Lecture 14

Generative Models: VAE and GAN

Speaker: Kaiming He

Overview

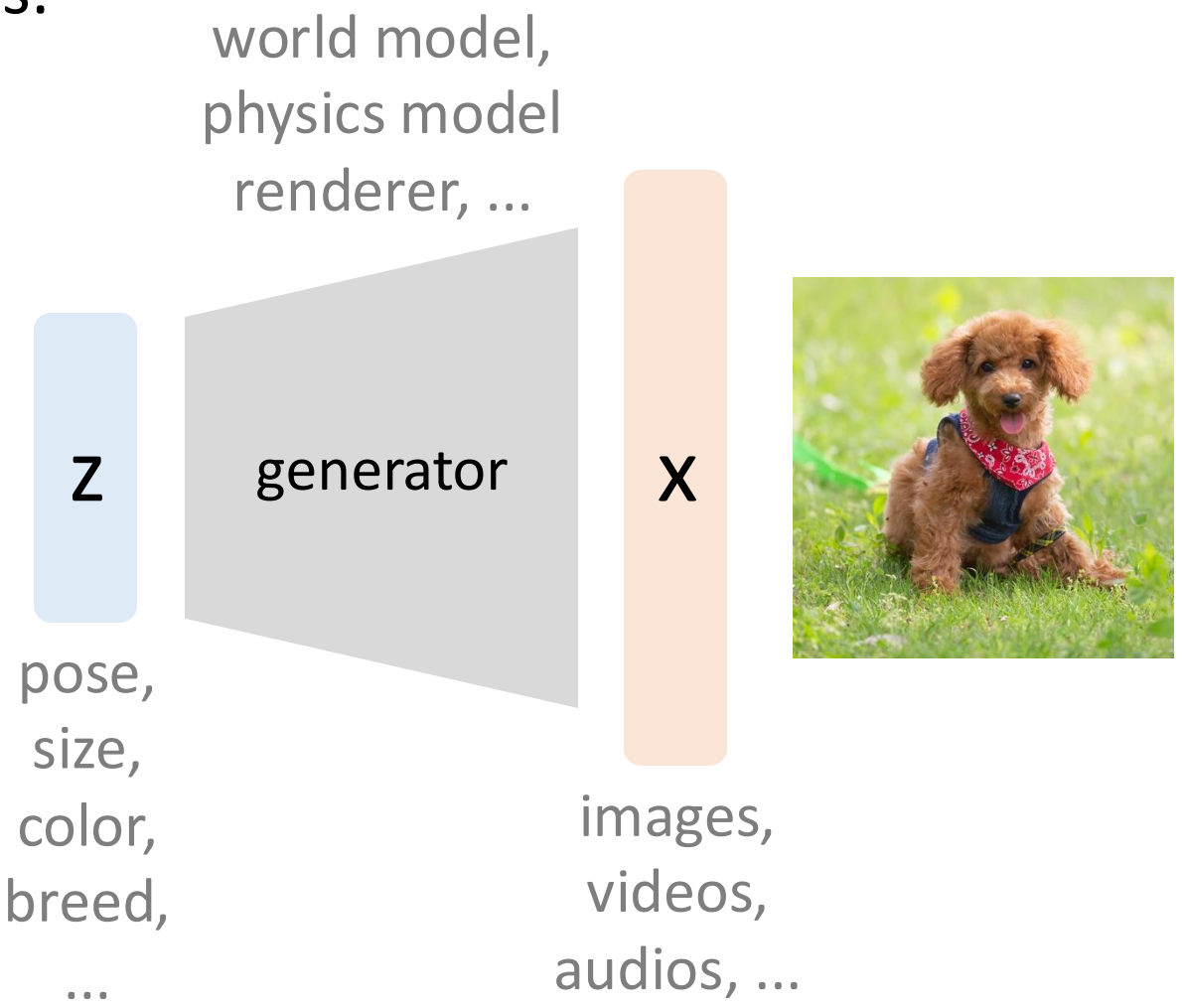
- Latent Variable Models
- Variational Autoencoder (VAE)
- Generative Adversarial Network (GAN)

Latent Variable Models

Latent Variable Models

Assuming a data generation process:

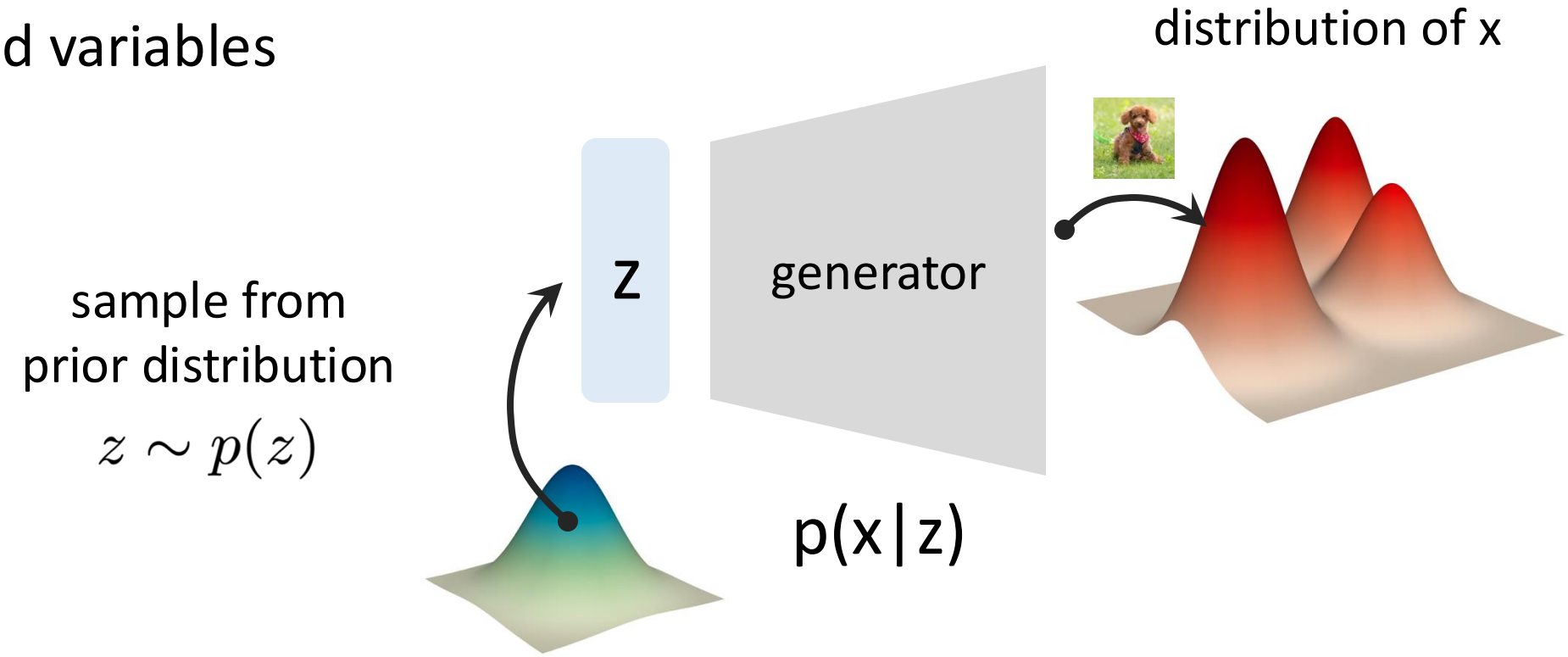
- z - latent variables
- x - observed variables



Latent Variable Models

Assuming a data generation process:

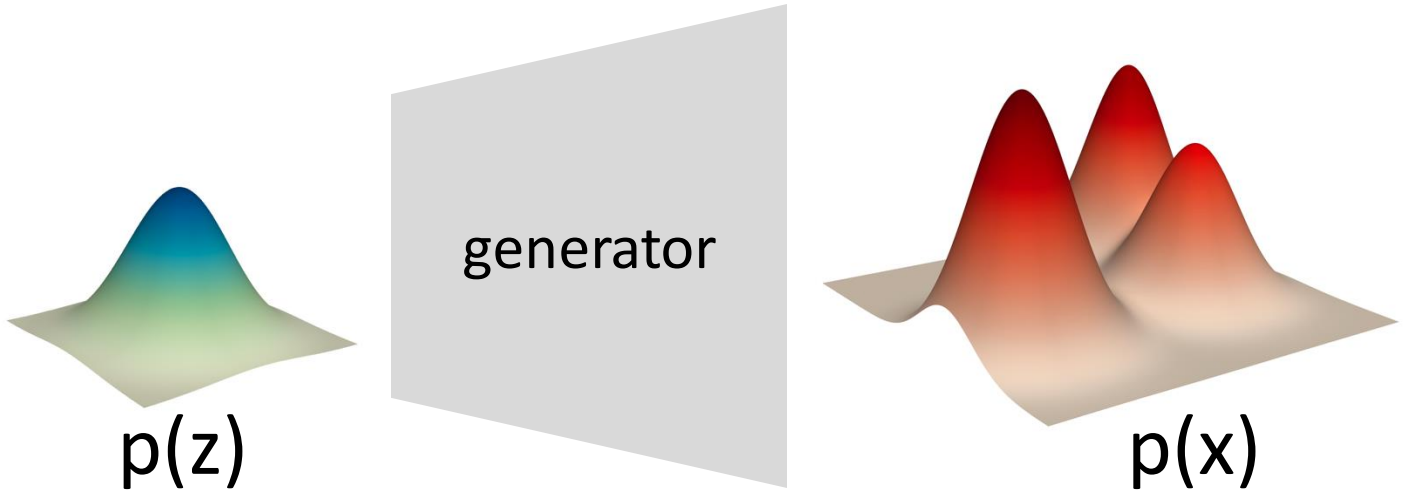
- z - latent variables
- x - observed variables



Latent Variable Models

Assuming a data generation process:

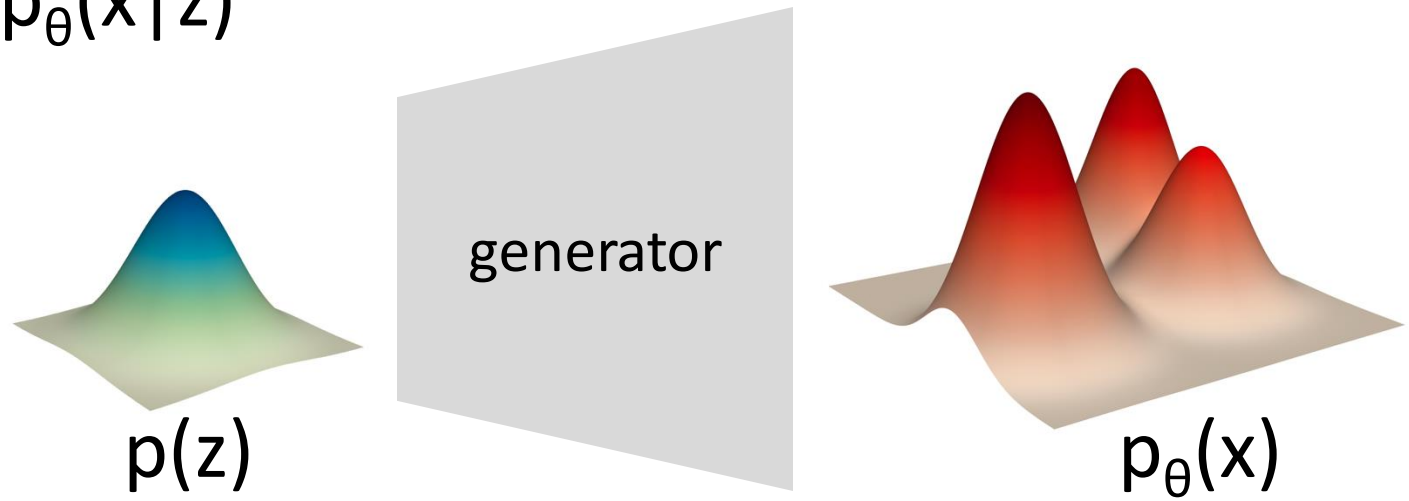
- z - latent variables
- x - observed variables



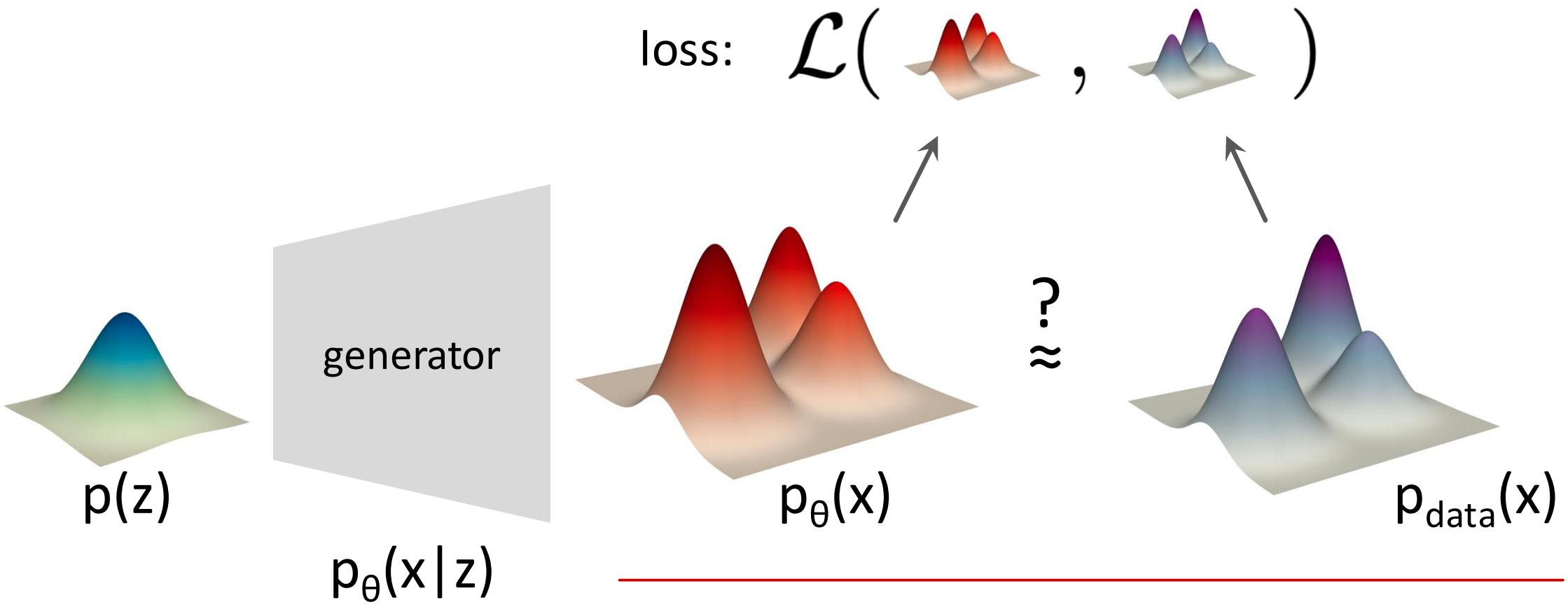
Latent Variable Models

Represent a distribution by a neural network

- θ - learnable parameters
- represent a conditional: $p_\theta(x|z)$



Measuring How Good A Distribution Is

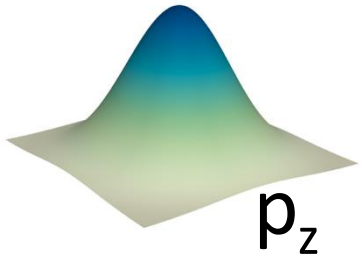


- A core topic in generative modeling: make it **differentiable**, **computable**, and **tractable**

VAE

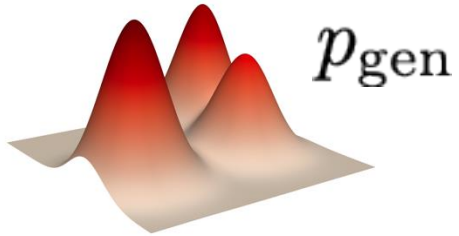
p_{data}

encoder



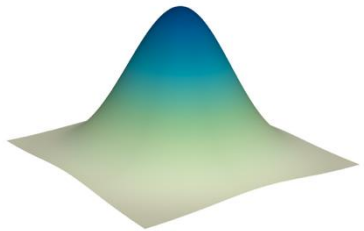
p_z

generator



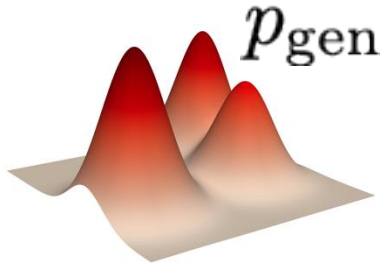
p_{gen}

GAN



p_z

generator



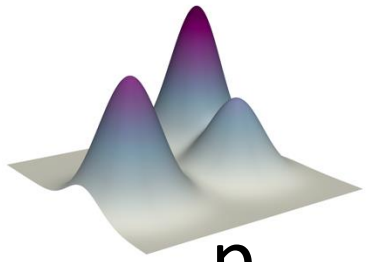
p_{gen}

p_{data}

discriminator

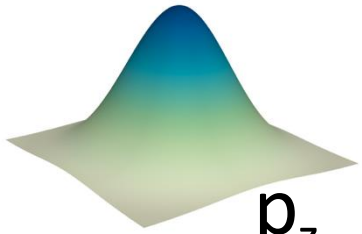
or ?

VAE



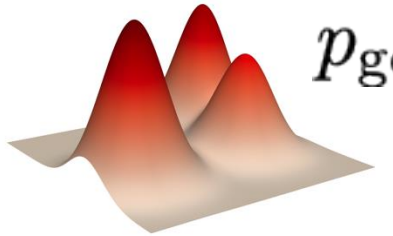
p_{data}

encoder



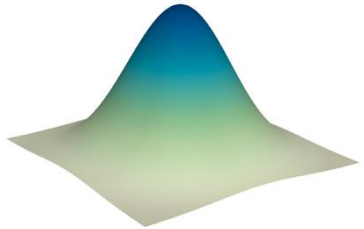
p_z

generator



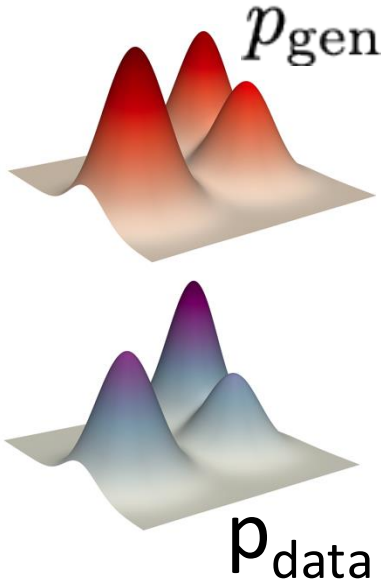
p_{gen}

GAN



p_z

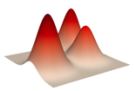
generator



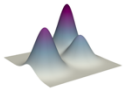
p_{gen}

p_{data}

discriminator



or

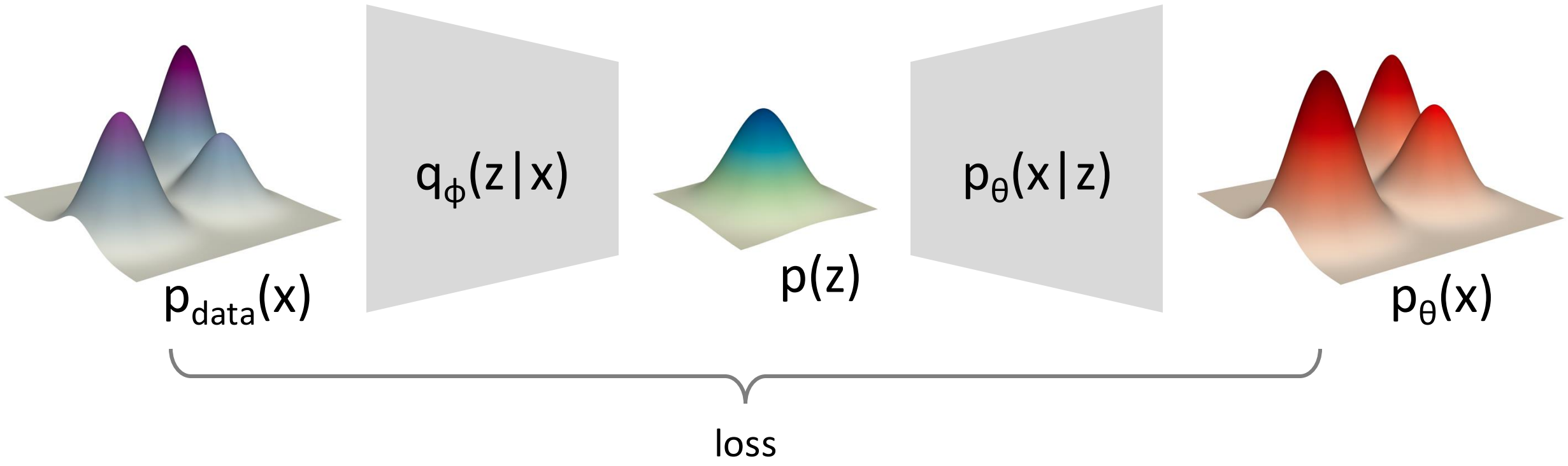


?

Variational Autoencoder (VAE)

Variational Autoencoder: Overview

- **Encoder**: data to latent, parameterized by ϕ
- **Decoder**: latent to data, parameterized by θ



Maximum Likelihood Estimation (MLE)

Minimize KL-Divergence:

$$\min_{\theta} \mathcal{D}_{\mathrm{KL}}(p_{data} \,||\, p_\theta)$$

⇒ **Maximize** likelihood:

$$\max_{\theta} \mathbb{E}_{x \sim p_{data}} \log p_\theta(x)$$

tl; dr

$$\begin{aligned}
& \arg\min_{\theta} \mathcal{D}_{\mathrm{KL}}(p_{data} \,||\, p_\theta) \\
= \; & \arg\min_{\theta} \sum_x p_{data}(x) \log \frac{p_{data}(x)}{p_\theta(x)} \\
= \; & \arg\min_{\theta} \sum_x -p_{data}(x) \log p_\theta(x) + const \\
= \; & \arg\max_{\theta} \sum_x p_{data}(x) \log p_\theta(x) \\
= \; & \arg\max_{\theta} \mathbb{E}_{x \sim p_{data}} \log p_\theta(x)
\end{aligned}$$

KL divergence와 likelihood의 관계

식을 분리하면 다음과 같습니다.

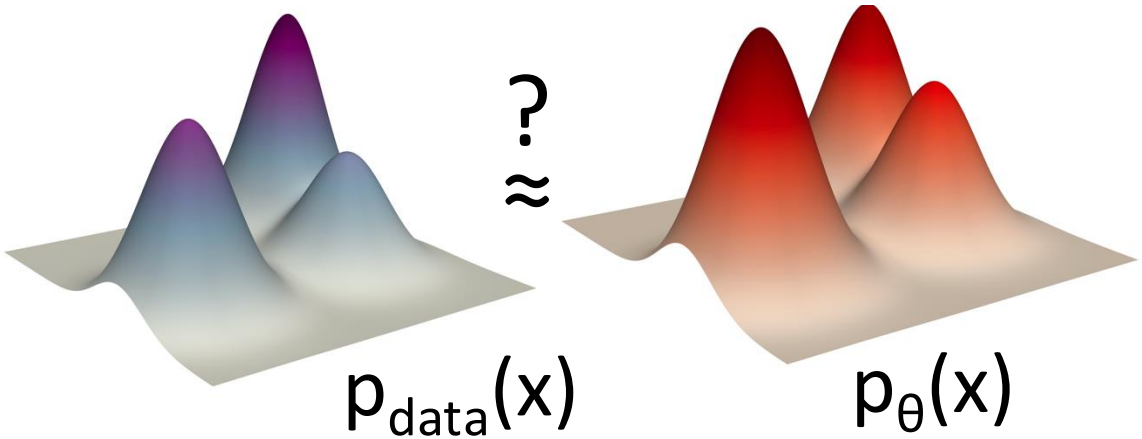
$$D_{\mathrm{KL}}(p_{\mathrm{data}} \,\|\, p_\theta) = \mathbb{E}_{p_{\mathrm{data}}}[\log p_{\mathrm{data}}(x)] - \mathbb{E}_{p_{\mathrm{data}}}[\log p_\theta(x)]$$

첫 번째 항은 θ와 관계없는 상수입니다. 따라서 KL divergence를 최소화하는 것은 다음 negative log-likelihood를 최소화하는 것과 같은 최적해를 가집니다.

$$-\mathbb{E}_{x \sim p_{\mathrm{data}}}[\log p_\theta(x)]$$

실제로는 p_{data}의 수식을 모르므로 학습 데이터로 이를 근사합니다.

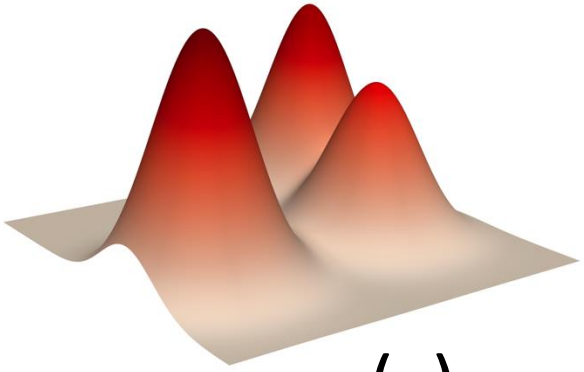
$$-\frac{1}{N}\sum_{i=1}^{N} \log p_\theta(x_i)$$



$p_{data}(x)$ $p_\theta(x)$

Variational Autoencoder (VAE)

We want to maximize $\mathbb{E}_{x \sim p_{data}} \log p_\theta(x)$



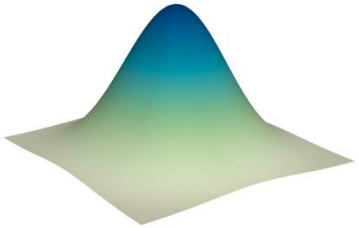
$p_\theta(x)$

Variational Autoencoder (VAE)

We want to maximize $\mathbb{E}_{x \sim p_{data}} \log p_\theta(x)$

with $p_\theta(x)$ represented as:

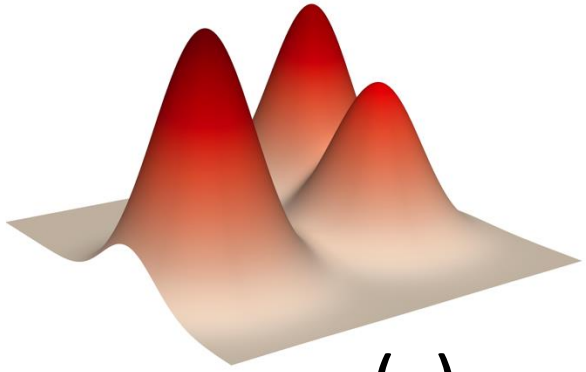
$$p_\theta(x) = \int_z p_\theta(x|z)p(z)dz$$



p(z)

generator

$p_\theta(x|z)$



$p_\theta(x)$

Variational Autoencoder (VAE)

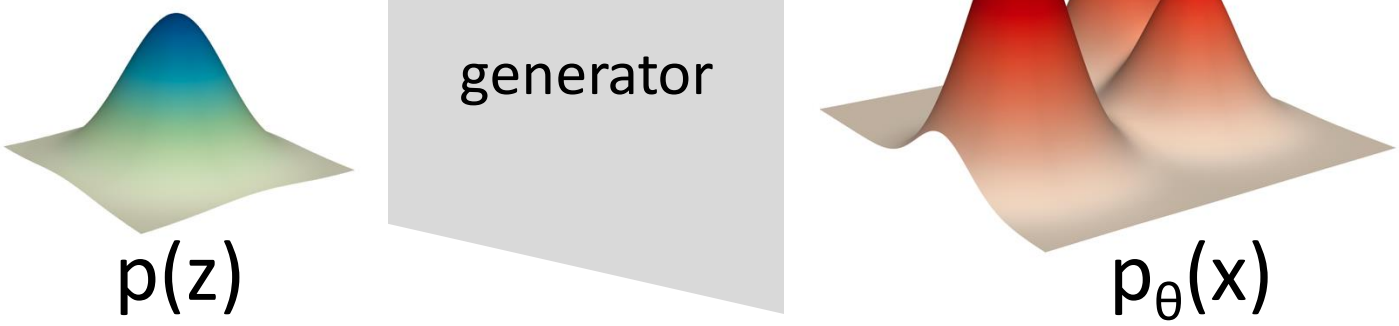
We want to maximize $\mathbb{E}_{x \sim p_{data}} \log p_\theta(x)$ (likelihood)

with $p_\theta(x)$ represented as:

$$p_\theta(x) = \int_z p_\theta(x|z) p(z) dz$$

Two sets of unknowns:

- We need to optimize for θ
- We can't control "true" p(z)



→ we observe x, not z. → $p_\theta(z|x)$ 계산 어려움.

Idea: introduce a "controllable" distribution $q_\phi(z)$

↳ encoder. real → latent.

Evidence Lower Bound (ELBO)

- Rewrite log-likelihood using latent z

$$
\begin{aligned}
& \log p_\theta(x) \\
= & \int_z q(z) \log p_\theta(x) dz \\
= & \int_z q(z) \log \Big(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \Big) dz \\
= & \int_z q(z) \log \Big(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \frac{q(z)}{q(z)} \Big) dz \\
= & \int_z q(z) \Big(\log p_\theta(x|z) + \log \frac{p_\theta(z)}{q(z)} + \log \frac{q(z)}{p_\theta(z|x)} \Big) dz \\
= & \mathbb{E}_{z \sim q(z)} \Big[\log p_\theta(x|z) \Big] - \mathcal{D}_{\mathrm{KL}} \Big(q(z) || p_\theta(z) \Big) + \mathcal{D}_{\mathrm{KL}} \Big(q(z) || p_\theta(z|x) \Big)
\end{aligned}
$$

- valid for any distribution q(z)
- Bayes' rule

Evidence Lower Bound (ELBO)

- Rewrite log-likelihood using latent z

$$\log p_\theta(x)$$

$$= \int_z q(z) \log p_\theta(x) dz$$

- valid for any distribution q(z)

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \right) dz$$

- Bayes' rule

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \frac{q(z)}{q(z)} \right) dz$$

- just algebra

$$= \int_z q(z) \left(\log p_\theta(x|z) + \log \frac{p_\theta(z)}{q(z)} + \log \frac{q(z)}{p_\theta(z|x)} \right) dz$$

- just algebra

$$= \mathbb{E}_{z \sim q(z)} \left[\log p_\theta(x|z) \right] - \mathcal{D}_{\mathrm{KL}} \left(q(z) || p_\theta(z) \right) + \mathcal{D}_{\mathrm{KL}} \left(q(z) || p_\theta(z|x) \right)$$

Evidence Lower Bound (ELBO)

- Rewrite log-likelihood using latent z

$$\log p_\theta(x)$$

$$= \int_z q(z) \log p_\theta(x) dz$$

- valid for any distribution q(z)

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \right) dz$$

- Bayes' rule

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \frac{q(z)}{q(z)} \right) dz$$

- just algebra

$$= \int_z q(z) \left(\log p_\theta(x|z) + \log \frac{p_\theta(z)}{q(z)} + \log \frac{q(z)}{p_\theta(z|x)} \right) dz$$

- just algebra

$$= \mathbb{E}_{z \sim q(z)} \left[\log p_\theta(x|z) \right] - \mathcal{D}_{\mathrm{KL}} \left(q(z) || p_\theta(z) \right) + \mathcal{D}_{\mathrm{KL}} \left(q(z) || p_\theta(z|x) \right)$$

Evidence Lower Bound (ELBO)

- Rewrite log-likelihood using latent z

$$\log p_\theta(x)$$

$$= \int_z q(z) \log p_\theta(x) dz$$

- valid for any distribution q(z)

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \right) dz$$

- Bayes' rule

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \frac{q(z)}{q(z)} \right) dz$$

- just algebra

$$= \int_z q(z) \left(\log p_\theta(x|z) + \log \frac{p_\theta(z)}{q(z)} + \log \frac{q(z)}{p_\theta(z|x)} \right) dz$$

- just algebra

$$= \mathbb{E}_{z \sim q(z)} \left[\log p_\theta(x|z) \right] - \mathcal{D}_{\mathrm{KL}} \left(q(z) || p_\theta(z) \right) + \mathcal{D}_{\mathrm{KL}} \left(q(z) || p_\theta(z|x) \right)$$

Evidence Lower Bound (ELBO)

- Rewrite log-likelihood using latent z

$$\log p_\theta(x)$$

$$= \int_z q(z) \log p_\theta(x) dz$$

- valid for any distribution q(z)

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \right) dz$$

- Bayes' rule

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \frac{q(z)}{q(z)} \right) dz$$

- just algebra

$$= \int_z q(z) \left(\log p_\theta(x|z) + \log \frac{p_\theta(z)}{q(z)} + \log \frac{q(z)}{p_\theta(z|x)} \right) dz$$

- just algebra

$$= \mathbb{E}_{z \sim q(z)} \left[\log p_\theta(x|z) \right] - \mathcal{D}_{\mathrm{KL}} \left(q(z) || p_\theta(z) \right) + \mathcal{D}_{\mathrm{KL}} \left(q(z) || p_\theta(z|x) \right)$$

Evidence Lower Bound (ELBO)

- Rewrite log-likelihood using latent z

$$\log p_\theta(x) \qquad \int_z q(z) = 1$$

$$= \int_z q(z) \log p_\theta(x) dz$$

- valid for any distribution q(z)

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \right) dz$$

- Bayes' rule

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \frac{q(z)}{q(z)} \right) dz$$

- just algebra

$$= \int_z q(z) \left(\log p_\theta(x|z) + \log \frac{p_\theta(z)}{q(z)} + \log \frac{q(z)}{p_\theta(z|x)} \right) dz$$

- just algebra

$$= \mathbb{E}_{z \sim q(z)} \left[\log p_\theta(x|z) \right] - \mathcal{D}_{\mathrm{KL}} \left(q(z) || p_\theta(z) \right) + \mathcal{D}_{\mathrm{KL}} \left(q(z) || p_\theta(z|x) \right)$$

Evidence Lower Bound (ELBO)

- Rewrite log-likelihood using latent z

intractable $\log p_\theta(x)$

$$= \int_z q(z) \log p_\theta(x) dz$$

- valid for any distribution q(z)

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \right) dz$$

- Bayes' rule

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \frac{q(z)}{q(z)} \right) dz$$

$$= \int_z q(z) \left(\log p_\theta(x|z) + \log \frac{p_\theta(z)}{q(z)} + \log \frac{q(z)}{p_\theta(z|x)} \right) dz$$

$$= \mathbb{E}_{z \sim q(z)} \left[\log p_\theta(x|z) \right] - \mathcal{D}_{\mathrm{KL}} \left(q(z) || p_\theta(z) \right) + \mathcal{D}_{\mathrm{KL}} \left(q(z) || p_\theta(z|x) \right)$$

tractable tractable intractable

Evidence Lower Bound (ELBO)

$z \sim p_\theta(z)$

intractable $\log p_\theta(x) - \mathcal{D}_{\mathrm{KL}}\left(q(z)||p_\theta(z|x)\right)$ intractable

$= \mathbb{E}_{z\sim q(z)}\left[\log p_\theta(x|z)\right] - \mathcal{D}_{\mathrm{KL}}\left(q(z)||p_\theta(z)\right)$

tractable tractable

z의 distribution은 Gaussian 등으로 가정하고 random sampling

→ 특정 z, 샘플 분포는 random sampling으로 비산가능.

but $\int z$ 어려움.

> 모든 z를 적분해야 하면 intractable, 특정 z만 평가하거나 단순한 분포끼리 비교하면 tractable입니다.

항	판단	결정적인 이유
$\log p_\theta(x)$	Intractable	$p_\theta(x) = \int p_\theta(x \mid z)p(z)dz$. 고차원인 모든 z를 적분해야 함
$\mathbb{E}_{q(z)}[\log p_\theta(x \mid z)]$	Tractable	계산 가능한 q에서 z 몇 개를 뽑고, decoder로 $p_\theta(x \mid z)$를 평가하면 됨
$D_{\mathrm{KL}}(q(z)\|\|p(z))$	Tractable	q와 $p(z)$를 Gaussian으로 설계하여 KL의 계산식이 존재함
$D_{\mathrm{KL}}(q(z)\|\|p_\theta(z \mid x))$	Intractable	$p_\theta(z \mid x)$를 구하려면 정규화 항 $p_\theta(x) = \int p_\theta(x \mid z)p(z)dz$가 필요함

한 문장으로 정리하면:

> $q(z)$, $p(z)$, $p_\theta(x \mid z)$는 우리가 계산하기 쉽게 설계했지만, $p_\theta(x)$와 $p_\theta(z \mid x)$는 모든 latent 가능성을 종합해야 하므로 계산하기 어렵습니다.

Evidence Lower Bound (ELBO)

intractable $\log p_\theta(x) - \mathcal{D}_{\mathrm{KL}}\left(q(z)||p_\theta(z|x)\right)$ intractable

$$= \underbrace{\mathbb{E}_{z\sim q(z)}\left[\log p_\theta(x|z)\right] - \mathcal{D}_{\mathrm{KL}}\left(q(z)||p_\theta(z)\right)}_{\text{ELBO}}$$

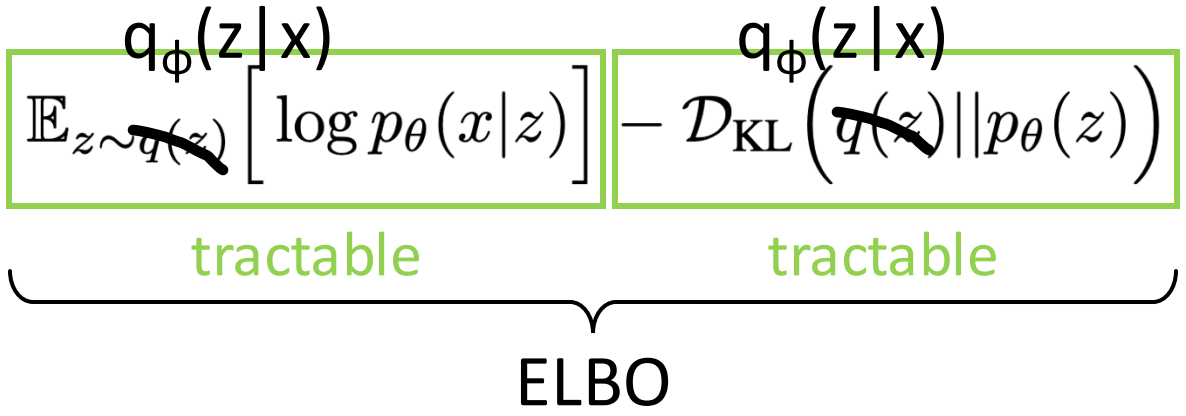
tractable tractable

ELBO:

- It's lower bound of $\log p_\theta(x)$

- This formulation holds for any distribution q(z)

Evidence Lower Bound (ELBO)



ELBO:

- It's lower bound of $\log p_\theta(x)$
- This formulation holds for <u>any</u> distribution q(z) , so ...
- <u>we can parameterize q(z) by $q_\phi(z|x)$</u>

⇓ encoder (data → latent)

Evidence Lower Bound (ELBO)

$q_\phi(z|x)$ $q_\phi(z|x)$ $p(z)$

$$\mathbb{E}_{z\sim q(z)}\left[\log p_\theta(x|z)\right] - \mathcal{D}_{\mathrm{KL}}\left(q(z)||p_\theta(z)\right)$$

tractable tractable

ELBO

latent ~ q 간 general 한 식.

*Note: ELBO is not specific to VAE.

any q(z)라는 식으로부터

q(z) → $q_\phi(z|x)$로 대치 = VAE에서의 ELBO.

ELBO:

- It's lower bound of $\log p_\theta(x)$
- This formulation holds for any distribution q(z) , so ...
- we can parameterize q(z) by $q_\phi(z|x)$
- and let $p_\theta(z)$ be a fixed, known prior p(z) (e.g., Gaussian)

Variational Autoencoder

Maximize ELBO ⇒ minimize:

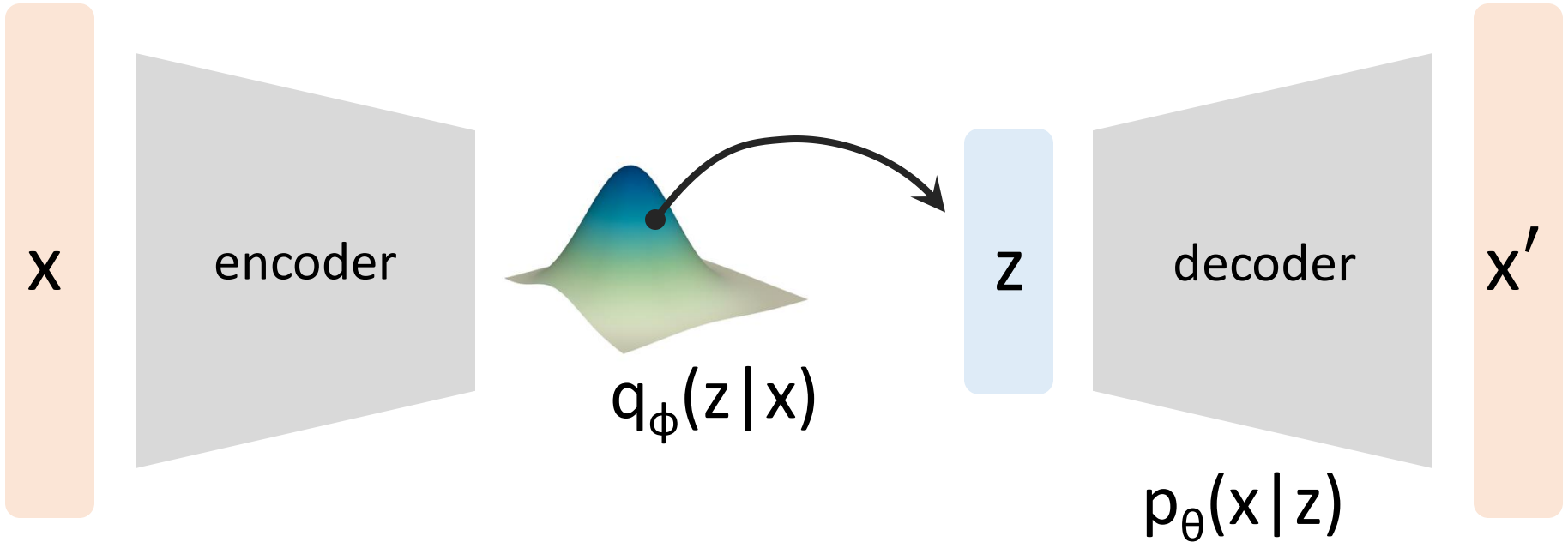
$$\mathcal{L}_{\theta,\phi}(x) = -\mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big] + \mathcal{D}_{\mathrm{KL}}\Big(q_\phi(z|x)||p(z)\Big)$$

(-ELBO)

Variational Autoencoder

Maximize ELBO ⇒ minimize:

$$\mathcal{L}_{\theta,\phi}(x) = -\mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big] + \mathcal{D}_{\mathrm{KL}}\Big(q_\phi(z|x)||p(z)\Big)$$



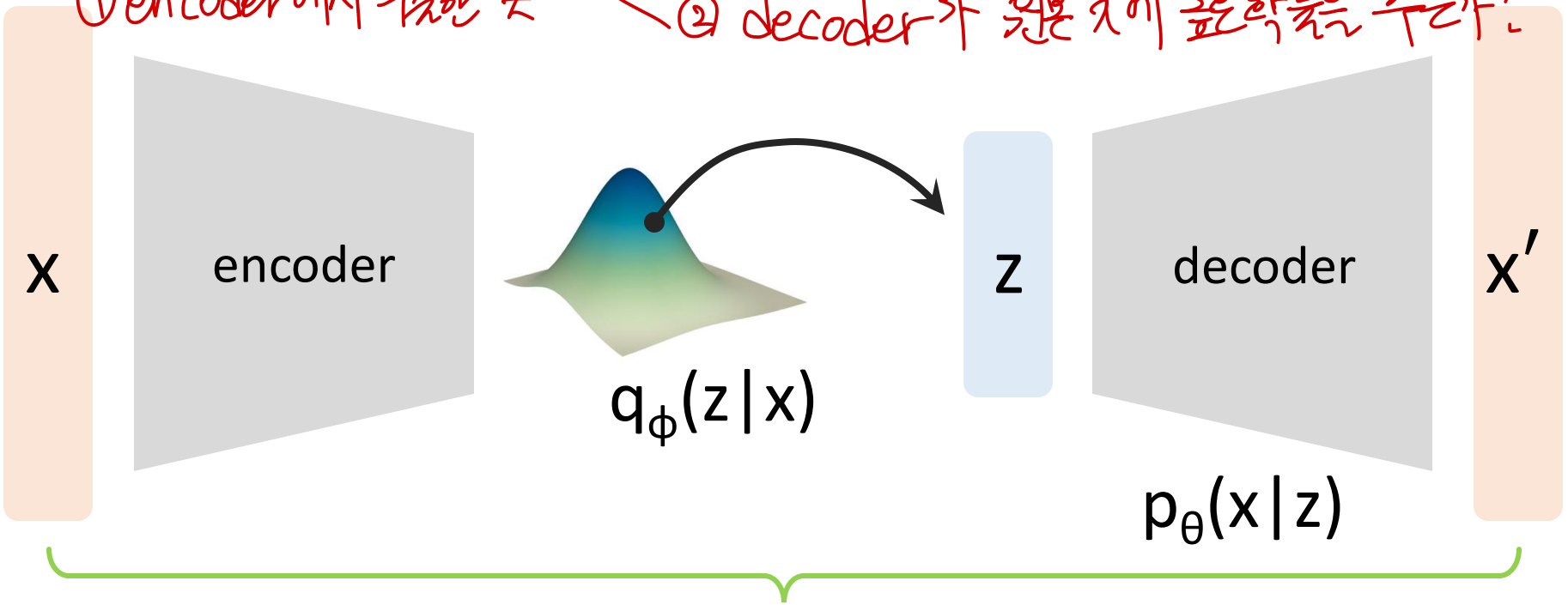
Variational Autoencoder

Maximize ELBO ⇒ minimize:

$$\mathcal{L}_{\theta,\phi}(x) = -\mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big] + \mathcal{D}_{\mathrm{KL}}\Big(q_\phi(z|x)||p(z)\Big)$$

① encoder에서 추출한 z

② decoder가 원본 x에 높은 확률을 주는가?



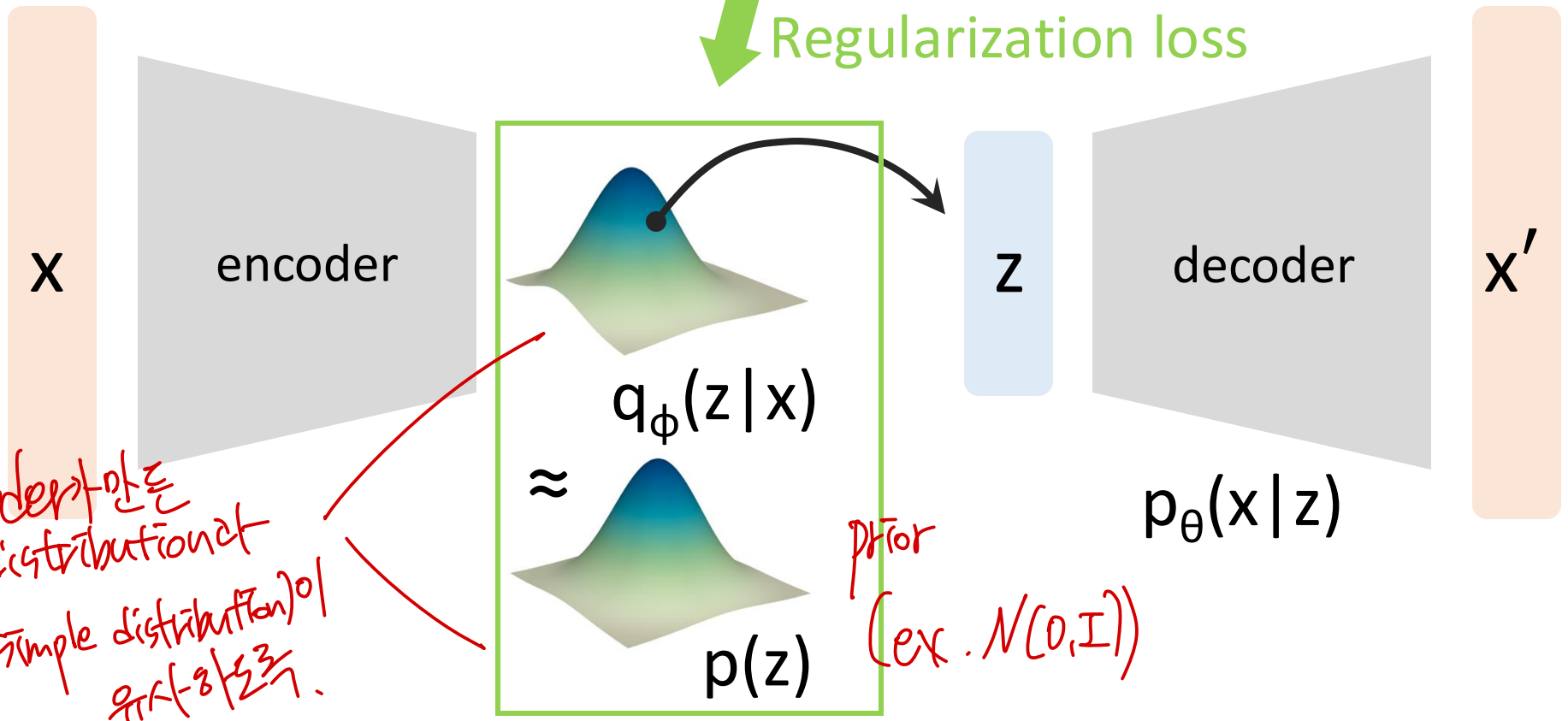
Variational Autoencoder

Maximize ELBO ⇒ minimize:

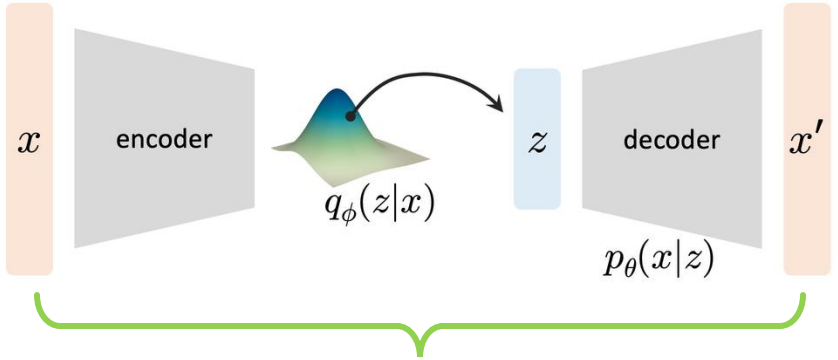
두 확률분포의 차이

$$\mathcal{L}_{\theta,\phi}(x) = -\mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big] + \mathcal{D}_{\mathrm{KL}}\Big(q_\phi(z|x)||p(z)\Big)$$

Regularization loss



Variational Autoencoder



Reconstruction loss

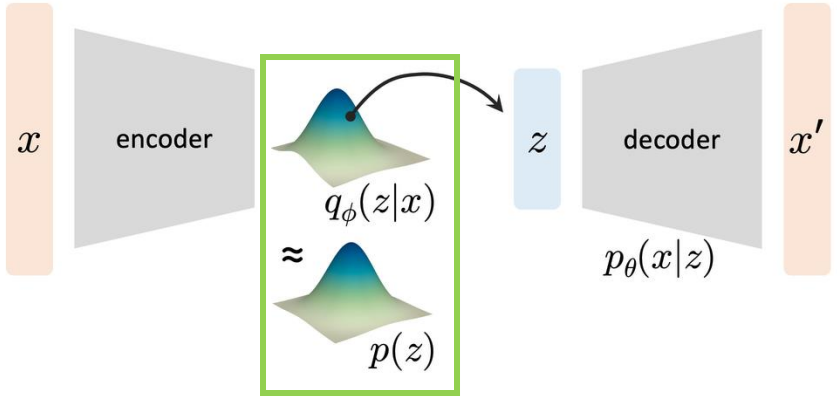
$$-\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x|z)\right]$$

Example: L2 loss

- ① one-step Monte Carlo: $z \sim q_\phi(z|x)$: encoder 통과해서 z를 sampling
- ② map z by decoder net: $g_\theta(z) \to x'$ network estimates distribution's parameters (Gaussian's mean)
- ③ model $p_\theta(x|z)$ by Gaussian: $p_\theta(x|z) = \mathcal{N}(x \mid x', \sigma_0^2)$ (assume fixed std)
- ④ negative log likelihood: $\frac{1}{2\sigma_0^2}||x - x'||^2 + const$ (상수, 상수, 최적화할때, 실제 x가 Gaussian을 따른다는 가정. (not latent z))

- L2 loss ⇒ a Gaussian neighborhood around data point x

Variational Autoencoder



Regularization loss

$$\mathcal{D}_{\mathrm{KL}}\left(q_\phi(z|x)||p(z)\right)$$

Example: Gaussian prior

- ① let $p(z) = \mathcal{N}(z \mid 0, \mathbf{I})$ 정규분포 가정
- ② model $q_\phi(z|x)$ by Gaussian: $\mathcal{N}(z \mid \mu, \sigma)$
- ③ map x by encoder net: $f_\phi(x) \to \mu, \sigma$ → encoder는 주어진 Gaussian의 parameter를 추정.
  again, network estimates distribution's parameters
- compute loss analytically: $\mathcal{D}_{\mathrm{KL}}\left(\mathcal{N}(z \mid \mu, \sigma) \,||\, \mathcal{N}(z \mid 0, \mathbf{I})\right)$

  → $\frac{1}{2}\sum_j \left(\mu_j^2 + \sigma_j^2 - \log \sigma_j^2 - 1\right)$ (const)

- fixed covariance ⇒ L2 loss on μ $\|\mu - 0\|^2$

  (σ 상수)

Variational Autoencoder

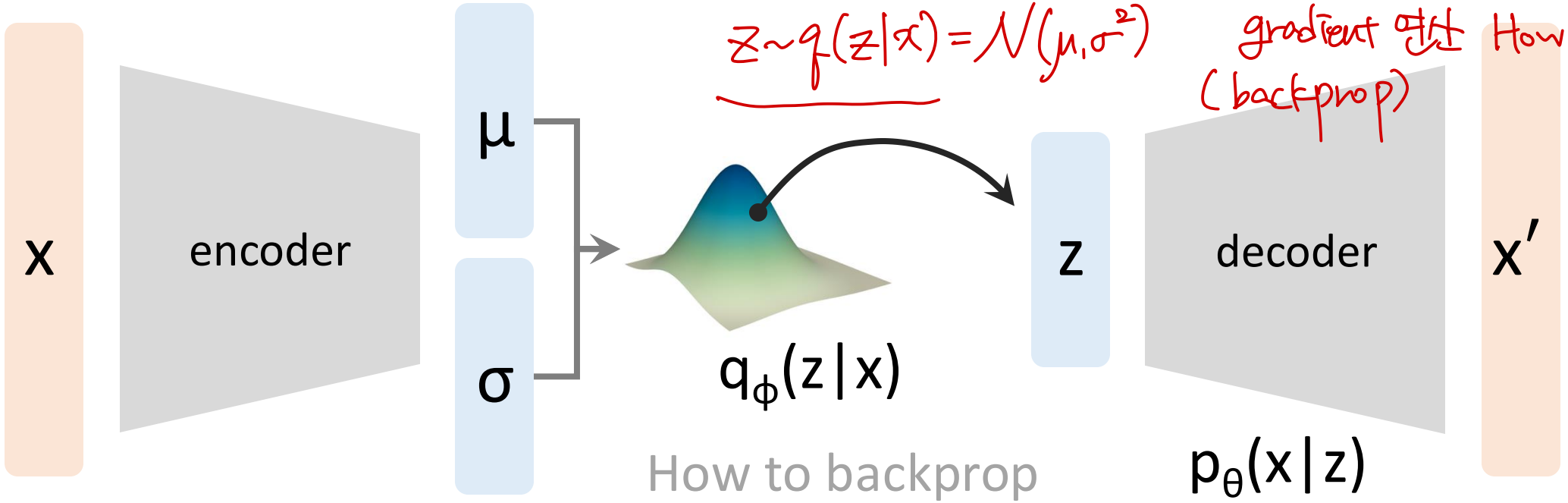
Maximize ELBO ⇒ minimize:

$$\mathcal{L}_{\theta,\phi}(x) = -\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x|z)\right] + \mathcal{D}_{\mathrm{KL}}\left(q_\phi(z|x)||p(z)\right)$$

$z \sim q(z|x) = \mathcal{N}(\mu, \sigma^2)$

확률적 sampling ≠ 일반적인 미분연산

gradient 연산 How? (backprop)

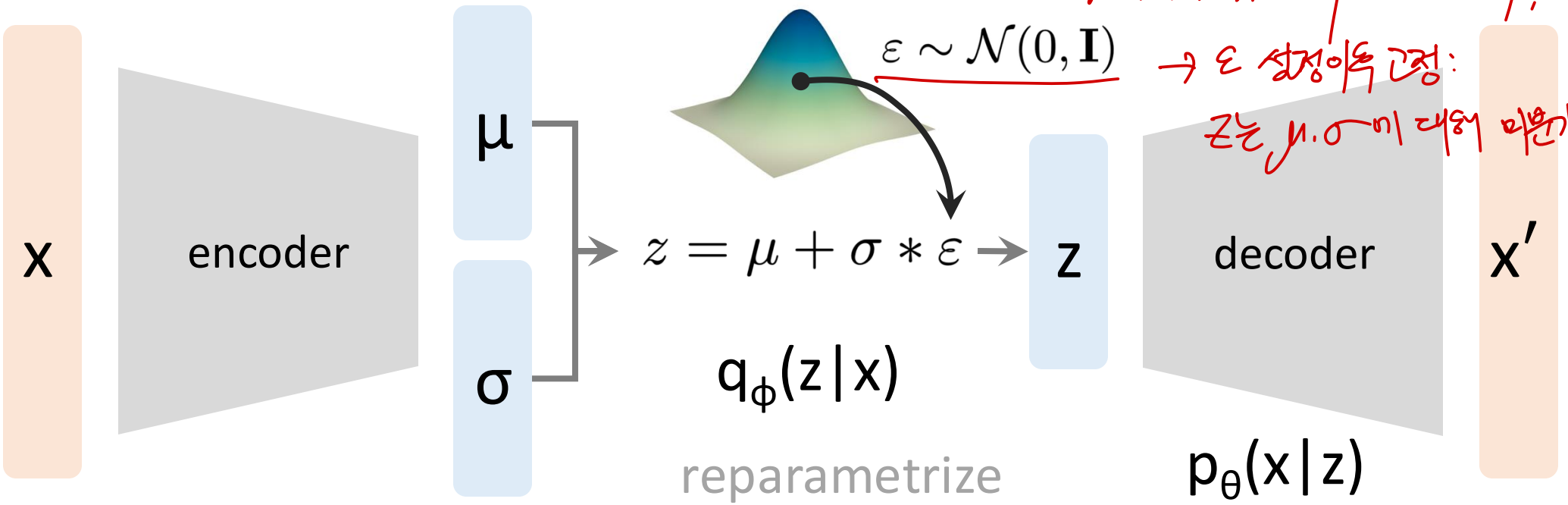


Variational Autoencoder

Maximize ELBO ⇒ minimize:

$$\mathcal{L}_{\theta,\phi}(x) = -\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x|z)\right] + \mathcal{D}_{\mathrm{KL}}\left(q_\phi(z|x)||p(z)\right)$$

randomness는 epsilon으로부터.
→ ε 설정이후 고정:
z는 μ, σ에 대해 미분가능.



Variational Autoencoder

... so far, we have discussed an objective on **one** x:

$$\mathcal{L}_{\theta,\phi}(x) = -\mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big] + \mathcal{D}_{\mathrm{KL}}\Big(q_\phi(z|x)||p(z)\Big)$$

Overall loss is expectation over data:

$$\mathcal{L}_{\theta,\phi} = \mathbb{E}_{x\sim p_{data}(x)}\Big[-\mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big] + \mathcal{D}_{\mathrm{KL}}\Big(q_\phi(z|x)||p(z)\Big)\Big]$$

한 x에 대한관계 → 전체

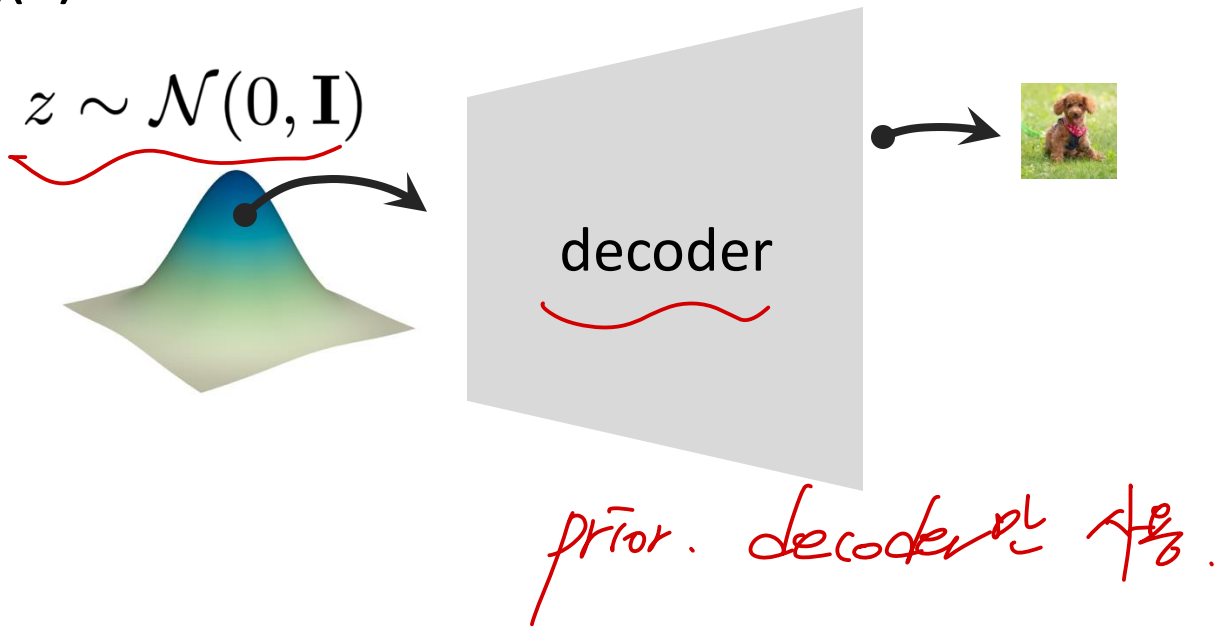
The formulation of "x-**conditional**" then "**marginalization**" is a common strategy

- Diffusion, Flow Matching, ...

Variational Autoencoder

Inference (generation): 생성 시에는 encoder X.

- sample z from: $\mathcal{N}(0, \mathbf{I})$
- map z by decoder net: $g_\theta(z)$

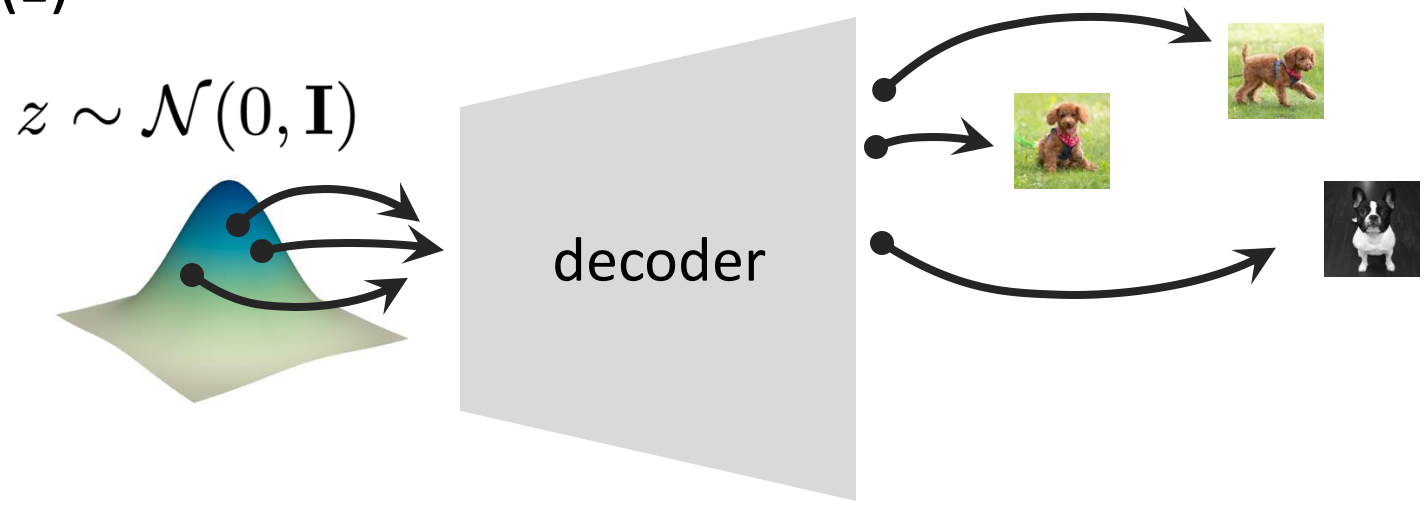


prior. decoder만 사용.

Variational Autoencoder

Inference (generation):

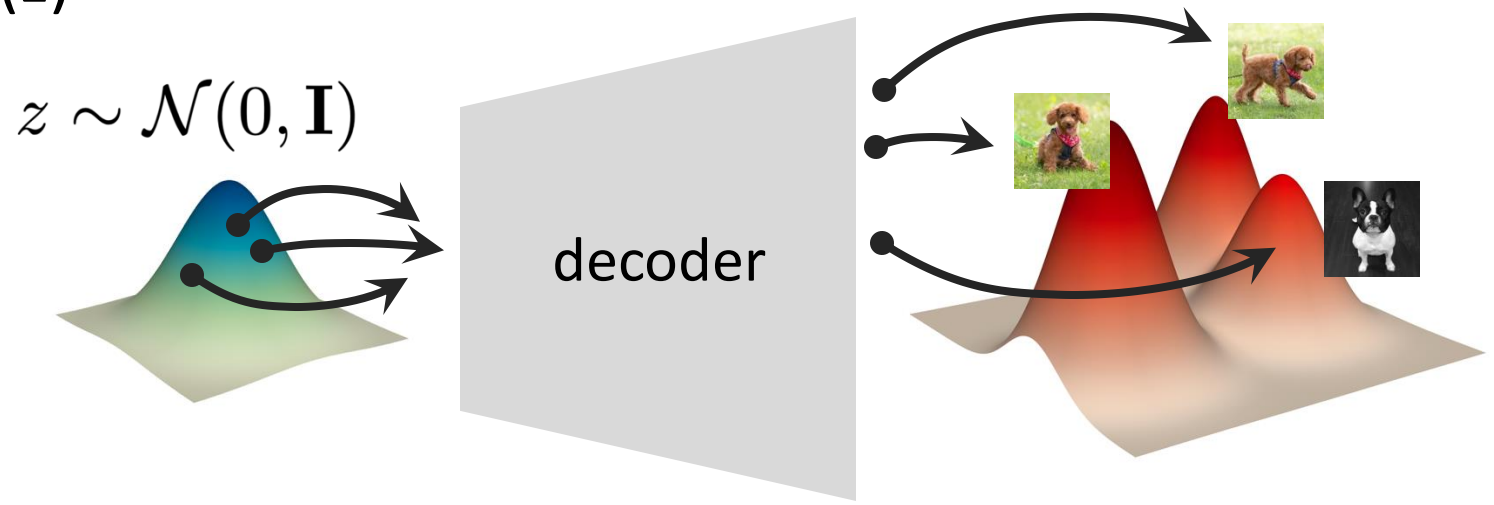
- sample z from: $\mathcal{N}(0, \mathbf{I})$
- map z by decoder net: $g_\theta(z)$



Variational Autoencoder

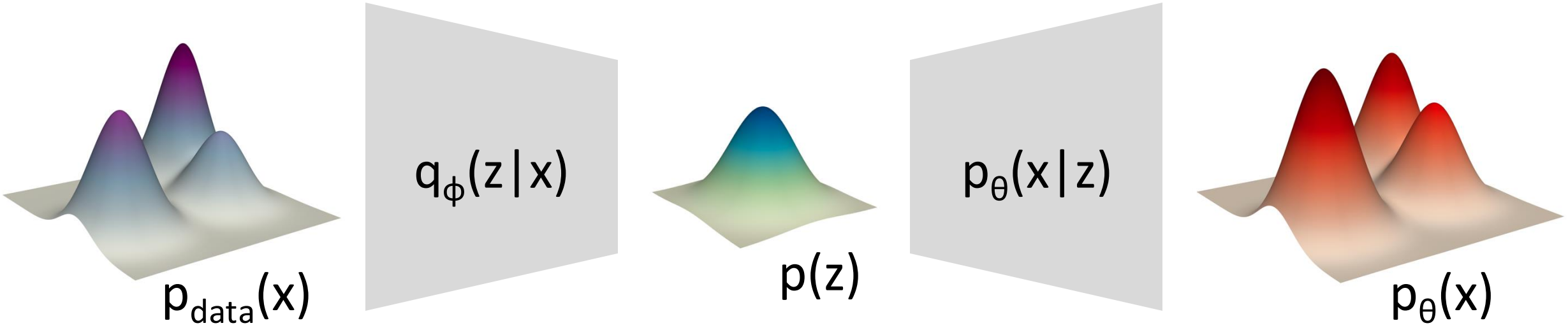
Inference (generation):

- sample z from: $\mathcal{N}(0, \mathbf{I})$
- map z by decoder net: $g_\theta(z)$



A view of “Autoencoding Distributions”

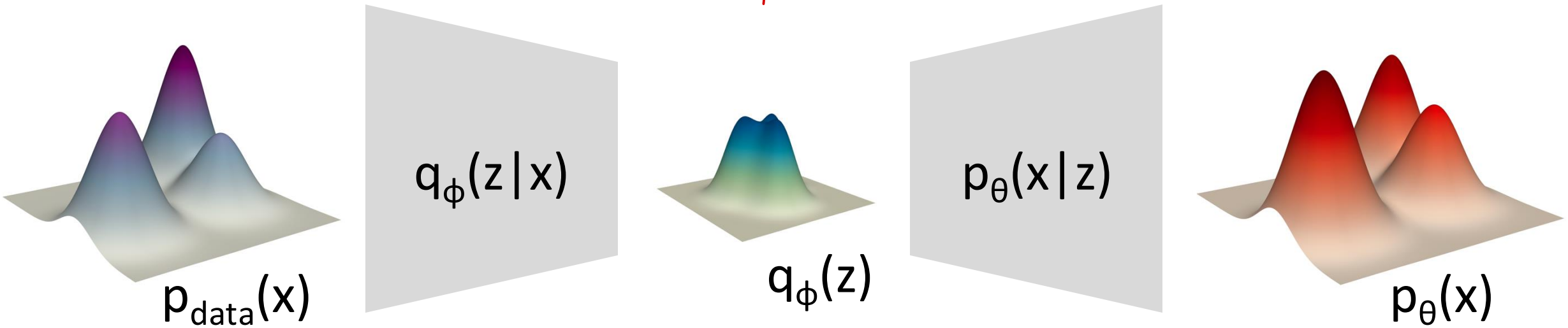
- **Encoder**: data to latent
- **Decoder**: latent to data



A view of “Autoencoding Distributions”

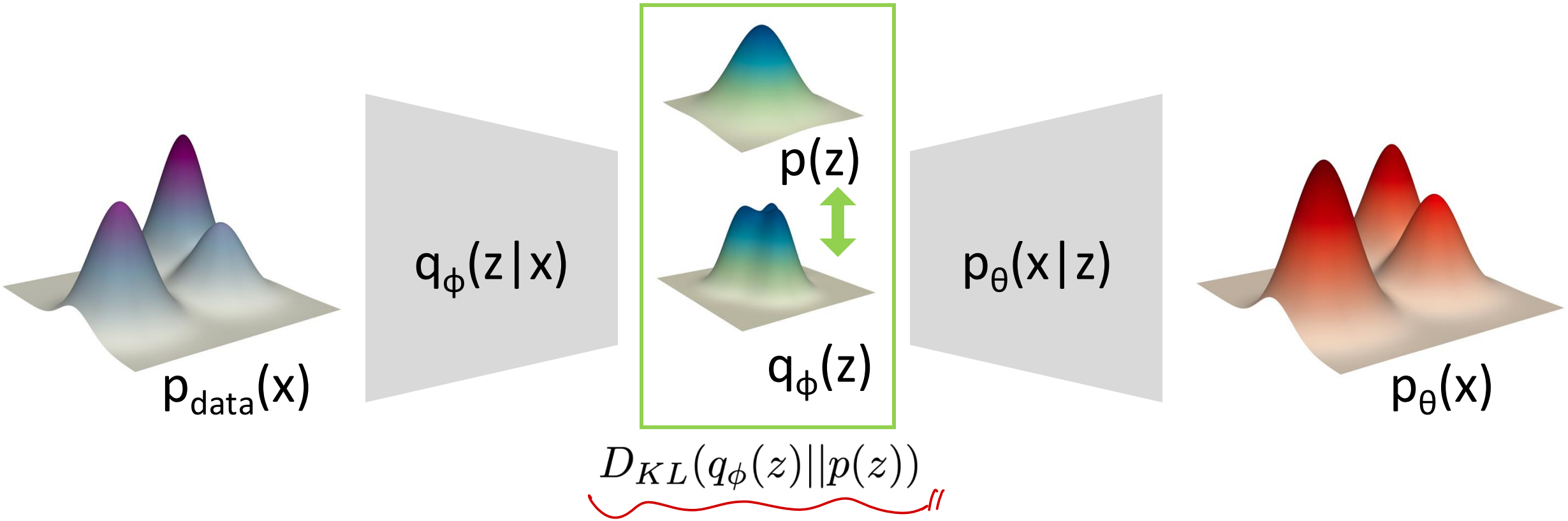
- encoded latent distribution: $q_\phi(z) = \int_x q_\phi(z|x) p_{data}(x) dx$

모든 x에 encoder에 넣어 얻은 latent의 합.



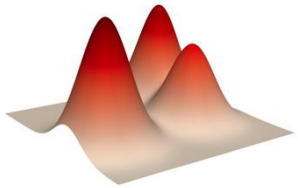
A view of "Autoencoding Distributions"

- encoded latent distribution: $q_\phi(z) = \int_x q_\phi(z|x)p_{data}(x)dx$

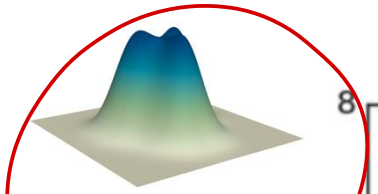


E.g., see "*InfoVAE: Information Maximizing Variational Autoencoders*", 2017

VAE: 2D latent space on MNIST

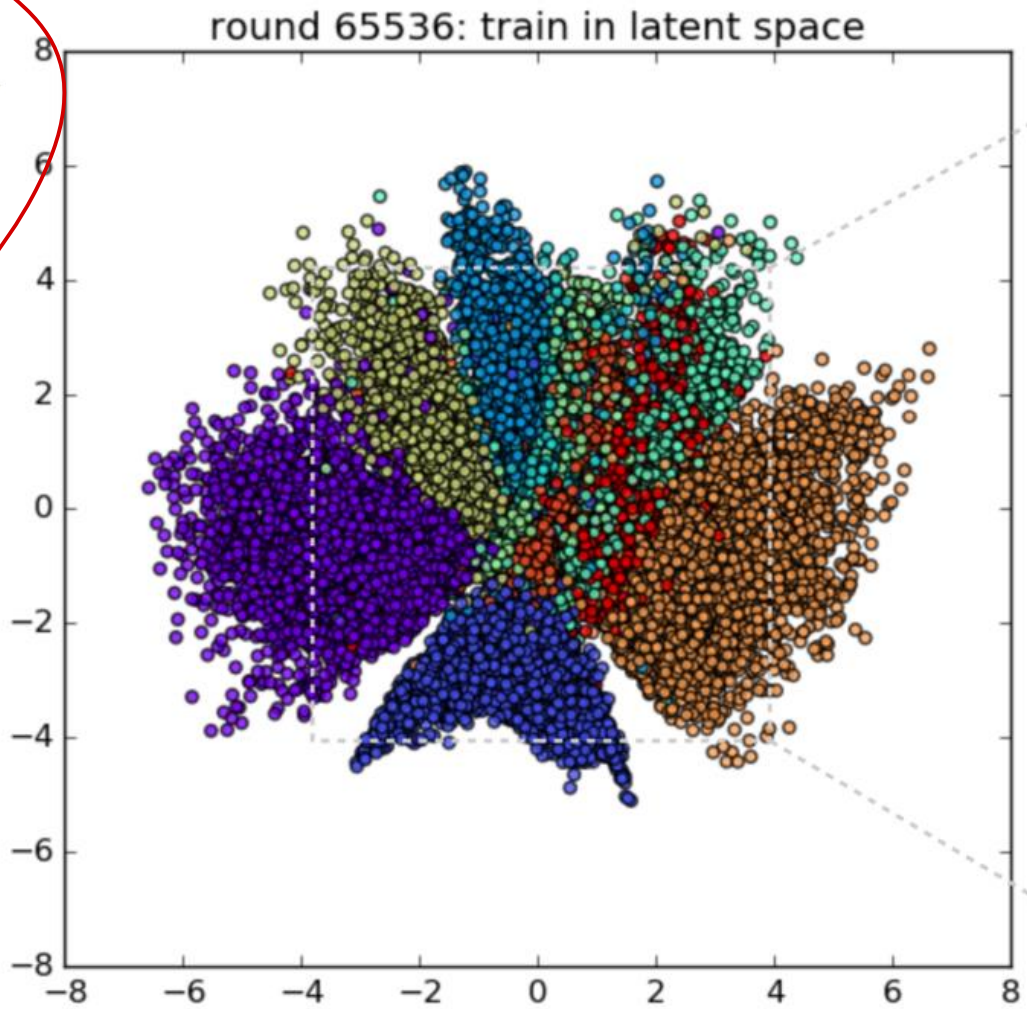


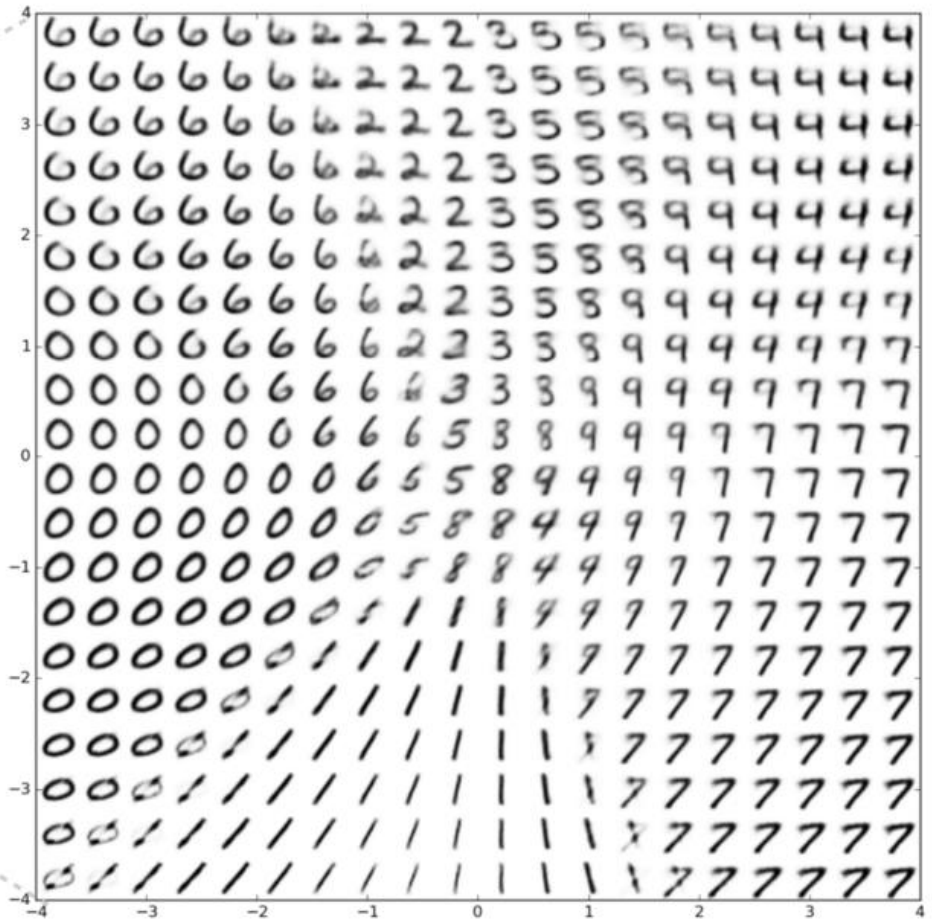
$p_\theta(x)$



$q_\phi(z)$

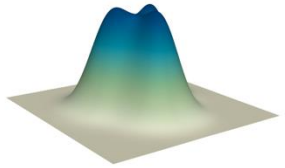




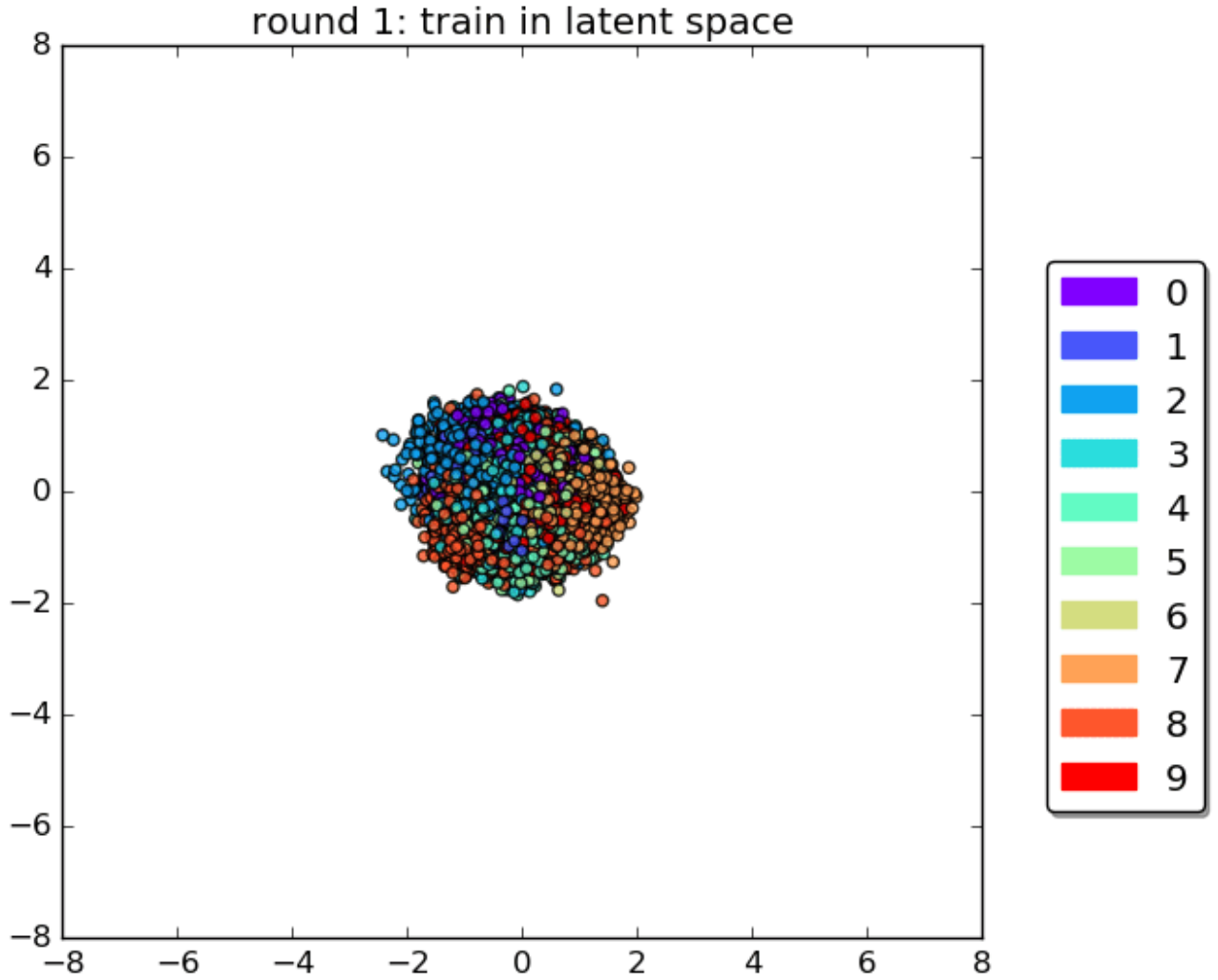


"Introducing Variational Autoencoders (in Prose and Code)"
https://blog.fastforwardlabs.com/2016/08/12/introducing-variational-autoencoders-in-prose-and-code.html

VAE: 2D latent space on MNIST

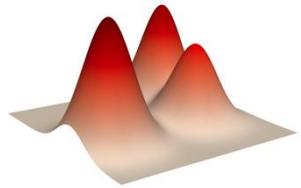


$q_\phi(z)$

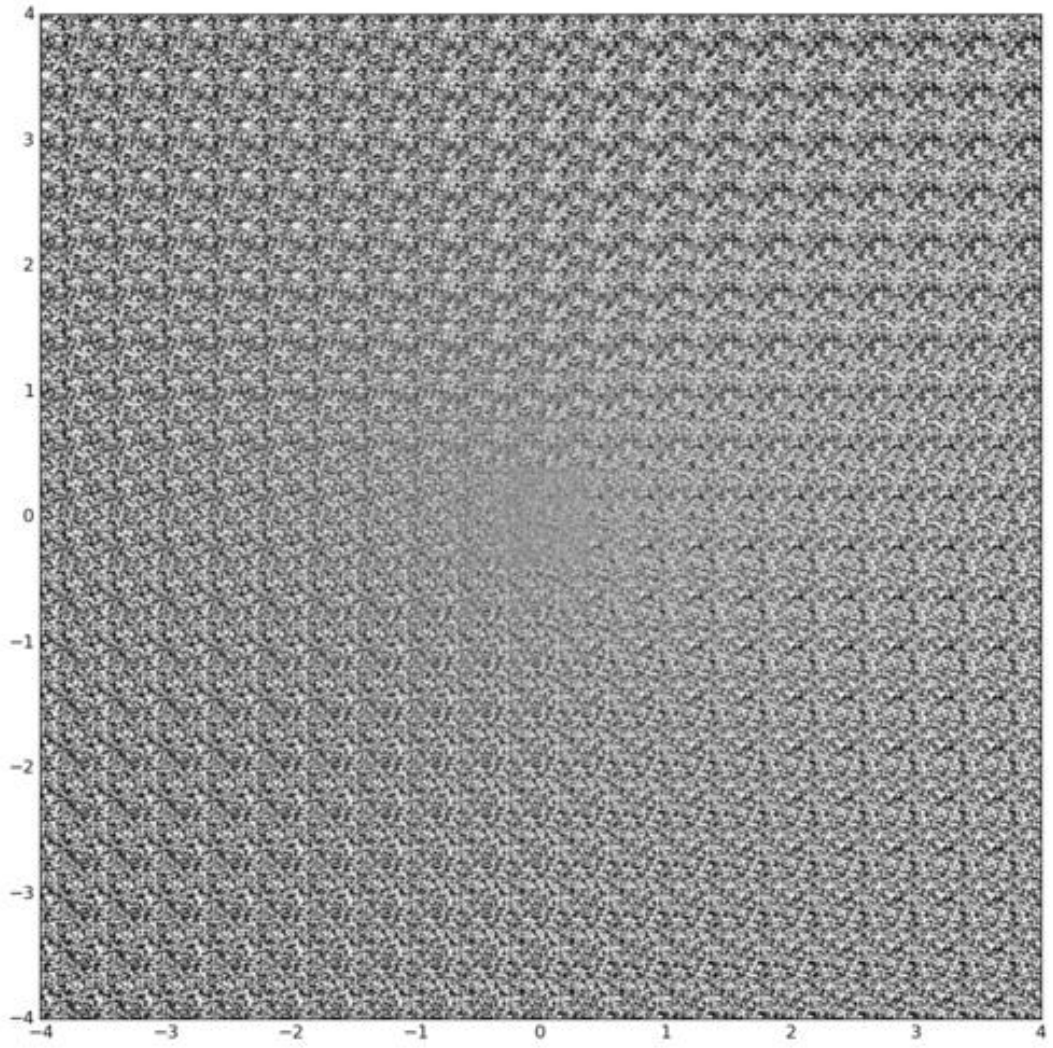


To pdf users: this is animation. Check it on: "Introducing Variational Autoencoders (in Prose and Code)"
https://blog.fastforwardlabs.com/2016/08/12/introducing-variational-autoencoders-in-prose-and-code.html

VAE: 2D latent space on MNIST



$p_\theta(x)$



To pdf users: this is animation. Check it on: "Introducing Variational Autoencoders (in Prose and Code)"
https://blog.fastforwardlabs.com/2016/08/12/introducing-variational-autoencoders-in-prose-and-code.html

VAE results on 784-d MNIST data



(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

Too strict to model the 784-d (28x28) joint distribution by independent distributions

Figure from: Kingma and Welling. “Auto-Encoding Variational Bayes”, ICLR 2014

VAE: 2D latent space on “Frey Face” dataset



Kingma and Welling. Auto-Encoding Variational Bayes, ICLR 2014

Issues of VAE

- Gaussian prior is too **strict** an assumption

$$p(z) = N(0, I)$$

latent 값 $p_\phi(z|x)$가 특정한 범위로 $N(0,I)$로 강제

- Reconstruction loss assumes **independence** across output elements

$p_\theta(x|z) = N(x; x', \sigma^2 I)$ → 각 픽셀들에 각 x'가서로 독립을 가정. (case of Image. pixel 간 관계 평가X)

- Difficult to model **high-dimensional**, **complex** distributions

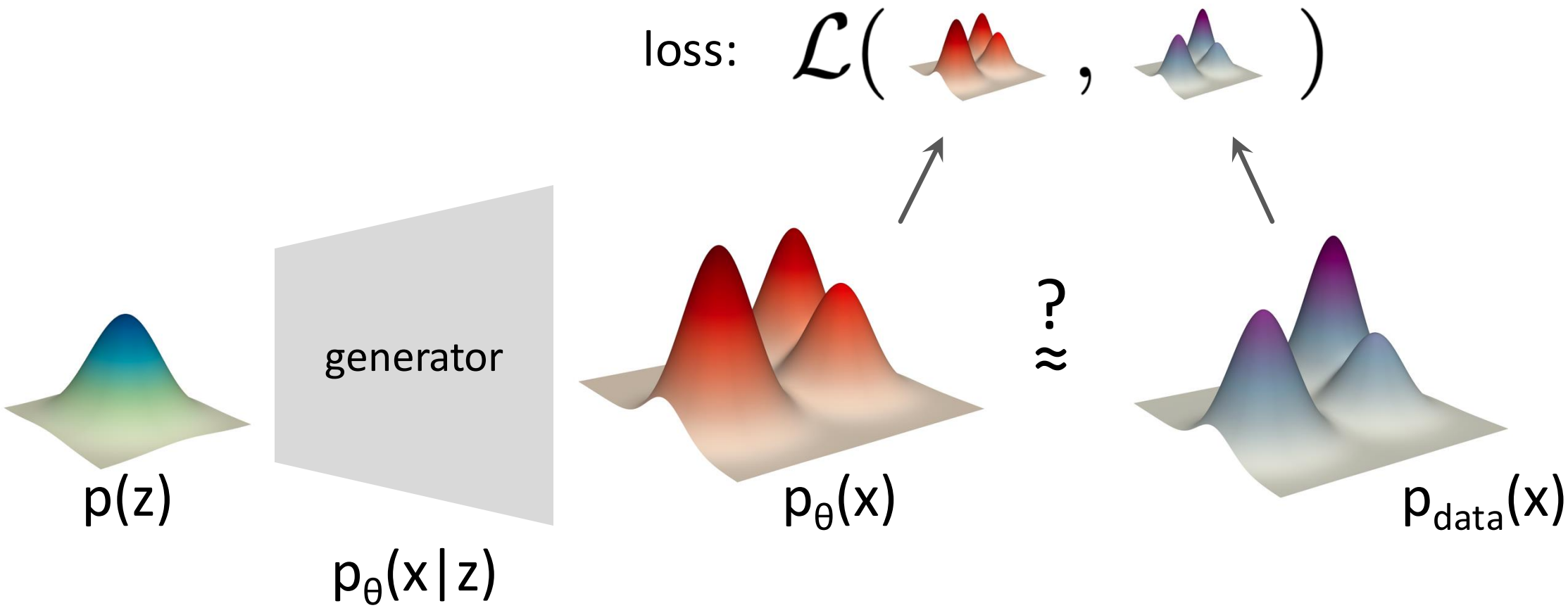
Gaussian latent factor, independence likelihood, reconstruction ↔ KL.

In practice - VAE is often used as a module (tokenizer), and the prior is learned by other models (AR or Diffusion).

ex. $x \xrightarrow{\text{VAE encode}} z \xrightarrow[\text{Diffusion}]{\text{AR}} z' \xrightarrow{\text{VAE decode}} \hat{x}$

Generative Adversarial Network (GAN)

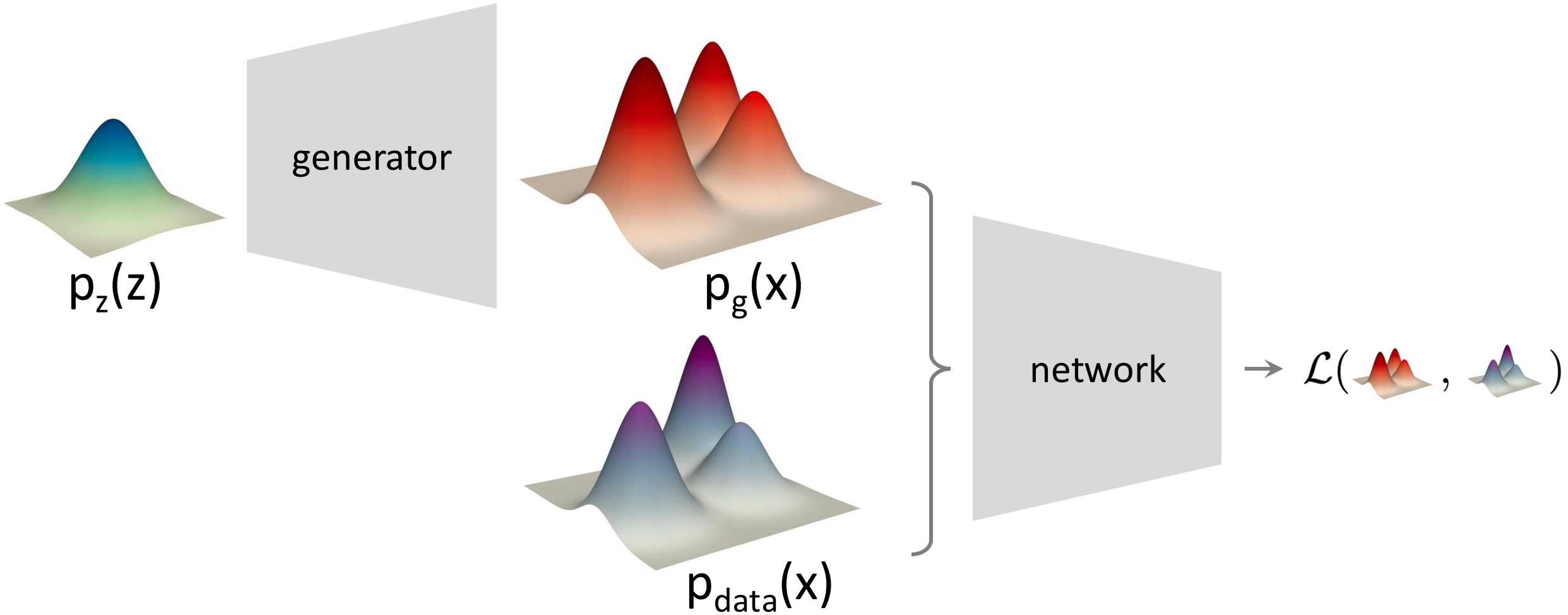
[Recap]: Measuring How Good A Distribution Is



- A core topic in generative modeling: make it **differentiable**, **computable**, and **tractable**

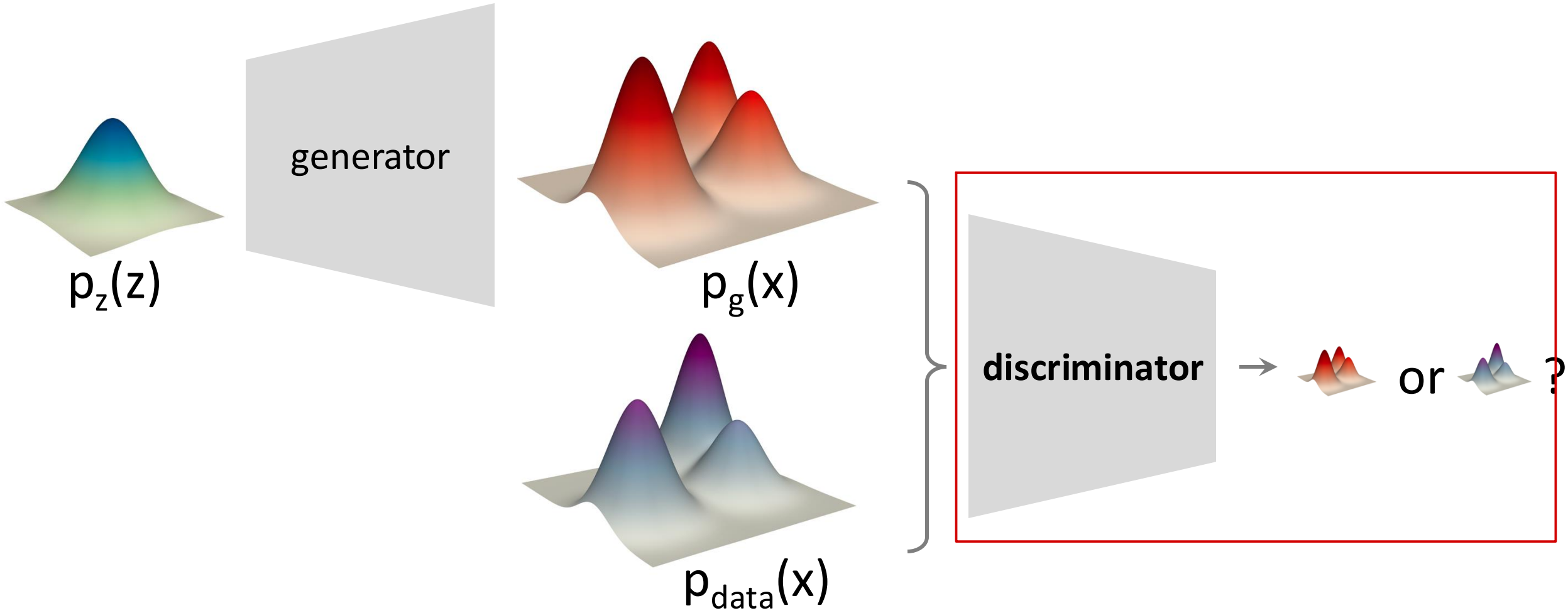
Generative Adversarial Networks

Estimating **distribution difference** by neural network 분포의 차이를 NN으로학습..



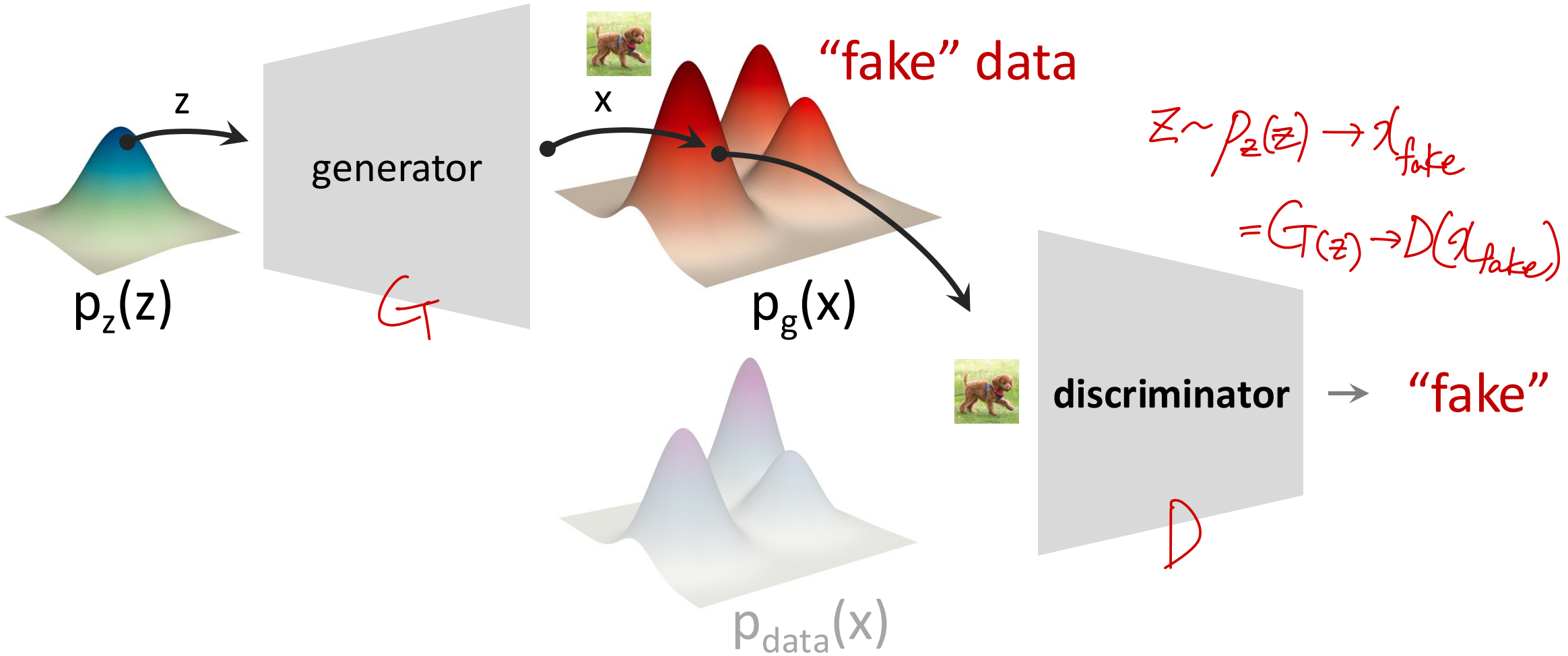
Generative Adversarial Networks

Estimating **distribution difference** by neural network



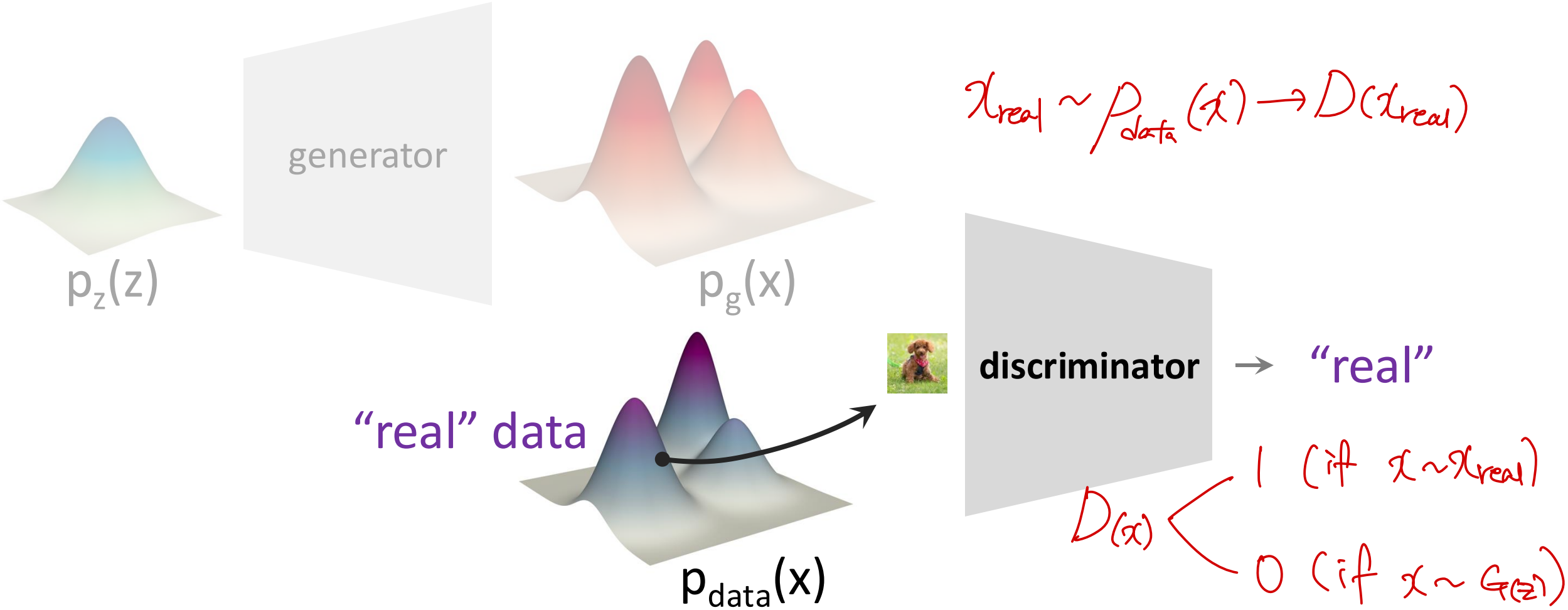
Generative Adversarial Networks

Estimating **distribution difference** by neural network



Generative Adversarial Networks

Estimating **distribution difference** by neural network



Adversarial Objective

$$\min_G \max_D \mathcal{L}(D, G) = \mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$$

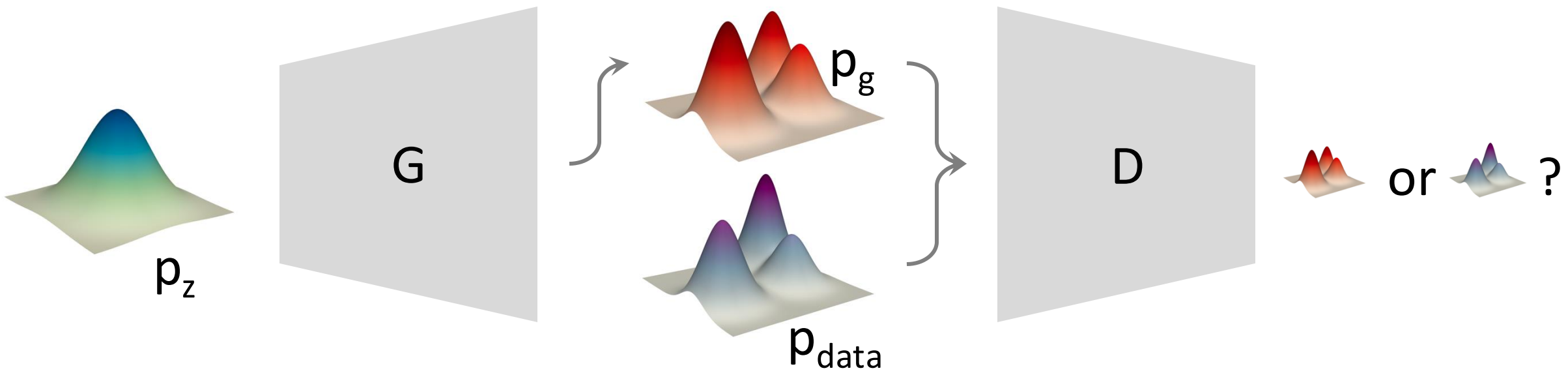
min-max process

D는 값을 최대화

G는 값을 최소화

real인 확률

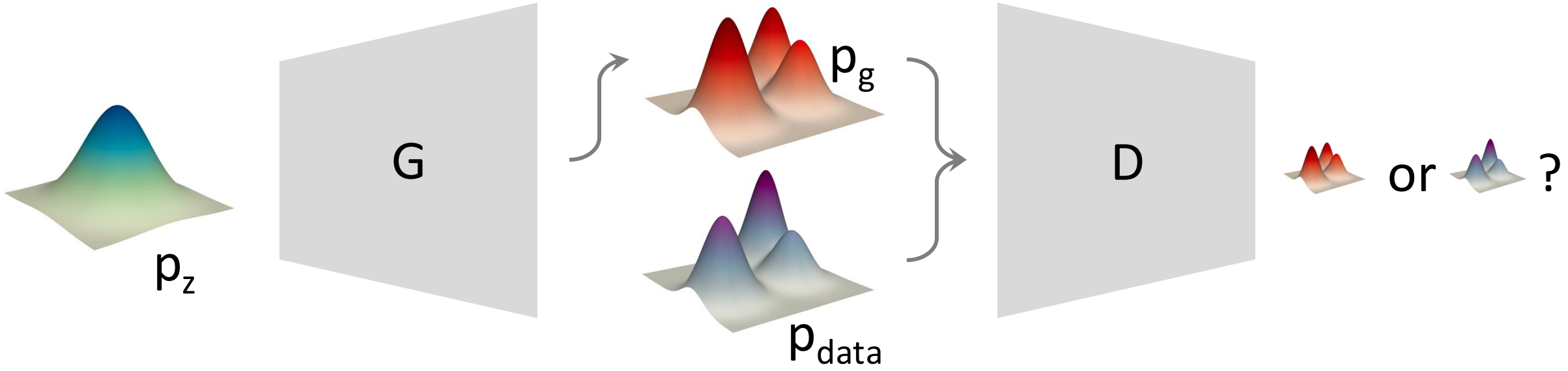
fake



Adversarial Objective: D-step

$$\min_G \max_D \mathcal{L}(D, G) = \mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$$

D-step: fix G, optimize D



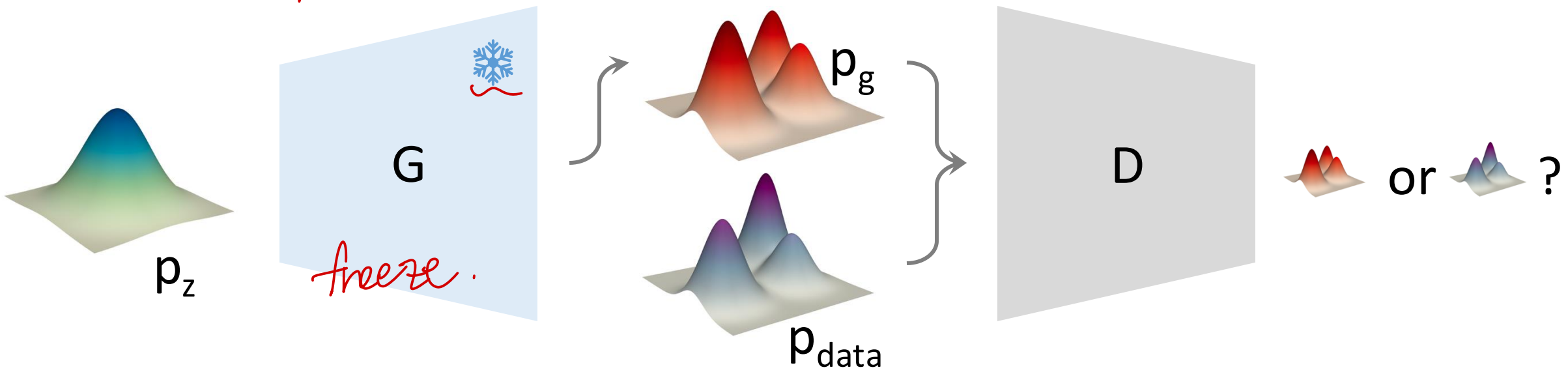
Adversarial Objective: D-step

$$\max_D \mathcal{L}(D) = \mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$$

push to 1 push to 0

D-step: fix G, optimize D

- D : classify real or fake
- binary logistic regression (sigmoid + cross-entropy)



Adversarial Objective: D-step

=> sample x from p_g

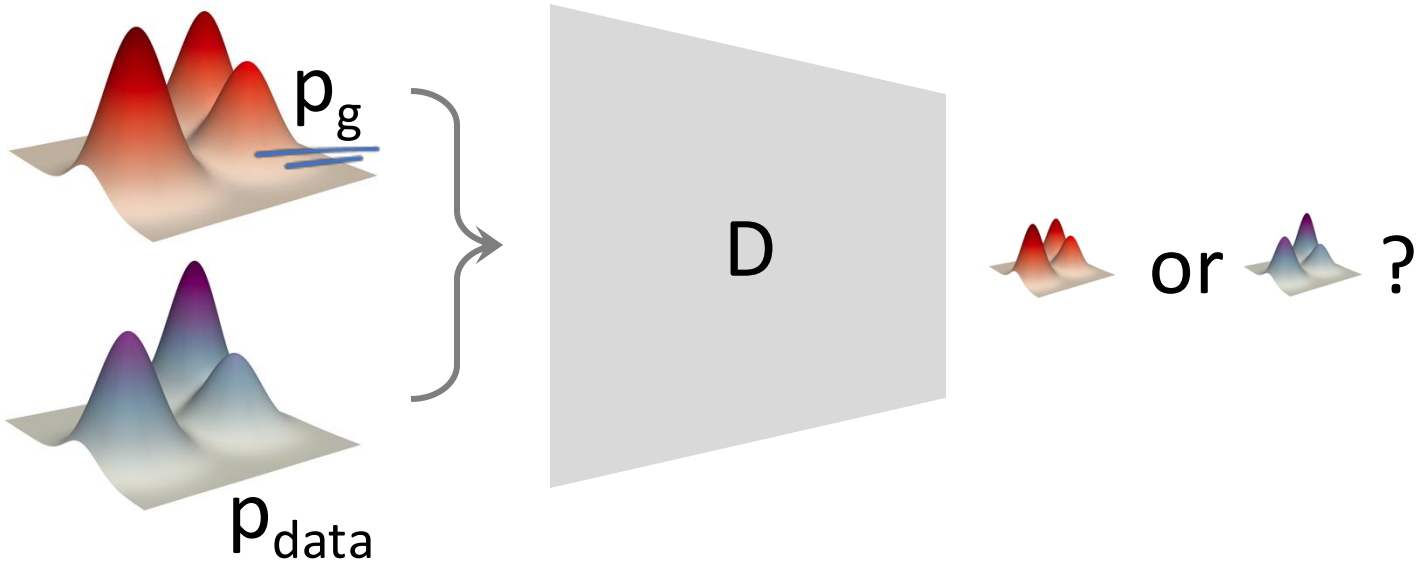
$$\max_D \mathcal{L}(D) = \mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{x \sim p_g}[\log(1 - D(x))]$$

push to 1 push to 0

D-step: fix G, optimize D

$x \sim p_z$, $x = G(z)$ 치환

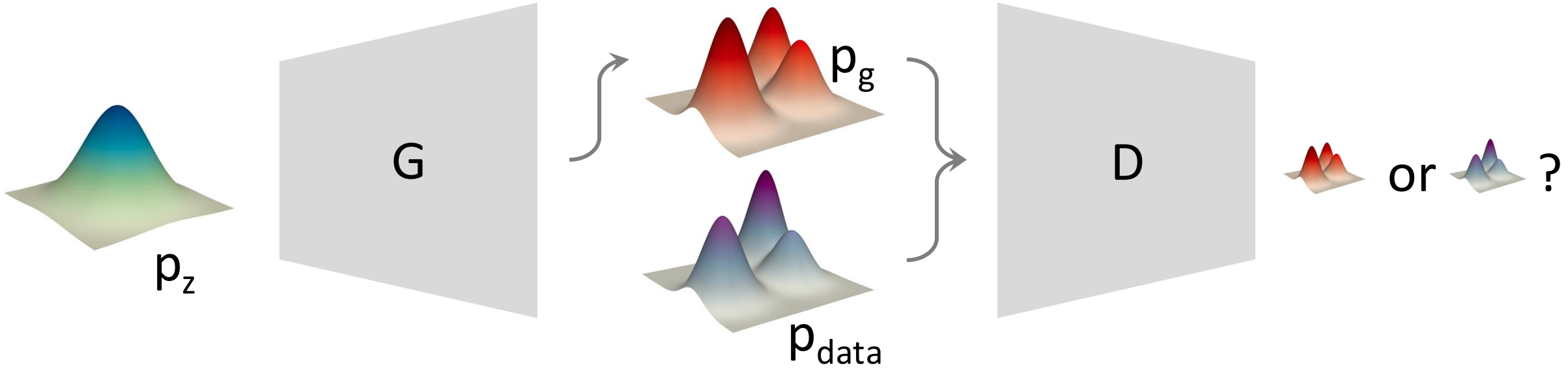
- D : classify real or fake
- binary logistic regression (sigmoid + cross-entropy)



Adversarial Objective: G-step

$$\min_G \max_D \mathcal{L}(D, G) = \mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$$

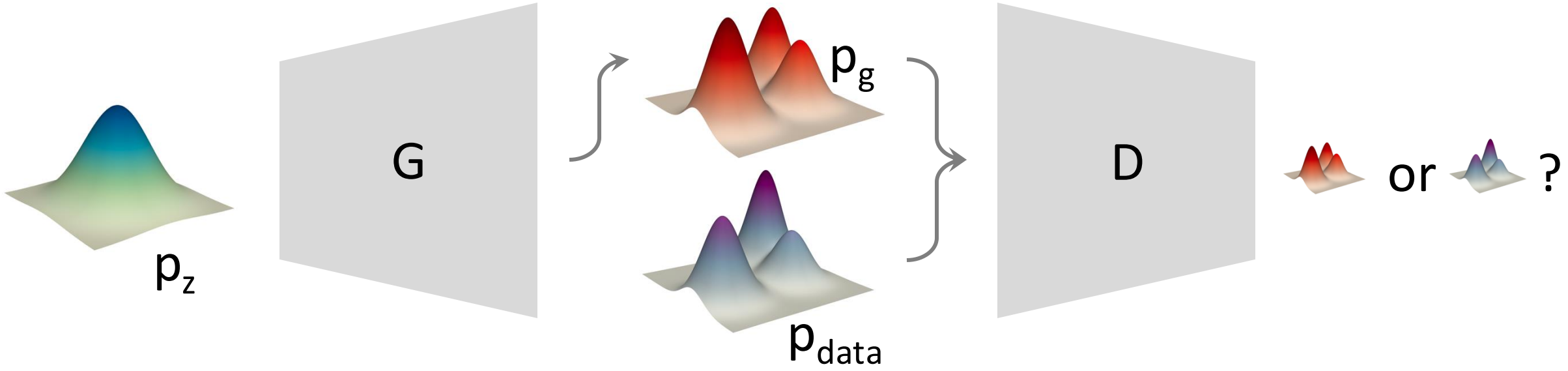
G-step: fix D, optimize G



Adversarial Objective: G-step

$$\min_G \max_D \mathcal{L}(D, G) = \mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$$

G-step: fix D, optimize G



Adversarial Objective: G-step

$$\min_G \mathcal{L}(G) = \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$$

push to 1

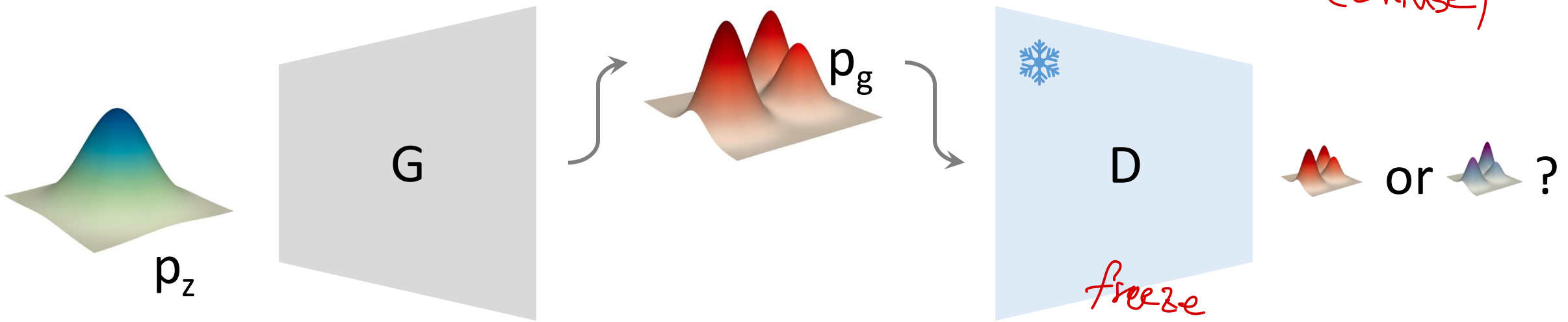
$D(G(z)) = 1$ mean

D가 fake를 real로 판단했다.

(confuse)

G-step: fix D, optimize G

- generate fake data such that D classifies it as "real"
- G to "confuse" D



Adversarial Objective: G-step

a "flip" trick:

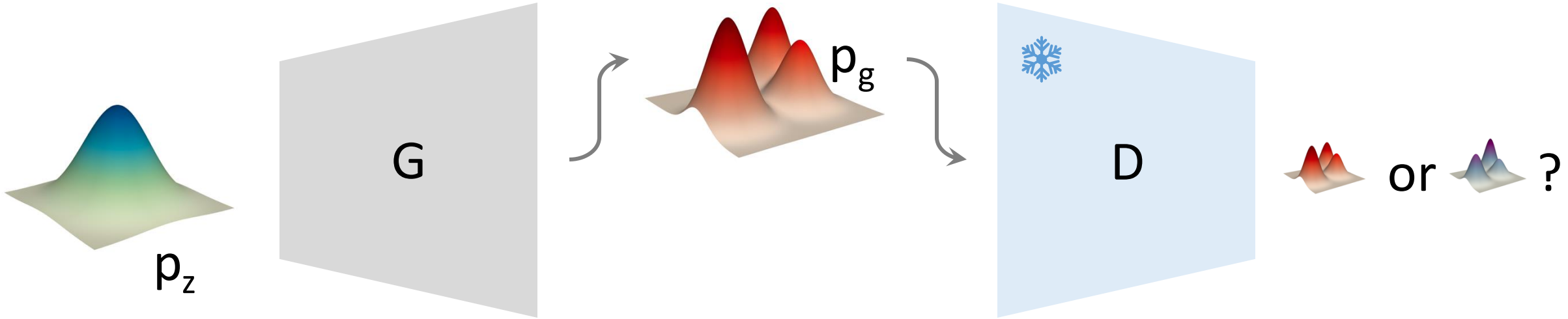
max

$$\min_G \mathcal{L}(G) = \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$$

push to 1

G-step: fix D, optimize G

- generate fake data such that D classifies it as "real"
- G to "confuse" D



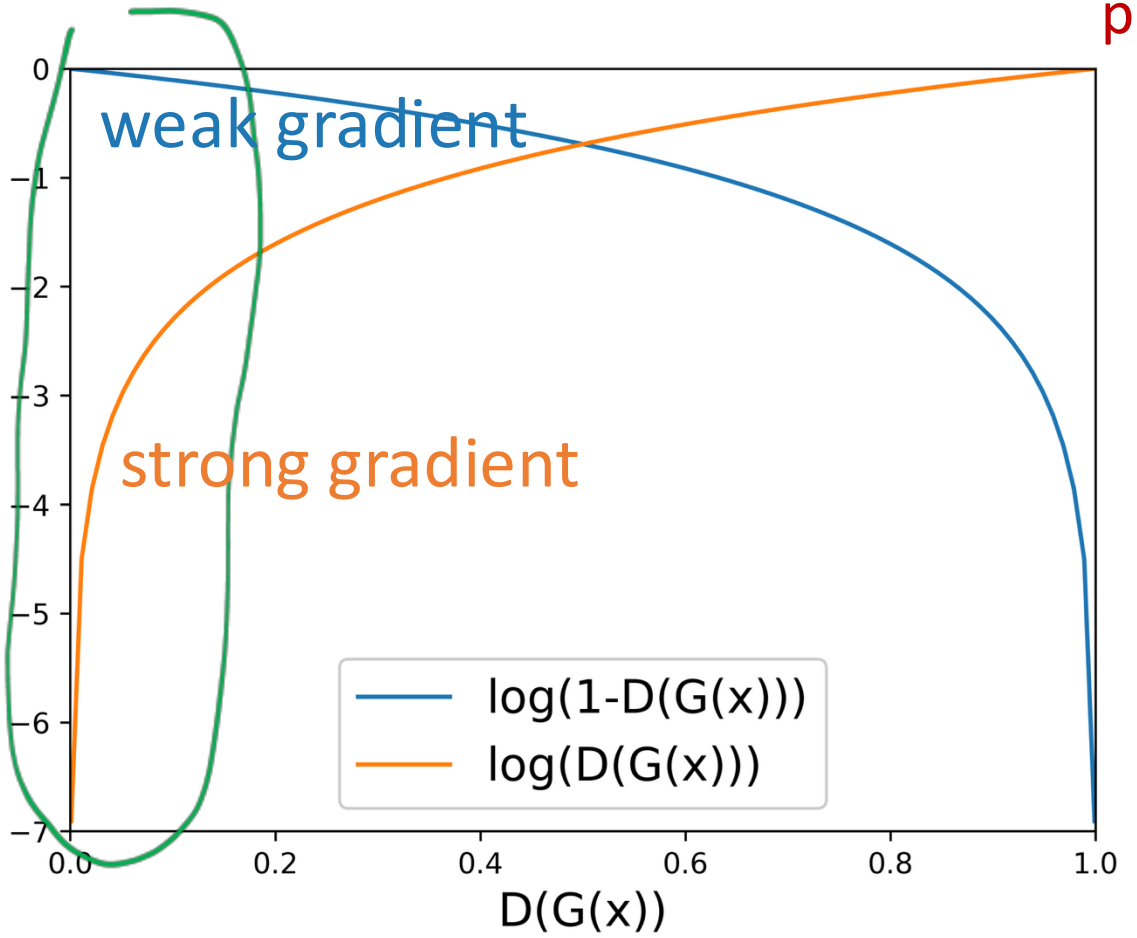
Adversarial Objective: G-step

Why?

a "flip" trick:

$$\max_G \mathcal{L}(G) = \mathbb{E}_{z \sim p_z}[\log(D(G(z)))]$$

push to 1



Early in training:

- G is poor
- D(G) is near 0

Generator의 성과가 안 좋은 초기 학습에 유리

GAN algorithm annotated

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

minibatch SGD

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

D step

  end for

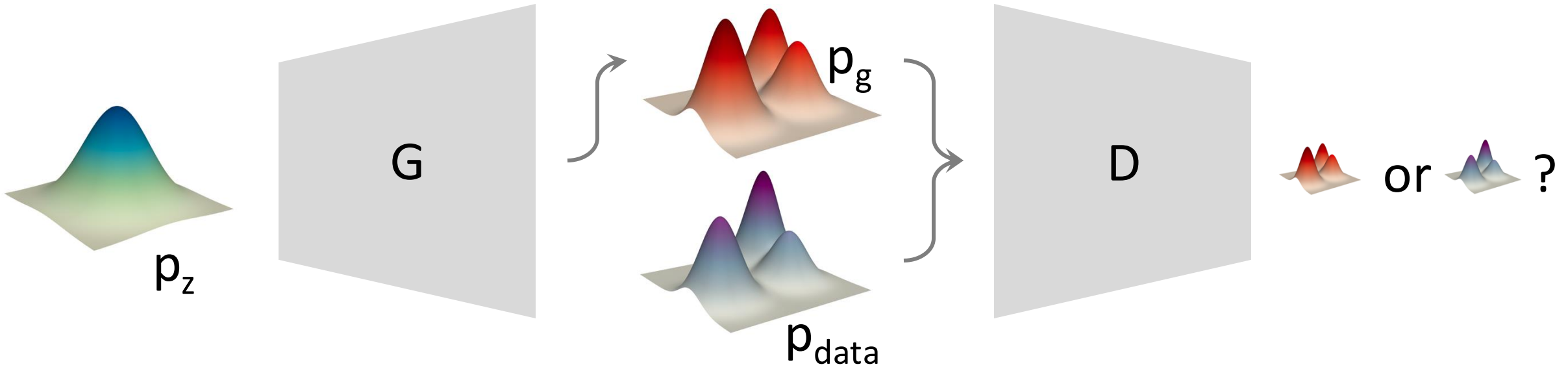
  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

G step.

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.



GAN algorithm annotated

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

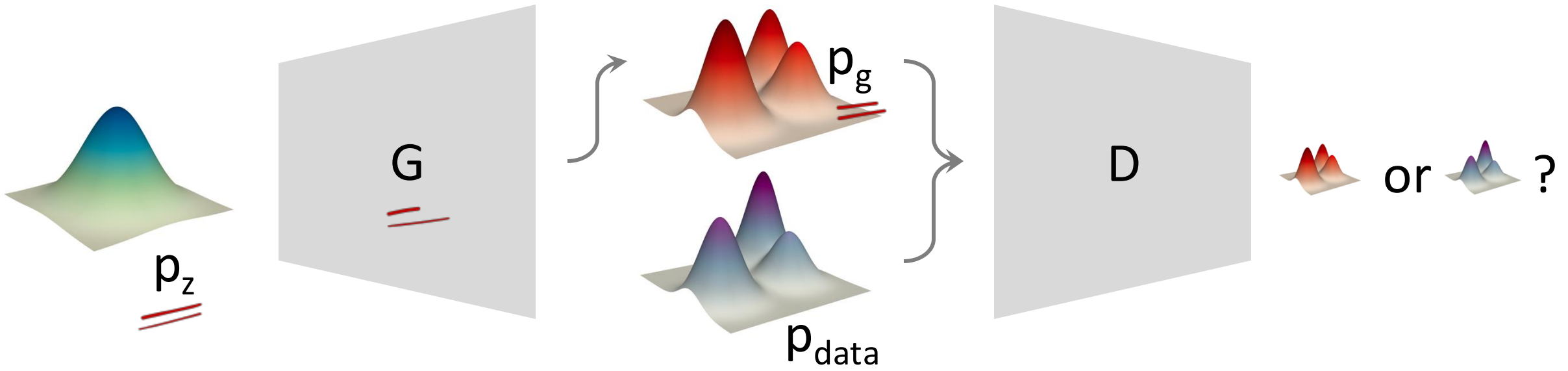
  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.



GAN algorithm annotated

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

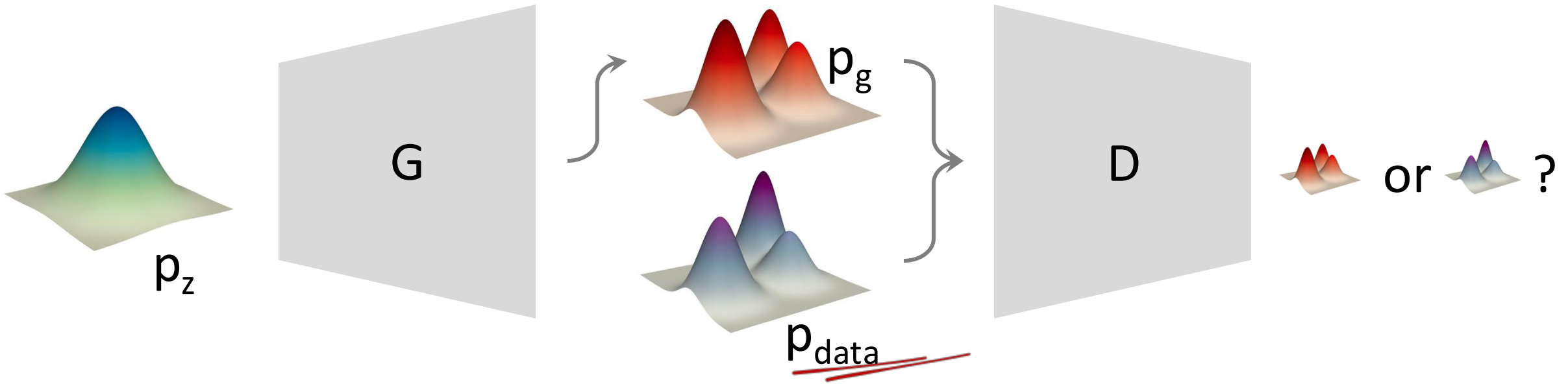
  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.



GAN algorithm annotated

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

D-step

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

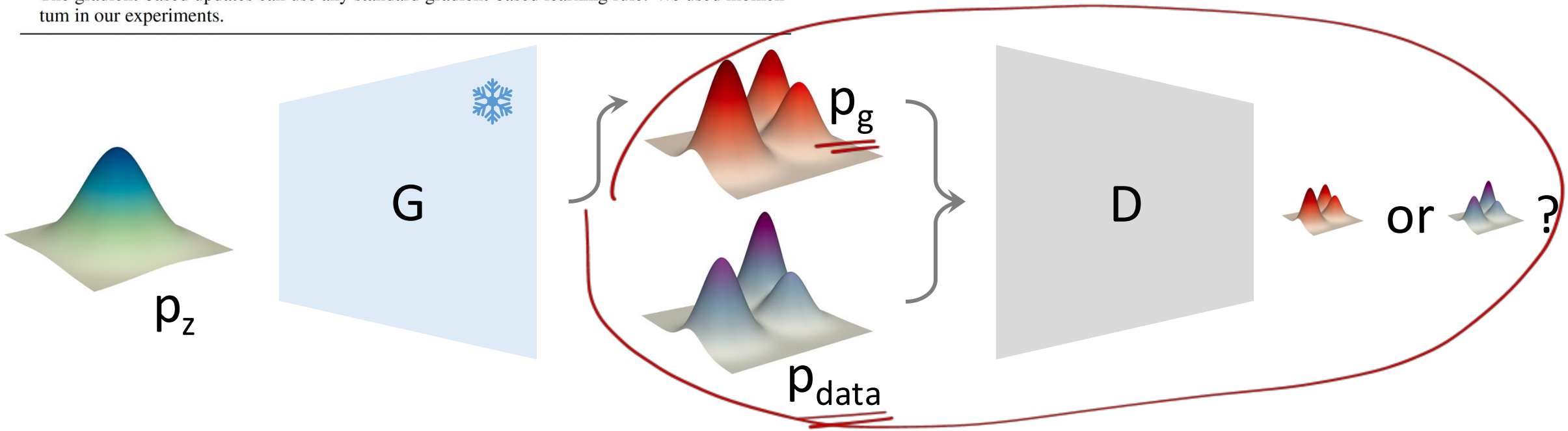
- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

gradient **ascend**
(maximize)



GAN algorithm annotated

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

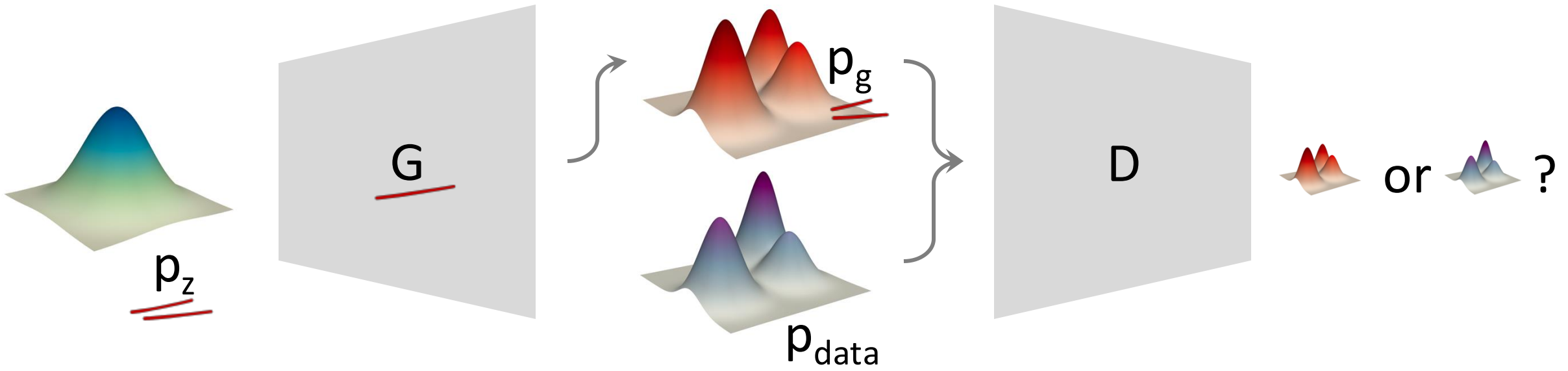
  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.



GAN algorithm annotated

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

G-step

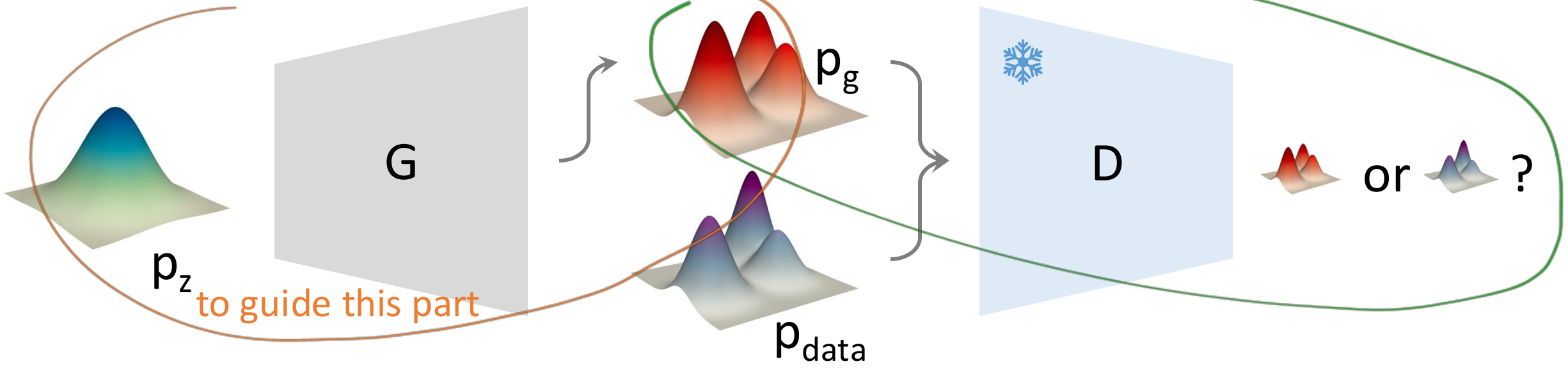
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

gradient **descend** (minimize)

a parameterized loss function



GAN algorithm annotated

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

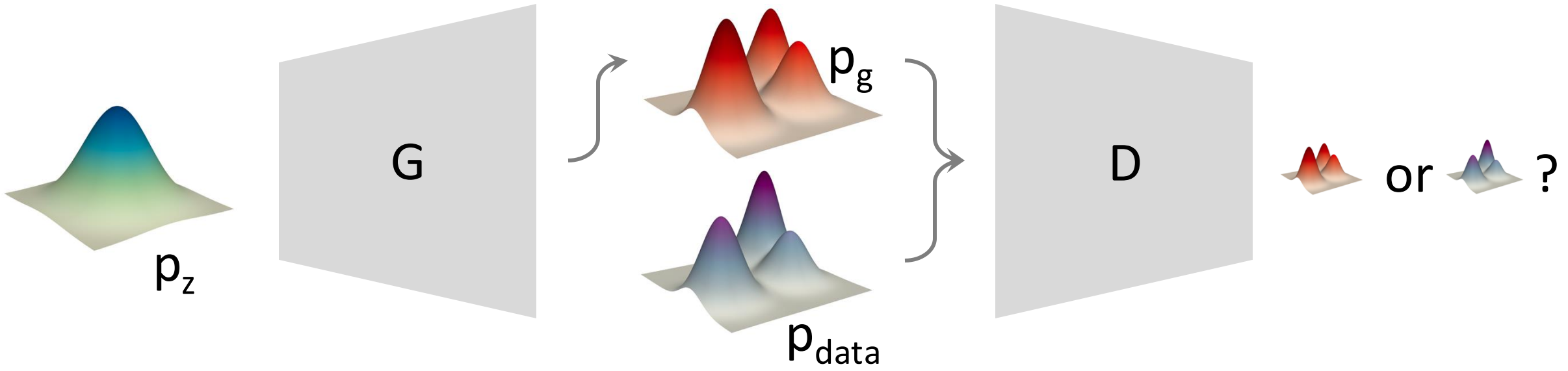
  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

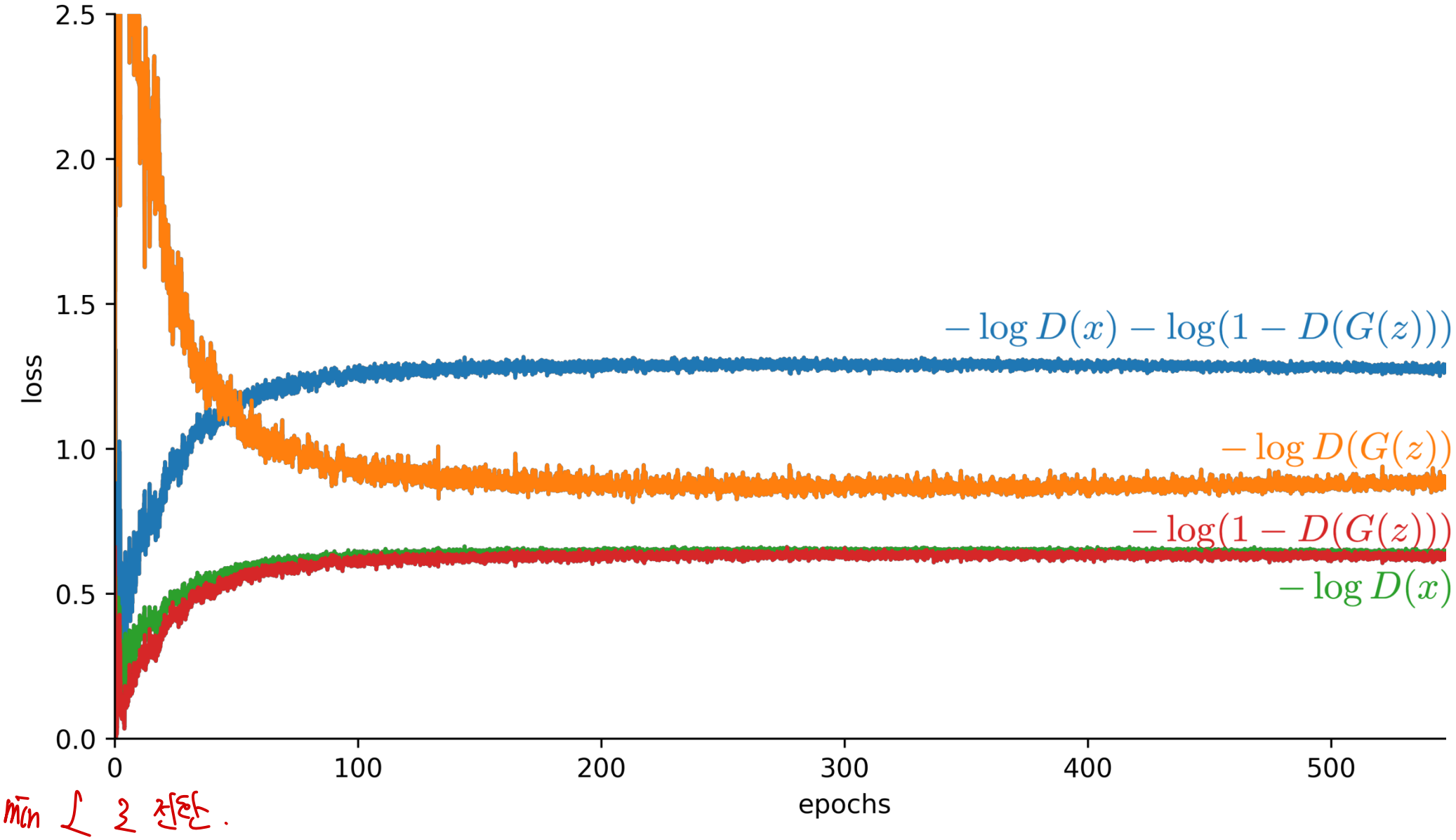
end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

iterating min-max



Running example: MNIST



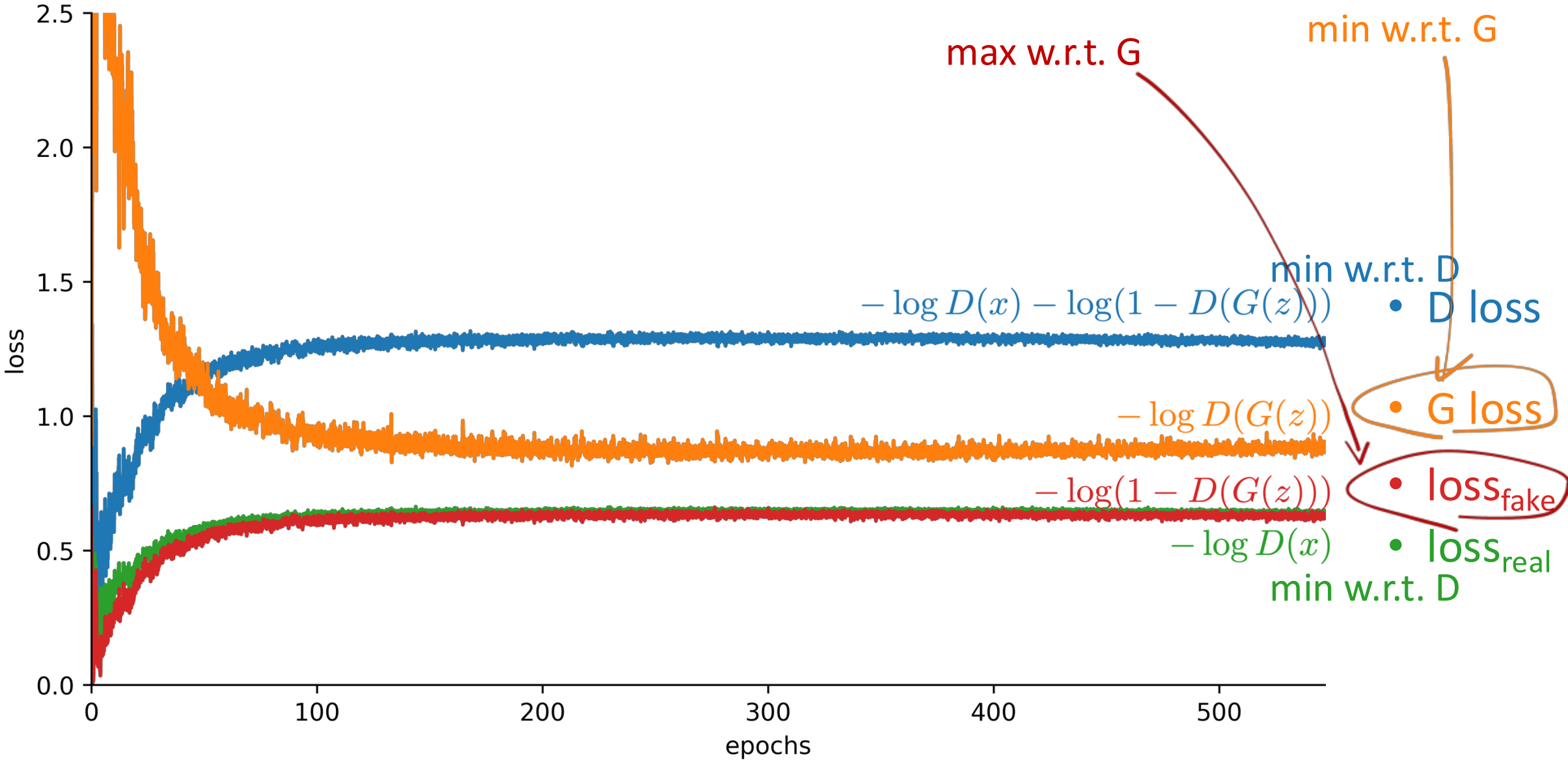
- D loss
- G loss
- $\text{loss}_{\text{fake}}$
- $\text{loss}_{\text{real}}$

min L 로 정함.

*All objectives are negative of their original form

Code adapted from: https://github.com/prcastro/pytorch-gan/tree/master

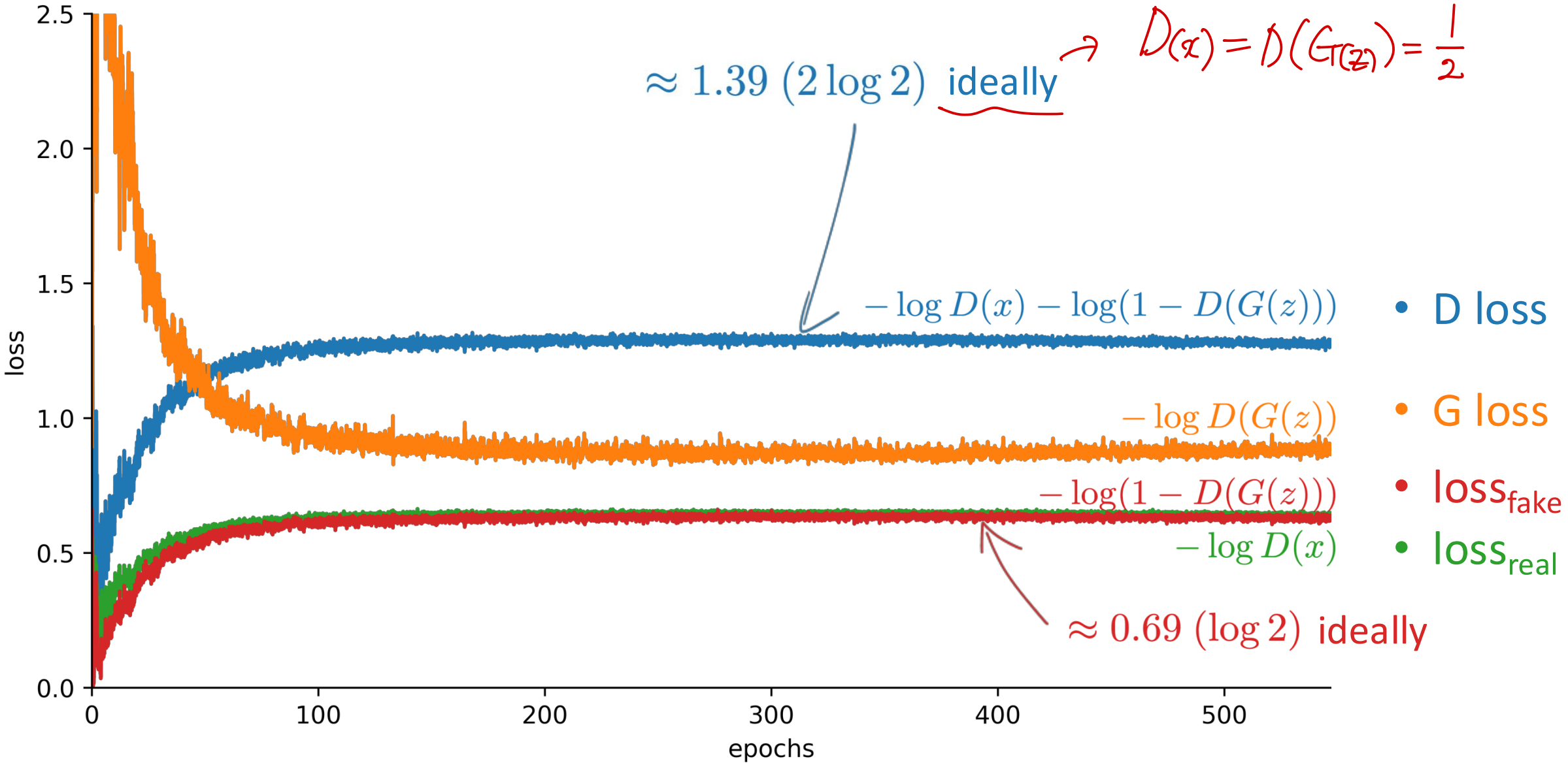
Running example: MNIST



*All objectives are negative of their original form

Code adapted from: https://github.com/prcastro/pytorch-gan/tree/master

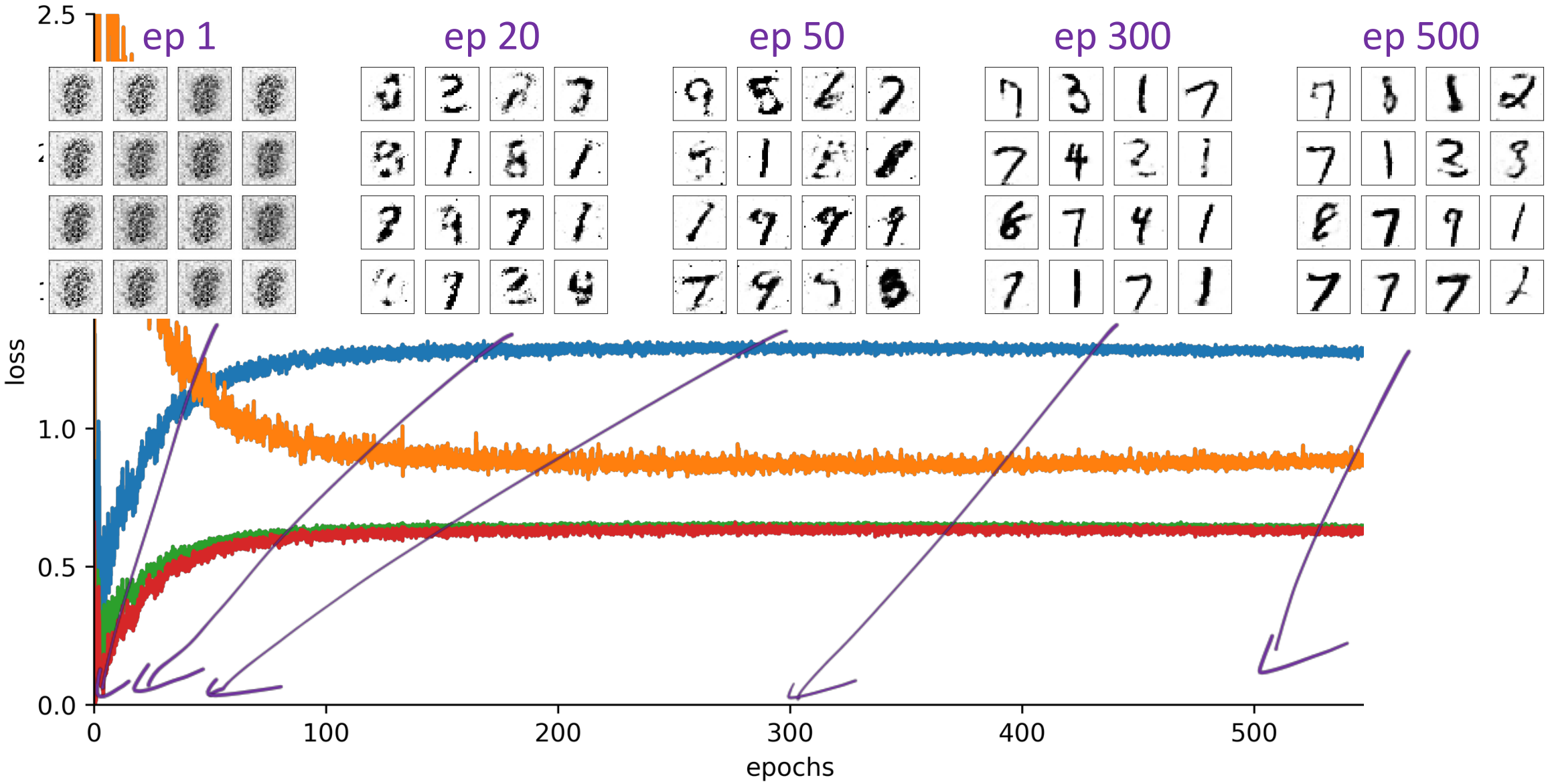
Running example: MNIST



*All objectives are negative of their original form

Code adapted from: https://github.com/prcastro/pytorch-gan/tree/master

Running example: MNIST

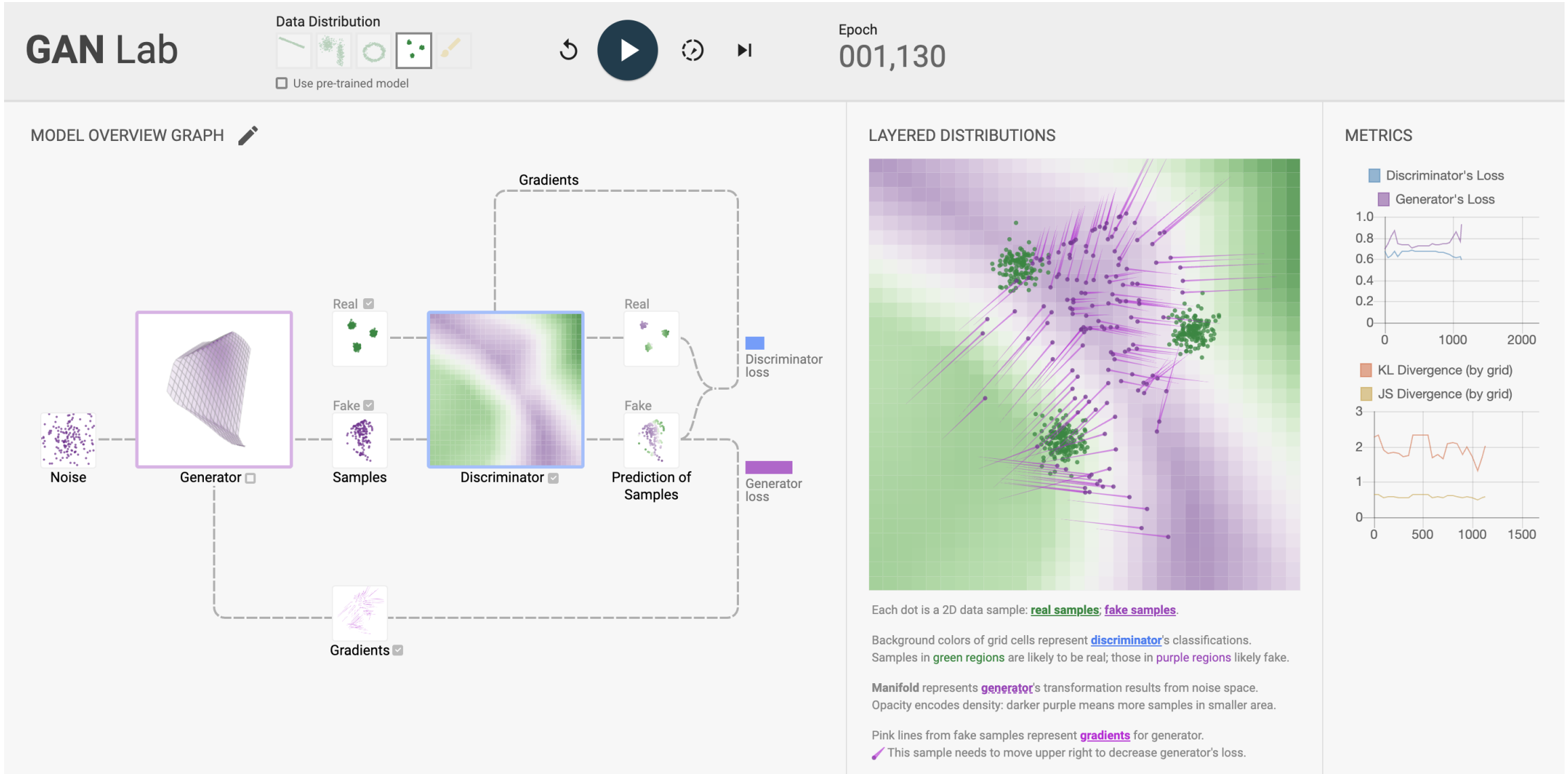


*All objectives are negative of their original form

Code adapted from: https://github.com/prcastro/pytorch-gan/tree/master

Running example: GAN Lab

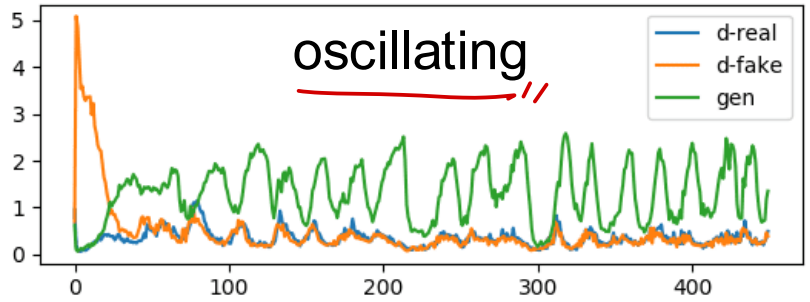
https://poloclub.github.io/ganlab/



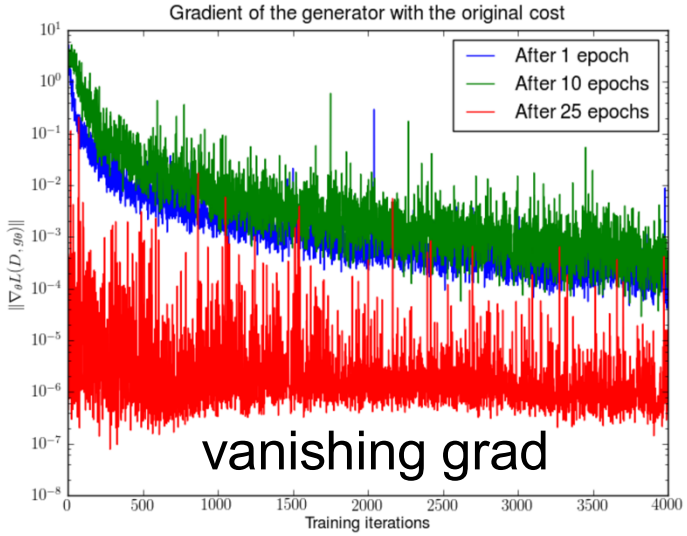
Issues of GAN

Difficult to train/converge

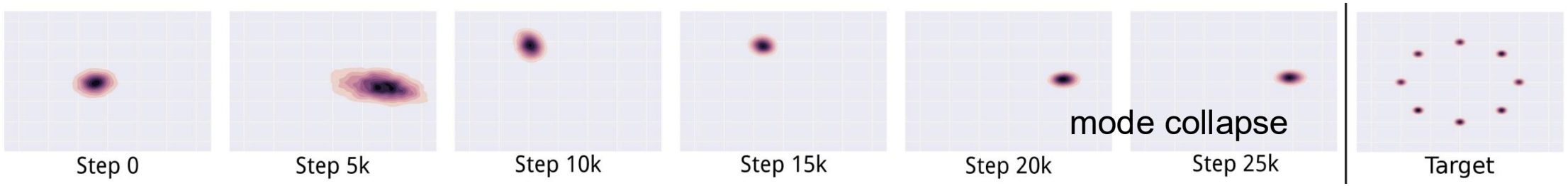
- Hard to achieve equilibrium
  min max G D → step마다 loss 진동
- Vanishing gradients
  if D is powerful. D(G(z)) ≈ 0
- Mode collapse
  입력 pattern 상관.



J. Brownlee, "How to Identify and Diagnose GAN Failure Modes"



Arjovsky & Bottou, "Towards Principled Methods for Training GANs"



L. Metz, "Unrolled Generative Adversarial Networks"

Adversary as a Loss Function

(D)

- GAN essentially defines an **adversarial loss** function

$L_{adv} = -\log D(G(\text{input}))$

- Input to networks is **not** necessarily random/noise

GAN: $z \rightarrow G(z)$ (z: noise) , z 대신 자연상의 image, 손상된 image 등.

- **Beyond L2/L1**: adversarial loss encourages output to look "realistic"

수치적 평가

- **Combined with L2/L1**: reconstruction loss largely stabilizes training

ex. $L_{total} = L_{reconstruc} + \lambda L_{adversarial}$

Example: Super-Resolution GAN

- **L2 loss**: better PSNR
- **Adv loss**: better visual quality

original

bicubic
(21.59dB/0.6423)

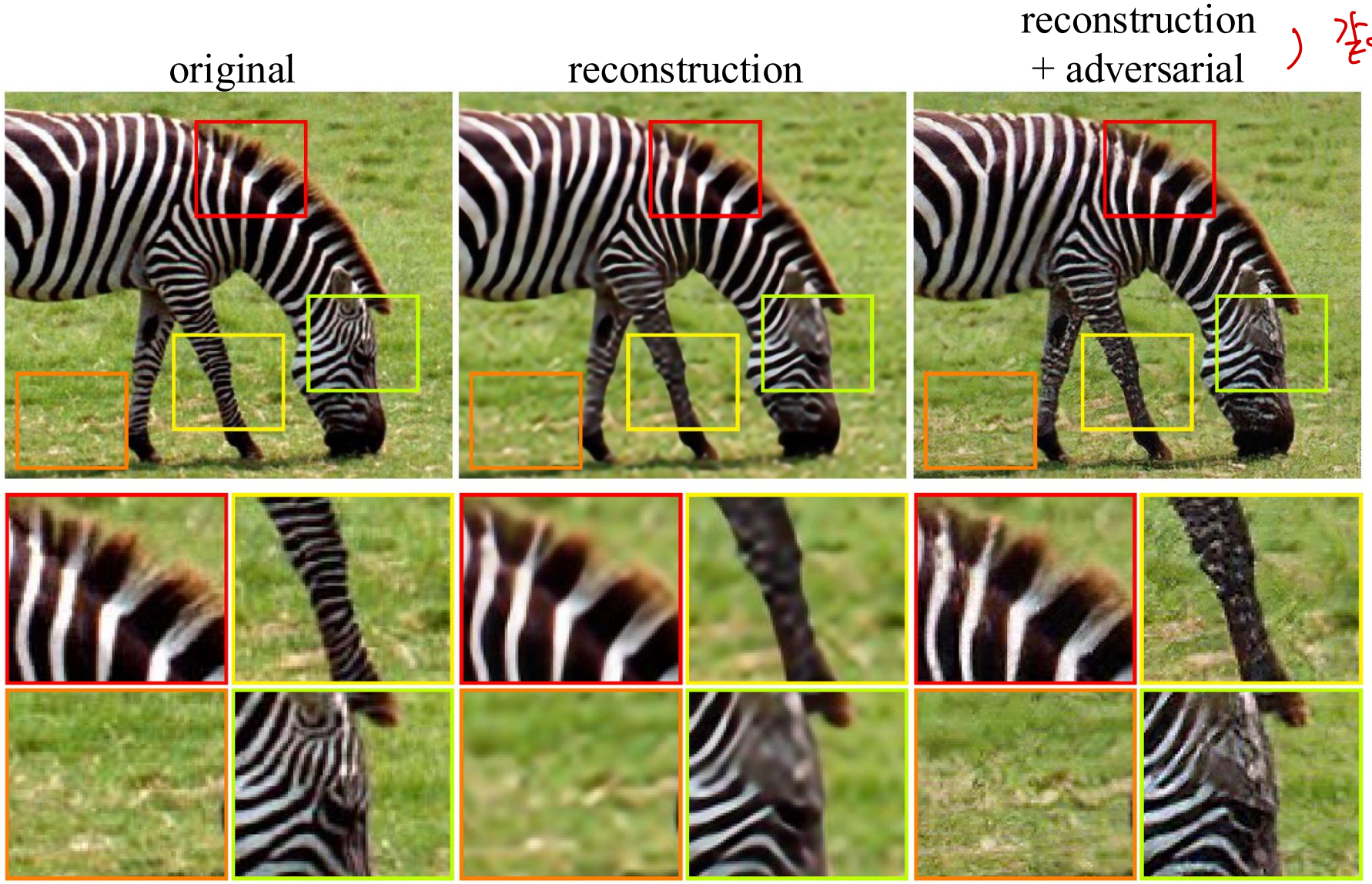
SRResNet
(23.44dB/0.7777)

SRGAN
(20.34dB/0.6562)



Ledig, et al., "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network", CVPR 2016

Example: Super-Resolution GAN



Ledig, et al., “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network”, CVPR 2016

Example: VQ-VAE to VQ-GAN

- Today's most popular image **tokenizer** for Diffusion or AR

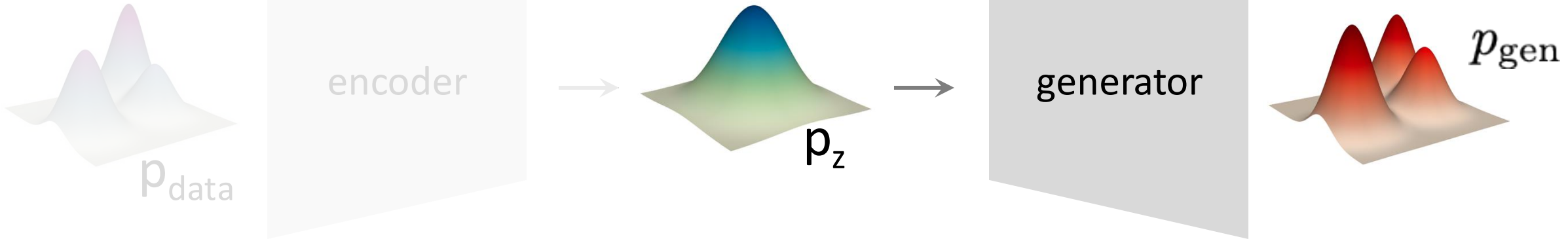
VQ-VAE (L2) **VQ-GAN (L2 + adv)**



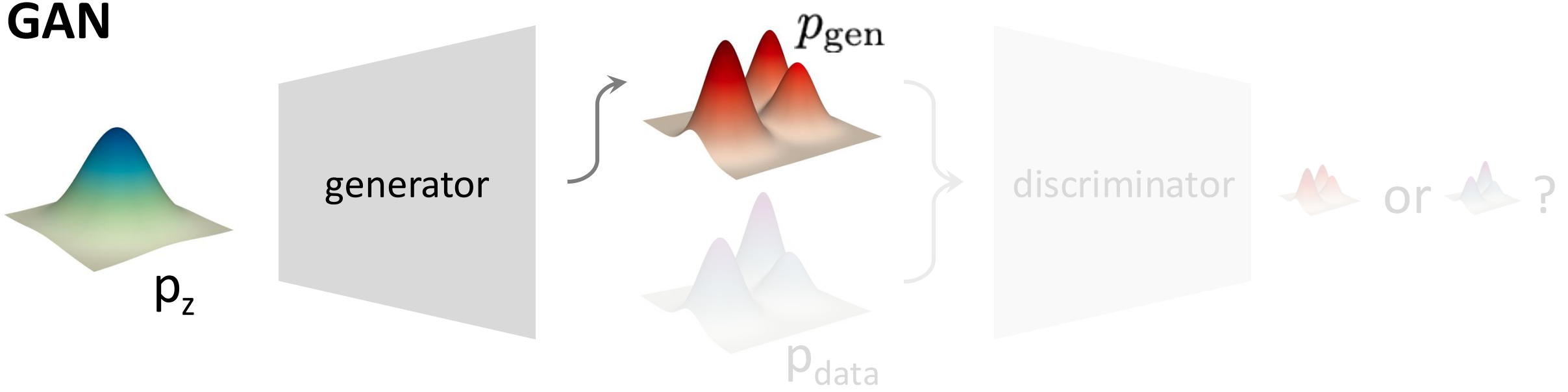
Esser, et al., "Taming Transformers for High-Resolution Image Synthesis", CVPR 2021

VAE

- VAE and GAN: **one-step** generation
- Diffusion, AR: **multi-step** generation (next lecture)



GAN



This Lecture

- Latent Variable Models
- Variational Autoencoder (VAE)
- Generative Adversarial Network (GAN)