Deep Learning Basics

Lecture 10: Generative Models Part 2

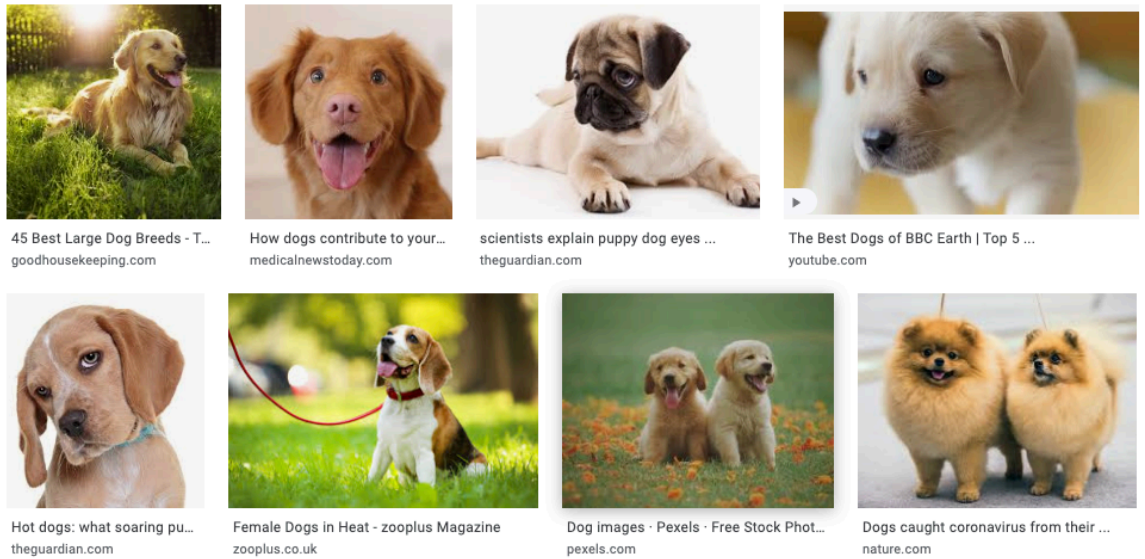
최성준 (고려대학교 인공지능학과)

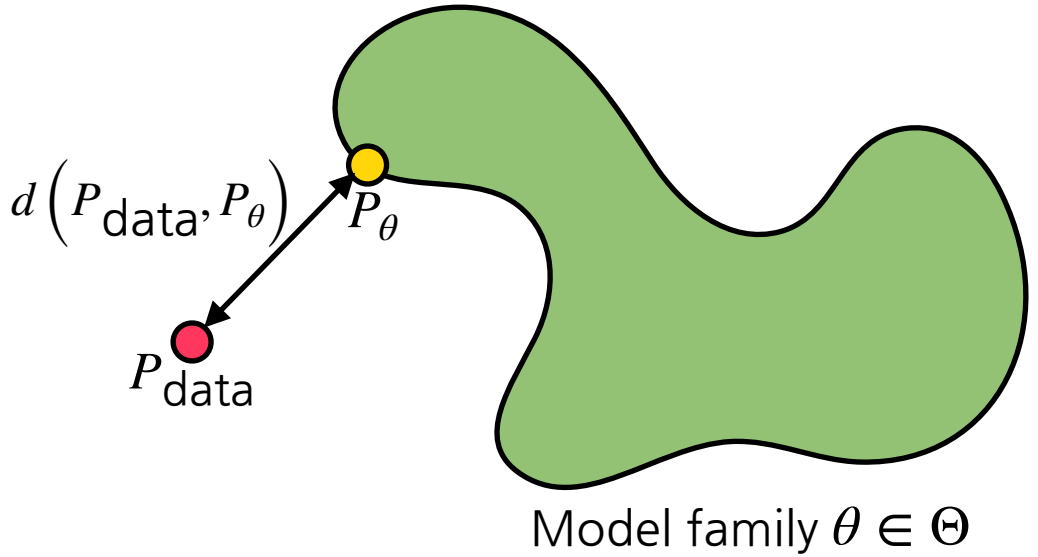
Contents

- Maximum Likelihood Learning
- Latent Variable Models
- Generative Adversarial Networks
- Diffusion Models

Maximum Likelihood Learning

Maximum Likelihood Learning





- Given a training set of examples, we can cast the generative model learning process as finding the best-approximating density model from the model family.
- Then, how can we evaluate the goodness of the approximation?

Maximum Likelihood Learning

- KL-divergence

$$\mathbf{D}(P_{\text{data}}||P_\theta) = \mathbf{E}_{\mathbf{x}\sim P_{\text{data}}}\left[\log\left(\frac{P_{\text{data}}(\mathbf{x})}{P_\theta(\mathbf{x})}\right)\right] = \sum_{\mathbf{x}} P_{\text{data}}(\mathbf{x}) \log \frac{P_{\text{data}}(\mathbf{x})}{P_\theta(\mathbf{x})}$$

- We can simplify this with

$$\begin{aligned}\mathbf{D}(P_{\text{data}}||P_\theta) &= \mathbf{E}_{\mathbf{x}\sim P_{\text{data}}}\left[\log\left(\frac{P_{\text{data}}(\mathbf{x})}{P_\theta(\mathbf{x})}\right)\right] \\ &= \mathbf{E}_{\mathbf{x}\sim P_{\text{data}}}\left[\log P_{\text{data}}(\mathbf{x})\right] - \mathbf{E}_{\mathbf{x}\sim P_{\text{data}}}\left[\log P_\theta(\mathbf{x})\right]\end{aligned}$$

- As the first term does not depend on P_θ, minimizing the KL-divergence is equivalent to maximizing the expected log-likelihood.

$$\arg\min_{P_\theta} \mathbf{D}(P_{\text{data}}||P_\theta) = \arg\min_{P_\theta} -\mathbf{E}_{\mathbf{x}\sim P_{\text{data}}}\left[\log P_\theta(\mathbf{x})\right] = \arg\max_{P_\theta} \mathbf{E}_{\mathbf{x}\sim P_{\text{data}}}\left[\log P_\theta(\mathbf{x})\right]$$

Maximum Likelihood Learning

- Approximate the expected log-likelihood

$$\mathbf{E}_{\mathbf{x}\sim P_{\text{data}}}\left[\log P_\theta(\mathbf{x})\right]$$

  with the empirical log-likelihood

$$\mathbf{E}_{\mathcal{D}}\left[\log P_\theta(\mathbf{x})\right] = \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \log P_\theta(\mathbf{x})$$

- Maximum likelihood learning is then:

$$\max_{P_\theta} \frac{1}{|\mathcal{D}|}\sum_{\mathbf{x}\in\mathcal{D}} \log P_\theta(\mathbf{x})$$

- Variance of Monte Carlo estimate is high:

$$V_P[\hat{g}] = V_P\left[\frac{1}{T}\sum_{t=1}^{T} g(x^t)\right] = \frac{V_P[g(x)]}{T}$$

Empirical Risk Minimization

- For maximum likelihood learning, **empirical risk minimization (ERM)** if often used.
- However, **ERM** often suffers from its overfitting.
  - Extreme case: The model remembers all training data

$$p(x) = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \delta(x, x_i).$$

- To achieve better generalization, we typically restrict the **hypothesis space** of distributions that we search over.
- However, it could deteriorate the performance of the generative model.

Maximum Likelihood Learning

- Usually, MLL is prone to under-fitting as we often use simple parametric distributions such as spherical Gaussians.

- What about other ways of measuring the similarity?

Maximum Likelihood Learning

- Usually, MLL is prone to under-fitting as we often use simple parametric distributions such as spherical Gaussians.

- What about other ways of measuring the similarity?
  - **KL-divergence** leads to maximum likelihood learning or Variational Autoencoder (VAE).
  - **Jensen-Shannon divergence** leads to Generative Adversarial Network (GAN).
  - **Wasserstein distance** leads to Wasserstein Autoencoder (WAE) or Adversarial Autoencoder (AAE).

Latent Variable Models

D. Kingma, "Variational Inference and Deep Learning: A New Synthesis," Ph.D. Thesis

Quick Question

Is an **autoencoder** a generative model?

Variational Autoencoder

- The objective is simple.

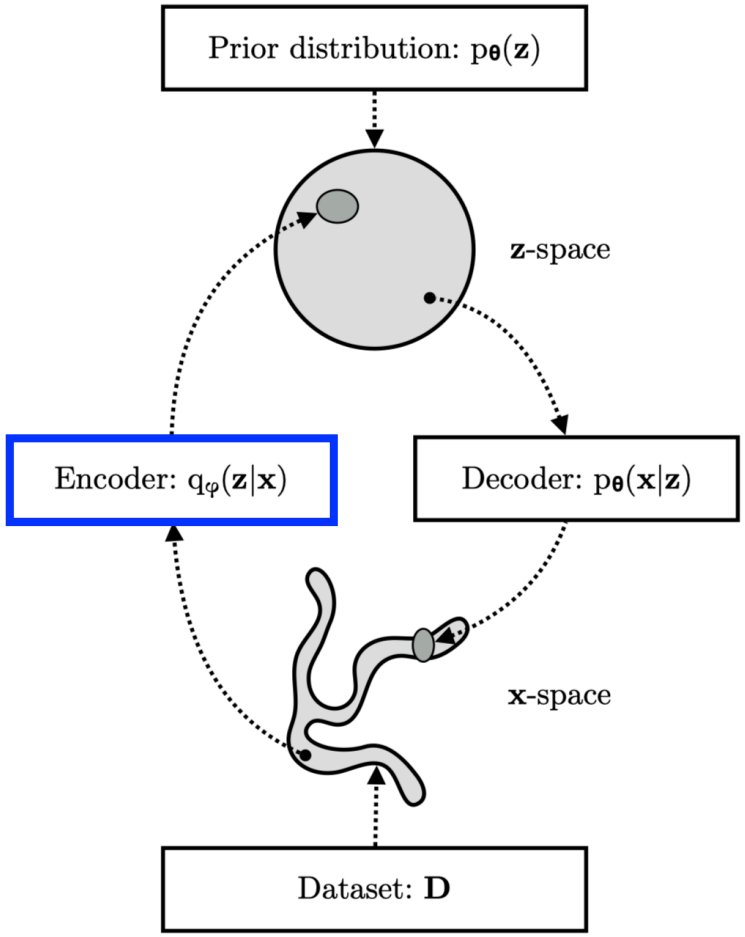
$$\text{Maximize } p_\theta(\mathbf{x})$$

Variational Autoencoder

- Variational inference (VI)
  - The goal of VI is to optimize the **variational distribution** that best matches the **posterior distribution**.
    - **Posterior distribution**: $p_\theta(z \mid x)$
    - **Variational distribution**: $q_\phi(z \mid x)$
  - In particular, we want to find the **variational distribution** that minimizes the KL divergence between the true posterior.

Variational Autoencoder

Our goal is to learn the **variational distribution** (aka **Encoder**)



Variational Autoencoder

$$
\begin{aligned}
\underbrace{\log p_\theta(x)}_{\text{Maximum Likelihood Learning} \uparrow} &= \int_z \underbrace{q_\phi(z|x)}_{\text{For any } q_\phi} \log p_\theta(x) dz \\
&= \mathbb{E}_{z\sim q_\phi(z|x)}\left[\log \frac{p_\theta(x)p_\theta(z|x)}{p_\theta(z|x)}\right] \\
&= \mathbb{E}_{z\sim q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)}{p_\theta(z|x)}\right] \\
&= \mathbb{E}_{z\sim q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}\right] \\
&= \mathbb{E}_{z\sim q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)}{q_\phi(z|x)}\right] + \mathbb{E}_{z\sim q_\phi(z|x)}\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right] \\
&= \underbrace{\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)}{q_\phi(z|x)}\right]}_{\text{ELBO} \uparrow} + \underbrace{D_{KL}\left(q_\phi(z|x)\|p_\theta(z|x)\right)}_{\text{Variational Gap} \downarrow}
\end{aligned}
$$

Variational Autoencoder

$$\underbrace{\log p_\theta(x)}_{\text{Maximum Likelihood Learning} \uparrow} = \int_z \underbrace{q_\phi(z|x)}_{\text{For any } q_\phi} \log p_\theta(x) dz$$

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

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

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

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

$$= \underbrace{\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)}{q_\phi(z|x)}\right]}_{\text{ELBO} \uparrow} + \underbrace{D_{KL}\left(q_\phi(z|x)\|p_\theta(z|x)\right)}_{\text{Variational Gap} \downarrow}$$

Importantly, **ELBO** is a **tractable** quantity.

VI minimizes the (intractable) **objective** via maximizing **ELBO**.

Evidence Lower Bound

$$\underbrace{\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)}{q_\phi(z|x)}\right]}_{\text{ELBO}\uparrow} = \int \log \frac{p_\theta(x|z)p(z)}{q_\phi(z|x)} q_\phi(z|x)dz$$

$$= \underbrace{\mathbb{E}_{q_\phi(z|x)}\left[p_\theta(x|z)\right]}_{\text{Reconstruction Term}} - \underbrace{D_{KL}\left(q_\phi(z|x)||p(z)\right)}_{\text{Prior Fitting Term}}$$

Evidence Lower Bound

$$\underbrace{\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)}{q_\phi(z|x)}\right]}_{\text{ELBO}\uparrow} = \int \log \frac{p_\theta(x|z)p(z)}{q_\phi(z|x)} q_\phi(z|x)dz$$

$$= \underbrace{\mathbb{E}_{q_\phi(z|x)}\left[p_\theta(x|z)\right]}_{\text{Reconstruction Term}} - \underbrace{D_{KL}\left(q_\phi(z|x)||p(z)\right)}_{\text{Prior Fitting Term}}$$

This term minimizes the **reconstruction loss** of an auto-encoder.

This term enforces the latent distribution to be similar to the **prior distribution**.

Evidence Lower Bound

$$\underbrace{\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)}{q_\phi(z|x)}\right]}_{\text{ELBO}\uparrow} = \int \log \frac{p_\theta(x|z)p(z)}{q_\phi(z|x)} q_\phi(z|x)dz$$

$$= \underbrace{\mathbb{E}_{q_\phi(z|x)}\left[p_\theta(x|z)\right]}_{\text{Reconstruction Term}} - \underbrace{D_{KL}\left(q_\phi(z|x)\|p(z)\right)}_{\text{Prior Fitting Term}}$$

- Key Limitation
  - It is an **intractable** model (hard to evaluate likelihood).
  - The prior fitting term should be differentiable, hence it is hard to use diverse latent prior distributions.
  - In most cases, we use an isotropic Gaussian where we have a closed-form for the prior fitting term.

$$\underbrace{D_{KL}\left(q_\phi(z|x)\|\mathcal{N}(0,I)\right)}_{\text{Prior Fitting Term}} = \frac{1}{2}\sum_{i=1}^{D}\left(\sigma_{z_i}^2 + \mu_{z_i}^2 - \log(\sigma_{z_i}^2) - 1\right)$$

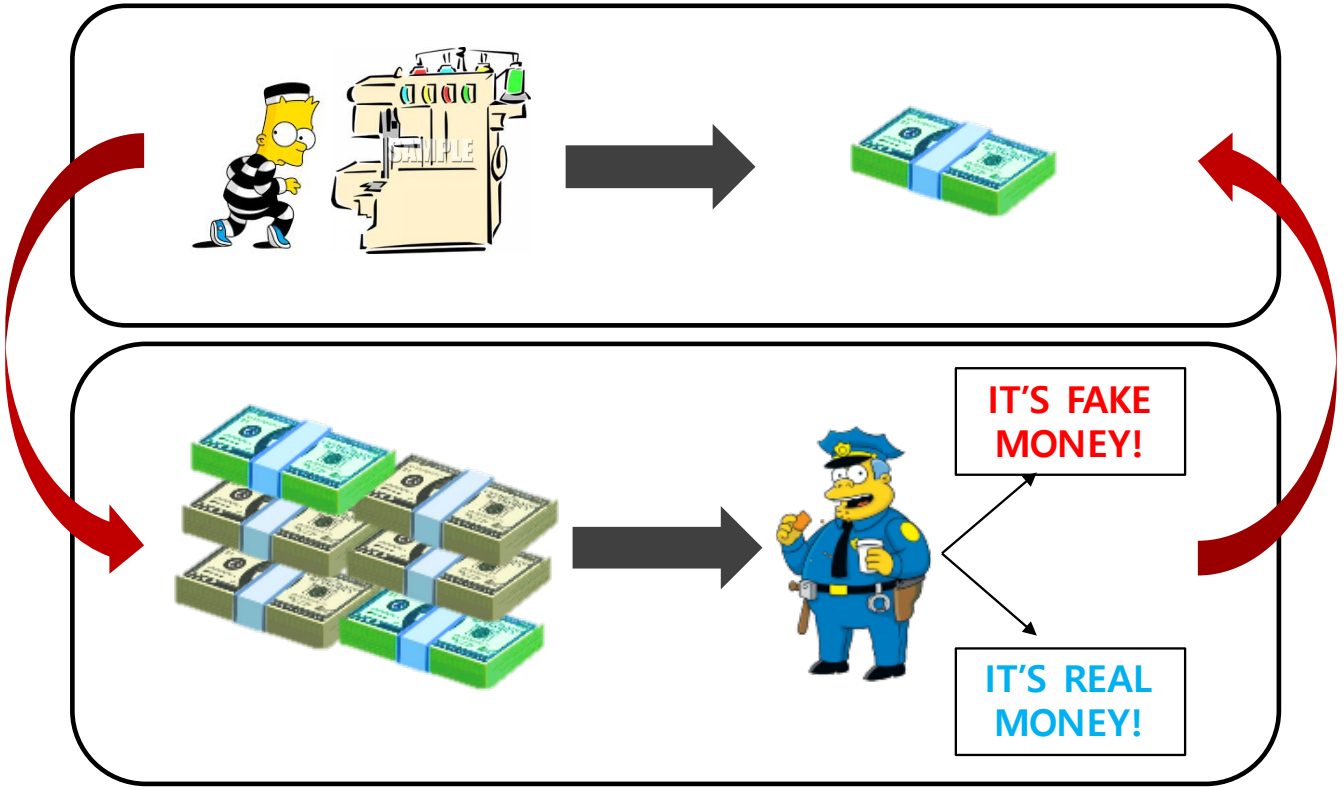
Variational Autoencoder

D. Kingma, "Variational Inference and Deep Learning: A New Synthesis," Ph.D. Thesis

Generative Adversarial Networks

Goodfellow et al., "Generative Adversarial Networks", NIPS, 2014

Generative Adversarial Networks



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

GAN Objective

- GAN is a two player minimax game between **generator** and **discriminator**.
  - **Discriminator** objective

$$\max_D V(G, D) = E_{\mathbf{x} \sim p_{\text{data}}}[\log D(\mathbf{x})] + E_{\mathbf{x} \sim p_G}[\log(1 - D(\mathbf{x}))]$$

  - The **optimal discriminator** is

$$D_G^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})}$$

GAN Objective

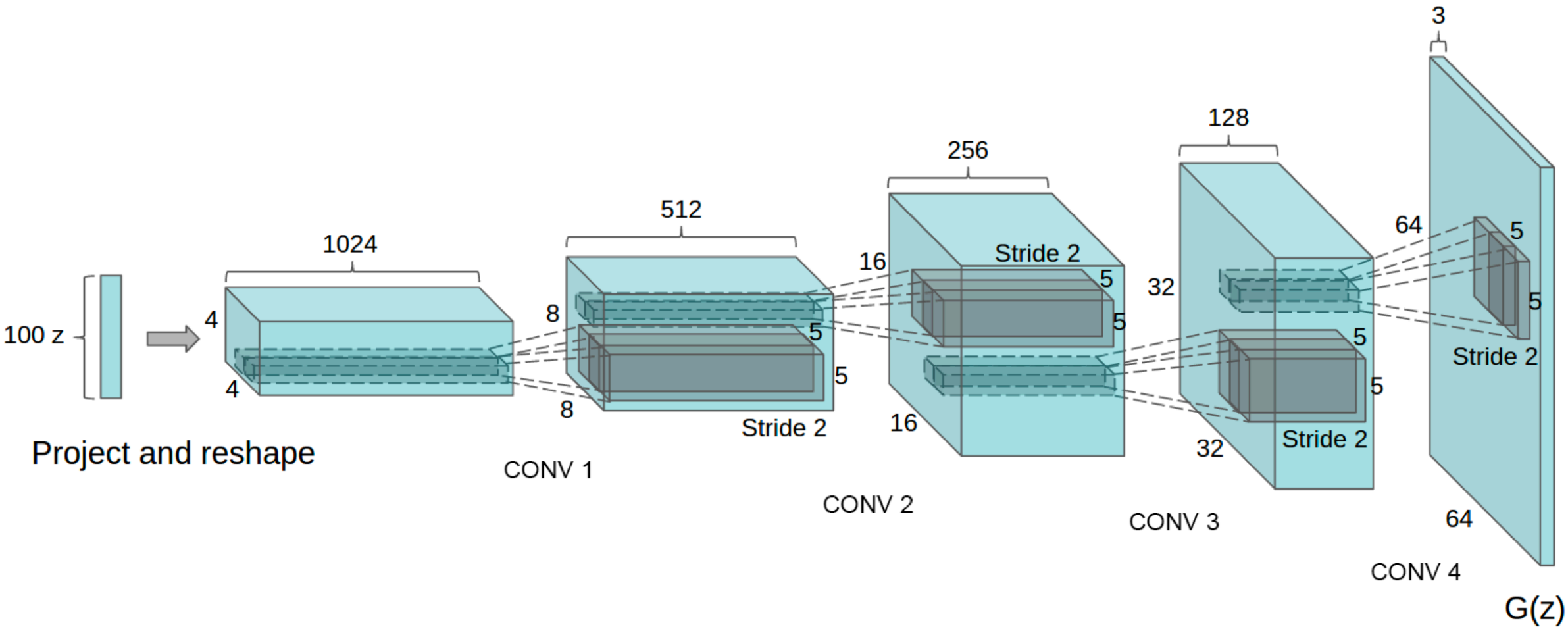
- GAN is a two player minimax game between **generator** and **discriminator**.
  - **Generator** objective

$$\min_G V(G, D) = E_{\mathbf{x} \sim p_{\text{data}}}[\log D(\mathbf{x})] + E_{\mathbf{x} \sim p_G}[\log(1 - D(\mathbf{x}))]$$

  - Plugging in the **optimal discriminator**, we get

$$\begin{aligned} V(G, D_G^*(\mathbf{x})) &= E_{\mathbf{x} \sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})}\right] + E_{\mathbf{x} \sim p_G}\left[\log \frac{p_G(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})}\right] \\ &= E_{\mathbf{x} \sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\mathbf{x})}{\frac{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})}{2}}\right] + E_{\mathbf{x} \sim p_G}\left[\log \frac{p_G(\mathbf{x})}{\frac{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})}{2}}\right] - \log 4 \\ &= \underbrace{D_{KL}\left[p_{\text{data}}, \frac{p_{\text{data}} + p_G}{2}\right] + D_{KL}\left[p_G, \frac{p_{\text{data}} + p_G}{2}\right]}_{2 \times \text{Jenson-Shannon Divergence (JSD)}} - \log 4 \\ &= 2D_{JSD}[p_{\text{data}}, p_G] - \log 4 \end{aligned}$$

Deep Convolutional GAN



https://arxiv.org/abs/1511.06434

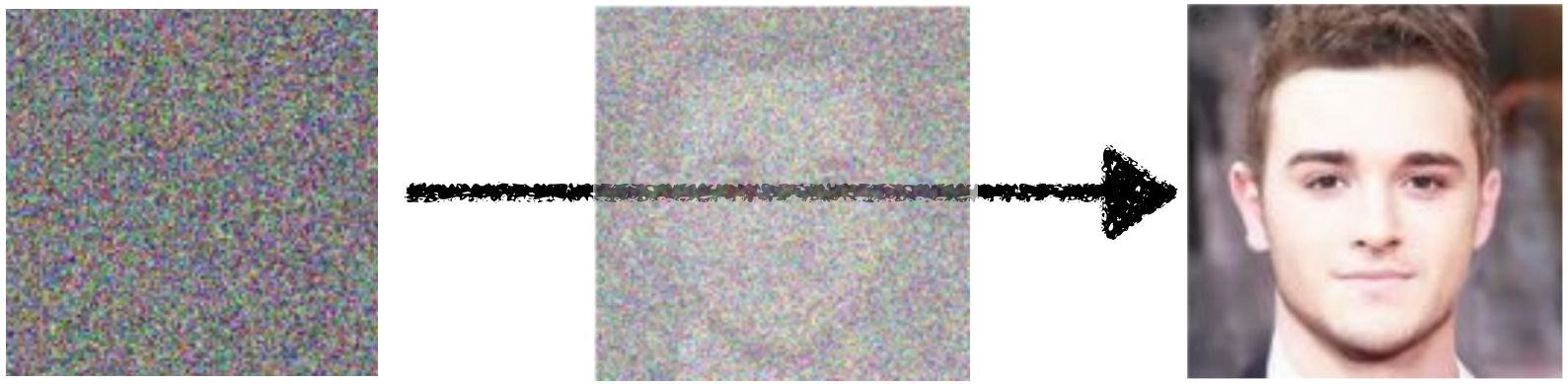
Diffusion Models

Ho et al., "Denoising Diffusion Probabilistic Models," 2020

Diffusion Models

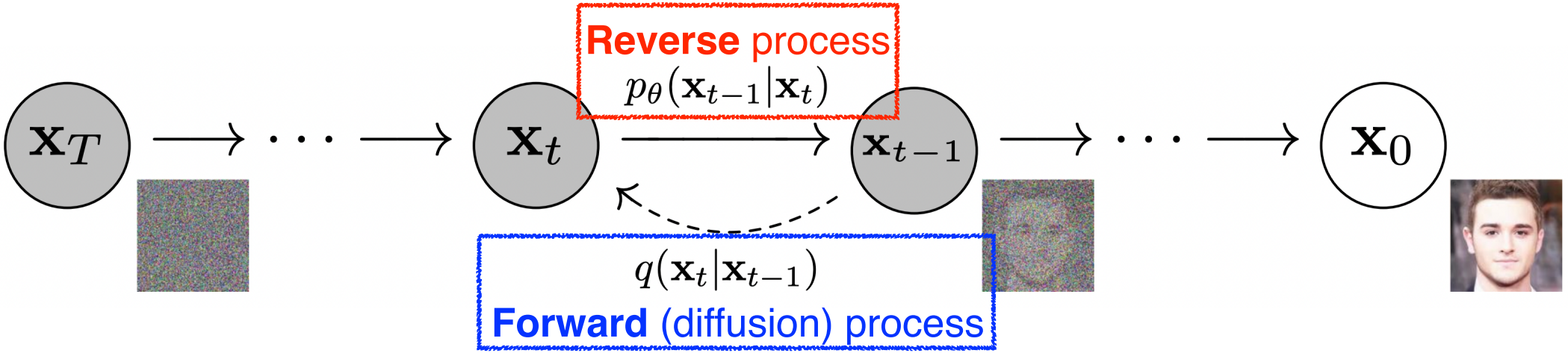
Diffusion models generate images from noise.

Diffusion Models



Diffusion models **progressively** generate images from noise.

Diffusion Models



- **Forward** (diffusion) process progressively injects noise to an image.

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \prod_{t=1}^{T} \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$$

- The **reverse** process is learned in such a way to denoise the perturbed image back to a clean image.

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

Diffusion Models

https://github.com/sjchoi86/2022-1-deep-learning-applications/

DALL-E2

vibrant portrait painting of Salvador Dalí with a robotic half face

a shiba inu wearing a beret and black turtleneck

a close up of a handpalm with leaves growing from it

an espresso machine that makes coffee from human souls, artstation

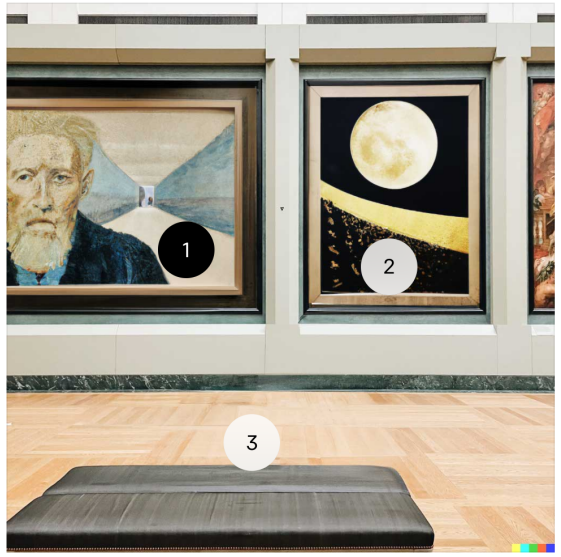
panda mad scientist mixing sparkling chemicals, artstation

a corgi's head depicted as an explosion of a nebula

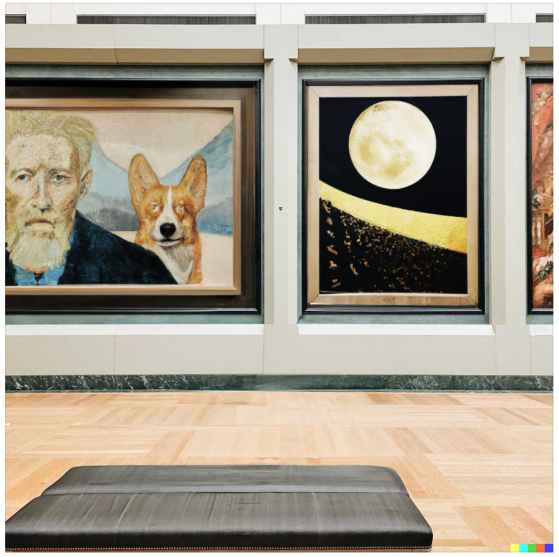
https://openai.com/dall-e-2

DALL-E2

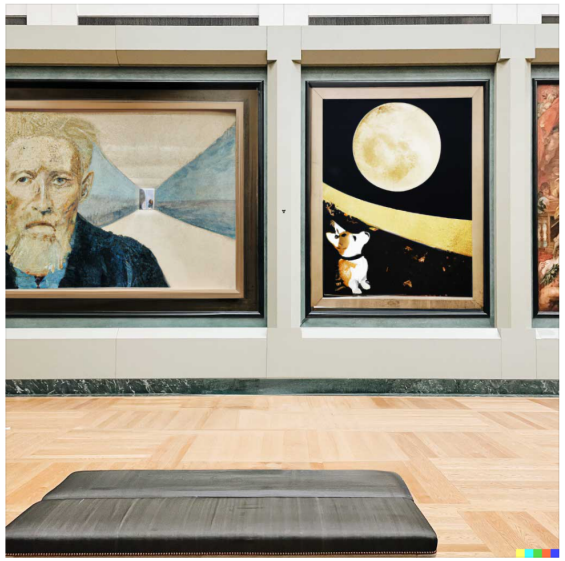
ORIGINAL IMAGE



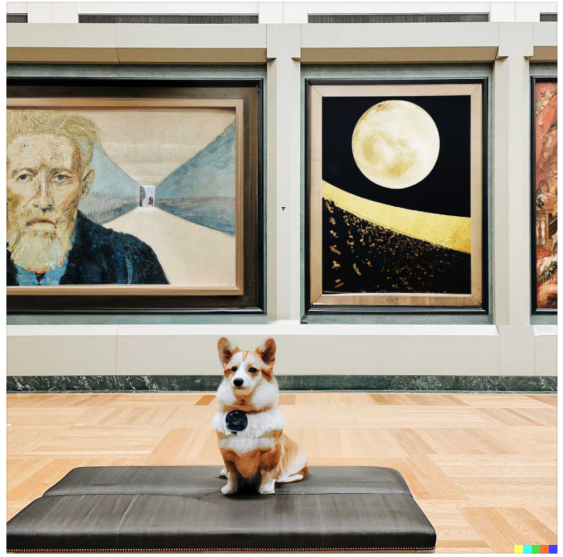
DALL·E 2 EDITS



DALL·E 2 EDITS



DALL·E 2 EDITS



https://openai.com/dall-e-2

Thank you for listening