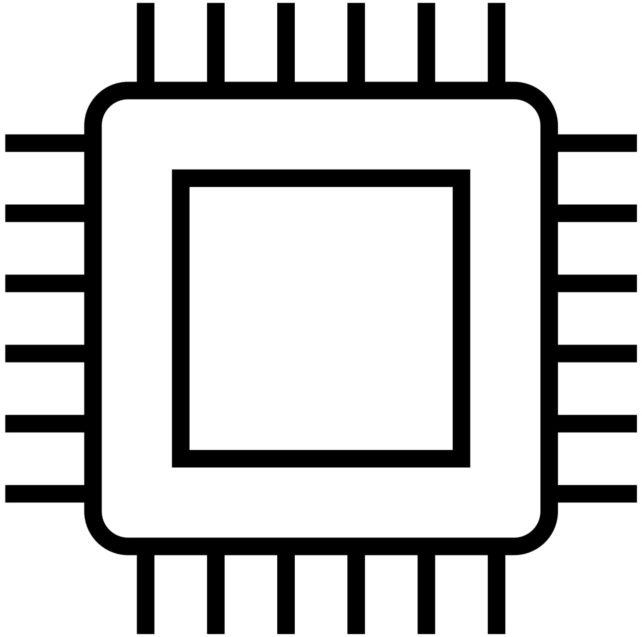
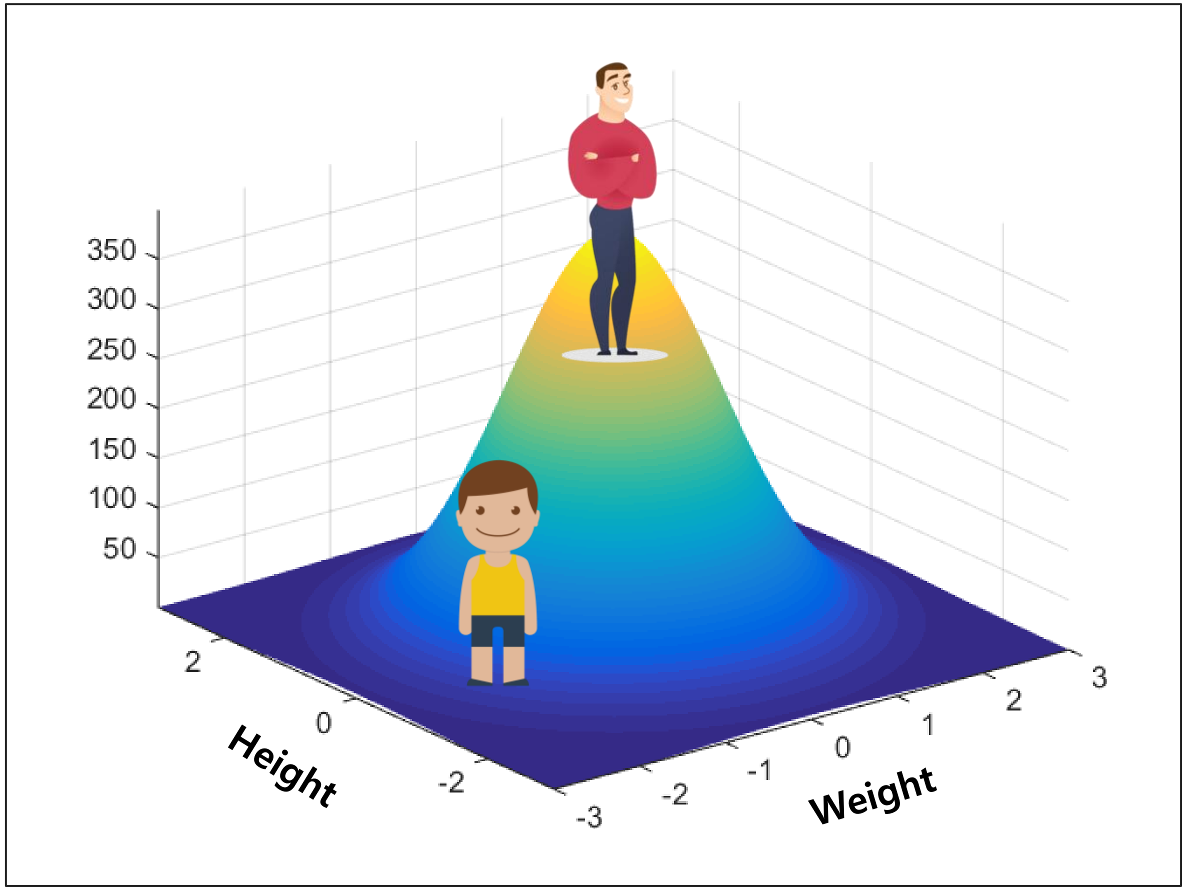
Generative Models

2025.07.10.



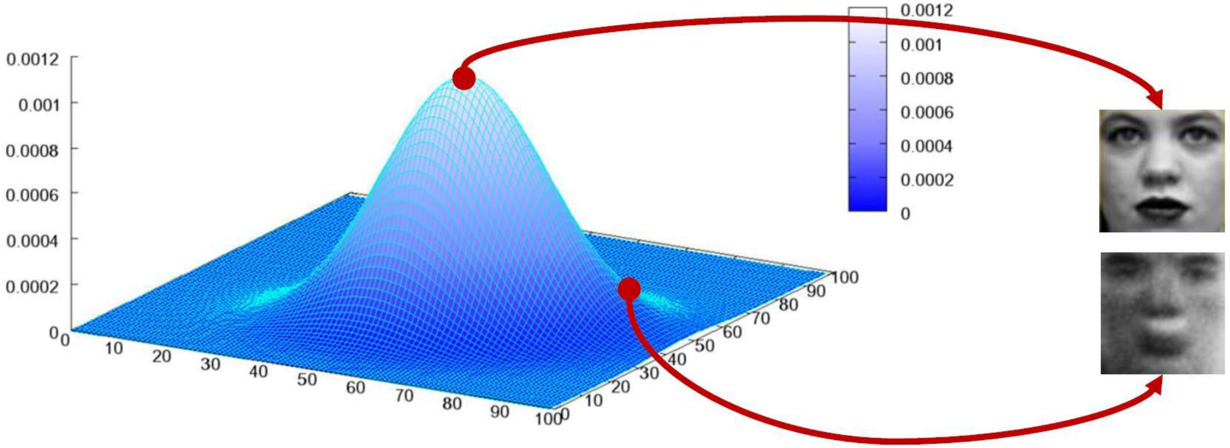
Generative Models [1]

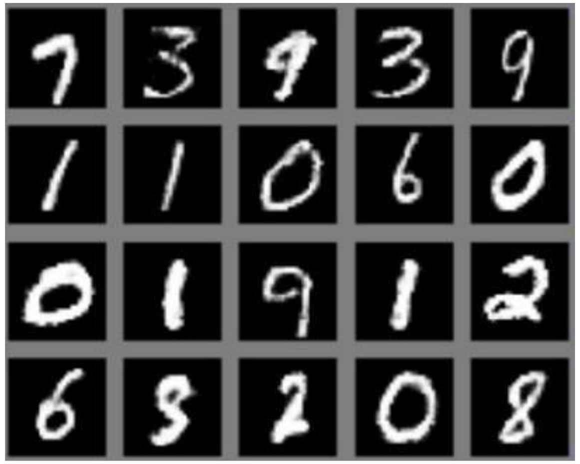
- **Generative Models**
  - Generative model aims to capture probability distribution of data, $\boldsymbol{P_{data}(x)}$
  - We assume that data comes from $\boldsymbol{P_{data}(x)}$

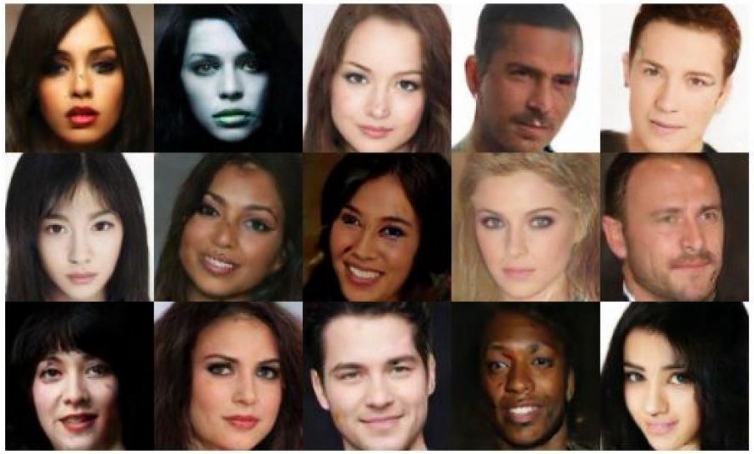


Generative Models [2]

- **Generative Models**
  - Probability distribution of images?
  - **Very complex!**

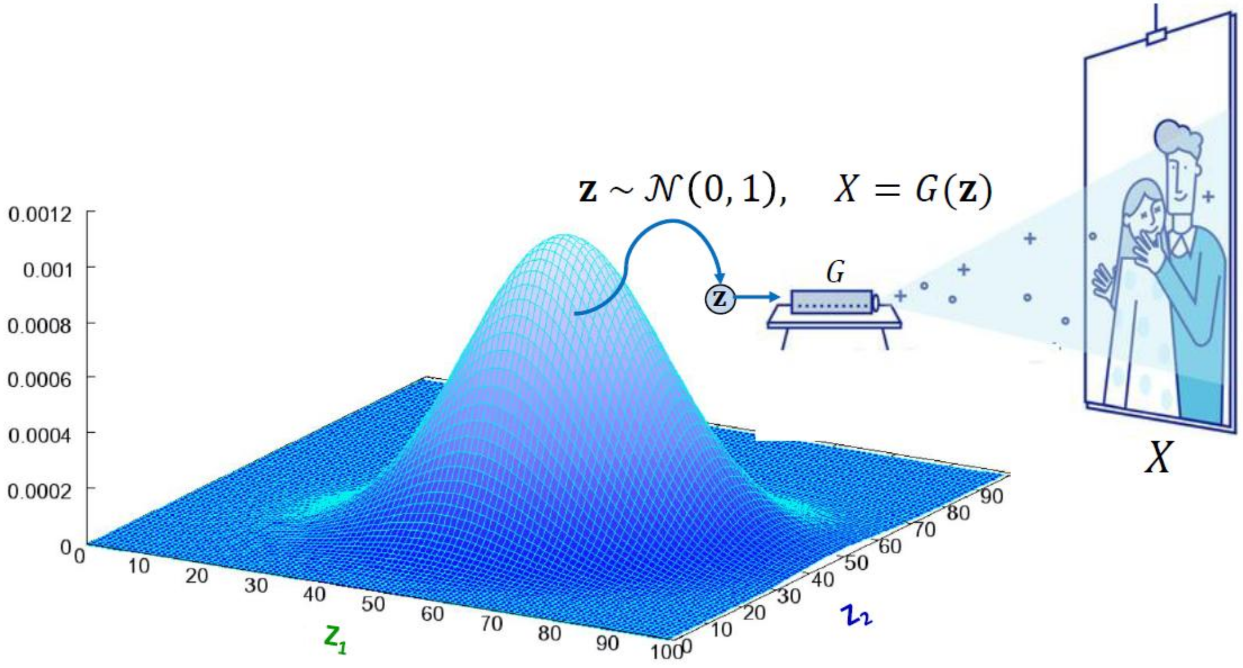






Generative Models [3]

- **Generative Models**
  - What about making a simple distribution, and then projecting to a real-world complex distribution?
    - ▶ Making a simple distribution (e.g., standard normal) is easy!
    - ▶ Sample a **latent variable *z*** and then transform it to an image!



z = np.random.normal (0, 1, size=[num_images, latent_dim])

Generative Models [4]

- **Generative Models**
  - Observed data ***x*** is originated from latent variables ***z***

Myth of Cave

Generative Models [5]

- **Variational Autoencoder (VAE)**
  - Assuming that training data $\boldsymbol{x}$ is originated from underlying (unobserved) latent variables $\boldsymbol{z}$
  - It defines intractable density function with latent **z**:

$$\boldsymbol{P_\theta(x) = \int P_\theta(x|z)\, P_\theta(z) dz}$$

  - We cannot optimize it directly because it is **intractable**!
  - Instead, we will derive and maximize a **lower bound** on the **likelihood**

$$= \boxed{\mathrm{E_z}[\log P_\theta(x)] - D_{KL}\left(q_\phi(z|x)||P_\theta(z)\right)} + \boxed{D_{KL}\left(q_\phi(z|x)||P_\theta(z|x)\right)}$$

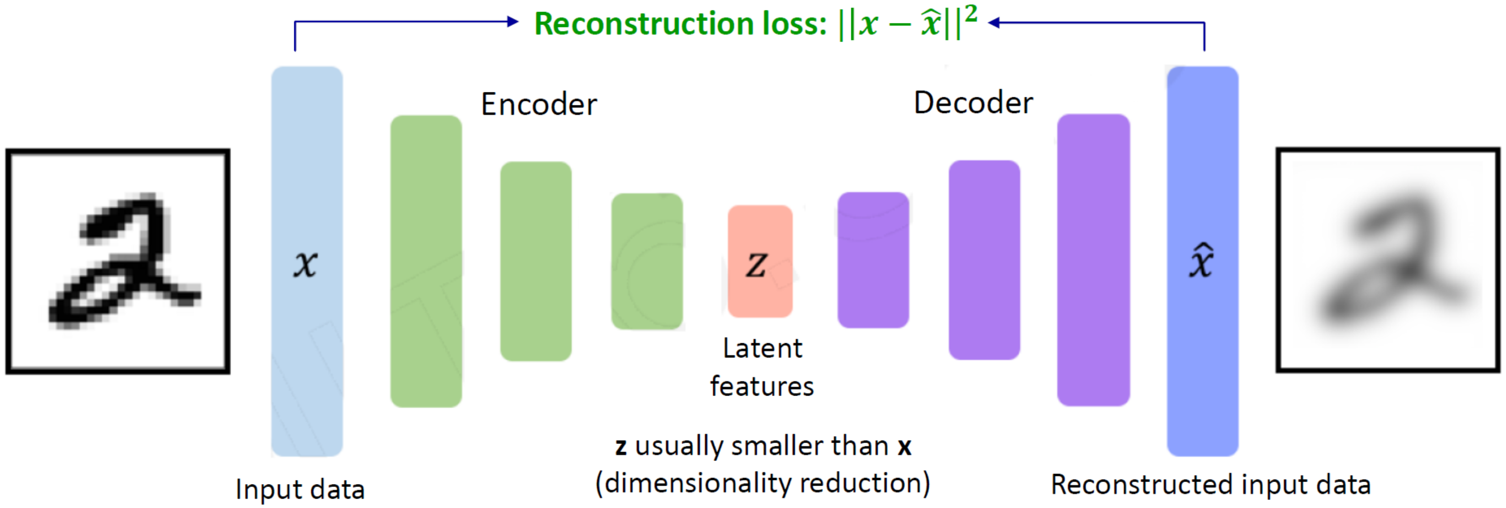
Tractable lower bound

Interactable
(KL-Divergence >= 0)

$$\geq \mathbf{E_z}[\mathbf{log}\boldsymbol{P_\theta(x)}] - \boldsymbol{D_{KL}\left(q_\phi(z|x)||P_\theta(z)\right)}$$

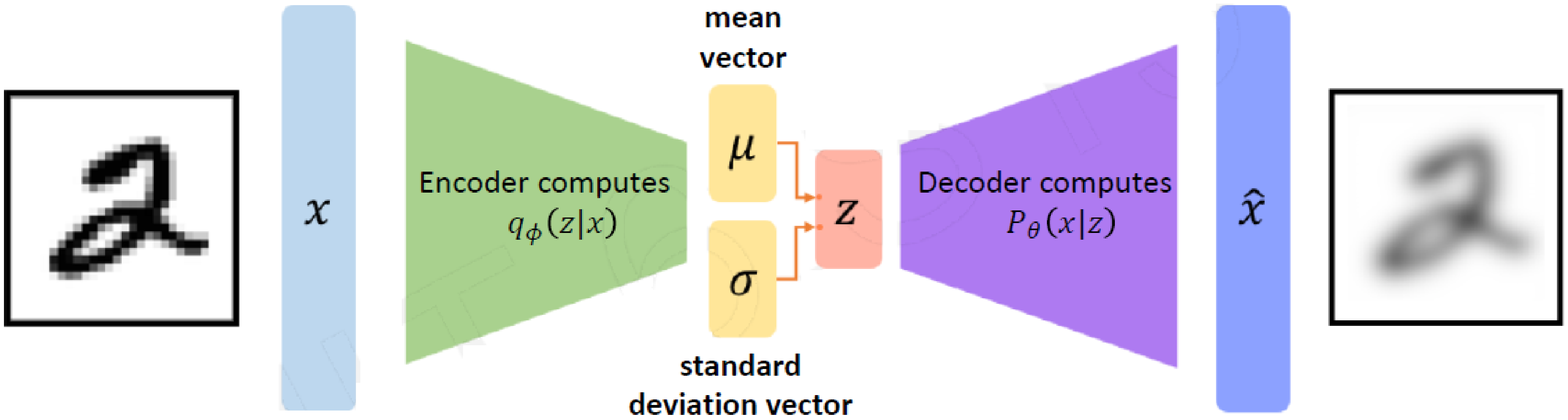
Generative Models [6]

- **Variational Autoencoder (VAE)**
  - Revisiting Autoencoder (AE)
    - ▶ Unsupervised approach for learning a **lower-dimensional feature** representation from training data (without explicit labels)
    - ▶ Train such that latent features can be used to **reconstruct** the original data ("Autoencoding" - encoding itself)
    - ▶ Goal is to learn the latent features ***z*** that capture (or encode) as much information about the data ***x*** as possible



Generative Models [7]

- **Variational Autoencoder (VAE)**
  - Assume training data $\boldsymbol{x}$ is generated from underlying (unobserved) **latent variables $\boldsymbol{z}$**
  - **Encoder network** models $q_{\phi}(z|x)$ and **decoder network** models $P_{\theta}(x|z)$



Sample z from $z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$

Generative Models [8]

- **Variational Autoencoder (VAE)**
  - Maximizing **log-likelihood** of training data $\boldsymbol{x}$: $\mathbf{log}\boldsymbol{P_\theta}(\boldsymbol{x})$

$$\log P_\theta(x) = \mathrm{E}_{z\sim q_\phi(z|x)}[\log P_\theta(x)]$$ (Taking expectation with regards to z)

$$= \mathrm{E}_z\left[\log \frac{P_\theta(x|z)P_\theta(z)}{P_\theta(z|x)}\right]$$ (Bayes' rule : $\frac{P(B|A)P(A)}{P(B)}$)

$$= \mathrm{E}_z\left[\log \frac{P_\theta(x|z)P_\theta(z)}{P_\theta(z|x)} \frac{q_\phi(z|x)}{q_\phi(z|x)}\right]$$ (Multiply & divide with the same term)

$$= \mathrm{E}_z[\log P_\theta(x|z)] - \mathrm{E}_z\left[\log \frac{q_\phi(z|x)}{P_\theta(z)}\right] + \mathrm{E}_z\left[\log \frac{q_\phi(z|x)}{P_\theta(z|x)}\right]$$

$$= \boxed{\mathrm{E}_\mathrm{z}[\log P_\theta(x|z)] - D_{KL}\left(q_\phi(z|x)||P_\theta(z)\right)} + \boxed{D_{KL}\left(q_\phi(z|x)||P_\theta(z|x)\right)}$$

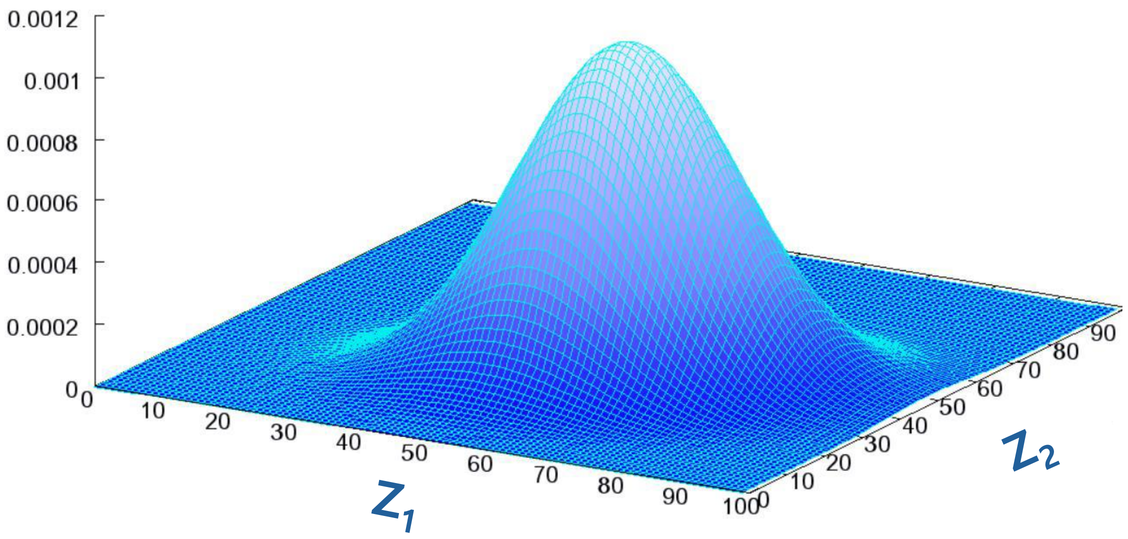
Tractable lower bound

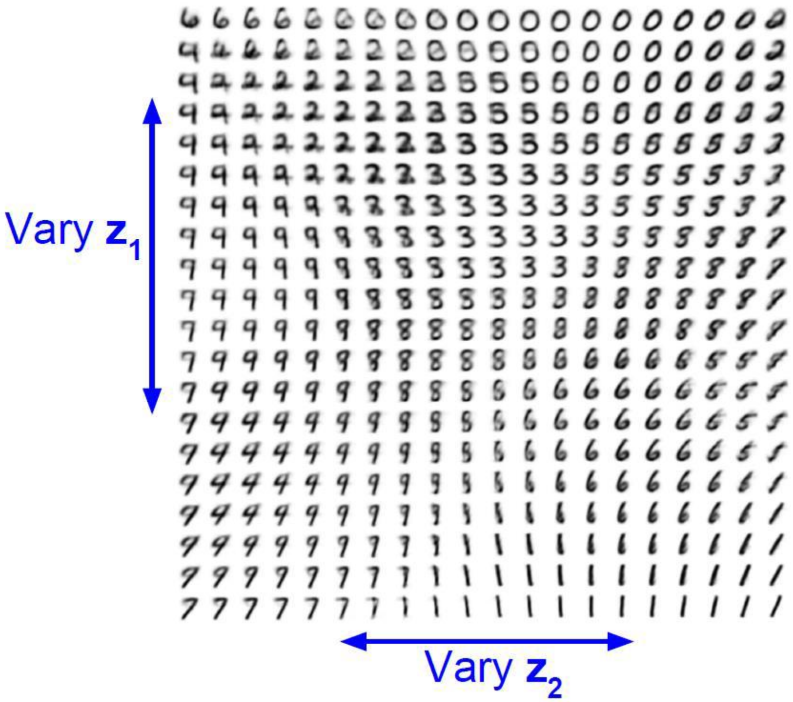
Interactable
(KL-Divergence >= 0)

$$\geq \mathbf{E}_\mathbf{z}[\mathbf{log}\boldsymbol{P_\theta}(\boldsymbol{x}|\boldsymbol{z})] - \boldsymbol{D_{KL}}\left(\boldsymbol{q_\phi}(\boldsymbol{z}|\boldsymbol{x})||\boldsymbol{P_\theta}(\boldsymbol{z})\right)$$

Generative Models [9]

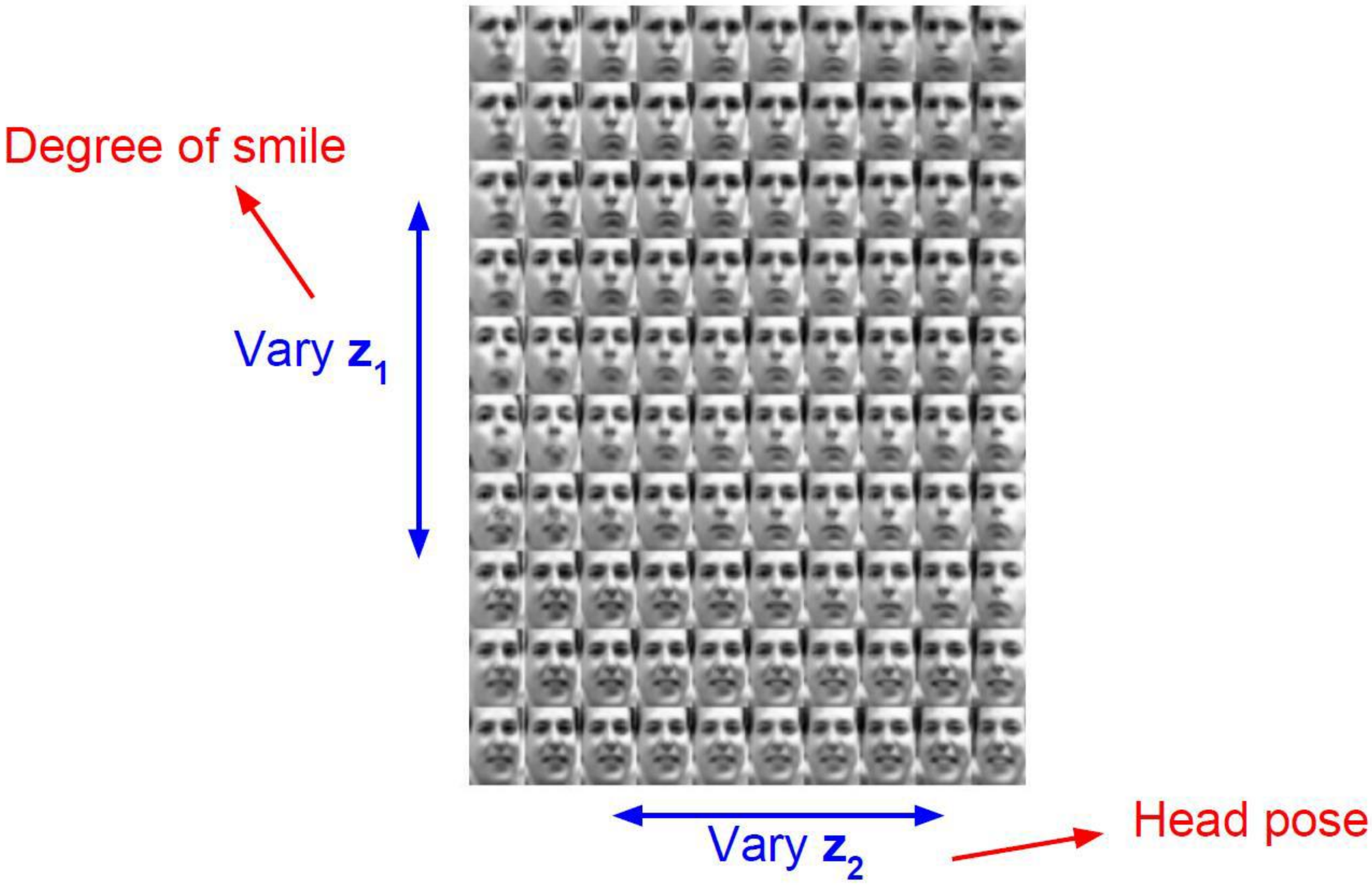
- **Variational Autoencoder (VAE)**
  - Generates samples with regarding to ***z***
  - Only decoder is used here, and ***z*** is produced and varied manually
  - Slowly increase or decrease a single latent variable while keeping all other variables fixed
  - Each number is smoothly transitioning to another number





Generative Models [10]

- **Variational Autoencoder (VAE)**



Generative Models [11]

- **Variational Autoencoder (VAE) 실습**

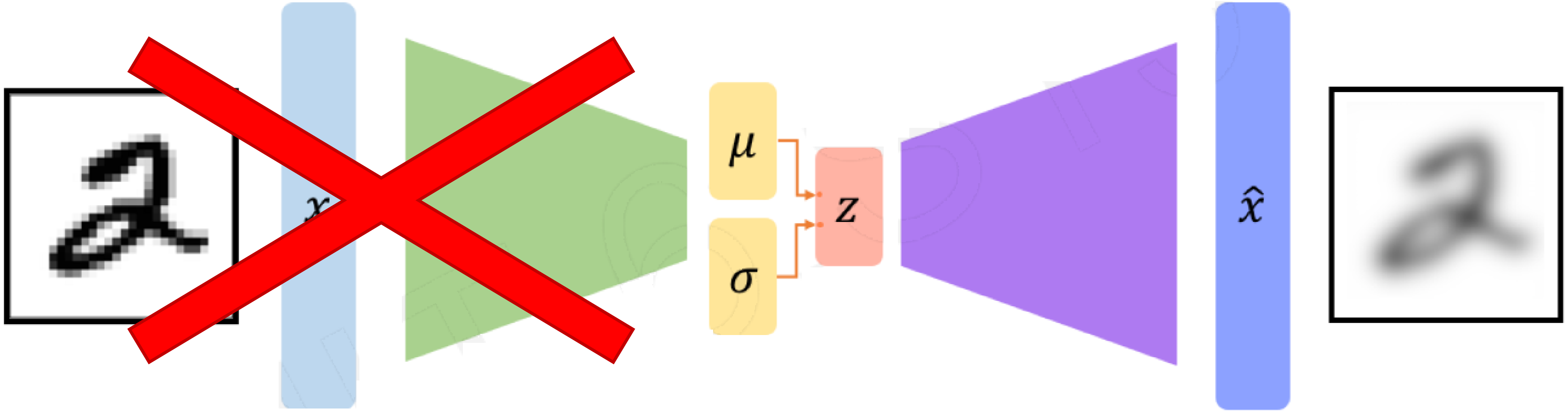
Variational Autoencoder를 활용한 MNIST 데이터셋 생성

(Google Colab. 환경)

Generative Models [12]

- **Variational Autoencoder (VAE)**
  - Defines an intractable density function
    - ▶ It derives and optimizes a lower bound on likelihood of training data instead

- **Generative Adversarial Network (GAN)**
  - Gives up on explicitly modeling density function, but just wants ability to generate data

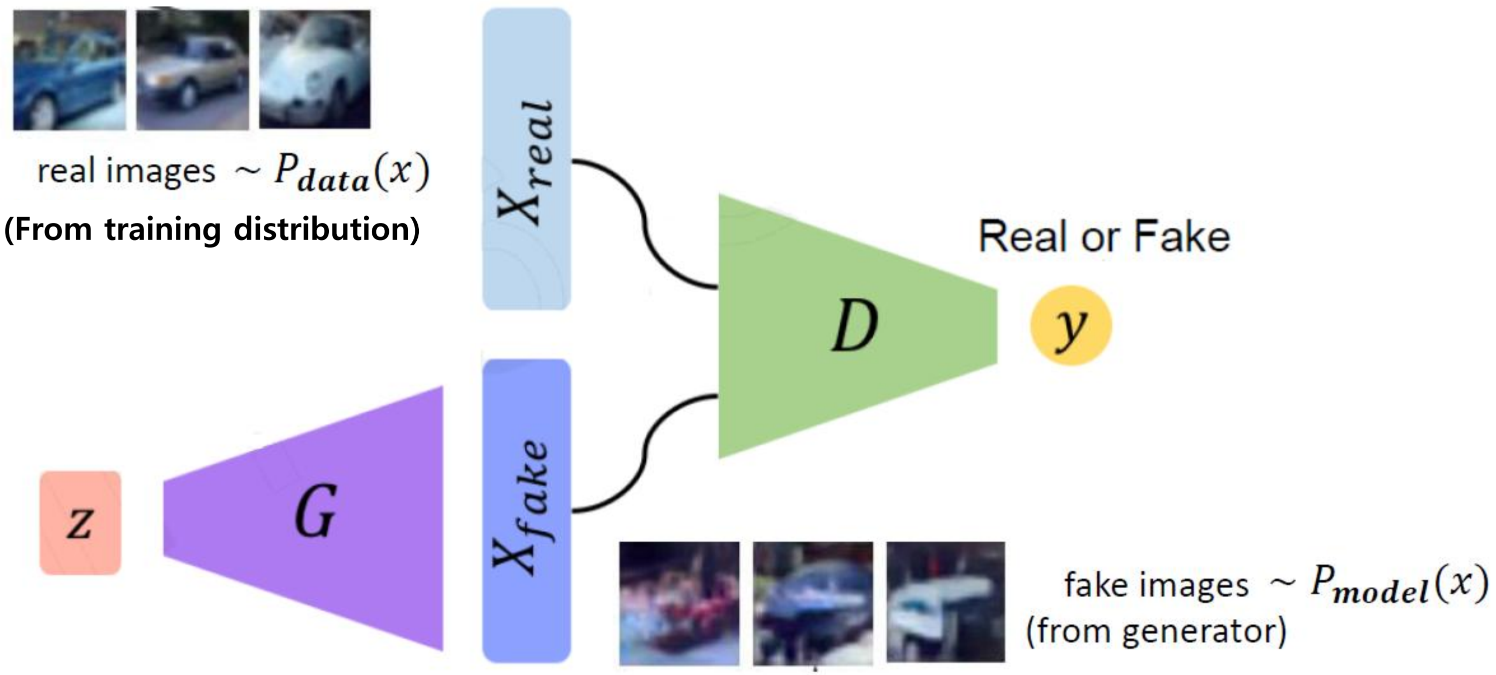


Generative Models [13]

- **Generative Adversarial Network (GAN)**
  - Gives up explicitly defining and estimating a probability
  - Just wants ability to generate data from training distribution $\boldsymbol{P_{data}(x)}$
  - Approach
    - ▶ Sample latent variable $\boldsymbol{z}$ from just **a simple distribution**, e.g., normal Gaussian $\boldsymbol{\mathcal{N}(0,1)}$ (a.k.a. random noise)
    - ▶ Then, learn a mapping function (**generator**) from $\boldsymbol{z}$ to training distribution $\boldsymbol{P_{data}(x)}$
  - Training Strategy
    - ▶ 2-player game strategy (경찰과 도둑)
    - ▶ Employ another model, named **discriminator** guiding **generator** to training distribution $\boldsymbol{P_{data}(x)}$
    - ▶ Make the generator and the discriminator compete with each other
      - ▷ Generator tries to **fool** the **discriminator** by generating **real-looking data**
      - ▷ Discriminator tries to **distinguish** between **real** and **fake data**

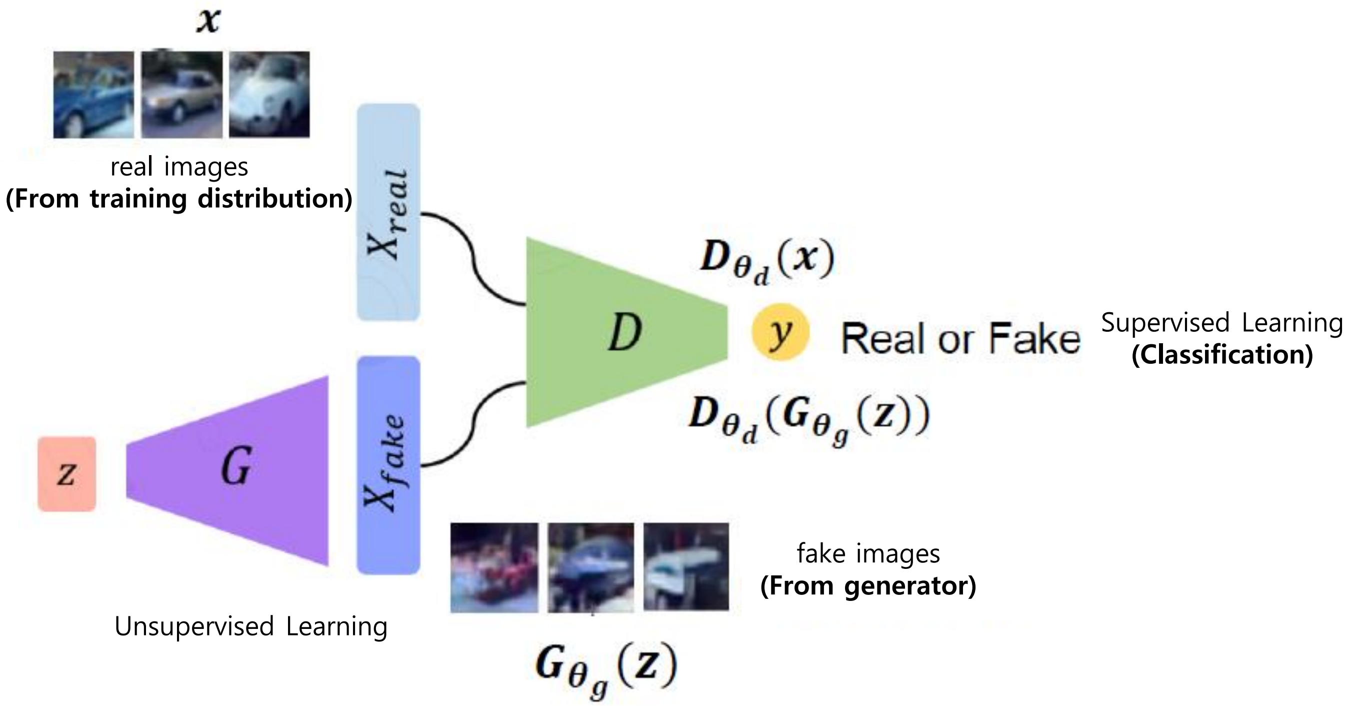
Generative Models [14]

- **Generative Adversarial Network (GAN)**
  - To succeed in this game, the generator must learn to generate data that is indistinguishable from real-world data
  - Hence, needs to generate data that **looks** drawn from the same distribution as the training data



Generative Models [15]

- **Generative Adversarial Network (GAN)**
  - Notations
    - ▶ $D_{\theta_d}(x)$ → Discriminator's output : likelihood that $\boldsymbol{x}$ is a real data, range of [0, 1]
    - ▶ $G_{\theta_g}(z)$ → Generated fake data
    - ▶ $D_{\theta_d}\left(G_{\theta_g}(z)\right)$ → Likelihood that $G_{\theta_g}(z)$ is a real data, range of [0, 1]



Generative Models [16]

- **Generative Adversarial Network (GAN)**
  - Notations
    - ▶ $D_{\theta_d}(x)$ → Discriminator's output : likelihood that $\boldsymbol{x}$ is a real data, range of [0, 1]
    - ▶ $G_{\theta_g}(z)$ → Generated fake data
    - ▶ $D_{\theta_d}\left(G_{\theta_g}(z)\right)$ → Likelihood that $G_{\theta_g}(z)$ is a real data, range of [0, 1]
  - Objective Function

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}) \right]$$

    - ▶ Discriminator wants to maximize objective such that $\boldsymbol{D_{\theta_d}(x)}$ is close to 1 (real) and $\boldsymbol{D_{\theta_d}\left(G_{\theta_g}(z)\right)}$ is close to 0 (fake)
    - ▶ Generator wants to minimize objective such that $\boldsymbol{D_{\theta_d}\left(G_{\theta_g}(z)\right)}$ is close to 1

      (discriminator is fooled into thinking generated $\boldsymbol{G_{\theta_g}(z)}$ is real)

Generative Models [17]

- **Generative Adversarial Network (GAN)**
  - Alternate between:
    - ▶ Gradient **ascent** on discriminator, generator is fixed (not updated) in this step

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

    - ▶ Gradient **descent** on generator, discriminator is fixed in this step

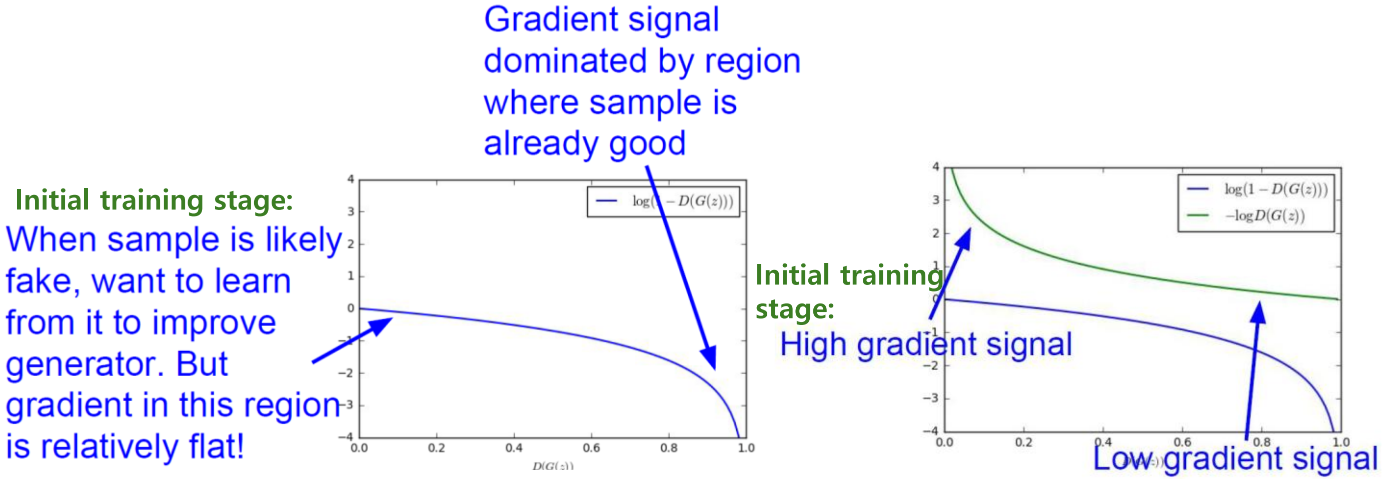
$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

    - ▶ In practice, however, optimizing this generator objective does **not** work well!
    - ▶ Hence, bottom objective function is widely-used!

$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

Generative Models [18]

- **Generative Adversarial Network (GAN)**
  - Training GAN



$$\min_{\theta_g} \mathbb{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

$$\max_{\theta_g} \mathbb{E}_{z\sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

▶ Instead of minimizing likelihood of discriminator being correct, now maximize likelihood of discriminator being wrong

▶ Same objective of fooling discriminator, but now **higher gradient signal for initial training stage**

▶ It works much better! Standard in practice!

Generative Models [19]

- **Generative Adversarial Network (GAN)**
  - Training GAN
    - ▶ Learning algorithm

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:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$

end for

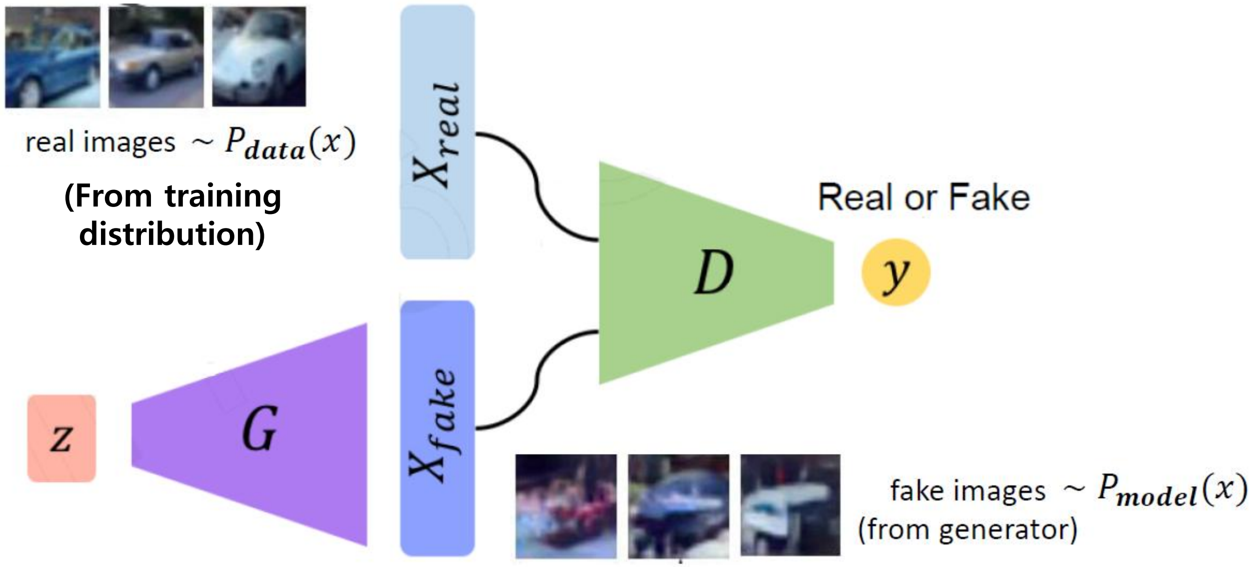
- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by ascending its stochastic gradient (improved objective):

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

end for

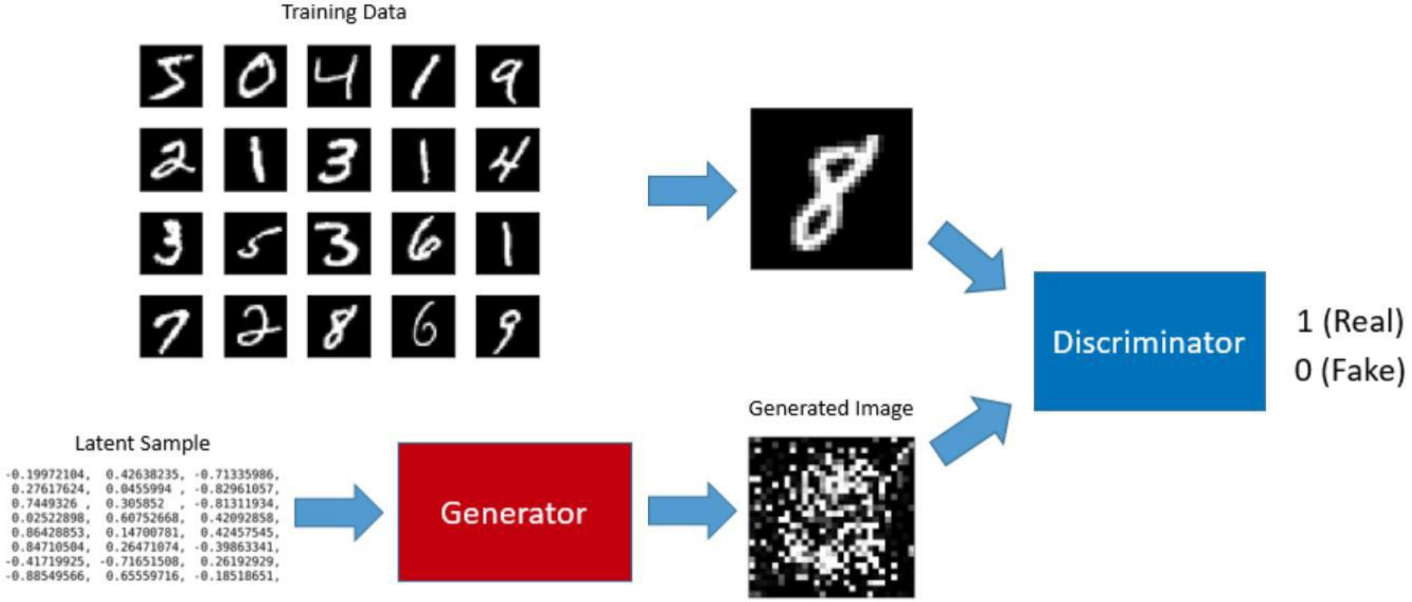
Generative Models [20]

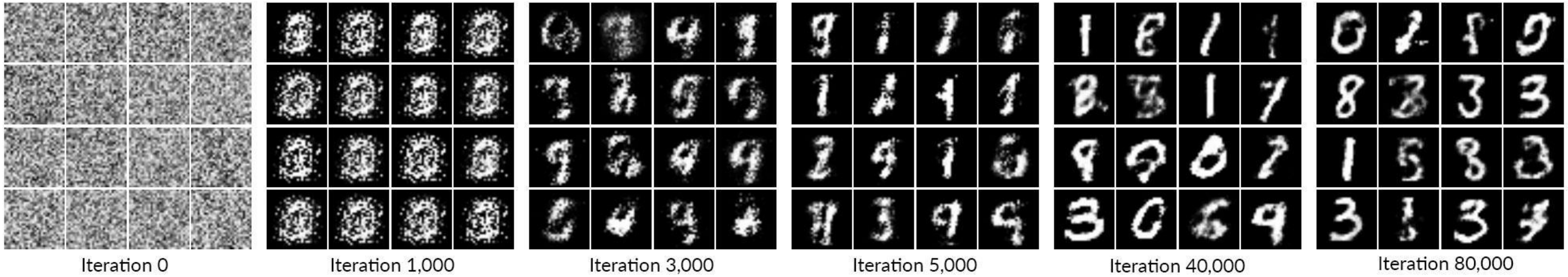
- **Generative Adversarial Network (GAN)**



Generative Models [21]

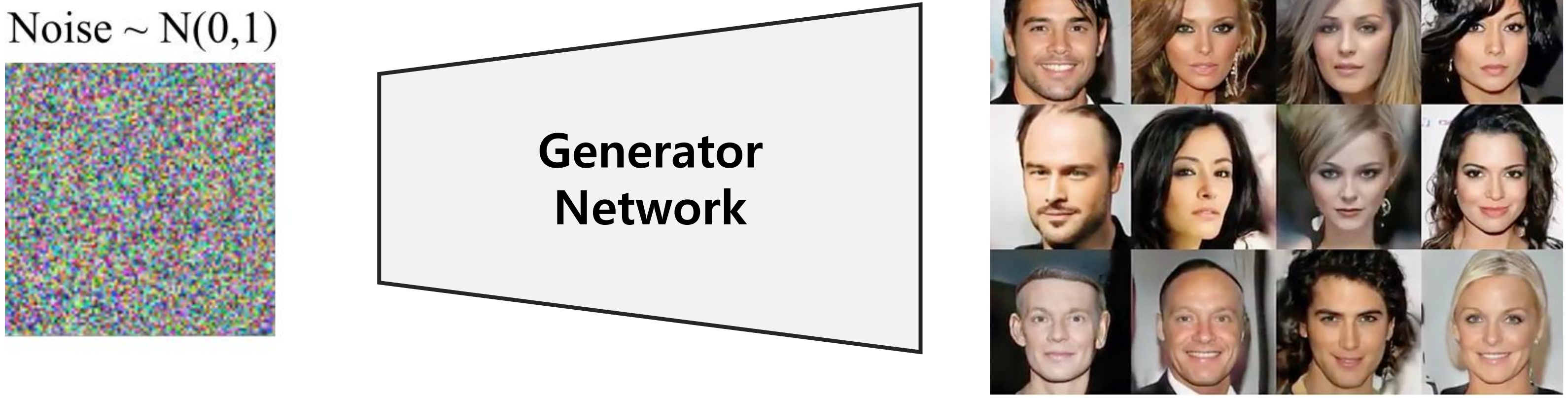
- **Generative Adversarial Network (GAN)**





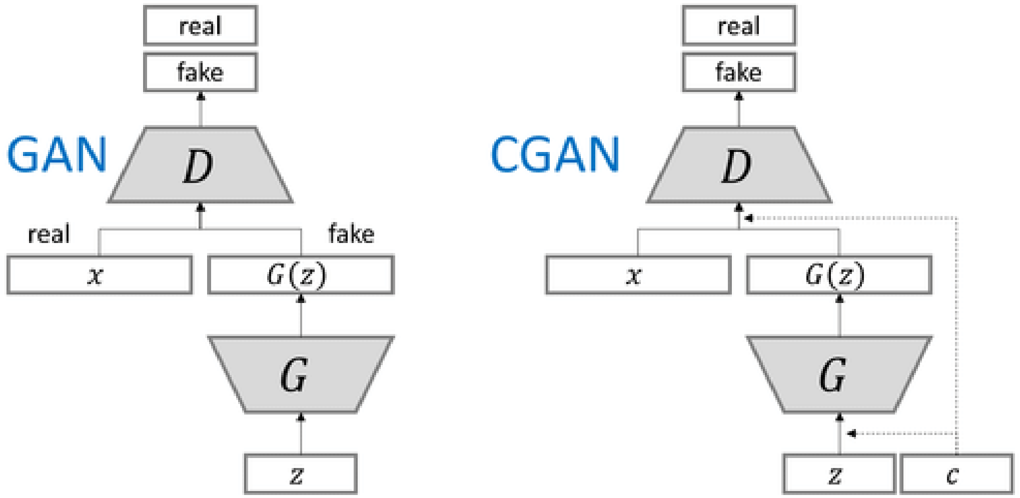
Generative Models [22]

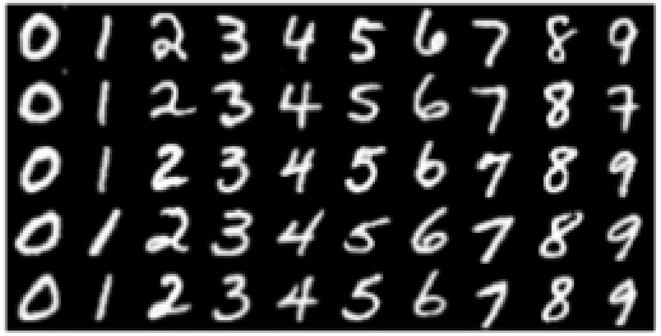
- **Generative Adversarial Network (GAN)**



Generative Models [23]

- **Generative Adversarial Network (GAN)**
  - Conditional GAN





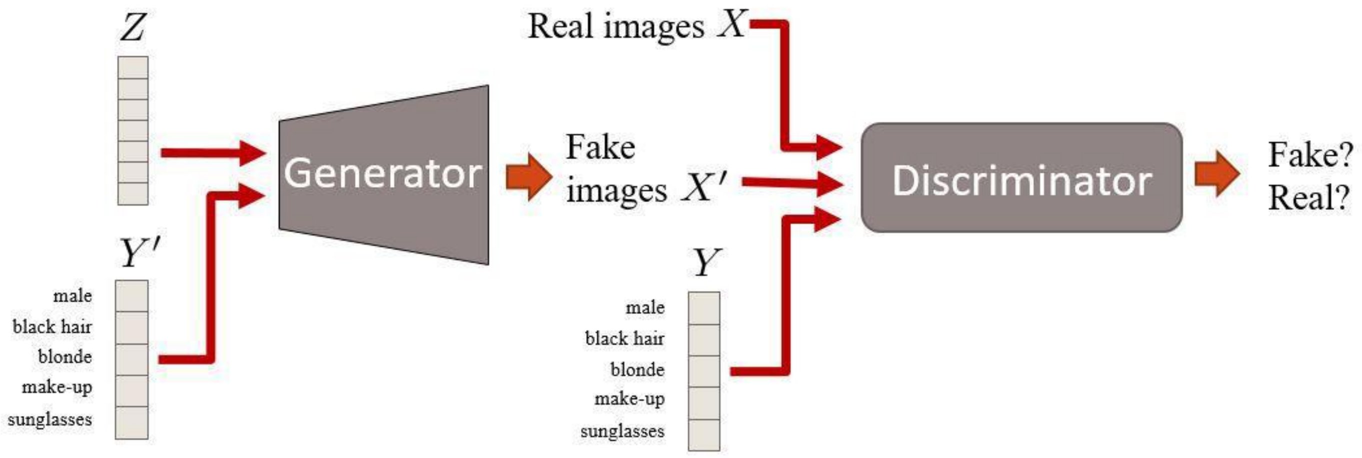
Class	0	1	2	·····	9
One-hot vector	$\begin{pmatrix}1\\0\\0\\0\\0\\0\\0\\0\\0\\0\end{pmatrix}$	$\begin{pmatrix}0\\1\\0\\0\\0\\0\\0\\0\\0\\0\end{pmatrix}$	$\begin{pmatrix}0\\0\\1\\0\\0\\0\\0\\0\\0\\0\end{pmatrix}$	·····	$\begin{pmatrix}0\\0\\0\\0\\0\\0\\0\\0\\0\\1\end{pmatrix}$

Generative Models [24]

- **Generative Adversarial Network (GAN)**
  - Conditional GAN
    - Training objective

$$\min_G \max_D \left(\mathbb{E}_{\mathbf{x},\mathbf{y}\sim p_{\text{data}}(\mathbf{x},\mathbf{y})}\left[\log D(\mathbf{x},\mathbf{y})\right] + \mathbb{E}_{\mathbf{y}\sim p_{\mathbf{y}},\, \mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}\left[\log(1 - D(G(\mathbf{z},\mathbf{y}),\mathbf{y}))\right]\right)$$

    - Implementation
      - ▷ Just concatenate the input vectors of **G** and **D** (i.e., **z** and **x**) with the condition vector **y**



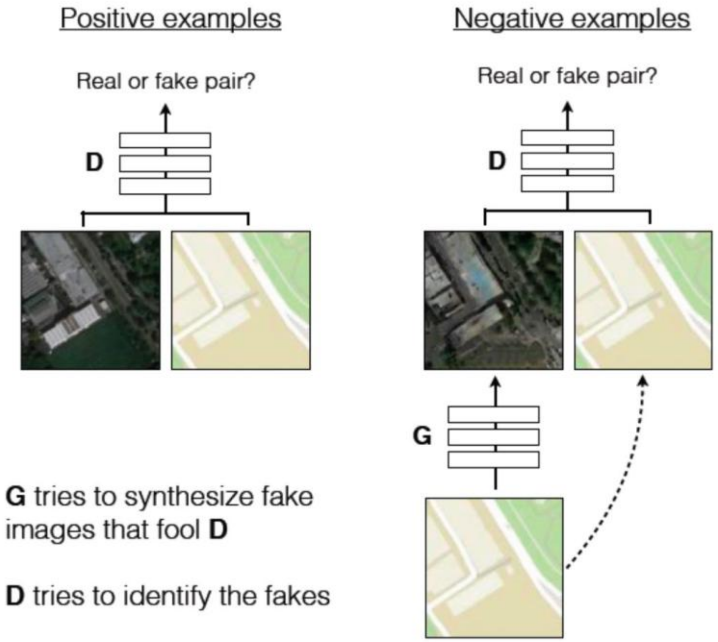
Generative Models [25]

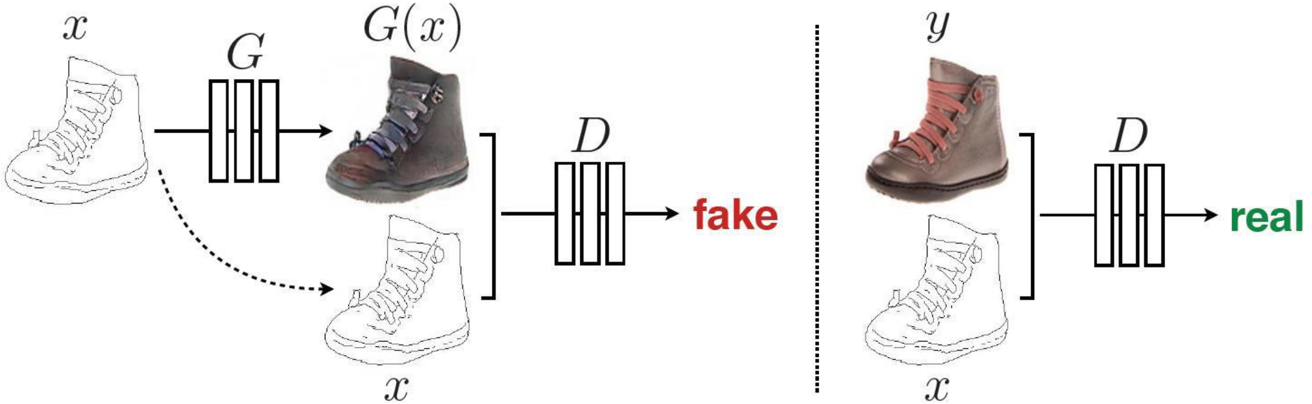
- **Generative Adversarial Network (GAN)**
  - More GANs
    - Pix2Pix
      - Image-to-Image (I2I) Translation

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G).$$

$$\mathcal{L}_{GAN}(G, D) = \mathbb{E}_y[\log D(y)] + \mathbb{E}_{x,z}[\log(1 - D(G(x, z)))].$$

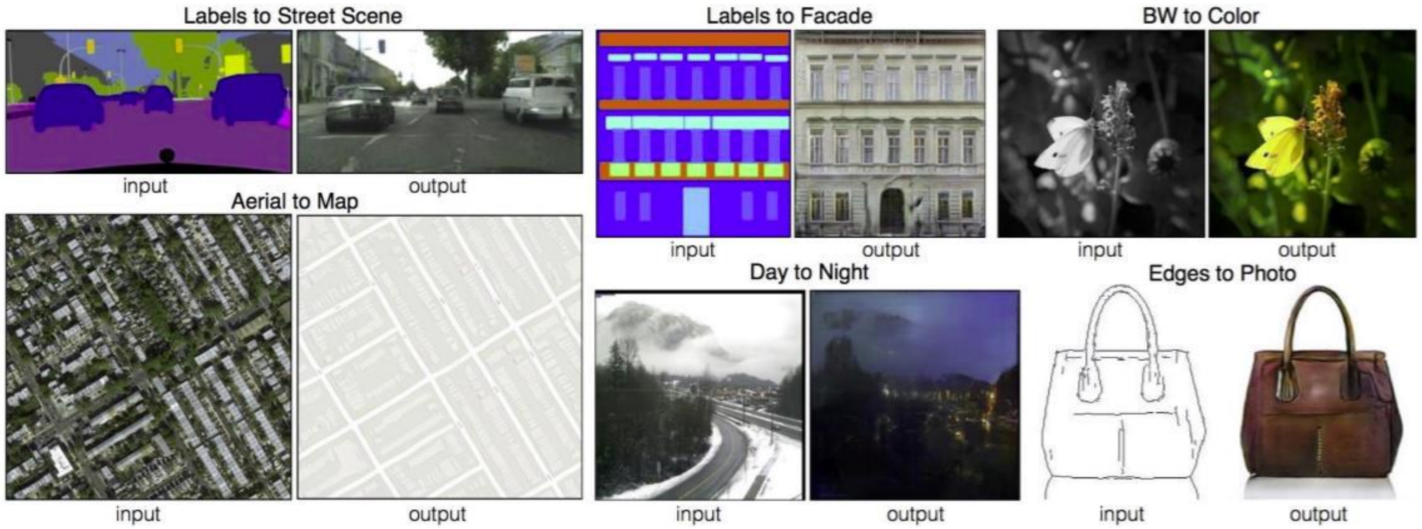
$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1].$$





Generative Models [26]

- **Generative Adversarial Network (GAN)**
  - More GANs
    - ▶ Pix2Pix
      - ▷ Image-to-Image (I2I) Translation

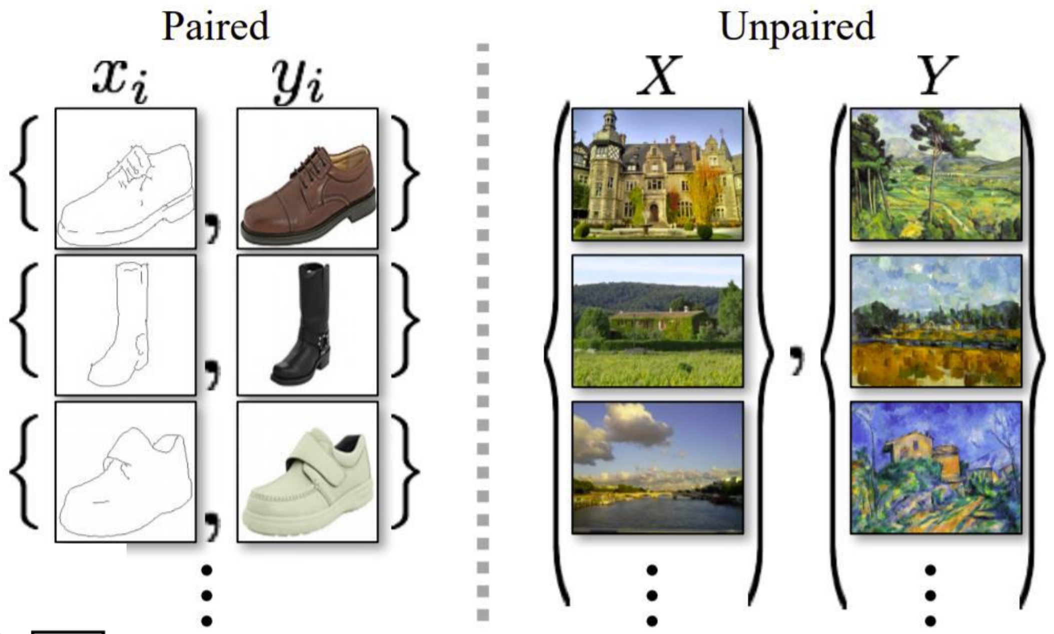


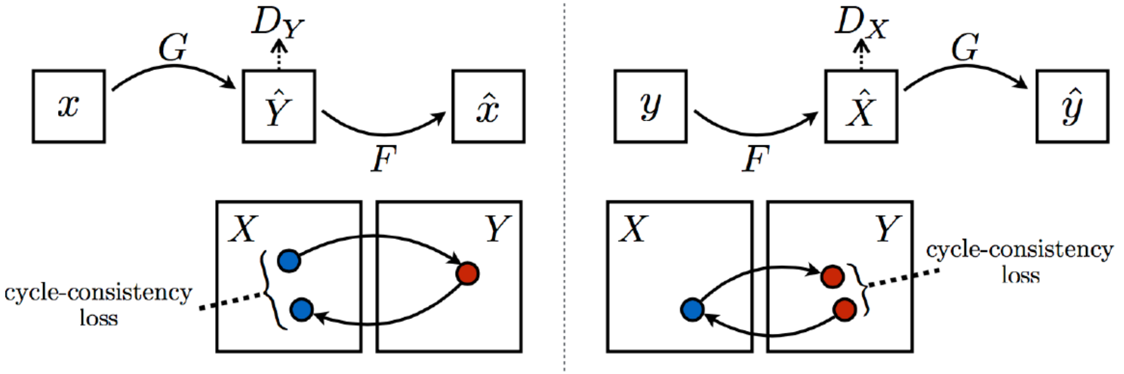
Generative Models [27]

- **Generative Adversarial Network (GAN)**
  - More GANs
    - ▶ DiscoGAN & CycleGAN
      - ▷ Image-to-Image (I2I) Translation (Paired & Unpaired)

$$G^*, F^* = \arg\min_{G,F} \max_{D_x, D_Y} \mathcal{L}(G, F, D_X, D_Y)$$

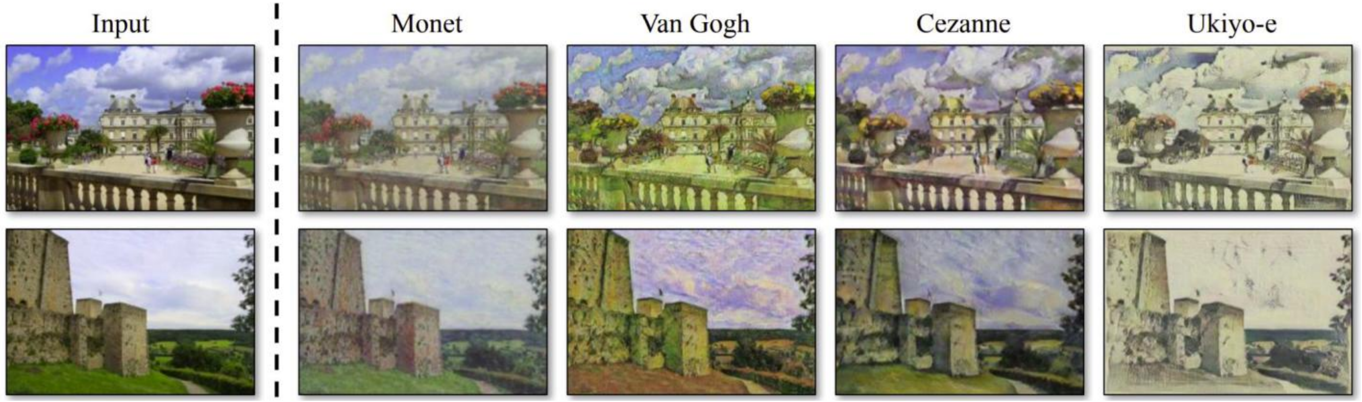
$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) = &\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ &+ \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ &+ \lambda \mathcal{L}_{\text{cyc}}(G, F),\end{aligned}$$

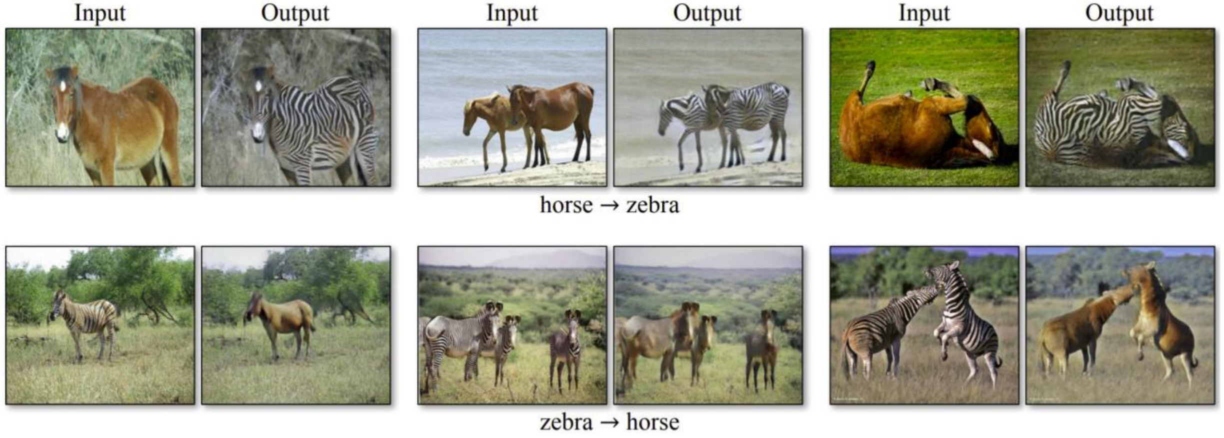




Generative Models [28]

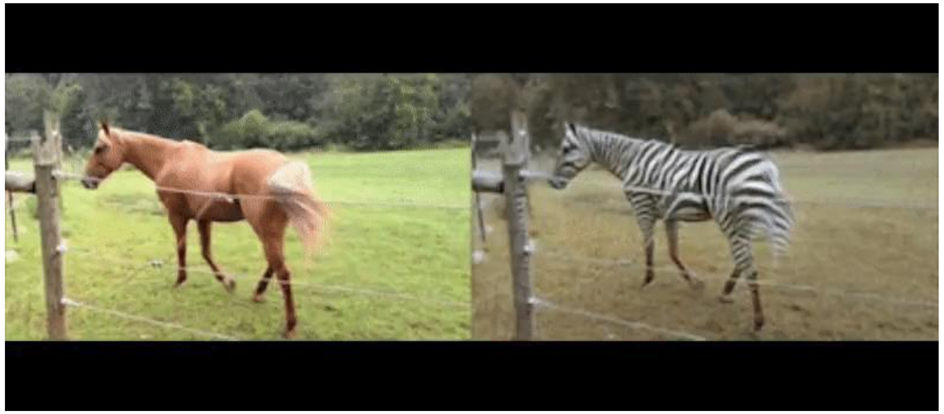
- **Generative Adversarial Network (GAN)**
  - More GANs
    - ▶ DiscoGAN & CycleGAN
      - ▷ Image-to-Image (I2I) Translation (Paired & Unpaired)





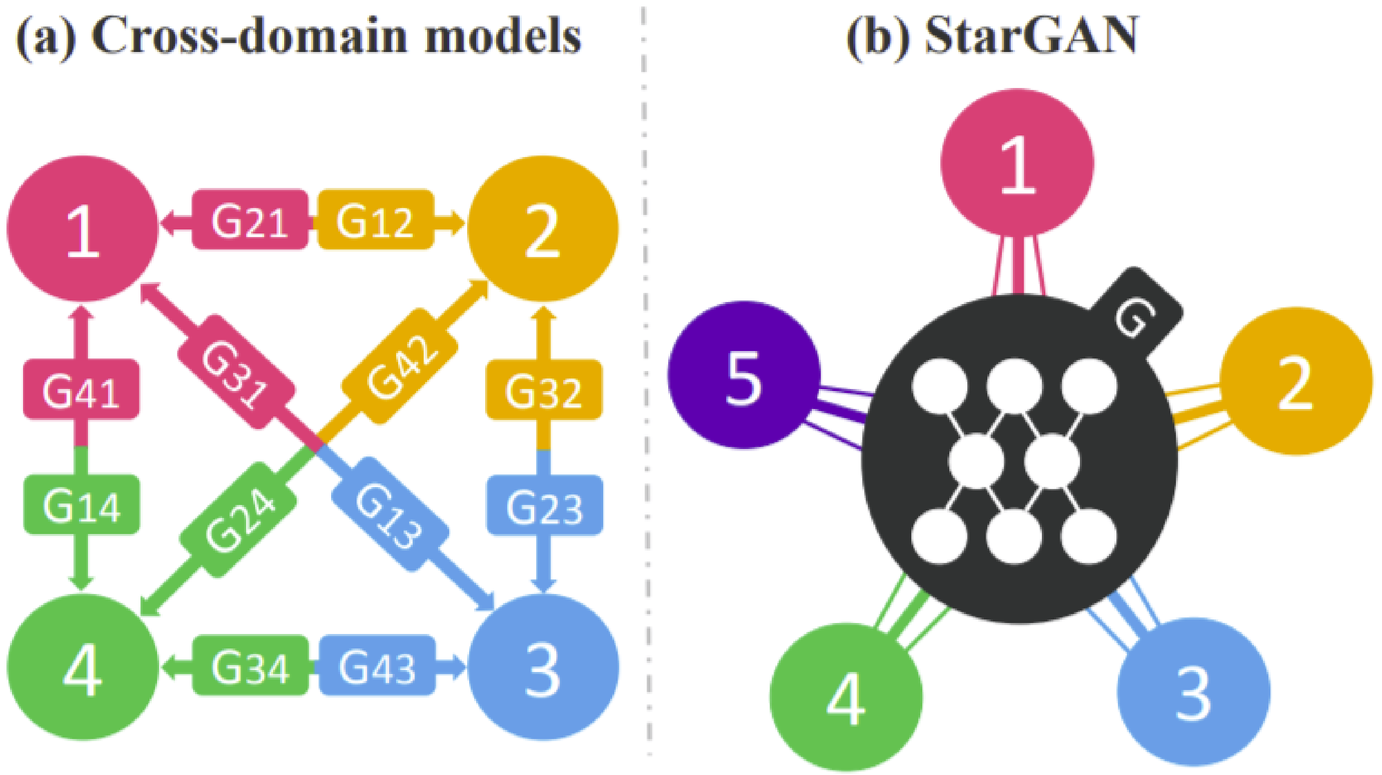
Generative Models [29]

- **Generative Adversarial Network (GAN)**
  - More GANs
    - ▶ DiscoGAN & CycleGAN
      - ▷ Image-to-Image (I2I) Translation (Paired & Unpaired)



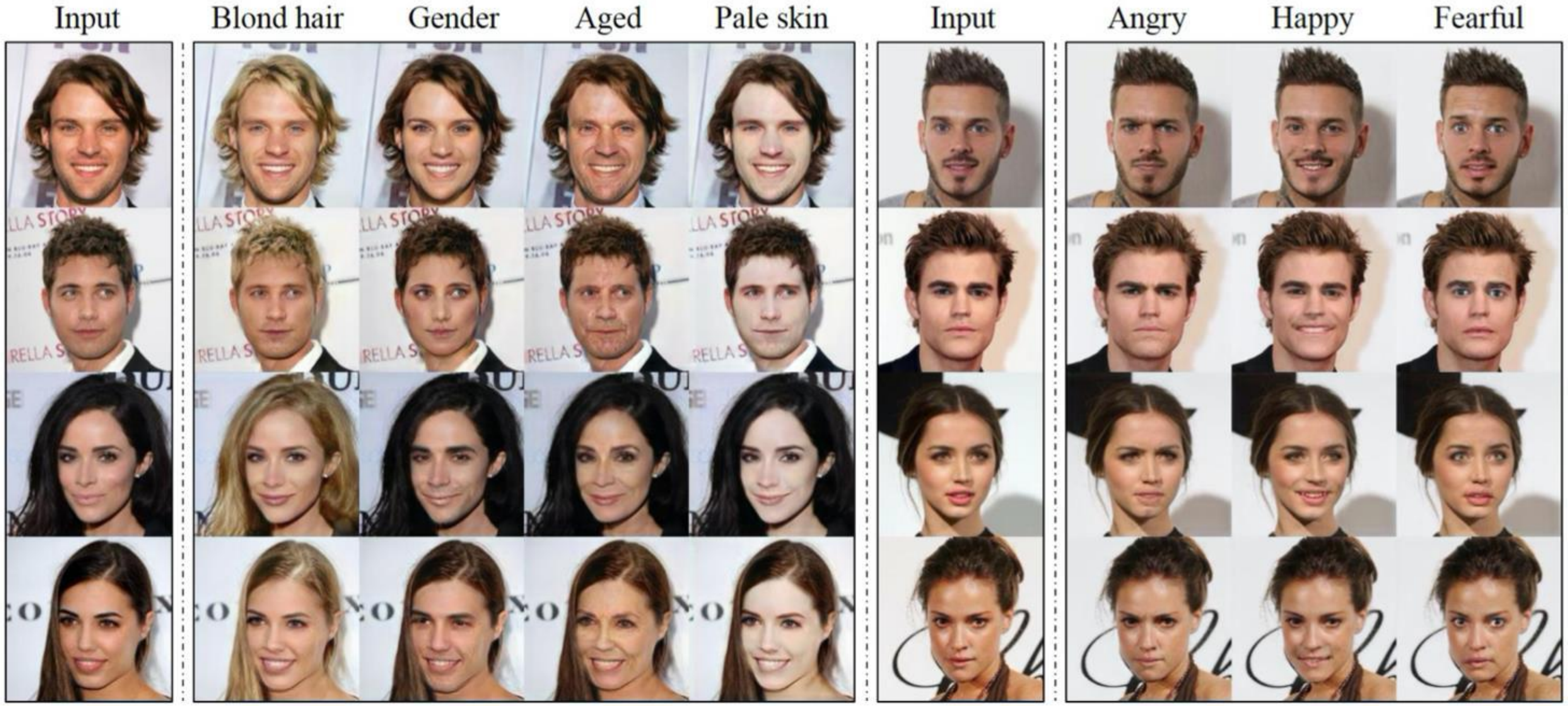
Generative Models [30]

- **Generative Adversarial Network (GAN)**
  - More GANs
    - ▶ StarGAN



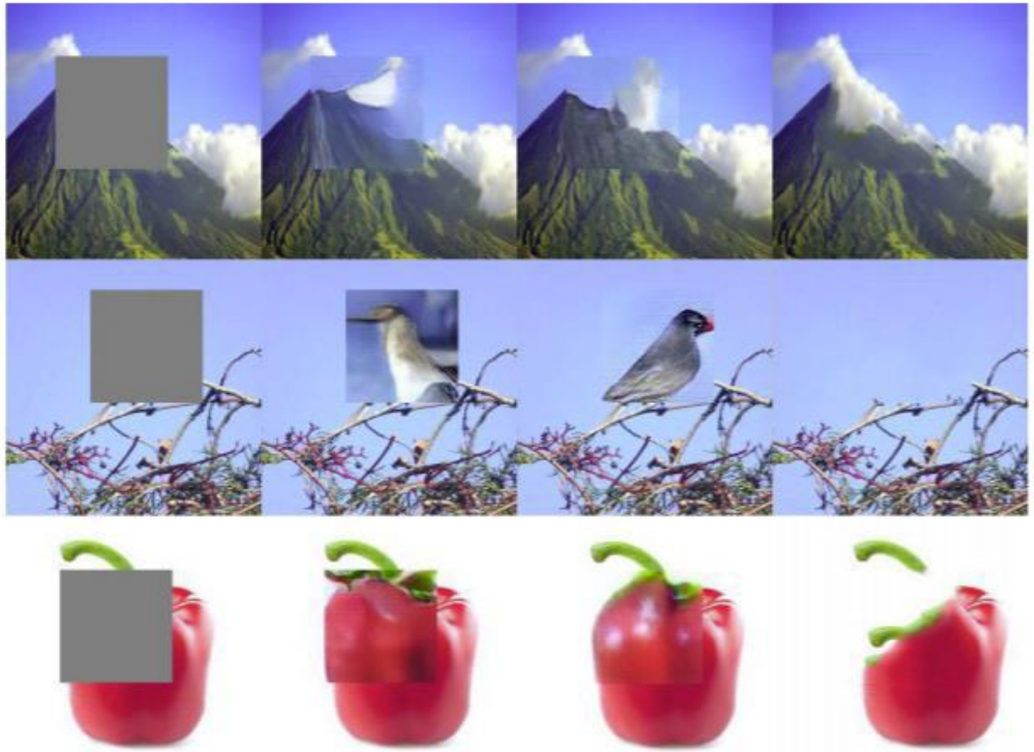
Generative Models [31]

- **Generative Adversarial Network (GAN)**
  - More GANs
    - ▶ StarGAN



Generative Models [32]

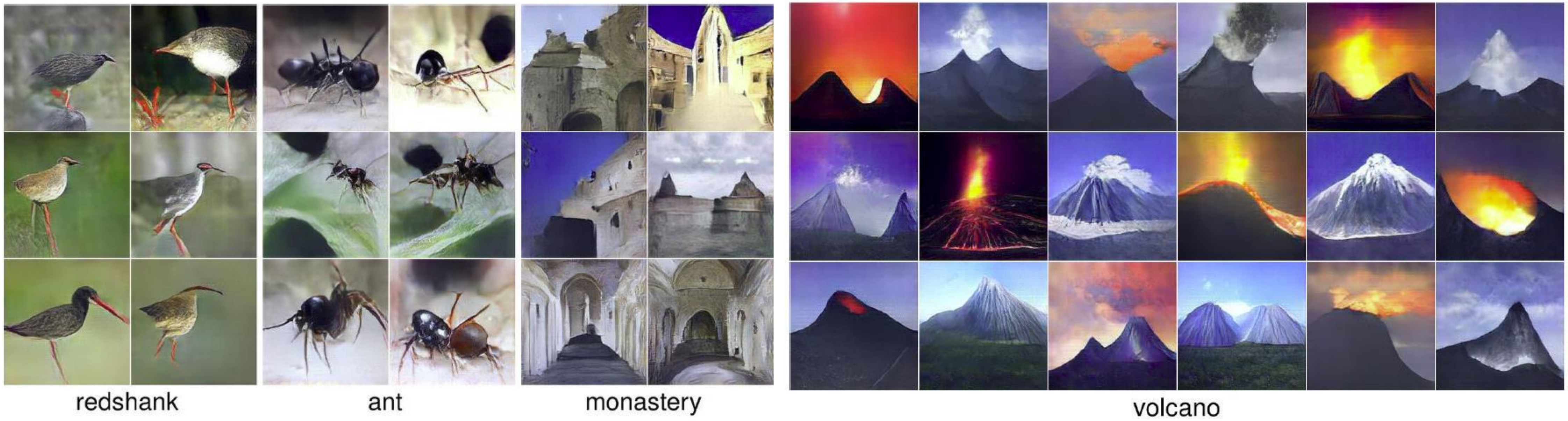
- **Generative Adversarial Network (GAN)**
  - More GANs
    - ▶ Image Inpainting



a) masked image b) PPGN c) PPGN-context d) Photoshop

Generative Models [33]

- **Generative Adversarial Network (GAN)**
  - More GANs
    - ▶ Category-to-Image



Generative Models [34]

- **Generative Adversarial Network (GAN)**
  - More GANs
    - ▶ Category-to-Image

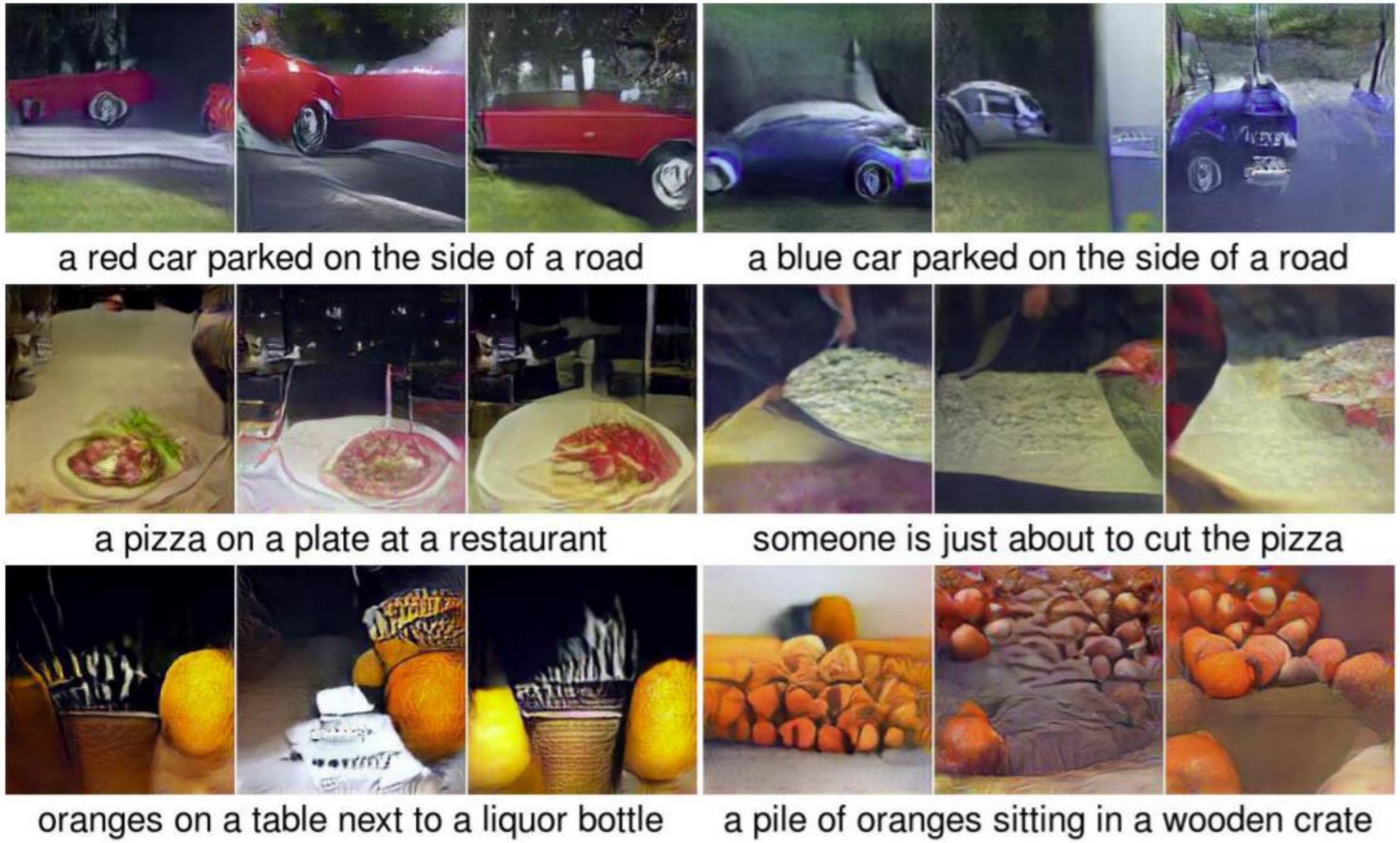
(a) Snail

(b) Studio couch

(c) Harvester

Generative Models [35]

- **Generative Adversarial Network (GAN)**
  - More GANs
    - ▶ Caption-to-Image (Text-to-Image)



Generative Models [36]

- **Generative Adversarial Network (GAN) 실습**

Generative Adversarial Network를 활용한 Fashion MNIST데이터셋 생성

(Google Colab. 환경)