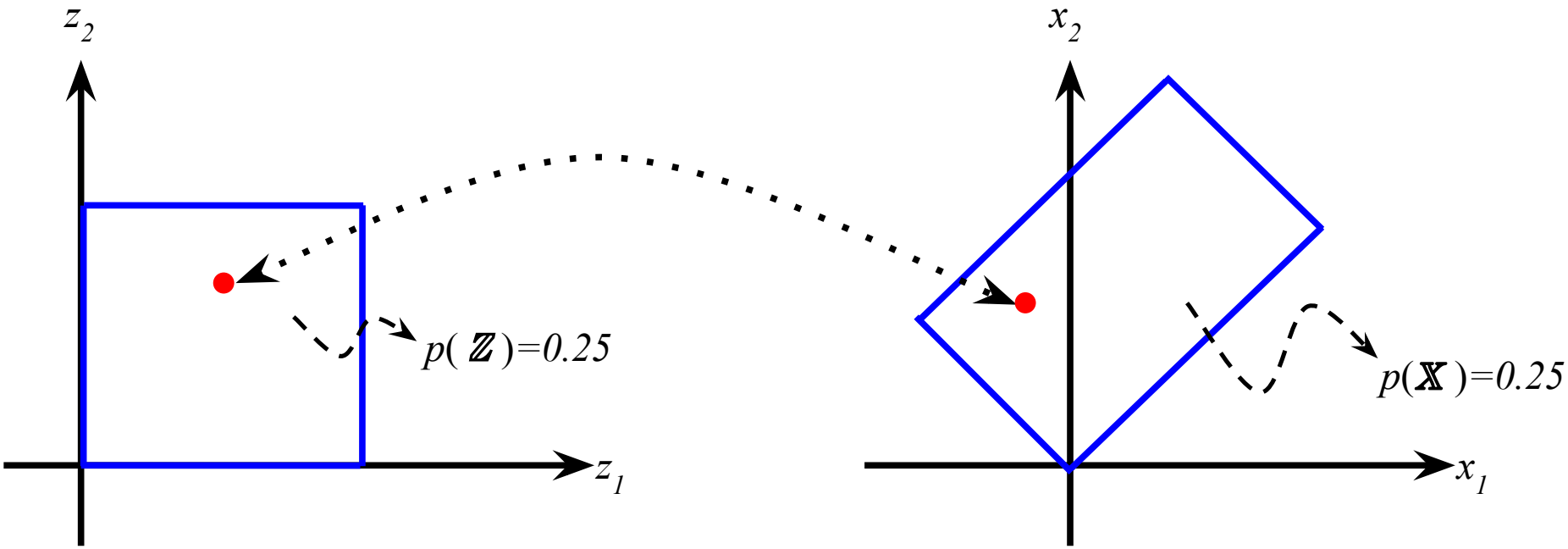
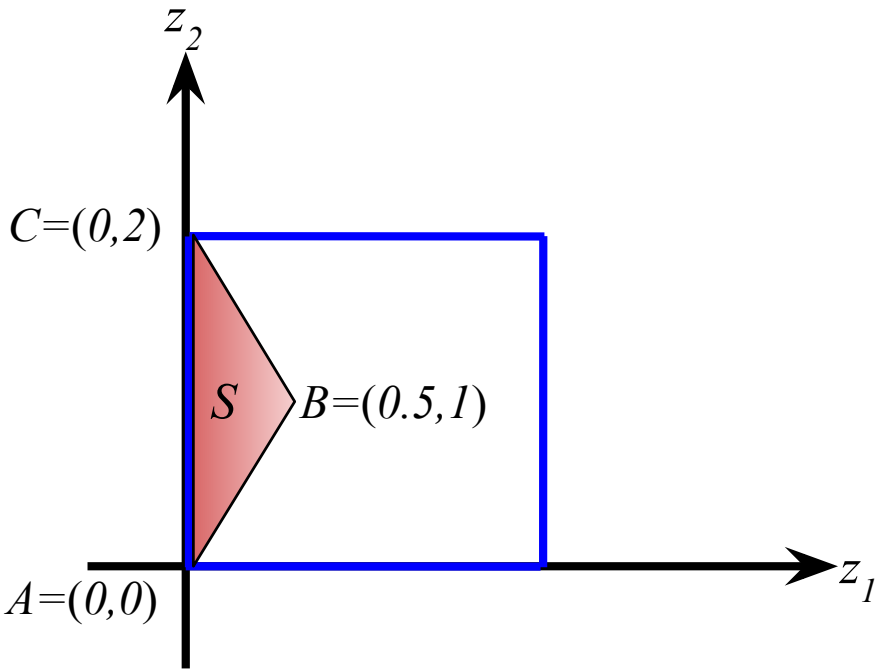
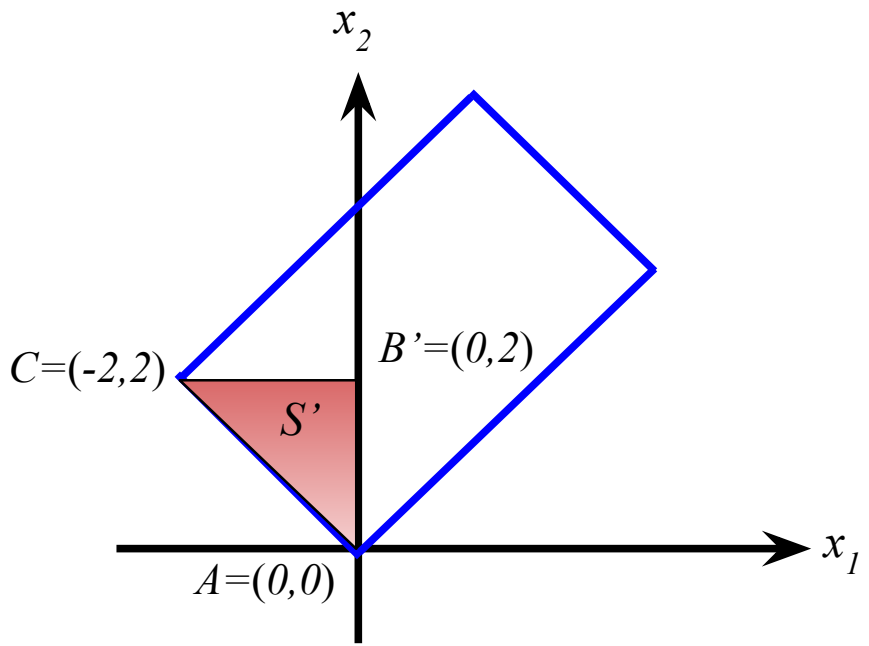


Figure: The point-to-point correspondence and probability density functions (we assume $p(\mathbb{Z}) = 0.25$ and conclude that $p(\mathbb{X}) = 0.25$ because of the point-to-point correspondence)

Equivalent Events



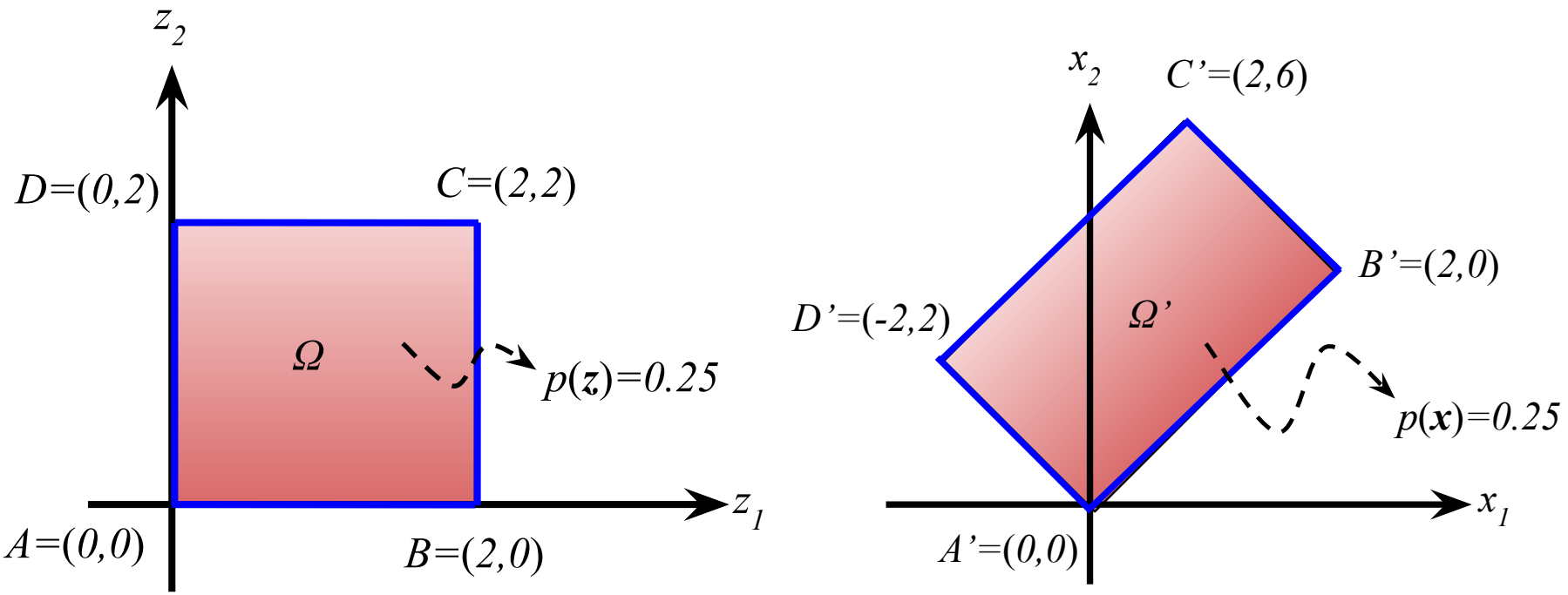
(a) $p(S) = \text{Area}_S \times p(\mathbb{Z}) = \frac{1}{2} \times \frac{1}{4} = \frac{1}{8}$



(b) $p(S') = \text{Area}_{S'} \times p(\mathbb{X}) = 2 \times \frac{1}{4} = \frac{1}{2}$

Figure: Probability calculations for equivalent events (while we expect $p(S) = p(S')$ because they are equivalent events, we see a contradiction)

Whole Sample Space Probability



(a) $p(\Omega) = \text{Area}_{\Omega} \times p(\mathbb{Z}) = 4 \times \frac{1}{4} = 1$

(b) $p(\Omega') = \text{Area}_{\Omega'} \times p(\mathbb{X}) = 16 \times \frac{1}{4} = 4$

Figure: Probability calculations for the whole sample space (while the Sample space must have probability 1, but for $p(\mathbb{X})$ this value is greater than one which shows a clear contradiction with probability axioms)

Change of Variable

Theorem

Assume $\mathbb{Z} \in \mathbb{R}^D$ is a random variable with base distribution $p_z(\mathbb{Z})$. Also assumes:

- $\mathbb{Z}$ is pass through an invertible function $\boldsymbol{f} : \mathbb{R}^D \to \mathbb{R}^D$ to generate random variable $\mathbb{X}$ with a more complex distribution.
- We denote the inverse function $\boldsymbol{f}^{-1}$ by $\boldsymbol{g}$ such that: $\begin{cases} \boldsymbol{x} = \boldsymbol{f}(\boldsymbol{z}) \\ \boldsymbol{z} = \boldsymbol{g}(\boldsymbol{x}) \end{cases}$

then:

$$p_x(\boldsymbol{x}) = p_z\left(\boldsymbol{g}(\boldsymbol{x})\right) \left| \det \boldsymbol{J}_g(\boldsymbol{x}) \right|$$

$$= p_z(\boldsymbol{z}) \left| \det \boldsymbol{J}_f(\boldsymbol{z}) \right|^{-1}$$

General Idea

Normalizing flow models approximate the data distribution by:

- Selecting a simple base distribution $p(\mathbb{Z})$ (e.g. standard Gaussian distribution)
- Selecting a flexible and *invertible* parameterized transformation $\boldsymbol{f}_\theta : \mathbb{R}^D \to \mathbb{R}^D$
- Assuming the observed random vector $\mathbb{X} = \boldsymbol{f}_\theta(\mathbb{Z})$
- Calculating the exact distribution over $\mathbb{X}$ using change of variable

General Idea

$$\boldsymbol{x} = \boldsymbol{f}_\theta(\boldsymbol{z}) \Rightarrow \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \theta_1 z_1 + \theta_2 z_2 \\ \theta_3 z_1 + \theta_4 z_2 \end{pmatrix}$$

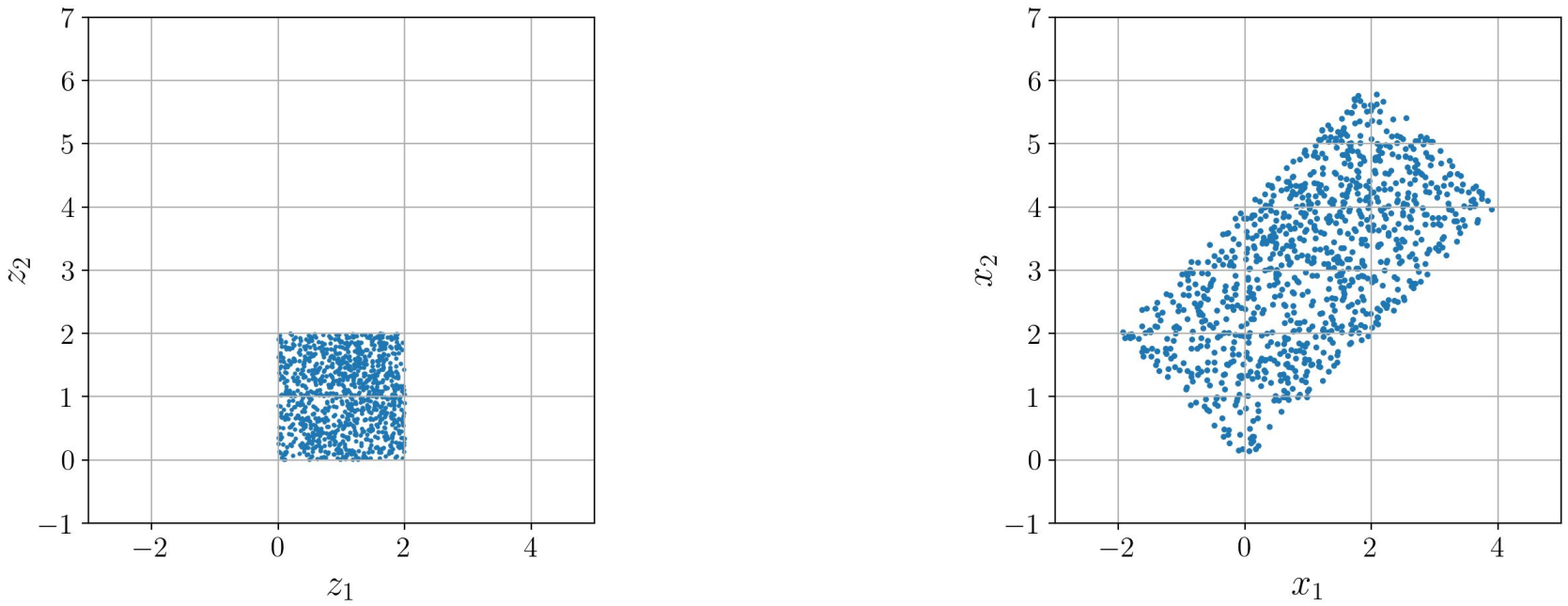


Figure: Sample of complex distribution using normalizing flow

Likelihood Calculation

Assume:

$$p(\mathbb{Z}) = \frac{1}{4}, \begin{cases} 0 \leq z_1 \leq 2 \\ 0 \leq z_2 \leq 2 \end{cases} \quad \text{and} \quad \boldsymbol{f}_\theta(\boldsymbol{z}) = \begin{bmatrix} \theta_1 z_1 + \theta_2 z_2 \\ \theta_3 z_1 + \theta_4 z_2 \end{bmatrix} = \begin{bmatrix} \theta_1 & \theta_2 \\ \theta_3 & \theta_4 \end{bmatrix} \overbrace{\begin{bmatrix} z_1 \\ z_2 \end{bmatrix}}^{\boldsymbol{z}}$$

Then:

$$p_\theta(\boldsymbol{x}) = \frac{p(\boldsymbol{z})}{|\det \boldsymbol{J}_f(\boldsymbol{u})|} = \frac{\frac{1}{4}}{\left|\det \begin{bmatrix} \theta_1 & \theta_2 \\ \theta_3 & \theta_4 \end{bmatrix}\right|} = \frac{\frac{1}{4}}{|\theta_1\theta_4 - \theta_2\theta_3|}$$

Note that the range for x_1 and x_2 is defined based on the inverse transformation and range for z_1 and z_2.

Transformation [1]

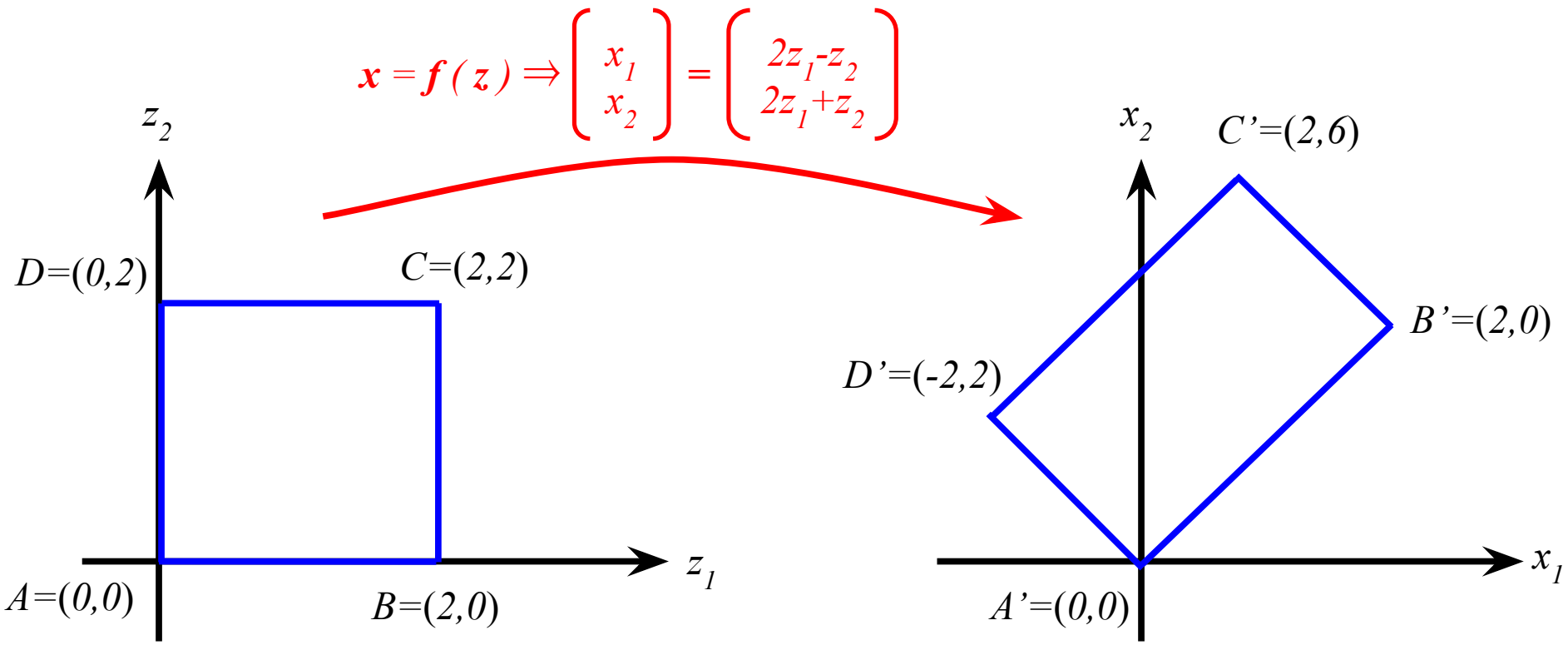
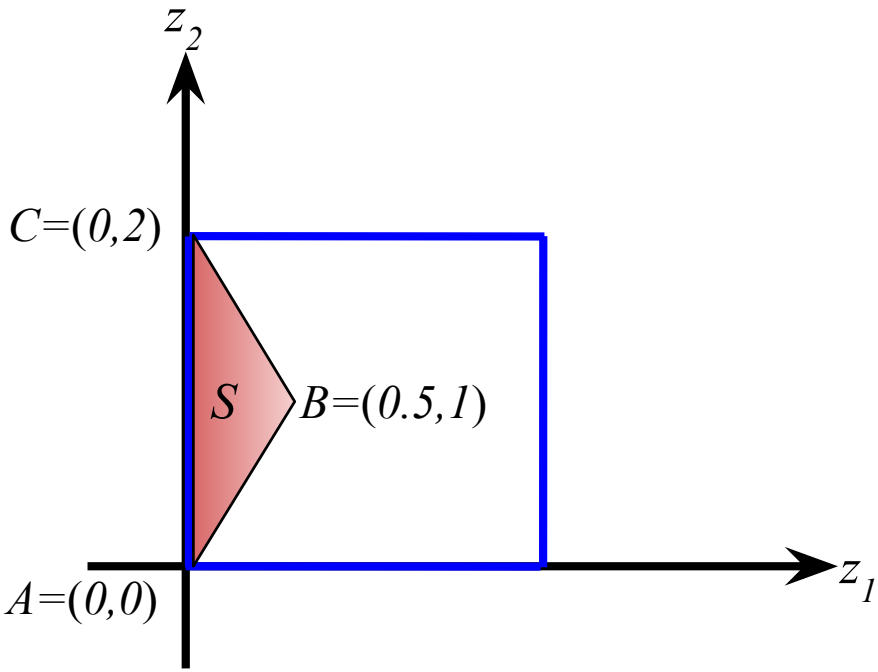


Figure: Sample transformation

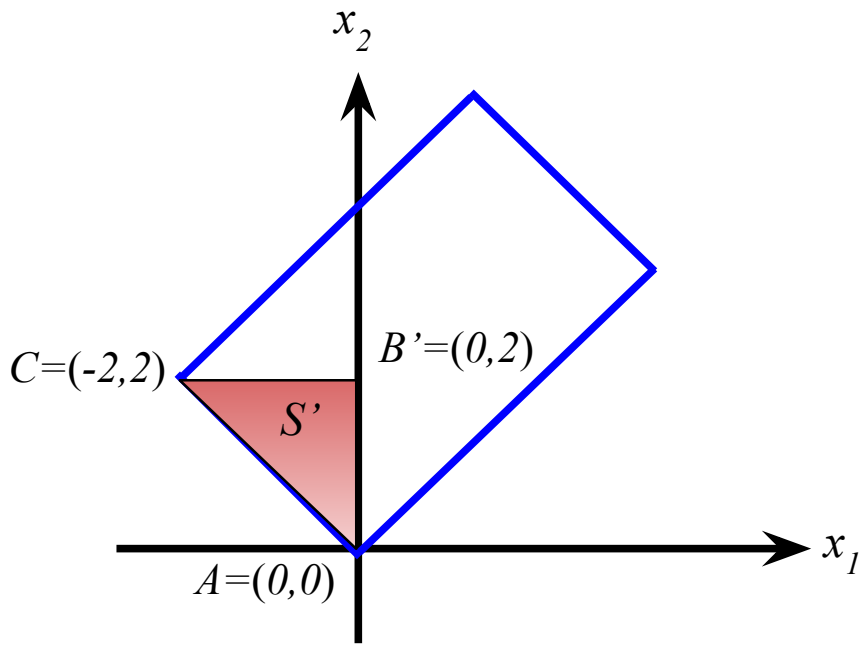
Probability Density Estimation

In the above example $\begin{bmatrix} \theta_1 & \theta_2 \\ \theta_3 & \theta_4 \end{bmatrix} = \begin{bmatrix} 2 & -1 \\ 2 & 1 \end{bmatrix}$, thus $p_\theta(\mathbb{X}) = \frac{\frac{1}{4}}{4} = \frac{1}{16}$

Equivalent Events (Revisited)



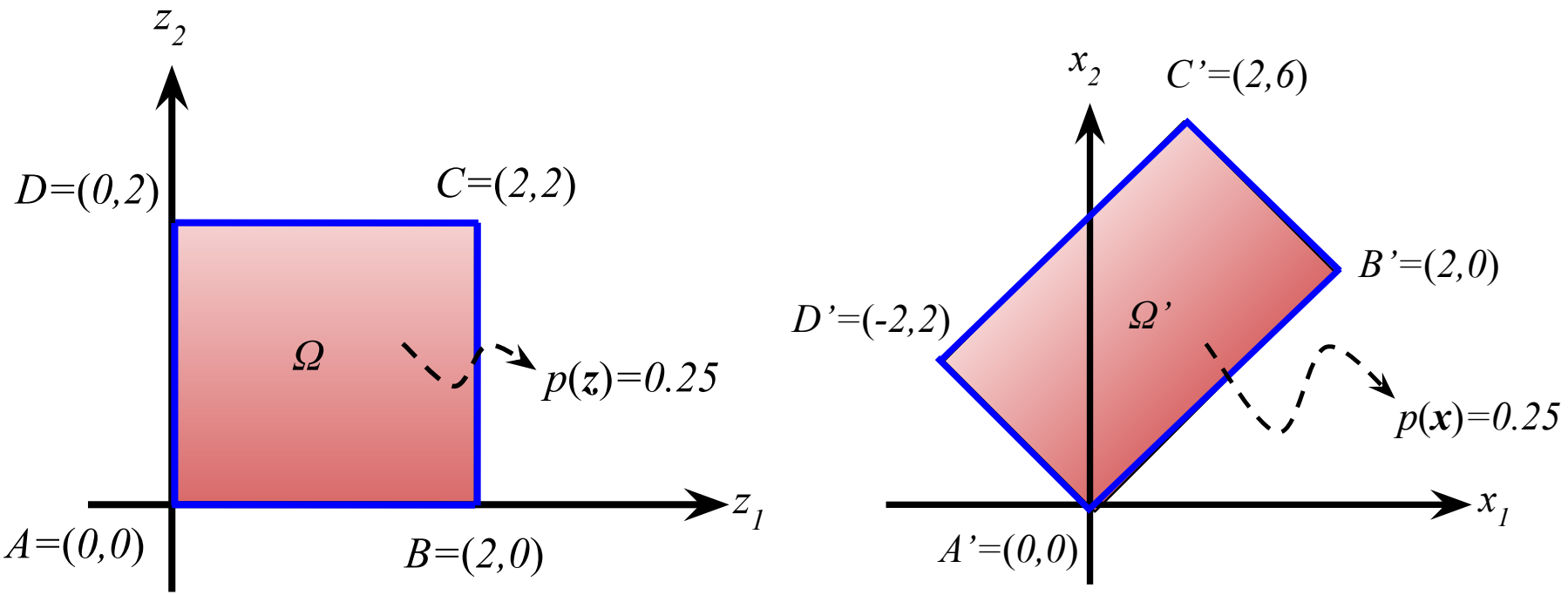
(a) $p(S) = \text{Area}_S \times p(\mathbb{Z}) = \frac{1}{2} \times \frac{1}{4} = \frac{1}{8}$



(b) $p(S') = \text{Area}_{S'} \times p(\mathbb{X}) = 2 \times \frac{1}{16} = \frac{1}{8}$

Figure: Probability calculations for equivalent events (Using change of variable technique, we see the probabilities are equal)

Whole Sample Space Probability (Revisited)



(a) $p(\Omega) = \text{Area}_{\Omega} \times p(\mathbb{Z}) = 4 \times \frac{1}{4} = 1$ (b) $p(\Omega') = \text{Area}_{\Omega'} \times p(\mathbb{X}) = 16 \times \frac{1}{16} = 1$

Figure: Probability calculations for the whole sample space (Using change of variable technique, we see the probability for the sample space in $\mathbb{X}$ equals to one as expected)

Section 3

Normalizing Flow Models

Normalizing Flow

Definition

In normalizing flow, we have the following building blocks:

- Assume $\boldsymbol{z} \sim p(\mathbb{Z})$, where $p(\mathbb{Z})$ is an ***easy to sample*** base distribution.
- We define a flexible parameterized transformation $\boldsymbol{f}_\theta$ from base distribution $p(\mathbb{Z})$ to estimate data distribution by $p_\theta(\mathbb{X})$
- We optimize the transformation parameters by:

$$\boldsymbol{\theta}^\star = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \mathrm{KL}\left(p_{\text{data}}(\mathbb{X}) \| p_\theta(\mathbb{X})\right) \equiv \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\mathbb{X})}\left[p_\theta(\boldsymbol{x})\right]$$

Normalizing Flow vs VAE

Comparison to VAE

- In contrast to VAE, here we can easily calculate the exact data likelihood as:

$$
\begin{aligned}
p_\theta(\boldsymbol{x}_i) &= p\Big(\boldsymbol{g}_\theta(\boldsymbol{x}_i)\Big)\Big|\det \boldsymbol{J}_{\boldsymbol{g}}(\boldsymbol{x}_i)\Big| \\
&= \frac{p(\boldsymbol{z}_i)}{\Big|\det \boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{z}_i)\Big|}, \text{where} \boldsymbol{z}_i = \boldsymbol{g}(\boldsymbol{x}_i)
\end{aligned}
$$

- Due to the invertibility of normalizing flow transformation, the latent vector $\mathbb{Z}$ and observed vector $\mathbb{X}$ are of the same dimension, thus:
  - There is no abstraction in the latent space.
  - Finding meaningful direction in the latent space is not possible.

Single Invertible Transformation

Forward Transformation

Assume $\boldsymbol{f}_\theta : \mathbb{R}^D \to \mathbb{R}^D$ is an invertible transformation defined as:

$$\boldsymbol{x} = \boldsymbol{f}_\theta(\boldsymbol{z}) \Rightarrow \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_D \end{bmatrix} = \boldsymbol{f}_\theta \left(\begin{bmatrix} z_1 \\ z_2 \\ \dots \\ z_D \end{bmatrix} \right) \Rightarrow \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_D \end{bmatrix} = \begin{bmatrix} u_1(\boldsymbol{z}) \\ u_2(\boldsymbol{z}) \\ \dots \\ u_D(\boldsymbol{z}) \end{bmatrix}$$

then the Jacobian matrix is:

$$\boldsymbol{J}_{\boldsymbol{f}} = \begin{bmatrix} \frac{\partial u_1}{\partial z_1} & \dots & \frac{\partial u_1}{\partial z_D} \\ \vdots & \vdots & \vdots \\ \frac{\partial u_D}{\partial z_1} & \dots & \frac{\partial u_D}{\partial z_D} \end{bmatrix}$$

and we can calculate the data likelihood as:

$$p_\theta(\boldsymbol{x}) = p\left(\boldsymbol{g}(\boldsymbol{x})\right) \left| \det \boldsymbol{J}_f(\boldsymbol{z}) \right|^{-1}$$

Inverse Transformation

Now assume the inverse transformation $\boldsymbol{g}_\theta = \boldsymbol{f}_\theta^{-1}$ defined as:

$$\boldsymbol{z} = \boldsymbol{g}_\theta(\boldsymbol{x}) \Rightarrow \begin{bmatrix} z_1 \\ z_2 \\ \dots \\ z_D \end{bmatrix} = \boldsymbol{g}_\theta \left(\begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_D \end{bmatrix} \right) \Rightarrow \begin{bmatrix} z_1 \\ z_2 \\ \dots \\ z_D \end{bmatrix} = \begin{bmatrix} v_1(\boldsymbol{x}) \\ v_2(\boldsymbol{x}) \\ \dots \\ v_D(\boldsymbol{x}) \end{bmatrix}$$

then the Jacobian matrix is:

$$\boldsymbol{J_g} = \begin{bmatrix} \frac{\partial v_1}{\partial x_1} & \dots & \frac{\partial v_1}{\partial x_D} \\ \vdots & \vdots & \vdots \\ \frac{\partial v_D}{\partial x_1} & \dots & \frac{\partial v_D}{\partial x_D} \end{bmatrix}$$

and we can calculate the data likelihood as:

$$p_\theta(\boldsymbol{x}) = p\left(\boldsymbol{g}(\boldsymbol{x})\right) \left| \det \boldsymbol{J}_g(\boldsymbol{x}) \right|$$

Composite Transformations

Forward Transformation

To have more flexibility in Normalizing flow, the transformation is realized using L layers of invertible transformations $\boldsymbol{f}_i, i = 1, \dots L$ as:

$$\boldsymbol{f}_\theta = \boldsymbol{f}_{L,\theta_L} \circ \boldsymbol{f}_{L-1,\theta_{L-1}} \circ \dots \circ \boldsymbol{f}_{2,\theta_2} \circ \boldsymbol{f}_{1,\theta_1}$$

where $\boldsymbol{f}_{i,\theta_i} : \mathbb{R}^D \to \mathbb{R}^D$ for $1 \leq i \leq L$ and θ_i represens the parameters for the i-th transformation.

- ☞ We omit θ_i subscript for simplicity.

Composite Transformations

Inverse Transformation

Assume the composite forward transformation in Slide 23 as :

$$\boldsymbol{f}_\theta = \boldsymbol{f}_L \circ \boldsymbol{f}_{L-1} \circ \ldots \circ \boldsymbol{f}_2 \circ \boldsymbol{f}_1$$

We know each transformation $\boldsymbol{f}_i$ is invertible, denoted by $\boldsymbol{g}_i$, then the inverse transformation can also be represented in a compositional way as:

$$\boldsymbol{g}_\theta = \boldsymbol{g}_1 \circ \boldsymbol{g}_2 \circ \ldots \circ \boldsymbol{g}_{L-1} \circ \boldsymbol{g}_L$$

☞ Note that the sequence is reversed in the inverse transformation.

Intermdeiate Random Variables

Assume $\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{z})$, then we name the intermediate variables as:

$$\boldsymbol{x} = \overbrace{\boldsymbol{f}_L \circ \boldsymbol{f}_{L-1} \circ \ldots \circ \underbrace{\boldsymbol{f}_2 \circ \overbrace{\boldsymbol{f}_1(\boldsymbol{z})}^{\boldsymbol{z}(1)}}_{\boldsymbol{z}(2)}}^{\boldsymbol{z}(L-1)}$$

If we define $\boldsymbol{z}(L) \triangleq \boldsymbol{x}$ and $\boldsymbol{z}(0) \triangleq \boldsymbol{z}$, then we have:

$$\boldsymbol{z}(l) = \boldsymbol{f}_l\big(\boldsymbol{z}(l-1)\big), \text{ for } l = 1, \ldots, L$$

Equivalently using the inverse transformation we have:

$$\boldsymbol{z}(l-1) = \boldsymbol{g}_l\big(\boldsymbol{z}(l)\big), \text{ for } i = 1 \ldots, L$$

Jacobian of Composite

Forward Transformation

As we see before, we have:

$$\boldsymbol{f}_\theta = \boldsymbol{f}_L \circ \boldsymbol{f}_{L-1} \circ \ldots \circ \boldsymbol{f}_2 \circ \boldsymbol{f}_1$$

Using the chain rule, we can easily show that:

$$\boldsymbol{J}_f(\boldsymbol{z}) = \boldsymbol{J}_{f_L}\Big(\boldsymbol{z}(L-1)\Big) \times \boldsymbol{J}_{f_{L-1}}\Big(\boldsymbol{z}(L-2)\Big) \times \ \ldots \times \boldsymbol{J}_{f_2}\Big(\boldsymbol{z}(1)\Big) \times \boldsymbol{J}_{f_1}\Big(\boldsymbol{z}\Big)$$

Having access to the Jacobian of forward transformation, you can calculate the likelihood as:

$$p_\theta(\boldsymbol{x}) = p\big(\boldsymbol{g}_\theta(\boldsymbol{x})\big)\Big|\det \boldsymbol{J}_f(\boldsymbol{z})\Big|^{-1}$$

Jacobian of Composite

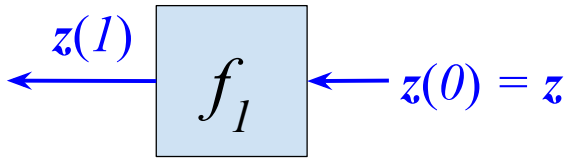


Figure: Base random variable

Jacobian of Composite

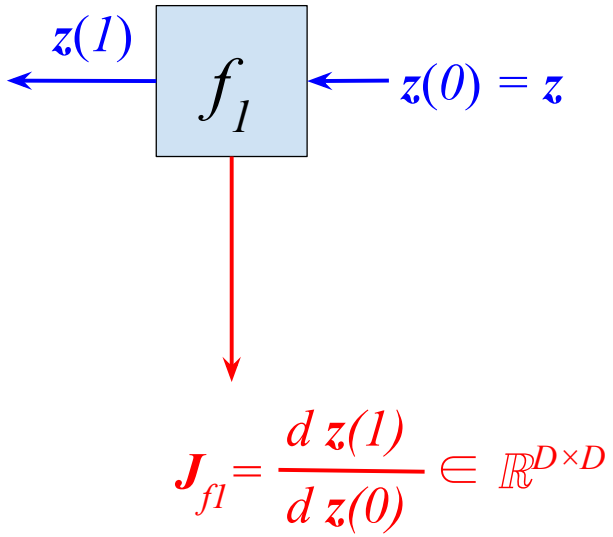


Figure: Transformation

Jacobian of Composite

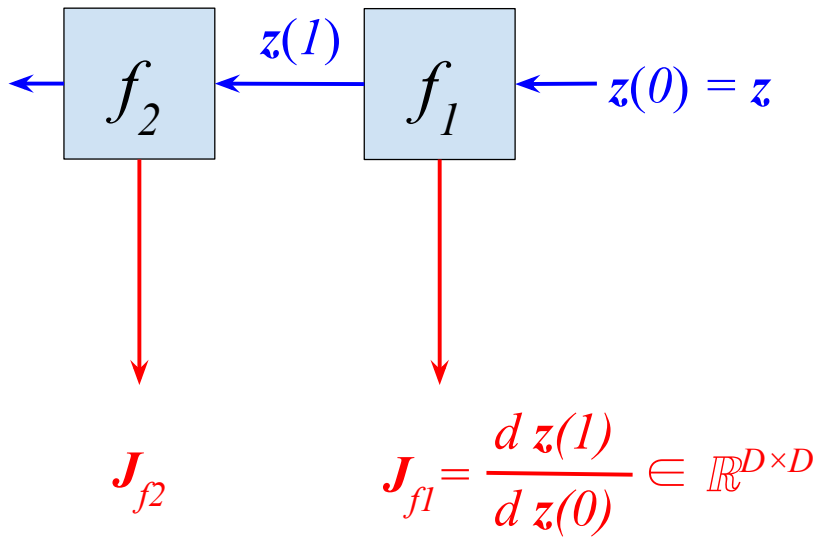


Figure: Transformed random variable

Jacobian of Composite

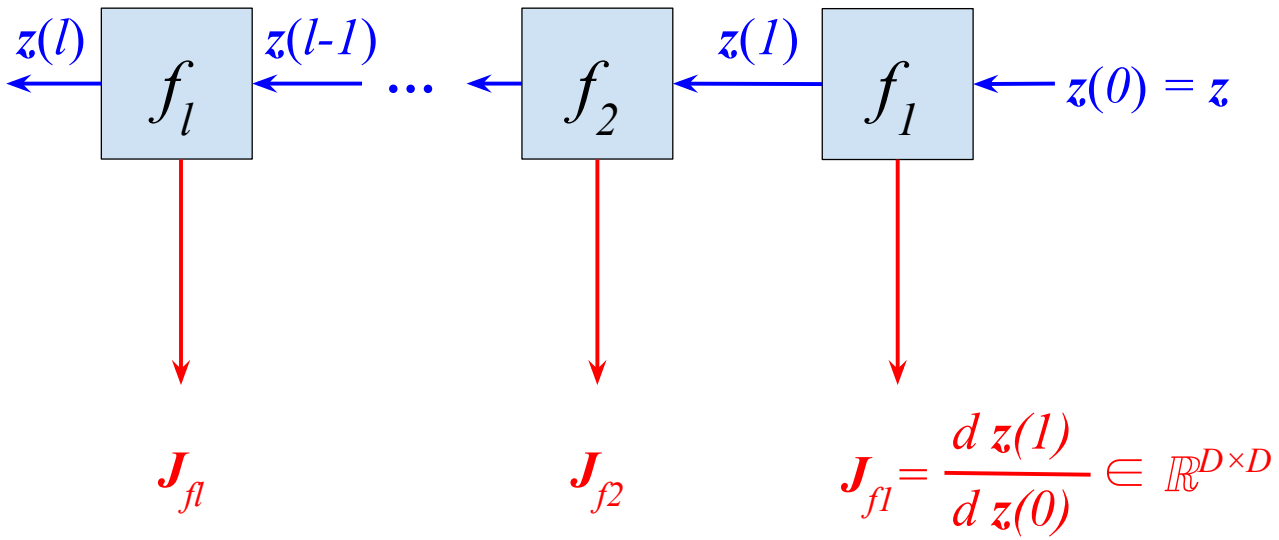


Figure: Transformed random variable

Jacobian of Composite

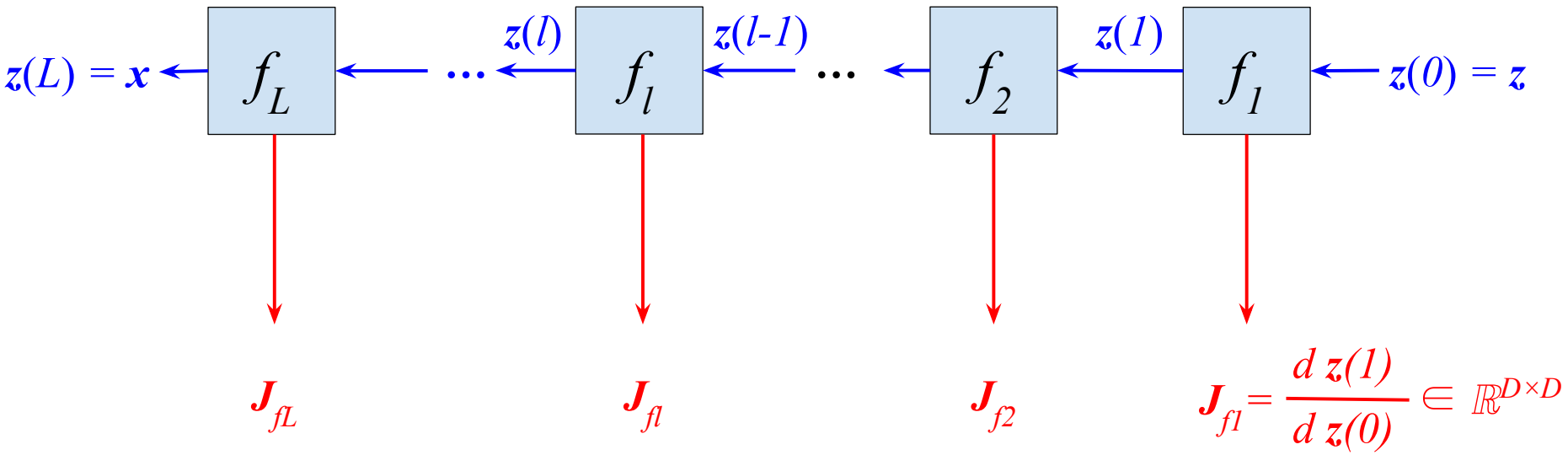


Figure: Transformed random variable

Jacobian of Composite

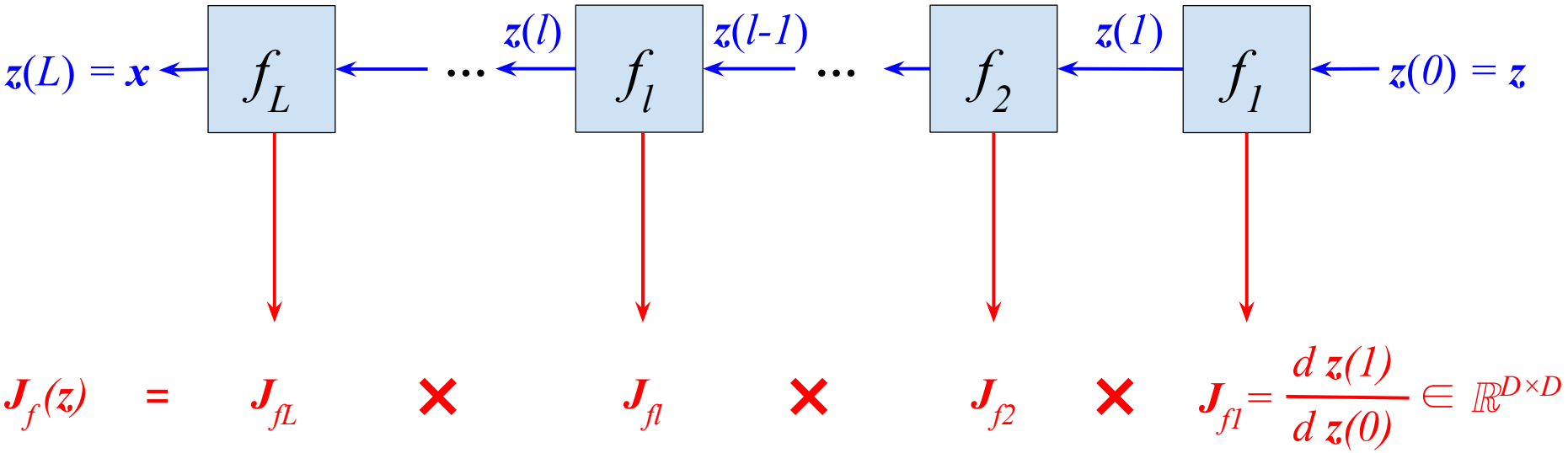


Figure: Transformed random variable

Jacobian of Composite

Inverse Transformation

Similarly to the forward composite transformation, we have:

$$\boldsymbol{g}_\theta = \boldsymbol{g}_1 \circ \boldsymbol{g}_2 \circ \ldots \circ \boldsymbol{g}_{L-1} \circ \boldsymbol{g}_L$$

Using the chain rule, we can easily show that:

$$\boldsymbol{J}_g(\boldsymbol{x}) = \boldsymbol{J}_{g_1}\Big(\boldsymbol{z}(1)\Big) \times \boldsymbol{J}_{g_2}\Big(\boldsymbol{z}(2)\Big) \times \ \ldots \times \boldsymbol{J}_{g_{L-1}}\Big(\boldsymbol{z}(L-1)\Big) \times \boldsymbol{J}_{g_L}\Big(\boldsymbol{x}\Big)$$

Having access to the Jacobian of inverse transformation, you can calculate the likelihood as:

$$p_\theta(\boldsymbol{x}) = p\left(\boldsymbol{g}(\boldsymbol{x})\right) \Big| \det \boldsymbol{J}_g(\boldsymbol{x}) \Big|$$

Jacobian of Composite

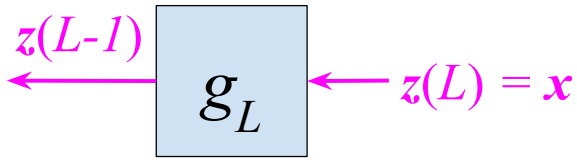


Figure: Base random variable

Jacobian of Composite

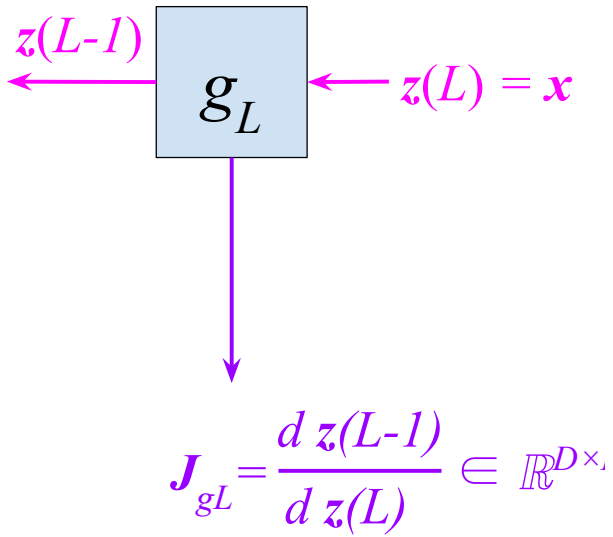


Figure: Transformation

Jacobian of Composite

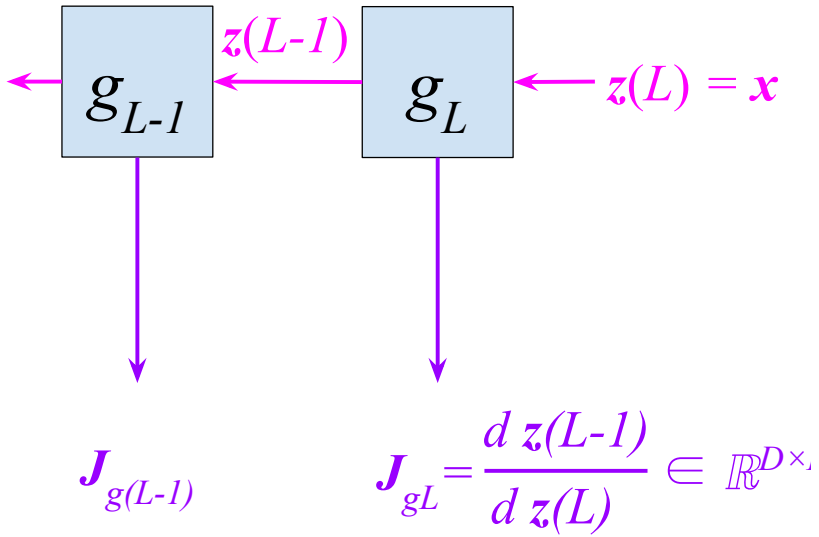


Figure: Transformed random variable

Jacobian of Composite

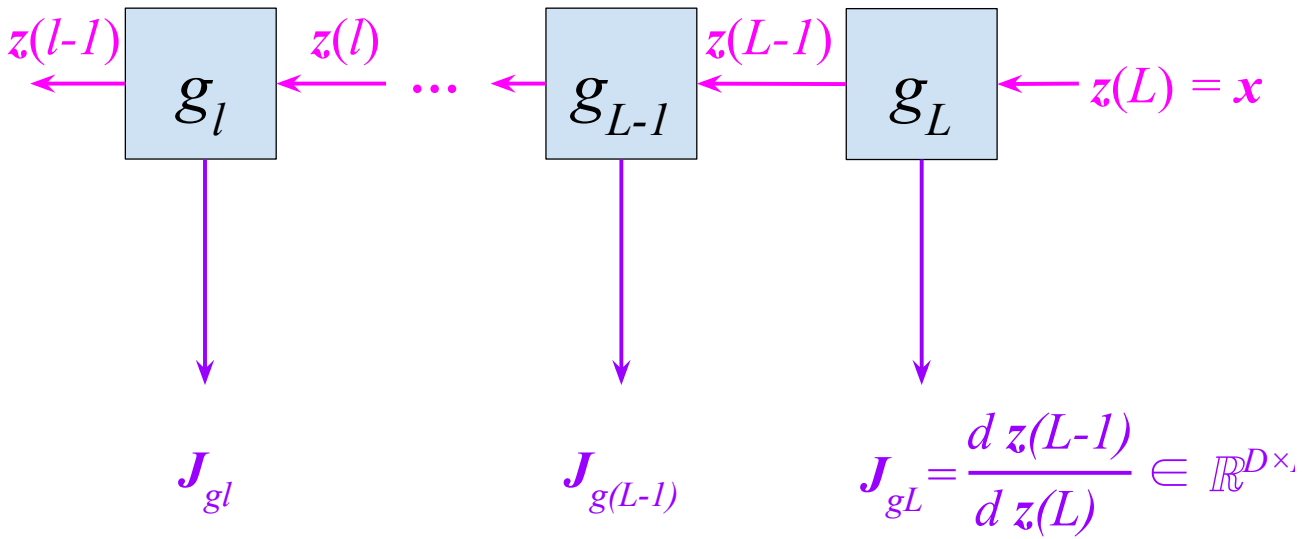


Figure: Transformed random variable

Jacobian of Composite

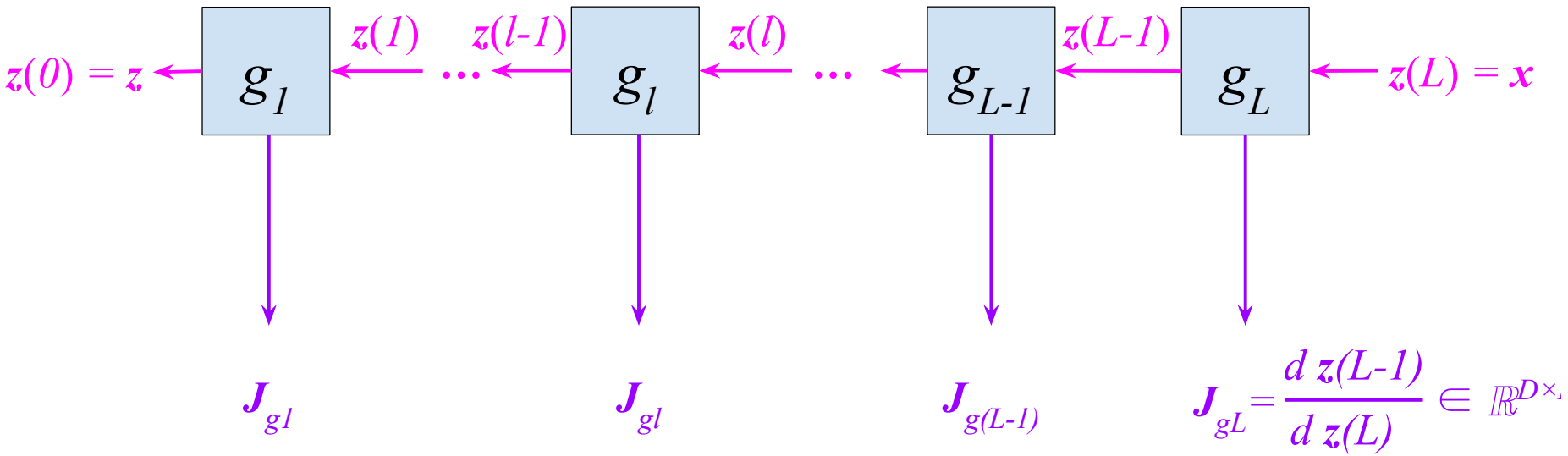


Figure: Transformed random variable

Jacobian of Composite

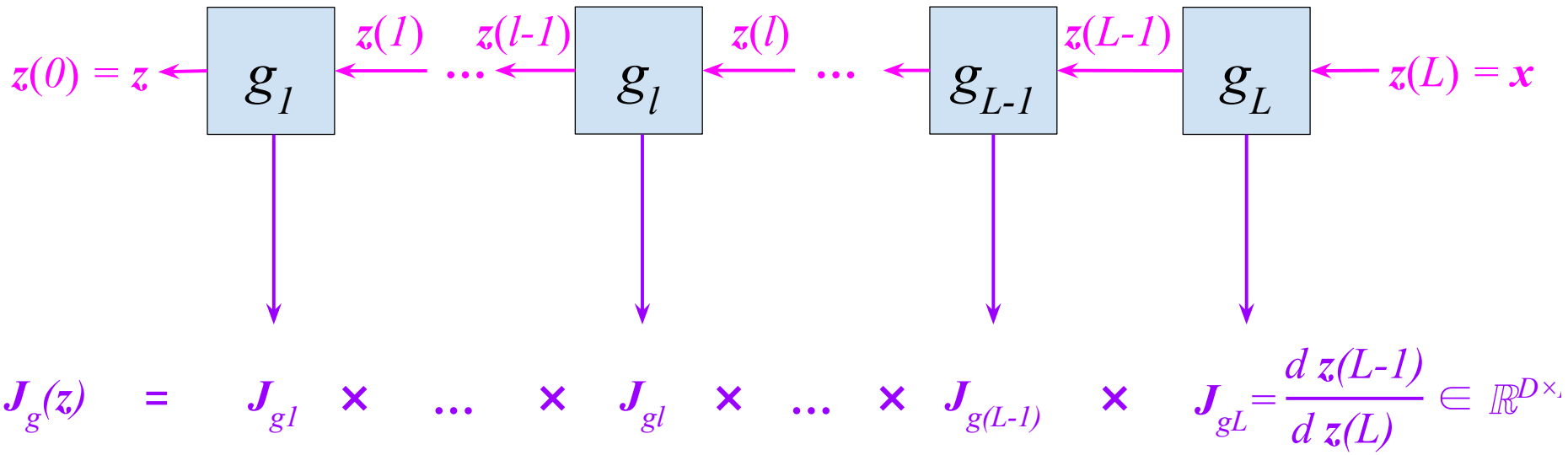


Figure: Transformed random variable

Why Normalizing Flow?

Why Flow?

The random variable $\mathbb{Z}$ with easy-to-sample distribution $p(\mathbb{Z})$ **flow** through L invertible transformations $\boldsymbol{f}_l, l = 1 \dots L$ to generate the random variable $\mathbb{X}$ with parametric distribution $p_\theta(\mathbb{X})$.

Why Normalizing?

As we see before, the distribution over $\mathbb{X}$ can be calculated as:

$$p_\theta(\boldsymbol{x}_i) = \frac{p(\boldsymbol{g}_\theta(\boldsymbol{x}_i))}{\left|\det \boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{z}_i)\right|}$$

in other word the term $\left|\det \boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{z}_i)\right|$ *Normalizes* $p(\boldsymbol{g}_\theta(\boldsymbol{x}_i))$ to be a valid probability distribution function.

Generative Modeling Tasks: Generation

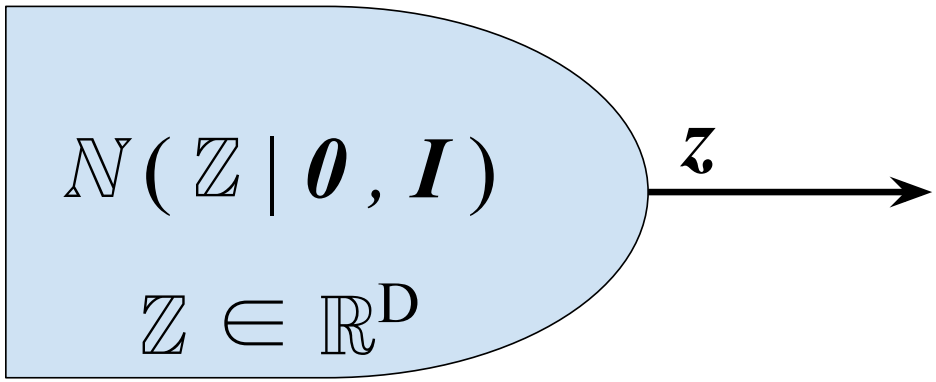


Figure: Sampling base distribution

Generative Modeling Tasks: Generation

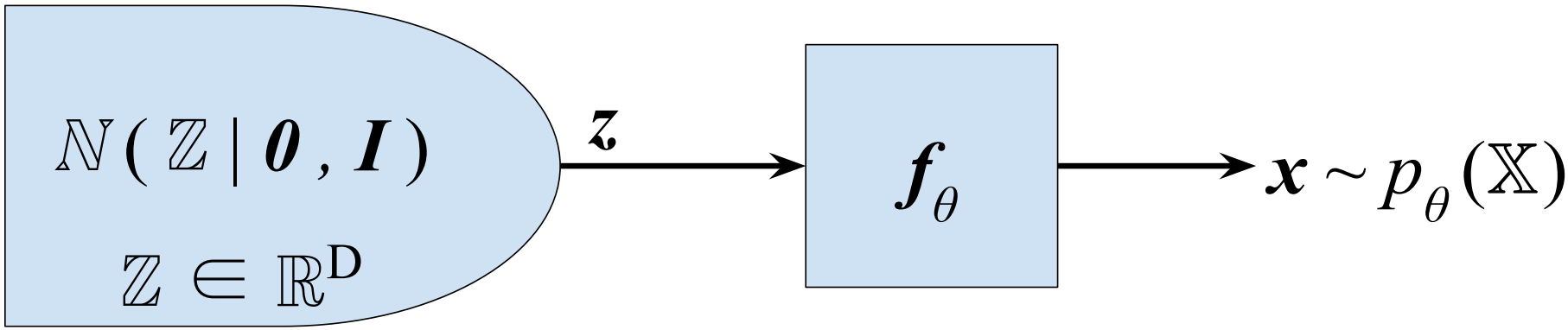


Figure: Passing generated sample through forward transformation

Generative Modeling Tasks: Density Estimation

$\boldsymbol{x}$

Figure: Input observed vectore $\boldsymbol{x}$

Generative Modeling Tasks: Density Estimation

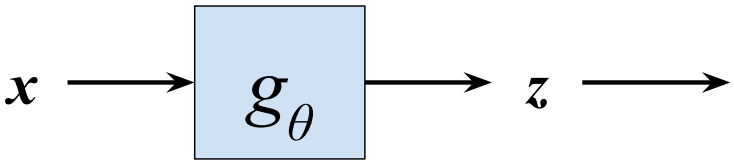


Figure: Corresponding latent vector $\boldsymbol{z}$

Generative Modeling Tasks: Density Estimation

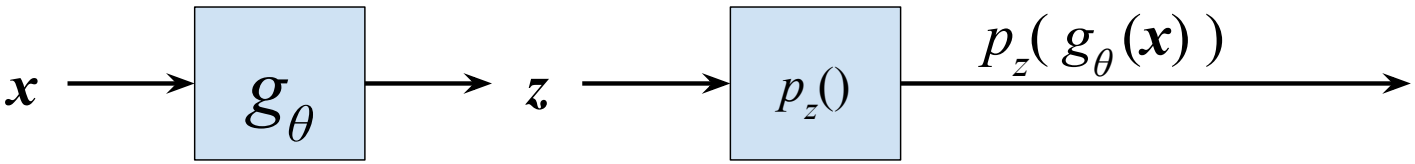


Figure: Corresponding latent vector $\boldsymbol{z}$

Generative Modeling Tasks: Density Estimation

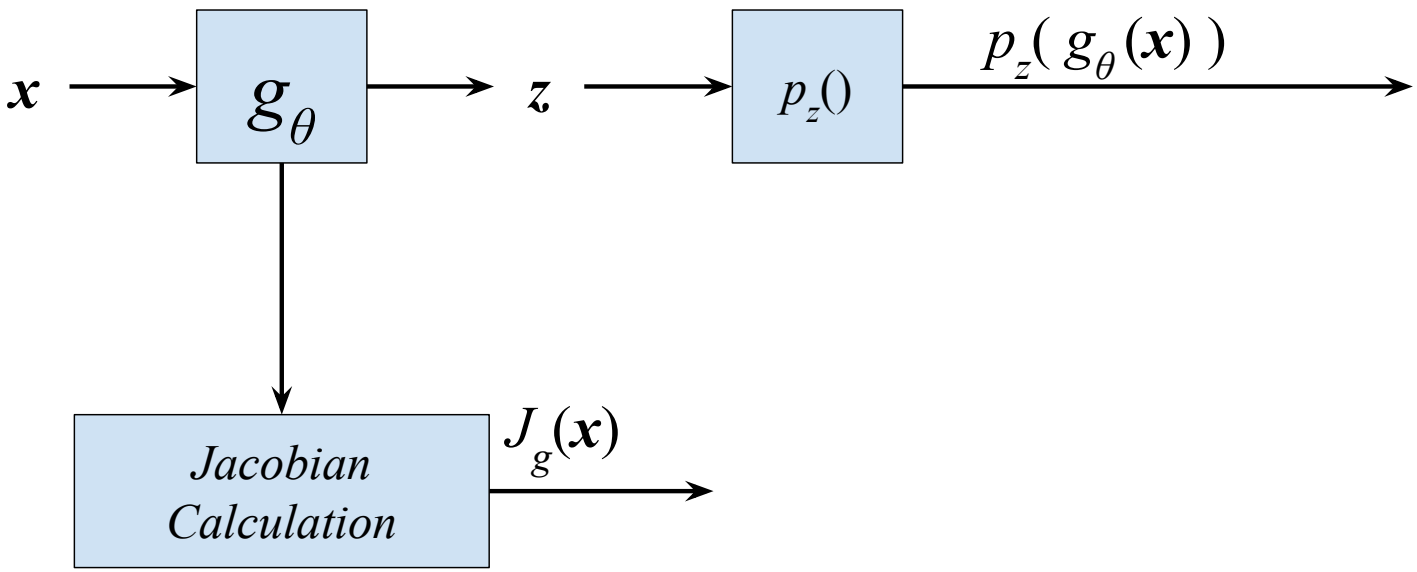


Figure: Jacobian calculations

Generative Modeling Tasks: Density Estimation

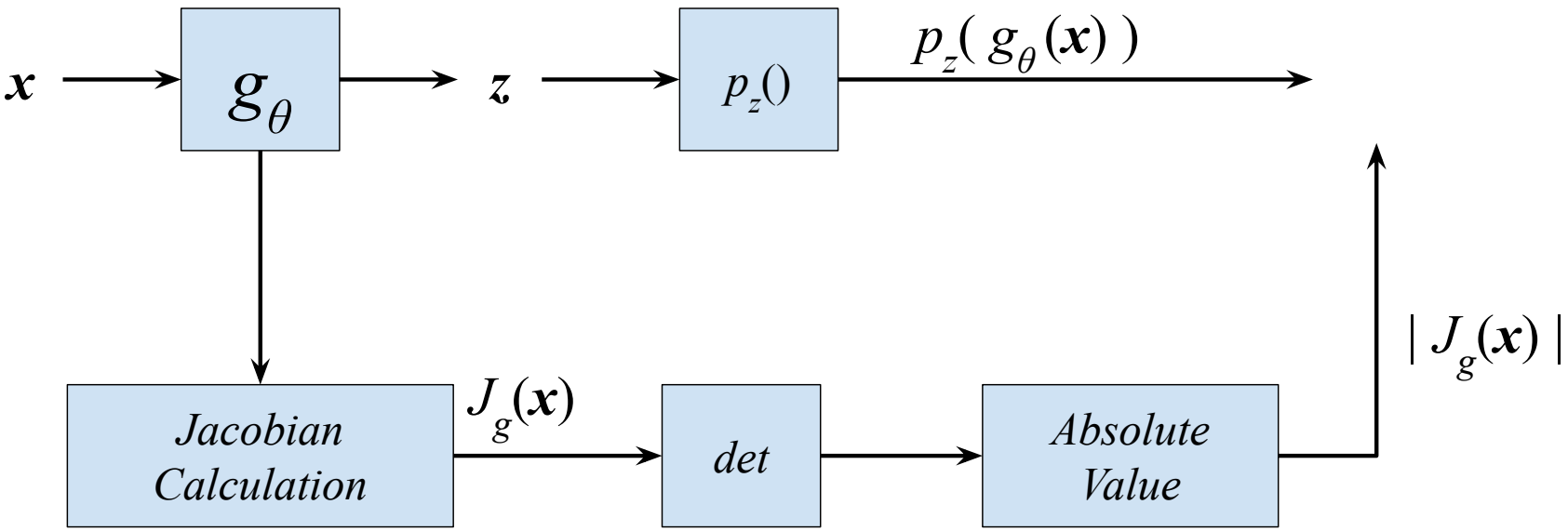


Figure: Calculating the normalization value

Generative Modeling Tasks: Density Estimation

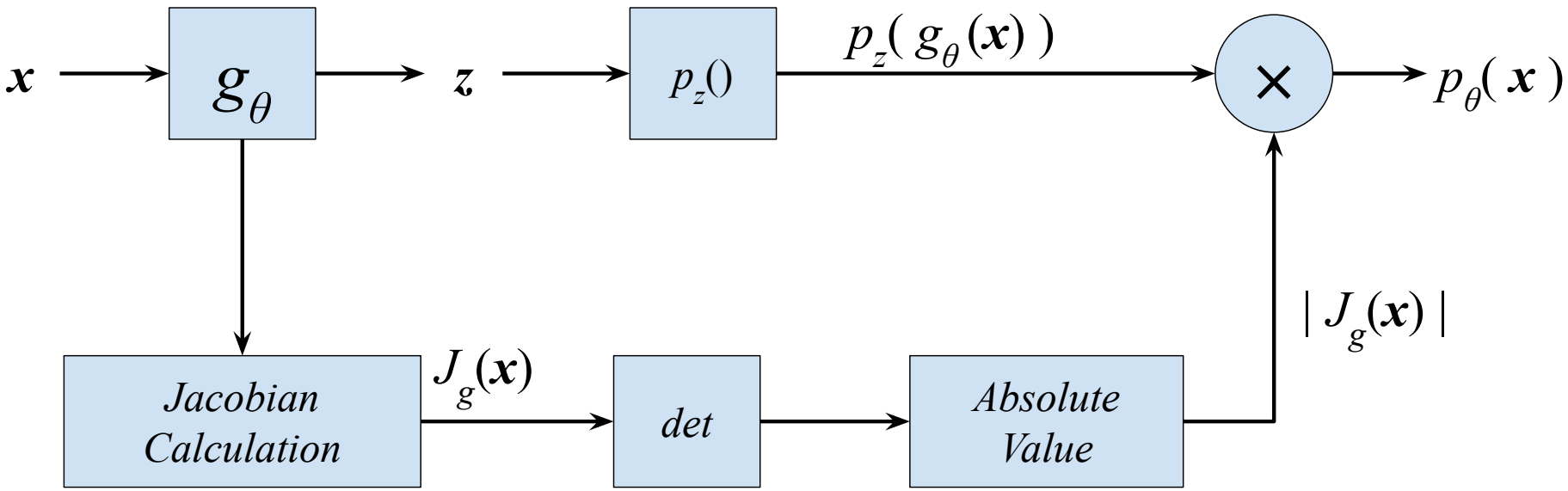


Figure: Density estimation

Generative Modeling Tasks: Latent Vector

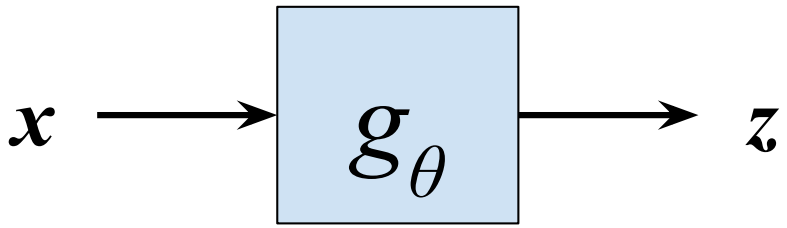


Figure: Sample of complex distribution using normalizing flow

Latent Representation

The latent variable representation corresponding to $\boldsymbol{x}$ is easily found by inverse transformation $\boldsymbol{g}_\theta(\cdot)$ but you can't find semantically meaningful latent directions (in contrast to VAE).

Normalizing Flow Limitations

Continuous Random Variable

The change of variable technique is only applicable to continuous random variables, thus $\mathbb{X}$ and $\mathbb{Z}$ must be continuous random variables.

Base Distribution

The base distribution should have the following properties:

- *Easy-to-sample*
- Straightforward to compute the likelihood $p(\boldsymbol{z})$

Equal Dimension

Invertibility of transformation necessitates $\mathbb{X}$ and $\mathbb{Z}$ to be of the same dimension. Thus:

- $\boldsymbol{J}_g(\boldsymbol{x}) \in \mathbb{R}^{D \times D}$
- The determinant calculations is of $\mathcal{O}(D^3)$

☞ The transformation $\boldsymbol{f}_\theta$ should be designed carefully to reduce the calculations

Section 4

Learning

Maximum Likelihood Estimation

To minimize the KL divergence between $p_{\text{data}}(\mathbb{X})$ and $p_\theta(\mathbb{X})$, we should maximize the training data likelihood with respect to model parameters $\boldsymbol{\theta}$ as:

$$\boldsymbol{\theta}^\star = \underset{\boldsymbol{\theta}}{\text{argmax}}\, \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\Big[\log p_\theta(\boldsymbol{x})\Big] \overset{(a)}{\approx} \underset{\boldsymbol{\theta}}{\text{argmax}}\, \frac{1}{N}\sum_{i=1}^{N} \log p_\theta(\boldsymbol{x}_i)$$

where (a) follows from Monte Carlo estimation.

Data Likelihood

Using Inverse Transformation

To train the Normalizing Flow model, we need to calculate the data likelihood as:

$$\log p_\theta(\boldsymbol{x}_i) = \log \left(p\Big(\boldsymbol{g}_\theta(\boldsymbol{x}_i)\Big) \Big| \det \boldsymbol{J}_{\boldsymbol{g}}(\boldsymbol{x}_i) \Big| \right)$$

$$= \log \left(p\Big(\boldsymbol{g}_\theta(\boldsymbol{x}_i)\Big) \Big| \det \prod_{l=1}^{L} \boldsymbol{J}_{g_l}(\boldsymbol{z}_i(l)) \Big| \right) \qquad \#\text{Using chain rule}$$

$$= \log \left(p\Big(\boldsymbol{g}_\theta(\boldsymbol{x}_i)\Big) \Big| \prod_{l=1}^{L} \det \boldsymbol{J}_{g_l}(\boldsymbol{z}_i(l)) \Big| \right) \qquad \#\text{Determinant properties}$$

Using Inverse Transformation (cont.)

$$\log p_\theta(\boldsymbol{x}_i) = \log \left(p\Big(\boldsymbol{g}_\theta(\boldsymbol{x}_i)\Big) \Big| \prod_{l=1}^{L} \det \boldsymbol{J}_{g_l}(\boldsymbol{z}_i(l)) \Big| \right)$$

$$= \log \left(p\Big(\boldsymbol{g}_\theta(\boldsymbol{x}_i)\Big) \prod_{l=1}^{L} \Big| \det \boldsymbol{J}_{g_l}(\boldsymbol{z}_i(l)) \Big| \right) \qquad \#\text{Absolute value properties}$$

$$= \log p\Big(\boldsymbol{g}_\theta(\boldsymbol{x}_i)\Big) + \sum_{l=1}^{L} \log \Big| \det \boldsymbol{J}_{g_l}(\boldsymbol{z}_i(l)) \Big| \qquad \#\text{Logarithm properties}$$

Data Likelihood

Using Inverse Transformation (Cont.)

Thus to calculate the data likelihood, we can use the following:

$$\boldsymbol{\theta}^{\star} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \frac{1}{N} \sum_{i=1}^{N} \log p\Big(\boldsymbol{g}_{\theta}(\boldsymbol{x}_i)\Big) + \sum_{l=1}^{L} \log \Big| \det \boldsymbol{J}_{g_l}(\boldsymbol{z}_i(l)) \Big|, \ \boldsymbol{z}_i(l) = \boldsymbol{f}_l(\boldsymbol{z}_i(l-1))$$

Using Forward Transformation

Similar to the inverse transformation, we can calculate the likelihood using forward transformation which leads to the following problem:

$$\boldsymbol{\theta}^{\star} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \frac{1}{N} \sum_{i=1}^{N} \log p\Big(\boldsymbol{g}_{\theta}(\boldsymbol{x}_i)\Big) - \sum_{l=1}^{L} \log \Big| \det \boldsymbol{J}_{f_l}(\boldsymbol{z}_i(l-1)) \Big|$$

Section 5

Nonlinear Independent Component Estimation

Additive Coupling Layer [2]

Transformation

- $\boldsymbol{f}_\theta : \boldsymbol{z} \to \boldsymbol{x}$: Assume $\boldsymbol{z}, \boldsymbol{x} \in \mathbb{R}^D$, then the forward transformation is defined as:
  - Select a random integer d such that $1 \leq d < D$
  - $\boldsymbol{x}_{1:d} = \boldsymbol{z}_{1:d}$
  - $\boldsymbol{x}_{d+1:D} = \boldsymbol{z}_{d+1:D} + \boldsymbol{m}_\theta(\boldsymbol{z}_{1:d})$

  where $\boldsymbol{m}_\theta(\cdot)$ can be any neural network architecutre.
- $\boldsymbol{g}_\theta : \boldsymbol{x} \to \boldsymbol{z}$: Using the forward transformation, one can easily show that:
  - $\boldsymbol{z}_{1:d} = \boldsymbol{x}_{1:d}$
  - $\boldsymbol{z}_{d+1:D} = \boldsymbol{x}_{d+1:D} - \boldsymbol{m}_\theta(\boldsymbol{z}_{1:d}) = \boldsymbol{x}_{d+1:D} - \boldsymbol{m}_\theta(\boldsymbol{x}_{1:d})$

Additive Coupling Layer [2]

Figure: Forward transformation of upper part

Additive Coupling Layer [2]

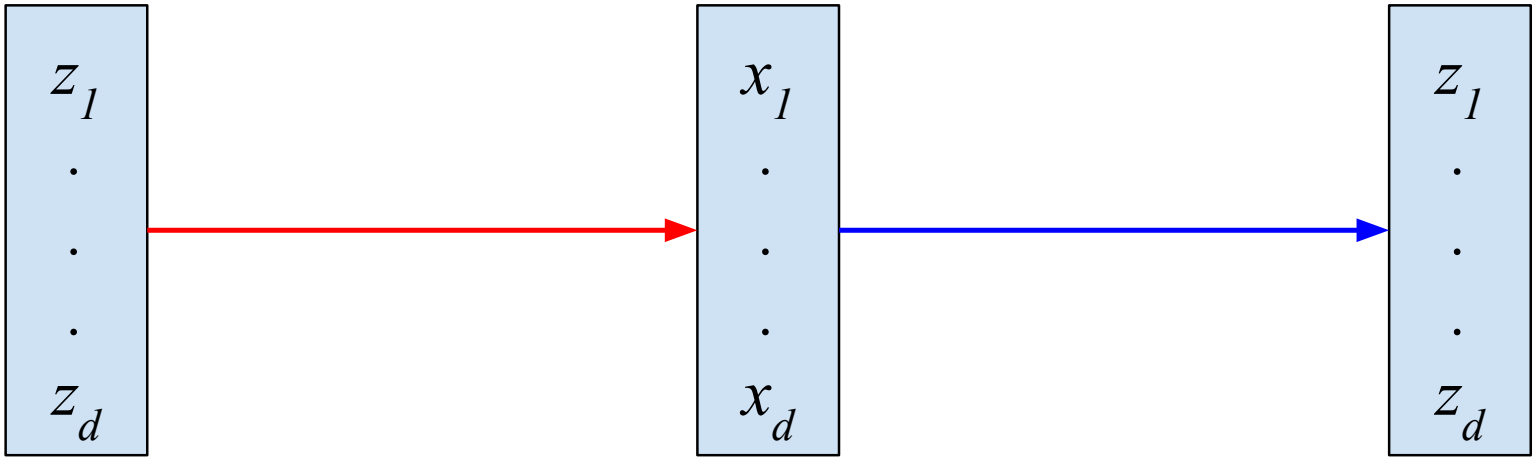


Figure: Inverse transformation of upper part

Additive Coupling Layer [2]

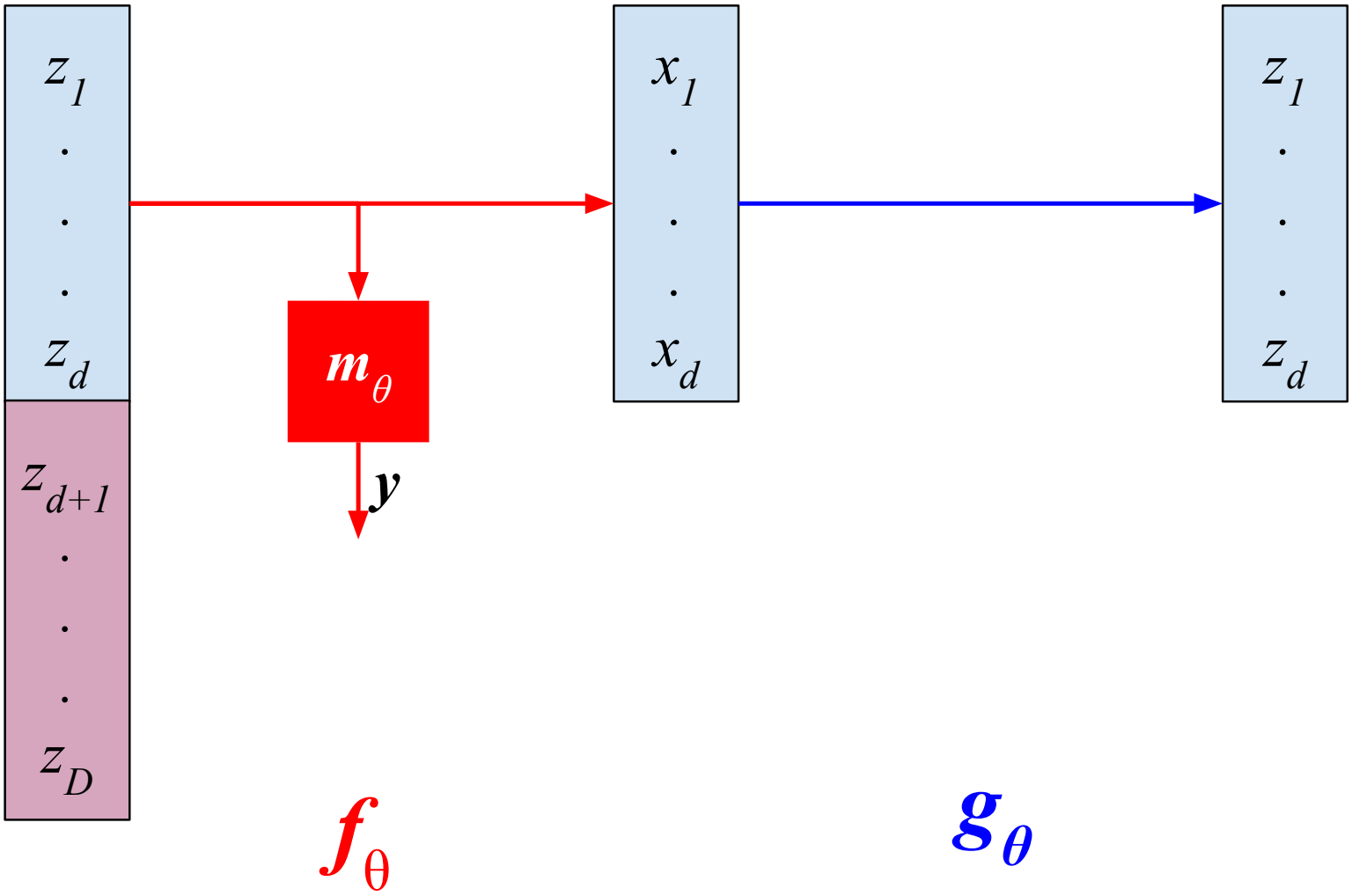


Figure: Forward transformation of lower part

Additive Coupling Layer [2]

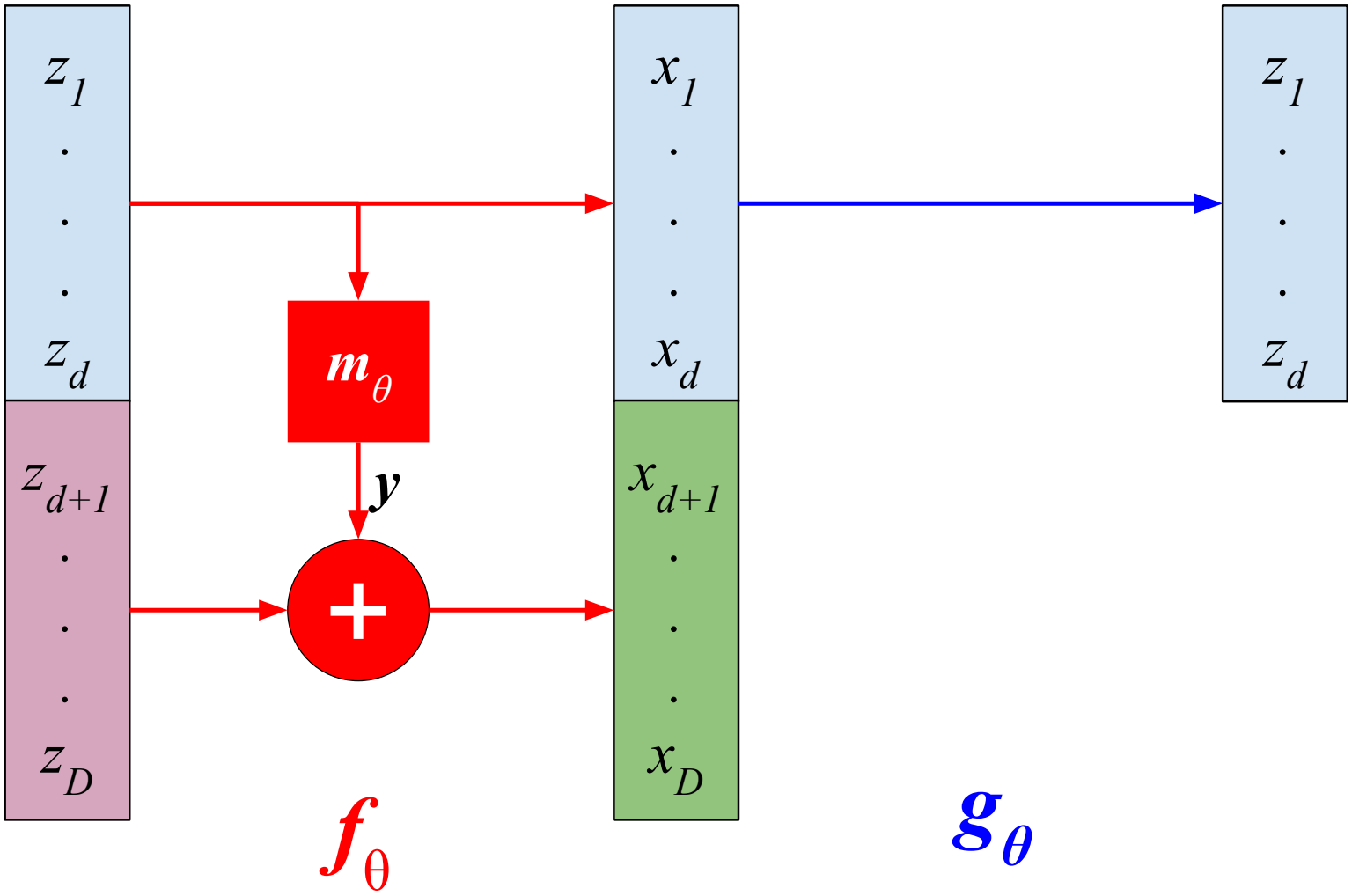


Figure: Forward transformation of lower part

Additive Coupling Layer [2]

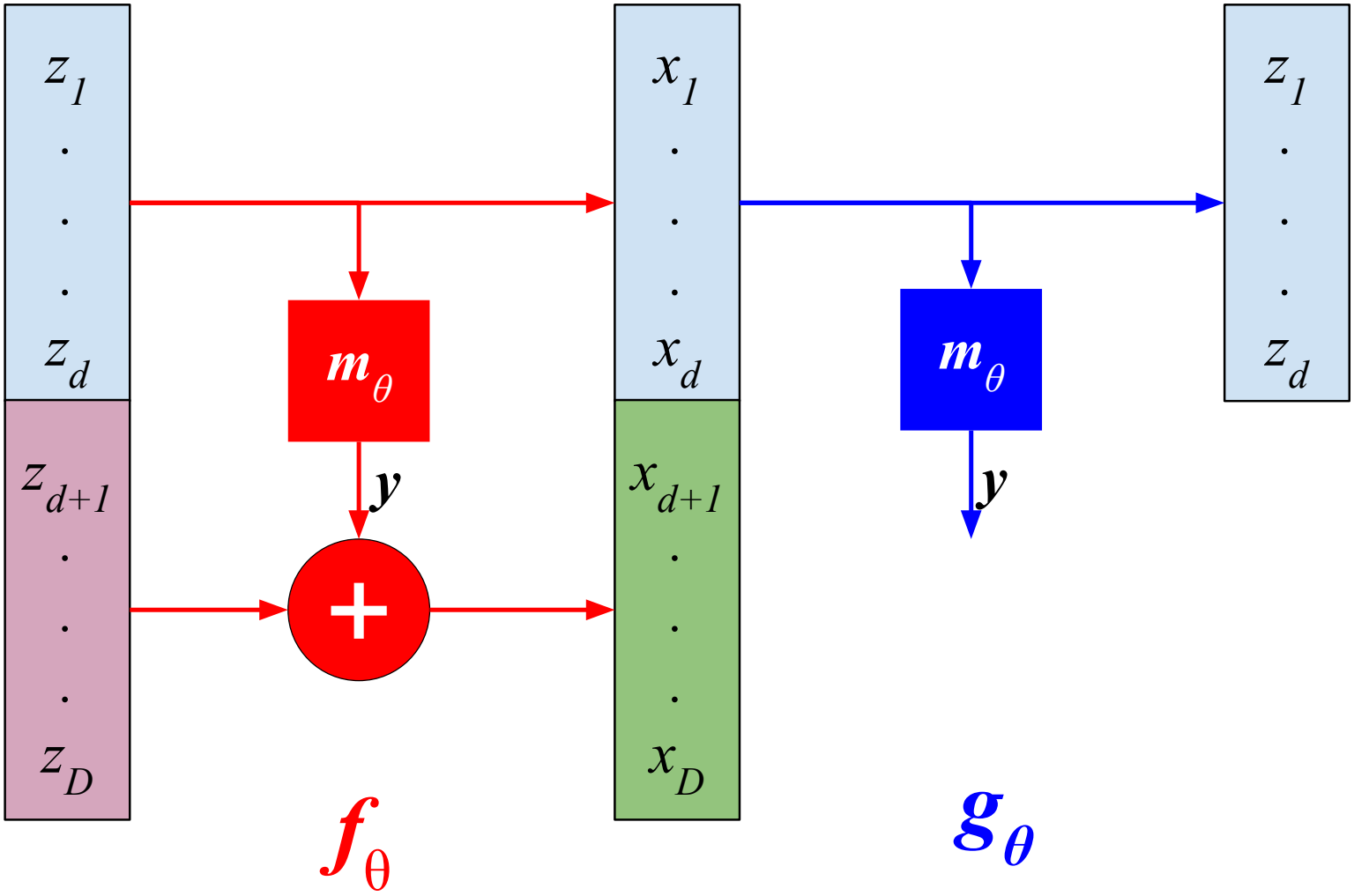


Figure: Inverse transformation of lower part

Additive Coupling Layer [2]

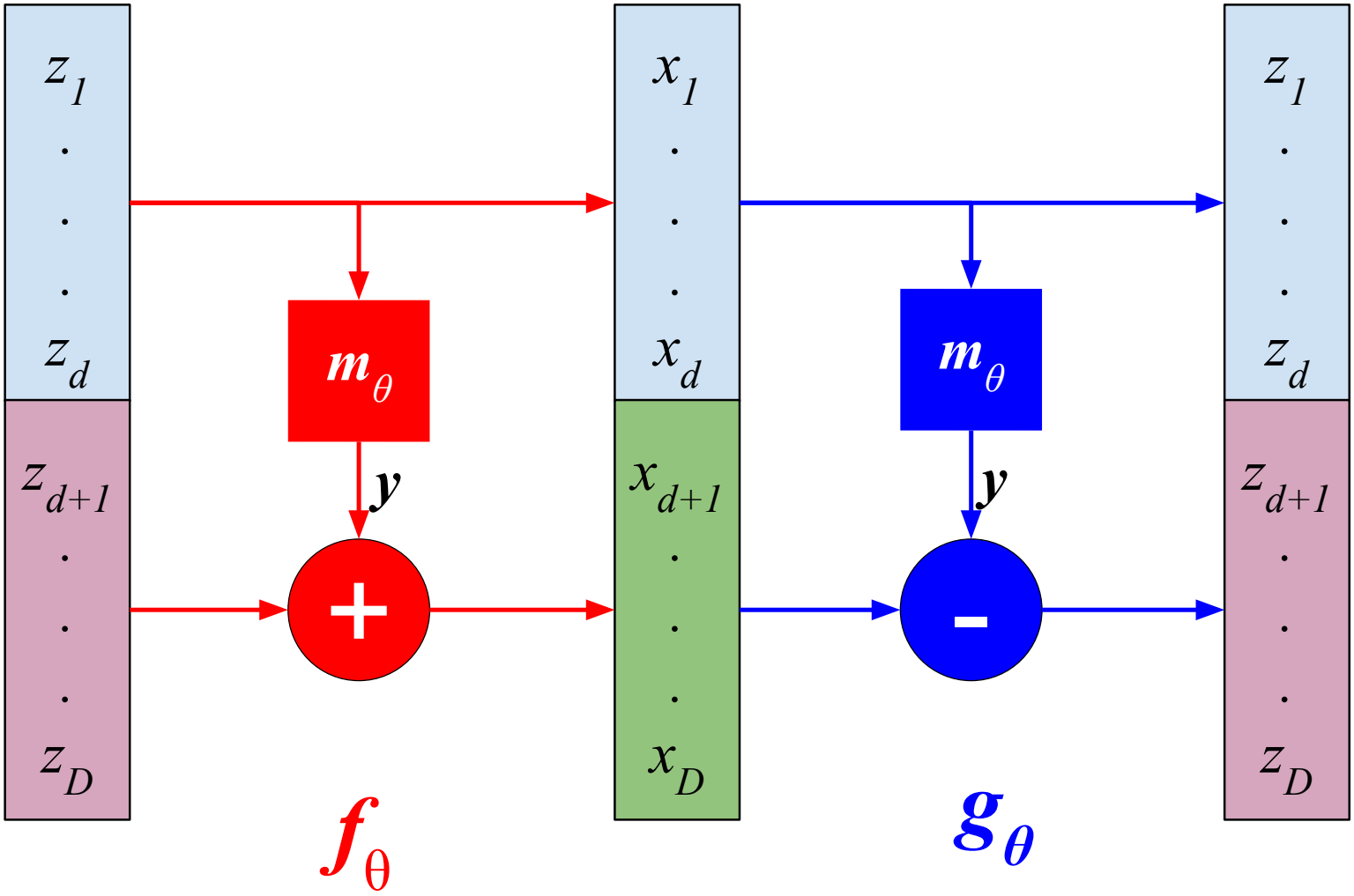


Figure: Inverse transformation of lower part

Additive Coupling Layer [2]

Jacobian

The Jacobian for the forward transformation of the additive coupling layer is:

$$\begin{aligned}
\boldsymbol{J}_f &= \det \begin{bmatrix} \left[\frac{\partial \boldsymbol{x}_{1:d}}{\partial \boldsymbol{z}_{1:d}}\right]_{d\times d} & \left[\frac{\partial \boldsymbol{x}_{1:d}}{\partial \boldsymbol{z}_{d+1:D}}\right]_{d\times(n-d)} \\ \left[\frac{\partial \boldsymbol{x}_{d+1:D}}{\partial \boldsymbol{z}_{1:d}}\right]_{(n-d)\times d} & \left[\frac{\partial \boldsymbol{x}_{d+1:D}}{\partial \boldsymbol{z}_{d+1:D}}\right]_{(n-d)\times(n-d)} \end{bmatrix} \\
&= \det \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \frac{\partial \boldsymbol{m}_\theta(\boldsymbol{z}_{1:d})}{\partial \boldsymbol{z}_{1:d}} & \boldsymbol{I} \end{bmatrix} \\
&\overset{(a)}{=} 1
\end{aligned}$$

where (a) follows the fact that for triangular matrices, the determinant is the product of diagonal elements.

- ☞ This value for the determinant can greatly reduce the computations to calculate the log-likelihood.
- ☞ This transformation is volume preserving.

Rescaling Layer [2]

Transformation

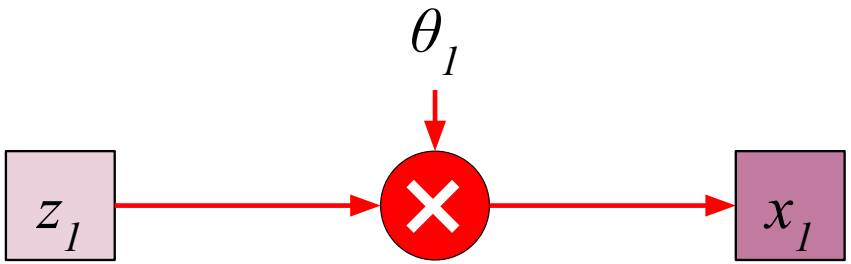
- $\boldsymbol{f}_\theta : \boldsymbol{z} \to \boldsymbol{x}$: Assume $\boldsymbol{z}, \boldsymbol{x} \in \mathbb{R}^D$, then the forward transformation is defined as:

$$x_i = \theta_i z_i \ , \ i = 1, \dots, D$$

- $\boldsymbol{g}_\theta \boldsymbol{x} \to \boldsymbol{z}$: Using the forward transformation, one can easily show that:

$$z_i = \frac{x_i}{\theta_i} \ , \ i = 1, \dots, D$$

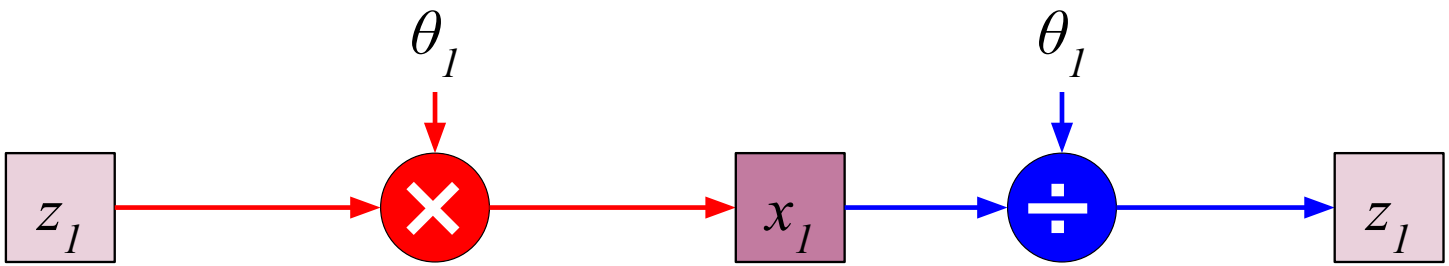
Rescaling Layer [2]



$$f_\theta$$

Figure: Forward transformation of first element

Rescaling Layer [2]



f_θ g_θ

Figure: Inverse transformation of first element

Rescaling Layer [2]

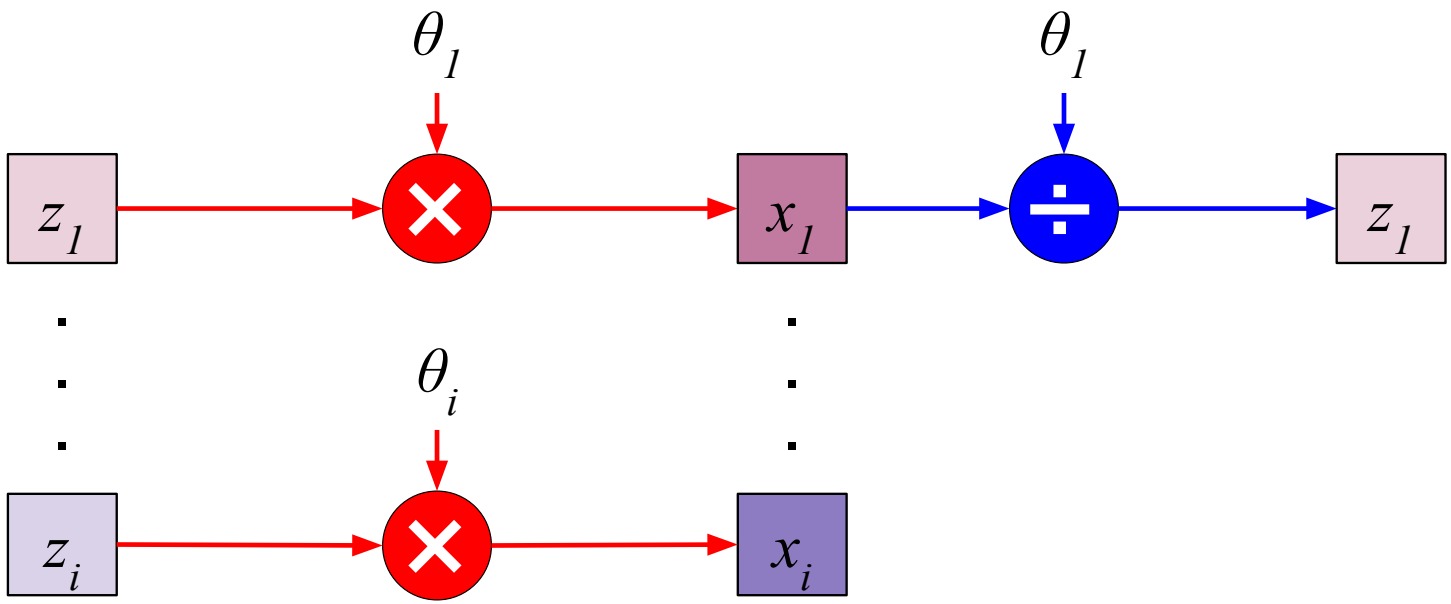


Figure: Forward transformation of i-th element

Rescaling Layer [2]

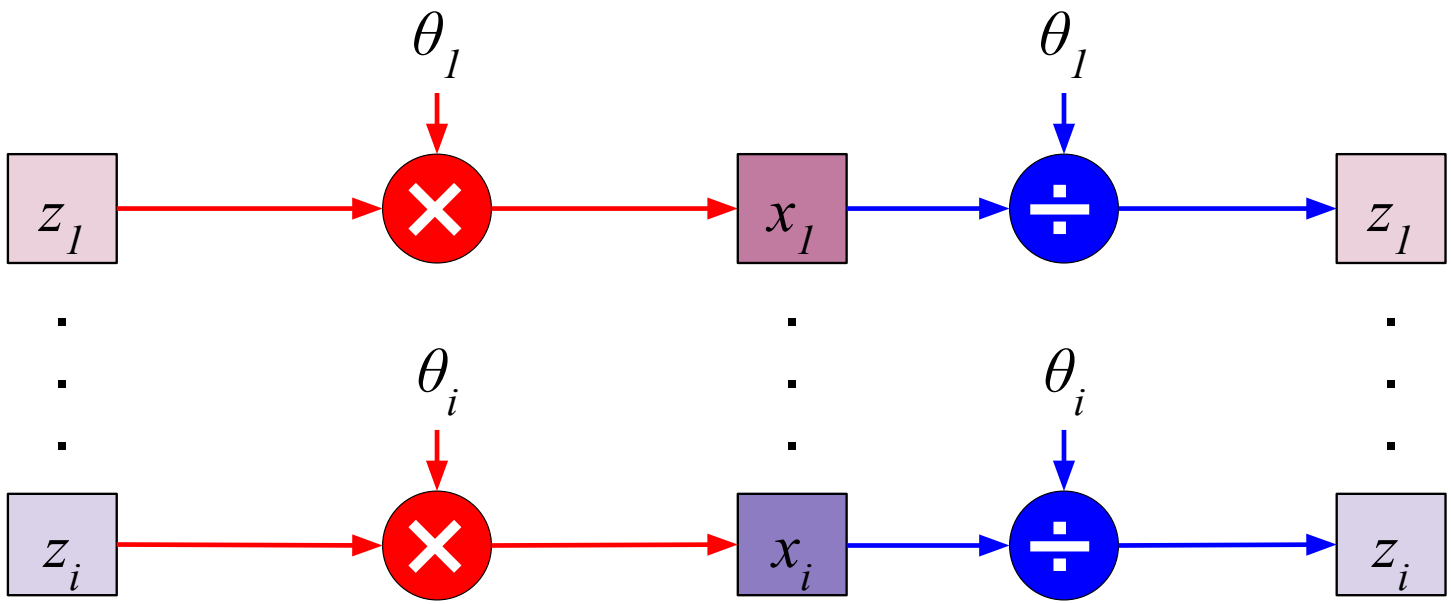


Figure: Inverse transformation of i-th element

Rescaling Layer [2]

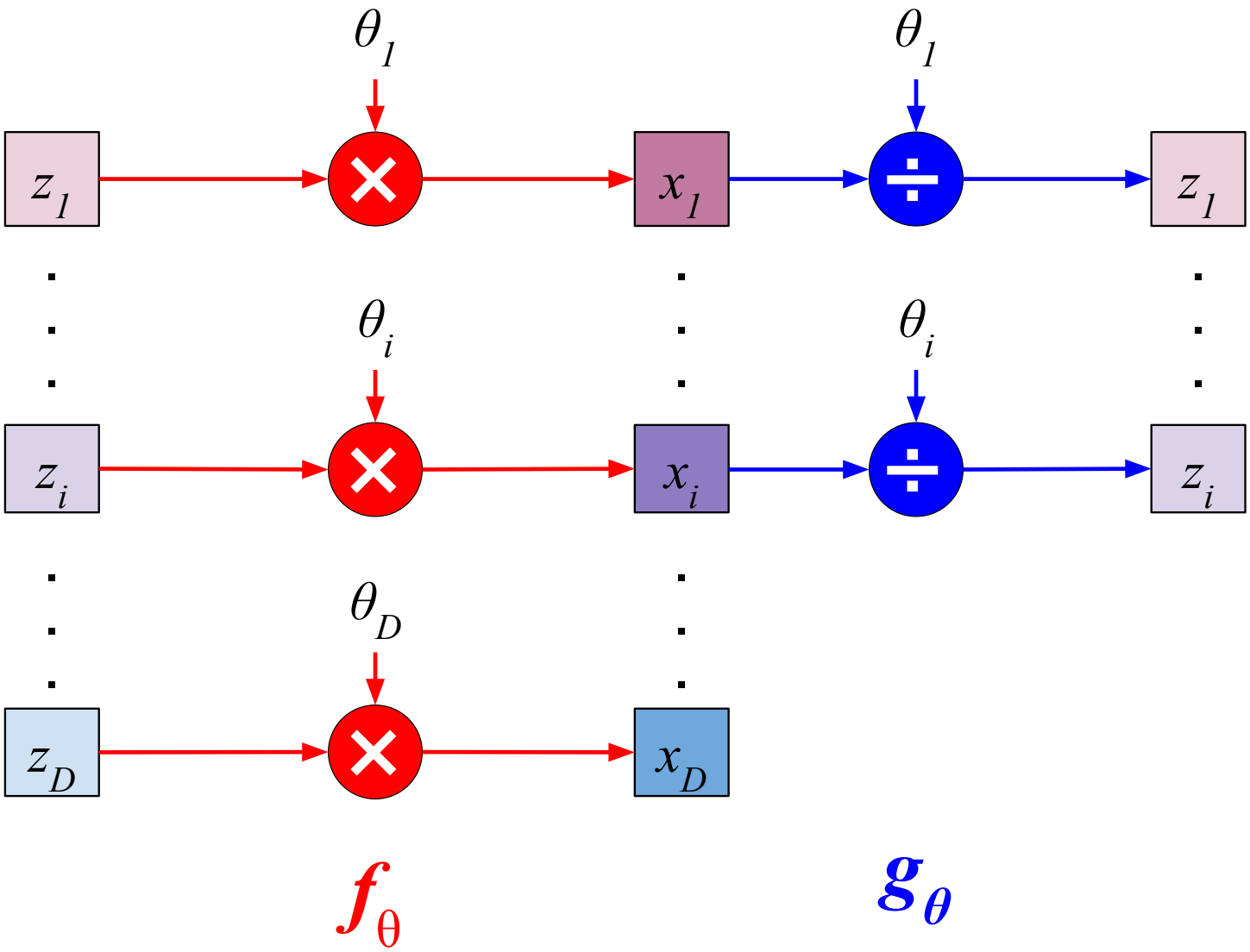


Figure: Forward transformation of last element

Rescaling Layer [2]

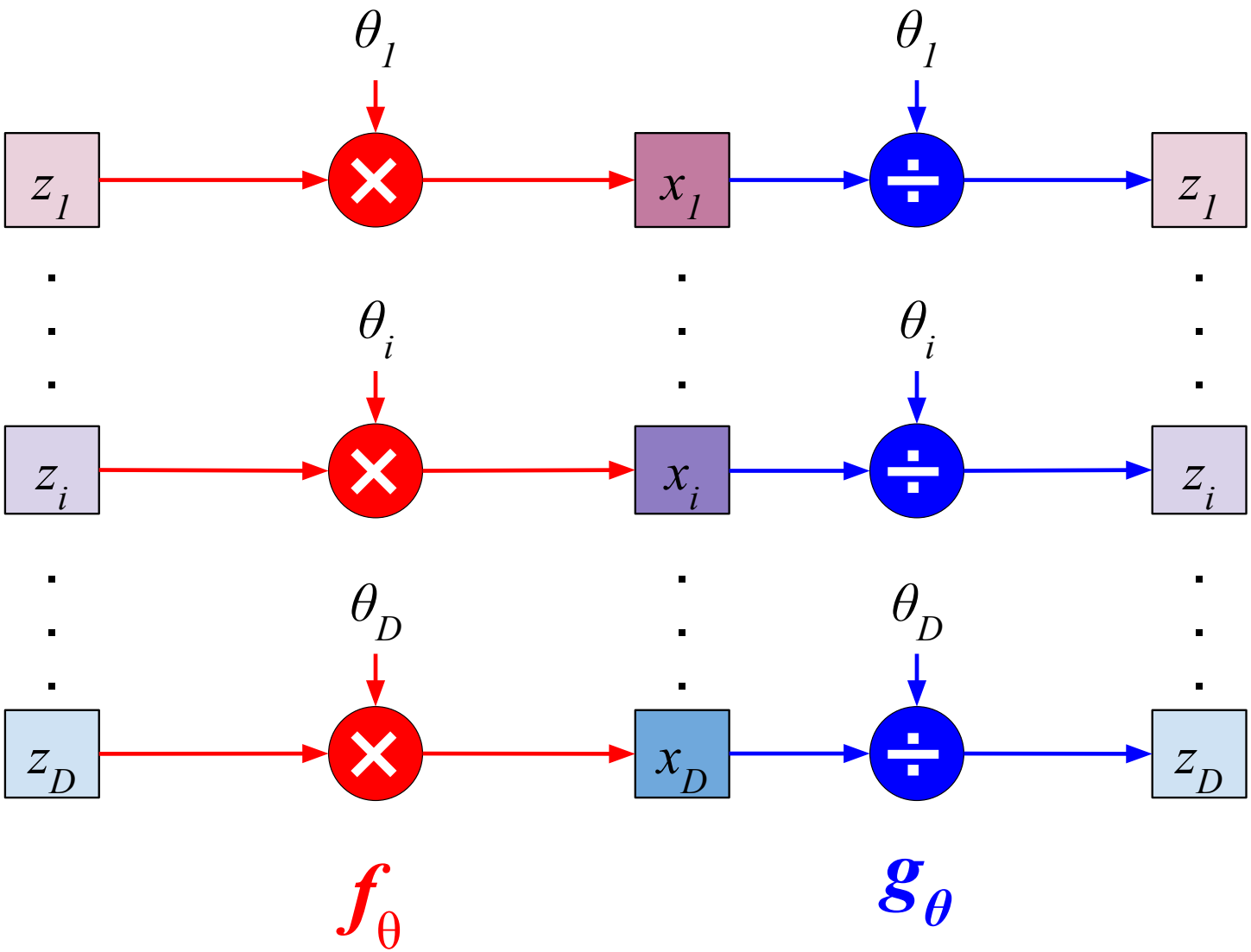


Figure: Inverse transformation of last element

Rescaling Layer [2]

Jacobian

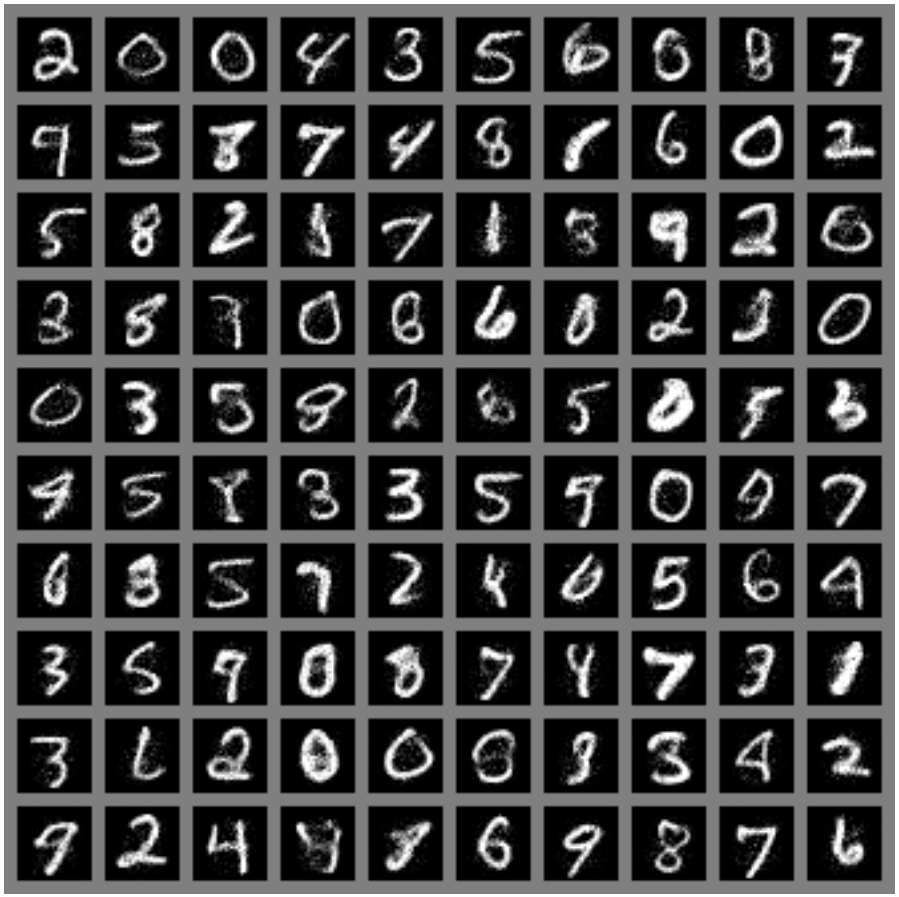
The Jacobian for the forward transformation of the rescaling layer is:

$$\boldsymbol{J}_f = \det \begin{bmatrix} \frac{\partial x_1}{\partial z_1} & \frac{\partial x_1}{\partial z_2} & \dots & \frac{\partial x_1}{\partial z_D} \\ \frac{\partial x_2}{\partial z_1} & \frac{\partial x_2}{\partial z_2} & \dots & \frac{\partial x_2}{\partial z_D} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial x_D}{\partial z_1} & \frac{\partial x_D}{\partial z_2} & \dots & \frac{\partial x_D}{\partial z_D} \end{bmatrix}$$

$$= \det \begin{bmatrix} \theta_1 & 0 & \dots & 0 \\ 0 & \theta_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \theta_D \end{bmatrix}$$

$$= \prod_{i=1}^{D} \theta_i$$

NICE Performance [2]



(a) MNIST dataset

(b) TFD dataset

Figure: NICE performance. The architecture comprises four additive coupling layers and one rescaling layer

NICE Performance [2]

(a) SVHN dataset

(b) CIFAR10 dataset

Figure: NICE performance. The architecture comprises four additive coupling layers and one rescaling layer

Image Inpainting Using NICE [2]

Image Inpainting

Assume:

- $p_\theta(\boldsymbol{x})$ to be a trained model over the MNIST dataset using NICE.
- You are give a sample $\boldsymbol{x}$ where the $\boldsymbol{x}_o$ is the observed part (clean) and $\boldsymbol{x}_h$ is hidden part (corrupted).

☞ Find an estimate to the corrupted part?

Maximum a Posteriori Estimation

To find MAP approximation to the corrupted part, we have:

$$\boldsymbol{x}_{h,map} = \underset{\boldsymbol{x}_h}{\operatorname{argmax}}\, p_\theta(\boldsymbol{x}_h|\boldsymbol{x}_o) \overset{(a)}{\equiv} \underset{\boldsymbol{x}_h}{\operatorname{argmax}}\, \frac{p_\theta(\boldsymbol{x}_h, \boldsymbol{x}_o)}{p_\theta(\boldsymbol{x}_o)}$$

$$\equiv \underset{\boldsymbol{x}_h}{\operatorname{argmax}}\, p_\theta(\boldsymbol{x}_h, \boldsymbol{x}_o)$$

Image Inpainting Using NICE [2]

Maximum a Posteriori Estimation (cont.)

Thus we should solve the following problem:

$$\boldsymbol{x}_{h,map} = \underset{\boldsymbol{x}_h}{\operatorname{argmax}}\, p_\theta(\boldsymbol{x}_h, \boldsymbol{x}_o)$$

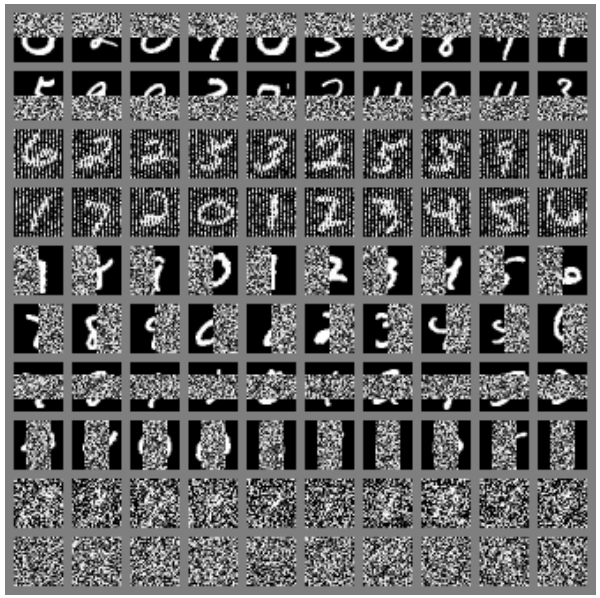
Note:

- The model parameters are fixed (trained).
- We should maximize $p_\theta(\boldsymbol{x}_h, \boldsymbol{x}_o)$ w.r.t. $\boldsymbol{x}_h$ using gradient descent:

$$\boldsymbol{x}_{h,k+1} = \boldsymbol{x}_{h,k} + \frac{\partial p_\theta(\boldsymbol{x}_h, \boldsymbol{x}_o)}{\partial \boldsymbol{x}_h}$$

NICE Performance [2]

(a) Original sample



(b) corrupted Sample

(c) MAP estimation

Figure: NICE inpainting. The corruption in the middle image are: top rows, bottom rows, odd pixels, even pixels, left side, right side, middle vertically, middle horizontally, 75% random and 90% random.

Section 6

Real-valued Non-volume Preserving Transformations

Coupling Layer [3]

Transformation

- $\boldsymbol{f}_\theta : \boldsymbol{z} \to \boldsymbol{x}$: Assume $\boldsymbol{z}, \boldsymbol{x} \in \mathbb{R}^D$, then the forward transformation is defined as:
  - Select a random integer d such that $1 \leq d < D$
  - $\boldsymbol{x}_{1:d} = \boldsymbol{z}_{1:d}$
  - $\boldsymbol{x}_{d+1:D} = \boldsymbol{z}_{d+1:D} \odot \exp\big(\boldsymbol{s}_\theta(\boldsymbol{z}_{1:d})\big) + \boldsymbol{t}_\theta(\boldsymbol{z}_{1:d})$

  where $\boldsymbol{s}_\theta(\cdot)$ and $\boldsymbol{t}_\theta(\cdot)$ can be any neural network architecutre.
- $\boldsymbol{g}_\theta : \boldsymbol{x} \to \boldsymbol{z}$: Using the forward transformation, one can easily show that:
  - $\boldsymbol{z}_{1:d} = \boldsymbol{x}_{1:d}$
  -

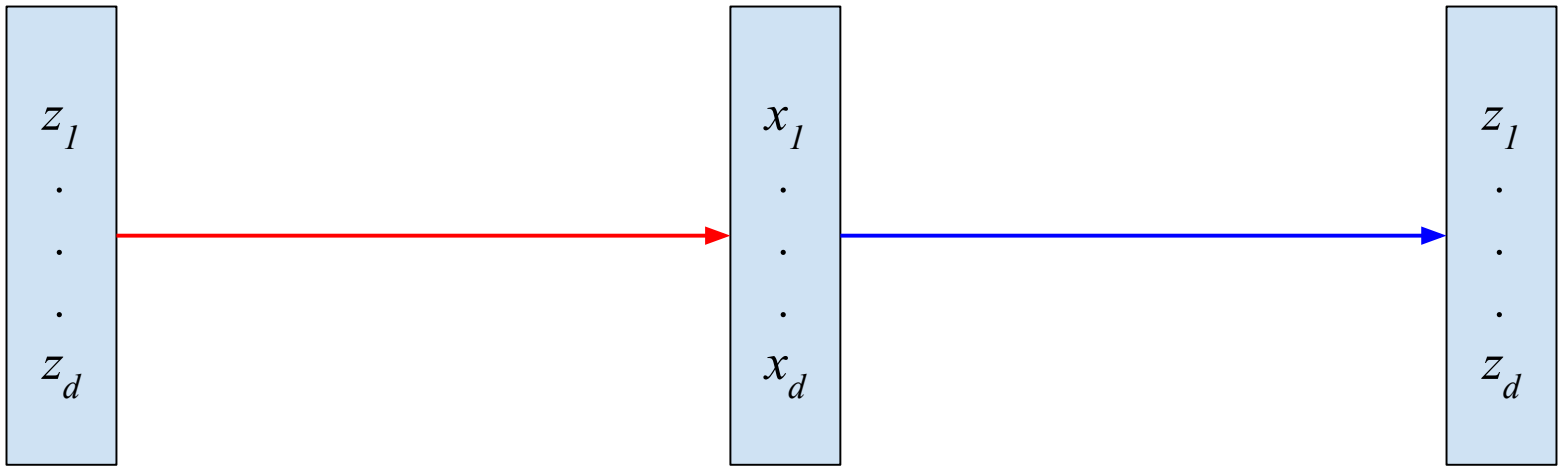
$$\begin{aligned}\boldsymbol{z}_{d+1:D} &= \big(\boldsymbol{x}_{d+1:D} - \boldsymbol{t}_\theta(\boldsymbol{z}_{1:d})\big) \odot \exp\big(-\boldsymbol{s}_\theta(\boldsymbol{z}_{1:d})\big) \\ &= \big(\boldsymbol{x}_{d+1:D} - \boldsymbol{t}_\theta(\boldsymbol{x}_{1:d})\big) \odot \exp\big(-\boldsymbol{s}_\theta(\boldsymbol{x}_{1:d})\big)\end{aligned}$$

Coupling Layer [3]

$$\boldsymbol{f}_\theta$$

Figure: Forward transformation of upper part

Coupling Layer [3]



f_θ g_θ

Figure: Inverse transformation of upper part

Coupling Layer [3]

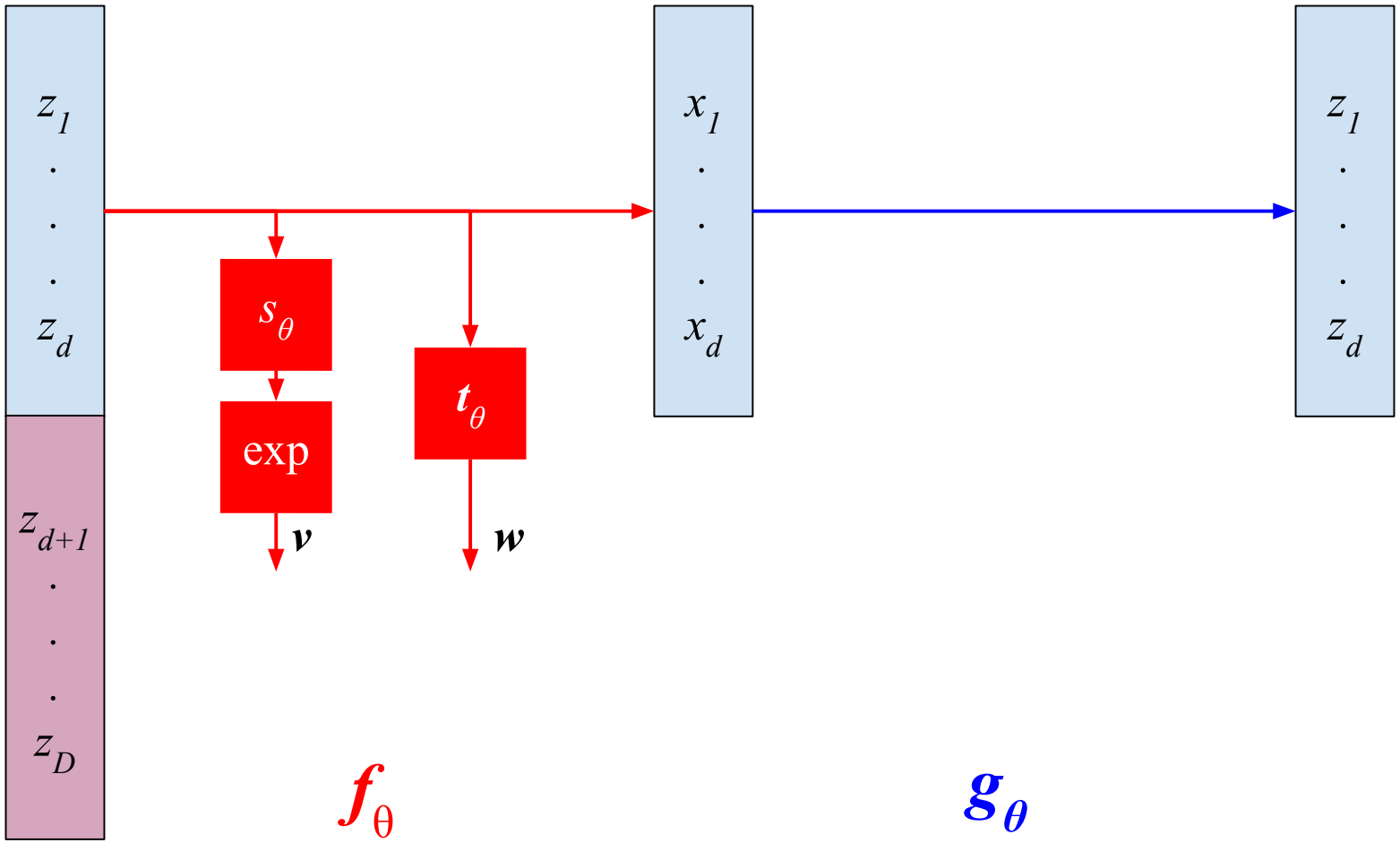


Figure: Forward transformation of lower part

Coupling Layer [3]

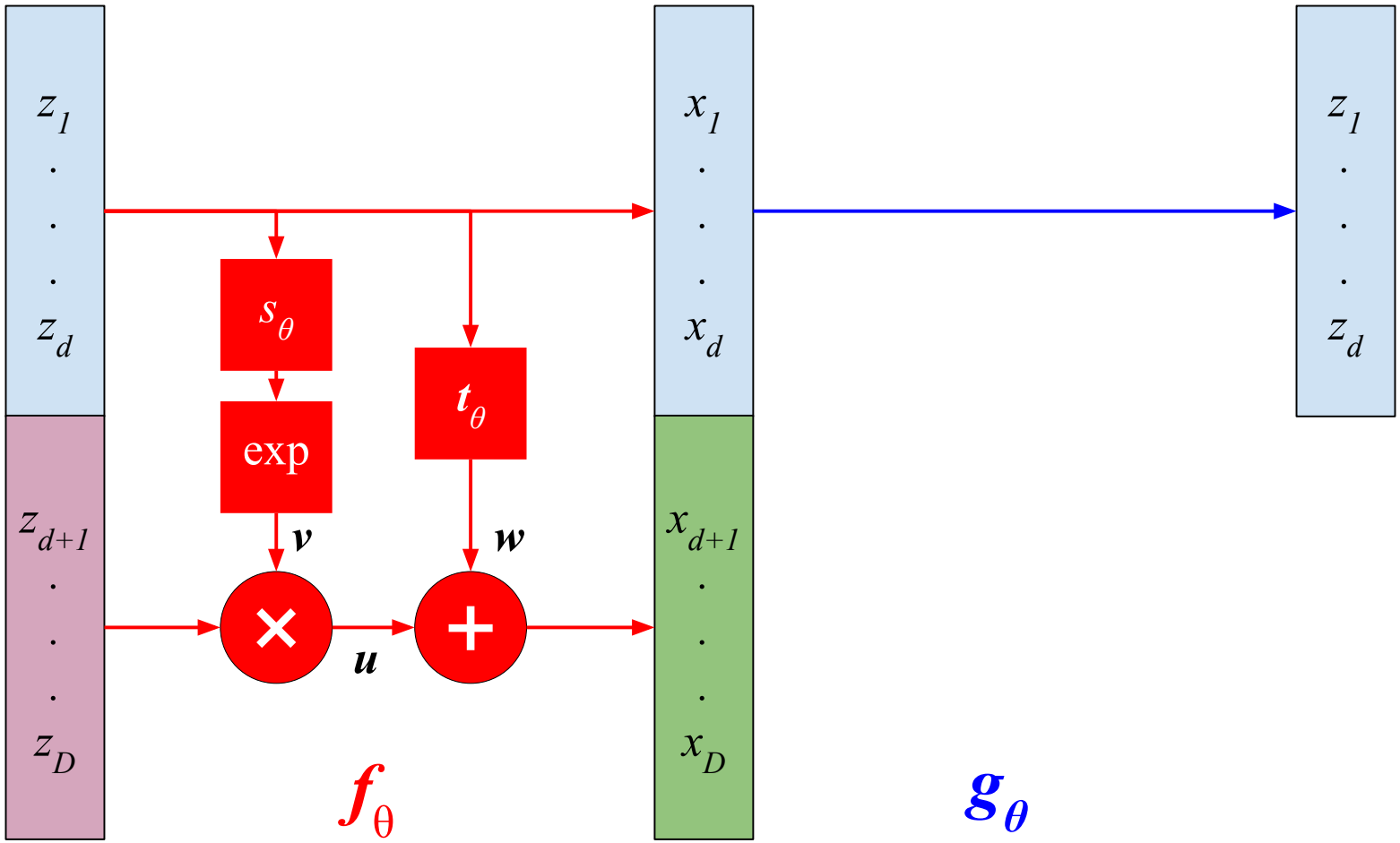


Figure: Forward transformation of lower part

Coupling Layer [3]

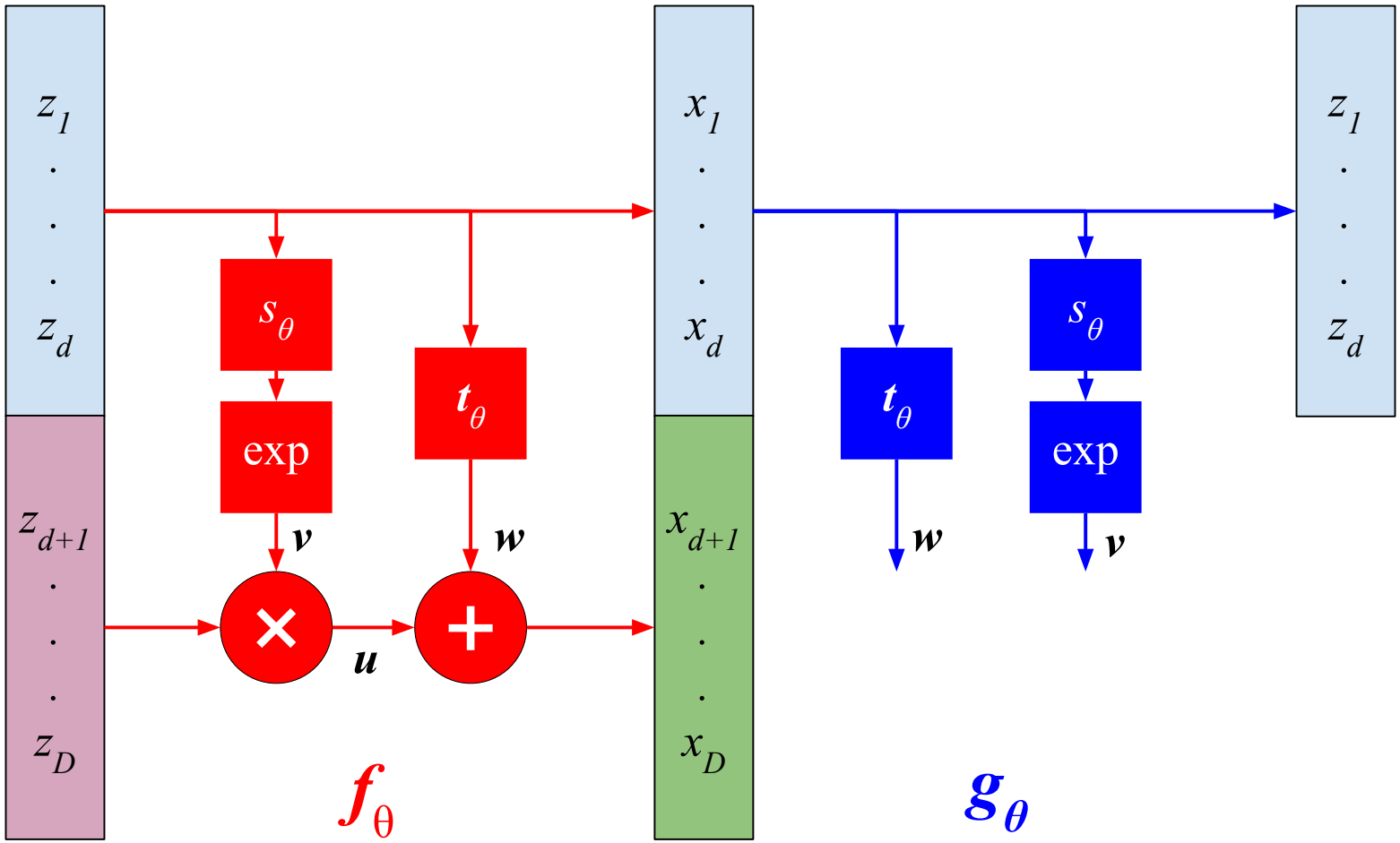


Figure: Inverse transformation of lower part

Coupling Layer [3]

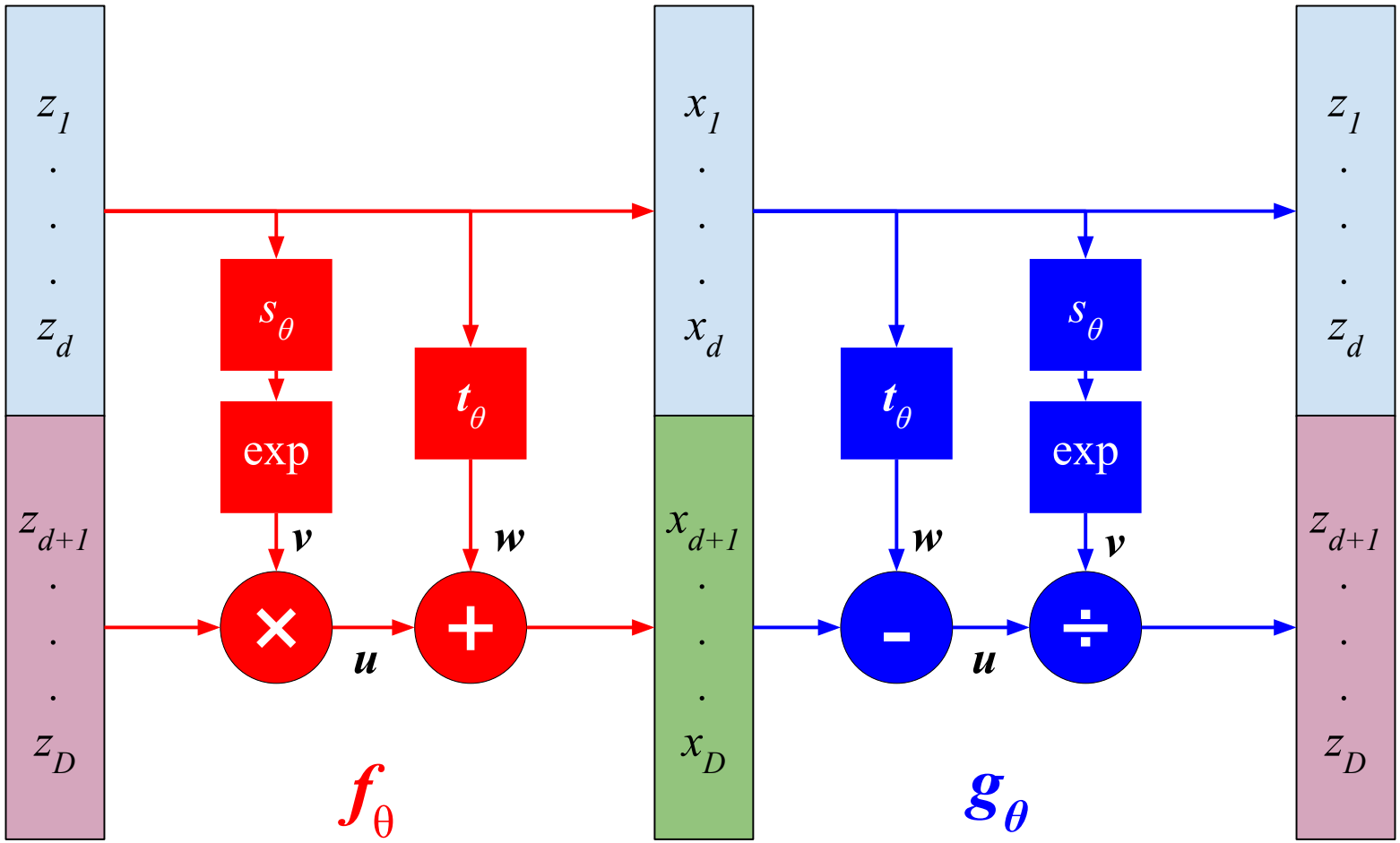


Figure: Inverse transformation of lower part

Coupling Layer [3]

Jacobian

The Jacobian for the forward transformation of the coupling layer is:

$$\boldsymbol{J}_f = \det \begin{bmatrix} \left[\frac{\partial \boldsymbol{x}_{1:d}}{\partial \boldsymbol{z}_{1:d}}\right]_{d\times d} & \left[\frac{\partial \boldsymbol{x}_{1:d}}{\partial \boldsymbol{z}_{d+1:D}}\right]_{d\times(n-d)} \\ \left[\frac{\partial \boldsymbol{x}_{d+1:D}}{\partial \boldsymbol{z}_{1:d}}\right]_{(n-d)\times d} & \left[\frac{\partial \boldsymbol{x}_{d+1:D}}{\partial \boldsymbol{z}_{d+1:D}}\right]_{(n-d)\times(n-d)} \end{bmatrix}$$

$$= \det \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \frac{\partial \boldsymbol{m}_\theta(\boldsymbol{z}_{1:d})}{\partial \boldsymbol{z}_{1:d}} & \begin{bmatrix} \exp(\boldsymbol{s}_{\theta,1}(\boldsymbol{x}_{1:d})) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \exp(\boldsymbol{s}_{\theta,D-d}(\boldsymbol{x}_{1:d})) \end{bmatrix} \end{bmatrix}$$

$$= \prod_{i=1}^{D-d} \exp(\boldsymbol{s}_{\theta,i}(\boldsymbol{x}_{1:d})) = \exp\left(\sum_{i=1}^{D-d} \boldsymbol{s}_{\theta,i}(\boldsymbol{x}_{1:d})\right)$$

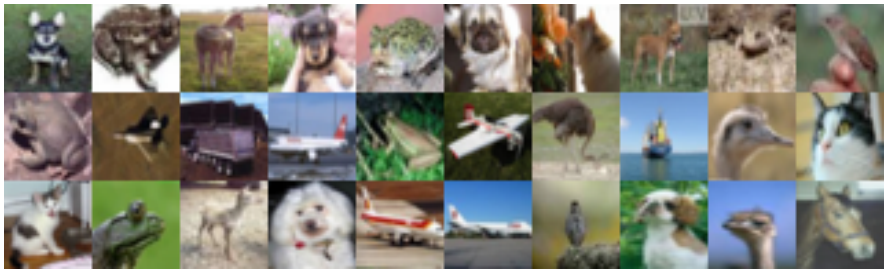
- The likelihood becomes more complicated in comparison to NICE.
- The transformation is not volume preserving anymore which justifies Real-NVP.

Real NVP Performance [3]

Other Note on Real NVP

- Combining several coupling layers
- Masked convolution for partitioning
- Multiscale architectures
- Batch-normalization

Real NVP Generation [3]



(a) Cifar10 dataset

(b) CelebA dataset

Figure: Left column: Dataset examples, right column: generated samples

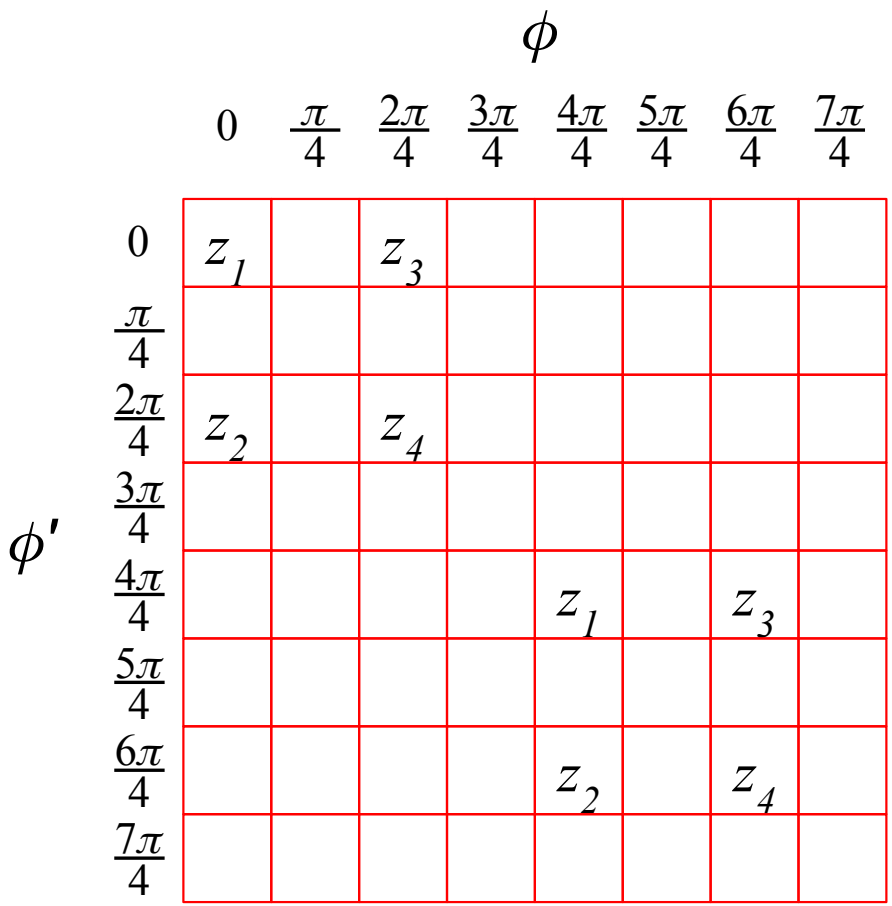
Real NVP Generation [3]

(a) ImageNet 32×32 dataset

(b) Imagenet 64×64 dataset

Figure: Left column: Dataset examples, right column: generated samples

Real NVP Latent Variable Interpolation [3]



Interpolation Formula

$$z = \cos\phi\left(\cos\phi' z_1 + \sin\phi' z_2\right) + \sin\phi\left(\cos\phi' z_3 + \sin\phi' z_4\right)$$

Real NVP Latent Variable Interpolation [3]

Figure: ImageNet 64×64 dataset

Real NVP Latent Variable Interpolation [3]

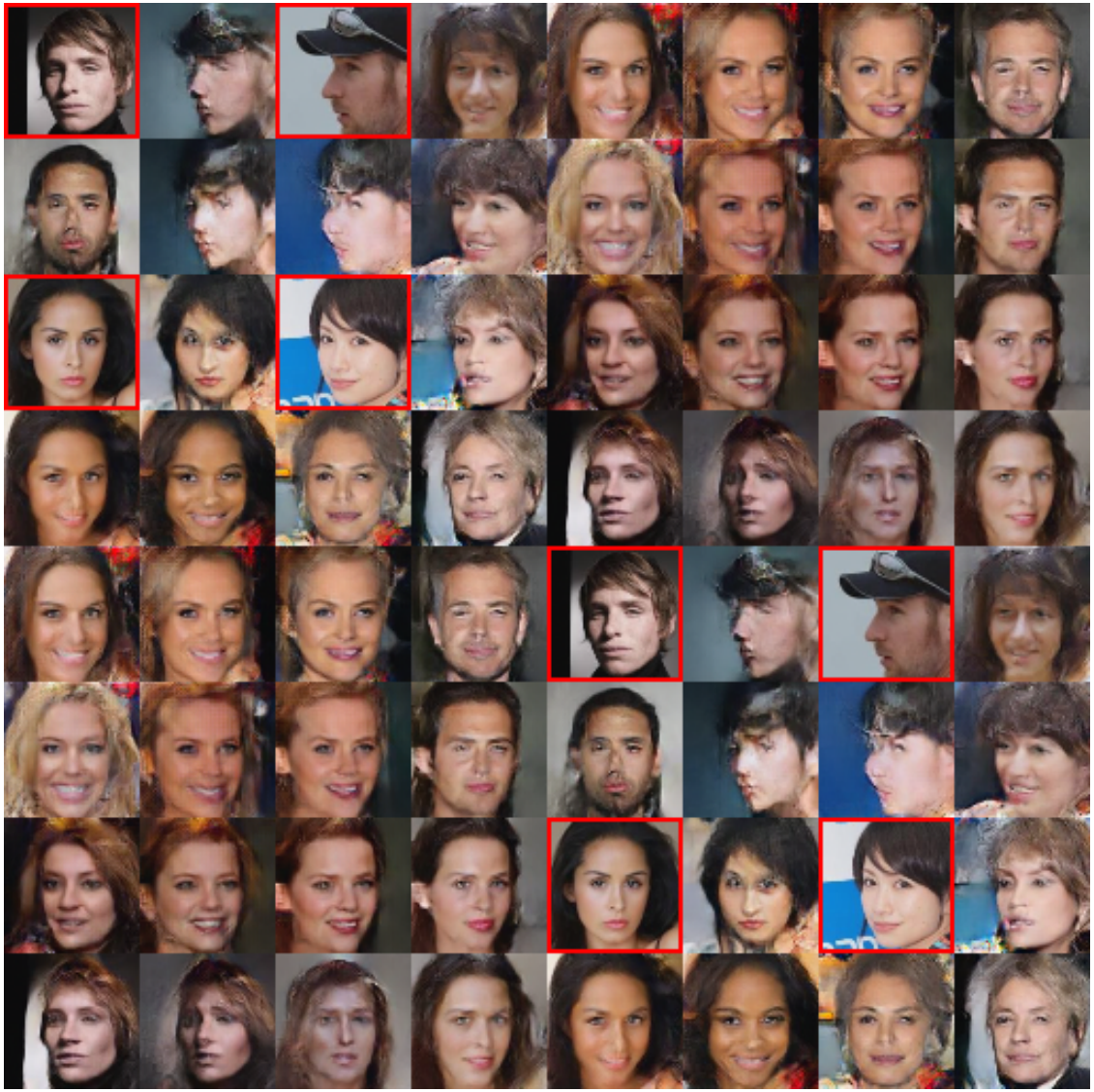


Figure: CelebA dataset

Real NVP Latent Variable Interpolation [3]

Figure: LSUN (bedroom category) dataset

Real NVP Latent Variable Interpolation [3]

Figure: LSUN (church outdoor category) dataset

Real NVP Latent Variable Interpolation [3]

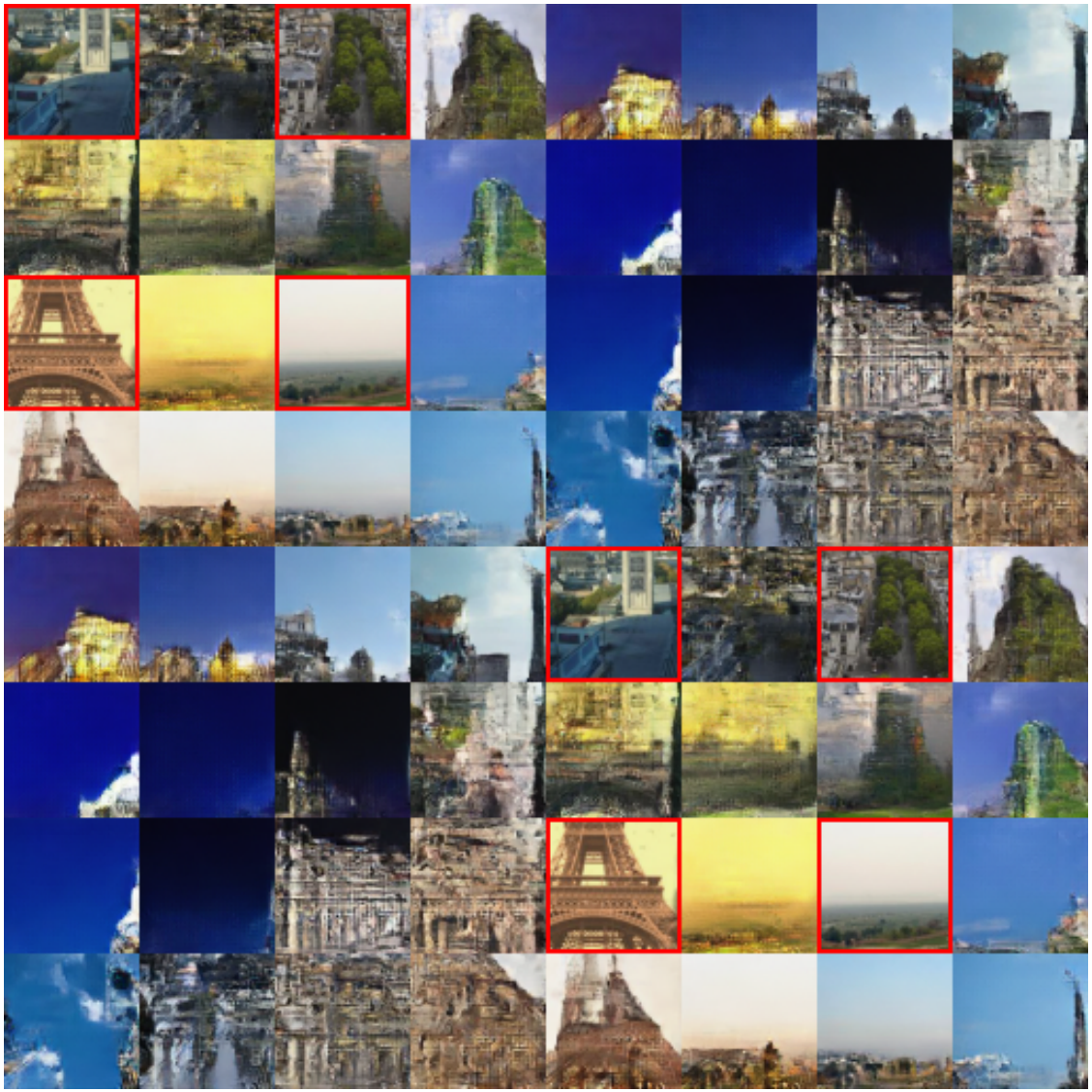


Figure: LSUN (tower category) dataset

Section 7

Masked Autoregressive Flows

Masked Autoregressive Flows [4]

Transformation

- $\boldsymbol{f}_\theta : \boldsymbol{z} \to \boldsymbol{x}$: Assume $\boldsymbol{z}, \boldsymbol{x} \in \mathbb{R}^D$, then the forward transformation is defined as:
  - $x_i = z_i \times \exp\left(\alpha_i(\boldsymbol{x}_{<i})\right) + \mu_i(\boldsymbol{x}_{<i})$
    - $i = 1: \; x_1 = z_1 \times \exp\left(\alpha_1(\boldsymbol{x}_{<1})\right) + \mu_1(\boldsymbol{x}_{<1}) = z_1 \times \exp(\alpha) + \mu$
    - $i = 2: \; x_2 = z_2 \times \exp\left(\alpha_2(\boldsymbol{x}_{<2})\right) + \mu_2(\boldsymbol{x}_{<2}) = z_2 \times \exp\left(\alpha_2(x_1)\right) + \mu_2(x_1)$
    - $\vdots$

  where $\alpha_i(\cdot)$ and $\mu_i(\cdot)$ can be any neural network architecutre with single output.

- $\boldsymbol{g}_\theta : \boldsymbol{x} \to \boldsymbol{z}$: Using the forward transformation, one can easily show that:

$$z_i = \frac{x_i - \mu_i(\boldsymbol{x}_{<i})}{\exp(\alpha_i(\boldsymbol{x}_{<i}))}$$

Masked Autoregressive Flows [4]

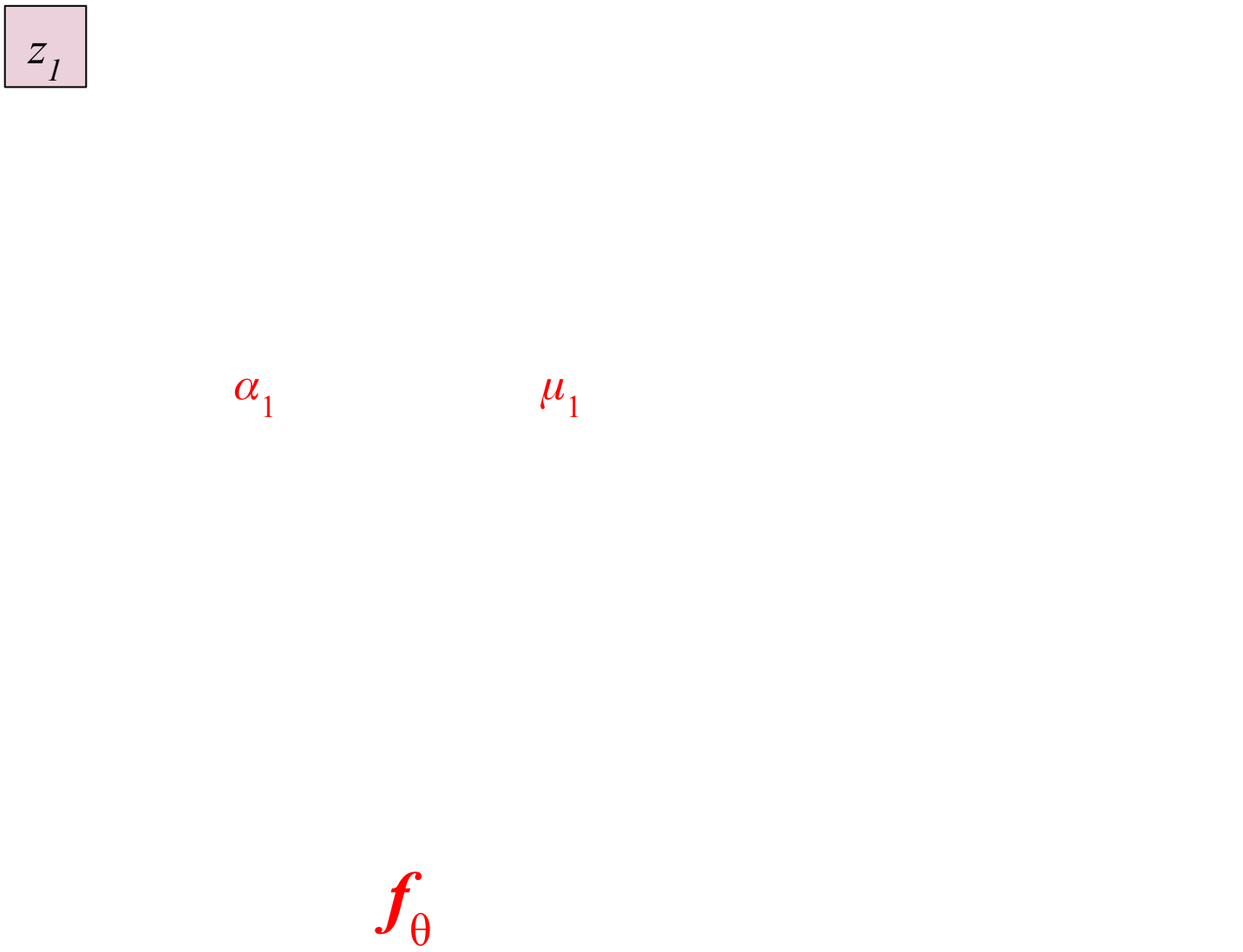
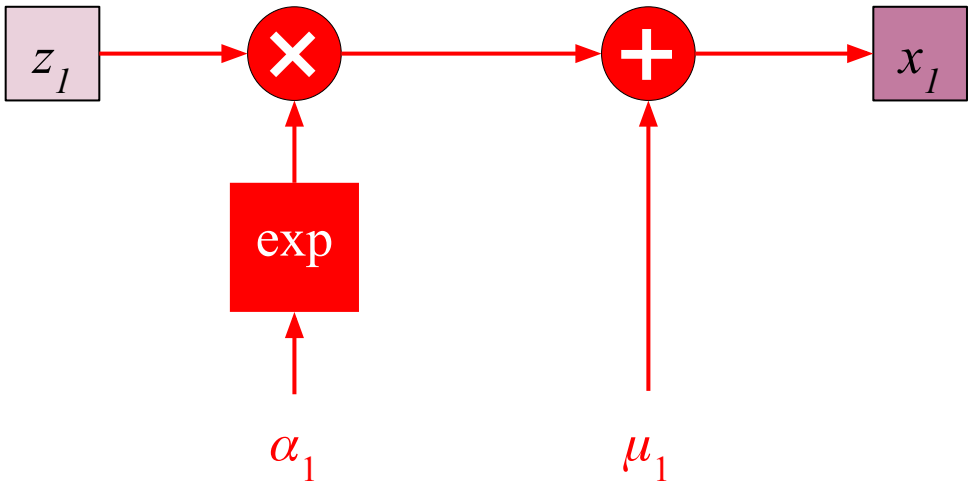


Figure: Forward transformation of first elment

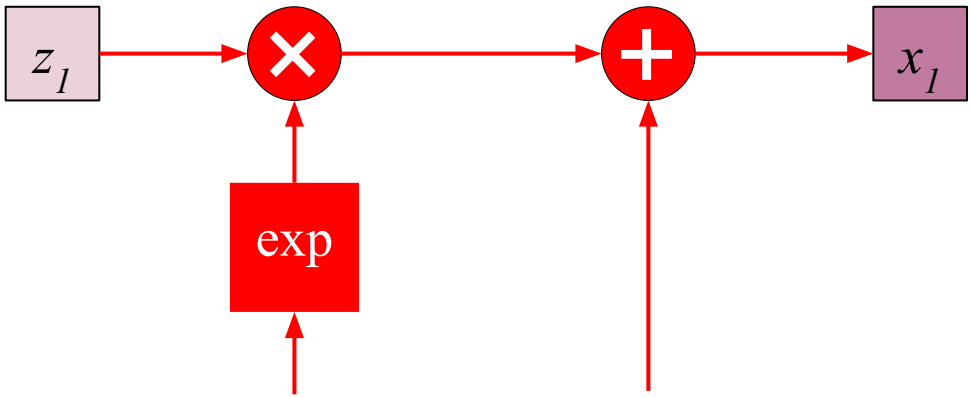
Masked Autoregressive Flows [4]



$$f_\theta$$

Figure: Forward transformation of first elment

Masked Autoregressive Flows [4]

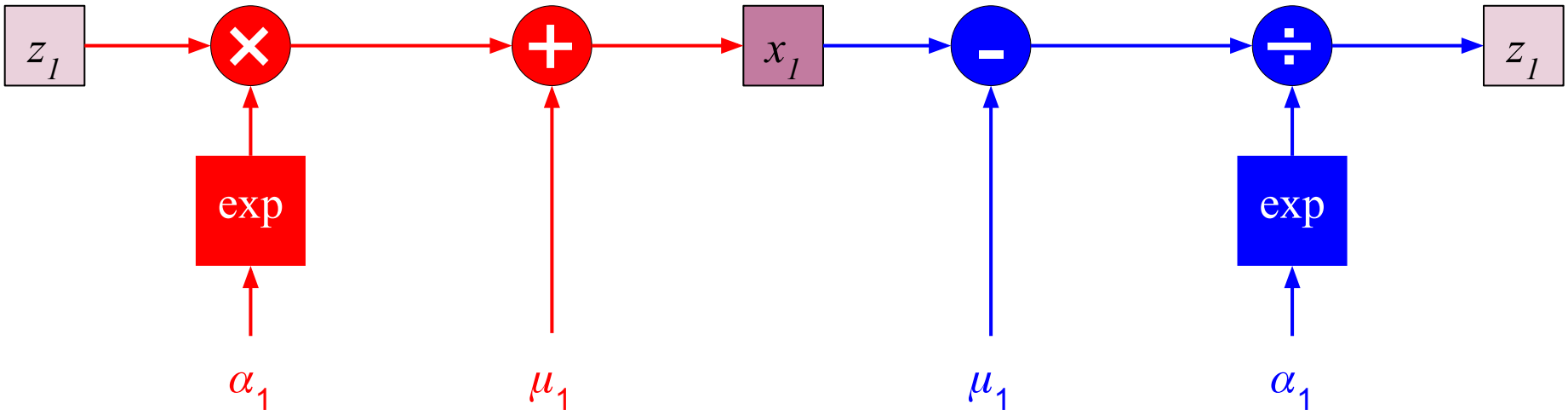


μ_1 α_1

f_θ g_θ

Figure: Inverse transformation of first elment

Masked Autoregressive Flows [4]



f_θ g_θ

Figure: Inverse transformation of first elment

Masked Autoregressive Flows [4]

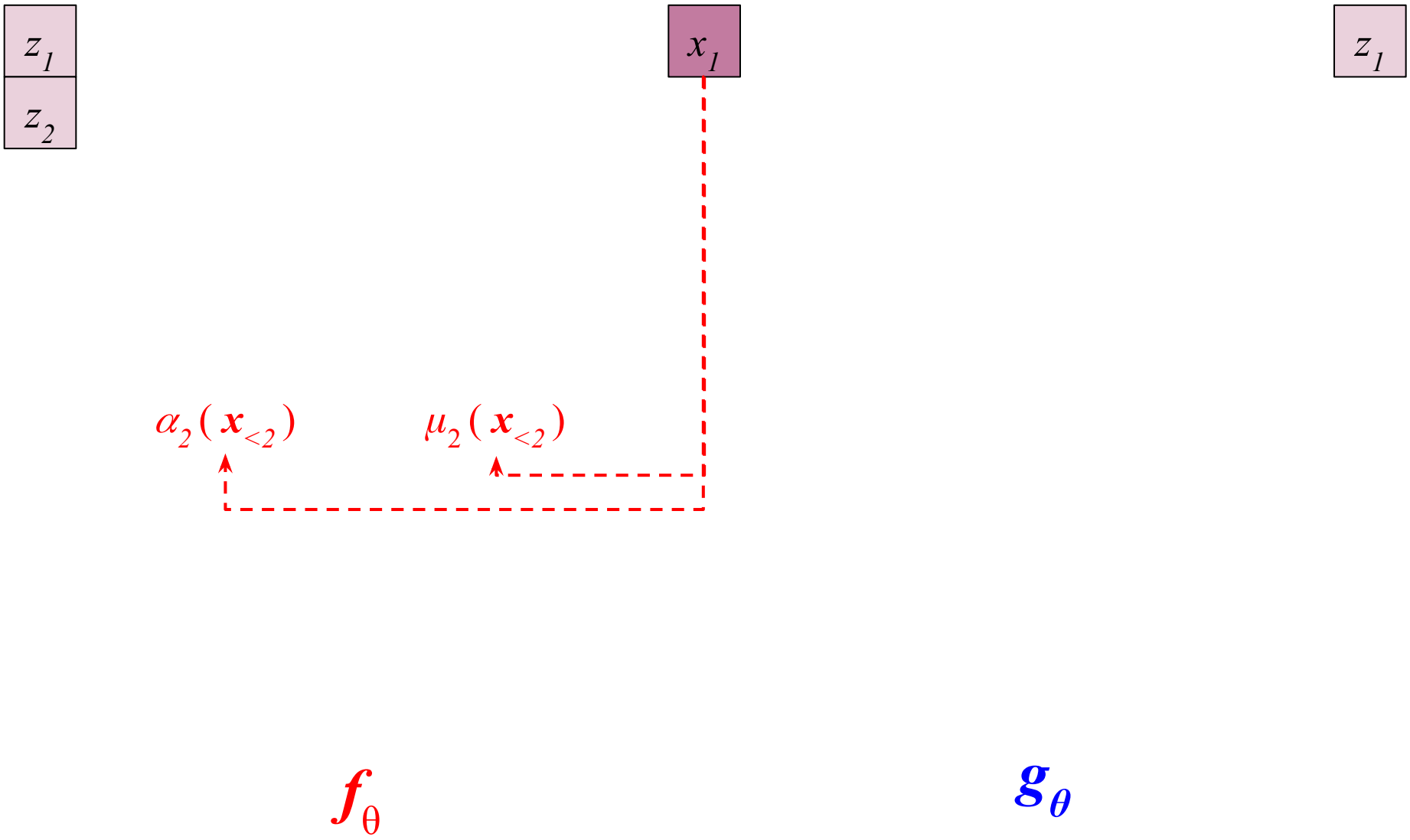
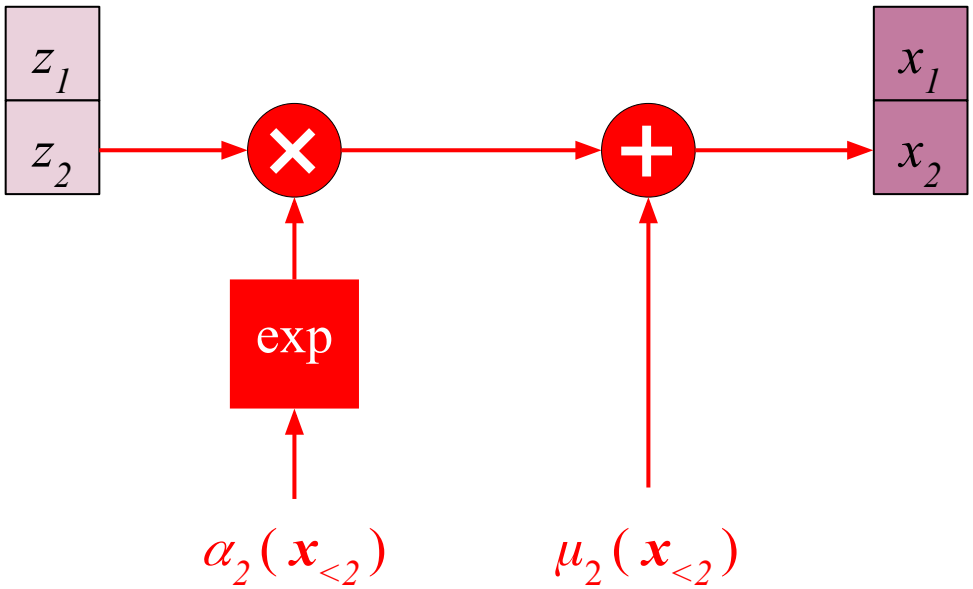


Figure: Forward transformation of second elment

Masked Autoregressive Flows [4]



$\boldsymbol{f}_\theta$ $\boldsymbol{g}_\theta$

Figure: Forward transformation of second elment

Masked Autoregressive Flows [4]

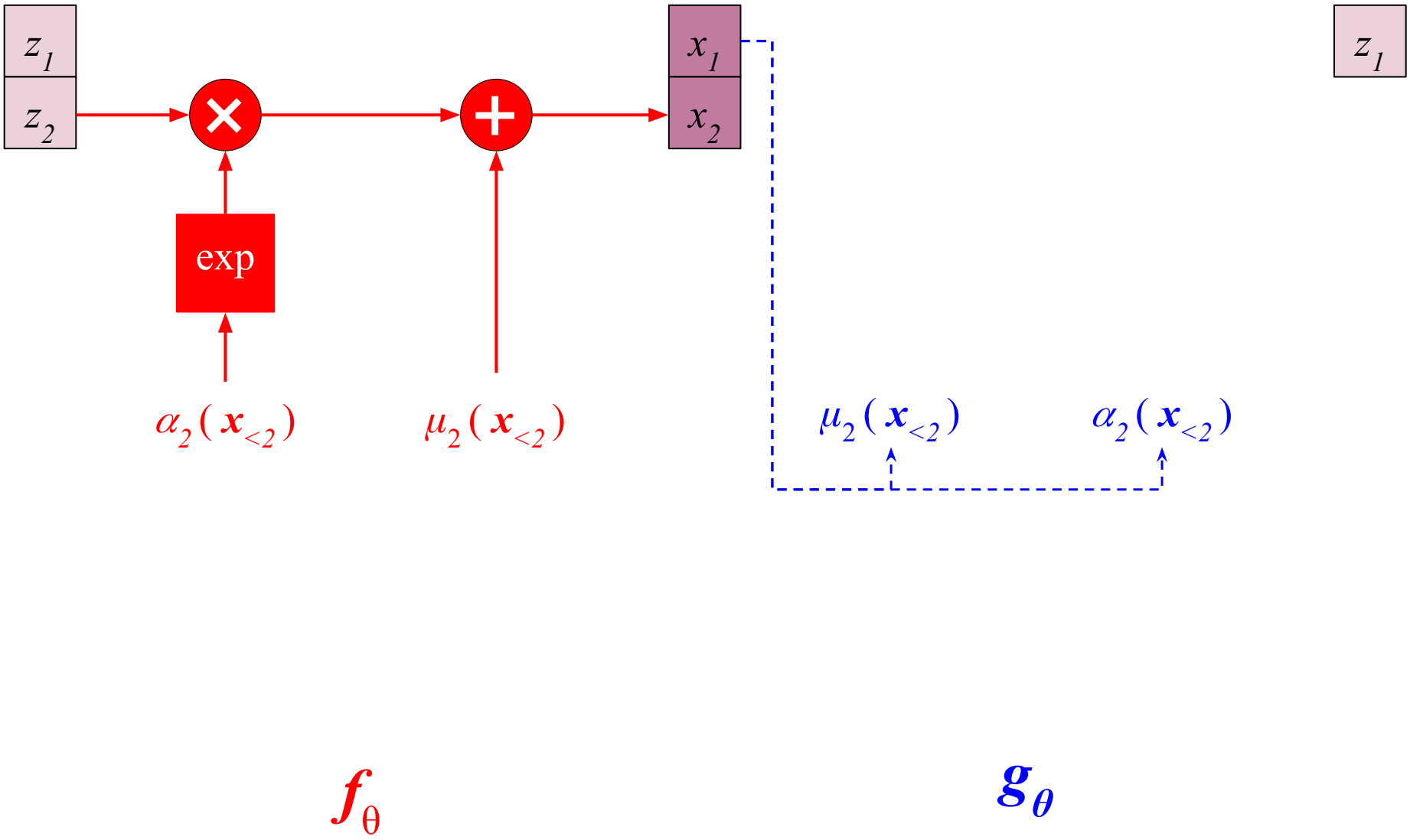
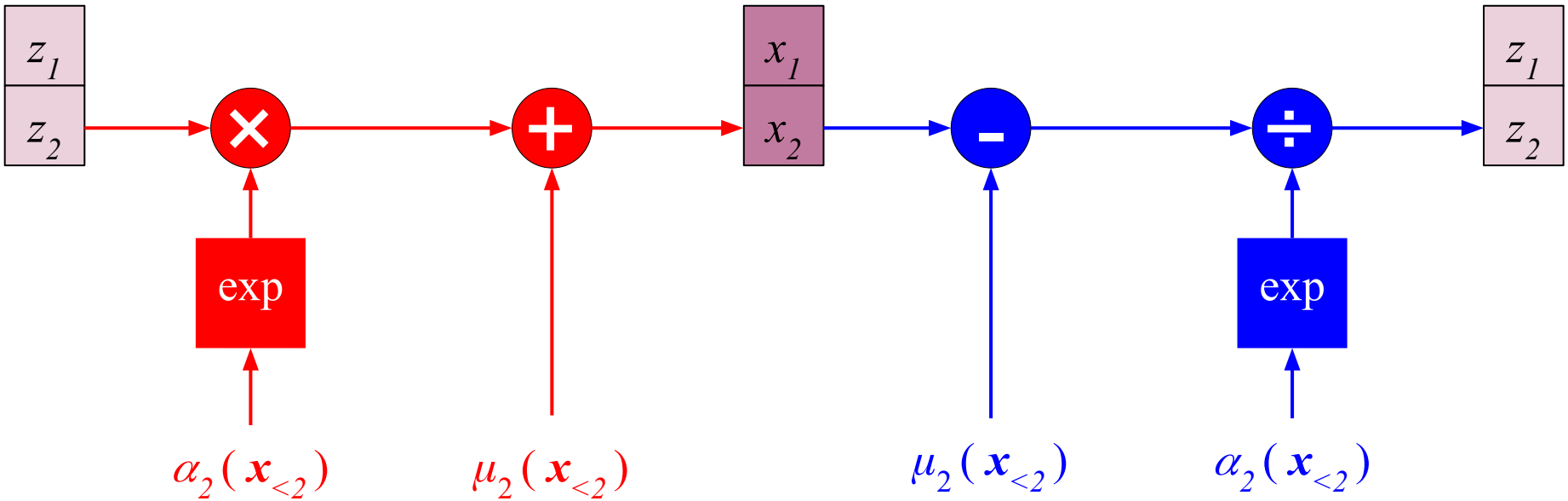


Figure: Inverse transformation of second elment

Masked Autoregressive Flows [4]



$\boldsymbol{f}_\theta$ $\boldsymbol{g}_{\boldsymbol{\theta}}$

Figure: Inverse transformation of second elment

Masked Autoregressive Flows [4]

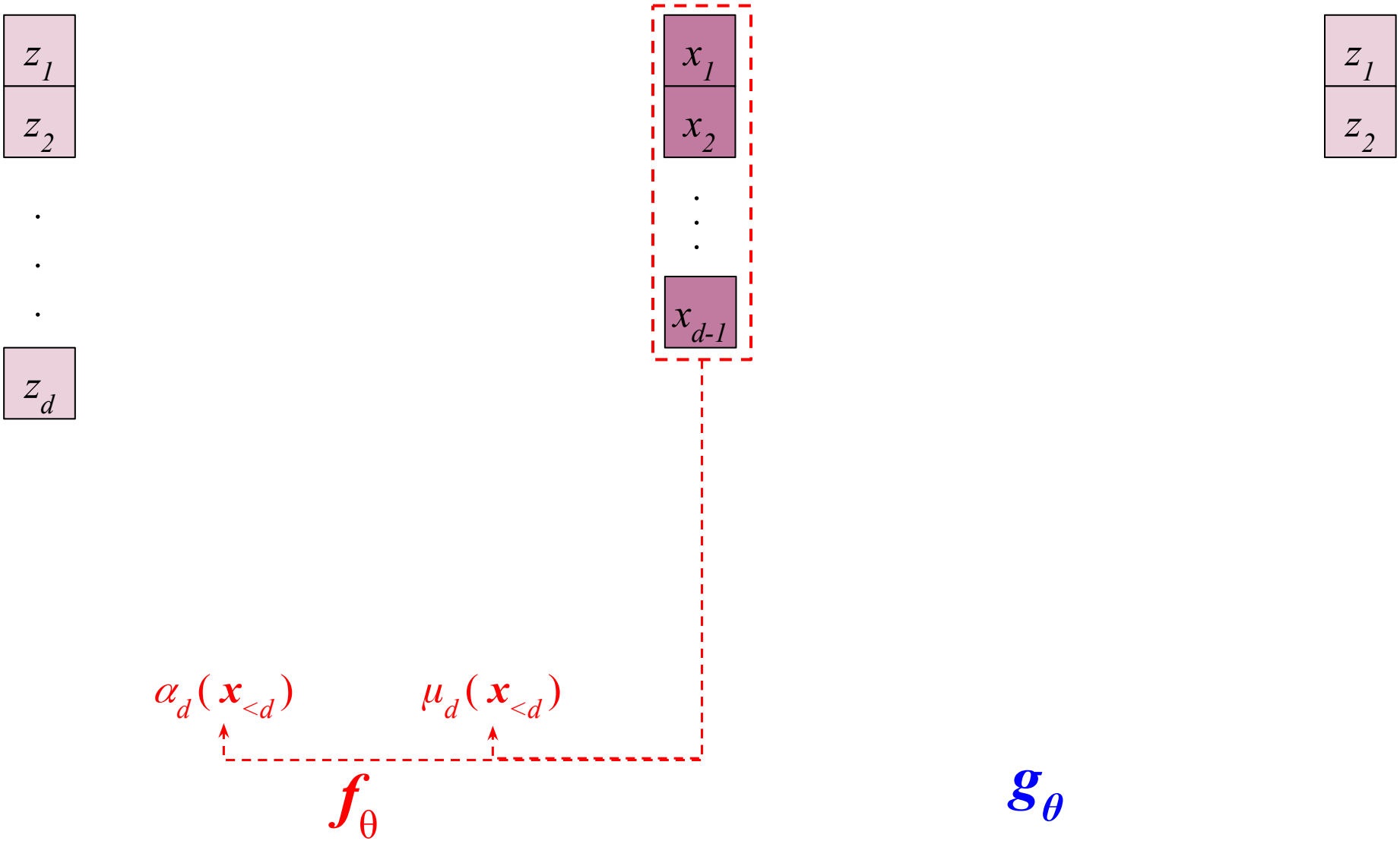


Figure: Forward transformation of d-th part

Masked Autoregressive Flows [4]

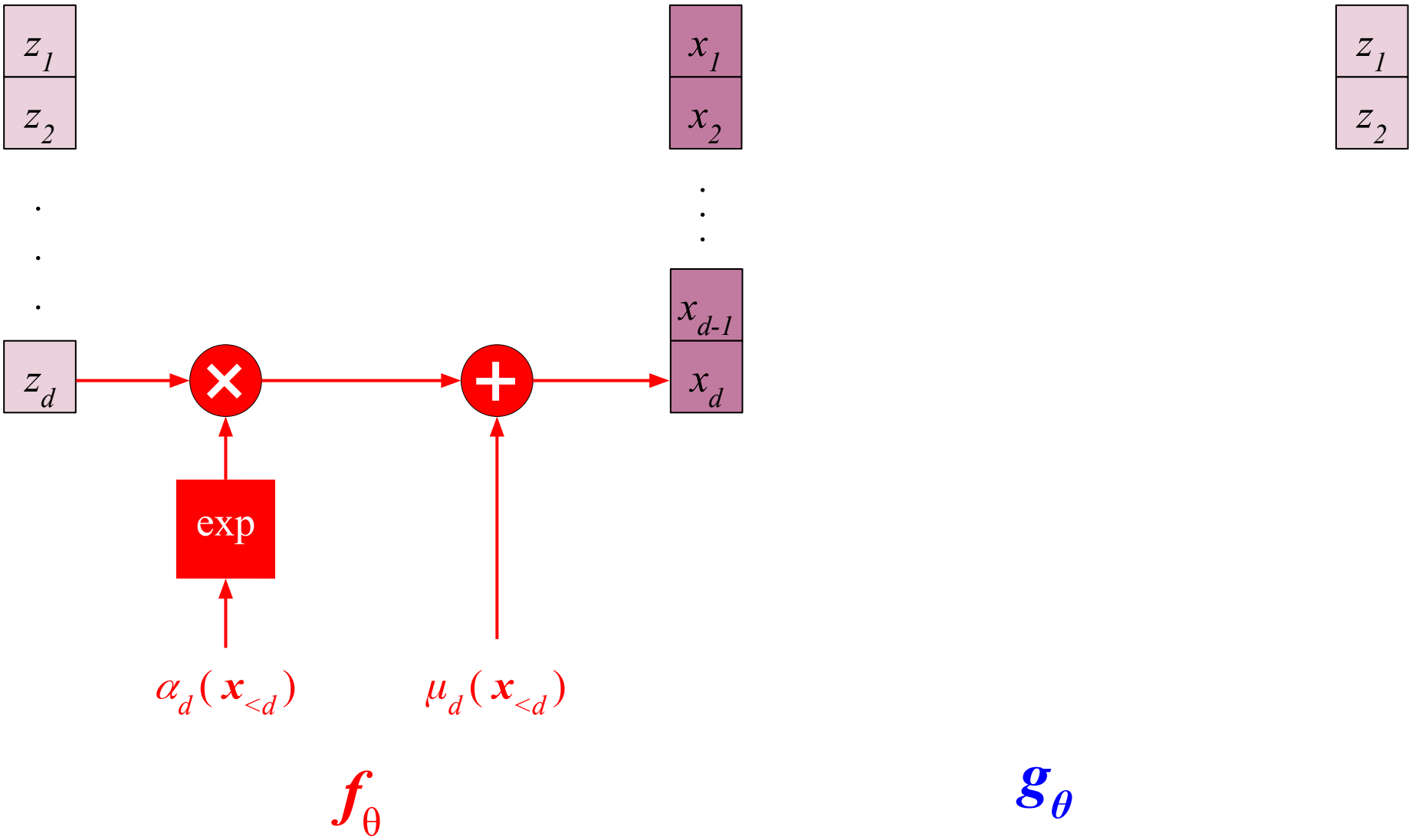


Figure: Forward transformation of d-th part

Masked Autoregressive Flows [4]

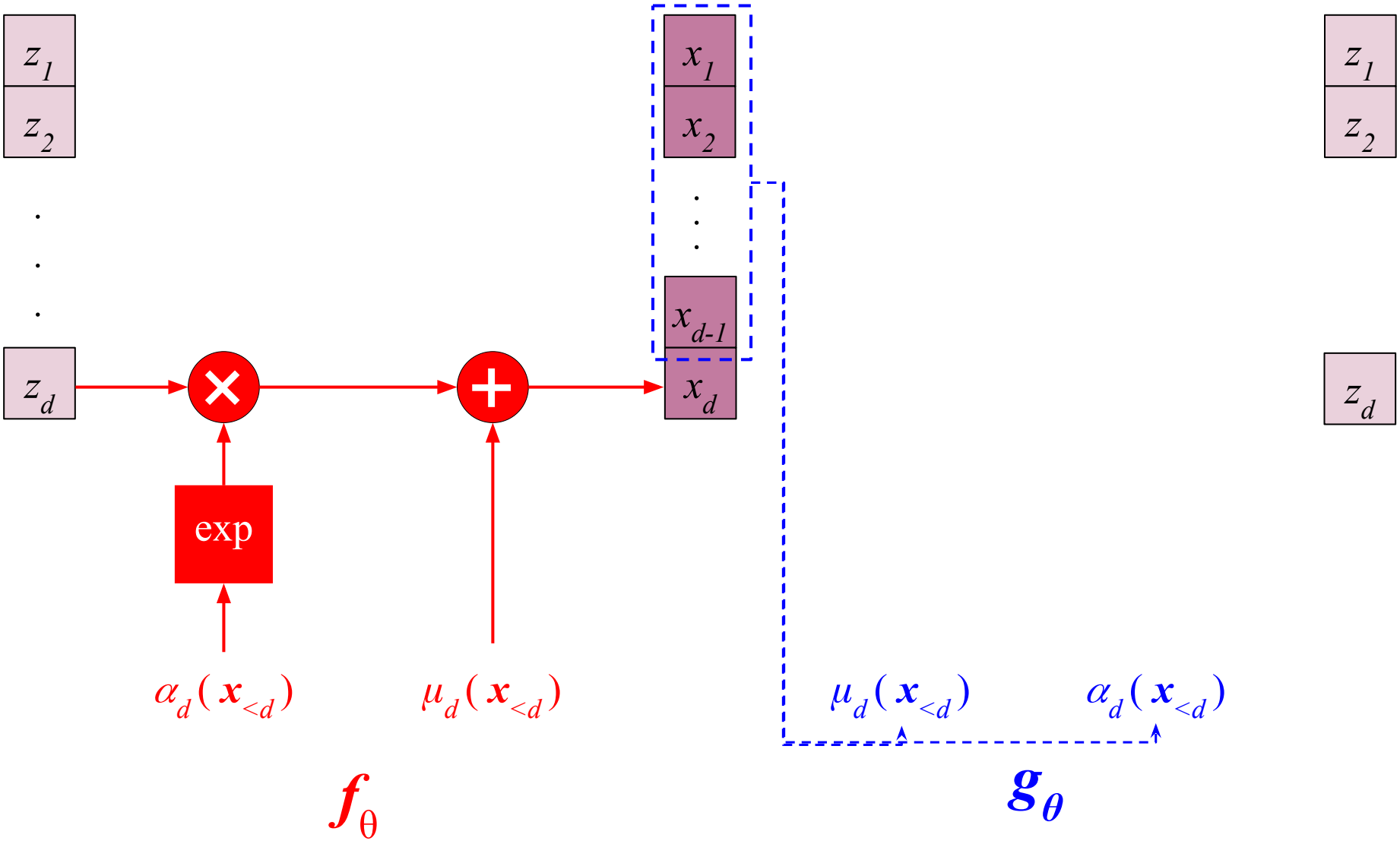


Figure: Inverse transformation of d-th part

Masked Autoregressive Flows [4]

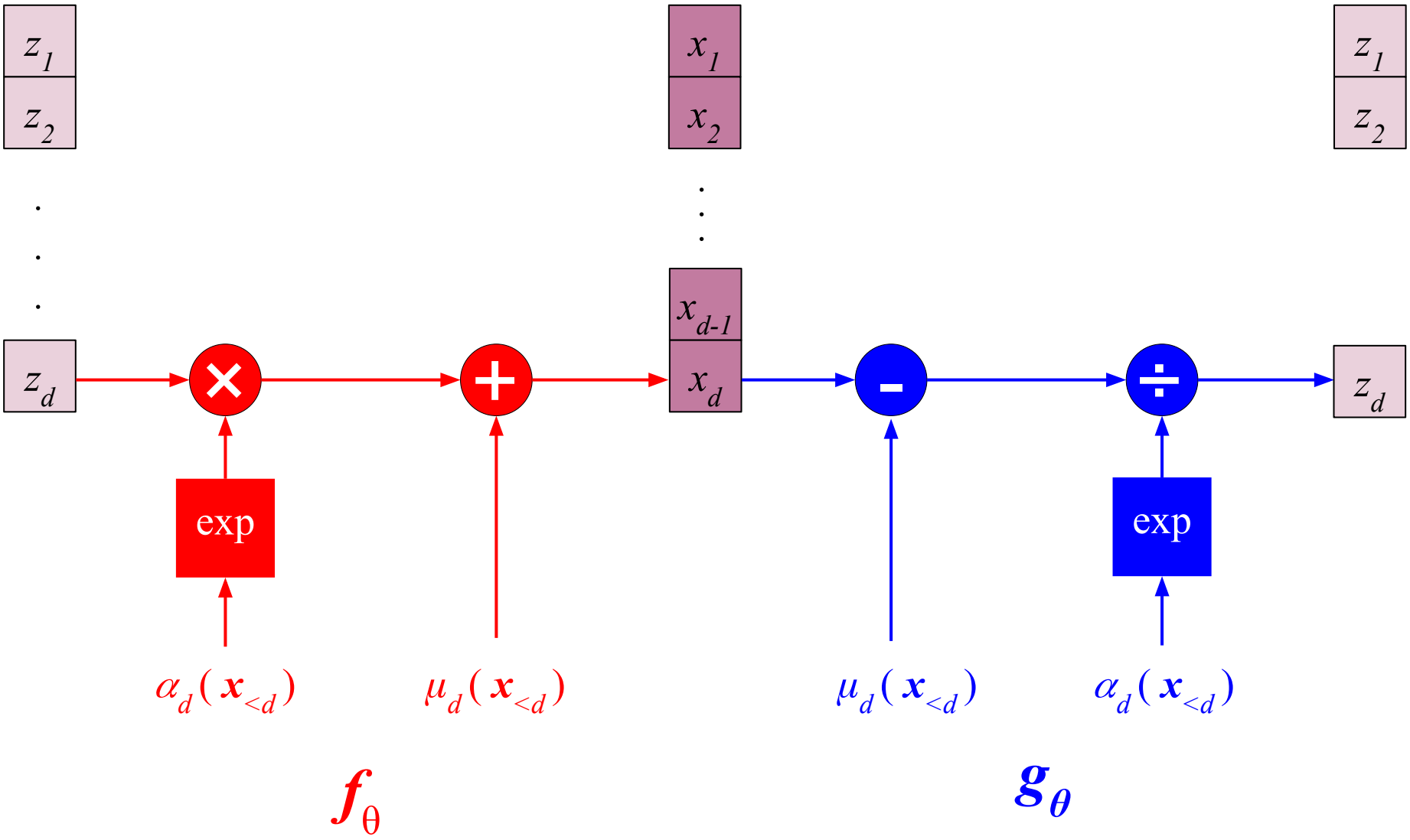


Figure: Inverse transformation of d-th part

Masked Autoregressive Flow [4]

Jacobian

The Jacobian for the forward transformation of MAF is:

$$\log|\det \boldsymbol{J}_f| = \log\left|\det \begin{bmatrix} \frac{\partial x_1}{\partial z_1} & \cdots & \frac{\partial x_1}{\partial z_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_D}{\partial z_1} & \cdots & \frac{\partial x_D}{\partial z_D} \end{bmatrix}\right|$$

$$= \log\left|\det \begin{bmatrix} \exp\left(\alpha_1(\boldsymbol{x}_{<1})\right) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \frac{\partial x_D}{\partial z_1} & \cdots & \exp\left(\alpha_D(\boldsymbol{x}_{<D})\right) \end{bmatrix}\right|$$

$$= \log\left|\prod_{d=1}^{D} \exp\left(\alpha_d(\boldsymbol{x}_{<d})\right)\right| = \sum_{d=1}^{D} \alpha_d(\boldsymbol{x}_{<d})$$

☞ The transformation is not volume preserving similar to Real-NVP.

Masked Autoregressive Flow [4]

Sampling

Sampling from an MAF can be done through a sequential procedure:

- Sample $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
- $x_1 = z_1 \exp(\alpha_1) + \mu_1$, Calculate $\begin{cases} \mu_2 \triangleq \mu_2(\boldsymbol{x}_{<2}) \\ \alpha_2 \triangleq \alpha_2(\boldsymbol{x}_{<2}) \end{cases}$
- $x_2 = z_2 \exp(\alpha_2) + \mu_2$, Calculate $\begin{cases} \mu_3 \triangleq \mu_3(\boldsymbol{x}_{<3}) \\ \alpha_3 \triangleq \alpha_3(\boldsymbol{x}_{<3}) \end{cases}$

$\vdots$

- $x_D = z_D \exp(\alpha_D) + \mu_D$

Sequential Sampling

The sampling is sequential in MAF as we expect from the autoregressive nature.

Masked Autoregressive Flow [4]

Likelihood Calculation

Probability calculations can be done in parallel as:

- Given $\boldsymbol{x}$ vector, calculate the following vectors (elements can be calculated in parallel):

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1(\boldsymbol{x}_{<1}) \\ \mu_2(\boldsymbol{x}_{<2}) \\ \vdots \\ \mu_D(\boldsymbol{x}_{<D}) \end{bmatrix}, \quad \boldsymbol{\alpha} = \begin{bmatrix} \alpha_1(\boldsymbol{x}_{<1}) \\ \alpha_2(\boldsymbol{x}_{<2}) \\ \vdots \\ \alpha_D(\boldsymbol{x}_{<D}) \end{bmatrix}$$

- Calculate $\boldsymbol{z} = \frac{\boldsymbol{x}-\boldsymbol{\mu}}{\exp(\boldsymbol{\alpha})}$ (all the operations are assumed element-wise)
- Calculate probabilty using $p_\theta(\boldsymbol{x}) = p(\boldsymbol{z}) \left| \prod_{d=1}^{D} \exp\left(\alpha_d(\boldsymbol{x}_{<d})\right) \right|^{-1}$

Parallel Likelihood

The likelihood calculation and thus learning which needs transformation Jacobian is in parallel for MAF.

Conditional Modeling [4]

Conditional Modeling

Assume we are interested in modeling the conditional distribution $p_\theta(\mathbb{X}|\boldsymbol{y})$. Then we can use our previous disscussion about unconditional distribution while pay attention to:

- $\boldsymbol{y}$ is not a random variable (such as $\mathbb{X}$ or latent random variable $\mathbb{Z}$)
- $\boldsymbol{y}$ is given. So we must design invertible transformation assuming you have access to $\boldsymbol{y}$ in both forward and inverse transformations.

Conditional Modeling [4]

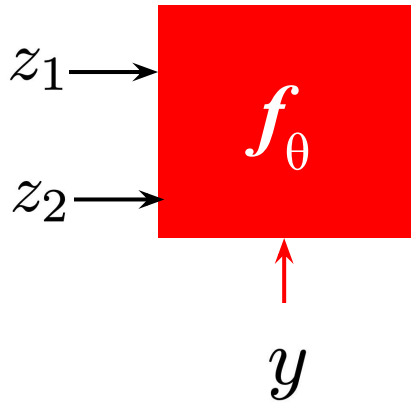


Figure: Forward transformation

Conditional Modeling [4]

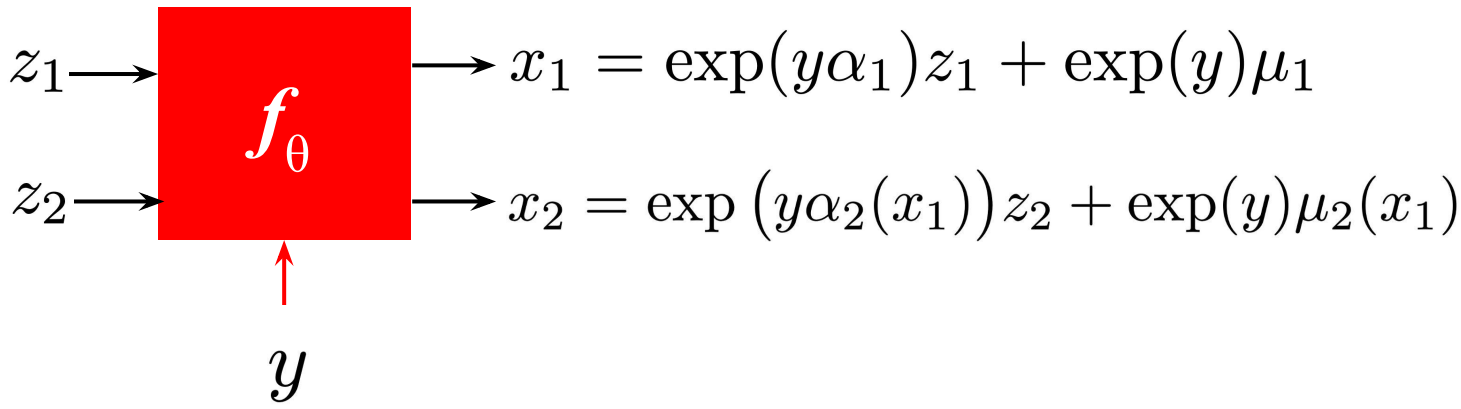


Figure: Forward transformation

Conditional Modeling [4]

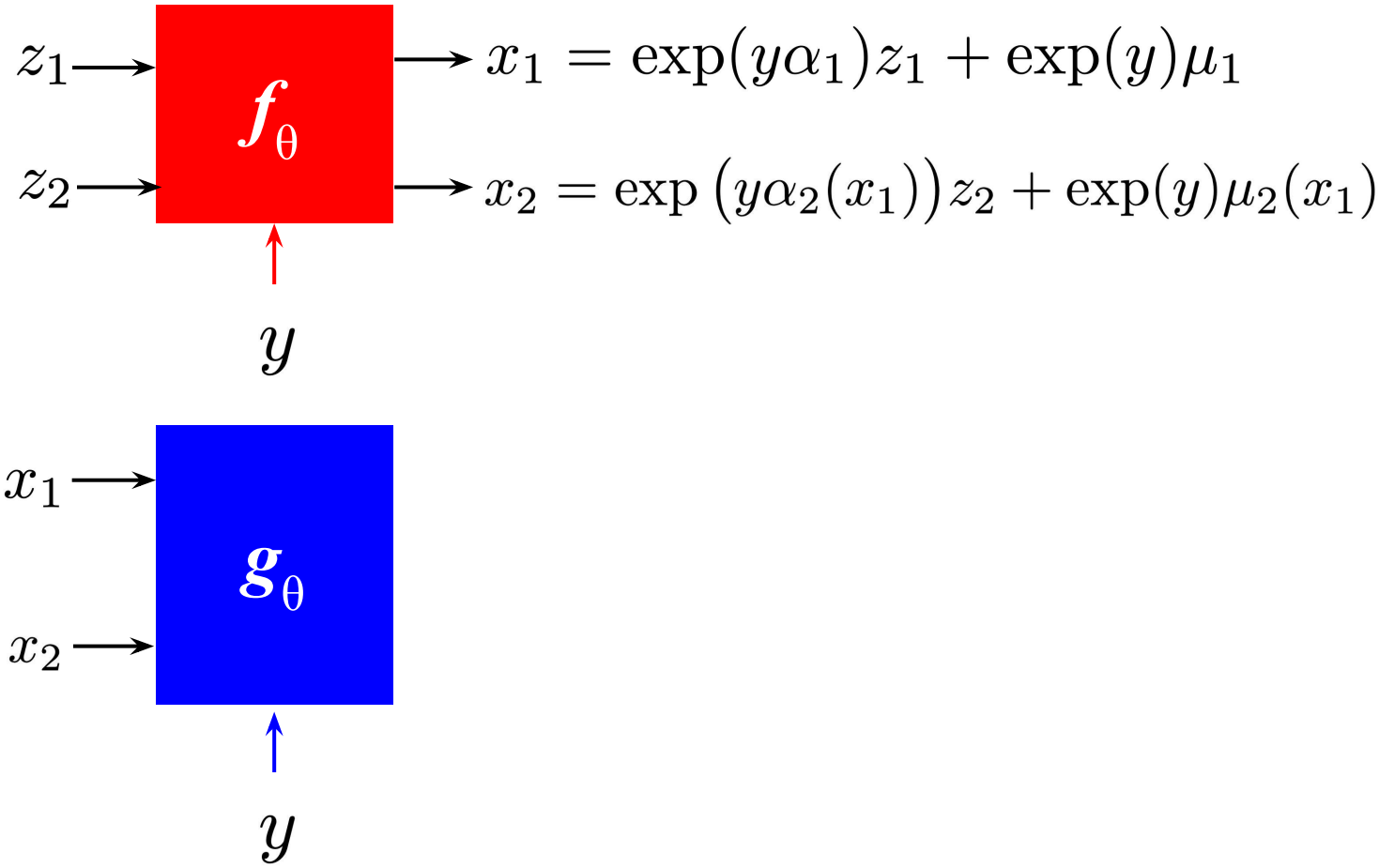


Figure: Inverse transformation

Conditional Modeling [4]

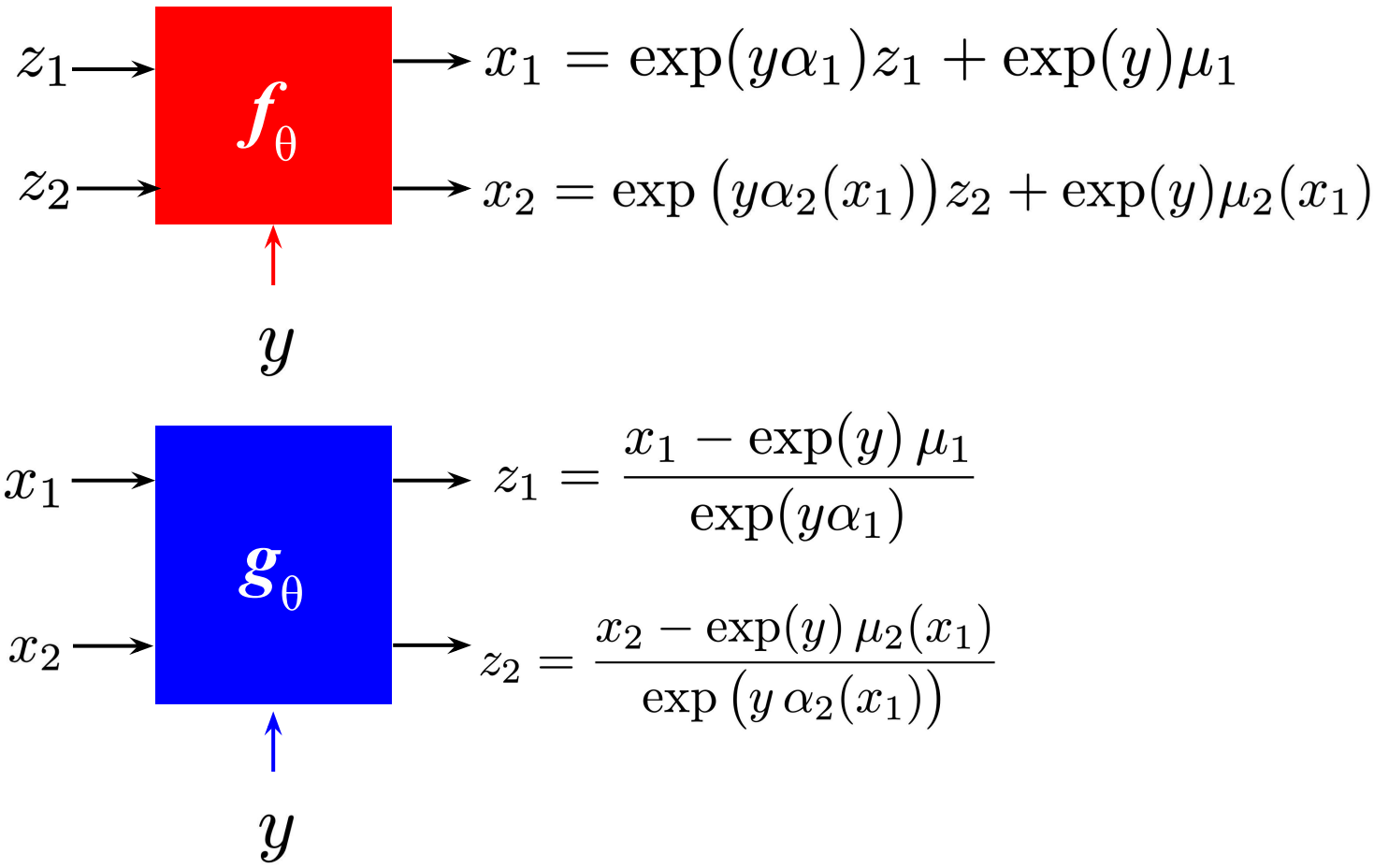
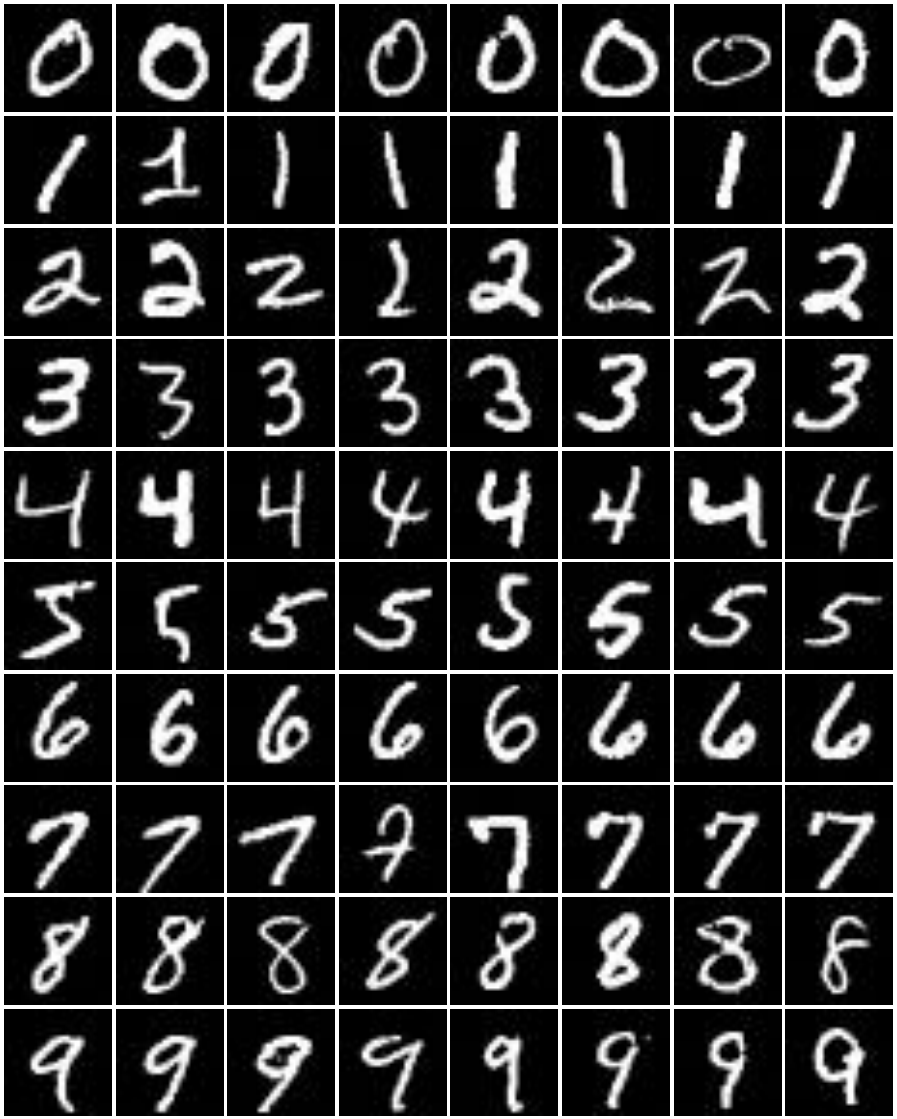
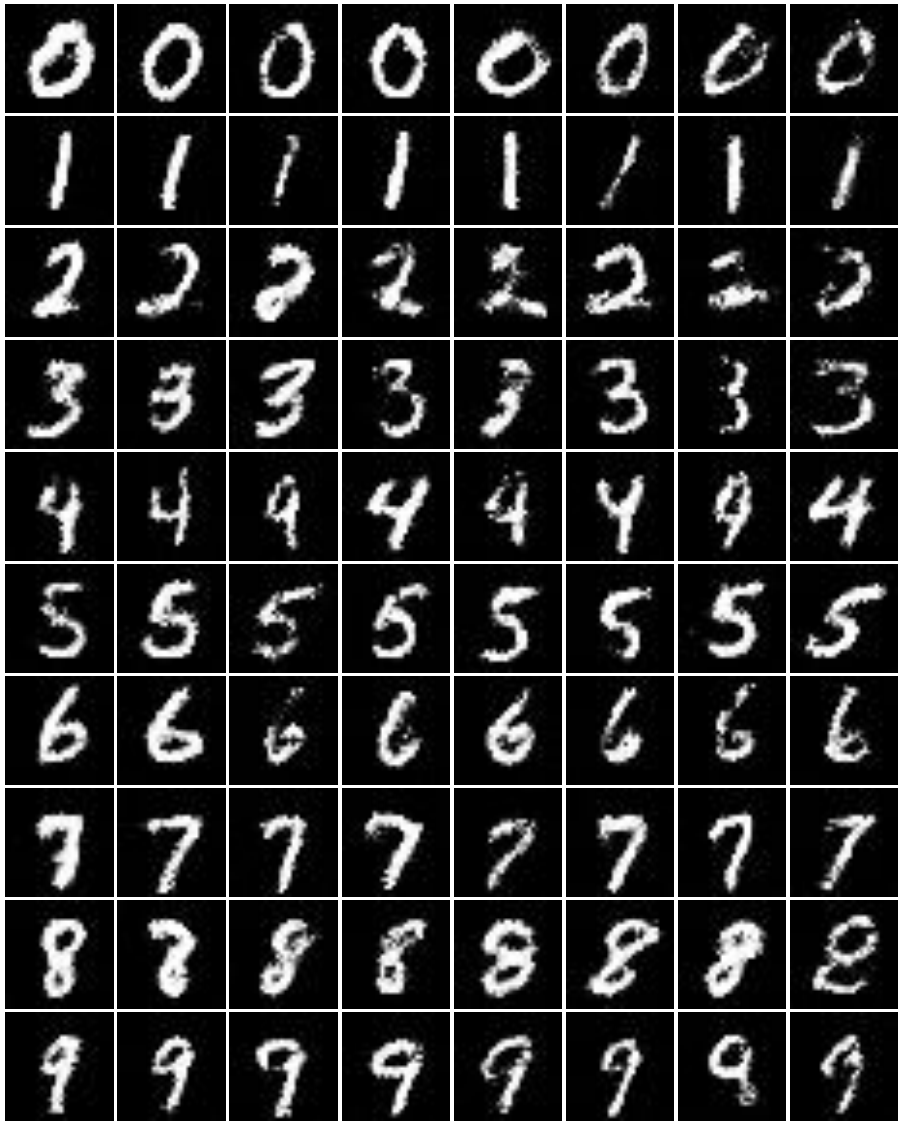


Figure: Inverse transformation

Class Conditional Generation (source: [4])



(a) Dataset samples



(b) Class-conditional samples

Section 8

Inverse Autoregressive Flows

Inverse Autoregressive Flows [5]

Forward Transformation

$\boldsymbol{f}_\theta : \boldsymbol{z} \to \boldsymbol{x}$: Assume $\boldsymbol{z}, \boldsymbol{x} \in \mathbb{R}^D$, then the forward transformation is defined as:

$$x_d = z_d \times \exp(\alpha_d(\boldsymbol{z}_{<d})) + \mu_d(\boldsymbol{z}_{<d})$$

$$\Rightarrow \begin{cases} d = 1: & x_1 = z_1 \times \exp(\alpha_1(\boldsymbol{z}_{<1})) + \mu_1(\boldsymbol{z}_{<1}) = z_1 \times \exp(\alpha_1) + \mu_1 \\ d = 2: & x_2 = z_2 \times \exp(\alpha_2(\boldsymbol{z}_{<2})) + \mu_2(\boldsymbol{z}_{<2}) = z_2 \times \exp(\alpha_2(z_1)) + \mu_2(z_1) \\ \vdots & \end{cases}$$

where $\alpha_d(\cdot)$ and $\mu_d(\cdot)$ can be any neural network architecutre with single output.

Inverse Autoregressive Flows [5]

Inverse Transformation

$\boldsymbol{g}_\theta : \boldsymbol{x} \to \boldsymbol{z}$: Using the forward transformation, one can easily show that:

$$z_d = \frac{x_d - \mu_d(\boldsymbol{z}_{<d})}{\exp(\alpha_d(\boldsymbol{z}_{<d}))}$$

$$\Rightarrow \begin{cases} d=1: & z_1 = \frac{x_1 - \mu_1(\boldsymbol{z}_{<1})}{\exp(\alpha_1(\boldsymbol{z}_{<d}))} = \frac{x_1 - \mu_1}{\exp(\alpha_1)} \\ d=2: & z_2 = \frac{x_2 - \mu_2(\boldsymbol{z}_{<2})}{\exp(\alpha_2(\boldsymbol{z}_{<2}))} = \frac{x_1 - \mu_2(z_1)}{\exp(\alpha_2(z_1))} \\ \vdots & \end{cases}$$

Inverse Autoregressive Flows [5]

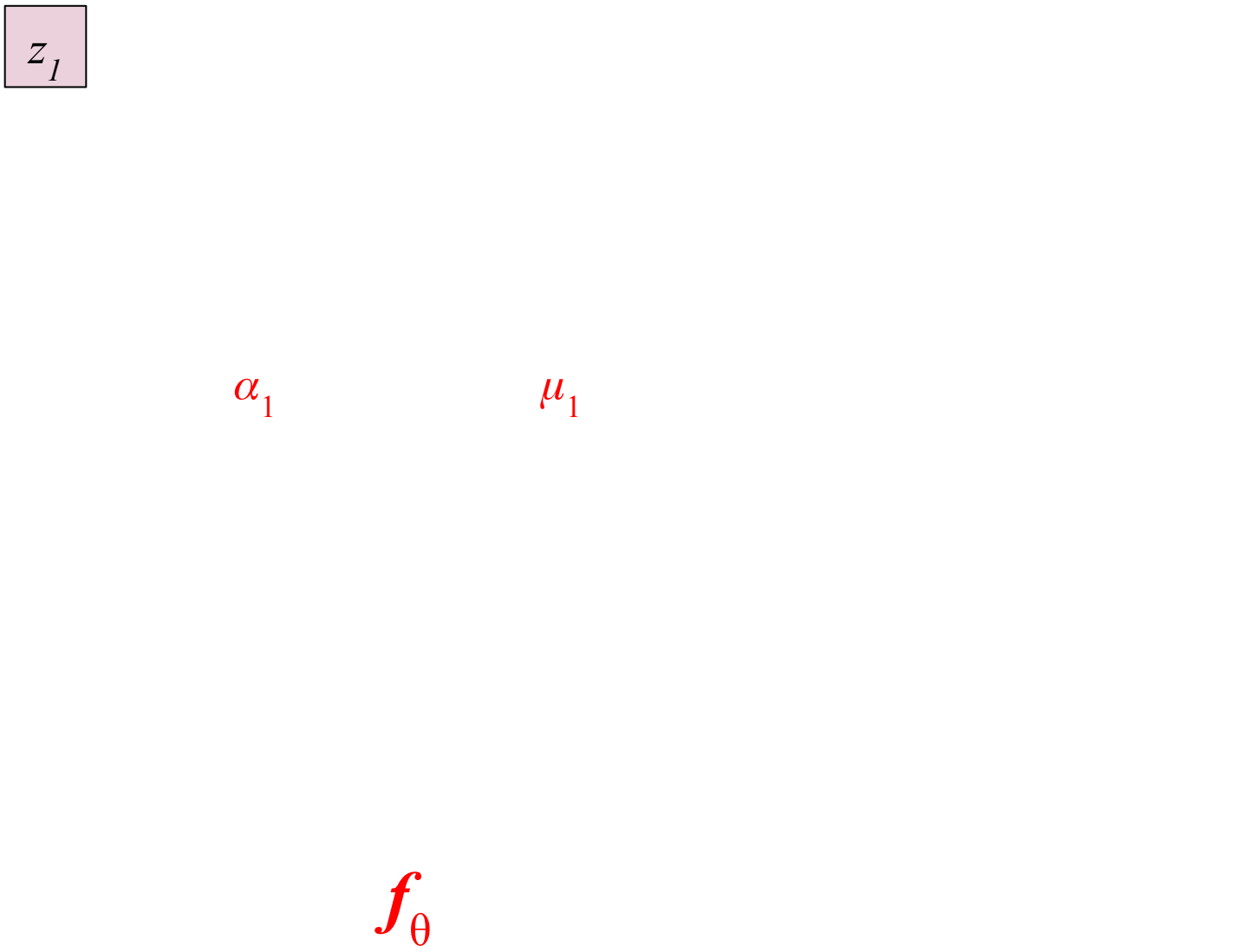
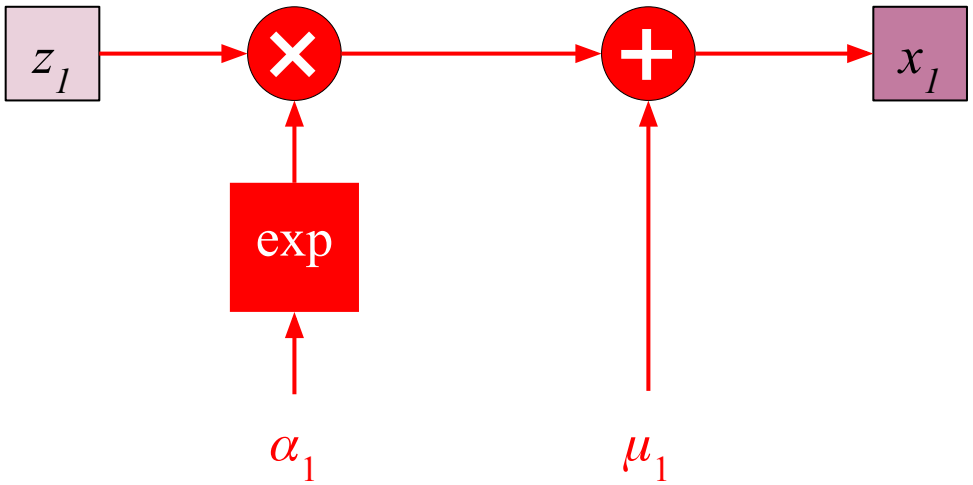


Figure: Forward transformation of first elment

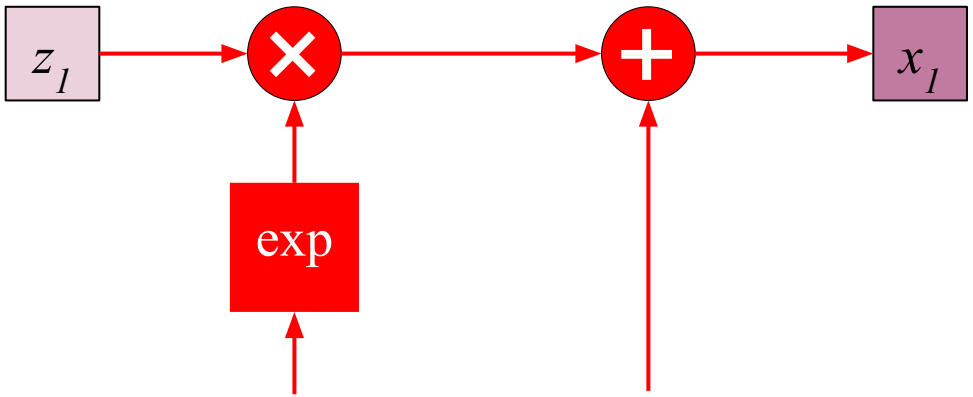
Inverse Autoregressive Flows [5]



$$\boldsymbol{f}_\theta$$

Figure: Forward transformation of first elment

Inverse Autoregressive Flows [5]

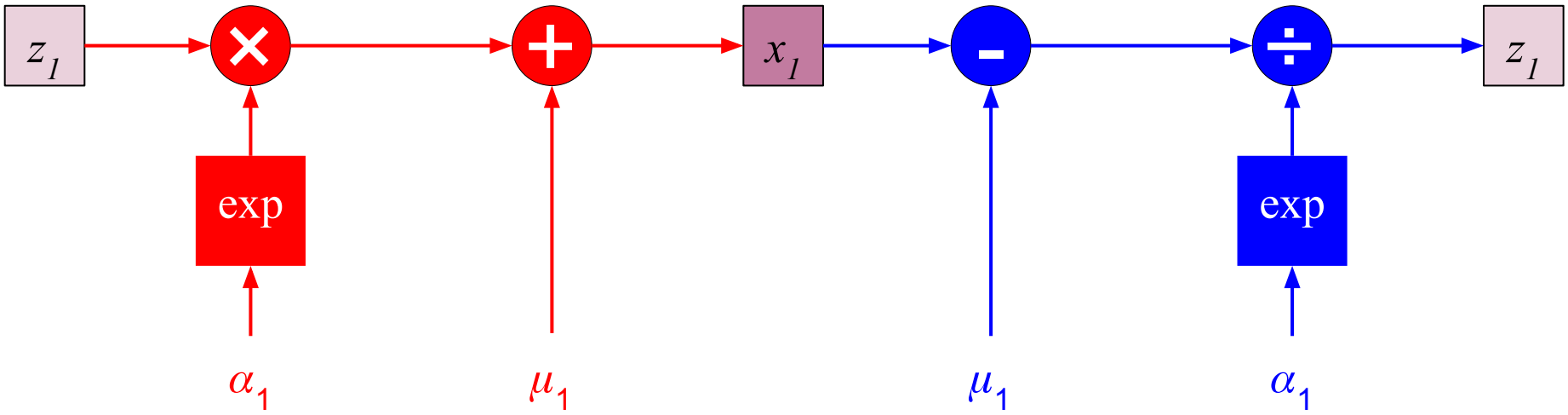


μ_1 α_1

f_θ g_θ

Figure: Inverse transformation of first elment

Inverse Autoregressive Flows [5]



f_θ g_θ

Figure: Inverse transformation of first elment

Inverse Autoregressive Flows [5]

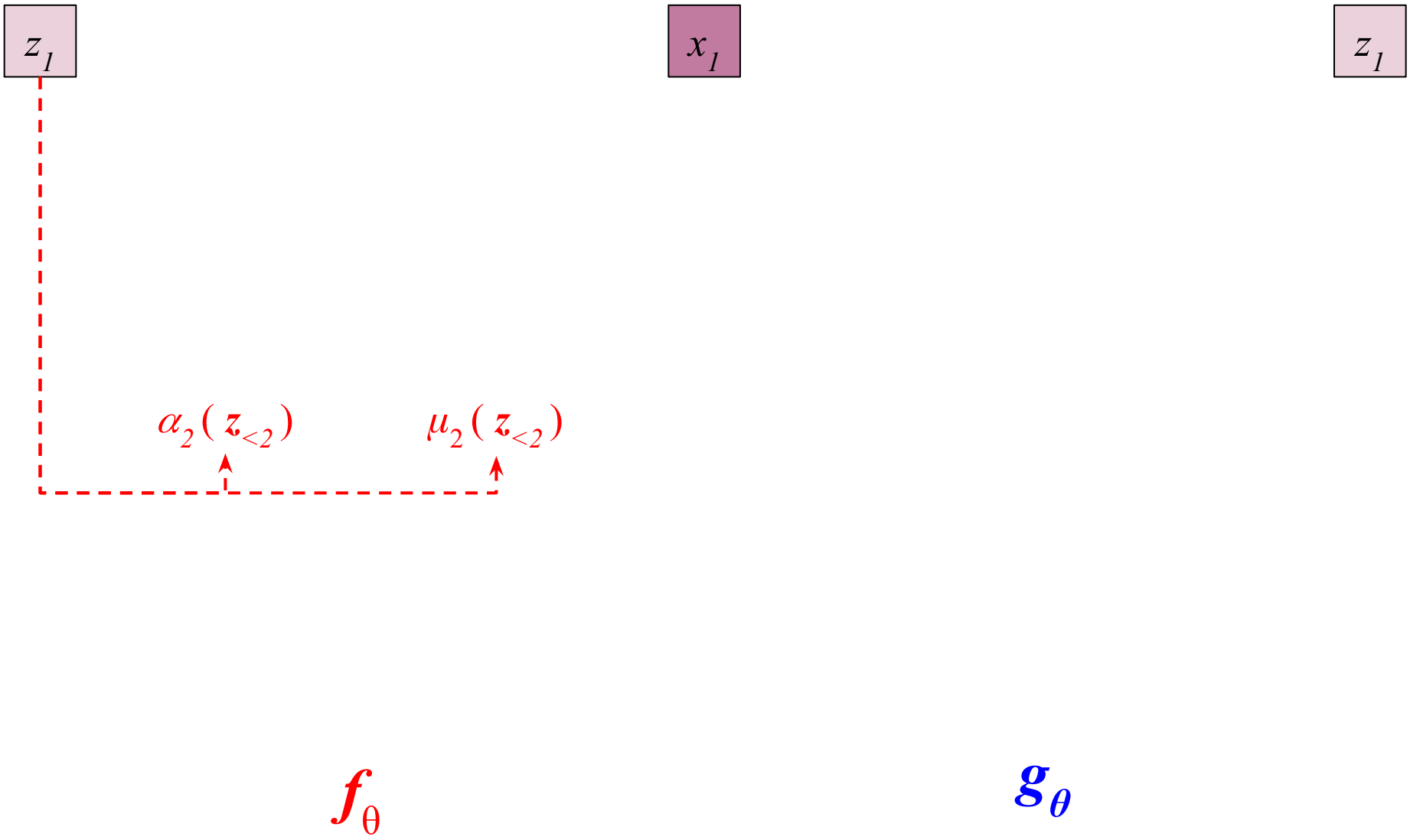
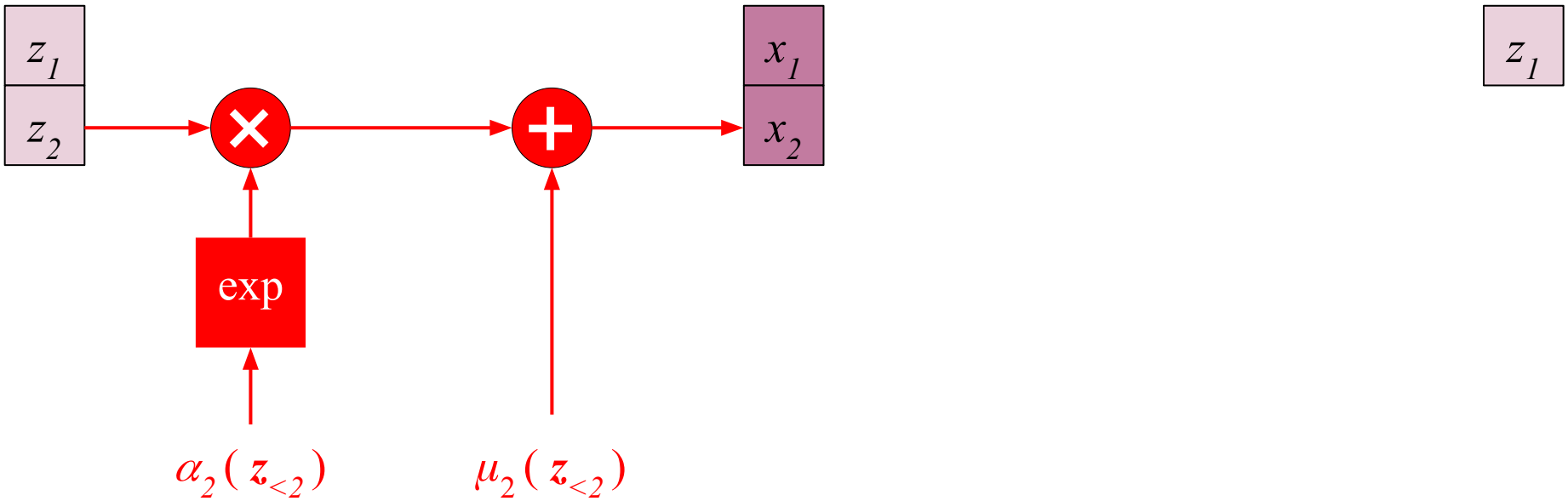


Figure: Forward transformation of second elment

Inverse Autoregressive Flows [5]



f_θ g_θ

Figure: Forward transformation of second elment

Inverse Autoregressive Flows [5]

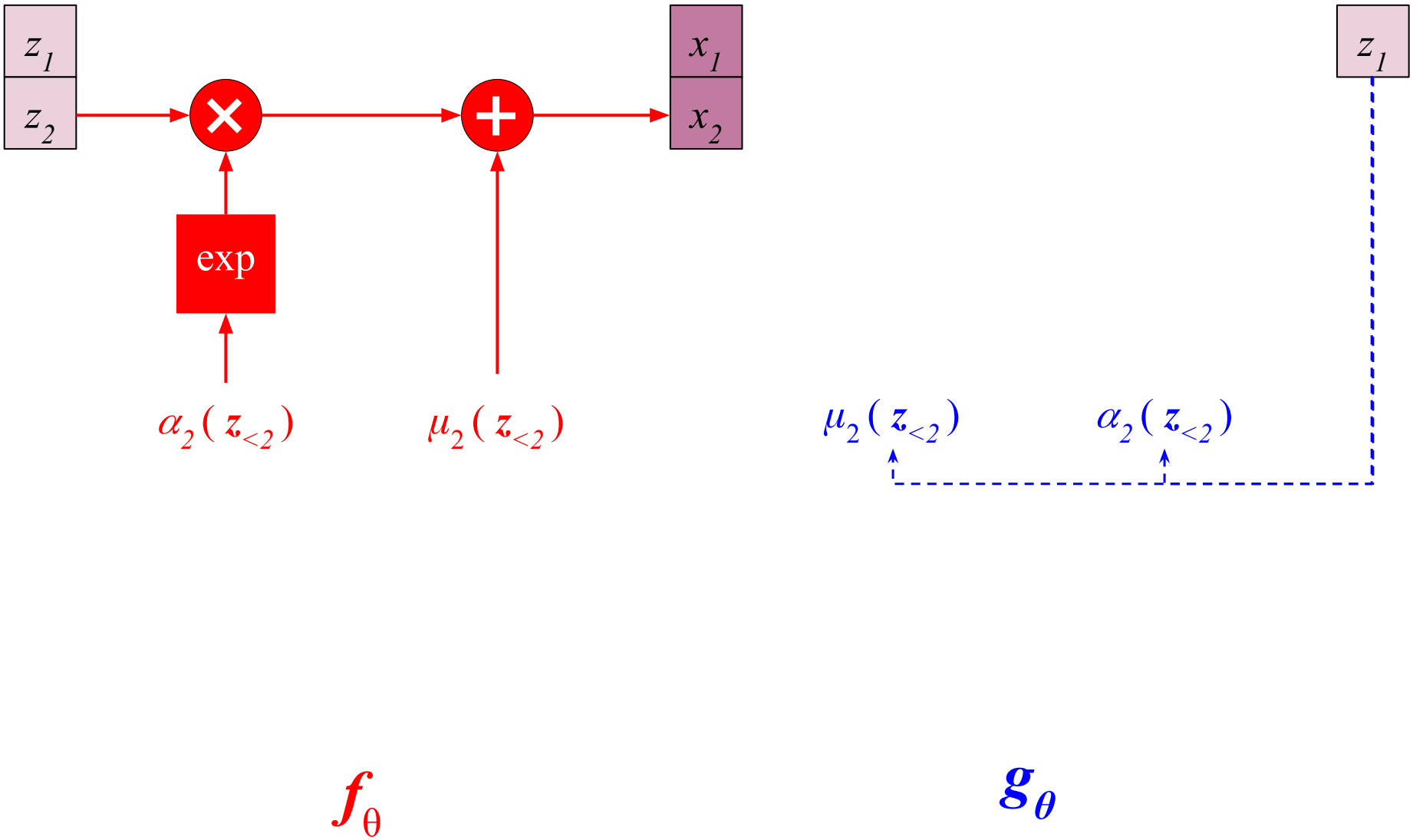
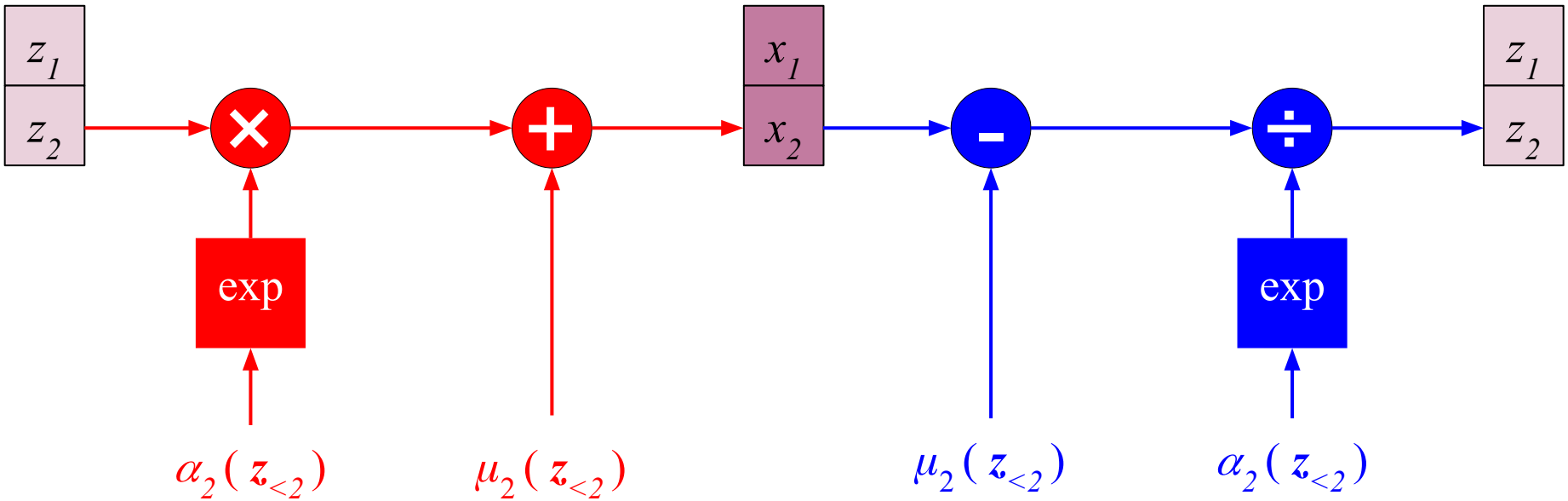


Figure: Inverse transformation of second elment

Inverse Autoregressive Flows [5]



f_θ g_θ

Figure: Inverse transformation of second elment

Inverse Autoregressive Flows [5]

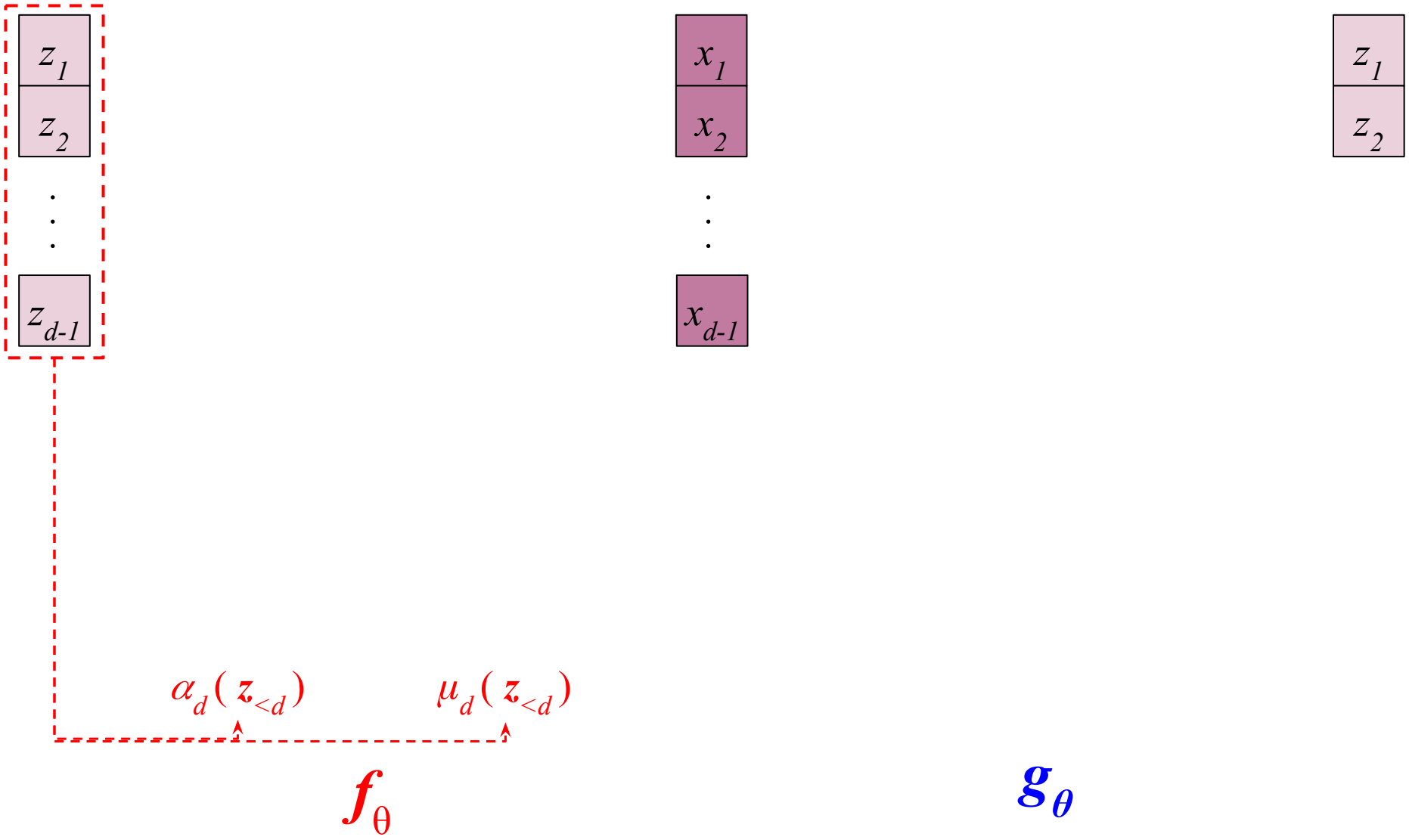


Figure: Forward transformation of d-th part

Inverse Autoregressive Flows [5]

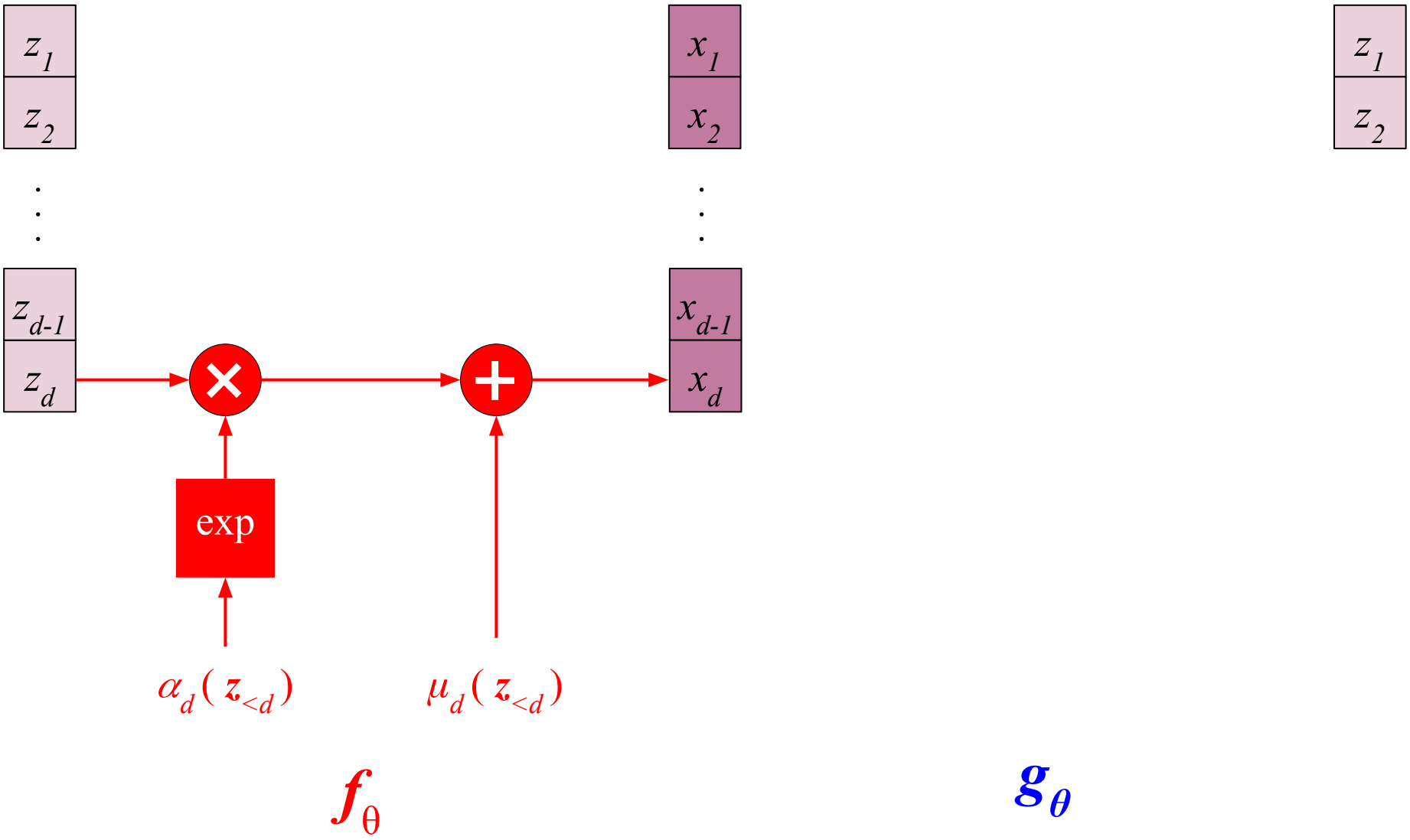


Figure: Forward transformation of d-th part

Inverse Autoregressive Flows [5]

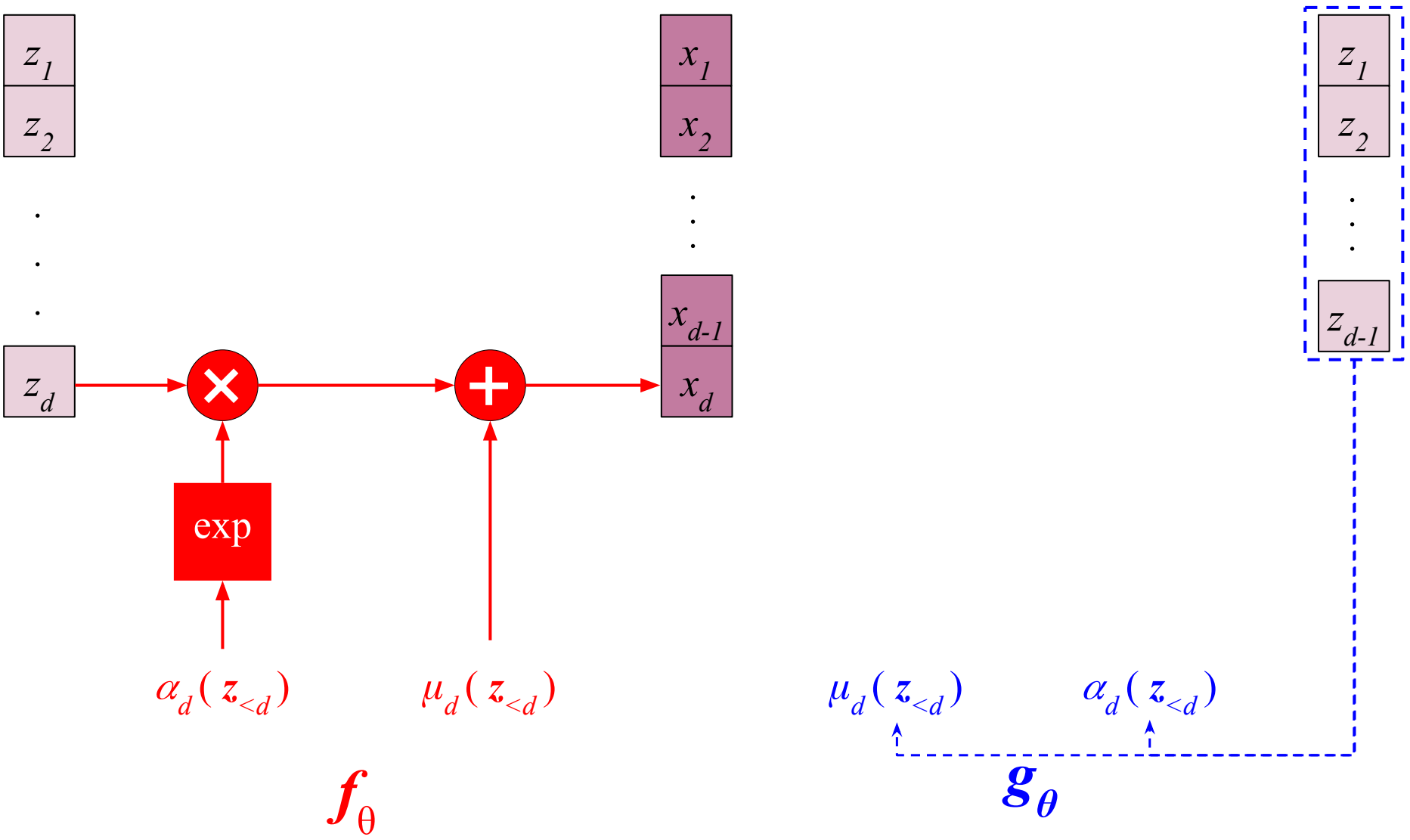


Figure: Inverse transformation of d-th part

Inverse Autoregressive Flows [5]

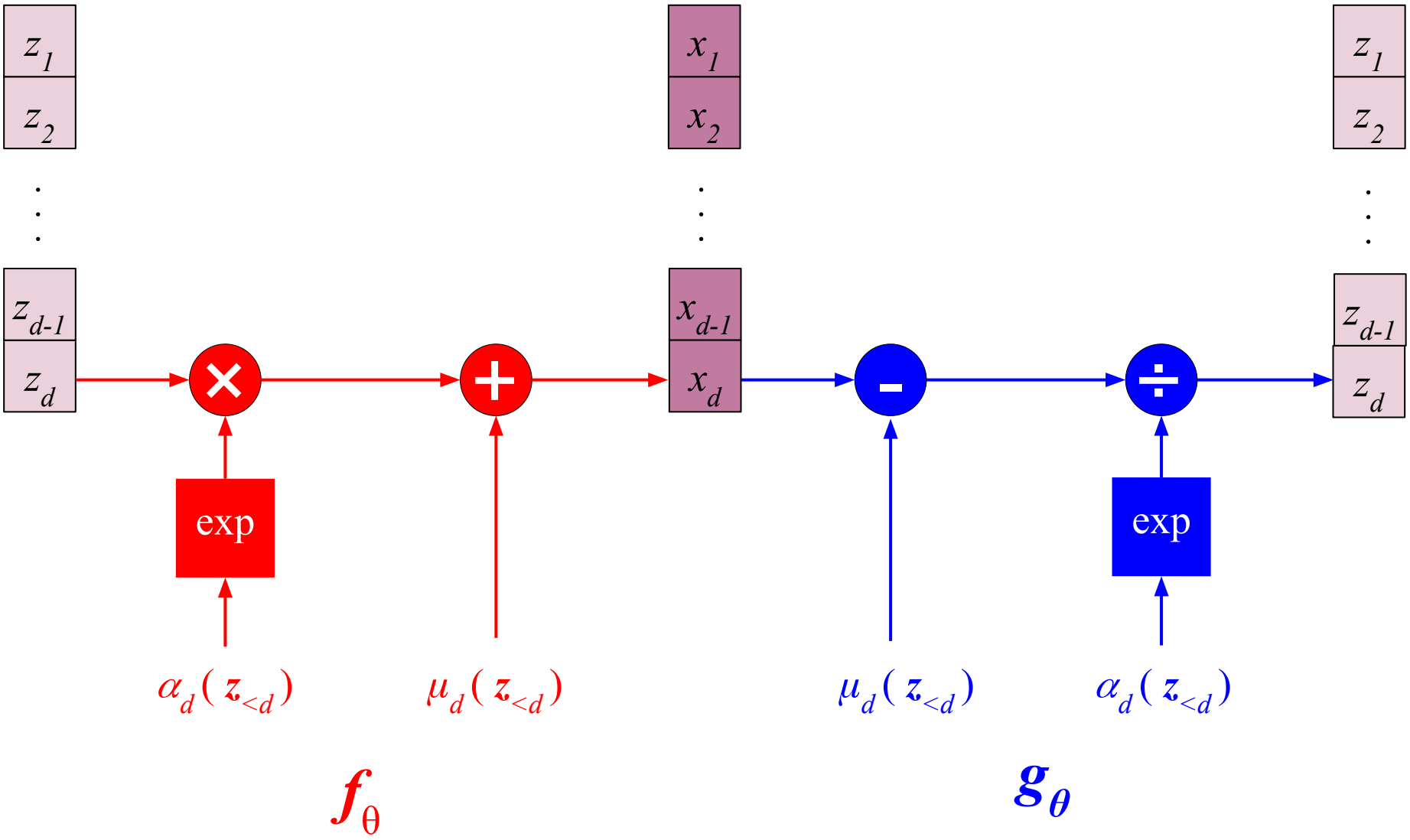


Figure: Inverse transformation of d-th part

Inverse Autoregressive Flow [5]

Jacobian

The Jacobian for the forward transformation of IAF is:

$$\log|\det \boldsymbol{J}_f| = \log\left|\det\begin{bmatrix}\frac{\partial x_1}{\partial z_1} & \cdots & \frac{\partial x_1}{\partial z_D}\\ \vdots & \ddots & \vdots \\ \frac{\partial x_D}{\partial z_1} & \cdots & \frac{\partial x_D}{\partial z_D}\end{bmatrix}\right|$$

$$= \log\left|\det\begin{bmatrix}\exp(\alpha_1(\boldsymbol{z}_{<1})) & \cdots & 0\\ \vdots & \ddots & \vdots \\ \frac{\partial x_D}{\partial z_1} & \cdots & \exp(\alpha_D(\boldsymbol{z}_{<D}))\end{bmatrix}\right|$$

$$= \log\left|\prod_{d=1}^{D}\exp(\alpha_d(\boldsymbol{z}_{<d}))\right| = \sum_{d=1}^{D}\alpha_d(\boldsymbol{z}_{<d})$$

- ☞ The transformation is not volume preserving similar to Real-NVP and MAF.

Inverse Autoregressive Flow [5]

Sampling

Sampling from an IAF can be done in parallel as:

- Sample $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
- Calculate the following vectors (elements can be calculated in parallel)

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1(\boldsymbol{z}_{<1}) \\ \mu_2(\boldsymbol{z}_{<2}) \\ \vdots \\ \mu_D(\boldsymbol{z}_{<D}) \end{bmatrix}, \quad \boldsymbol{\alpha} = \begin{bmatrix} \alpha_1(\boldsymbol{z}_{<1}) \\ \alpha_2(\boldsymbol{z}_{<2}) \\ \vdots \\ \alpha_D(\boldsymbol{z}_{<D}) \end{bmatrix}$$

- Generate $\boldsymbol{x}$ as:

$$\boldsymbol{x} = \boldsymbol{z} \odot \exp(\boldsymbol{\alpha}) + \boldsymbol{\mu}$$

Parallel Sampling

The sampling can be done in parallel which can be done fast.

Inverse Autoregressive Flow [5]

Likelihood Calculation

Probability calculation for a given $\boldsymbol{x}$ can be done sequentially as:

- $z_1 = \frac{x_1 - \mu_1}{\exp(\alpha_1)}$, Calculate $\begin{cases} \mu_2 \triangleq \mu_2(\boldsymbol{z}_{<2}) \\ \alpha_2 \triangleq \alpha_2(\boldsymbol{z}_{<2}) \end{cases}$
- $z_2 = \frac{x_2 - \mu_2}{\exp(\alpha_2)}$, Calculate $\begin{cases} \mu_3 \triangleq \mu_3(\boldsymbol{z}_{<3}) \\ \alpha_3 \triangleq \alpha_3(\boldsymbol{z}_{<3}) \end{cases}$

$\vdots$

- $z_D = \frac{x_D - \mu_D}{\exp(\alpha_D)}$

Calculate probabilty using $p_\theta(\boldsymbol{x}) = p(\boldsymbol{z}) \left| \prod_{d=1}^{D} \exp\left(\alpha_d(\boldsymbol{z}_{<d})\right) \right|^{-1}$

Sequential Likelihood Calculation

The likelihood calculation and thus learning which needs transformation Jacobian is sequential.

Section 9

Conclusion

Conclusion

Normalizing Flow Models

- Pros
  - Exact likelihood calculation
  - Invertible latent transformation
  - Latent representation extrapolation
- Cons
  - Same latent and visible space dimension
  - Limitation to invertible transformation
  - Scalability issues due to latent dimension

List of Abbreviations

Complete	Abbreviation
Inverse autoregressive Flow	IAF
Masked Autoregressive Flow	MAF
Nonlinear Independent Component Estimation	NICE
Real-valued Non-volume Preserving Transformations	Real-NVP
Variational AutoEncoder	VAE

References I


Christine Breiner,
"Change of variables | mit 18.02sc multivariable calculus, fall 2010," .


Laurent Dinh, David Krueger, and Yoshua Bengio,
"Nice: Non-linear independent components estimation,"
arXiv preprint arXiv:1410.8516, 2014.


Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio,
"Density estimation using real nvp,"
arXiv preprint arXiv:1605.08803, 2016.


George Papamakarios, Theo Pavlakou, and Iain Murray,
"Masked autoregressive flow for density estimation,"
Advances in neural information processing systems, vol. 30, 2017.


Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling,

"Improved variational inference with inverse autoregressive flow,"
Advances in neural information processing systems, vol. 29, 2016.