

# Waveglow

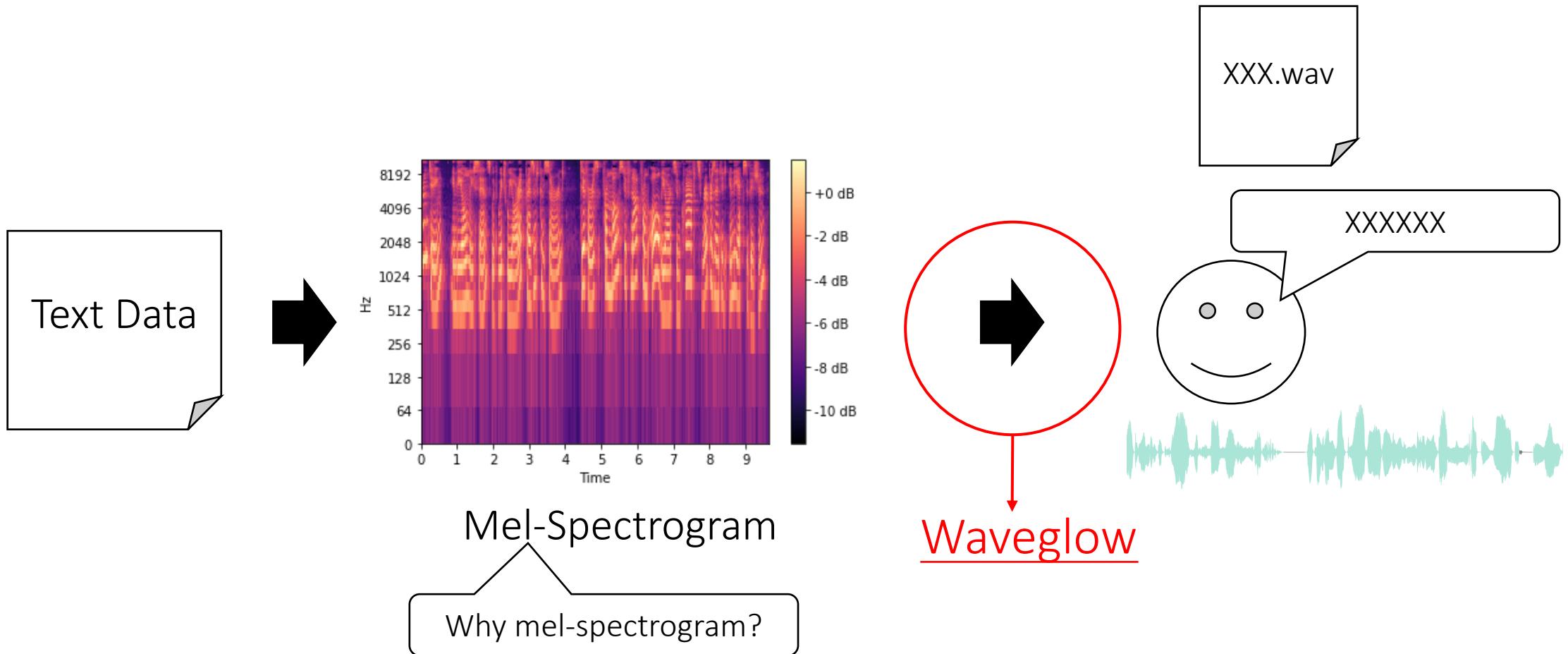
# Agenda

1. Text-to-speech synthesis
2. Flow-based deep generative models
3. Waveglow
4. Code Analysis

# 1. Text-to-speech synthesis

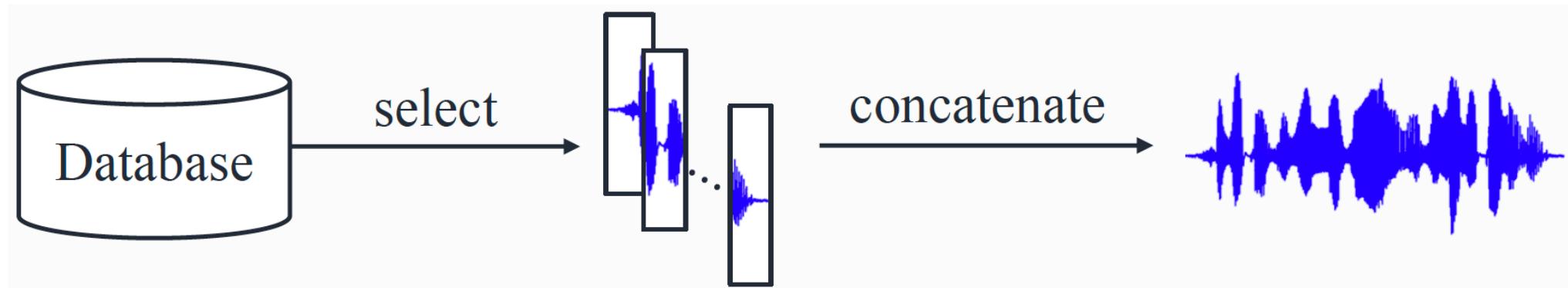
# What is Waveglow?

- Waveglow is used for text-to-speech synthesis



# Text-to-speech synthesis

- Concatenative TTS
  - Very large database of short speech fragments are recorded from a single speaker and then recombined to form complete utterances.



Sounds natural 😊



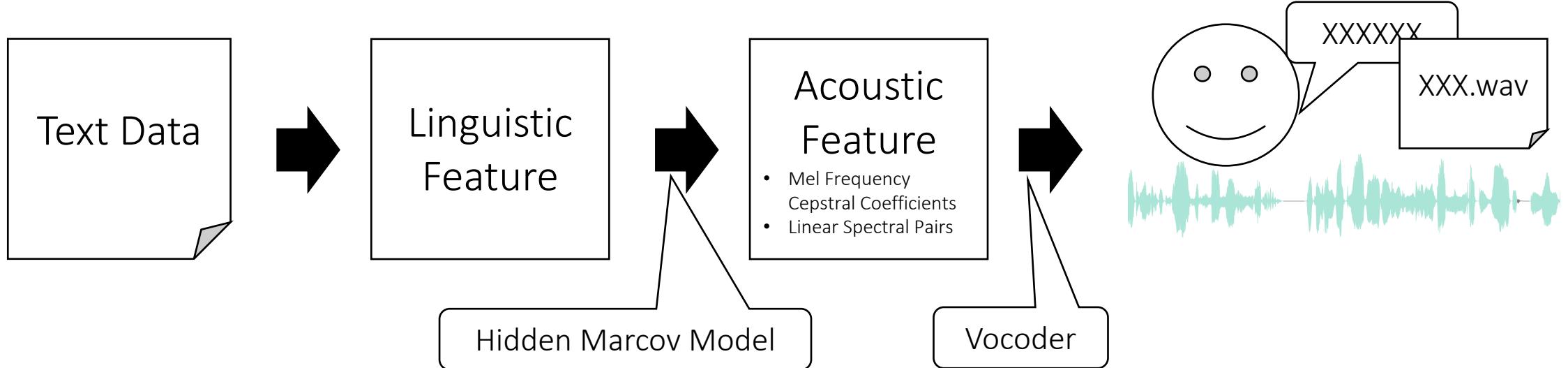
Can't modify the voice without recording a whole database 😞

<https://deepmind.com/blog/article/wavenet-generative-model-raw-audio>

[https://www.slideshare.net/t\\_koshikawa/wavenet-87105461?from\\_action=save](https://www.slideshare.net/t_koshikawa/wavenet-87105461?from_action=save)

# Text-to-speech synthesis

- Statistical Parametric TTS
  - Divide the problem into 3 subproblems and solve them separately.



Sounds less natural 😞



Can modify the voice without recording a whole database 😊

Zen H, Tokuda K, Black AW. Statistical parametric speech synthesis. speech communication. 2009;51(11):1039-64.

<https://deepmind.com/blog/article/wavenet-generative-model-raw-audio>

# Text-to-speech synthesis

- DL-based TTS
  - Use DL-based models
  - Merge some of the steps

Char2Wav,...

Tacotron2,...

Text Data

Wavenet

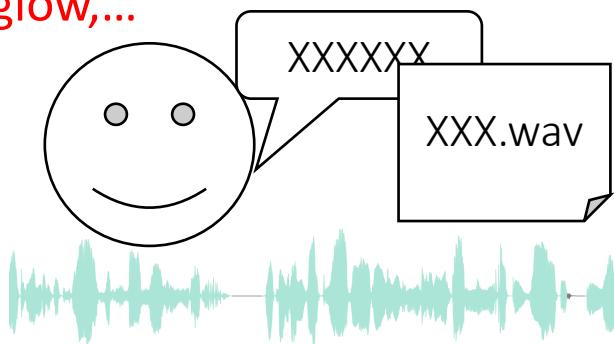


Linguistic Feature

Wavenet Vocoder, Waveglow,...

Acoustic Feature

- Mel Frequency Cepstral Coefficients
- Linear Spectral Pairs



# Text-to-speech synthesis

- Concatenative TTS 
- Parametric TTS 
- DL-based TTS (wavenet) 

## 2. Flow-based generative models

# What is "Flow-based deep generative model"?

- GAN is popular as a generative model
- Flow-based model is another type of generative model
- It is often used for image generation
- WaveGlow is flow-based,  
the most similar to "Glow"

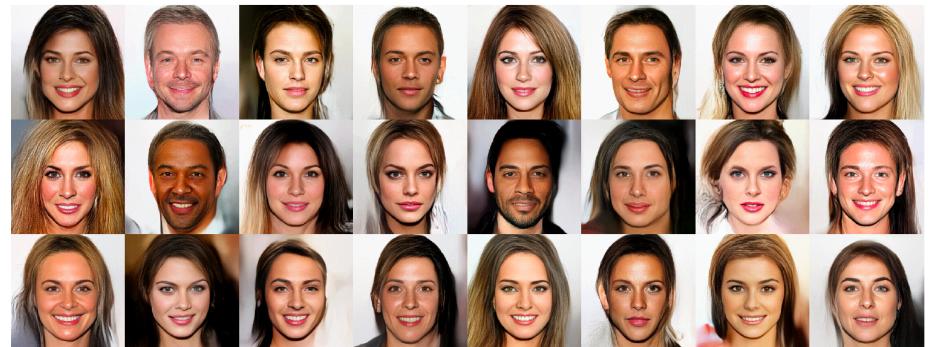
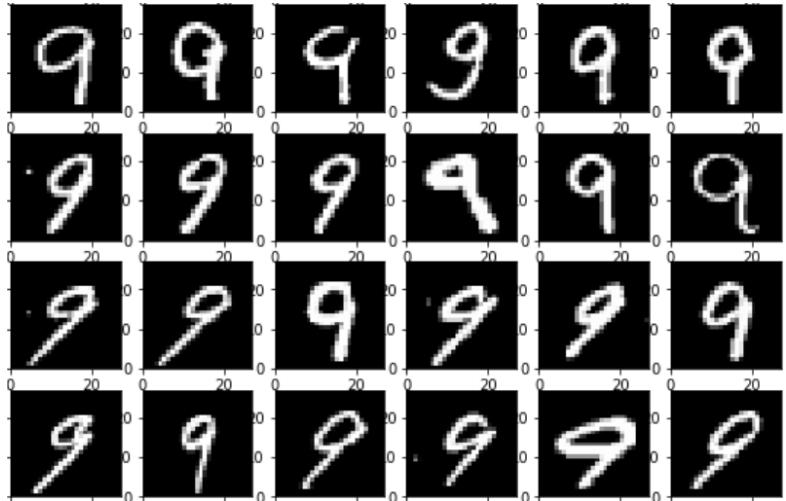


Figure 4: Random samples from the model, with temperature 0.7.

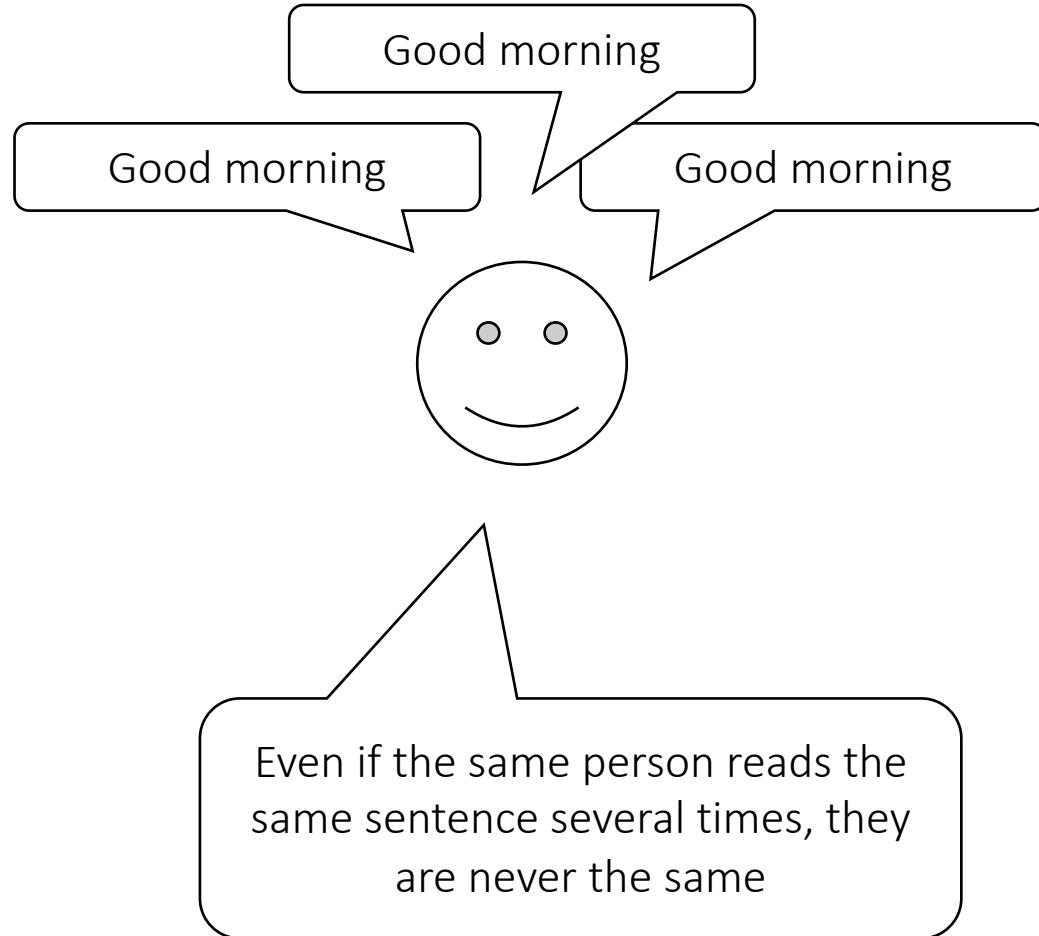


Figure 5: Linear interpolation in latent space between real images.

# What is “Generative Model” in general?



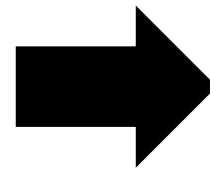
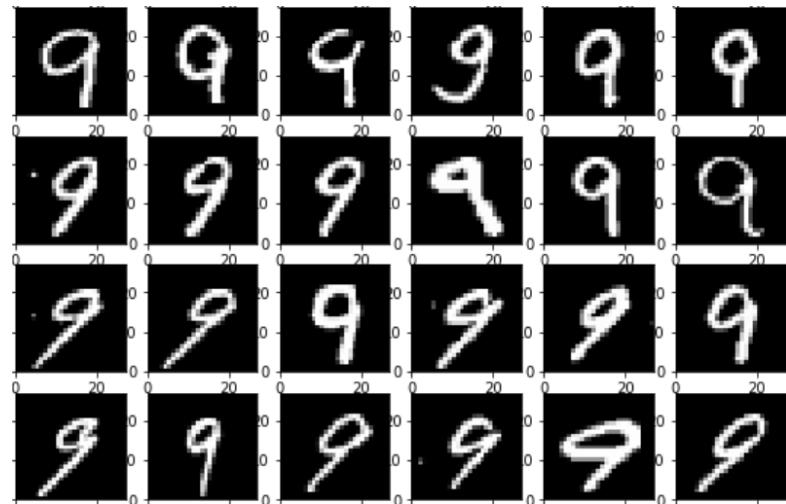
Same number, but so different...  
Even when one person writes the same number multiple times, they're not the same



Even if the same person reads the same sentence several times, they are never the same

Human speech, writing, drawing, or photo image are not deterministic...

# Represent it as “Probability distribution”...

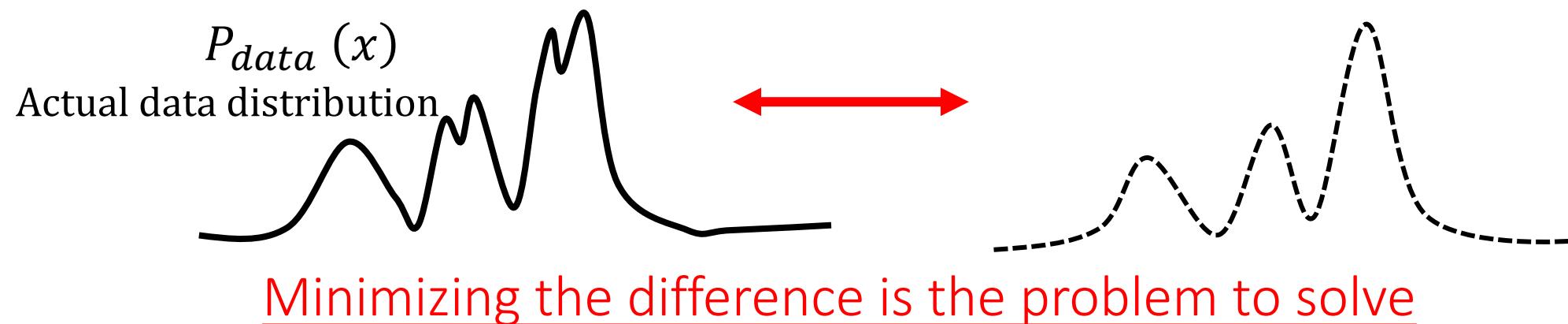


Probability to output  $x$  when  $\theta$  is given

$$P(x; \theta)$$

$x$  : Output (audio, image, etc ...)

$\theta$  : Parameters



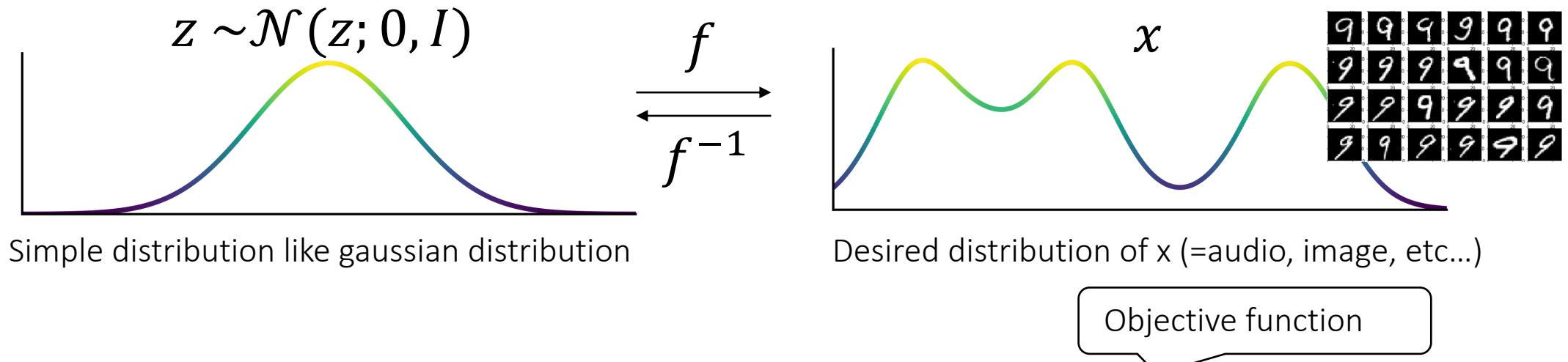
# How can this be solved?

	Approach	Model Architecture	How to train / Objective Function	Computational Cost
GAN	Model a generator and train it to indirectly minimize the difference between $P(x; \theta)$ and $P_{data}(x)$	Generator $x = G(z)$ $z : Latent\ Variable$ Discriminator $D(x)$	Adversarial learning (Min-Max game)	Low
VAE (Variational Auto Encoder)		Encoder $z = E(x)$ $z : Latent\ Variable$ Decoder $x = D(z)$	Maximize lower bound	Low
Flow-based		Flow-based invertible model $x = f_0 \cdot f_1 \cdots f_k(z)$ $z : Latent\ Variable$	Maximize log likelihood	Low
Auto-regressive	Model probability distribution itself	Probability distribution conditioned on the past data $\prod_i P(x_i ; x_1, x_2, \dots, x_{i-1})$	Maximize log likelihood	High (parallel computation is difficult due to auto-regression)

Wavenet is an auto-regressive model, but very slow...

# Basic idea of “Flow-based generative model”

- What if there is an invertible transformation  $f$  from a simple distribution  $z$  to the desired distribution of  $x$  ?



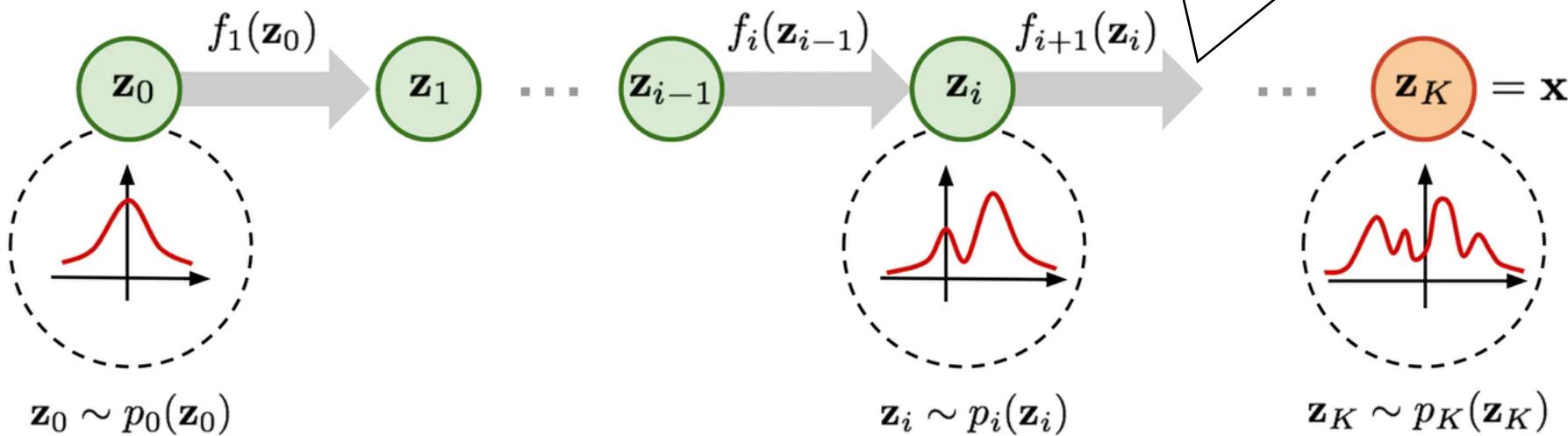
- For training, find  $f^{-1}(x)$ , which maximizes the probability distribution  $P(x)$  with given training dataset of  $x$  (=audio, image, etc.)
- For inference, randomly pick  $z$  from the simple distribution and calculating  $f(z)$

Difficult to find a single  $f$  because the desired distribution is usually complex

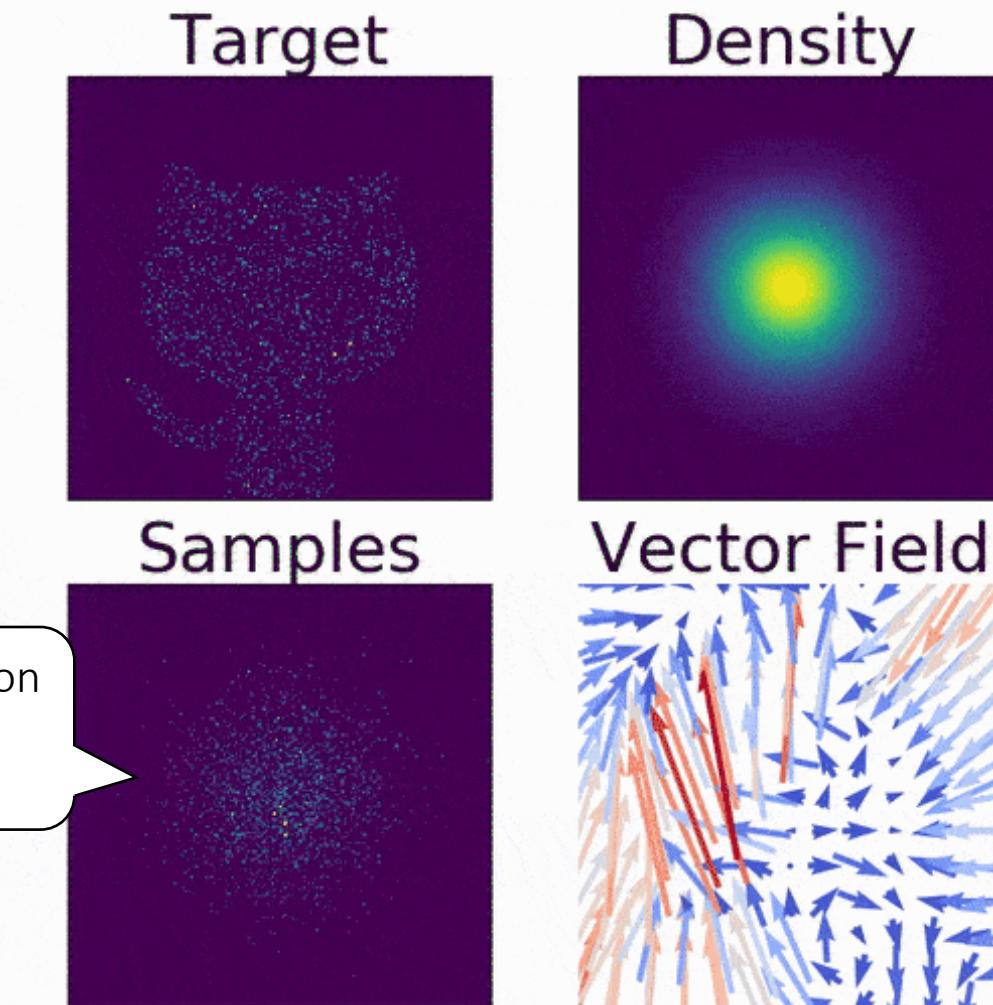
# How can such complex $f$ be found?

- Think about applying multiple simple functions :  $f_i \in f_1, f_2, \dots, f_K$ 
  - Invertible  $z_i = f_i(z_{i-1})$   
 $z_{i-1} = f_i^{-1}(z_i)$
  - $z_0, z_1, \dots, z_k = x$  have the same dimension

Each  $f_i$  is simple, but  
 $f_1 \cdot f_2 \cdots f_K(z)$  can represent  $x$  with  
complex probability distribution



# 2D Example of flow-based generative model



Simple 2D gaussian distribution  
gradually transfer to the  
complex distribution...

[https://github.com/rtqichen/ffjord/blob/master/assets/github\\_flow.gif](https://github.com/rtqichen/ffjord/blob/master/assets/github_flow.gif)

It can generate more complex, multi-dimensional data.

How can  $x = f_0 \cdot f_1 \cdots f_k(z)$  maximize the likelihood  $P(x)$ ?

We don't know the likelihood  $P(x)....$

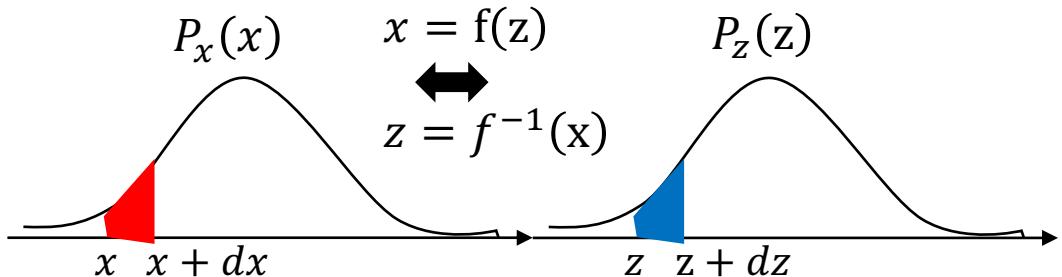
- Convert  $P(x)$  to  $P(z)$  using “change-of-variable” technique in the probability density function

Change of variable (1D)

$$P_x(x)dx = P_z(z)dz$$

$$\begin{aligned} P_x(x) &= P_z(z) \left| \frac{dz}{dx} \right| \\ &= P_z(z) \left| \frac{df^{-1}(x)}{dx} \right| \end{aligned}$$

Blue area and red area should stay the same through the change of variable



Change of variable (Multi-dimensional)

$$\begin{aligned} P_x(x) &= P_z(z) \left| \det \frac{dz}{dx} \right| \\ &= P_z(z) \left| \det \frac{df^{-1}(x)}{dx} \right| \end{aligned}$$

: Jacobian determinant

# How can $x = f_0 \cdot f_1 \cdots f_k(z)$ maximize the likelihood $P(x)$ ?

- At each layer, the likelihood can be represented with the variables of the previous layer

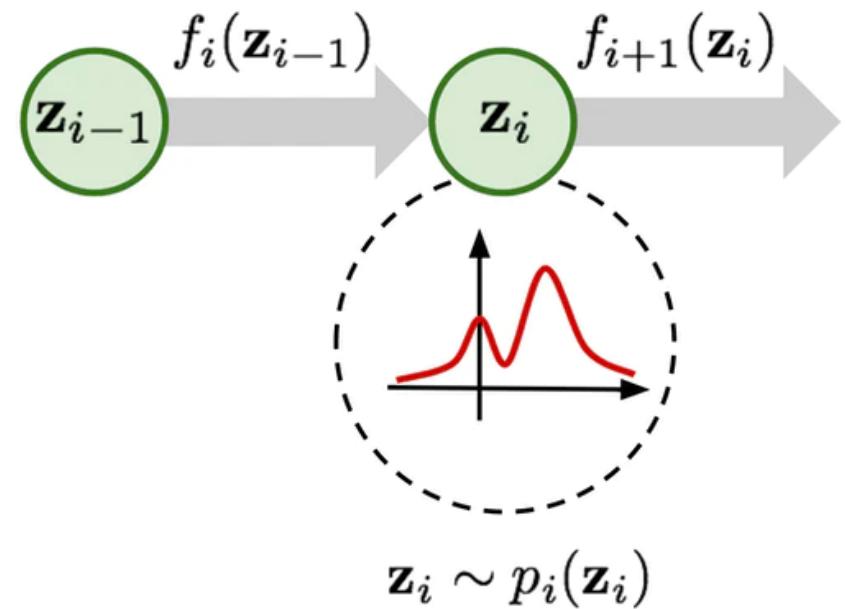
Change of variable (Multi-dimensional)

$$P_x(x) = P_z(z) \left| \det \frac{dz}{dx} \right|$$

$$P_i(z_i) = P_{i-1}(z_{i-1}) \left| \det \frac{dz_{i-1}}{dz_i} \right|$$

$$= P_{i-1}(z_{i-1}) \left| \det \left( \frac{dz_i}{dz_{i-1}} \right)^{-1} \right|$$

$$= P_{i-1}(z_{i-1}) \left| \det \frac{df_i(z_{i-1})}{dz_{i-1}} \right|^{-1}$$



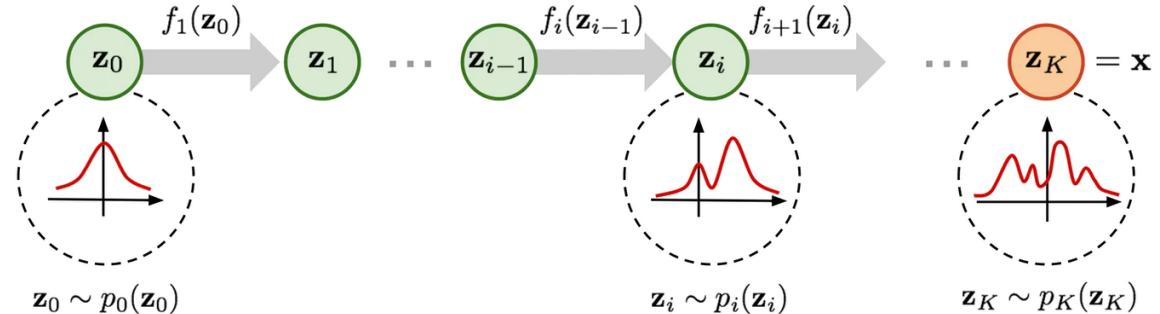
$$\det(X^{-1}) = (\det(X))^{-1}$$

How can  $x = f_0 \cdot f_1 \cdots f_k(z)$  maximize the likelihood  $P(x)$ ?

Change of variable (Multi-dimensional)

$$= P_{i-1}(z_{i-1}) \left| \det \frac{df_i(z_{i-1})}{dz_{i-1}} \right|^{-1}$$

- By applying this recursively...



$$\begin{aligned} \log P_x(x) &= \log P_K(z_K) = \log P_{K-1}(z_{K-1}) - \log \left| \det \frac{df_K(z_{K-1})}{dz_{K-1}} \right| \\ &= \left( \log P_{K-2}(z_{K-2}) - \log \left| \det \frac{df_K(z_{K-2})}{dz_{K-2}} \right| \right) - \log \left| \det \frac{df_K(z_{K-1})}{dz_{K-1}} \right| \end{aligned}$$

$z_0$  is simple gaussian distribution, so we can calculate  $P_0(z_0)$  easily!

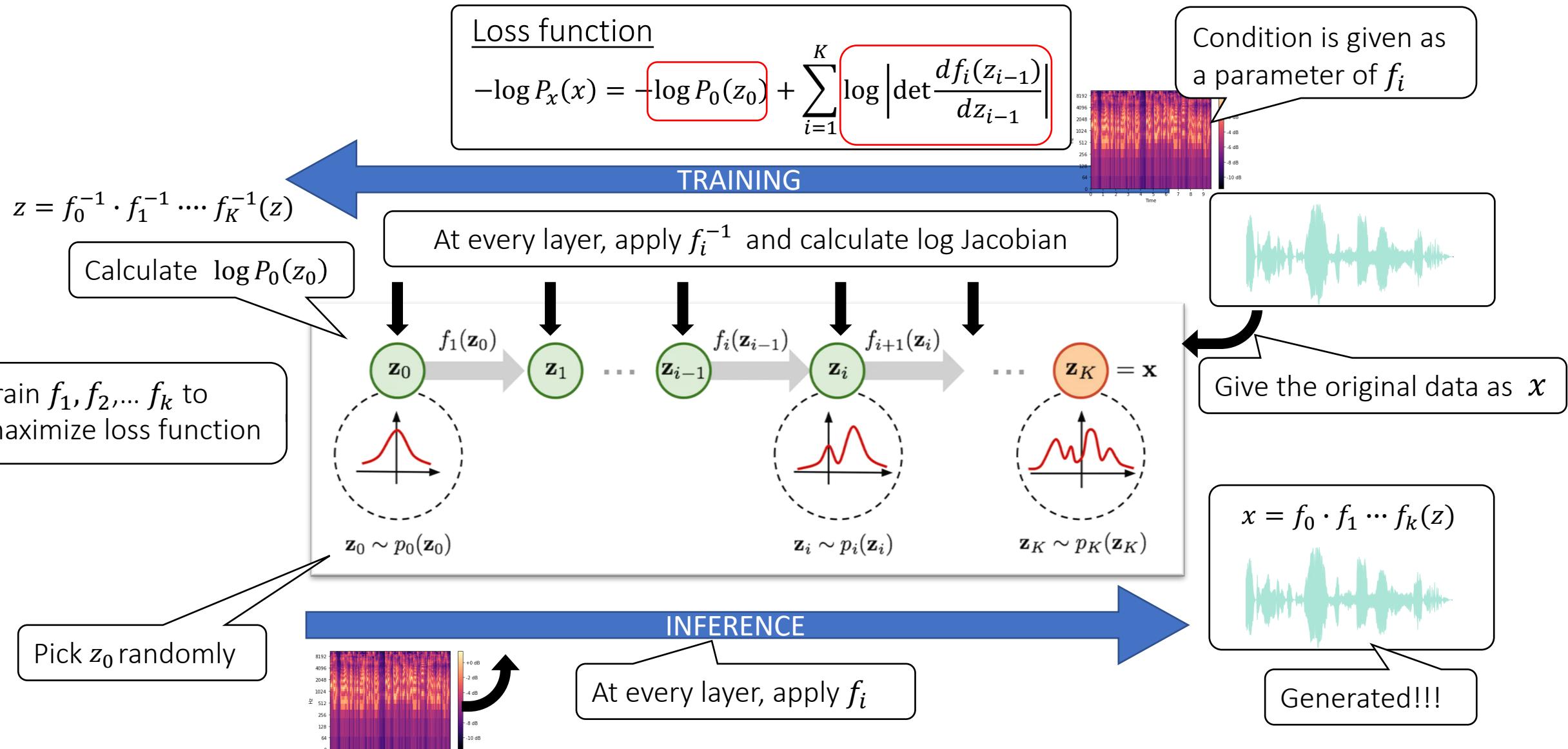
⋮

$$= \log P_0(z_0) - \sum_{i=1}^K \log \left| \det \frac{df_i(z_{i-1})}{dz_{i-1}} \right|$$

Log Jacobian at each layer...

For the ease of calculation, instead of maximizing the likelihood, minimize the negative log likelihood as loss function

# How does actual training & inference work?



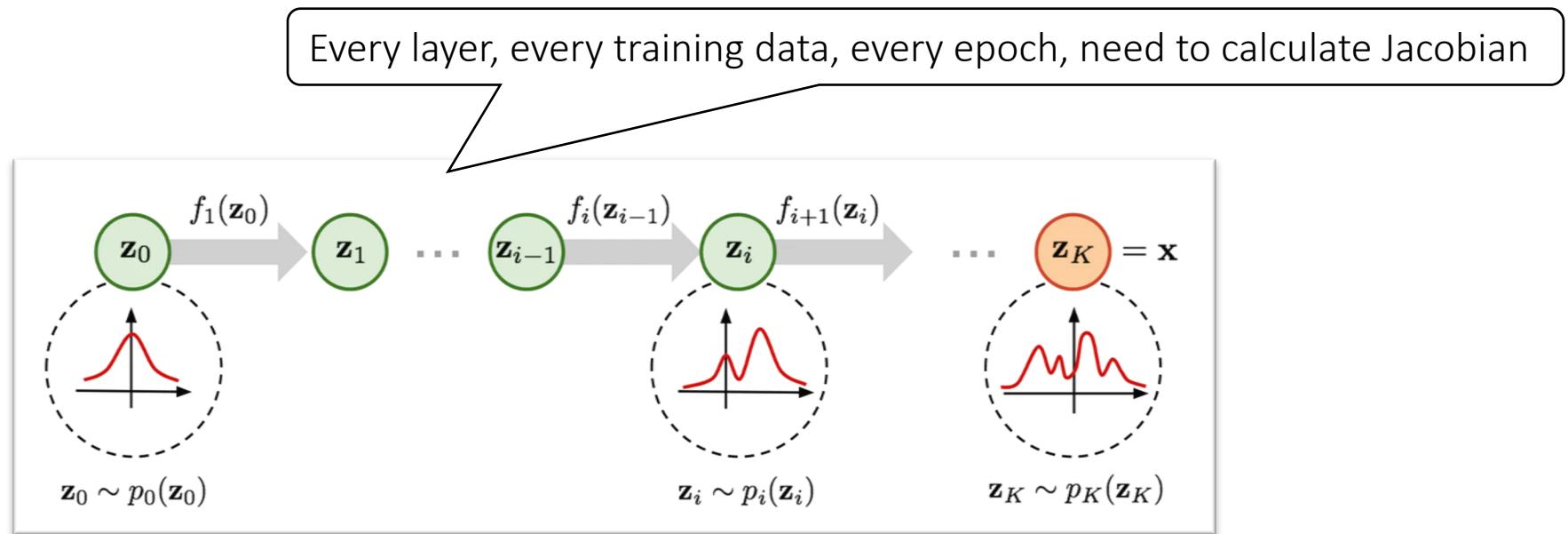
# Flow-based generative models

TABLE 1  
List of Normalizing Flows for which we show performance results.

Architecture	Coupling function	Flow name
Coupling, 3.4.1	Affine, 3.4.4.1	RealNVP Glow
	Mixture CDF, 3.4.4.3	Flow++
	Splines, 3.4.4.4	quadratic (C) cubic RQ-NSF(C)
	Piecewise Bijective, 3.4.4.7	RAD
Autoregressive, 3.4.2	Affine	MAF
	Polynomial, 3.4.4.6	SOS
	Neural Network, 3.4.4.5	NAF UMNN
	Splines	quadratic (AR) RQ-NSF(AR)
		iResNet Residual flow
Residual, 3.5		FFJORD
ODE, 3.6.1		

# Computational cost of flow-based models

- Computational cost of Jacobian :  $O(D^3)$  (D : dimension)



As dimension becomes higher, computation cost becomes very big 😩

# How can computational cost be reduced?

- Introduce “Coupling layer”

- Divide input  $x$  into 2 blocks  $x = [x_{1:d}, x_{d+1:D}]$
- Introduce the following transformation

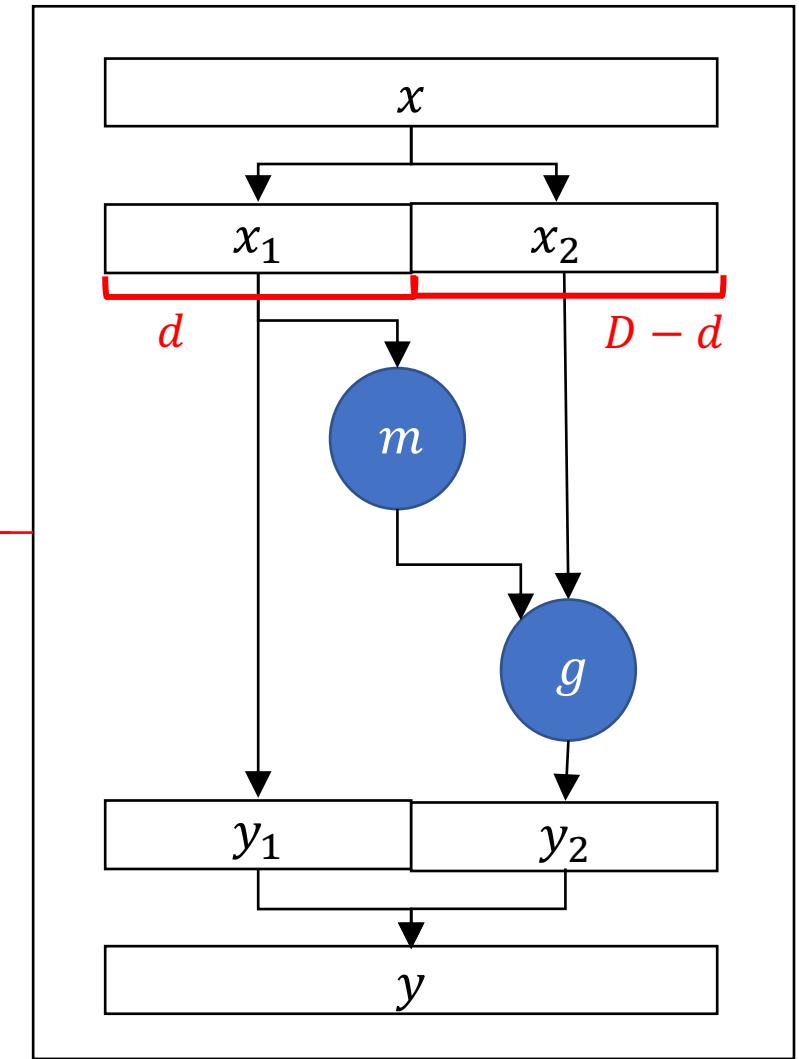
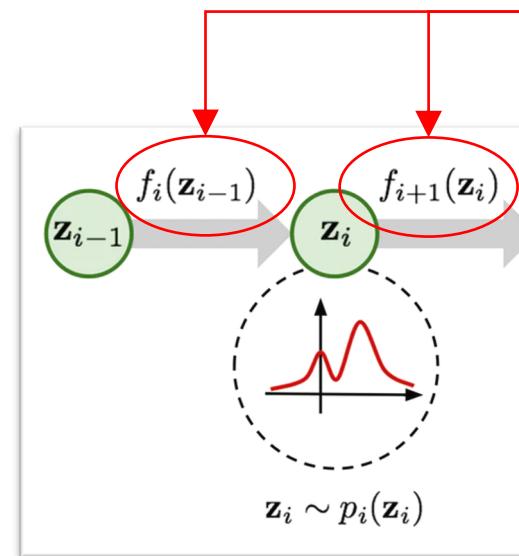
$$\begin{cases} y_1 = x_1 \\ y_2 = g(x_2, m(x_1)) \end{cases}$$

$g$  : Invertible function

$m$  : Arbitrary function

$$\Rightarrow \begin{cases} x_1 = y_1 \\ x_2 = g^{-1}(y_2) - m(x_1) \end{cases}$$

This layer is invertible!



# How can computational cost be reduced?

$$\begin{cases} y_1 = x_1 \\ y_2 = g(x_2, m(x_1)) \end{cases} \Rightarrow \frac{dy_1}{dx_1} = I_d \quad \frac{dy_1}{dx_2} = 0$$

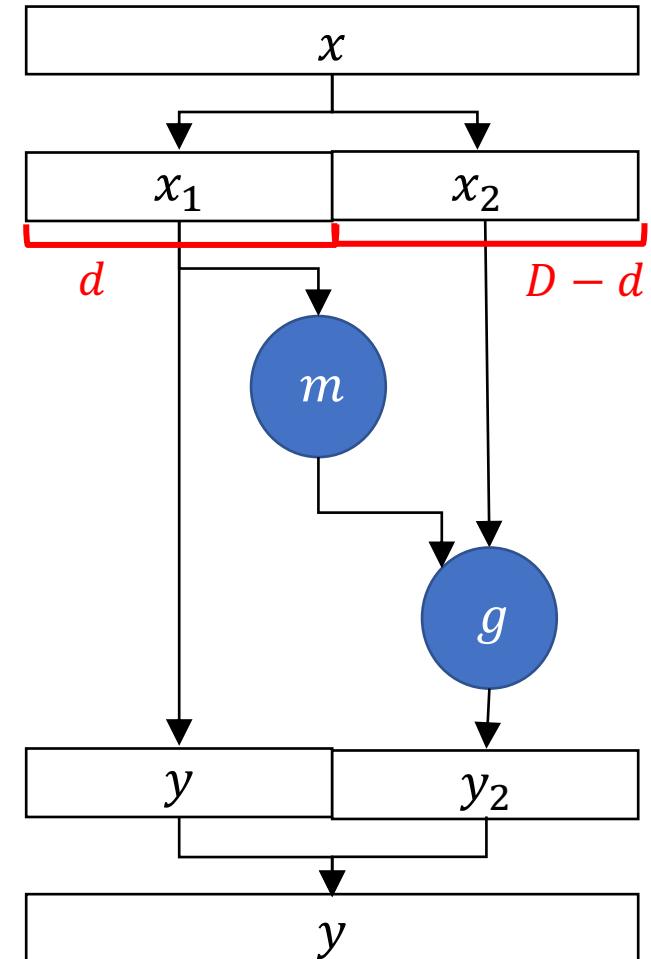
$g$  : Invertible function

$m$  : Arbitrary function

$$\frac{dy}{dx} = \begin{bmatrix} \frac{dy_1}{dx_1} & \frac{dy_1}{dx_2} \\ \frac{dy_2}{dx_1} & \frac{dy_2}{dx_2} \end{bmatrix} = \begin{bmatrix} I_d & 0 \\ \frac{dy_2}{dx_1} & \frac{dy_2}{dx_2} \end{bmatrix}$$

$$\det \frac{dy}{dx} = \det \frac{dy_2}{dx_2} = \det \frac{dg}{dx_2}$$

Computational cost :  $O(D^3) \rightarrow O((D - d)^3)$

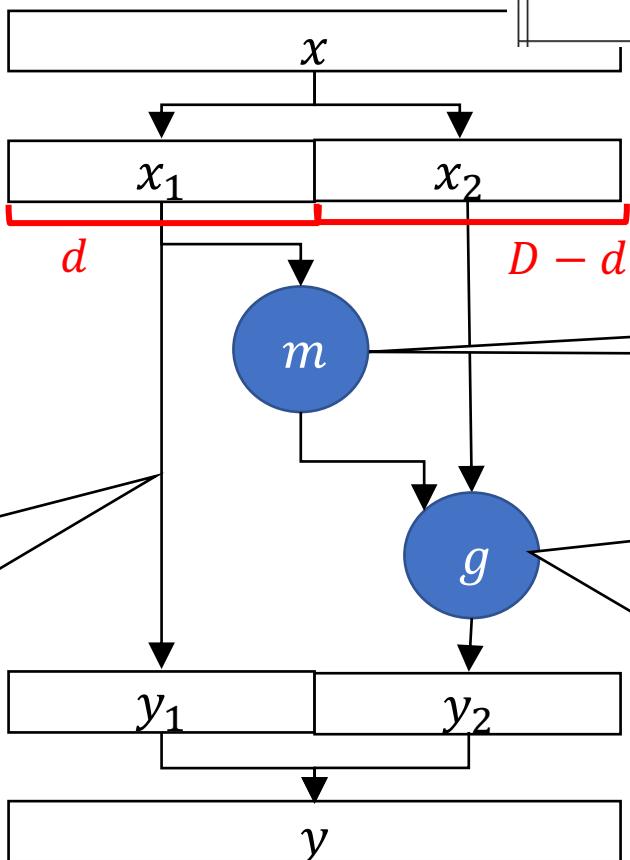


# Various types of coupling flow-based models

Architecture	Coupling function	Flow name
Coupling, 3.4.1	Affine, 3.4.4.1	RealNVP Glow
	Mixture CDF, 3.4.4.3	Flow++
	Splines, 3.4.4.4	quadratic (C) cubic RQ-NSF(C)
	Piecewise Bijective, 3.4.4.7	RAD

$$\begin{cases} y_1 = x_1 \\ y_2 = g(x_2, m(x_1)) \end{cases}$$

$g$  : Invertible function  
 $m$  : Arbitrary function



Severe restriction..  
A few methods have been proposed to reorder the channels.

This function can be designed freely.  
Challenging point when designing  $g$  is ...  

- Can it represent a transformation to complex distribution as whole?
- It should be invertible.
- Computational cost of  $\det \frac{dg}{dx_2}$  can be high...