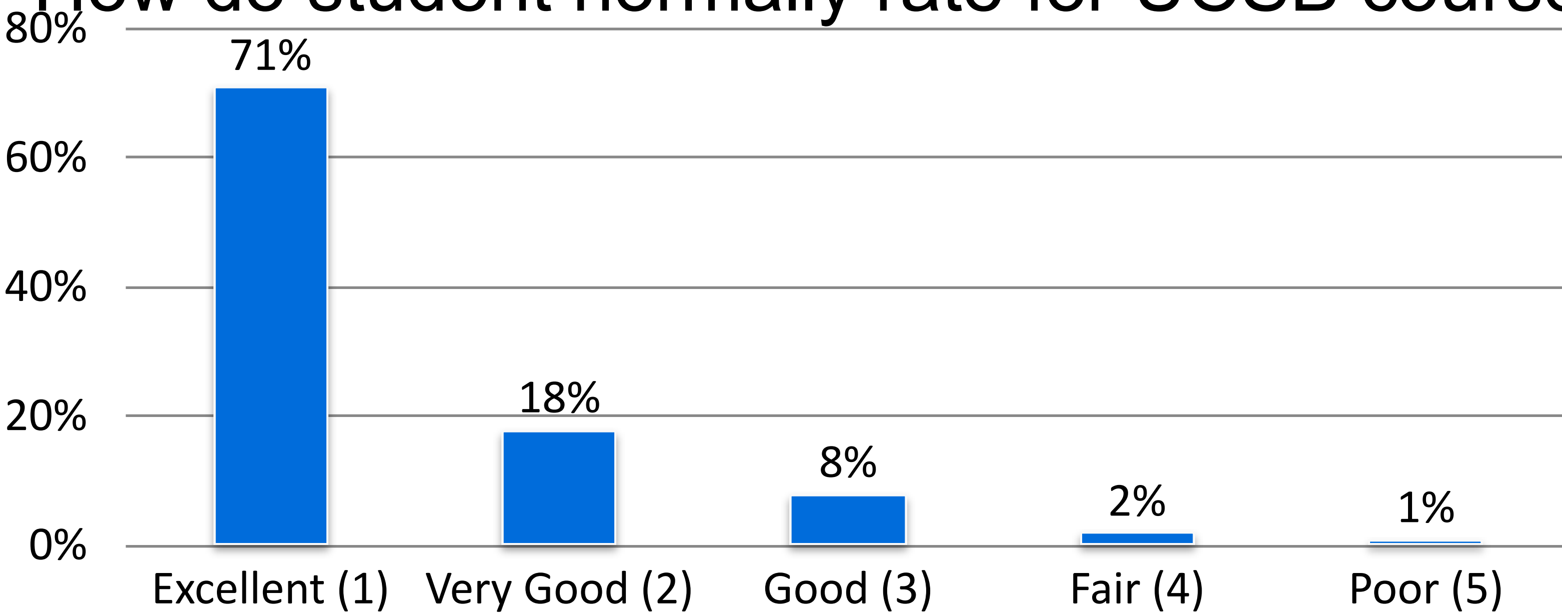
165B
Machine Learning
Generative Adversarial Networks

Lei Li (leili@cs)
UCSB

Course Evaluation

- https://esci.id.ucsb.edu
- Feedback is important and helpful for improving the course
- Encourage narrative comments:
  - specific aspects of the course and instruction

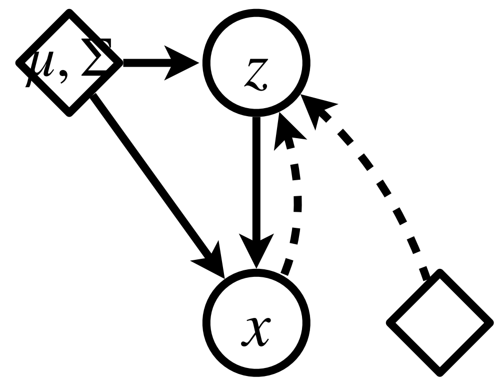
How do student normally rate for UCSB courses?



Summary

- Auto-Encoder: learning representation by reconstruction
- Variational Auto-Encoder: put prior on latent representation and use variational method to train

Graphical Model for VAE



- Assuming data X is generated from a latent variable Z
- Generation process
  - draw $Z \sim N(\mu, \Sigma)$
  - draw $X | Z \sim p(f(Z))$, defined by a neural network f
- The goal is to maximize the data log-likelihood

$$\log p(X;\theta) = \log \int p(X \mid Z) p(Z) dZ$$

- Hard to optimize over θ, if f(Z) is very complex such as a CNN, RNN, or Transformer.

Training VAE

gradient descent(ascent for max)

$$\max_{\theta} \max_{\phi} \text{ELBO} = \sum_n \mathrm{E}_{q(z_n|x_n;\theta)} \left[\log \frac{p(x_n \mid z_n; \theta) p_0(z_n)}{q(z_n \mid x_n; \theta)} \right]$$

$$= \sum_n \mathrm{E}_{q(z_n|x_n;\theta)} \left[r(\theta, z_n, x_n) \right]$$

$$r(\theta, z_n, x_n) = \log \frac{p(x_n \mid z_n; \theta) p_0(z_n)}{q(z_n \mid x_n; \theta)}$$

Computing gradient:

$$\nabla_\theta \mathrm{E}_{q(z_n|x_n;\theta)} \left[r(\theta, z_n, x_n) \right]$$

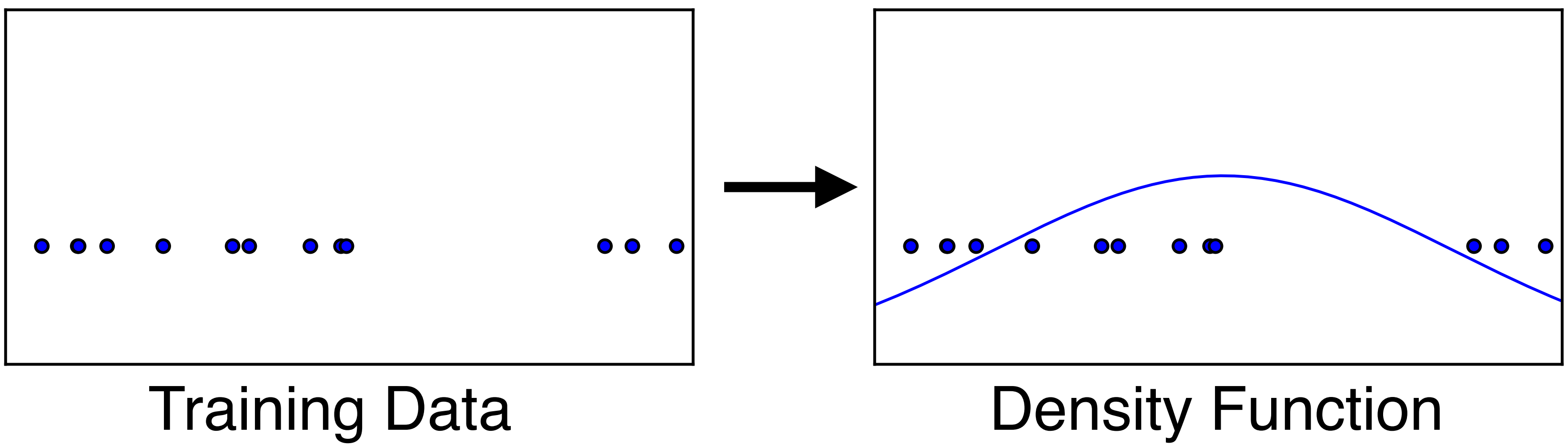
Generative Model

- Density estimation
- Generate new and similar data

Training Data
(CelebA)

Sample Generator
(Karras et al, 2017)

Density Estimation



Motivation for Generative Adversarial Training

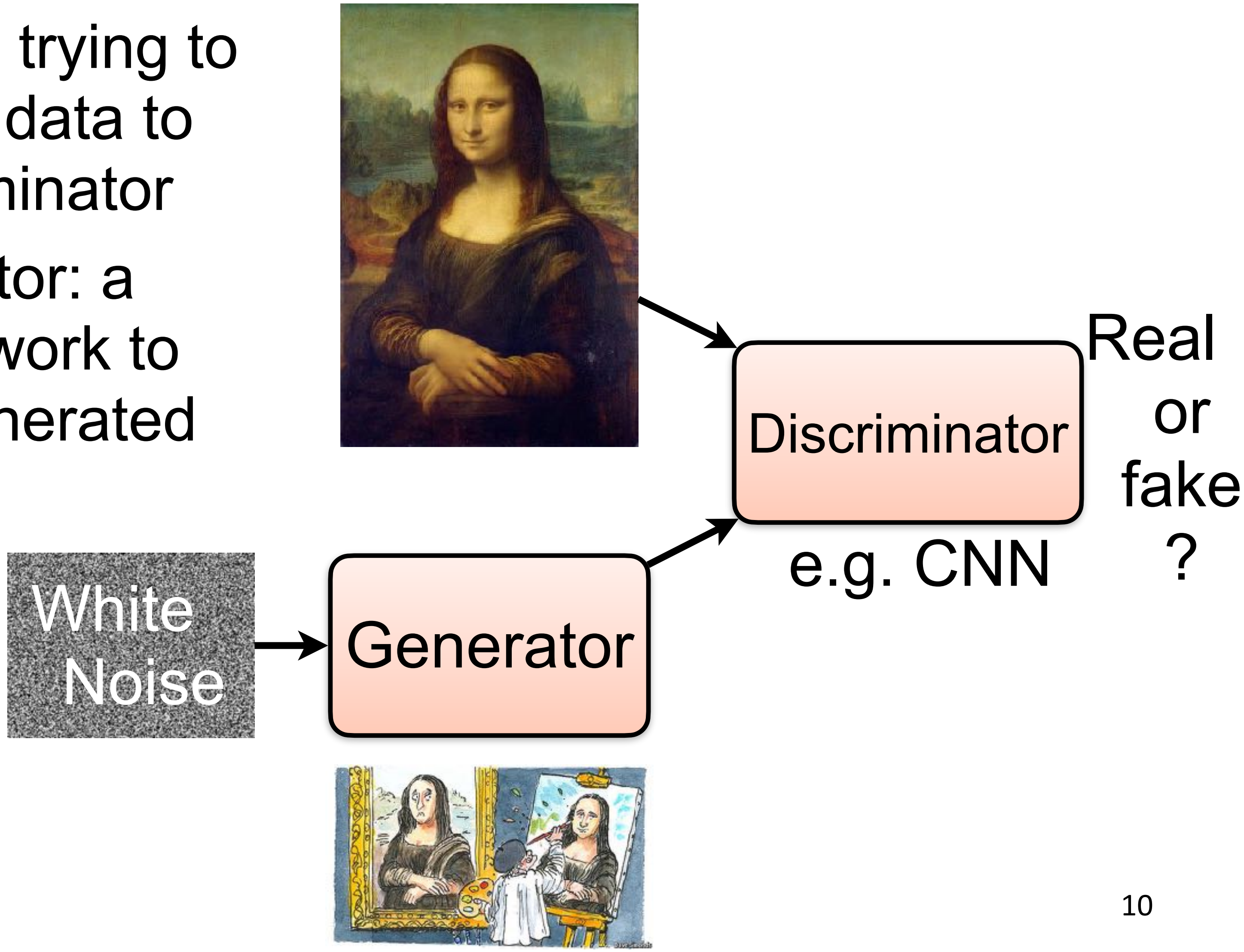
- Fitting a distribution is hard, maximum-likelihood estimation may have issues (overestimate/underestimate)
- Why don't we simultaneous train a generative model and a model to measure the quality of fitting?
- Likelihood-free: could not explicitly write down a likelihood, but will be able to generate samples.

Generative Adversarial Network (GAN)

- Learn a generative model that has distribution close to empirical distribution
- Game theoretic idea: two networks playing adversarial games against each other
- Generator: a neural network with distribution P_g, trying to mimic real data
- Discriminator: a neural network to distinguish the samples generated from the model and the real data

GAN

- Generator: trying to mimic real data to fool discriminator
- Discriminator: a neural network to identify generated samples



Adversarial Game

- Generator: G(z), z~N(0,1)
- Discriminator: D(x) either taking a real sample as input or a generated sample
- Objective:
  – G tries to maximize the chances that Discriminator will think the generated samples are real, D(G(z))
  – D tries to maximize the probability to identify real data D(x), and minimize the chances that the generated samples will pass checking D(G(z))

Training Loss of GAN

- Generator: G(z), z~N(0,1)

$$\min_G \ell_G = E_z\left[\log D(G(z))\right]$$

- Discriminator: D(x) either taking a real sample (=0) as input or a fake sample (=1)

$$\min_D \ell_D = -\frac{1}{2}E_{x\sim P_{data}}\left[\log(1-D(x))\right] - \frac{1}{2}E_z\left[\log D(G(z))\right]$$

- Combine together:

$$\min_G \max_D \ell = \frac{1}{2}E_{x\sim P_{data}}\left[\log(1-D(x))\right] + \frac{1}{2}E_z\left[\log D(G(z))\right]$$

What does GAN actually optimize?

- What is theoretically optimal Discriminator?

$$\max_D \ell = \frac{1}{2} E_{x \sim P_{data}} \left[\log(1 - D(x)) \right] + \frac{1}{2} E_z \left[\log D(G(z)) \right]$$

$$= \frac{1}{2} \left(E_{x \sim P_{data}} \left[\log(1 - D(x)) \right] + \frac{1}{2} E_{x \sim P_G} \left[\log D(x) \right] \right)$$

$$= \frac{1}{2} \int \left(p_{data}(x) \log(1 - D(x)) + p_G(x) \log D(x) \right) dx$$

$$D^*(x) = \frac{p_G(x)}{p_{data}(x) + p_G(x)}$$

What does GAN actually optimize?

- What is theoretically optimal Discriminator?

$$\max_D \ell = \frac{1}{2} E_{x \sim P_{data}} \left[\log(1 - D(x)) \right] + \frac{1}{2} E_z \left[\log D(G(z)) \right]$$

$$= \frac{1}{2} \left(E_{x \sim P_{data}} \left[\log(1 - D(x)) \right] + \frac{1}{2} E_{x \sim P_G} \left[\log D(x) \right] \right)$$

$$= \frac{1}{2} \int \left(p_{data}(x) \log(1 - D(x)) + p_G(x) \log D(x) \right) dx$$

$$D^*(x) = \frac{p_G(x)}{p_{data}(x) + p_G(x)}$$

What does GAN actually optimize?

Plug in D^* in ℓ

$$D^*(x) = \frac{p_G(x)}{p_{data}(x) + p_G(x)}$$

$$\min_G \max_D \ell = \min_G \max_D \frac{1}{2} \int \left(p_{data}(x) \log(1 - D(x)) + p_G(x) \log D(x)\right) dx$$

$$= \min_G \frac{1}{2} \int \left(p_{data}(x) \log(1 - D^*(x)) + p_G(x) \log D^*(x)\right) dx$$

$$= \min_G \frac{1}{2} \int \left(p_{data}(x) \log \frac{p_{data}(x)}{p_{data}(x) + p_G(x)} + p_G(x) \log \frac{p_G(x)}{p_{data}(x) + p_G(x)}\right) dx$$

$$= \min_G \frac{1}{2} \left(\mathrm{KL}\left(p_{data} \| \frac{p_{data}(x) + p_G(x)}{2}\right) + \mathrm{KL}\left(p_G \| \frac{p_{data}(x) + p_G(x)}{2}\right)\right) - \log 2$$

$$= \min_G \mathrm{JSD}\left(p_{data} \| p_G\right) - \log 2$$

GAN is essentially minimizing Jensen-Shannon divergence between observed data distribution and generation distribution

Better distance in GAN?

Instead of using Jensen-Shannon Divergence, use Wasserstein distance (or Earth-Moving Distance)

$$\min_G \mathrm{EMD}(p_{data} || p_G)$$

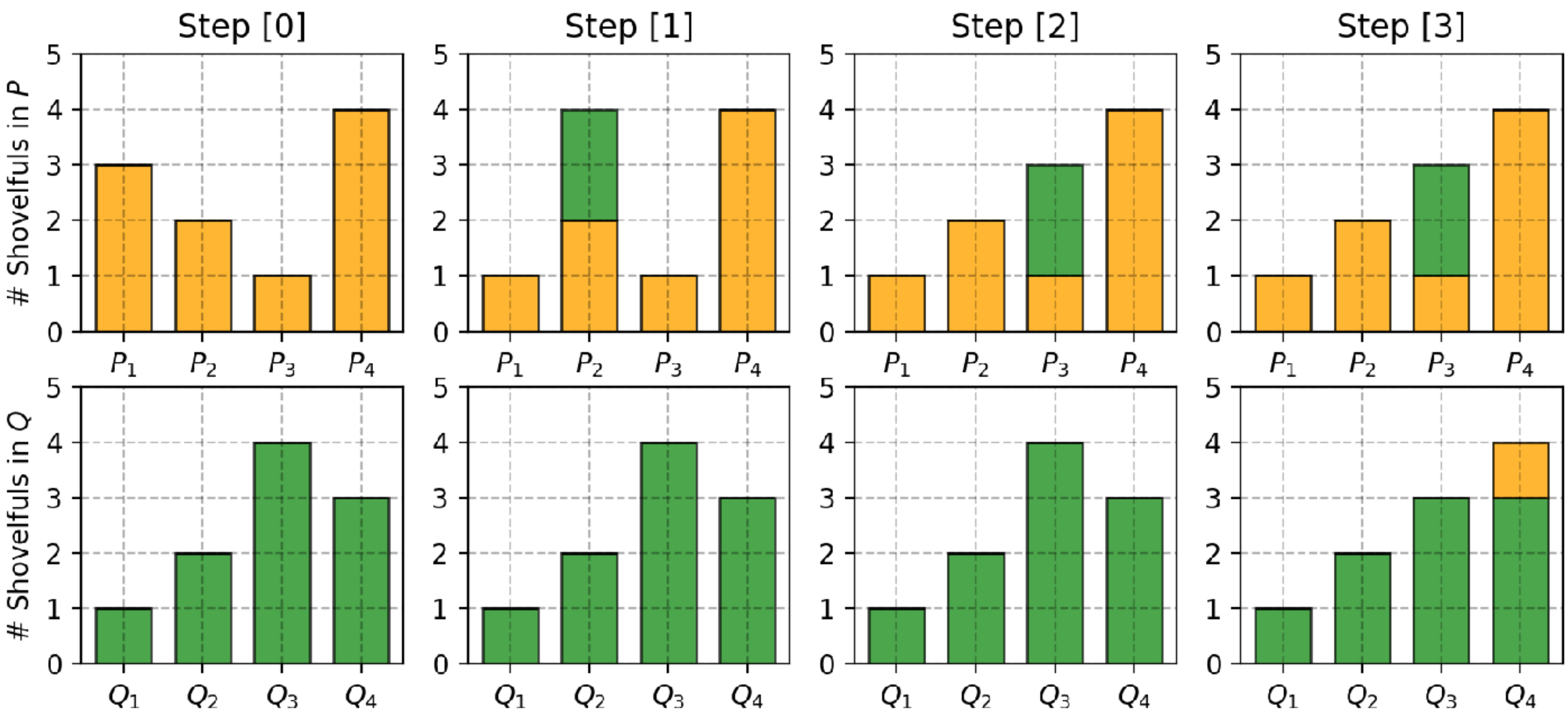
and EMD is the minimum cost to transfer a distribution p(x) into q(y).

$$\mathrm{EMD}(p(x) || q(y)) = \inf_{\pi(x,y)} E_{(x,y)\sim\pi}[\,|x-y|\,]$$

s.t $\int \pi(x, y)dx = q(y)$ and $\int \pi(x, y)dy = p(x)$

Earth Moving Distance (Wasserstein Distance)

Moving yellow distribution to green one



Why Wasserstein?

- Wasserstein is smooth while JSD may not be

Wasserstein GAN

- Instead of directly optimizing EMD, which is intractable.
- From Kantorovich-Rubinstein duality,

$$\mathrm{EMD}(p\|q) = \sup_f E_{x\sim P_{data}}[f(x)] - E_{x\sim P_G}[f(x)]$$

- Therefore, the objective becomes

$$\min_G \max_f E_{x\sim P_{data}}[f(x)] - E_{x\sim P_G}[f(x)]$$

Training WGAN

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**

  for $t = 0, ..., n_{\text{critic}}$ **do**

    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.

    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.

    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$

    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$

    $w \leftarrow \text{clip}(w, -c, c)$

  end for

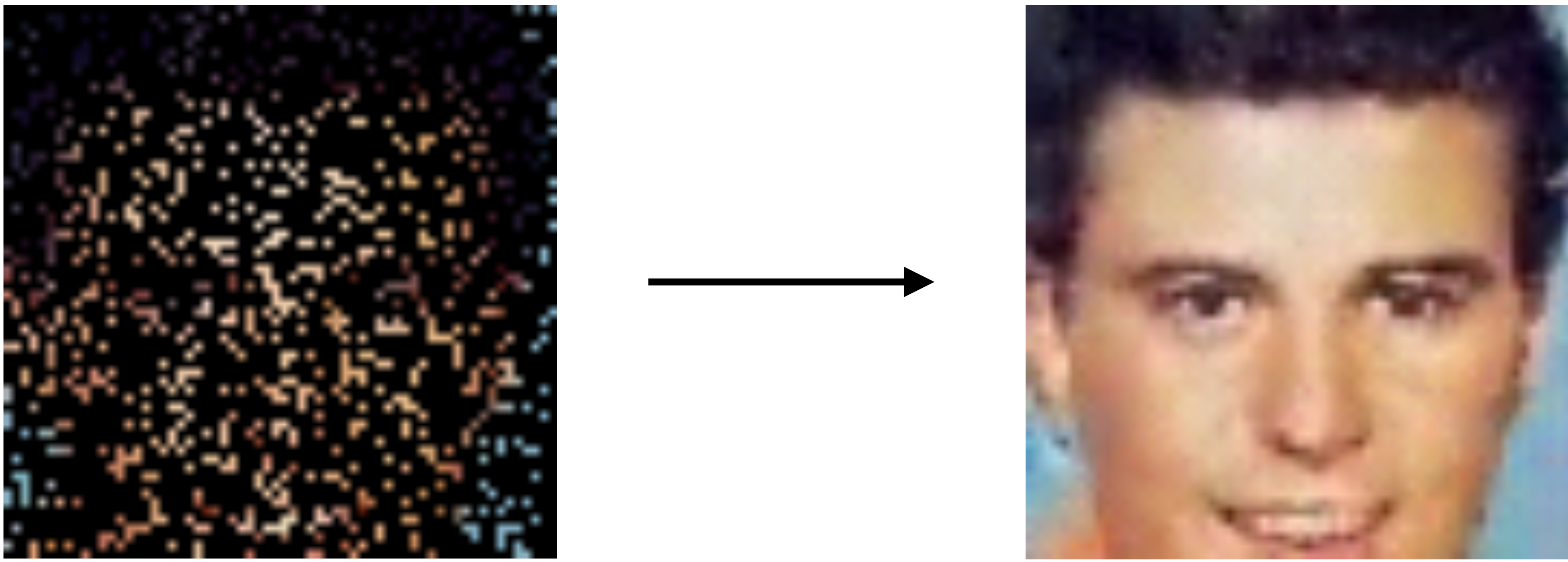
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.

  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$

  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$

end while

Generative modeling reveals a face



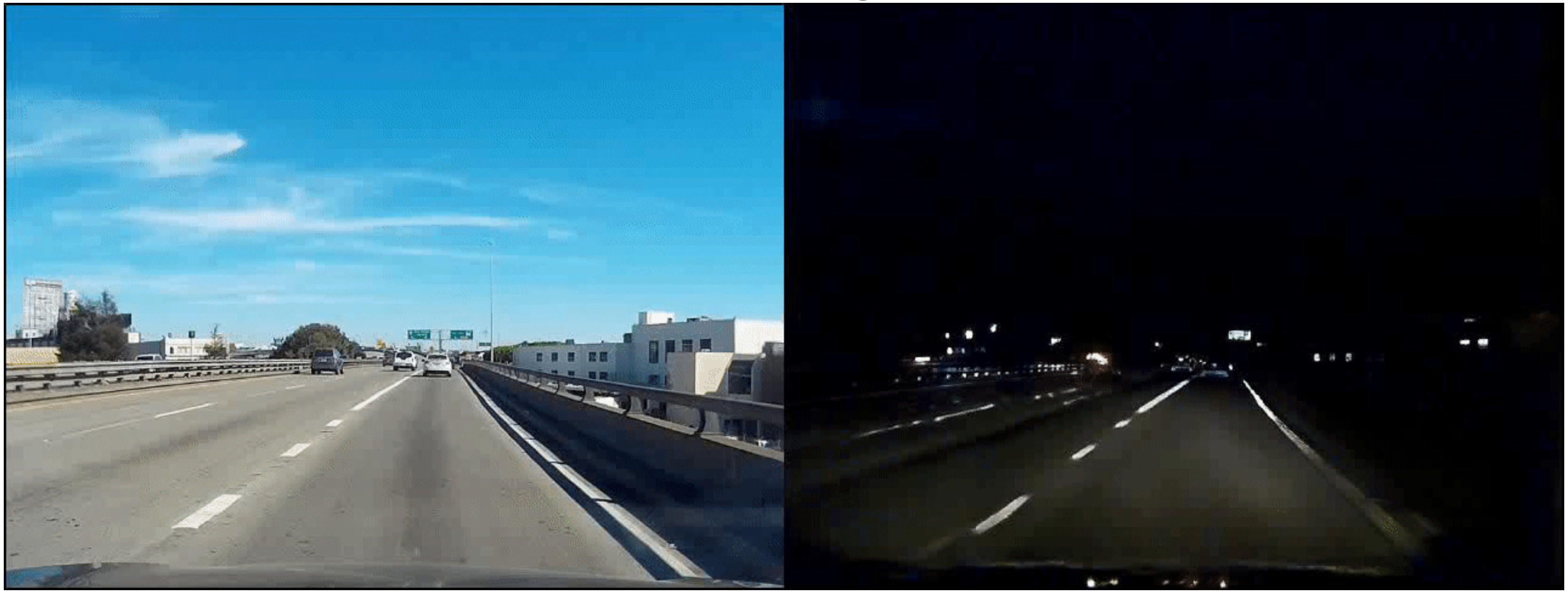
(Yeh et al., 2016)

Image to Image Translation

(Isola et al., 2016)

Unsupervised Image-to-Image Translation

Day to night



(Liu et al., 2017)

CycleGAN

(Zhu et al., 2017)

Text-to-Image Synthesis

This bird has a yellow belly and tarsus, grey back, wings, and brown throat, nape with a black face

(Zhang et al., 2016)

Summary

- GAN as a minimax game
  - Generator tries to fool the discriminator
  - Discriminator tries to distinguish real from fake.
- Original GAN corresponds to minimizing the Jensen-Shannon divergence
- WGAN improves by using Earth-Moving distance.
  - another minimax game.