VI & BNN (2)

Flow-based Models

Keywords : Normalizing Flow, Masked Autoregressive Flow,
Inverse Autoregressive Flow, Glow, RealNVP, NICE

Seunghan Lee
2021.03.15.Mon

Contents

Paper List / References

Papers

- Non-linear Independent Components Estimation (NICE) (Dinh et al., 2014)
- MADE : Masked Autoencoder for Distribution Estimation (Germain et al., 2015)
- Variational Inference with Normalizing Flows (Rezende and Mohamed, 2016)
- Improved Variational Inference with Inverse Autoregressive Flow (Kingma et al., 2016)
- Density Estimation using Real NVP (Dinh et al., 2017)
- Masked Autoregressive Flow for Density Estimation (Papamakarios et al., 2017)
- Glow : Generative Flow with Invertible 1x1 Convolutions (Kingma and Dhariwal, 2018)

Blog

- **https://seunghan96.github.io/**
- https://lilianweng.github.io/lil-log/2018/10/13/flow-based-deep-generative-models.html#glow
- https://sites.google.com/view/berkeley-cs294-158-sp20/home

1. What is Normalizing Flow?

1. What is Normalizing Flow?

1-1. Generative Modeling

Generative Model vs Discriminative Model

Generative Model : captures the joint probability $P(X, Y)$

(or just $P(X)$ if Y does not exists)

Discriminative Model : captures the conditional probability$P(Y \mid X)$

1. What is Normalizing Flow?

1-1. Generative Modeling

Generative Model vs Discriminative Model

Generative Model : captures the joint probability $P(X, Y)$

(or just $P(X)$ if Y does not exists)

Discriminative Model : captures the conditional probability$P(Y \mid X)$

1. What is Normalizing Flow?

1-1. Generative Modeling

Generative Model vs Discriminative Model

Generative Model : captures the joint probability $P(X, Y)$

(or just $P(X)$ if Y does not exists)

Discriminative Model : captures the conditional probability $P(Y \mid X)$

What do we want to do from this Generative Model?

1) Sampling
2) Density Evaluation

1. What is Normalizing Flow?

1-1. Generative Modeling

Generative Model : models that can generate samples (data)!

ex) GAN, VAE

- both shows an impressive result on hard tasks, such as images!
- But cannot evaluate exact probability density of new points!

Using **Normalizing Flow**…

we can do both **"(1) sampling" & "(2) density evaluation"** efficiently & exacty!

So, what is Normalizing Flow?

1-1. Generative Modeling

Generative Model : models that can generate samples (data)!

ex) GAN, VAE

- both shows … har…
- But cannot … ty de…

(1) Generate new image!

(2) Find the density of a given image!

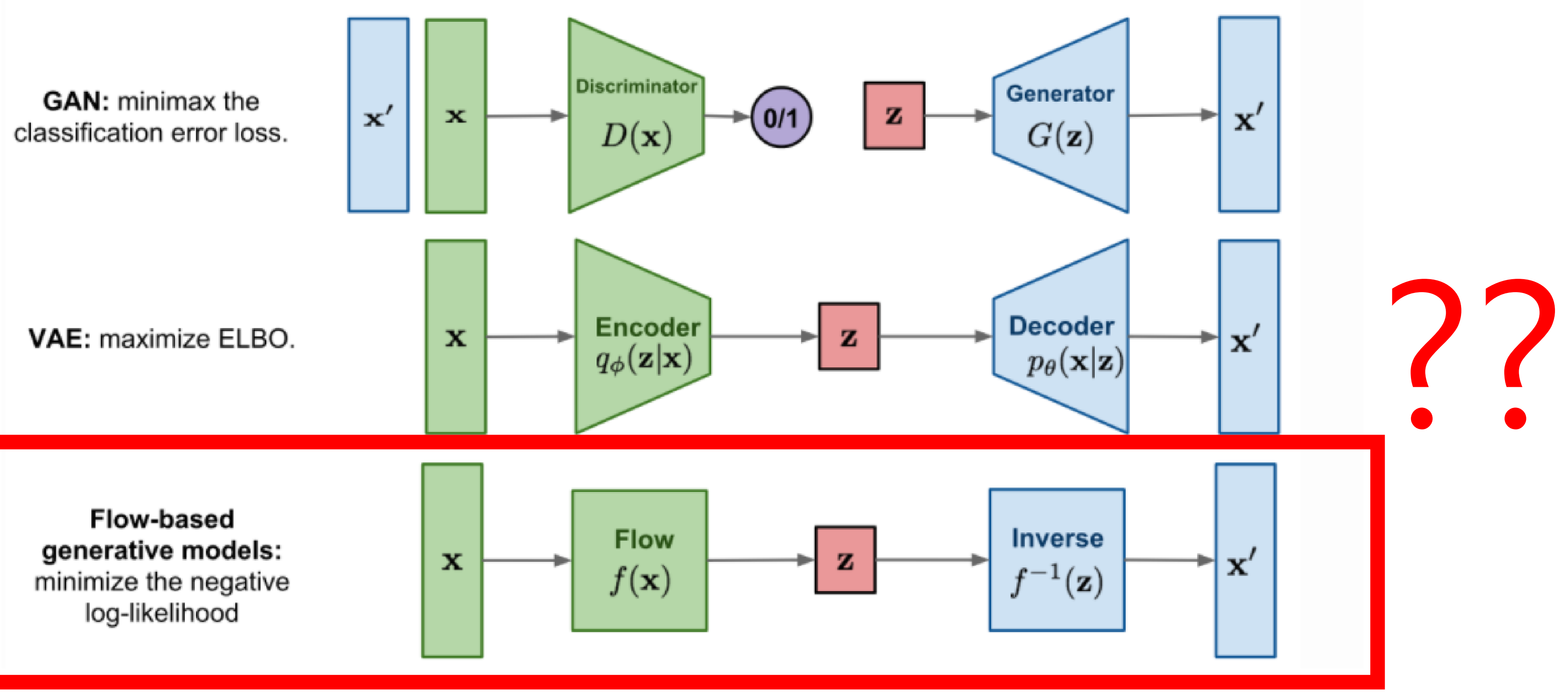
Using **Normalizing Flow**…

we can do both "**(1) sampling**" & "**(2) density evaluation**" efficiently & exacty!

So, what is Normalizing Flow?

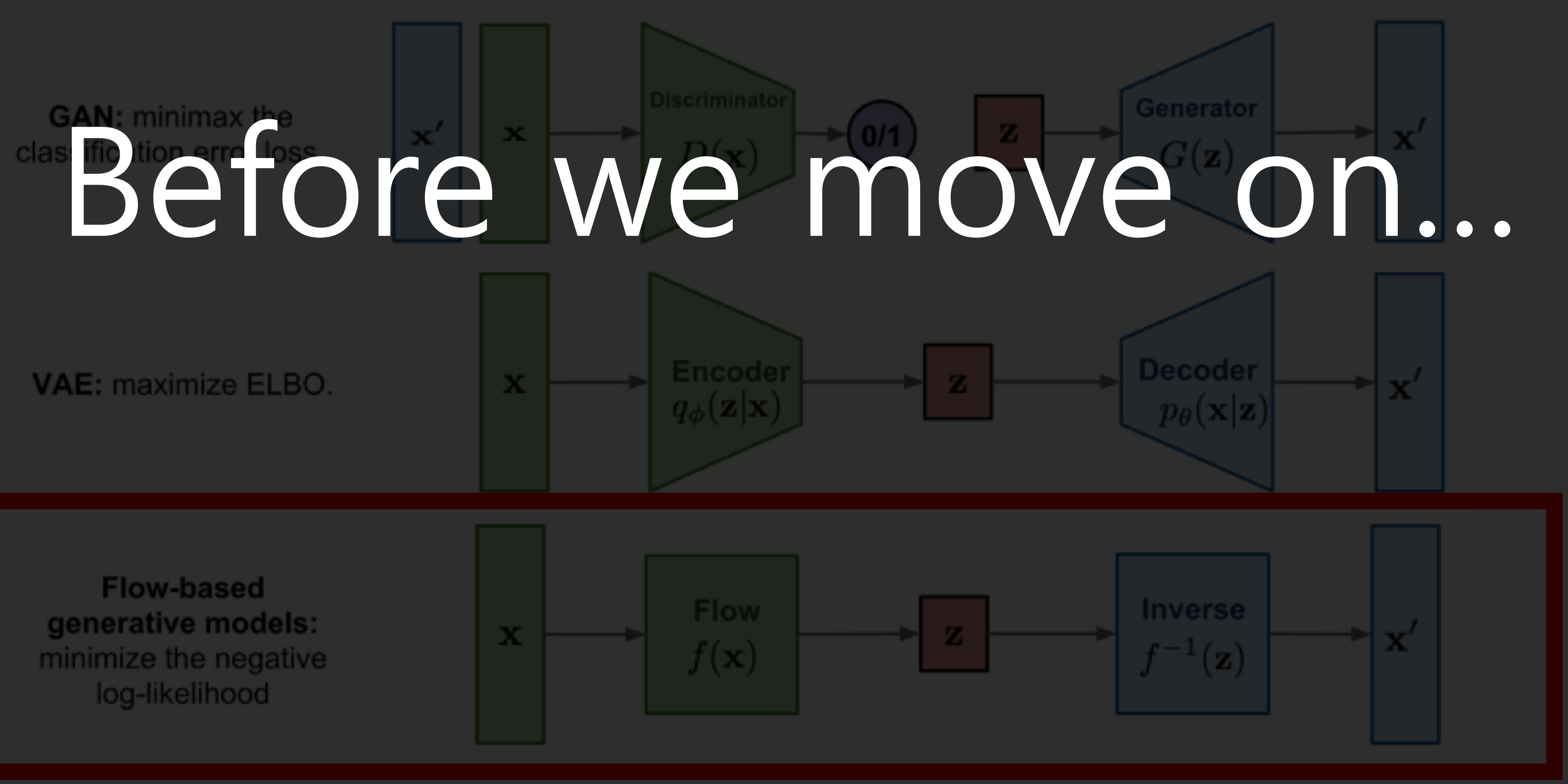
1. What is Normalizing Flow?

1-1. Generative Modeling



1. What is Normalizing Flow?

1-1. Generative Modeling



Before we move on…

1. What is Normalizing Flow?

1-2. Change of Variables

Univariate Case

$$\int p(x)dx = \int \pi(z)dz = 1 \text{ ; Definition of probability distribution.}$$

$$p(x) = \pi(z)\left|\frac{dz}{dx}\right| = \pi(f^{-1}(x))\left|\frac{df^{-1}}{dx}\right| = \pi(f^{-1}(x))|(f^{-1})'(x)|$$

Multivariate Case

$$\mathbf{z} \sim \pi(\mathbf{z}), \mathbf{x} = f(\mathbf{z}), \mathbf{z} = f^{-1}(\mathbf{x})$$

$$p(\mathbf{x}) = \pi(\mathbf{z})\left|\det\frac{d\mathbf{z}}{d\mathbf{x}}\right| = \pi(f^{-1}(\mathbf{x}))\left|\det\frac{df^{-1}}{d\mathbf{x}}\right|$$

1. What is Normalizing Flow?

1-2. Change of Variables

Univariate Case

$$\int p(x)dx = \int \pi(z)dz = 1 \text{ ; Definition of probability distribution.}$$

$$p(x) = \pi(z)\left|\frac{dz}{dx}\right| = \pi(f^{-1}(x))\left|\frac{df^{-1}}{dx}\right| = \pi(f^{-1}(x))|(f^{-1})'(x)|$$

Multivariate Case

$$\mathbf{z} \sim \pi(\mathbf{z}), \mathbf{x} = f(\mathbf{z}), \mathbf{z} = f^{-1}(\mathbf{x})$$

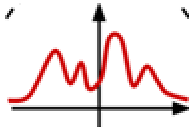
$$p(\mathbf{x}) = \pi(\mathbf{z})\left|\det \frac{d\mathbf{z}}{d\mathbf{x}}\right| = \pi(f^{-1}(\mathbf{x}))\left|\det \frac{df^{-1}}{d\mathbf{x}}\right|$$

Key point :

Need to calculate the **Jacobian term!**

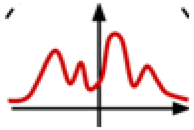
1. What is Normalizing Flow?

1-3. What is Normalizing Flow?

Goal : We want to find out the unknown, complex pdf $p(x)$ 

1. What is Normalizing Flow?

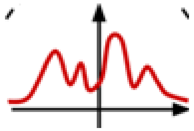
1-3. What is Normalizing Flow?

Goal : We want to find out the unknown, complex pdf $p(x)$ 

1) Want to **SAMPLE IMAGE** from this distribution!
2) Want to **EVALUATE the DENSITY** of the image!

1. What is Normalizing Flow?

1-3. What is Normalizing Flow?

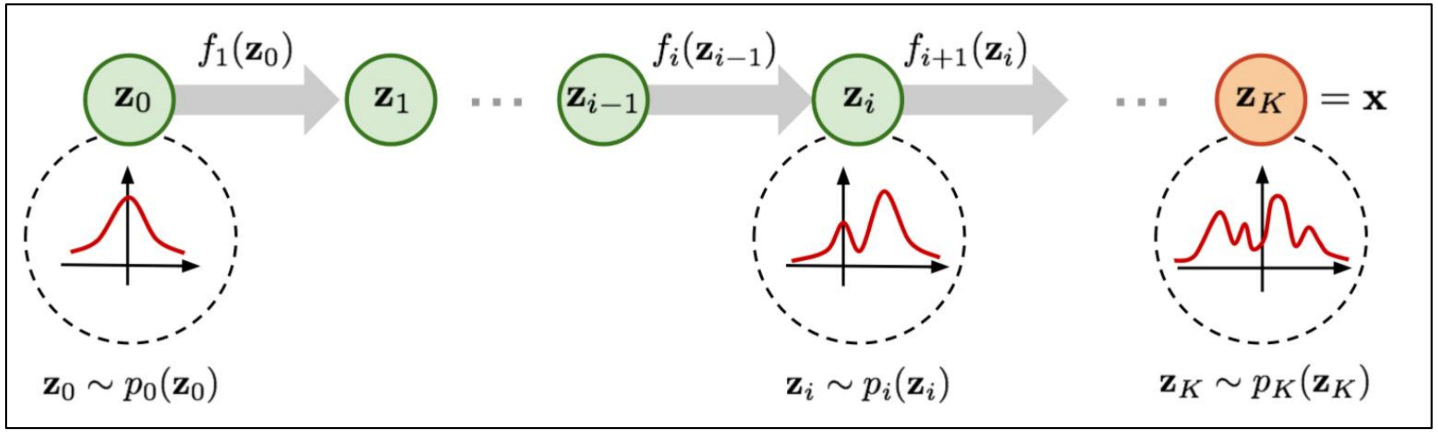
Goal : We want to find out the unknown, complex pdf $p(x)$ 

<u>Intuitive idea of NF</u> :

(1) Make a simple distribution!

(2) Change x N until it becomes the distribution that we want! (sequentially)

(from $p_0(\mathbf{z}_0)$ to $p_K(\mathbf{z}_K)$)



1. What is Normalizing Flow?

1-3. What is Normalizing Flow?

By the change of variables formula..

$$\mathbf{z}_{i-1} \sim p_{i-1}(\mathbf{z}_{i-1})$$

$$\mathbf{z}_i = f_i(\mathbf{z}_{i-1}), \text{ thus } \mathbf{z}_{i-1} = f_i^{-1}(\mathbf{z}_i)$$

$$p_i(\mathbf{z}_i) = p_{i-1}(f_i^{-1}(\mathbf{z}_i)) \left| \det \frac{d f_i^{-1}}{d \mathbf{z}_i} \right|$$

$$= p_{i-1}(\mathbf{z}_{i-1}) \left| \det \left(\frac{d f_i}{d \mathbf{z}_{i-1}} \right)^{-1} \right|$$

$$= p_{i-1}(\mathbf{z}_{i-1}) \left| \det \frac{d f_i}{d \mathbf{z}_{i-1}} \right|^{-1}$$

$$\frac{df^{-1}(y)}{dy} = \frac{dx}{dy} = \left(\frac{dy}{dx}\right)^{-1} = \left(\frac{df(x)}{dx}\right)^{-1}$$

$$\det(M^{-1}) = (\det(M))^{-1}$$

1. What is Normalizing Flow?

1-3. What is Normalizing Flow?

$$\log p_i(\mathbf{z}_i) = \log p_{i-1}(\mathbf{z}_{i-1}) - \log\left|\det \frac{df_i}{d\mathbf{z}_{i-1}}\right|$$

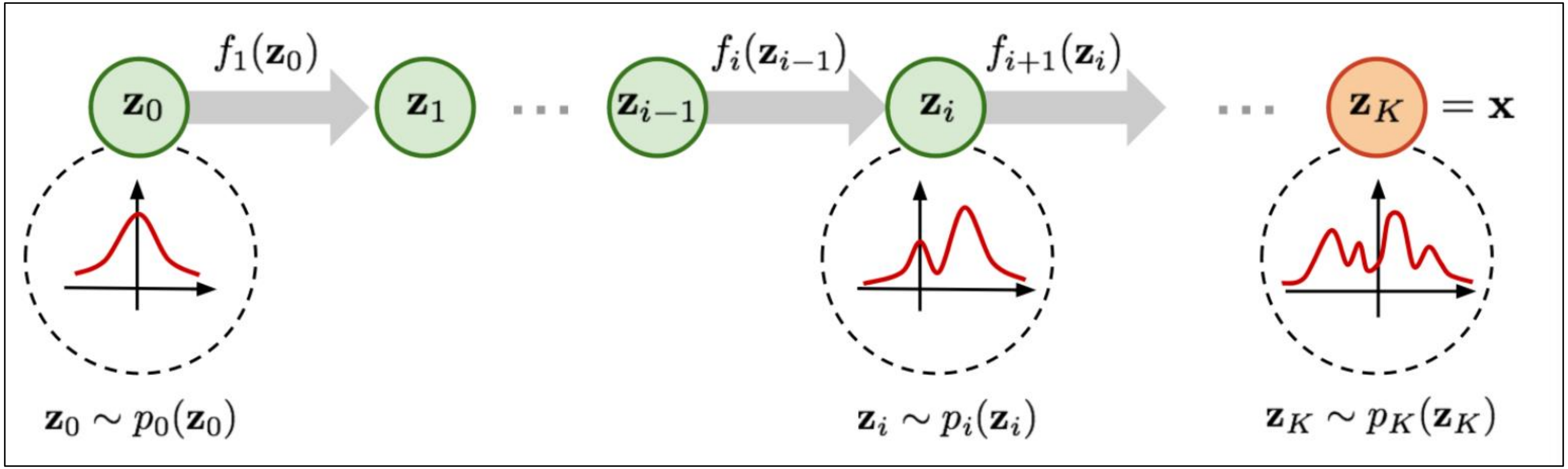
Sequentially apply the multiple functions to our base distribution!

$$\boxed{\mathbf{x} = \mathbf{z}_K = f_K \circ f_{K-1} \circ \cdots \circ f_1(\mathbf{z}_0)}$$

$$\begin{aligned}
\log p(\mathbf{x}) = \log \pi_K(\mathbf{z}_K) &= \log \pi_{K-1}(\mathbf{z}_{K-1}) - \log\left|\det \frac{df_K}{d\mathbf{z}_{K-1}}\right| \\
&= \log \pi_{K-2}(\mathbf{z}_{K-2}) - \log\left|\det \frac{df_{K-1}}{d\mathbf{z}_{K-2}}\right| - \log\left|\det \frac{df_K}{d\mathbf{z}_{K-1}}\right| \\
&= \dots \\
&= \underline{\log \pi_0(\mathbf{z}_0) - \sum_{i=1}^{K} \log\left|\det \frac{df_i}{d\mathbf{z}_{i-1}}\right|}
\end{aligned}$$

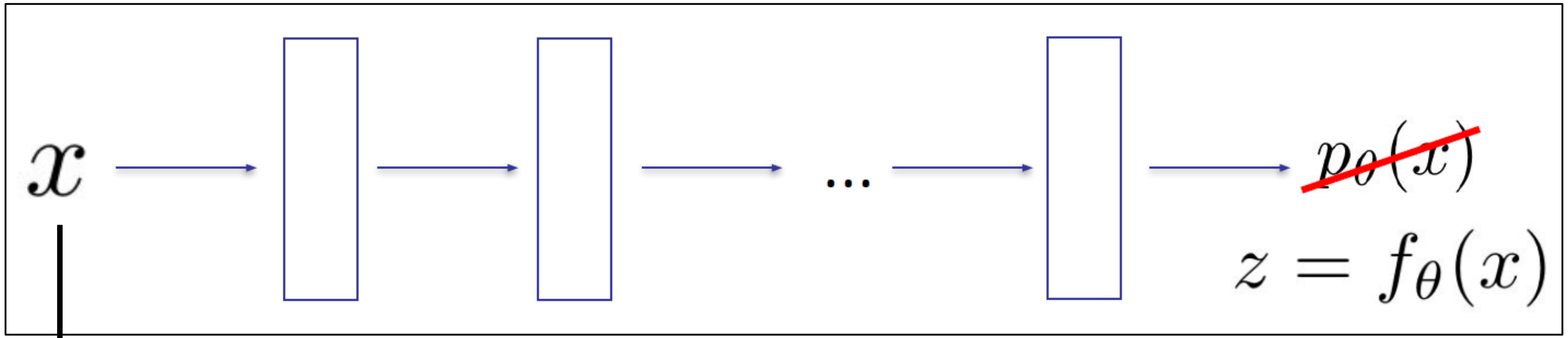
1. What is Normalizing Flow?

1-3. What is Normalizing Flow?



1. What is Normalizing Flow?

1-4. How to train NF?



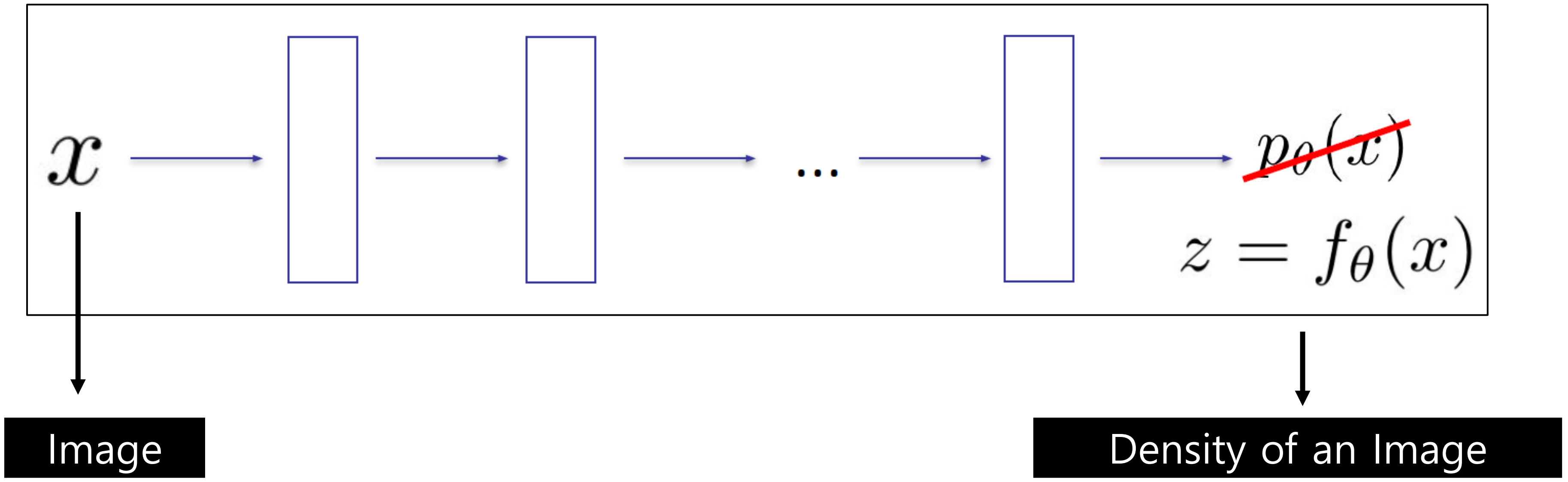
Image

Density of an Image

$$\boxed{z \sim \mathcal{N}(0, 1)}$$

1. What is Normalizing Flow?

1-4. How to train NF?

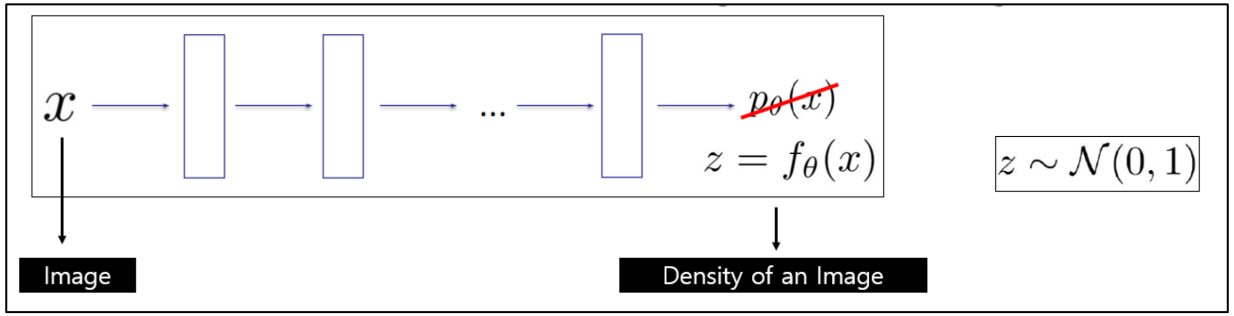


$\boxed{z \sim \mathcal{N}(0, 1)}$

How do we train this model?

1. What is Normalizing Flow?

1-4. How to train NF?



$$z^{(i)} = f_\theta(x^{(i)})$$

$$\begin{aligned} p_\theta(x^{(i)}) &= \quad p_Z(z^{(i)}) \left| \frac{\partial z}{\partial x}(x^{(i)}) \right| \\ &= p_Z(f_\theta(x^{(i)})) \left| \frac{\partial f_\theta}{\partial x}(x^{(i)}) \right| \end{aligned}$$

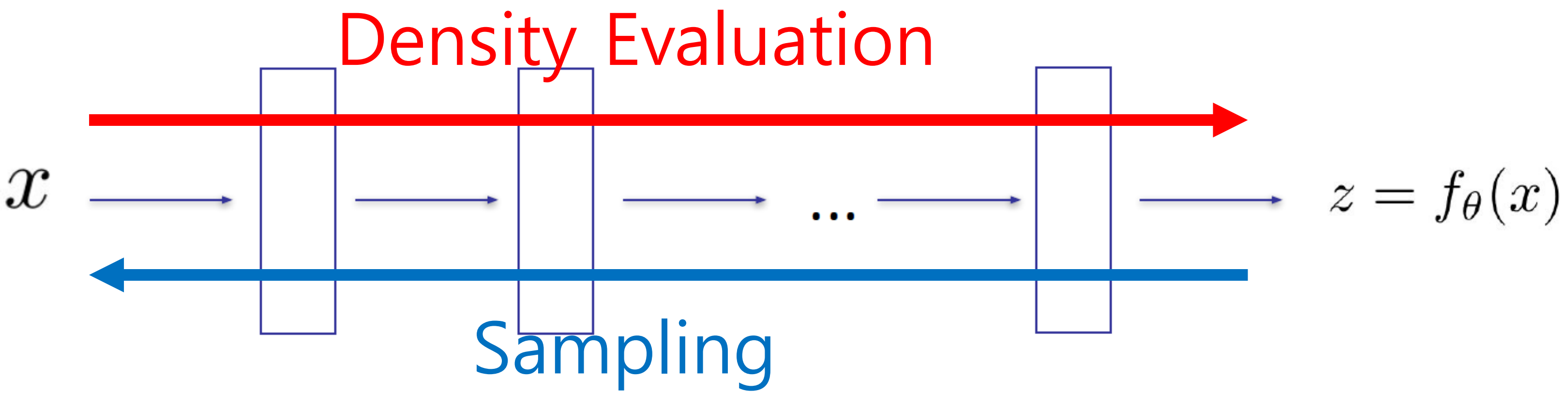
$$\max_\theta \sum_i \log p_\theta(x^{(i)})$$

$$\max_\theta \sum_i \log p_Z(f_\theta(x^{(i)})) + \log \left| \frac{\partial f_\theta}{\partial x}(x^{(i)}) \right|$$

Train this using SGD!

1. What is Normalizing Flow?

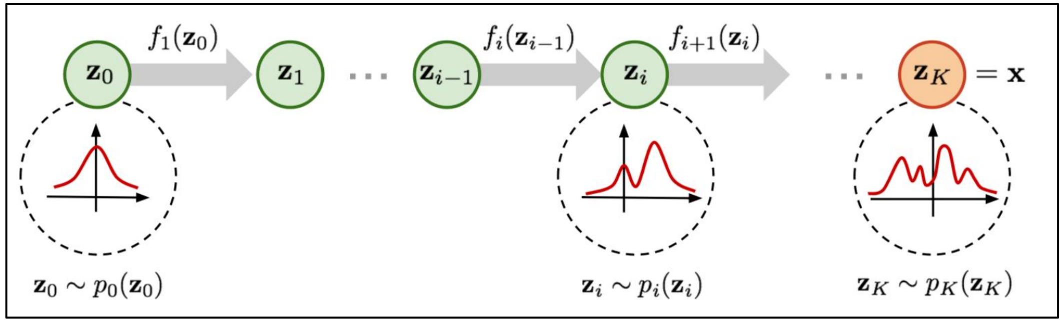
1-4. How to train NF?



1. What is Normalizing Flow?

1-4. How to train NF?

So, what should we choose for our function f_i ?



1. Easily Invertible

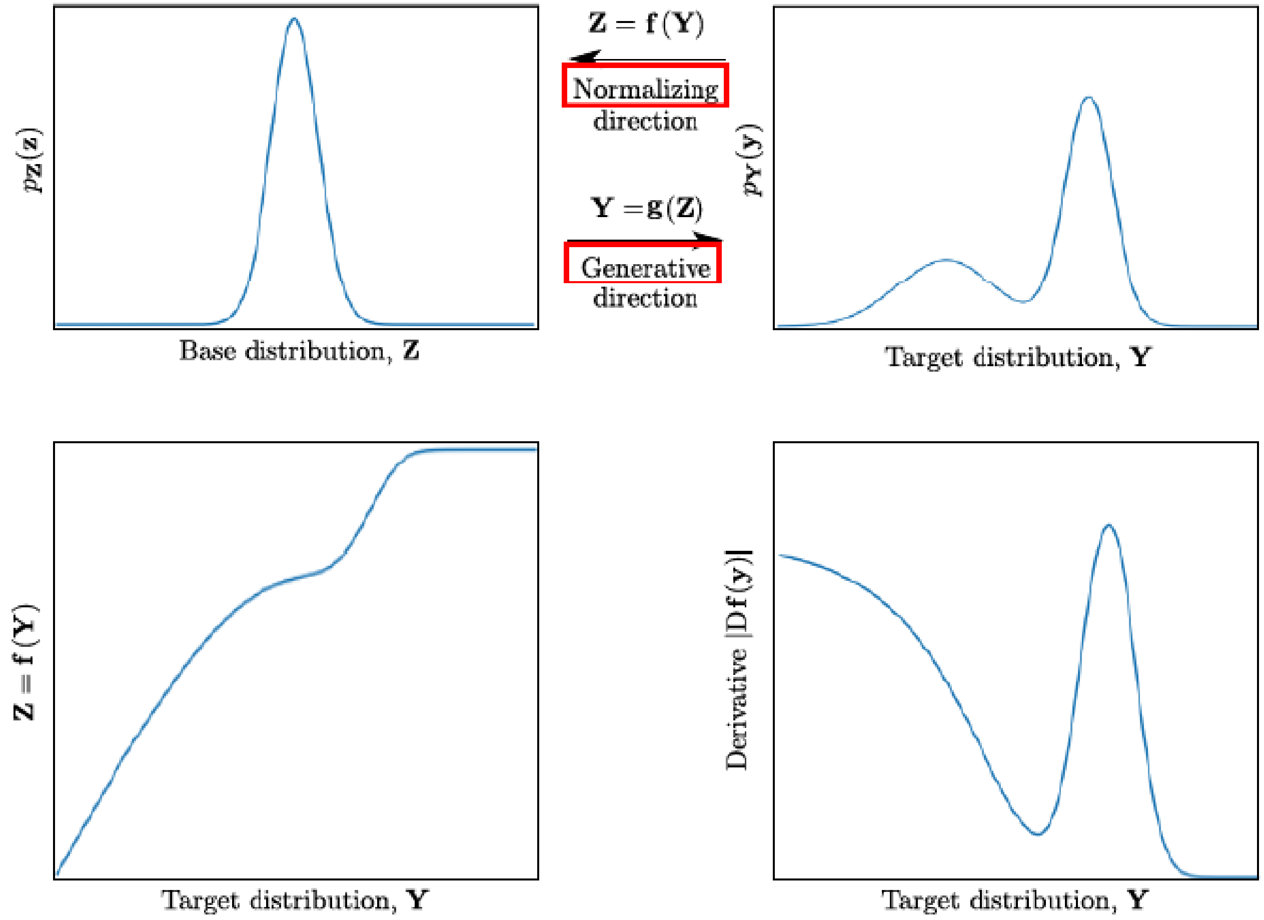
2. Determinant of Jacobian to be easily computed

(Computation time for finding the Jacobian for matrix of D-dim : O(D^3))

Lots of algorithms have been proposed that meets those two goals!

2. Models using Normalizing Flow

2. Models using Normalizing Flow



2. Models using Normalizing Flow

2-1. Real NVP

Real-valued Non-Volume Preserving (Dinh et al., 2017)

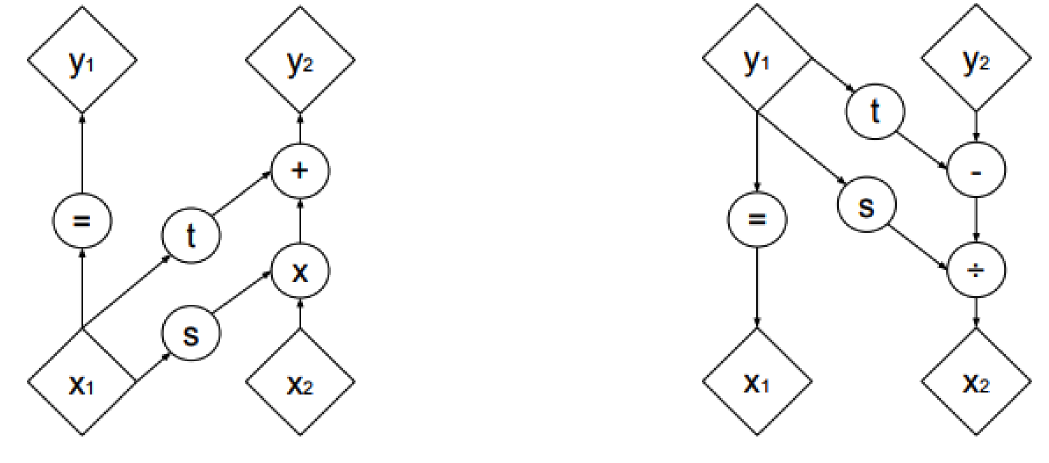
Key point : stack multiple invertible bijective transformation, “Affine Coupling Layer”

Affine Coupling Layer

$$\mathbf{y}_{1:d} = \mathbf{x}_{1:d}$$
$$\mathbf{y}_{d+1:D} = \mathbf{x}_{d+1:D} \odot \exp(s(\mathbf{x}_{1:d})) + t(\mathbf{x}_{1:d})$$

Split the dimensions into 2 parts

- one part : Transformation (O)
- second part : Transformation (X)



(a) Forward propagation (b) Inverse propagation

2. Models using Normalizing Flow

2-1. Real NVP

Real-valued Non-Volume Preserving (Dinh et al., 2017)

1. Easily Invertible

$$\begin{cases} \mathbf{y}_{1:d} & = \mathbf{x}_{1:d} \\ \mathbf{y}_{d+1:D} & = \mathbf{x}_{d+1:D} \odot \exp(s(\mathbf{x}_{1:d})) + t(\mathbf{x}_{1:d}) \end{cases} \Leftrightarrow \begin{cases} \mathbf{x}_{1:d} & = \mathbf{y}_{1:d} \\ \mathbf{x}_{d+1:D} & = (\mathbf{y}_{d+1:D} - t(\mathbf{y}_{1:d})) \odot \exp(-s(\mathbf{y}_{1:d})) \end{cases}$$

2. Determinant of Jacobian to be easily computed

$$\frac{\partial y}{\partial x^T} = \begin{bmatrix} \mathbb{I}_d & 0 \\ \frac{\partial y_{d+1:D}}{\partial x^T_{1:d}} & \operatorname{diag}(\exp[s(x_{1:d})]) \end{bmatrix} \longrightarrow \exp\left[\sum_j s(x_{1:d})_j\right]$$

2. Models using Normalizing Flow

2-2. NICE

Non-linear Independent Component Estimation (Dinh et al., 2015)

Just make the Affine Coupling layer (of RealNVP) -> Additive Coupling layer

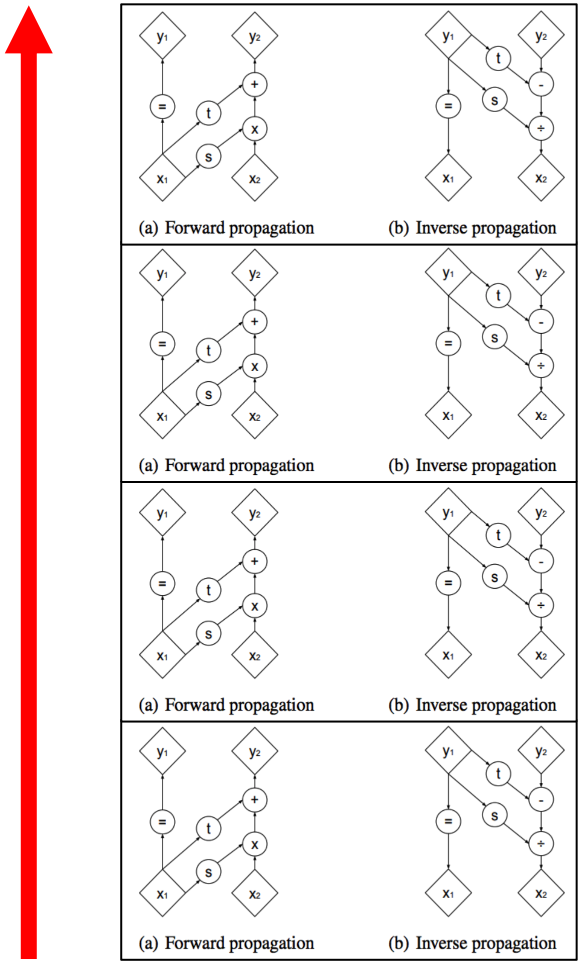
$$\begin{cases} \mathbf{y}_{1:d} & = \mathbf{x}_{1:d} \\ \mathbf{y}_{d+1:D} & = \mathbf{x}_{d+1:D} \odot \exp(s(\mathbf{x}_{1:d})) + t(\mathbf{x}_{1:d}) \end{cases} \Leftrightarrow \begin{cases} \mathbf{x}_{1:d} & = \mathbf{y}_{1:d} \\ \mathbf{x}_{d+1:D} & = (\mathbf{y}_{d+1:D} - t(\mathbf{y}_{1:d})) \odot \exp(-s(\mathbf{y}_{1:d})) \end{cases}$$

Affine Coupling Layer of RealNVP

$$\begin{cases} \mathbf{y}_{1:d} & = \mathbf{x}_{1:d} \\ \mathbf{y}_{d+1:D} & = \mathbf{x}_{d+1:D} + m(\mathbf{x}_{1:d}) \end{cases} \Leftrightarrow \begin{cases} \mathbf{x}_{1:d} & = \mathbf{y}_{1:d} \\ \mathbf{x}_{d+1:D} & = \mathbf{y}_{d+1:D} - m(\mathbf{y}_{1:d}) \end{cases}$$

Additive Coupling Layer of NICE

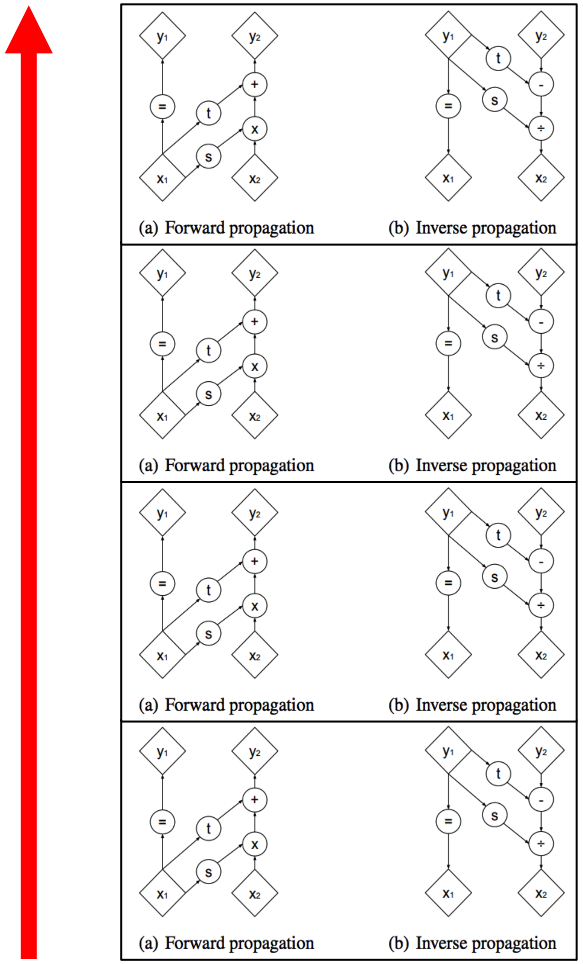
2. Models using Normalizing Flow



Think in advance. What would be the problem,

if we just stack the layers of RealNVP, NICE?

2. Models using Normalizing Flow



Think in advance. What would be the problem,

if we just stack the layers of RealNVP, NICE?

Need permutation between the dimensions!

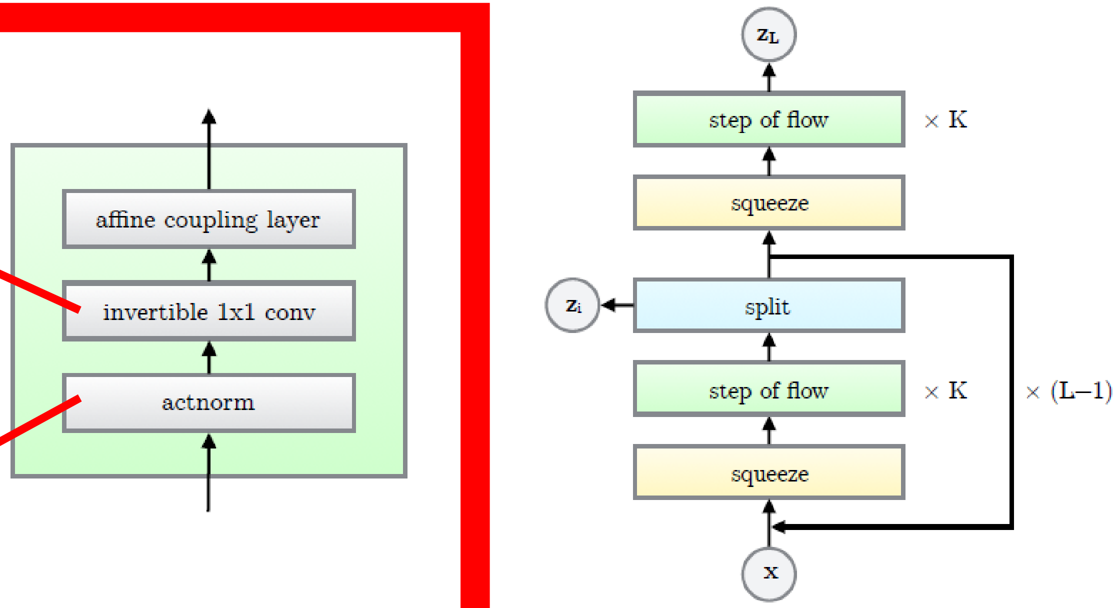
2. Models using Normalizing Flow

2-3. Glow

Generative Flow (Kingma and Dhariwal, 2018)

Generalization of Permutation!

Activation Normalization is a type of normalization used for flow-based generative models; specifically it was introduced in the GLOW architecture. An ActNorm layer performs an affine transformation of the activations using a scale and bias parameter per channel, similar to batch normalization. These parameters are initialized such that the post-actnorm activations per-channel have zero mean and unit variance given an initial minibatch of data. This is a form of data dependent initilization. After initialization, the scale and bias are treated as regular trainable parameters that are independent of the data.



(a) One step of our flow.

(b) Multi-scale architecture (Dinh et al., 2016).

2-3. Glow

Generative Flow (Kingma and Dhariwal, 2018)

1x1 Convolution

$h \times w \times c$ tensor $\mathbf{h}$ with $c \times c$ weight matrix $\mathbf{W}$

$$\boxed{\log \left| \det \left(\frac{d\,\mathrm{conv2D}(\mathbf{h}; \mathbf{W})}{d\,\mathbf{h}} \right) \right| = h \cdot w \cdot \log |\det(\mathbf{W})|}$$

➡ cost of computing or differentiating $\det(\mathbf{W})$ is $\mathcal{O}(c^3)$

How to solve this impracticability??

2-3. Glow

Generative Flow (Kingma and Dhariwal, 2018)

1x1 Convolution

Use **LU Decomposition :** $\mathcal{O}\left(c^3\right) \to \mathcal{O}(c)$

$$\mathbf{W} = \mathbf{PL}(\mathbf{U} + \mathrm{diag}(\mathbf{s}))$$

- P : permutation matrix
- L is a lower triangular matrix with ones on the diagonal
- U is an upper triangular matrix with zeros on the diagonal

$$\boxed{\log|\det(\mathbf{W})| = \mathrm{sum}(\log|\mathbf{s}|)}$$

2. Models using Normalizing Flow

2-3. Glow

Generative Flow (Kingma and Dhariwal, 2018)

Description	Function	Reverse Function	Log-determinant
Actnorm. See Section 3.1.	$\forall i,j : \mathbf{y}_{i,j} = \mathbf{s} \odot \mathbf{x}_{i,j} + \mathbf{b}$	$\forall i,j : \mathbf{x}_{i,j} = (\mathbf{y}_{i,j} - \mathbf{b})/\mathbf{s}$	$h \cdot w \cdot \mathtt{sum}(\log \lvert\mathbf{s}\rvert)$
Invertible 1×1 convolution. $\mathbf{W} : [c \times c]$. See Section 3.2.	$\forall i,j : \mathbf{y}_{i,j} = \mathbf{W}\mathbf{x}_{i,j}$	$\forall i,j : \mathbf{x}_{i,j} = \mathbf{W}^{-1}\mathbf{y}_{i,j}$	$h \cdot w \cdot \log \lvert\det(\mathbf{W})\rvert$ or $h \cdot w \cdot \mathtt{sum}(\log \lvert\mathbf{s}\rvert)$ (see eq. (10))
Affine coupling layer. See Section 3.3 and (Dinh et al., 2014)	$\mathbf{x}_a, \mathbf{x}_b = \mathtt{split}(\mathbf{x})$ $(\log \mathbf{s}, \mathbf{t}) = \mathtt{NN}(\mathbf{x}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{y}_a = \mathbf{s} \odot \mathbf{x}_a + \mathbf{t}$ $\mathbf{y}_b = \mathbf{x}_b$ $\mathbf{y} = \mathtt{concat}(\mathbf{y}_a, \mathbf{y}_b)$	$\mathbf{y}_a, \mathbf{y}_b = \mathtt{split}(\mathbf{y})$ $(\log \mathbf{s}, \mathbf{t}) = \mathtt{NN}(\mathbf{y}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{x}_a = (\mathbf{y}_a - \mathbf{t})/\mathbf{s}$ $\mathbf{x}_b = \mathbf{y}_b$ $\mathbf{x} = \mathtt{concat}(\mathbf{x}_a, \mathbf{x}_b)$	$\mathtt{sum}(\log(\lvert\mathbf{s}\rvert))$

Fig. 4. Three substeps in one step of flow in Glow. (Image source: Kingma and Dhariwal, 2018)

3. Models using Autoregressive Flow

3. Models using Autoregressive Flow

Autoregressive = only depends on the "previous" parts

$$p(\mathbf{x}) = \prod_{i=1}^{D} p\left(x_i \mid x_1, \ldots, x_{i-1}\right) = \prod_{i=1}^{D} p\left(x_i \mid x_{1:i-1}\right)$$

3. Models using Autoregressive Flow

Autoregressive = only depends on the "previous" parts

$$p(\mathbf{x}) = \prod_{i=1}^{D} p\left(x_i \mid x_1, \ldots, x_{i-1}\right) = \prod_{i=1}^{D} p\left(x_i \mid x_{1:i-1}\right)$$

This structure enables "triangular Jacobian"

3 popular model using autoregressive structure

1) MADE (Masked Autoencoder Density Estimation)

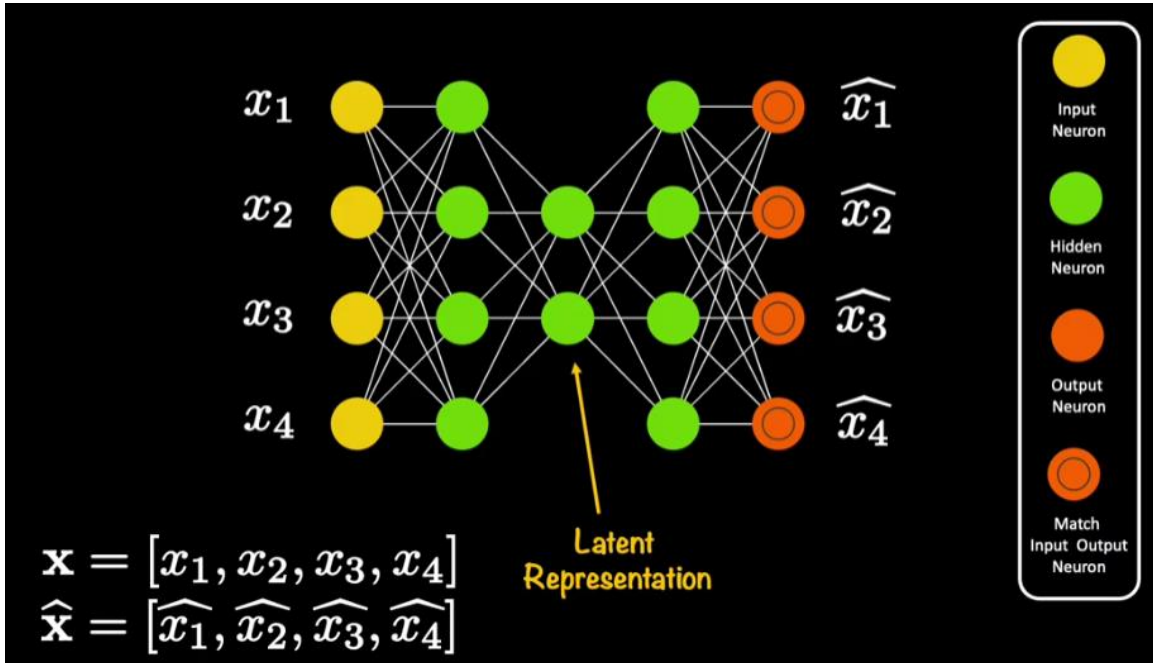
(Autoregressive Flow)

2) MAF (Masked Autoregressive Flow)

3) IAF (Inverse Autoregressive Flow)

3-1. MADE

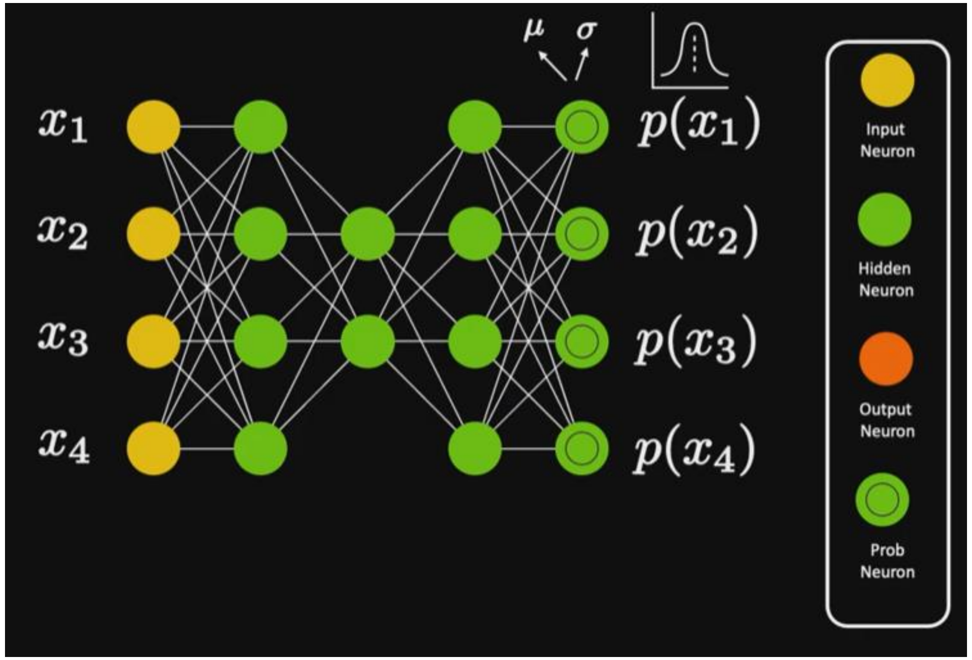
Masked Autoencoder Density Estimation (Germain et al., 2015)



https://www.youtube.com/watch?v=7q4ueFiJjAY

3-1. MADE

Masked Autoencoder Density Estimation (Germain et al., 2015)



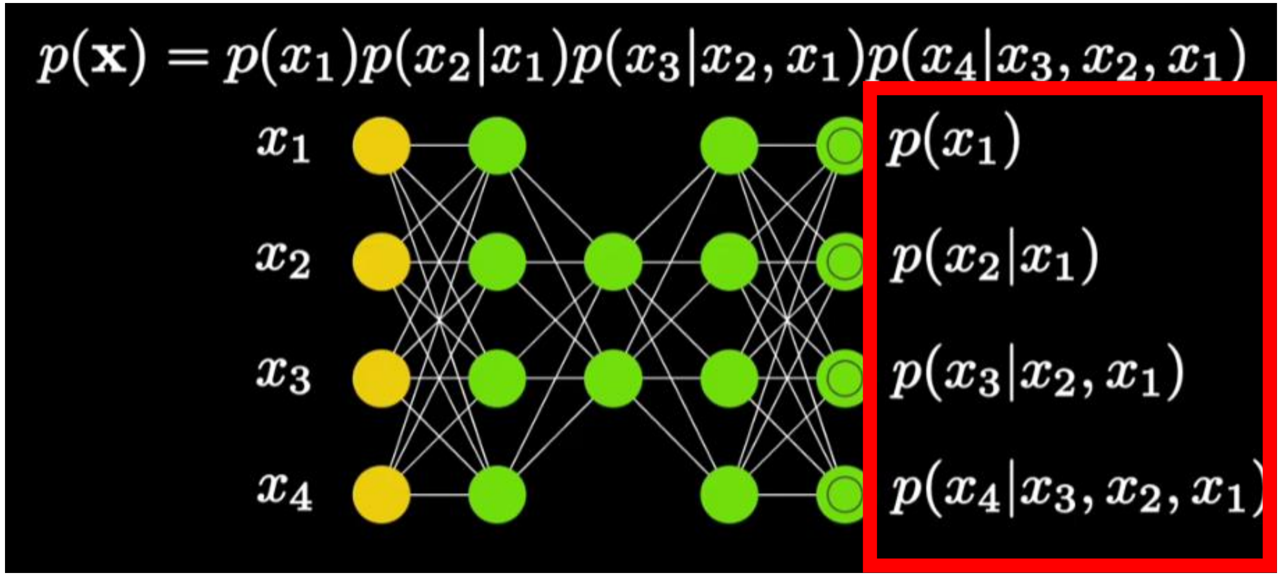
$$p(\mathbf{x}) = p(x_1)p(x_2)p(x_3)p(x_4)$$

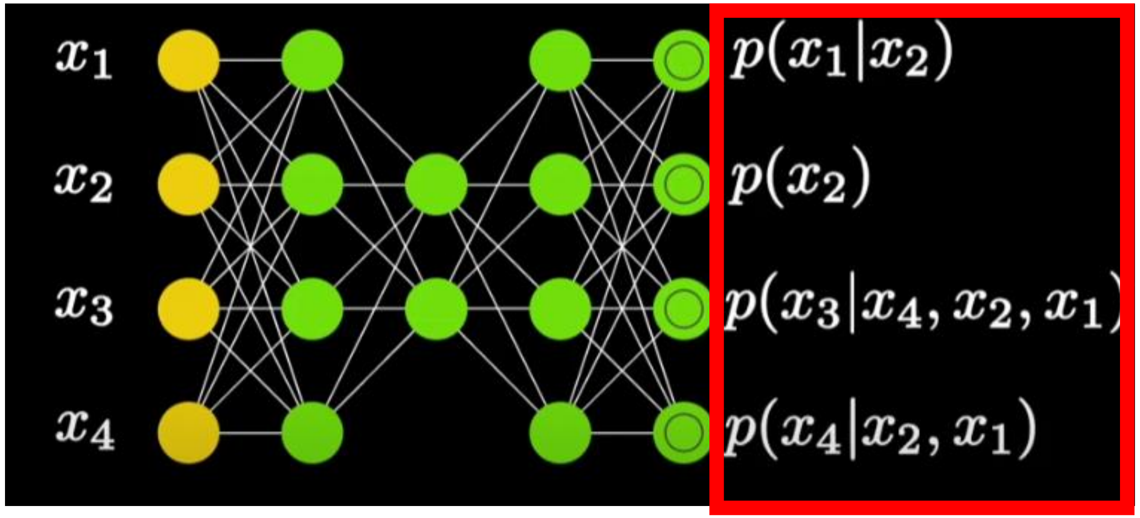
No conditional dependence between the input variables!

3. Models using Autoregressive Flow

3-1. MADE

Masked Autoencoder Density Estimation (Germain et al., 2015)





conditional dependence (O) **(Output Ordering is arbitrary)**

3. Models using Autoregressive Flow

3-1. MADE

Masked Autoencoder Density Estimation (Germain et al., 2015)

Goal : Constrain the AE to learn joint pdf with conditional dependence!

(+agnostic dimension ordering)

3-1. MADE

Masked Autoencoder Density Estimation (Germain et al., 2015)

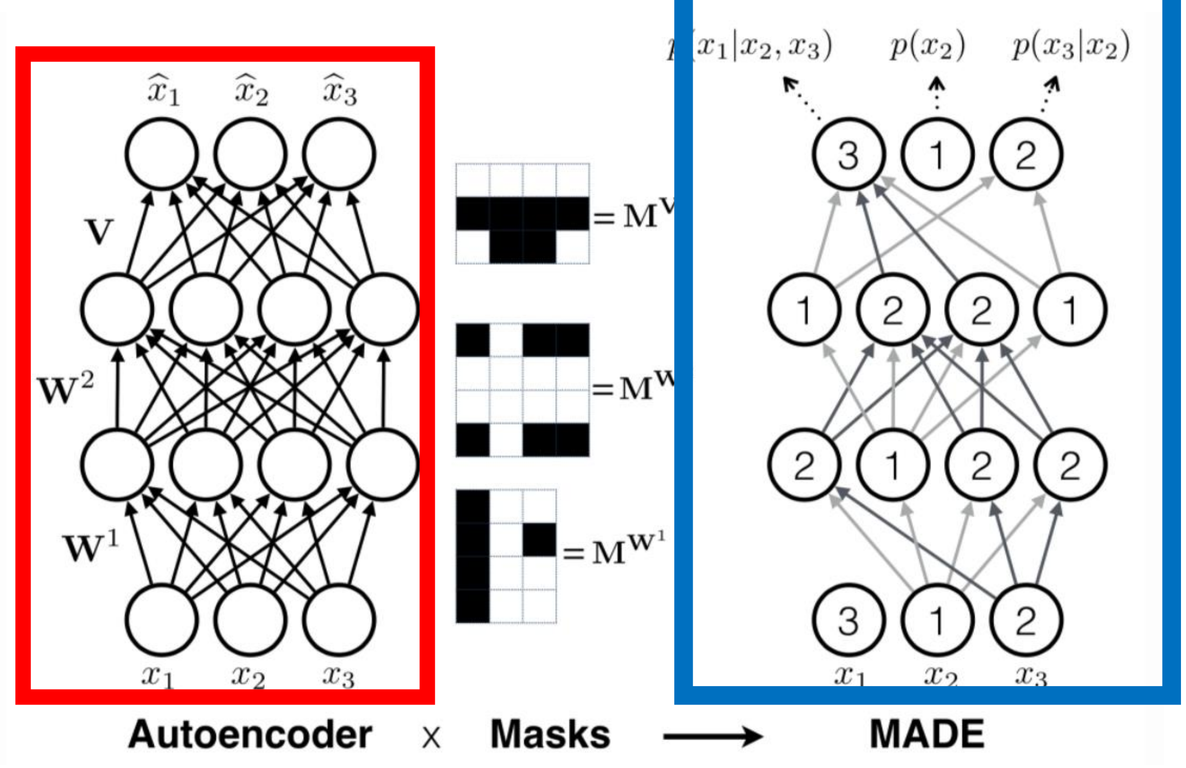
Way to disconnect the weights : use binary Masks!

$$
\begin{aligned}
\mathbf{h}^0 &= \mathbf{x} \\
\mathbf{h}^l &= \text{activation}^l(\mathbf{W}^l\mathbf{h}^{l-1} + \mathbf{b}^l) \\
\hat{\mathbf{x}} &= \sigma(\mathbf{V}\mathbf{h}^L + \mathbf{c})
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{h}^l &= \text{activation}^l((\mathbf{W}^l \odot \mathbf{M}^{\mathbf{W}^l})\mathbf{h}^{l-1} + \mathbf{b}^l) \\
\hat{\mathbf{x}} &= \sigma((\mathbf{V} \odot \mathbf{M}^{\mathbf{V}})\mathbf{h}^L + \mathbf{c})
\end{aligned}
$$

$$
M^{\mathbf{W}^l}_{k',k} = \mathbf{1}_{m^l_{k'} \geq m^{l-1}_k} = \begin{cases} 1, & \text{if } m^l_{k'} \geq m^{l-1}_k \\ 0, & \text{otherwise} \end{cases}
$$

$$
M^{\mathbf{V}}_{d,k} = \mathbf{1}_{d \geq m^L_k} = \begin{cases} 1, & \text{if } d > m^L_k \\ 0, & \text{otherwise} \end{cases}
$$



3-1. MADE

Masked Autoencoder Density Estimation (Germain et al., 2015)

Then, how can we impose “Autoregressive” property to mask?

- 1) assign each unit (in hidden layer) an integer m (between 1 and $D-1$)

  ($m(k)$ = maximum number of input units to which it can be connected)

  ($m(k) \neq 1$, $m(k) \neq D$)

Notation & Meaning

- M^{V} and M^{W} : network's connectivity
- matrix product $\mathbf{M}^{\mathrm{V,W}} = \mathrm{M}^{\mathrm{V}}\mathrm{M}^{\mathrm{W}}$: connectivity between the input and the output layer
- $M^{\mathrm{V,W}}_{d',d}$: number of network paths between output unit $\hat{x}_{d'}$ and input unit x_d.

- 2) [MASK] matrix masking the connections between "input & hidden units"

  constraints on the maximum number of inputs to each hidden unit are encoded in it!

$$\boxed{M^{\mathrm{W}}_{k,d} = 1_{m(k)\geq d} = \begin{cases} 1 & \text{if } m(k) \geq d \\ 0 & \text{otherwise} \end{cases}}$$

- 3) [MASK] matrix masking the connections between "hidden & output" units

$$\boxed{M^{\mathrm{V}}_{d,k} = 1_{d>m(k)} = \begin{cases} 1 & \text{if } d > m(k) \\ 0 & \text{otherwise} \end{cases}}$$

(single hidden layer NN)

3. Models using Autoregressive Flow

3-1. MADE

Masked Autoencoder Density Estimation (Germain et al., 2015)

Then, how can we impose "Autoregressive" property to mask?

Mask

- $\boxed{M^{\mathbf{W}^l}_{k',k} = 1_{m^l(k') \geq m^{l-1}(k)} = \begin{cases} 1 & \text{if } m^l(k') \geq m^{l-1}(k) \\ 0 & \text{otherwise} \end{cases}}$
- $\boxed{M^{\mathbf{V}}_{d,k} = 1_{d > m^L(k)} = \begin{cases} 1 & \text{if } d > m^L(k) \\ 0 & \text{otherwise} \end{cases}}$

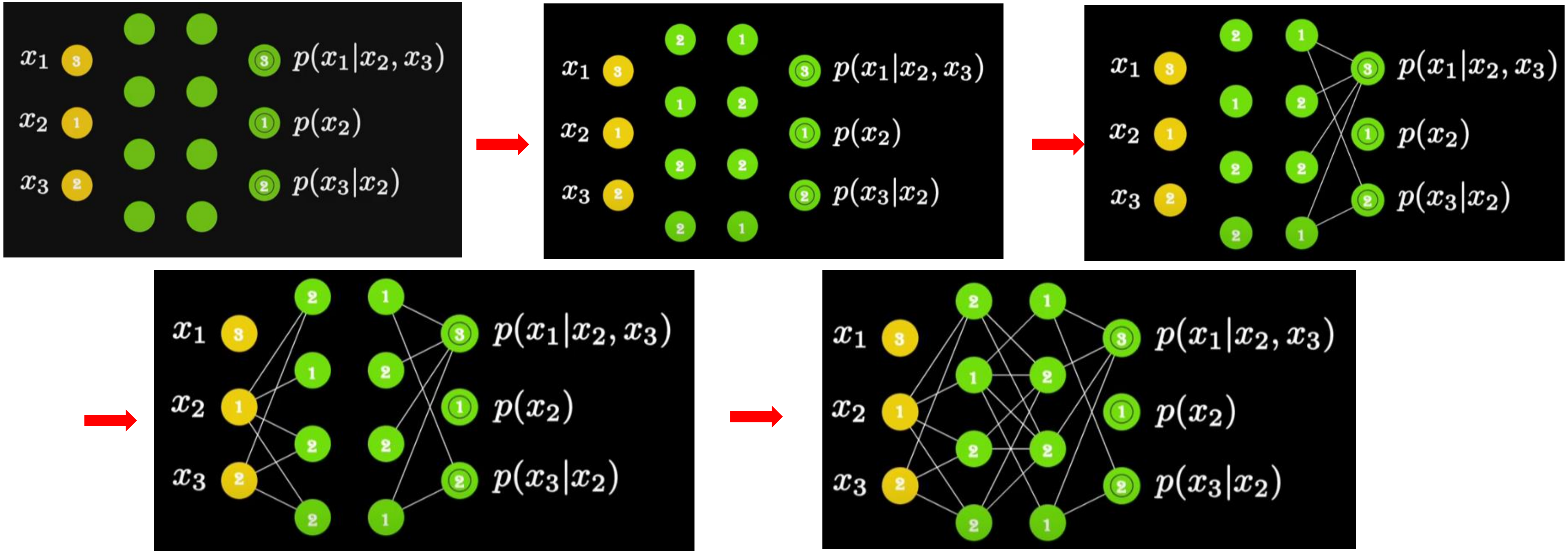
Notation

- W^1 : first hidden layer matrix
- W^2 : second hidden layer matrix
- K^l : number of hidden units in layer l
- $m^l(k)$: maximum number of connected inputs of the k^{th} unit in the l^{th} layer

(Deep NN)

3-1. MADE

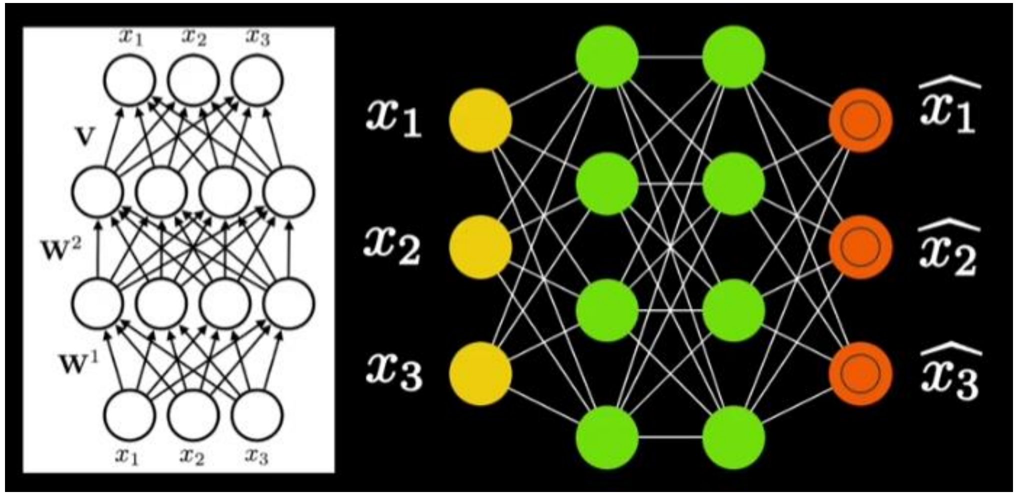
Masked Autoencoder Density Estimation (Germain et al., 2015)

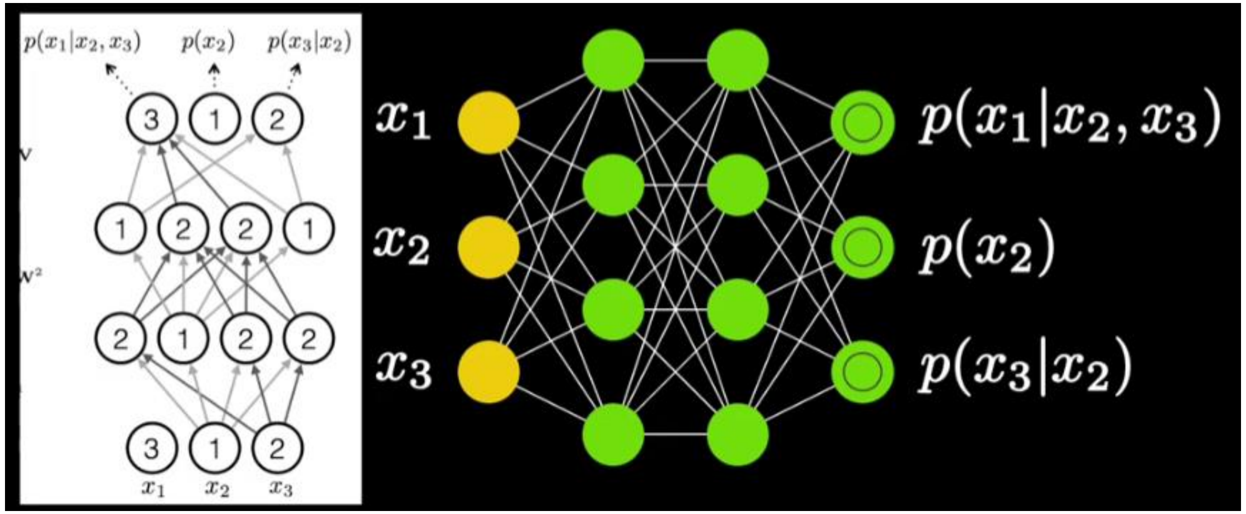


3. Models using Autoregressive Flow

3-1. MADE

Masked Autoencoder Density Estimation (Germain et al., 2015)

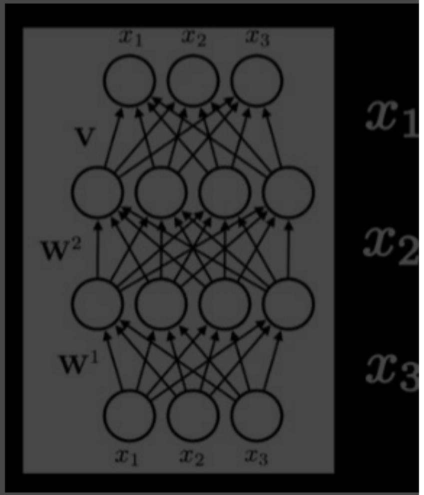


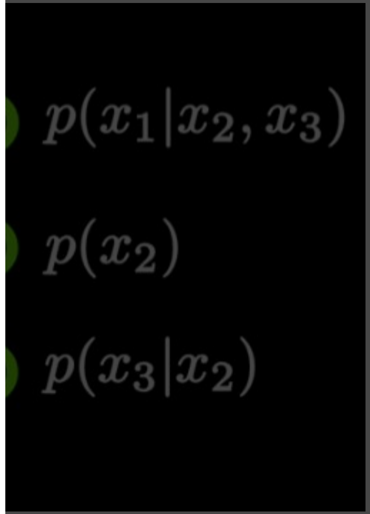


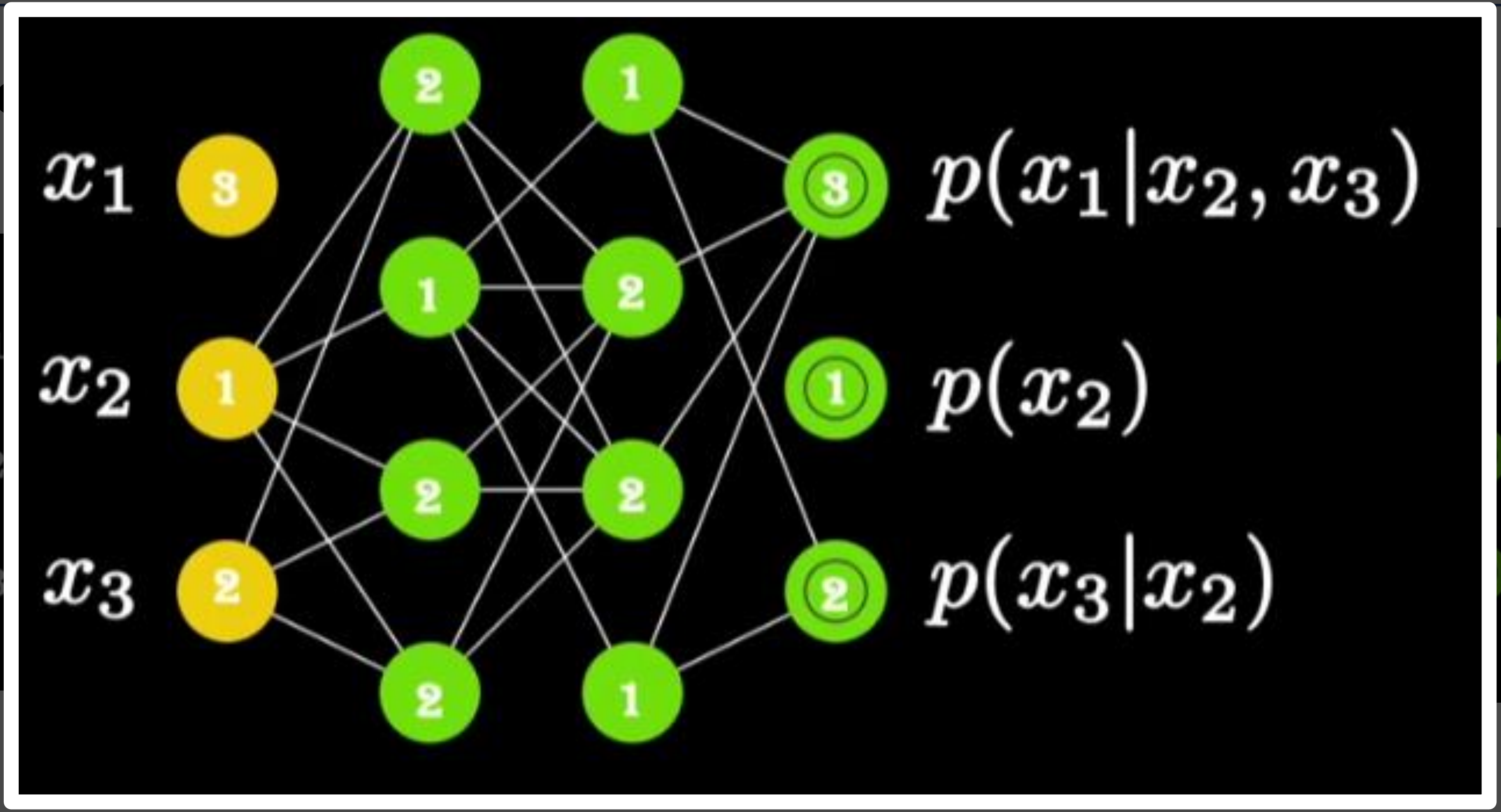
3. Models using Autoregressive Flow

3-1. MADE

Masked Autoc







3. Models using Autoregressive Flow

Autoregressive Flow

$$x_1 \sim p_\theta(x_1)$$
$$x_2 \sim p_\theta(x_2 \mid x_1)$$
$$x_3 \sim p_\theta(x_3 \mid x_1, x_2)$$

3. Models using Autoregressive Flow

Autoregressive Flow

$$x_1 \sim p_\theta(x_1)$$
$$x_2 \sim p_\theta(x_2 \mid x_1)$$
$$x_3 \sim p_\theta(x_3 \mid x_1, x_2)$$

$$x_1 = f_\theta^{\boxed{-1}}(z_1)$$
$$x_2 = f_\theta^{\boxed{-1}}(z_2; x_1)$$
$$x_3 = f_\theta^{\boxed{-1}}(z_3; x_1, x_2)$$

$$z_i \sim N(0,1)$$

1) Sampling

3. Models using Autoregressive Flow

Autoregressive Flow

$$x_1 \sim p_\theta(x_1)$$
$$x_2 \sim p_\theta(x_2 \mid x_1)$$
$$x_3 \sim p_\theta(x_3 \mid x_1, x_2)$$

$$x_1 = f_\theta^{\boxed{-1}}(z_1)$$
$$x_2 = f_\theta^{\boxed{-1}}(z_2; x_1)$$
$$x_3 = f_\theta^{\boxed{-1}}(z_3; x_1, x_2)$$

$$z_i \sim N(0,1)$$

1) Sampling **Can be done in “D" pass (Slow)**

3. Models using Autoregressive Flow

Autoregressive Flow

$$x_1 \sim p_\theta(x_1)$$
$$x_2 \sim p_\theta(x_2 \mid x_1)$$
$$x_3 \sim p_\theta(x_3 \mid x_1, x_2)$$

$$x_1 = f_\theta^{\boxed{-1}}(z_1)$$
$$x_2 = f_\theta^{\boxed{-1}}(z_2; x_1)$$
$$x_3 = f_\theta^{\boxed{-1}}(z_3; x_1, x_2)$$

$z_i \sim N(0, 1)$

1) Sampling

Can be done in “D" pass (Slow)

$$z_1 = f_\theta(x_1)$$
$$z_2 = f_\theta(x_2; x_1)$$
$$z_3 = f_\theta(x_3; x_1, x_2)$$

2) Density Evaluation

3. Models using Autoregressive Flow

Autoregressive Flow

$$x_1 \sim p_\theta(x_1)$$
$$x_2 \sim p_\theta(x_2 \mid x_1)$$
$$x_3 \sim p_\theta(x_3 \mid x_1, x_2)$$

$$x_1 = f_\theta^{\boxed{-1}}(z_1)$$
$$x_2 = f_\theta^{\boxed{-1}}(z_2; x_1)$$
$$x_3 = f_\theta^{\boxed{-1}}(z_3; x_1, x_2)$$

$z_i \sim N(0,1)$

1) Sampling **Can be done in "D" pass (Slow)**

$$z_1 = f_\theta(x_1)$$
$$z_2 = f_\theta(x_2; x_1)$$
$$z_3 = f_\theta(x_3; x_1, x_2)$$

2) Density Evaluation **Can be done in 1 pass (Fast)**

3. Models using Autoregressive Flow

3-2. MAF

Masked Autoregressive Flow (Papamakarios et al, 2017)

MAF = Flow version of MADE

It also follows an autoregressive property $p(\mathbf{x}) = \prod_{i=1}^{D} p(x_i | \mathbf{x}_{1:i-1})$

3-2. MAF

Masked Autoregressive Flow (Papamakarios et al, 2017)

MAF = Flow version of MADE

It also follows an autoregressive property $p(\mathbf{x}) = \prod_{i=1}^{D} p(x_i | \mathbf{x}_{1:i-1})$

Autoregressive model whose conditionals are parameterized as **single Gaussians**

$$p\left(x_i \mid \mathbf{x}_{1:i-1}\right) = \mathcal{N}\left(x_i \mid \mu_i, \left(\exp \alpha_i\right)^2\right)$$

$$\text{where} \quad \mu_i = f_{\mu_i}\left(\mathbf{x}_{1:i-1}\right) \text{ and } \alpha_i = f_{\alpha_i}\left(\mathbf{x}_{1:i-1}\right)$$

3. Models using Autoregressive Flow

3-2. MAF

Masked Autoregressive Flow (Papamakarios et al, 2017)

MAF = Flow version of MADE

It also follows an autoregressive property $p(\mathbf{x}) = \prod_{i=1}^{D} p(x_i|\mathbf{x}_{1:i-1})$

Autoregressive model whose conditionals are parameterized as **single Gaussians**

$$p\left(x_i \mid \mathbf{x}_{1:i-1}\right) = \mathcal{N}\left(x_i \mid \mu_i, \left(\exp \alpha_i\right)^2\right)$$

$$\text{where} \quad \mu_i = \underline{f_{\mu_i}\left(\mathbf{x}_{1:i-1}\right)} \text{ and } \alpha_i = \underline{f_{\alpha_i}\left(\mathbf{x}_{1:i-1}\right)}$$

use “MADE”

3-2. MAF

Masked Autoregressive Flow (Papamakarios et al, 2017)

MAF = Flow version of MADE

It also follows an autoregressive property $p(\mathbf{x}) = \prod_{i=1}^{D} p(x_i|\mathbf{x}_{1:i-1})$

Autoregressive model whose conditionals are parameterized as **single Gaussians**

$$p\left(x_i \mid \mathbf{x}_{1:i-1}\right) = \mathcal{N}\left(x_i \mid \mu_i, (\exp\alpha_i)^2\right)$$

$$\text{where} \quad \mu_i = \underline{f_{\mu_i}\left(\mathbf{x}_{1:i-1}\right)} \text{ and } \alpha_i = \underline{f_{\alpha_i}\left(\mathbf{x}_{1:i-1}\right)}$$

use "MADE"

$\rightarrow$

$x_i = u_i \exp\alpha_i + \mu_i$

- $\mu_i = f_{\mu_i}\left(\mathbf{x}_{1:i-1}\right)$
- $\alpha_i = f_{\alpha_i}\left(\mathbf{x}_{1:i-1}\right)$
- $u_i \sim \mathcal{N}(0,1)$

3. Models using Autoregressive Flow

3-2. MAF

Masked Autoregressive Flow (Papamakarios et al, 2017)

MAF = Flow version of MADE

It also follows an autoregressive property $p(\mathbf{x}) = \prod_{i=1}^{D} p(x_i|\mathbf{x}_{1:i-1})$

(SAMPLING) We can generate data, using recursion :

$$x_i = u_i \exp \alpha_i + \mu_i$$

- $\mu_i = f_{\mu_i}(\mathrm{x}_{1:i-1})$
- $\alpha_i = f_{\alpha_i}(\mathrm{x}_{1:i-1})$
- $u_i \sim \mathcal{N}(0,1)$

$\Rightarrow$

1. $x_1 = f_{\mu_1} + \exp(f_{\sigma_1})z_1$ for $z_1 \sim N(0,1)$
2. $x_2 = f_{\mu_2}(x_1) + \exp(f_{\sigma_2}(x_1))z_2$ for $z_2 \sim N(0,1)$
3. $x_3 = f_{\mu_3}(x_1, x_2) + \exp(f_{\sigma_3}(x_1, x_2))z_3$ for $z_3 \sim N(0,1)$

3. Models using Autoregressive Flow

Inverse Autoregressive Flow

$$z_1 \sim p_\theta\left(z_1\right)$$
$$z_2 \sim p_\theta\left(z_2 \mid z_1\right)$$
$$z_3 \sim p_\theta\left(z_3 \mid z_1, z_2\right)$$

(Autoregressive Flow)

$$x_1 \sim p_\theta\left(x_1\right)$$
$$x_2 \sim p_\theta\left(x_2 \mid x_1\right)$$
$$x_3 \sim p_\theta\left(x_3 \mid x_1, x_2\right)$$

3. Models using Autoregressive Flow

Inverse Autoregressive Flow

$$z_1 \sim p_\theta(z_1)$$
$$z_2 \sim p_\theta(z_2 \mid z_1)$$
$$z_3 \sim p_\theta(z_3 \mid z_1, z_2)$$

(Autoregressive Flow)

$$x_1 \sim p_\theta(x_1)$$
$$x_2 \sim p_\theta(x_2 \mid x_1)$$
$$x_3 \sim p_\theta(x_3 \mid x_1, x_2)$$

$$x_1 = f_\theta(z_1)$$
$$x_2 = f_\theta(z_2; z_1)$$
$$x_3 = f_\theta(z_3; z_1, z_2)$$

1) Sampling

$$z_i \sim N(0,1)$$

3. Models using Autoregressive Flow

Inverse Autoregressive Flow

$$z_1 \sim p_\theta(z_1)$$
$$z_2 \sim p_\theta(z_2 \mid z_1)$$
$$z_3 \sim p_\theta(z_3 \mid z_1, z_2)$$

(Autoregressive Flow)

$$x_1 \sim p_\theta(x_1)$$
$$x_2 \sim p_\theta(x_2 \mid x_1)$$
$$x_3 \sim p_\theta(x_3 \mid x_1, x_2)$$

$$x_1 = f_\theta(z_1)$$
$$x_2 = f_\theta(z_2; z_1)$$
$$x_3 = f_\theta(z_3; z_1, z_2)$$

$$z_i \sim N(0, 1)$$

1) Sampling **Can be done in 1 pass (Fast)**

3. Models using Autoregressive Flow

Inverse Autoregressive Flow

$$z_1 \sim p_\theta(z_1)$$
$$z_2 \sim p_\theta(z_2 \mid z_1)$$
$$z_3 \sim p_\theta(z_3 \mid z_1, z_2)$$

(Autoregressive Flow)

$$x_1 \sim p_\theta(x_1)$$
$$x_2 \sim p_\theta(x_2 \mid x_1)$$
$$x_3 \sim p_\theta(x_3 \mid x_1, x_2)$$

$$z_i \sim N(0,1)$$

$$x_1 = f_\theta(z_1)$$
$$x_2 = f_\theta(z_2; z_1)$$
$$x_3 = f_\theta(z_3; z_1, z_2)$$

1) Sampling **Can be done in 1 pass (Fast)**

$$z_1 = f_\theta^{\boxed{-1}}(x_1)$$
$$z_2 = f_\theta^{\boxed{-1}}(x_2; z_1)$$
$$z_3 = f_\theta^{\boxed{-1}}(x_3; z_1, z_2)$$

2) Density Evaluation

3. Models using Autoregressive Flow

Inverse Autoregressive Flow

$$z_1 \sim p_\theta\left(z_1\right)$$
$$z_2 \sim p_\theta\left(z_2 \mid z_1\right)$$
$$z_3 \sim p_\theta\left(z_3 \mid z_1, z_2\right)$$

$$x_1 \sim p_\theta\left(x_1\right)$$
$$x_2 \sim p_\theta\left(x_2 \mid x_1\right)$$
$$x_3 \sim p_\theta\left(x_3 \mid x_1, x_2\right)$$

(Autoregressive Flow)

$$x_1 = f_\theta(z_1)$$
$$x_2 = f_\theta(z_2; z_1)$$
$$x_3 = f_\theta(z_3; z_1, z_2)$$

$z_i \sim N(0,1)$

1) Sampling **Can be done in 1 pass (Fast)**

$$z_1 = f_\theta^{\boxed{-1}}(x_1)$$
$$z_2 = f_\theta^{\boxed{-1}}(x_2; z_1)$$
$$z_3 = f_\theta^{\boxed{-1}}(x_3; z_1, z_2)$$

2) Density Evaluation **Can be done in "D" pass (Slow)**

3. Models using Autoregressive Flow

3-3. IAF

Inverse Autoregressive Flow (Kingma et al, 2016)

IAF = Inverse version of MAF

Instead of "X", **"U" passes through the flow!**

$$\boxed{x_i = u_i \exp \alpha_i + \mu_i}$$

- $u_i \sim N(0,1) \qquad i = 1, \ldots, D$

< MAF >

$$\boxed{u_i = \frac{(x_i - \mu_i)}{\exp \alpha_i}}$$

- $u_1 \sim N(0,1)$

< IAF >

3. Models using Autoregressive Flow

3-3. IAF

Inverse Autoregressive Flow (Kingma et al, 2016)

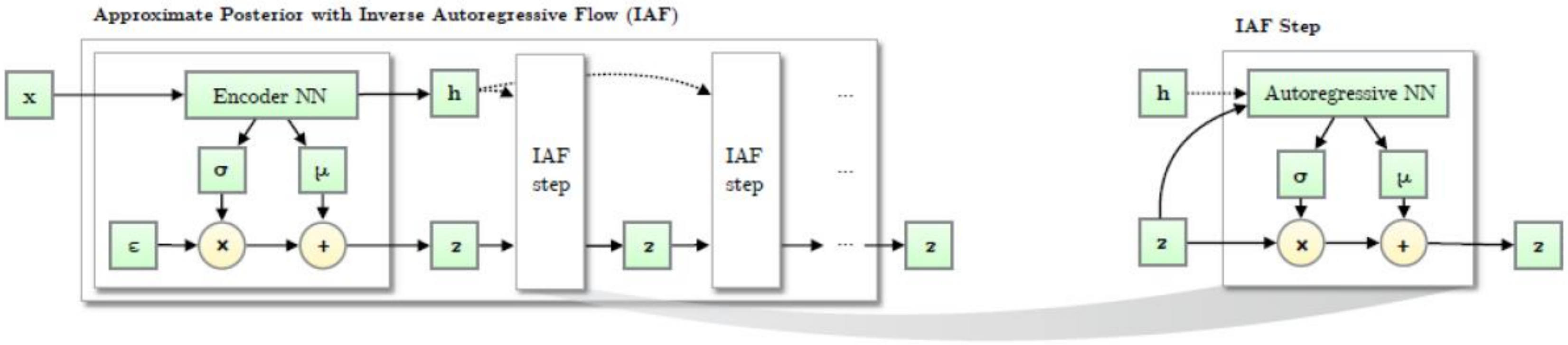


Figure 2: Like other normalizing flows, drawing samples from an approximate posterior with Inverse Autoregressive Flow (IAF) consists of an initial sample **z** drawn from a simple distribution, such as a Gaussian with diagonal covariance, followed by a chain of nonlinear invertible transformations of **z**, each with a simple Jacobian determinants.

3. Models using Autoregressive Flow

3-3. IAF

Inverse Autoregressive Flow (Kingma et al, 2016)

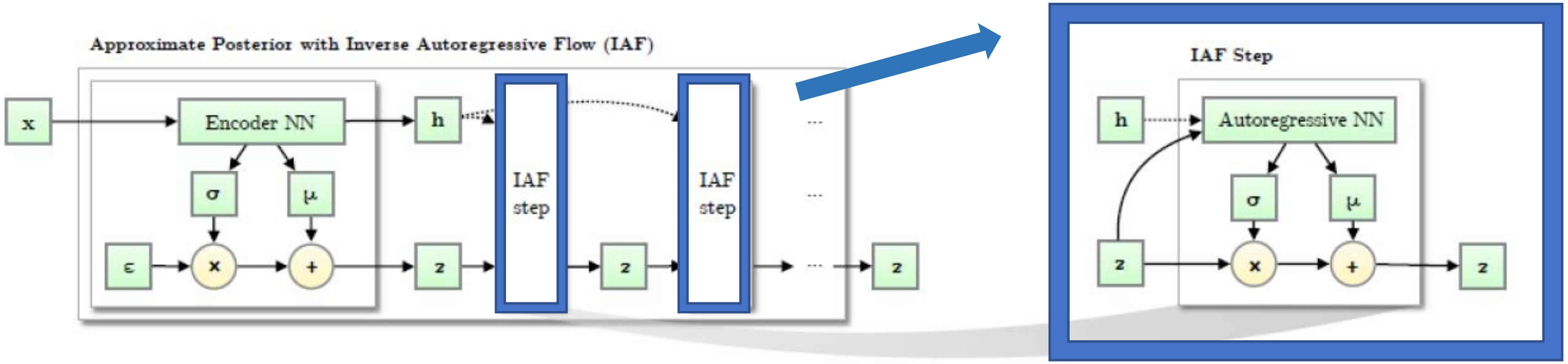
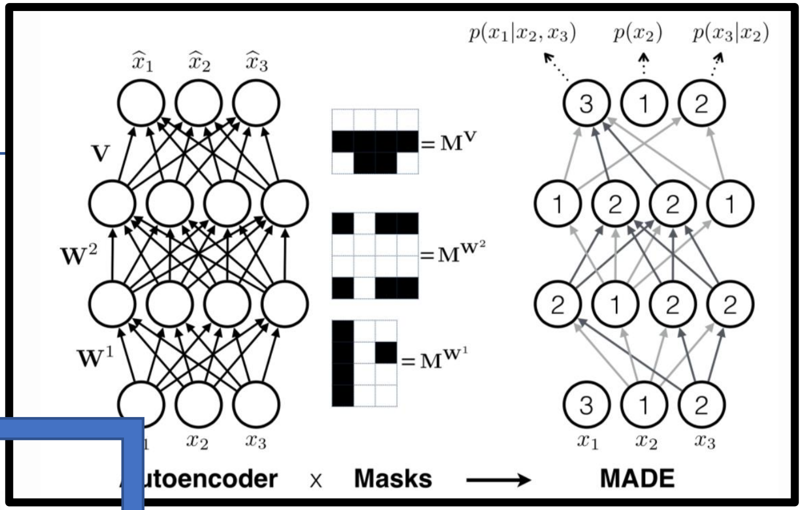


Figure 2: Like other normalizing flows, drawing samples from an approximate posterior with Inverse Autoregressive Flow (IAF) consists of an initial sample **z** drawn from a simple distribution, such as a Gaussian with diagonal covariance, followed by a chain of nonlinear invertible transformations of **z**, each with a simple Jacobian determinants.

3. Models using Autoregressive Flow

3-3. IAF

Inverse Autoregressive Flow (Kingma et al, 2016)



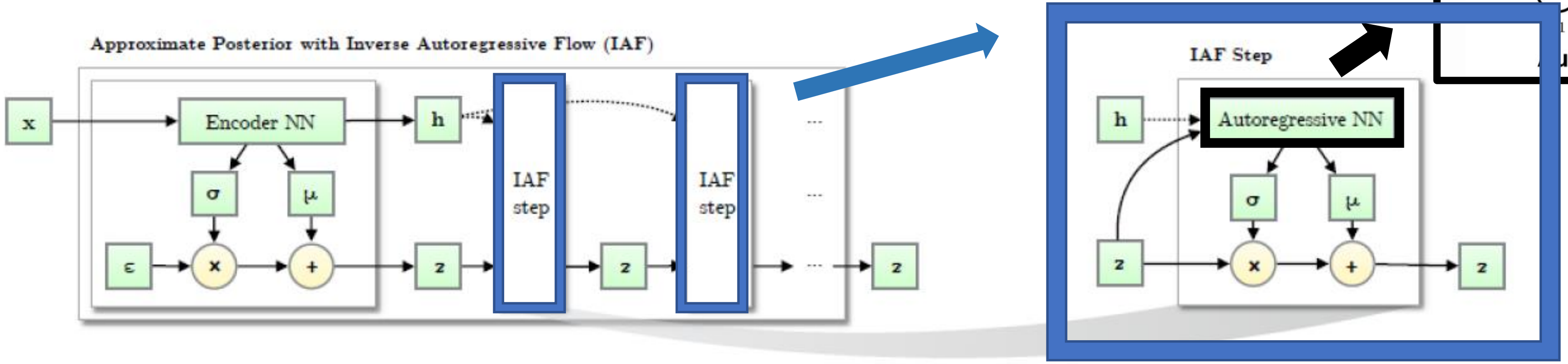


Figure 2: Like other normalizing flows, drawing samples from an approximate posterior with Inverse Autoregressive Flow (IAF) consists of an initial sample **z** drawn from a simple distribution, such as a Gaussian with diagonal covariance, followed by a chain of nonlinear invertible transformations of **z**, each with a simple Jacobian determinants.

3-3. IAF

Algorithm 1: Pseudo-code of an approximate posterior with Inverse Autoregressive Flow (IAF)

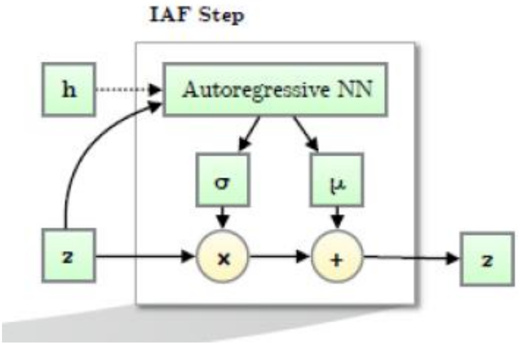
Data:

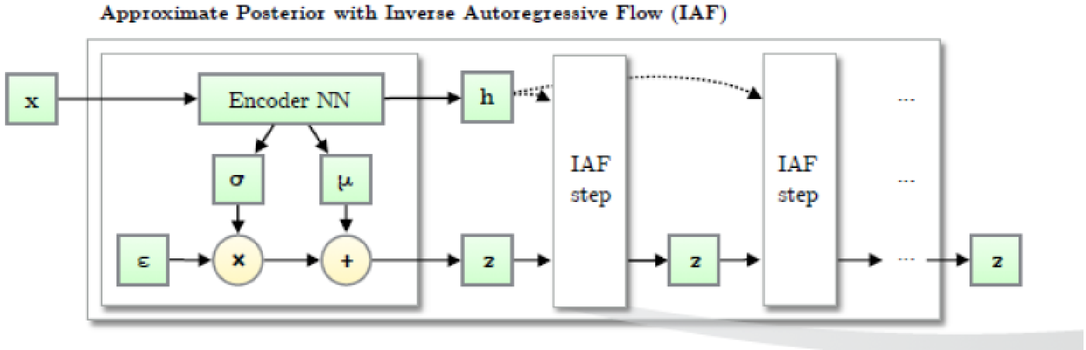
$\mathbf{x}$: a datapoint, and optionally other conditioning information
$\boldsymbol{\theta}$: neural network parameters
$\texttt{EncoderNN}(\mathbf{x};\boldsymbol{\theta})$: encoder neural network, with additional output $\mathbf{h}$
$\texttt{AutoregressiveNN}[*](\mathbf{z},\mathbf{h};\boldsymbol{\theta})$: autoregressive neural networks, with additional input $\mathbf{h}$
$\texttt{sum}(.)$: sum over vector elements
$\texttt{sigmoid}(.)$: element-wise sigmoid function

Result:

$\mathbf{z}$: a random sample from $q(\mathbf{z}|\mathbf{x})$, the approximate posterior distribution
l: the scalar value of $\log q(\mathbf{z}|\mathbf{x})$, evaluated at sample 'z'

$[\boldsymbol{\mu},\boldsymbol{\sigma},\mathbf{h}] \leftarrow \texttt{EncoderNN}(\mathbf{x};\boldsymbol{\theta})$
$\epsilon \sim \mathcal{N}(0, I)$
$\mathbf{z} \leftarrow \boldsymbol{\sigma} \odot \epsilon + \boldsymbol{\mu}$
$l \leftarrow -\texttt{sum}(\log \boldsymbol{\sigma} + \frac{1}{2}\epsilon^2 + \frac{1}{2}\log(2\pi))$
for $t \leftarrow 1$ **to** T **do**
  $[\mathbf{m},\mathbf{s}] \leftarrow \texttt{AutoregressiveNN}[t](\mathbf{z},\mathbf{h};\boldsymbol{\theta})$
  $\boldsymbol{\sigma} \leftarrow \texttt{sigmoid}(\mathbf{s})$
  $\mathbf{z} \leftarrow \boldsymbol{\sigma} \odot \mathbf{z} + (1-\boldsymbol{\sigma}) \odot \mathbf{m}$
  $l \leftarrow l - \texttt{sum}(\log \boldsymbol{\sigma})$
end





3-3. IAF

IAF vs MAF

- [MAF] FAST density evaluation, SLOW sampling
  - μ_i and α_i are directly computed from previous "data variables $x_{1:i-1}$"
  - capable of calculating the density $p(x)$ of any data point in one pass

    but sampling requires D sequential passes

$$z_1 = f_\theta(x_1)$$
$$z_2 = f_\theta(x_2; x_1)$$
$$z_3 = f_\theta(x_3; x_1, x_2)$$

- [IAF] FAST sampling, SLOW density evaluation
  - μ_i and α_i are directly computed from previous "random numbers $u_{1:i-1}$"
  - sampling requires only one pass

    but calculating the density $p(x)$ of any data point requires D passes

$$x_1 = f_\theta(z_1)$$
$$x_2 = f_\theta(z_2; z_1)$$
$$x_3 = f_\theta(z_3; z_1, z_2)$$

Summary

Flow-based Model

- Change of variables
- Normalizing Flow
- Models using NF (Normalizing Flow)
    - Real NVP : affine coupling layer
    - NICE : additive coupling layer
    - Glow : 1x1 convolution & PLU decomposition
- Models using AF (Autoregressive Flow)
    - MADE
    - MAF : flow using MADE
    - IAF : inverse version of MAF

Thank You