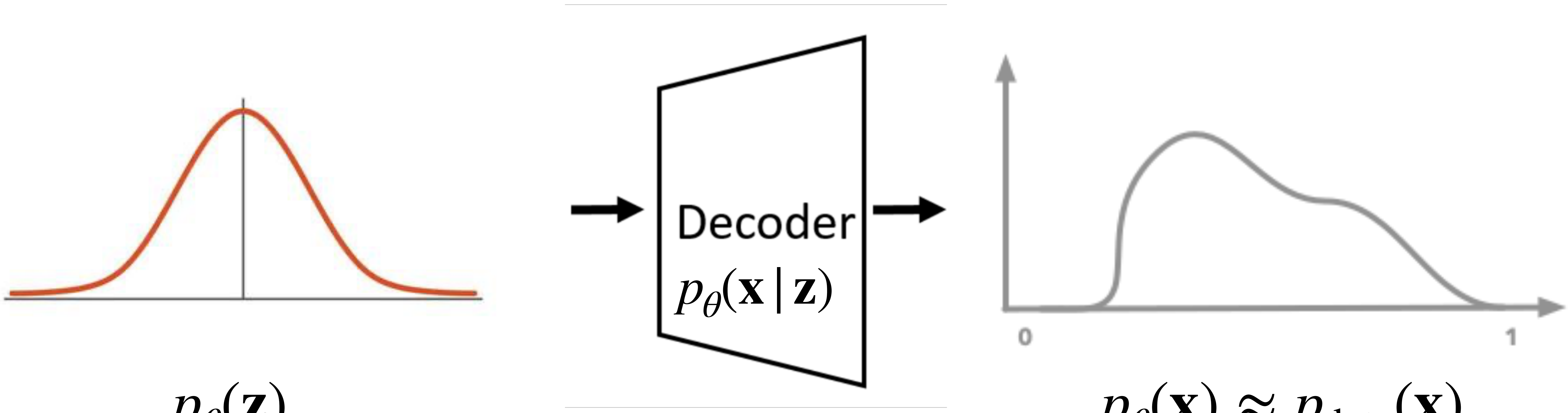
20. Generative Models (cont’d)

EEC454 Introduction to Machine Learning Systems

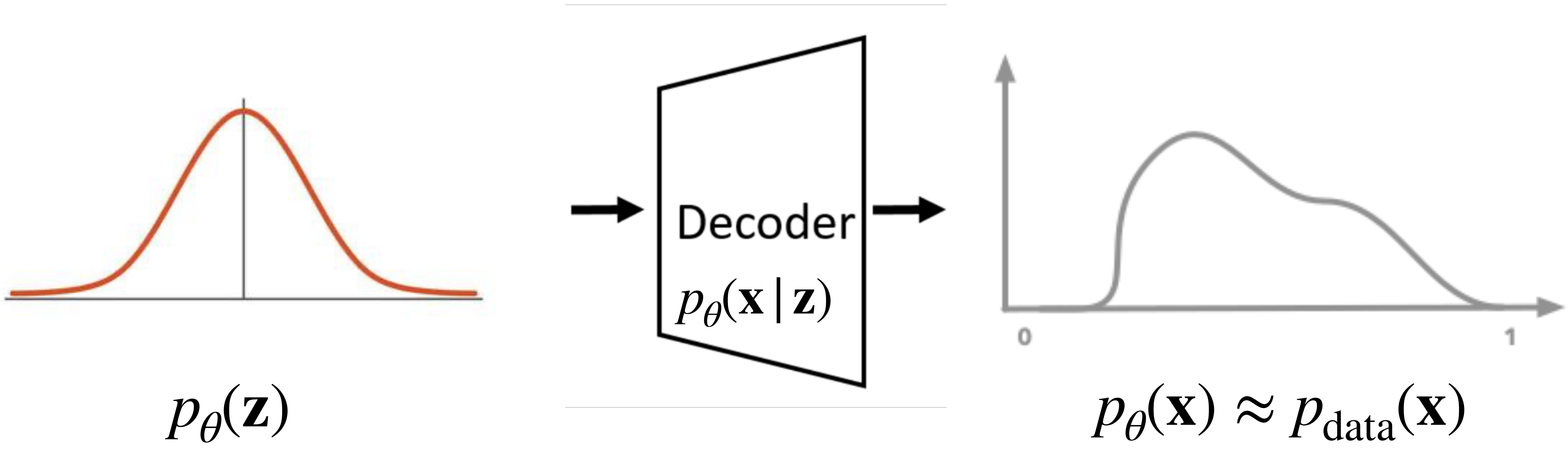
2023 Fall, Jaeho Lee

Recap: Variational Autoencoder

- Train a decoder and a distribution such that
  if we send in a distribution, we get a data-generating distribution.
  - For simplicity, we select θ so that $p_\theta(\mathbf{z})$ is $\mathcal{N}(0, I_k)$.



$p_\theta(\mathbf{z})$ $\quad$ $p_\theta(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$



- Similar to Naïve Bayes, we want to optimize the log probability

$$\max_{\theta} \sum_{i=1}^{n} \log p_\theta(\mathbf{x}_i)$$

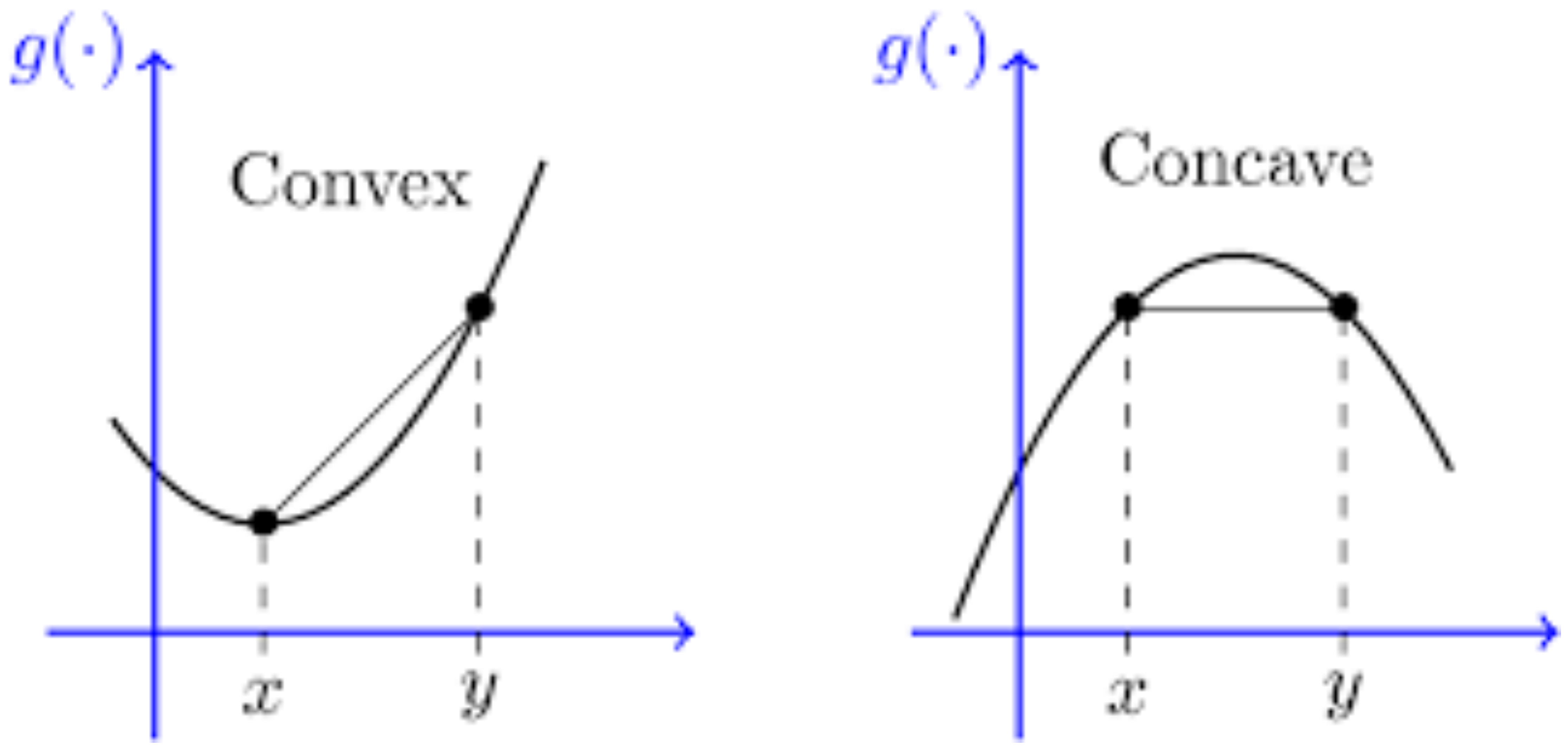
- Unfortunately, computing the marginal distribution is intractible:

$$p_\theta(\mathbf{x}_i) = \int p_\theta(\mathbf{x}_i \mid \mathbf{z}) p_\theta(\mathbf{z}) \, \mathrm{d}\mathbf{z}$$

Idea: Evidence Lower bound

- **Idea.** We maximize the lower bound of $p_\theta(\mathbf{x})$, not itself.
- **Tool.** Jensen's inequality
  - For a concave function $f(\,\cdot\,)$, we have

$$\mathbb{E}[f(X)] \leq f(\mathbb{E}[X])$$



- Compute the lower bound, for some **arbitrary** $q_\phi(\mathbf{z})$

$$\log p_\theta(\mathbf{x}) = \log \int p_\theta(\mathbf{z}) p_\theta(\mathbf{x} \mid \mathbf{z}) \, d\mathbf{z}$$

$$= \log \int q_\phi(\mathbf{z}) \frac{p_\theta(\mathbf{z})}{q_\phi(\mathbf{z})} p_\theta(\mathbf{x} \mid \mathbf{z}) \, d\mathbf{z} \qquad \text{(any } q_\phi \text{ works; take max)}$$

$$\geq \int q_\phi(\mathbf{z}) \cdot \log \left[\frac{p_\theta(\mathbf{z})}{q_\phi(\mathbf{z})} p_\theta(\mathbf{x} \mid \mathbf{z}) \right] \, d\mathbf{z} \qquad \text{(Jensen's ineq.)}$$

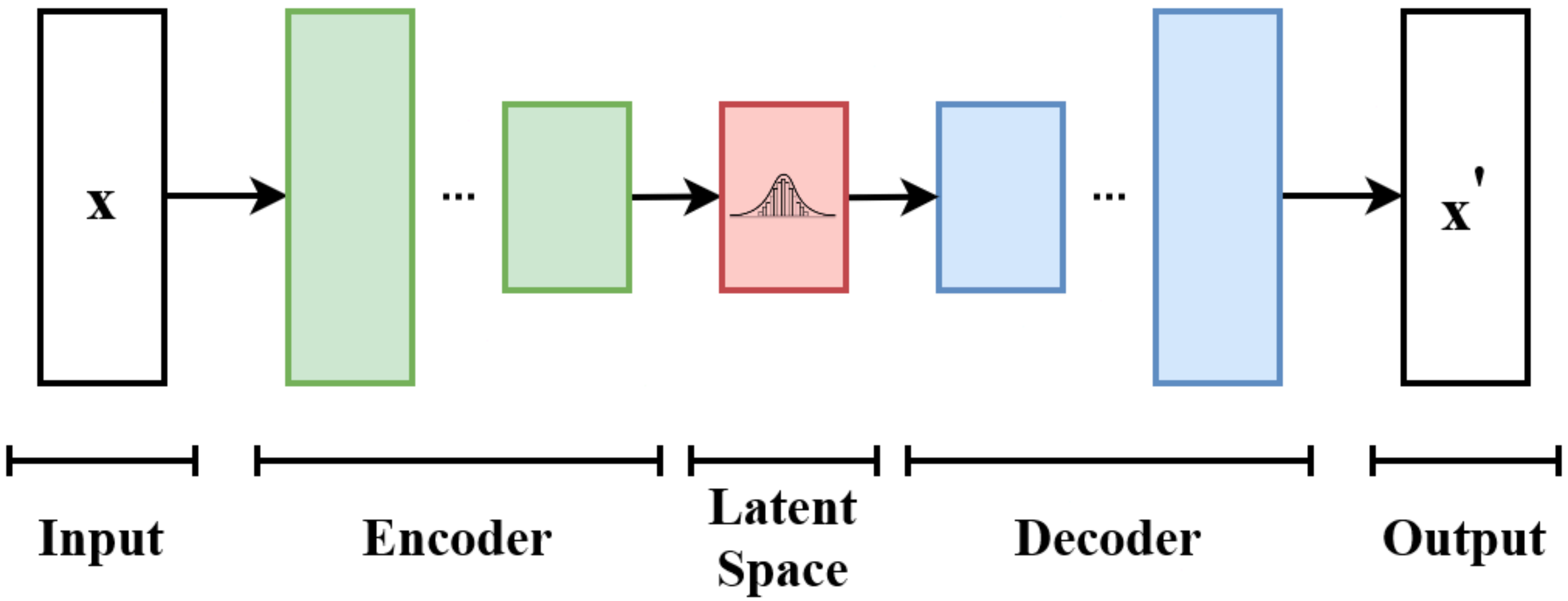
$$= -D(q_\phi(\mathbf{z}) \| p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi}[\log p_\theta(\mathbf{x} \mid \mathbf{z})]$$

- The optimal $q_\phi(\mathbf{z})$ may depend on $\mathbf{x}$... thus write as $q_\phi(\mathbf{z} \mid \mathbf{x})$

- Thus, we have

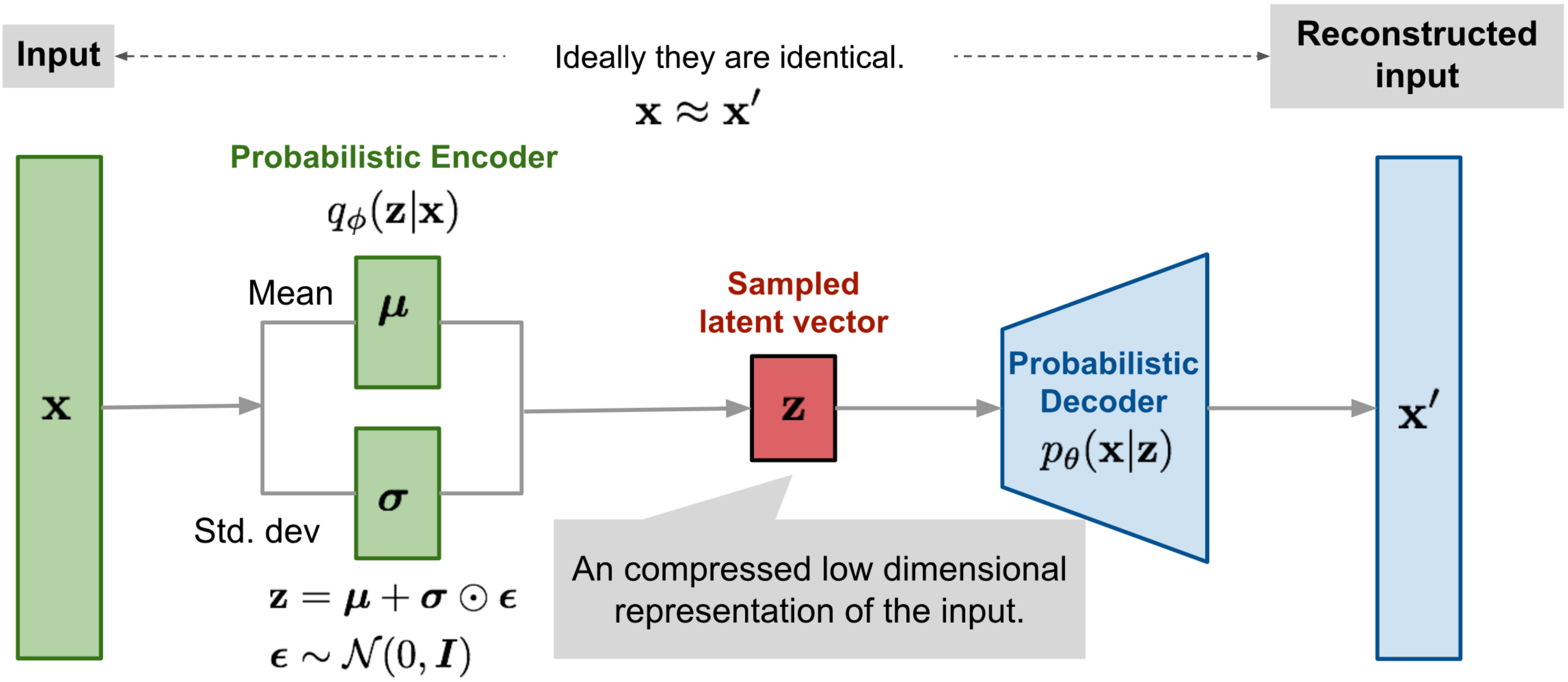
$$\max_{\theta} \log p_\theta(\mathbf{x}_i) \geq \max_{\theta} \max_{\phi} \left(- D(q_\phi(\mathbf{z} \,|\, \mathbf{x}_i) \| p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\cdot|\mathbf{x}_i)}[\log p_\theta(\mathbf{x}_i \,|\, \mathbf{z})] \right)$$

- In VAE, we jointly train a probabilistic encoder that expresses $q_\phi(\mathbf{z} \,|\, \mathbf{x}_i)$
    - **Question.** How to implement a probabilistic function?



Idea: Reparameterization Trick

- **Idea.** We model $q_\phi(\mathbf{z} \mid \mathbf{x})$ as a conditional Gaussian $\mathcal{N}(\mu_\mathbf{x}, \sigma_\mathbf{x}^2)$, and let the function learn $\mu_\mathbf{x}, \sigma_\mathbf{x}$ instead.



- Now, look at the optimization problem

$$\max_{\theta} \max_{\phi} \left(-D(q_\phi(\mathbf{z} \mid \mathbf{x}_i) \| p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\cdot \mid \mathbf{x}_i)}[\log p_\theta(\mathbf{x}_i \mid \mathbf{z})] \right)$$

- **Second term.** If we model with

$$p_\theta(\mathbf{x}_i \mid \mathbf{z}) = \mathcal{N}(f_\theta(\mathbf{z}), \eta \cdot I_d),$$

then this is equivalent to

$$-\mathbb{E}_{q_\phi(\cdot \mid \mathbf{x}_i)} \left[\frac{1}{2\eta} \| \mathbf{x}_i - f_\theta(\mathbf{z}_i) \|^2 \right] + \text{const}.$$

(i.e., simply use the squared loss!)

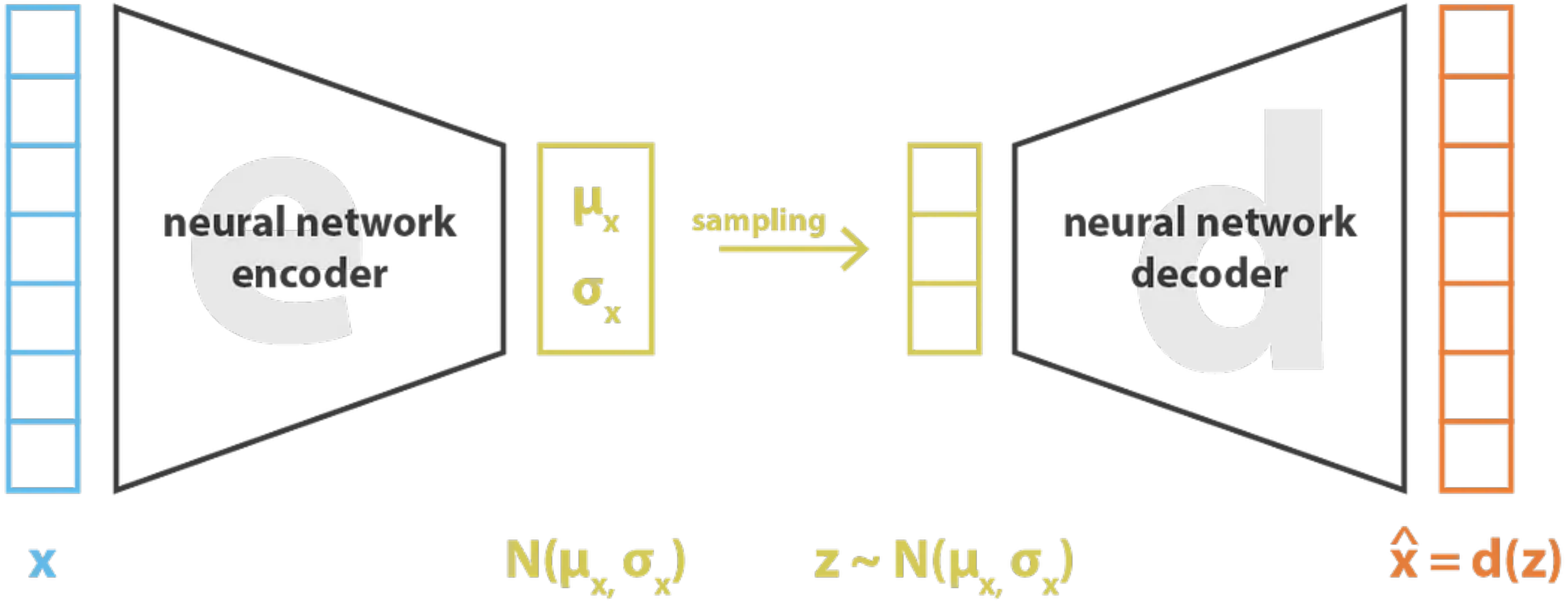
$$\max_{\theta} \max_{\phi} \left(- D(q_\phi(\mathbf{z} \mid \mathbf{x}_i) \| p_\theta(\mathbf{z})) - \frac{1}{2\eta} \mathbb{E}_{q_\phi(\cdot|\mathbf{x}_i)}[\|\mathbf{x}_i - f(\mathbf{z}_i)\|^2] \right)$$

- **First term.** If we use the Gaussian encoder

$$q_\phi = \mathcal{N}(\mu_{\mathbf{x}_i}, \sigma_{\mathbf{x}_i} \cdot I_k),$$

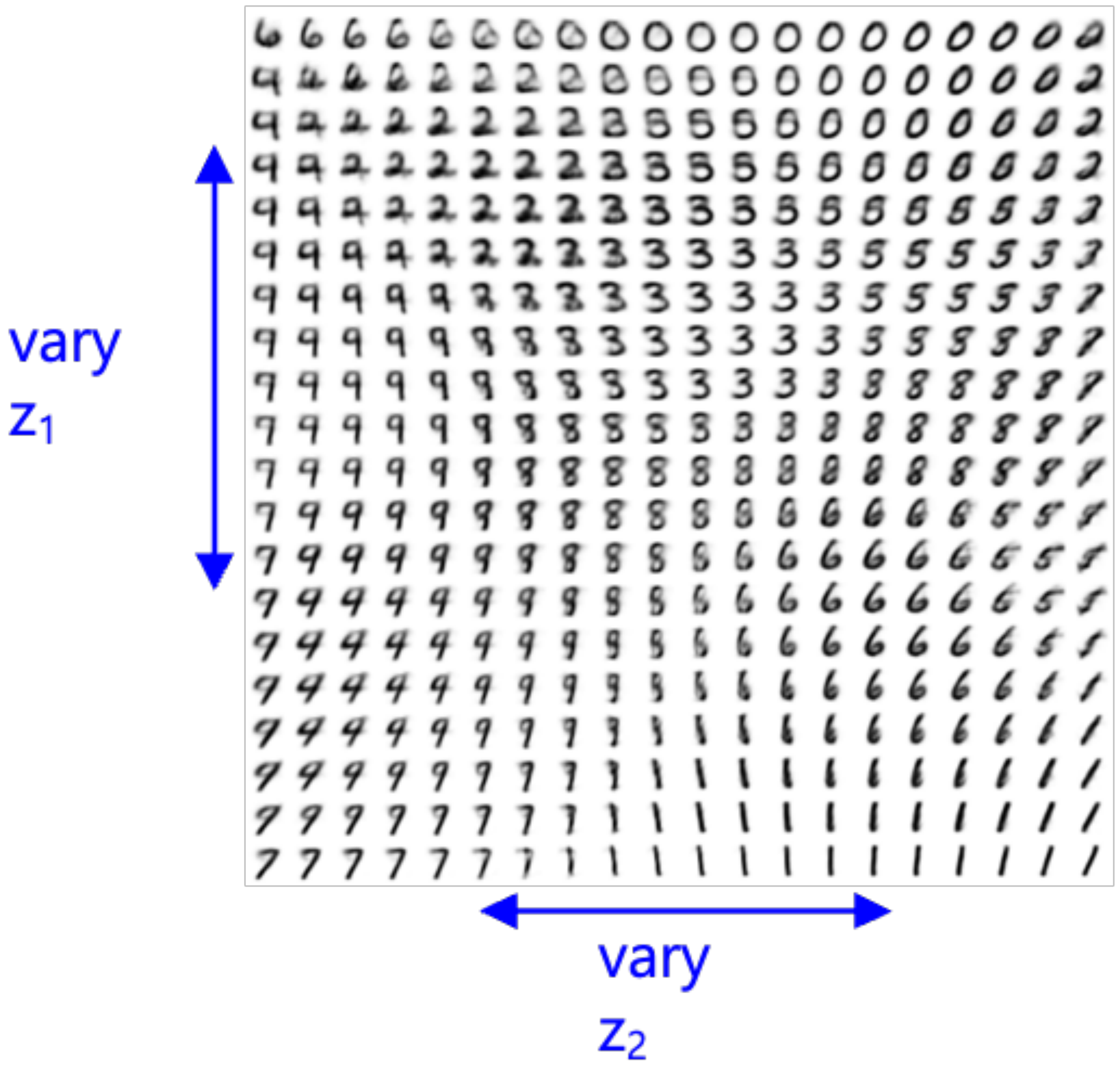
then this is nothing but squared regularizers on μ, σ!

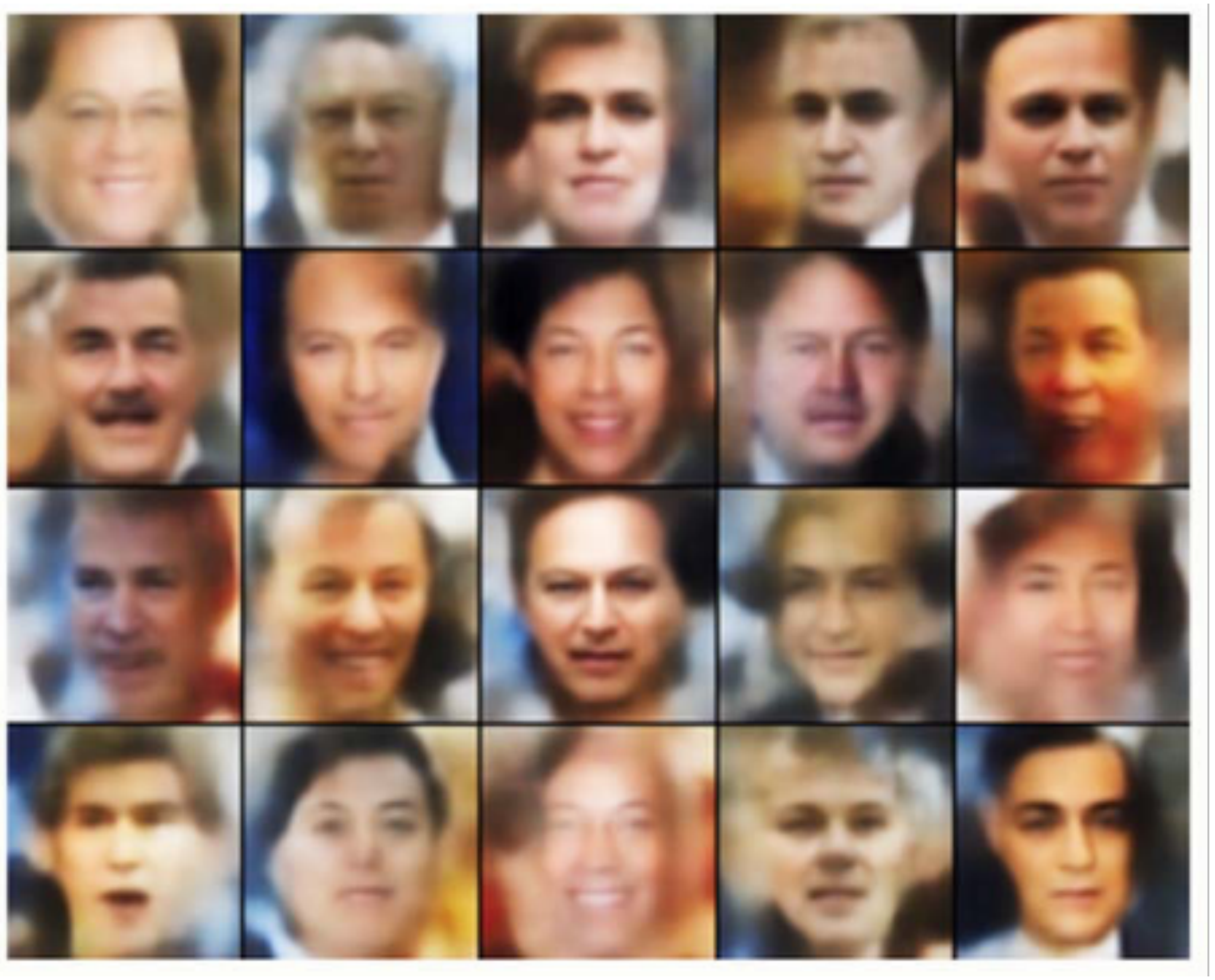
(check by yourself)

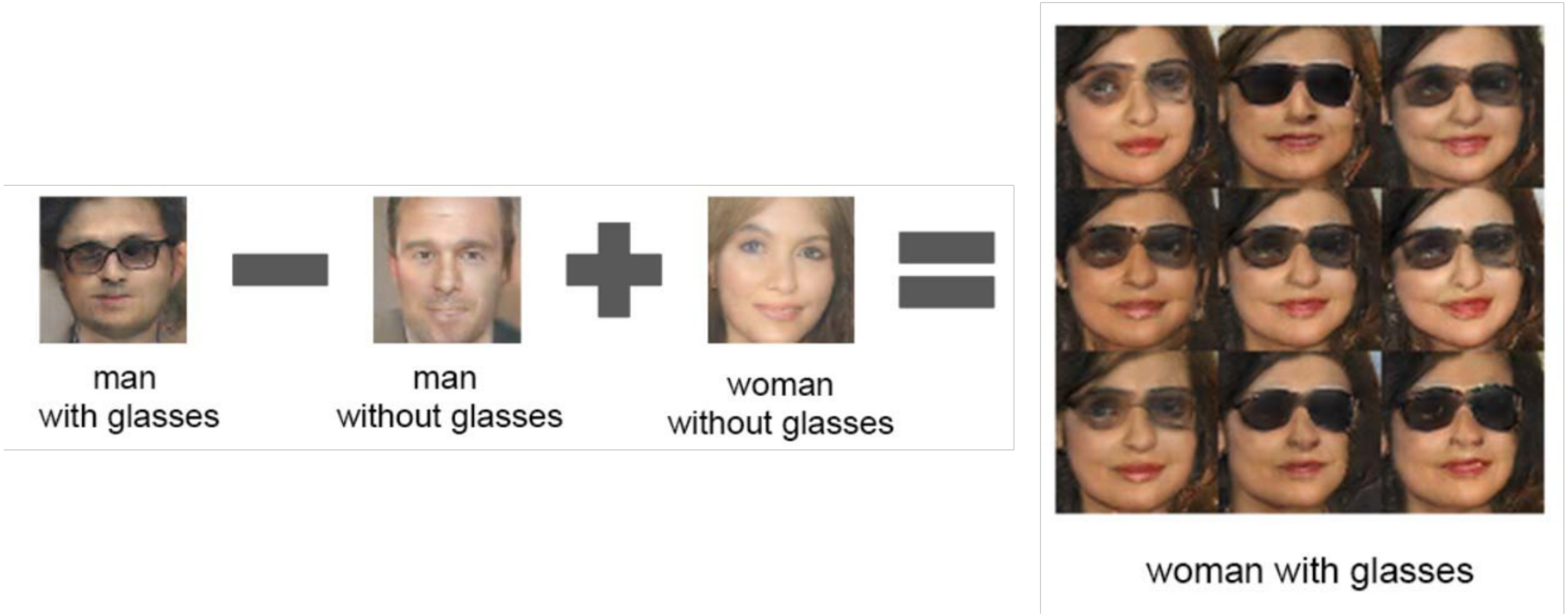


$$\text{loss} = \| x - \hat{x} \|^2 + \mathrm{KL}[\, N(\mu_x, \sigma_x), N(0, I) \,] = \| x - d(z) \|^2 + \mathrm{KL}[\, N(\mu_x, \sigma_x), N(0, I) \,]$$

data manifold for 2-d z



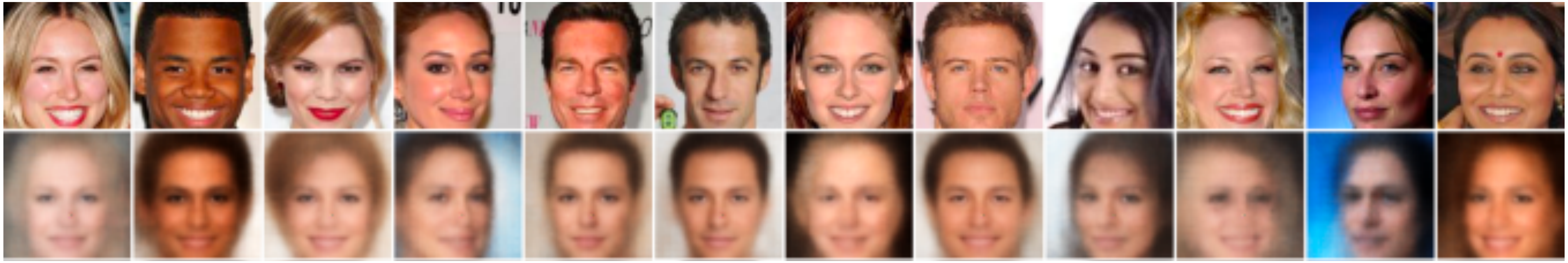



man
with glasses
man
without glasses
woman
without glasses
woman with glasses

Generative Adversarial Nets

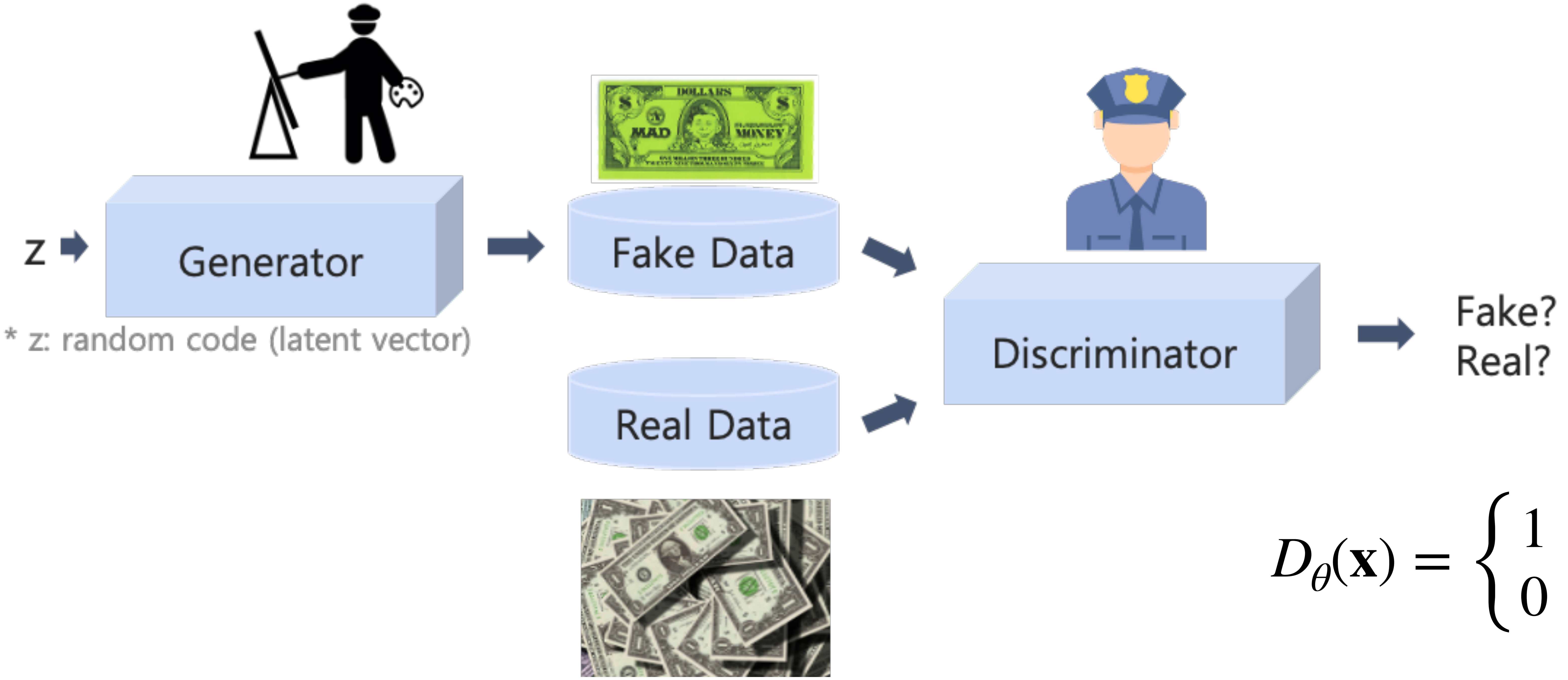
Limitations of VAE

- VAE often produces blurry images
  - Clearly distinguishable from real images...



Generative Adversarial Nets

- **Idea.** View generative process as a two-player game
  - **Generator.** Tries to fool the discriminator
  - **Discriminator.** Tries to distinguish the real / fake images.



$$D_\theta(\mathbf{x}) = \begin{cases} 1 & \cdots \quad \text{if fake} \\ 0 & \cdots \quad \text{if real} \end{cases}$$

Generative Adversarial Nets

- **Training.** Jointly train the Generator and Discriminator
  - **Objective.** Minimax function

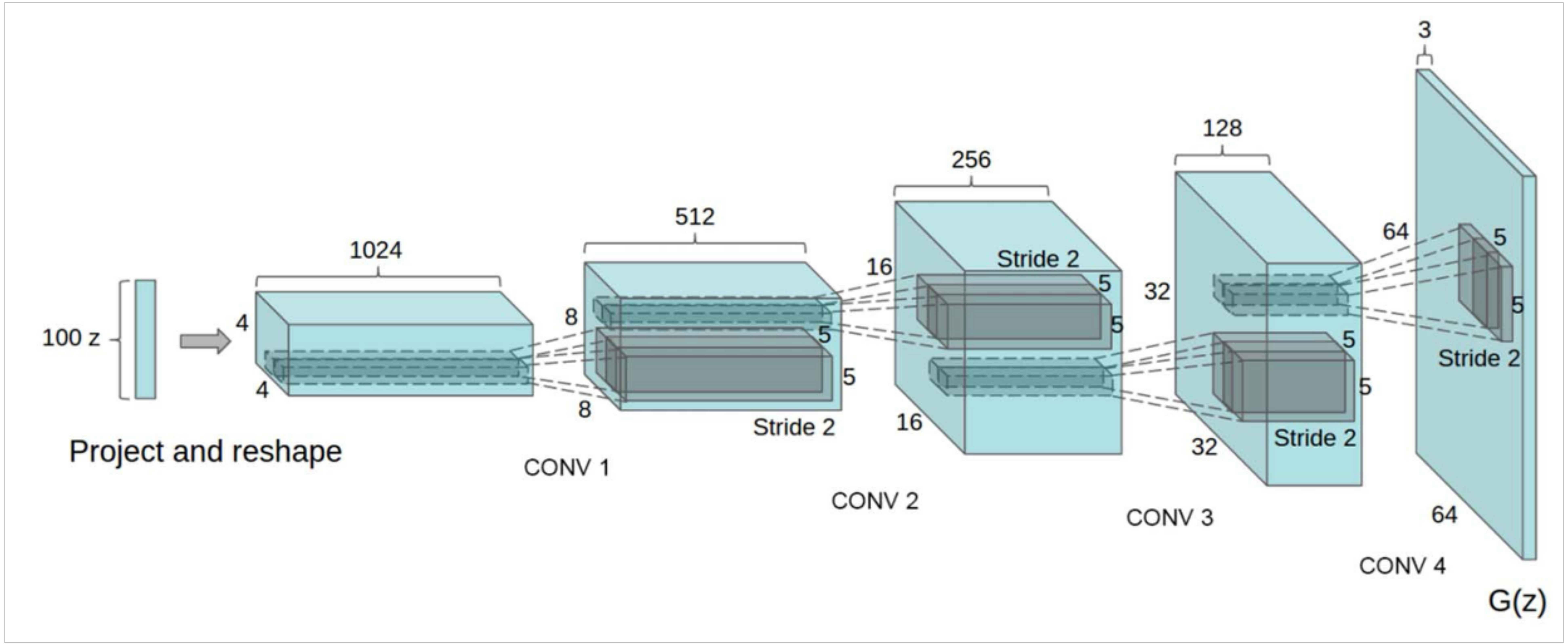
$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \log D_{\theta_d}(\mathbf{x}) + \mathbb{E}_{\mathbf{z}\sim p(z)} \log(1 - D_{\theta_d} \circ G_{\theta_g}(z))\right]$$

Discriminator declares real image to be real

Discriminator declares fake image to be fake

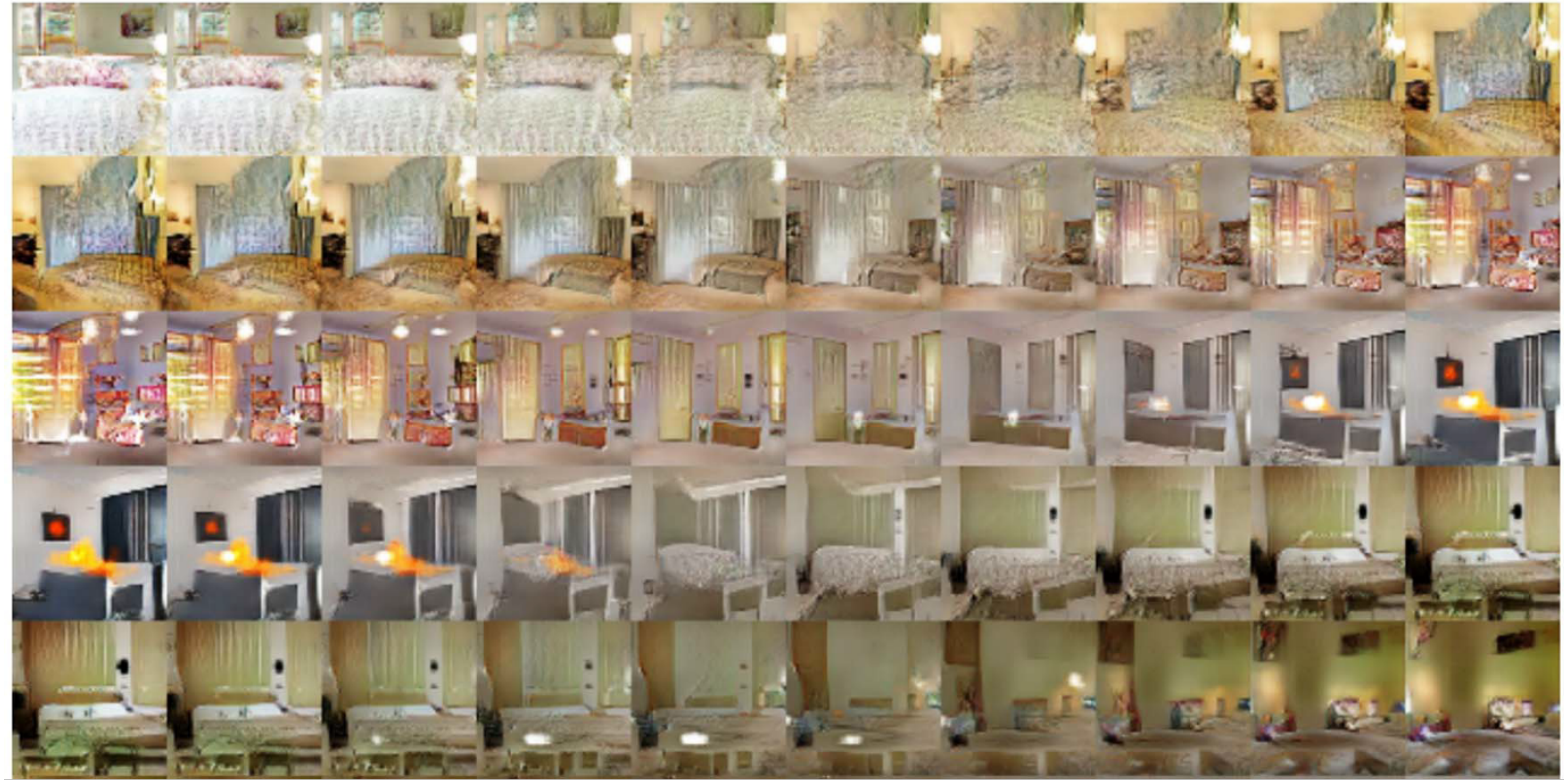
  - Discriminator outputs likelihood of being real

Architecture: Generator



Sharper Images

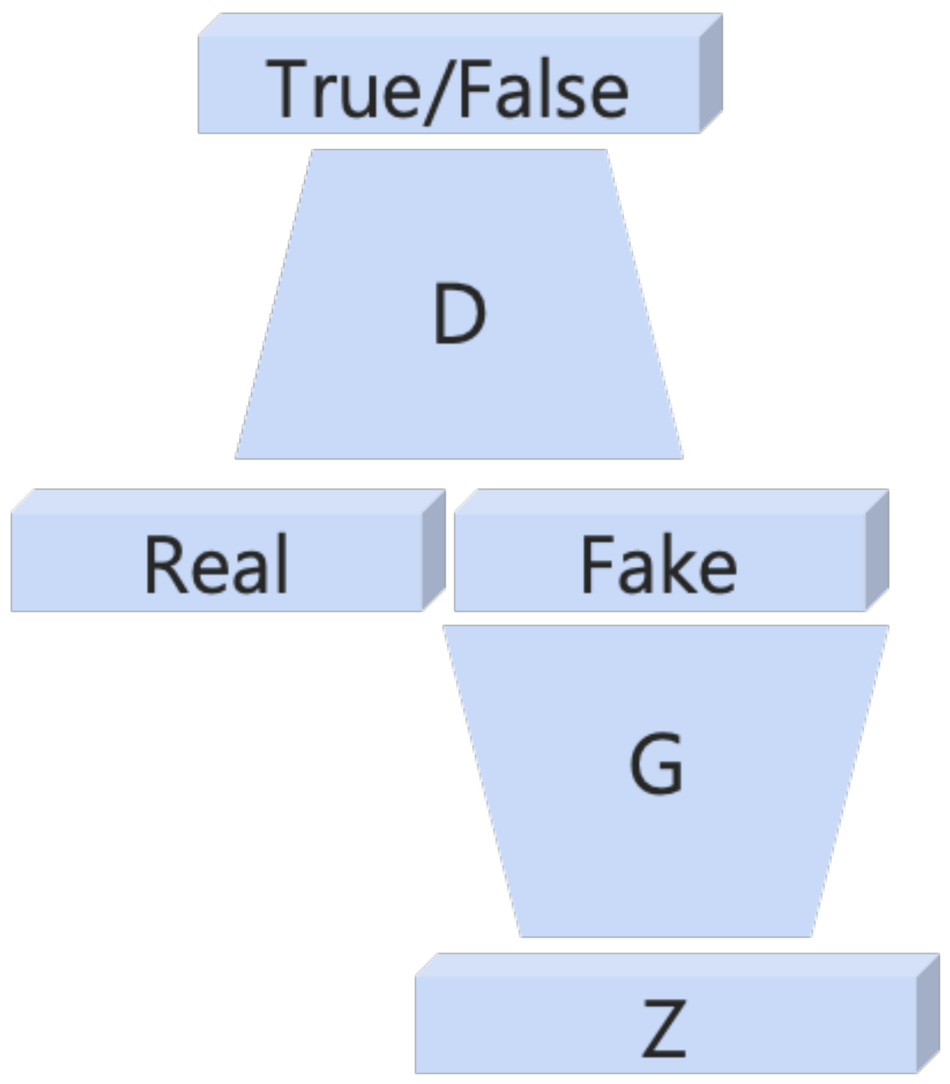
Interpolating between images



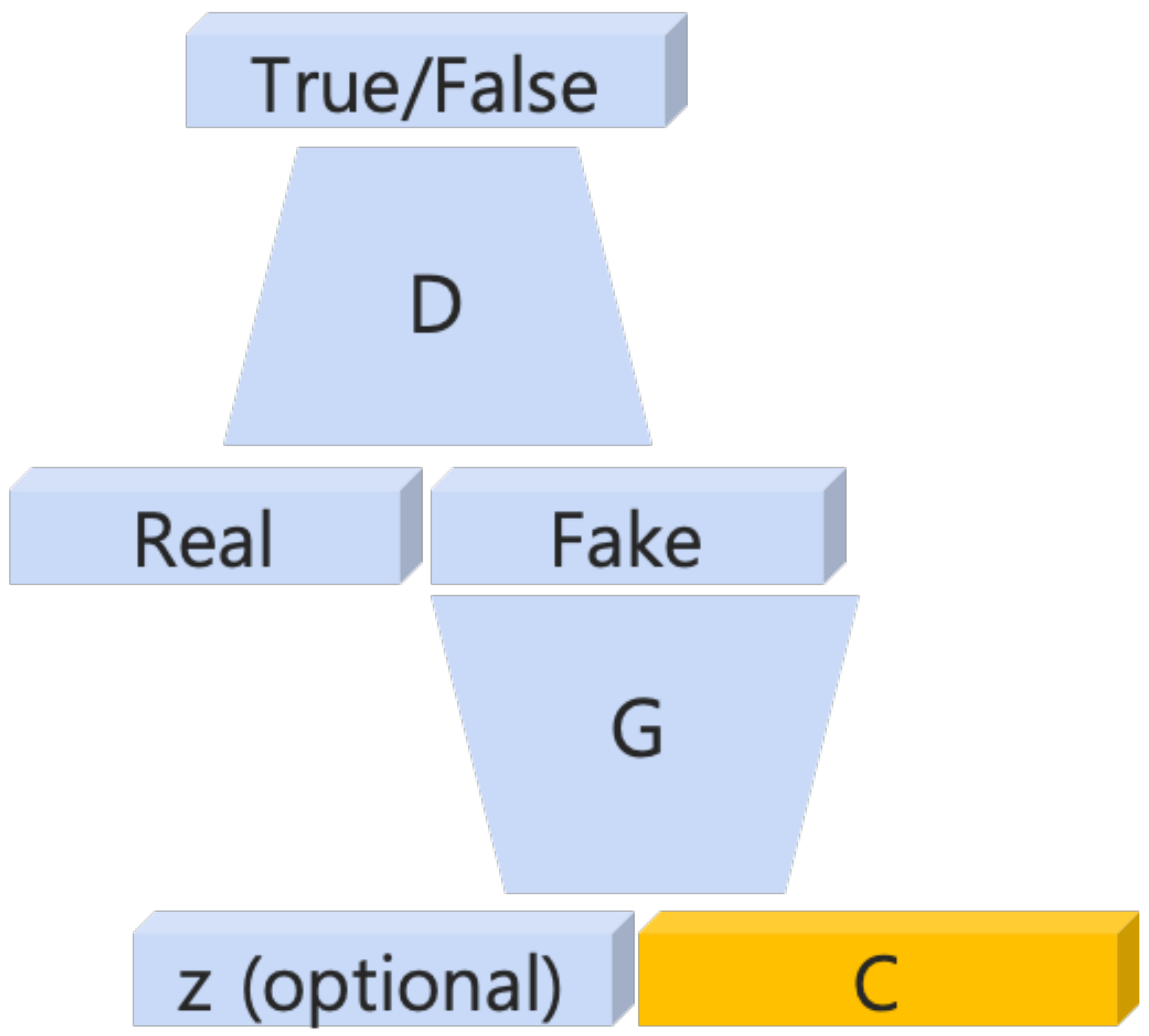
BigGAN

Conditional GAN

- We add class/text information to the latent code, to generate realistic images under specific conditions



GAN

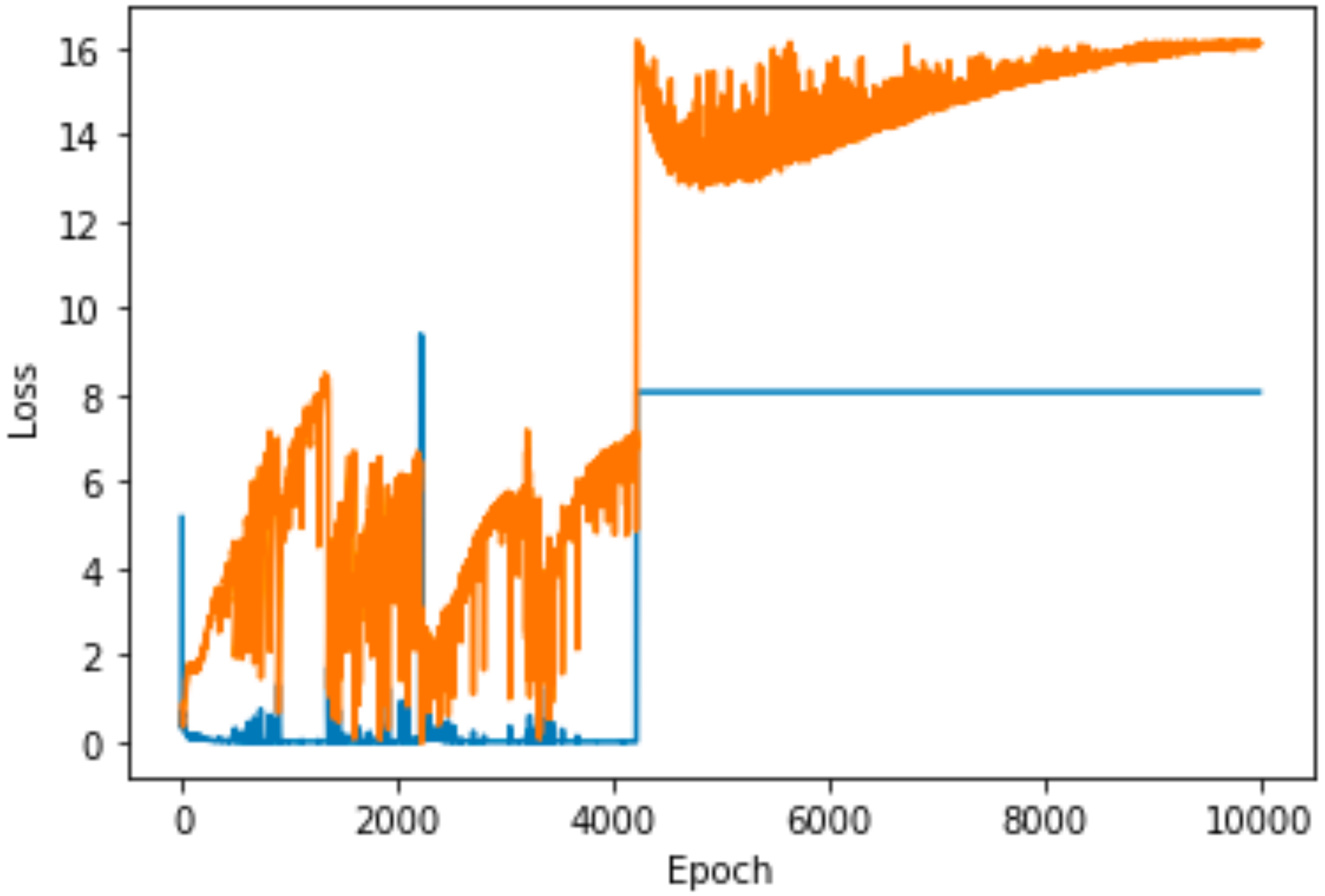


Conditional GAN

Conditional GAN

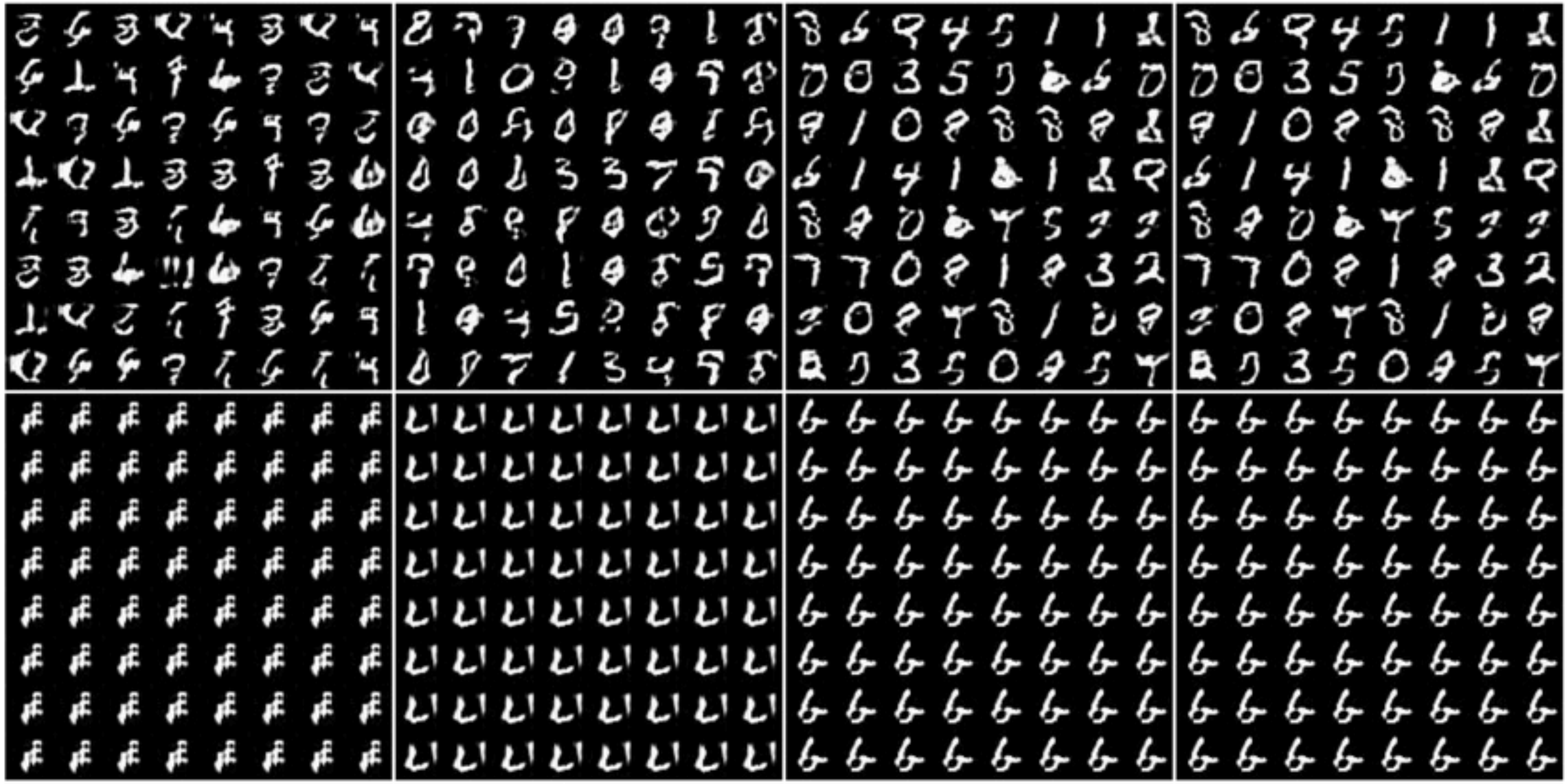
Pitfalls

- Training GANs are known to be very unstable—
  - If discriminator works too well, generator cannot learn
  - If generator works too well, discriminator cannot learn



Pitfalls

- Very easy to resort to not-too-diverse solutions
  (called mode collapse)

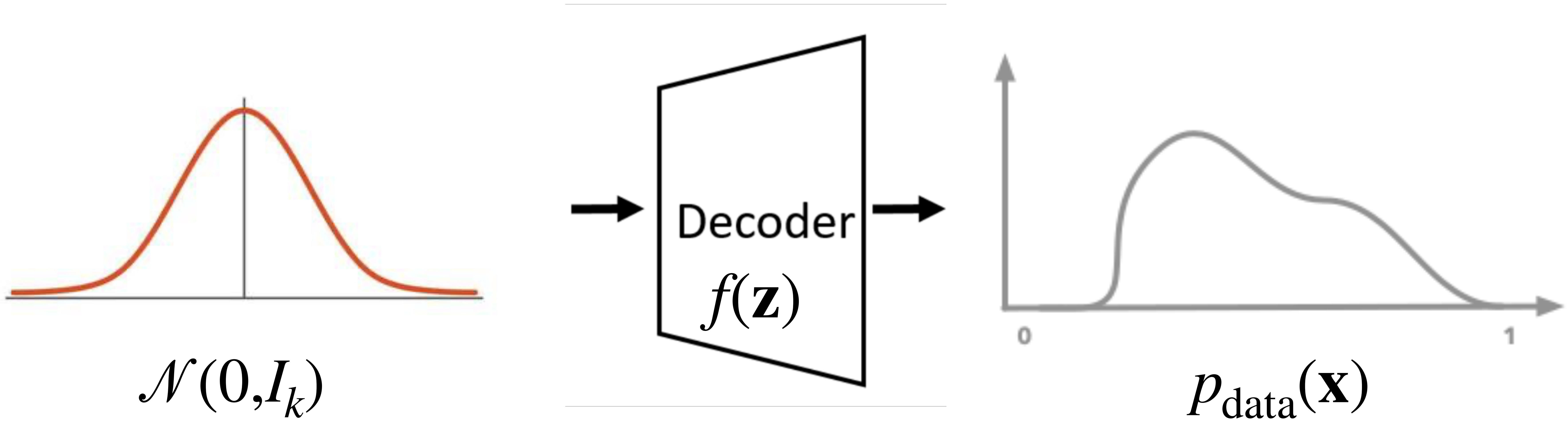


10k steps 20k steps 50K steps 100k steps

Diffusion Models

Motivation

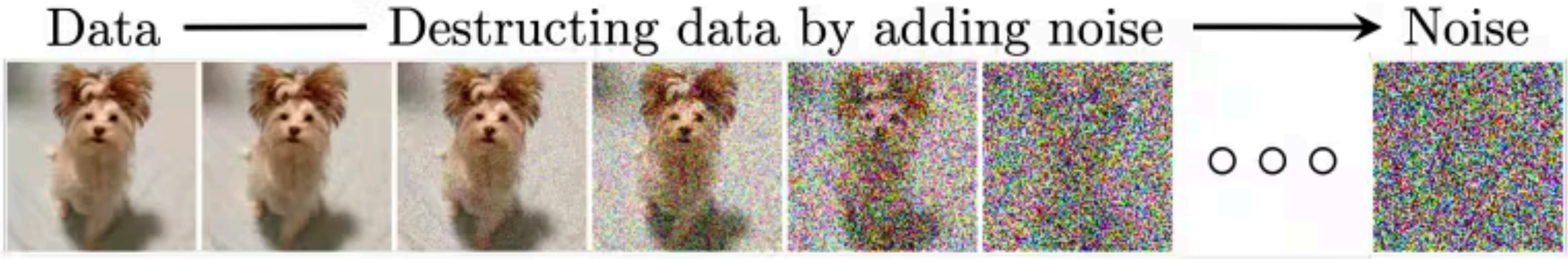
- We have been finding ways to generate $p_{\text{data}}(\mathbf{x})$ from $\mathcal{N}(0, I_k)$



Motivation

- We have been finding ways to generate $p_{\text{data}}(\mathbf{x})$ from $\mathcal{N}(0, I_k)$
- If we wanted to to the ***opposite***, this is quite easy…
  - Repeatedly apply

$$\mathbf{x} \mapsto \sqrt{t}\mathbf{x} + \sqrt{1-t} \cdot \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I_d)$$

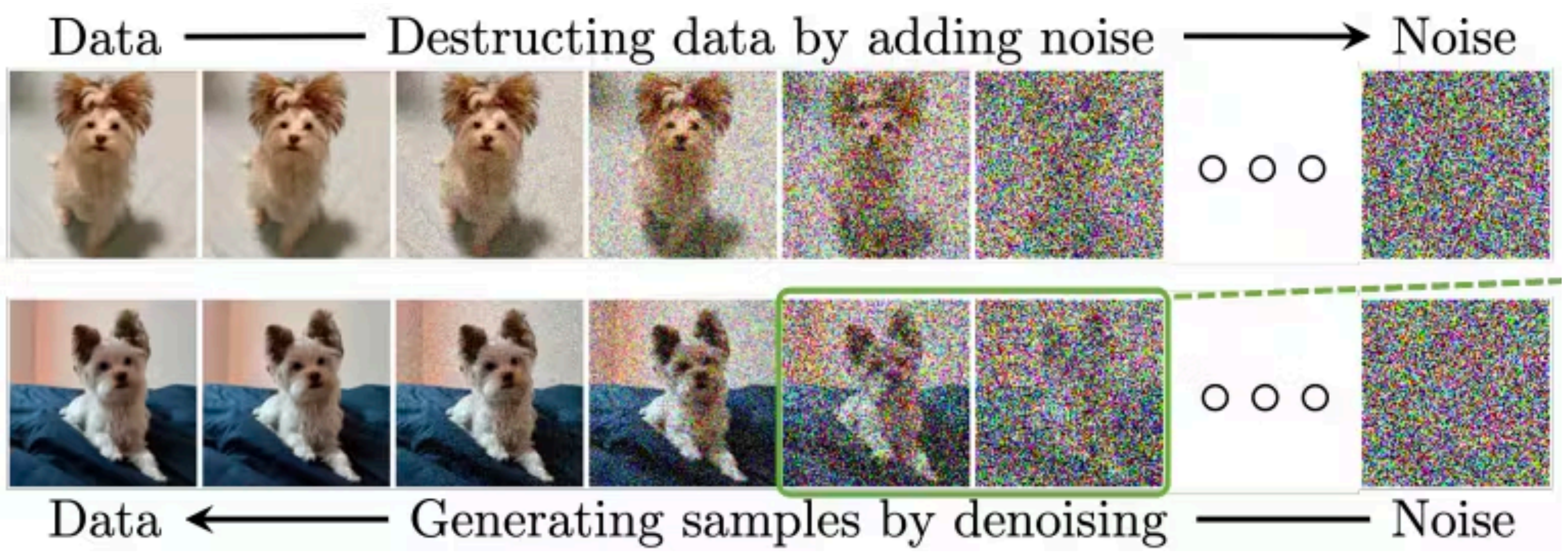


Motivation

- We have been finding ways to generate $p_{\text{data}}(\mathbf{x})$ from $\mathcal{N}(0, I_k)$
- If we wanted to to the ***opposite***, this is quite easy...
    - Repeatedly apply

$$\mathbf{x} \mapsto \sqrt{t}\mathbf{x} + \sqrt{1-t} \cdot \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I_d)$$

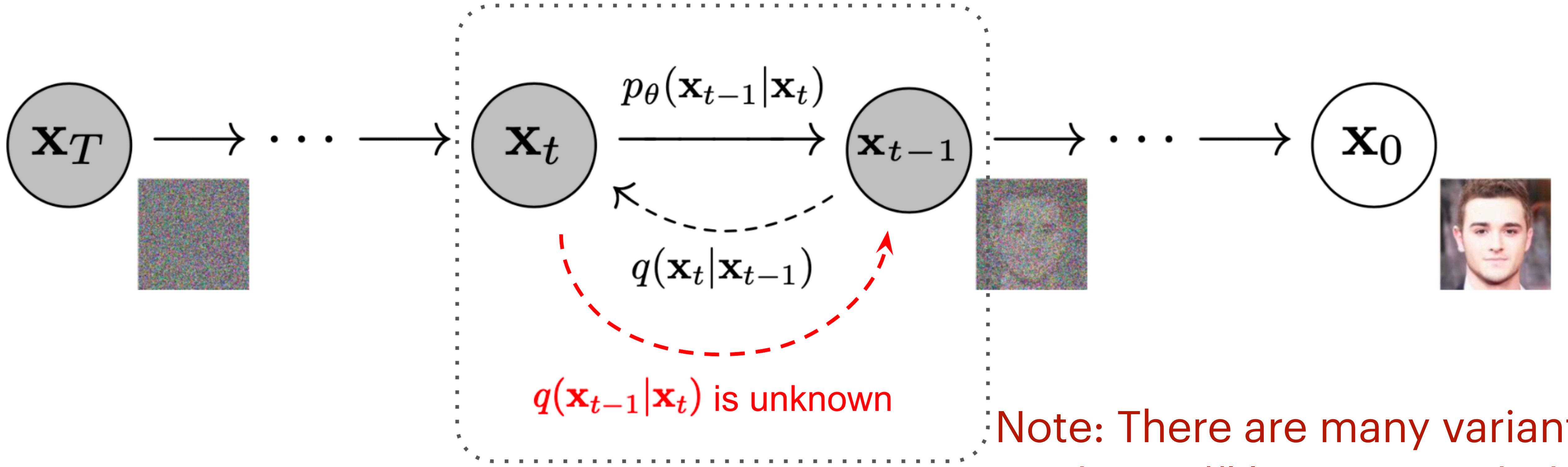
- **Idea.** Why don't we train a function that can invert this process? (Note: we can use the ELBO again)



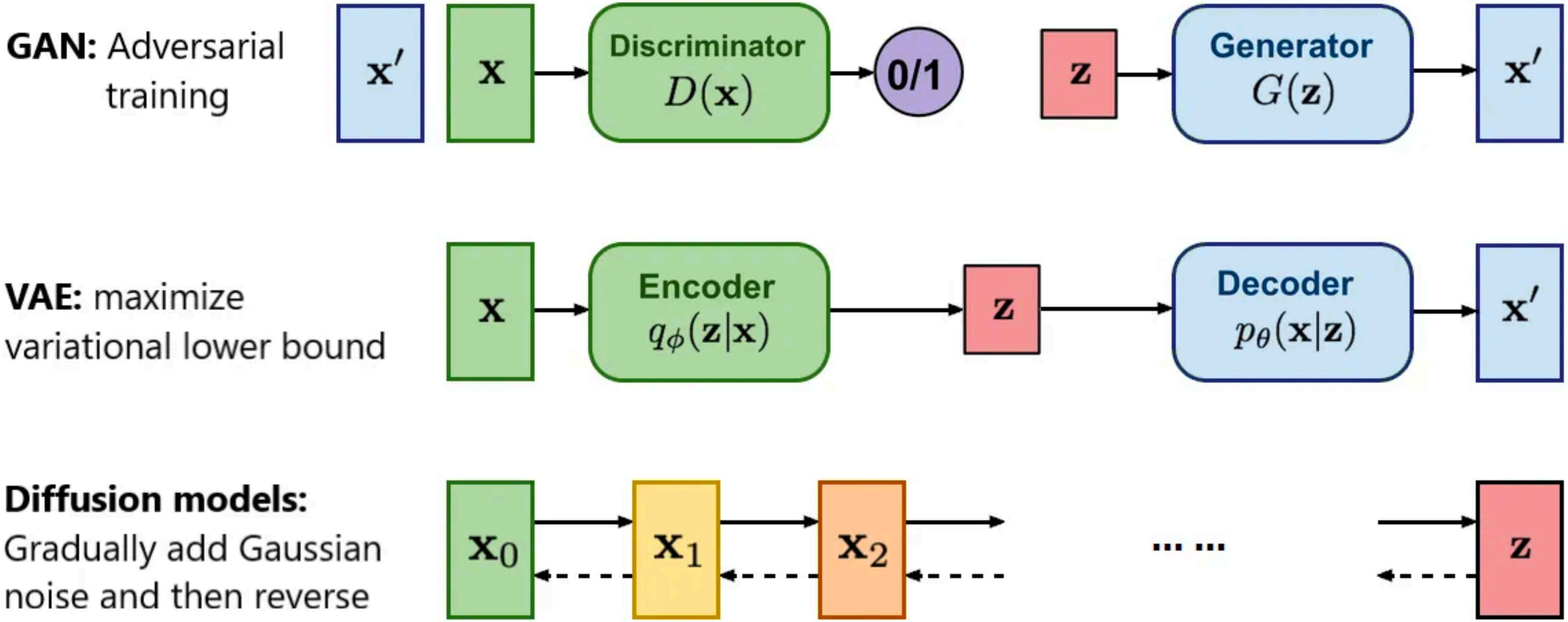
Training

- **Repeat four steps** until convergence.
  - Sample an image $\mathbf{x}_0$ from the dataset.
  - Sample some time interval $t \in \text{Unif}(\{1, \ldots, T\})$
  - Sample a noise $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
  - Train a function to minimize $\left\| \mathbf{x}_0 - f\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon; t\right) \right\|^2$

Use variational lower bound



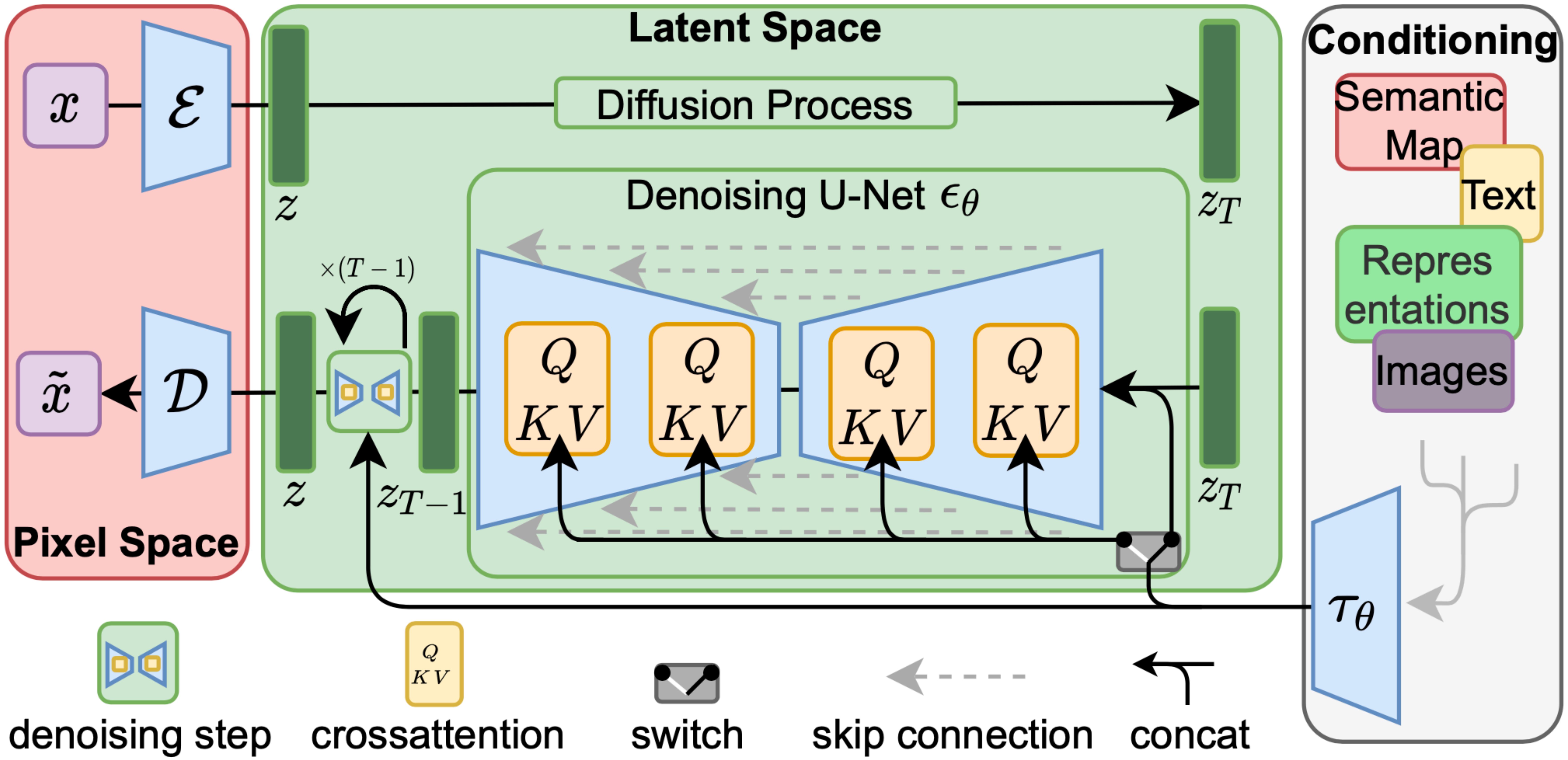
Note: There are many variants, e.g., DDPM, DDIM
see https://lilianweng.github.io/posts/2021-07-11-diffusion-models/



GAN: Adversarial training
x′
x
Discriminator
D(x)
0/1
z
Generator
G(z)
x′
VAE: maximize variational lower bound
x
Encoder
qφ(z|x)
z
Decoder
pθ(x|z)
x′
Diffusion models: Gradually add Gaussian noise and then reverse
x0
x1
x2
z

Latent Diffusion

- We do the diffusion process inside some latent space.



More references

- For simple implementations:
  - https://huggingface.co/blog/annotated-diffusion
- For mathematical details:
  - https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Cheers

- *Next up.* Transformer Basics