Generative Adversarial Networks (GANs)

Naeemullah Khan
naeemullah.khan@kaust.edu.sa

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

KAUST Academy
King Abdullah University of Science and Technology

June 2, 2025

Table of Contents

1. Objectives
2. Motivation
3. Comparing Distributions via Samples
4. Definition
5. Training GANs
6. Understanding Objective Function
7. Results of GANs
8. Problem with GANs
9. Variants of GANs
   - 9.1 Wasserstein GANs
   - 9.2 Conditional GANs
   - 9.3 CycleGANs

Table of Contents (cont.)

9.4 StyleGANs

10. Latent Space Interpolation

11. More Results

12. Summary

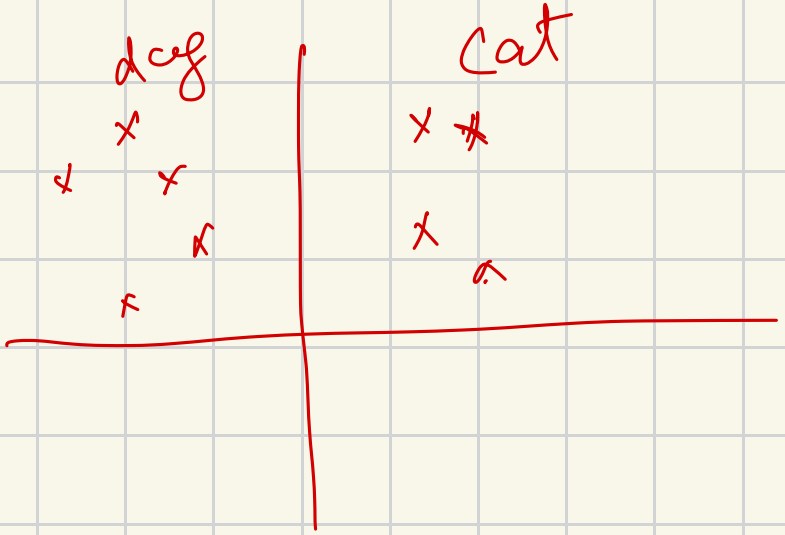
GANs: Objectives

1. **Understand the fundamental concepts and motivations behind GANs.**
   - 1.1 Learn why GANs were introduced and what problems they aim to solve in generative modeling.
2. **Explore the mathematical formulation and training process of GANs.**
   - 2.1 Study the minimax objective and the roles of the generator and discriminator.
   - 2.2 Analyze how GANs compare distributions via samples.
3. **Examine objective functions and training challenges.**
   - 3.1 Review standard and alternative objective functions.
   - 3.2 Identify common problems in GAN training, such as mode collapse and instability.
4. **Survey popular GAN variants and latent space concepts.**
   - 4.1 Wasserstein GANs, Conditional GANs, CycleGANs, StyleGANs.
   - 4.2 Understand the role and selection of latent spaces in GANs.

GANs: Learning Outcomes

After this lecture, you will be able to:

1. Explain the motivation and core principles of GANs.
2. Describe the minimax objective and the training process involving generator and discriminator.
3. Analyze and compare different GAN objective functions and training challenges.
4. Identify and contrast key GAN variants such as Wasserstein GANs, Conditional GANs, CycleGANs, and StyleGANs.
5. Discuss the role of latent spaces and their impact on GAN performance.
6. Assess the applications, results, and limitations of GAN-based models.
7. Summarize key concepts and advancements in the field of GANs.



AE

GAN → maximum likelihood

VAE

random noise

↓

Trained Decode

↓

Output.

$X_{Data} = [X_i]^N$

$\max_\theta \log P(X)$

① produce samples

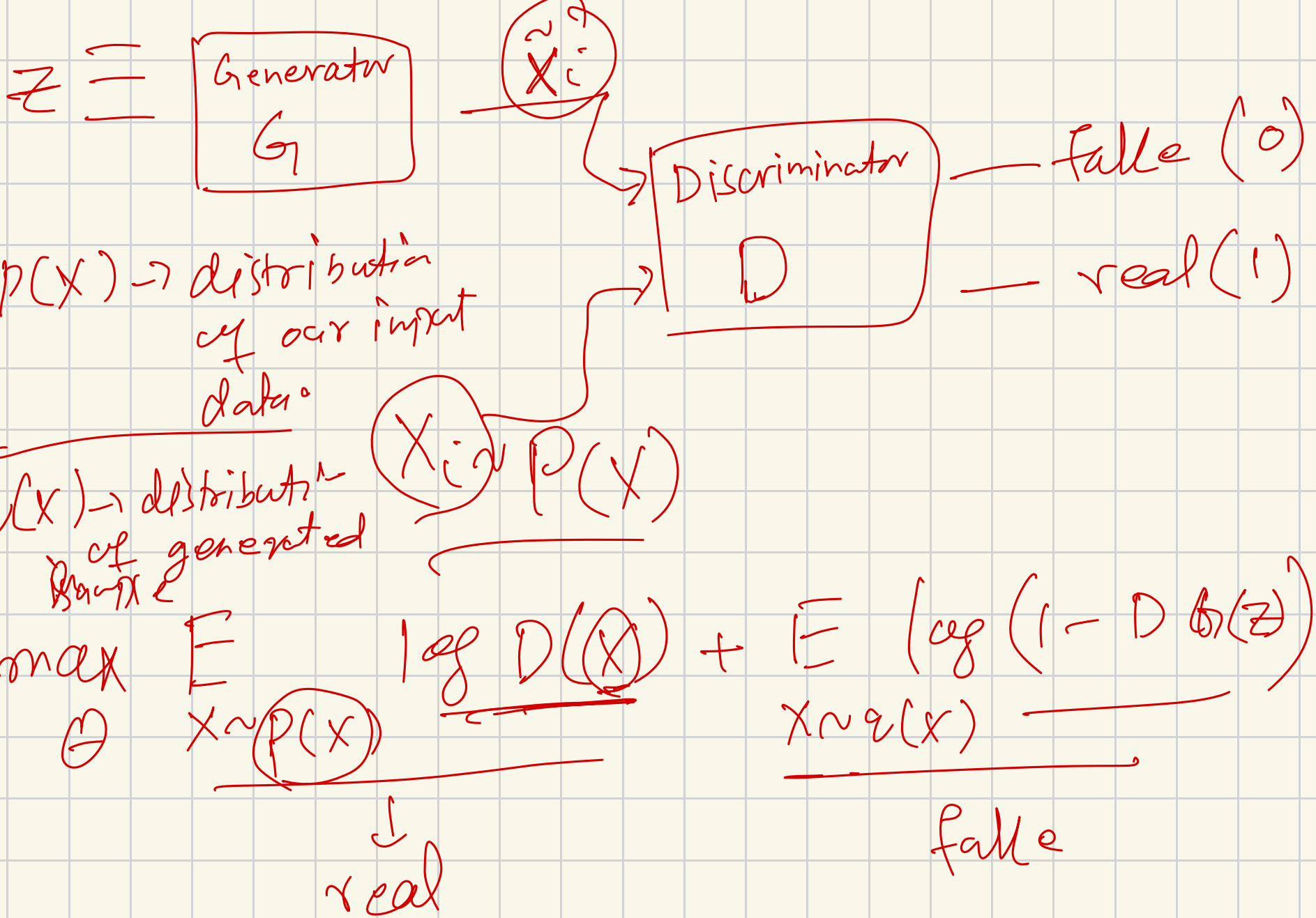
② validate
see if they're closer to real
in your data

$Q \to \tilde{X}$

$z \to \underline{\text{decoder}} \to \tilde{X}$
VAE

Train a classifier

Real Fake


z
Generator
G
$\tilde{X}_i$
Discriminator
D
fake (0)
real (1)
p(X) → distribution of our input data.
q(X) → distribution of generated data
$X_i \sim p(X)$
$\max_\theta \; \mathbb{E}_{X \sim p(X)} \log D(X) + \mathbb{E}_{X \sim q(X)} \log(1 - D_\theta(z))$
real
fake

generator

discriminator

$$\min_{\phi} \left\{ \max_{\theta} \; \underset{x \sim p(x)}{E} \log D(x) + \underset{x \sim q(x)}{E} \log\left(1 - D(G(z))\right) \right\}$$

① maximization of expectation.

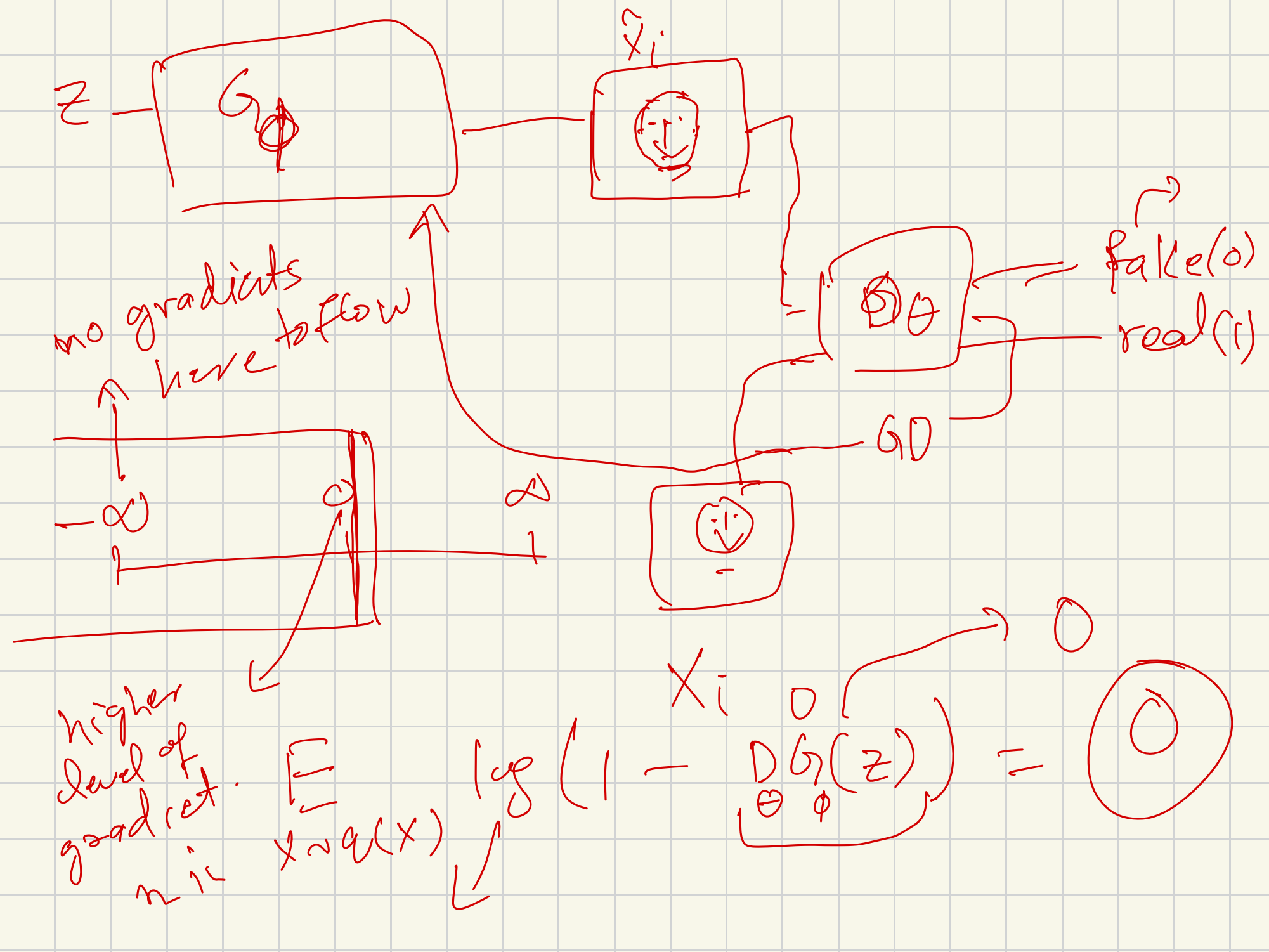
$$\max_{\theta} \underbrace{\mathop{E}_{x\sim p(x)} \log D_\theta(x) + \mathop{E}_{x\sim q(x)} (1 - \log D_\theta(z))}_{T}$$

② minimization step.

$$\min_{\phi} \left(\max_{\theta} \mathop{E}_{x\sim p(x)} \log D_\theta(x) + \mathop{E}_{x\sim q(x)} \log(1 - D_\theta(G_\phi(z))) \right)$$

real

fake


z
G_ϕ
$\hat{x}_i$
D_θ
fake(0)
real(1)
GD
no gradients flow here
$-\alpha$
α
1
higher level of gradient.
$\min_G \mathbb{E}_{x \sim q(x)} \log(1 - D_\theta(G_\phi(z)))$
x_i
0
0

$$\max_{\phi} \log D_{\phi}(G_{\phi}(z)) = \log(1 - D G(z))$$

$$\max \underbrace{\mathop{E}_{x \sim p(x)} \log D(x) + \mathop{E}_{x \sim q(x)} \log(D G(z))}$$

Dis → gradient ascent

Gen → a descent.

Generative Adversarial Networks: Motivation

So far, we have studied generative models based on maximizing likelihood or its approximations:

1. **Autoregressive Models**: $p_\theta(x) = \prod_{i=1}^{N} p_\theta(x_i|x_{<i})$
2. **Variational Autoencoders (VAEs)**: $p_\theta(x) = \int_z p_\theta(x|z)p_\theta(z)$
3. **Normalizing Flow Models**: $p_X(x;\theta) = p_Z(f_\theta^{-1}(x)) \left|\det\left(\frac{\partial f_\theta^{-1}(x)}{\partial x}\right)\right|$

Problem with Traditional Generative Models:

- They often require complex approximations or sampling methods.
- They can produce blurry or unrealistic samples.
- They depend on explicit density estimation, which can be computationally expensive.

What if we give up on explicitly modeling density, and just want ability to sample?

Generative Adversarial Networks: Motivation (cont.)

Generative Adversarial Networks (GANs) provide a solution to this problem by introducing a novel approach to generative modeling:

- (introduced by Ian Goodfellow et al., 2014) do not require explicit density estimation or Markov chains.
- consist of two neural networks—a **generator** and a **discriminator**—in a 2-player (zero-sum) game.
- The adversarial setup enables the generator to learn complex data distributions and produce high-fidelity, realistic samples.

Comparing Distributions via Samples

Given a finite set of samples from two distributions $S_1 = \{x \sim P\}$ and $S_2 = \{x \sim Q\}$, how can we tell if these samples are from the same distribution? (i.e., $P = Q$?)

$S_1 = \{\mathbf{x} \sim P\}$ $S_2 = \{\mathbf{x} \sim Q\}$

GANs aim to replicate data distributions, but we rarely have access to the true probability distribution function. Instead, we compare samples from the real data distribution with those generated by the model.

Comparing Distributions via Samples (cont.)

- Let's consider a test statistic T for this purpose.
- Test statistic T compares S_1 and S_2 e.g., difference in means, variances of the two sets of samples.
- **Key observation**: Test statistic is likelihood-free since it does not involve the densities P or Q (only samples)

GANs: Definition and Core Components

- Let $S_1 = D = \{x \sim p_{\text{data}}\}$ and $S_2 = \{x \sim p_\theta\}$.
- **Idea**: Train the generative model to minimize a two-sample test objective between S_1 and S_2, i.e., the generator.

GANs: Definition and Core Components

- Let $S_1 = D = \{x \sim p_{\text{data}}\}$ and $S_2 = \{x \sim p_\theta\}$.
- **Idea**: Train the generative model to minimize a two-sample test objective between S_1 and S_2, i.e., the generator.
- **Question**: How do we obtain a two-sample test objective?

GANs: Definition and Core Components

- Let $S_1 = D = \{x \sim p_{\text{data}}\}$ and $S_2 = \{x \sim p_\theta\}$.
- **Idea**: Train the generative model to minimize a two-sample test objective between S_1 and S_2, i.e., the generator.
- **Question**: How do we obtain a two-sample test objective?
- **Another Idea**: Train another neural network to discriminate between the two samples, i.e., the discriminator.

GANs: Definition and Core Components

- Let $S_1 = D = \{x \sim p_{\text{data}}\}$ and $S_2 = \{x \sim p_\theta\}$.
- **Idea**: Train the generative model to minimize a two-sample test objective between S_1 and S_2, i.e., the generator.
- **Question**: How do we obtain a two-sample test objective?
- **Another Idea**: Train another neural network to discriminate between the two samples, i.e., the discriminator.
- And Voila! That's how we arrive at a GAN. The remaining step is to define the training method.

GANs: Definition and Core Components

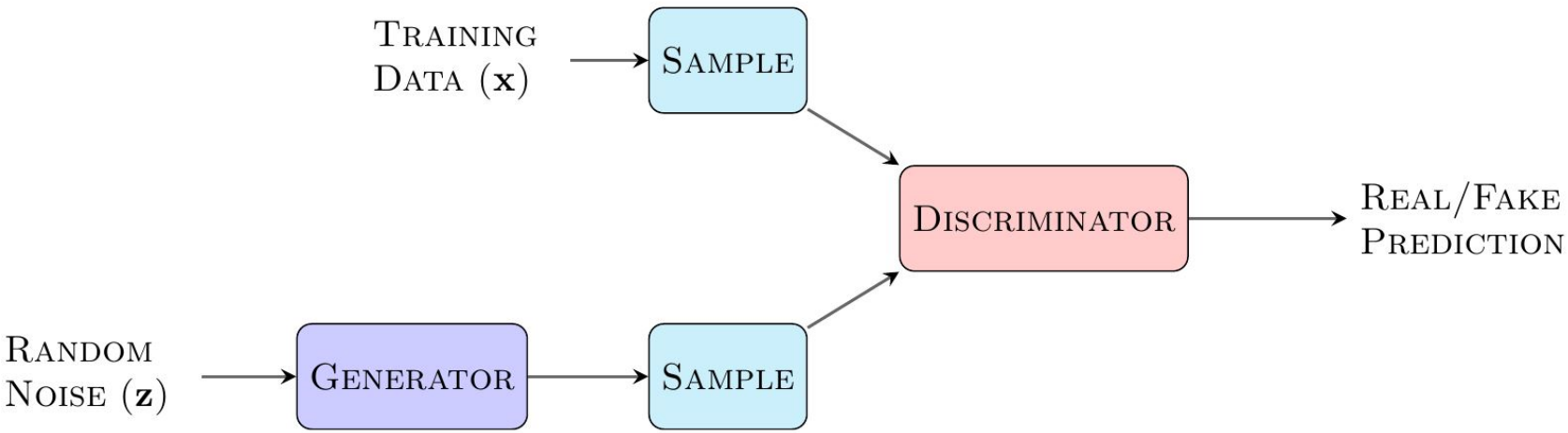


Figure 2: A GAN model diagram[1]

GANs: Definition and Core Components (cont.)

- Generative Adversarial Networks were introduced by Ian Goodfellow et al. (2014).
- The idea behind GANs is to train two networks jointly:
  - A **discriminator (D)** to classify samples as "real" or "fake".
  - A **generator (G)** to map a simple fixed distribution to samples that fool **D**.
- The approach is **adversarial** since the two networks have opposing objectives.

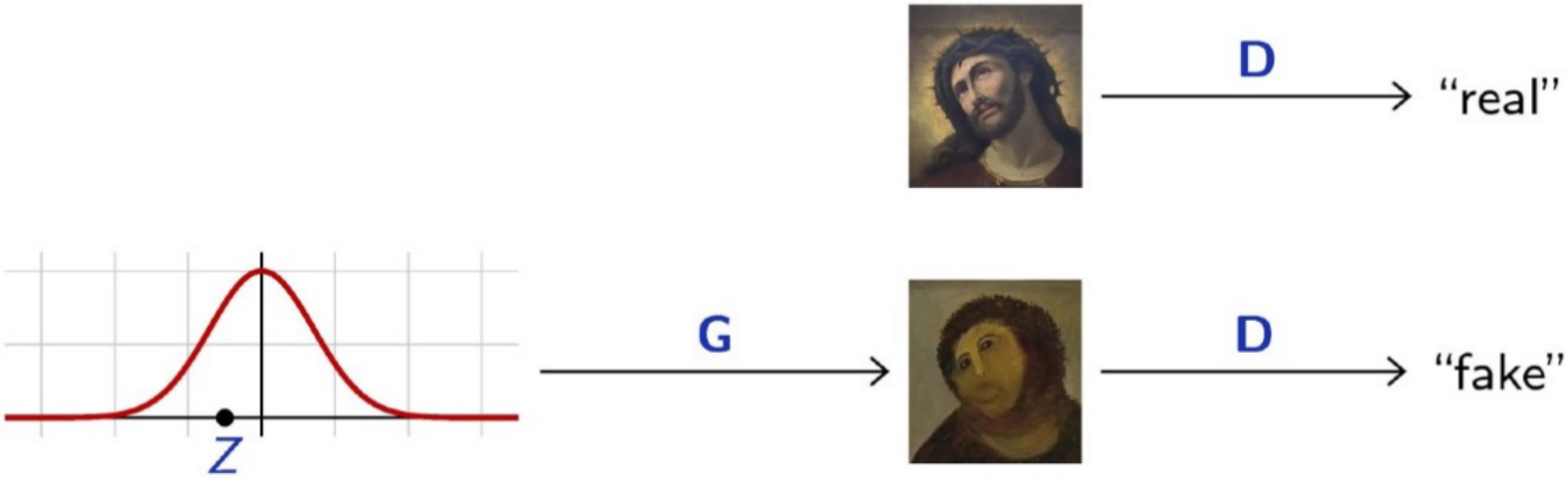


Figure 3: The generator transforms a simple distribution into samples resembling real data. The discriminator distinguishes between real and fake samples.

[1] https://github.com/JamesAllingham/LaTeX-TikZ-Diagrams

GANs: Training

- Both Discriminator and Generator are trained jointly in a min-max game.
- Minimax objective function:

$$\min_{\theta_G} \max_{\theta_D} = E_{x \sim p_{data}} log(\underbrace{D_{\theta_D}(x)}_{\text{Discriminator output for real data } x}) + E_{x \sim p(z)} log(1 - \underbrace{D_{\theta_D}(G_{\theta_G}(z))}_{\text{Discriminator output for generated fake data } G(z)})$$

- Discriminator θ_D wants to maximise objective such that $D(x)$ is close to 1 (real) and $D(G(z))$ is close to 0 (fake).
- Generator θ_G wants to minimise objective such that $D(G(z))$ is close to 1 (discriminator is fooled into thinking generated $G(z)$ is real).

GANs: Training (cont.)

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

GANs: Training (cont.)

The Adversarial Process:

1. Discriminator is trained to distinguish real from fake.
2. Generator is trained to fool the discriminator.
3. Repeat: Alternate steps until convergence.

Training Tips:

- Use batch normalization.
- Avoid sigmoid output in discriminator.
- Use Leaky ReLU in discriminator.

Practical Issues:

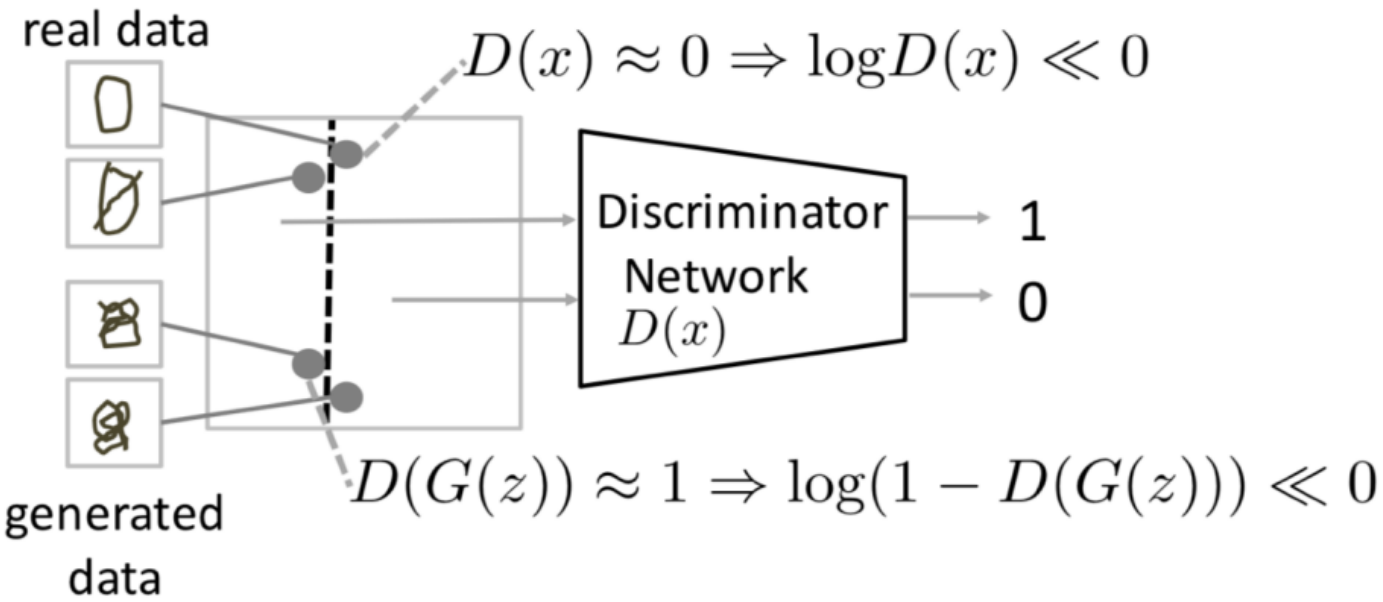
- Discriminator becomes too strong.
- Gradient vanishing for generator.
- Careful balance of training speeds is necessary.

Understanding the Objective function

The goal of GANs is to find a **Nash equilibrium** between the generator and discriminator. The generator tries to minimize the probability of the discriminator correctly classifying fake data.

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

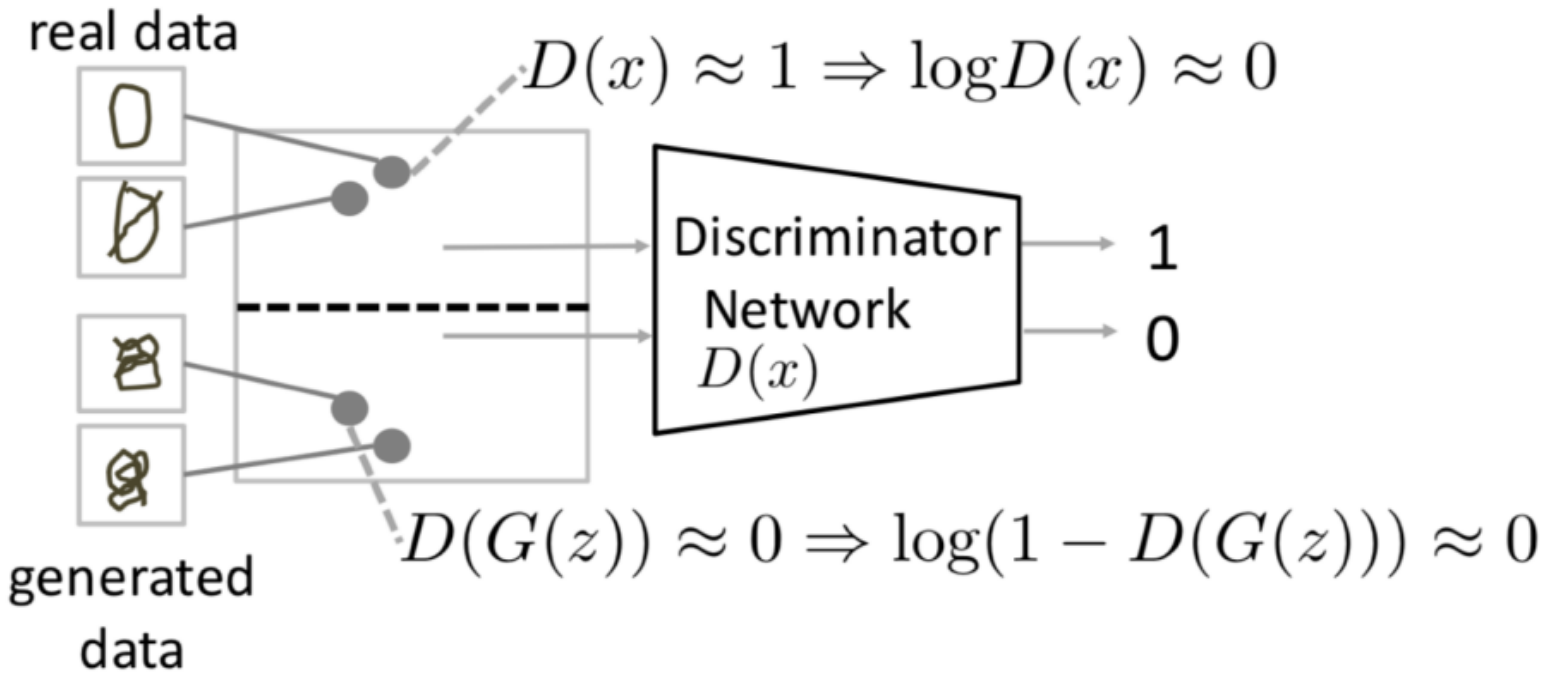
$D(x)$ should be 1 $\quad$ $D(G(z))$ should be 0



Understanding the Objective function (cont.)

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

$D(x)$ should be 1 $D(G(z))$ should be 0

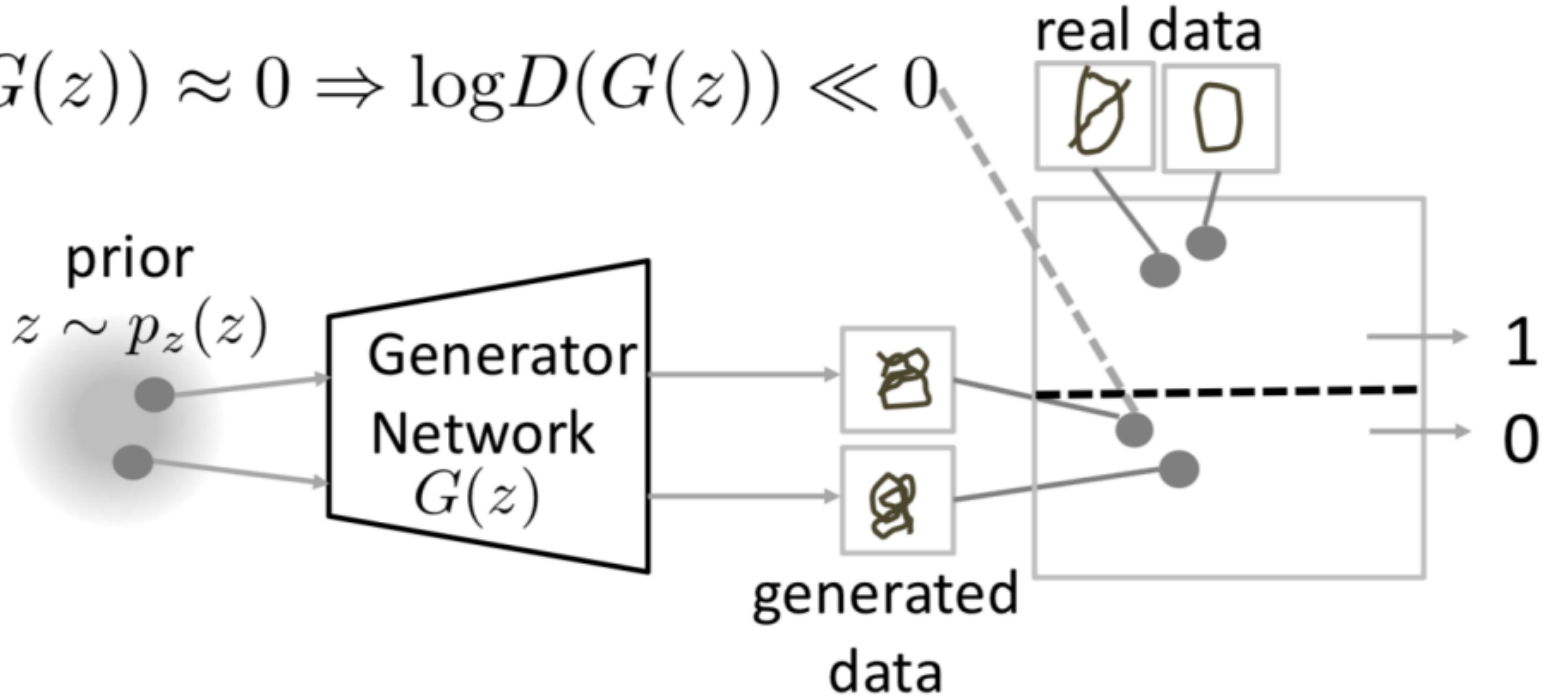


Understanding the Objective function (cont.)

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

$D(G(z))$ should be 1

$$D(G(z)) \approx 0 \Rightarrow \log D(G(z)) \ll 0$$

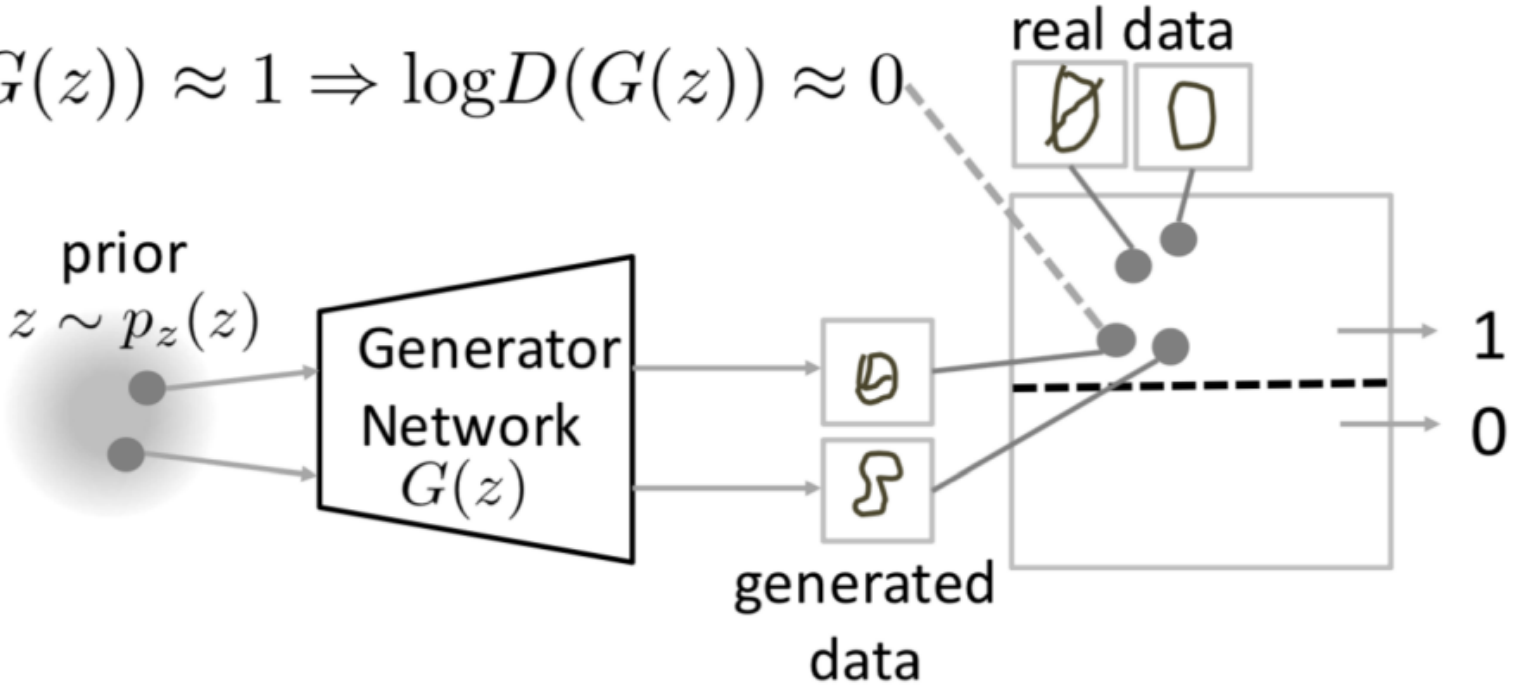


Understanding the Objective function (cont.)

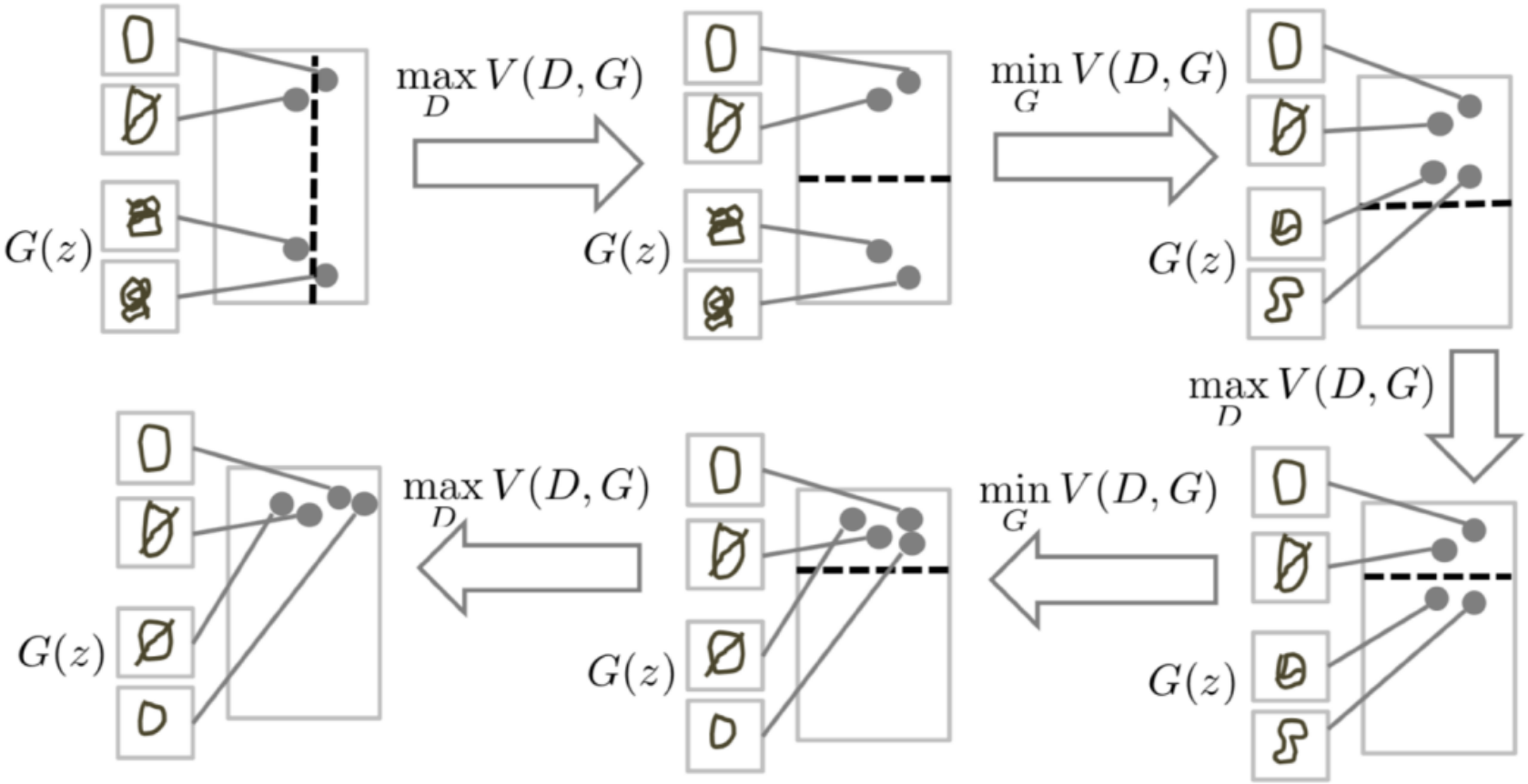
$$\max_G(\mathbb{E}_{z\sim p_z(z)}[\log(D(G(z))])$$

$D(G(z))$ should be 1

$D(G(z)) \approx 1 \Rightarrow \log D(G(z)) \approx 0$



Understanding the Objective function (cont.)



[1] https://www.slideshare.net/ckmarkohchang/generative-adversarial-networks

GANs - Interactive Demo

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

GANs - Results

2014

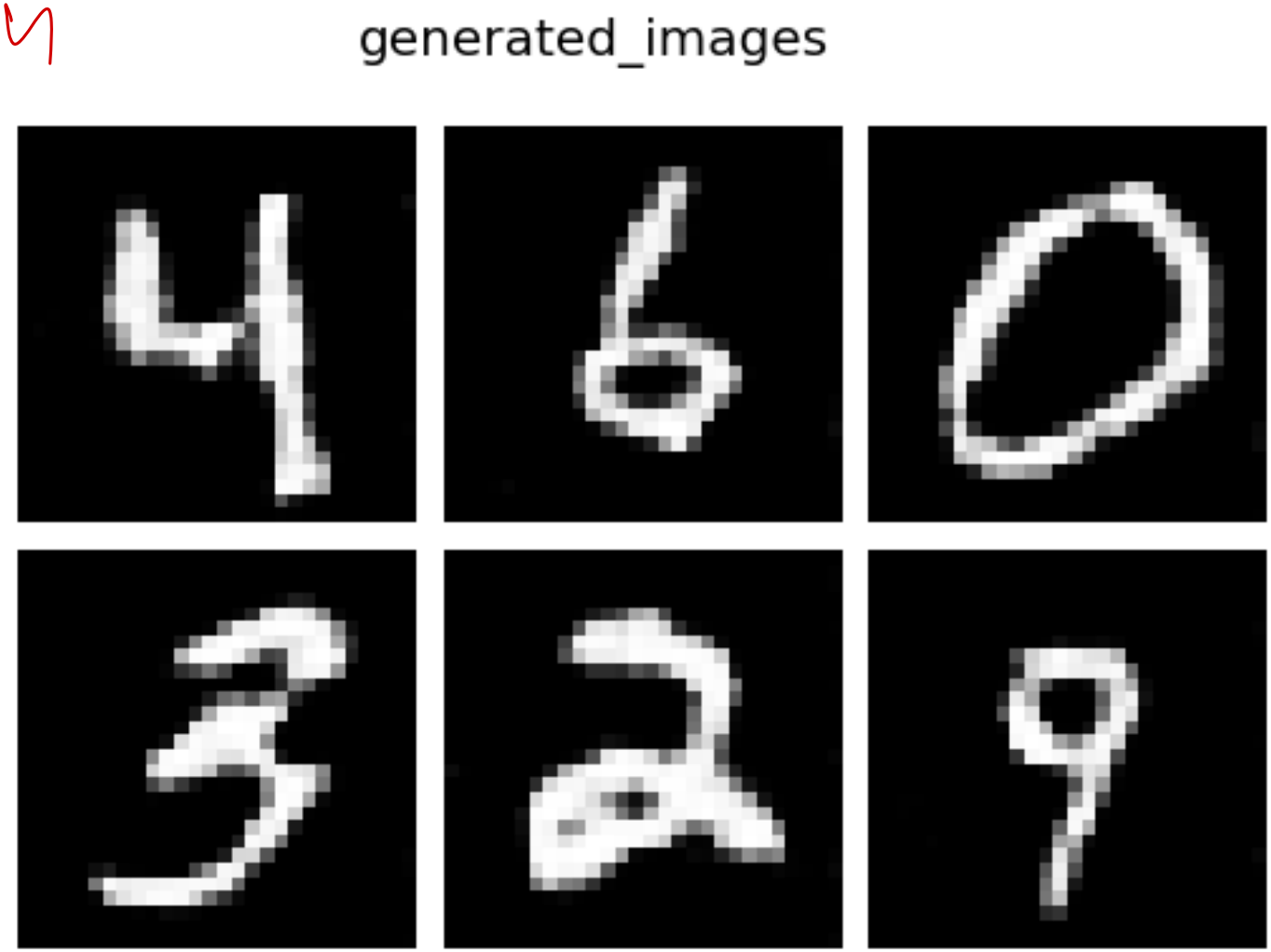


Figure 4: GAN generated samples for MNIST digits dataset

GANs - Results (cont.)

2015

Figure 5: GAN generated samples for bedroom images

GANs - Results (cont.)

Figure 6: GAN generated samples for anime character faces

Problems with GANs

- Vanishing Gradients
  - When the discriminator is perfect, we are guaranteed with $D(x) = 1, \forall x \in p_{data}$ and $D(x) = 0, \forall x \in p_G$.
  - The loss function drops to zero, resulting in no gradient for updates.
  - **Solution**: Perform gradient ascent on the generator, i.e., use a different objective:

$$\max_{\theta_G} E_{x \sim p(z)} log(D_{\theta_D}(G_{\theta_G}(z)))$$

Problems with GANs (cont.)

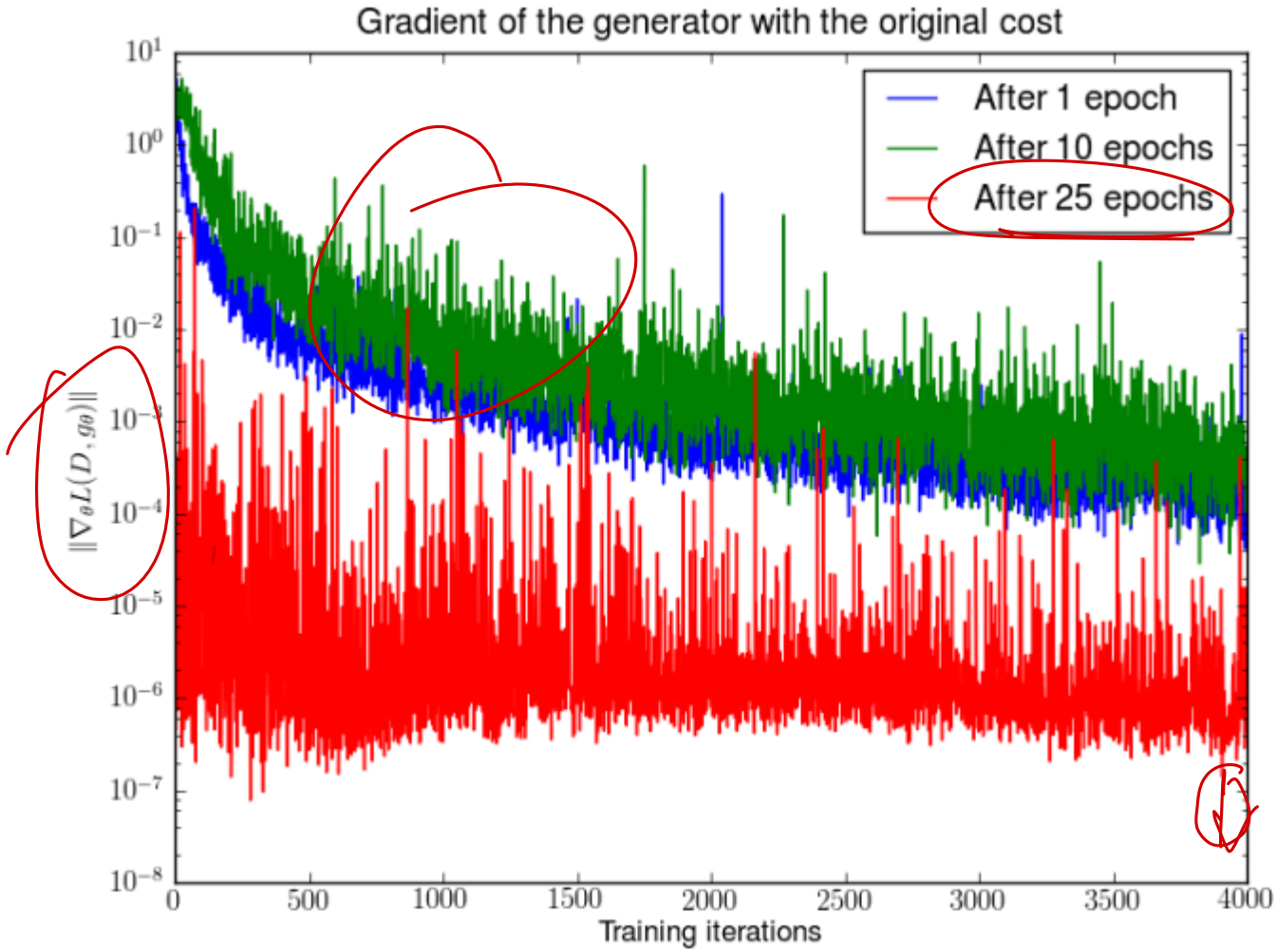
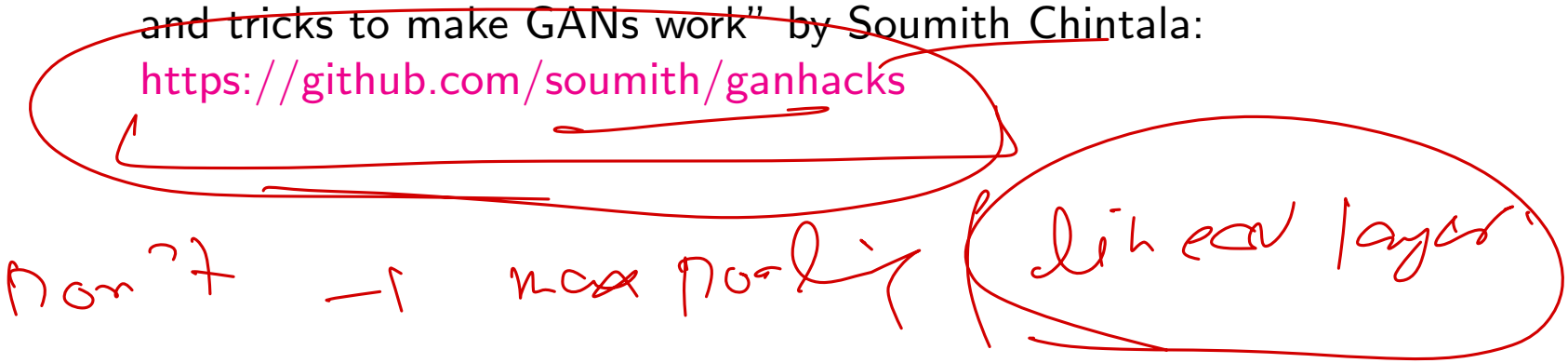


Figure 7: Vanishing gradient in GANs (Image source: Arjovsky and Bottou, 2017)

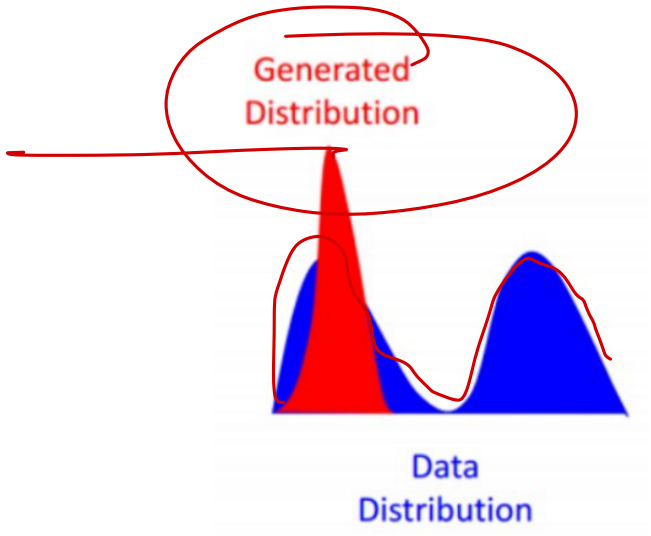
Problems with GANs (cont.)

- Difficulty in achieving Nash equilibrium
  - GANs involve training two models simultaneously to reach a Nash equilibrium in a two-player non-cooperative game. However, since each model updates its cost independently, convergence is not guaranteed.
  - For practical tips on training GANs, see "How to Train a GAN? Tips and tricks to make GANs work" by Soumith Chintala: https://github.com/soumith/ganhacks



Problems with GANs (cont.)

- Mode collapse
  - The generator aims to fool the discriminator D into classifying its outputs as real. If the generator G finds a single output that consistently fools D, it may repeatedly produce that output, leading to mode collapse.
  - Solutions to mode collapse are mostly empirical, including alternative architectures, modified GAN losses, and additional regularization terms.



Problems with GANs (cont.)

Figure 8: GAN mode collapse on MNIST digits dataset

GAN Variants

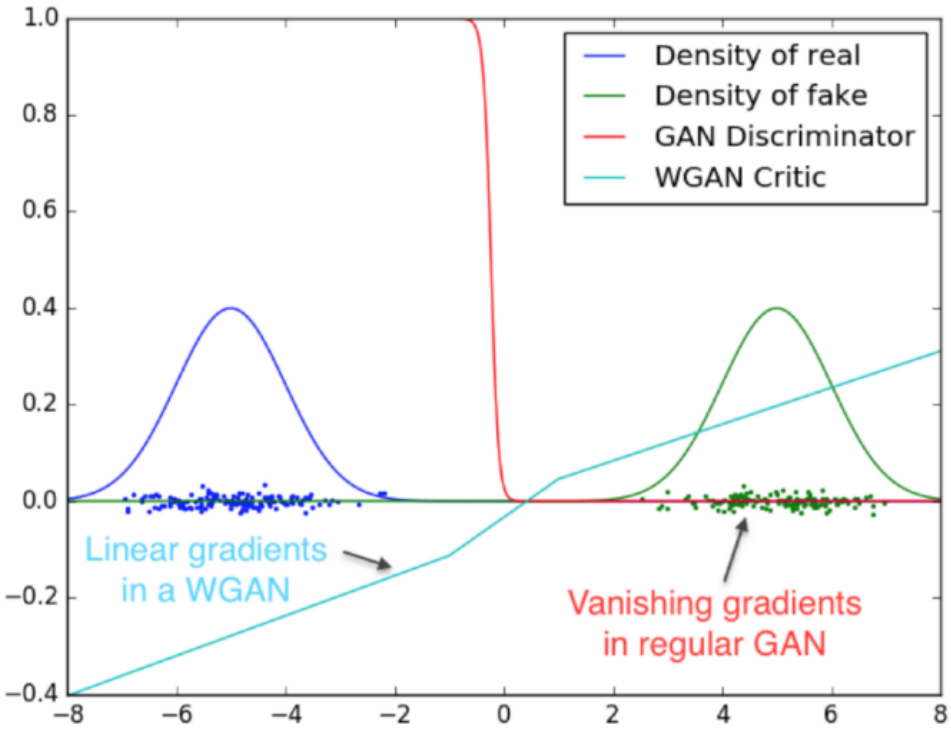
- Since the introduction of GANs, extensive research has led to many variants.
- A comprehensive list of GAN variants can be found here.
- We will discuss the following variants:
  - **Wasserstein GAN (WGAN):**
    - Introduces Wasserstein distance as the objective.
    - Replaces sigmoid with linear output.
    - Uses weight clipping or gradient penalty.
  - **Conditional GAN (cGAN):**
    - Conditions both the generator and discriminator on additional information (e.g., class labels).
    - Useful for targeted generation.

GAN Variants (cont.)

- **CycleGAN:**
  - Enables unpaired image-to-image translation.
  - Uses cycle consistency loss to ensure invertibility.
- **StyleGAN:**
  - Developed by NVIDIA for high-fidelity face generation.
  - Introduces style vectors at each generator layer.
  - Allows fine-grained control over image features.

Wasserstein GAN

- Wasserstein GAN uses wasserstein distance instead of crossentropy loss.
- Wasserstein distance that has a smoother gradient everywhere.



Wasserstein Distance

- Intuitively, it is the shovels of earth moved to make two distributions look alike.

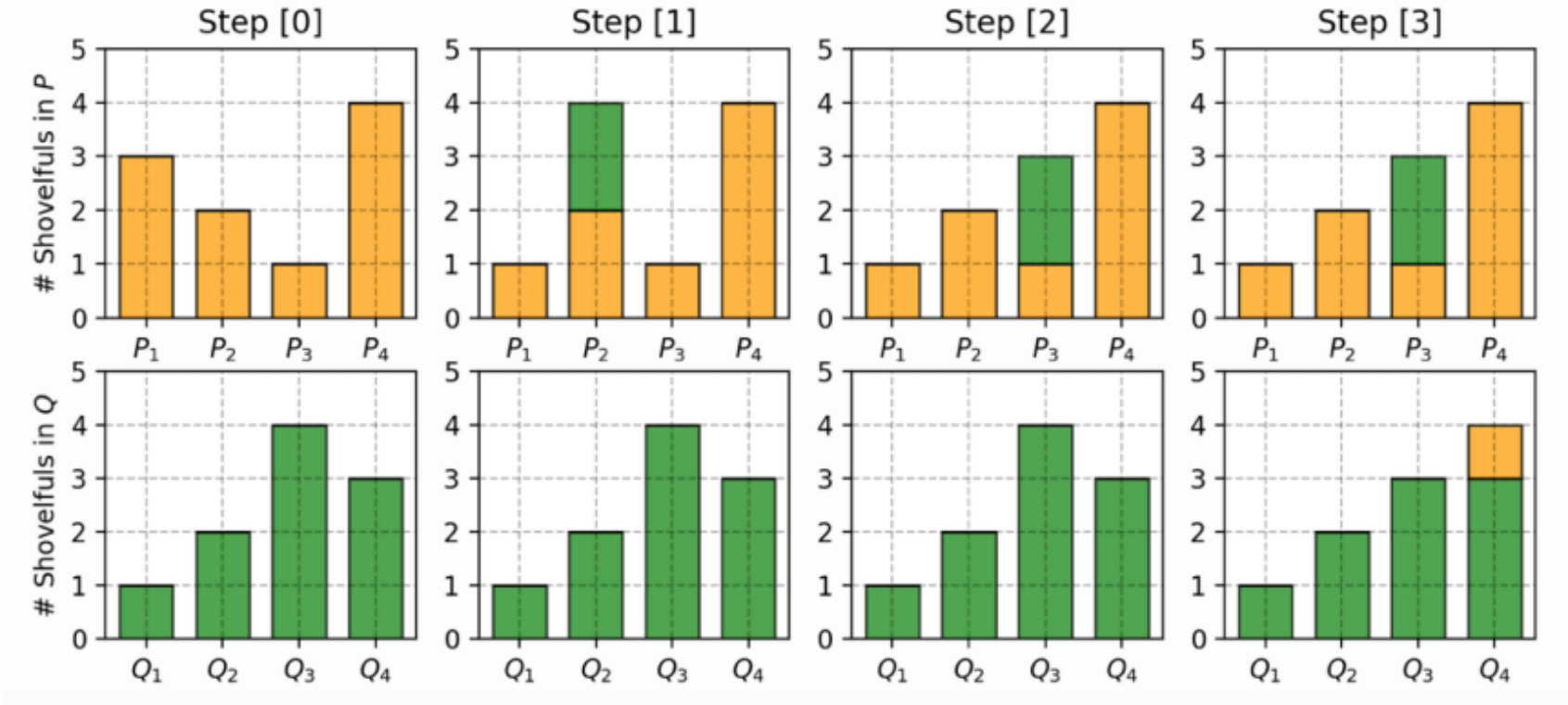
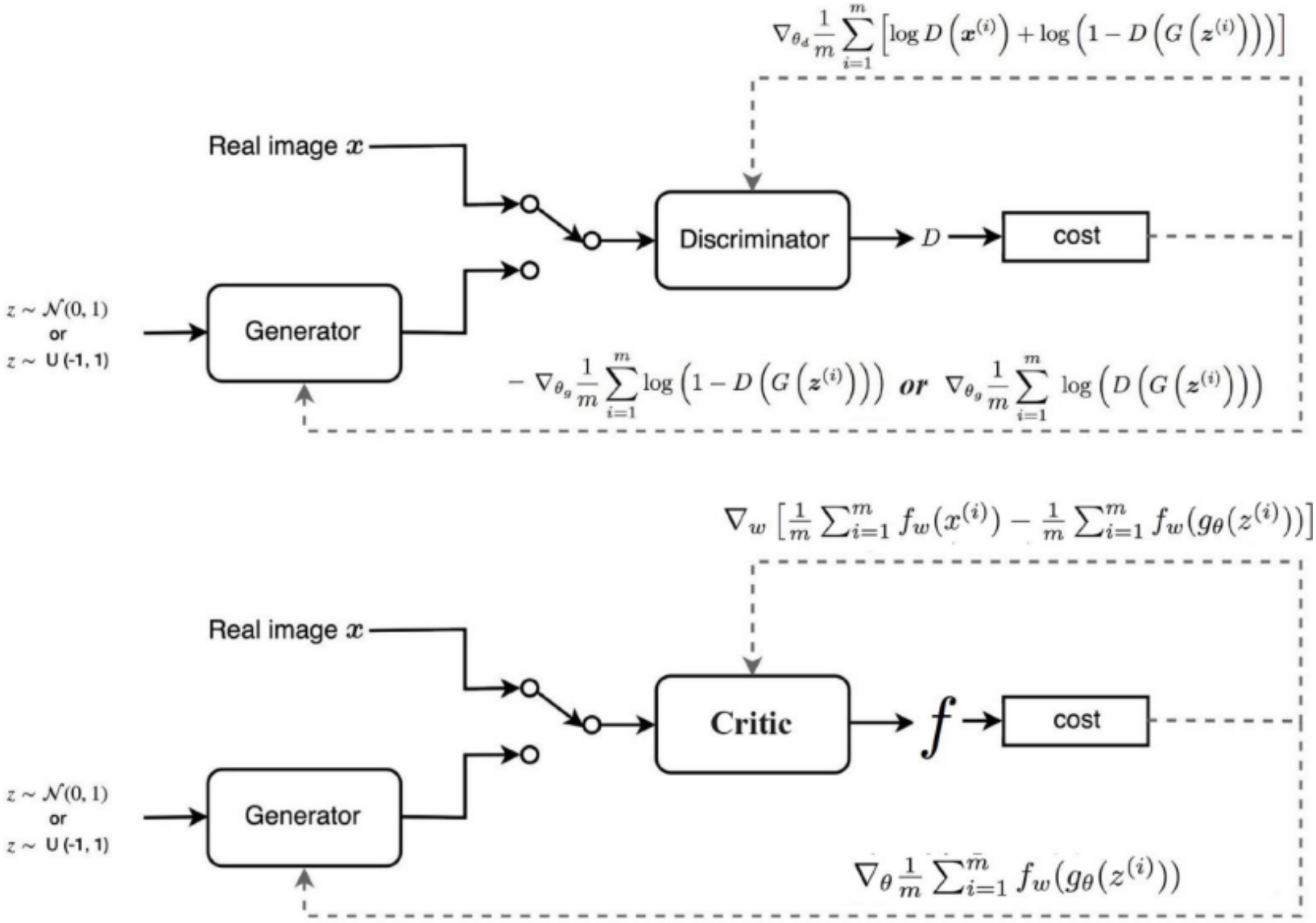
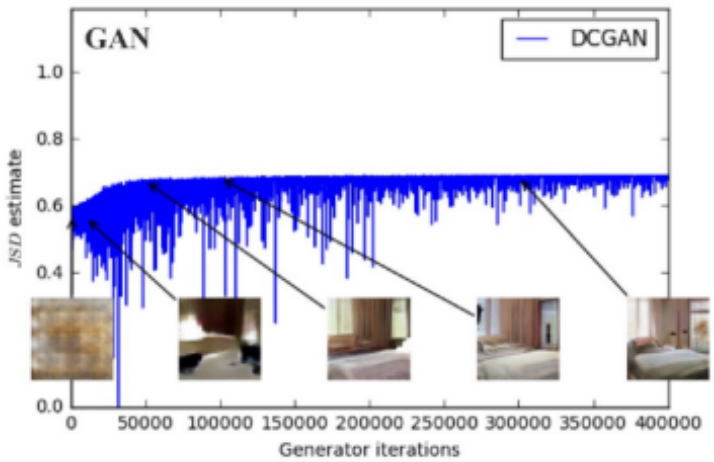


Figure 9: Step by Step plan of moving dirt between piles P and Q to make them match

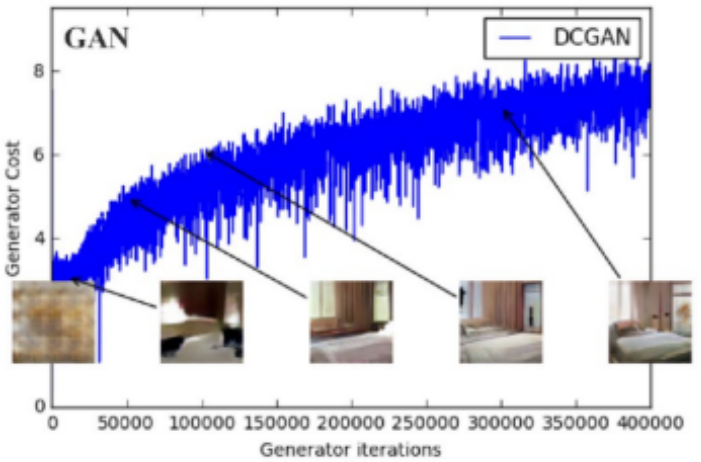
WGAN


$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \right]$
Real image x
Discriminator
D
cost
$z \sim \mathcal{N}(0, 1)$
or
$z \sim$ U (-1, 1)
Generator
$-\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)$ or $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(D\left(G\left(z^{(i)}\right)\right)\right)$
$\nabla_w \left[\frac{1}{m} \sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)})) \right]$
Real image x
Critic
f
cost
$z \sim \mathcal{N}(0, 1)$
or
$z \sim$ U (-1, 1)
Generator
$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$

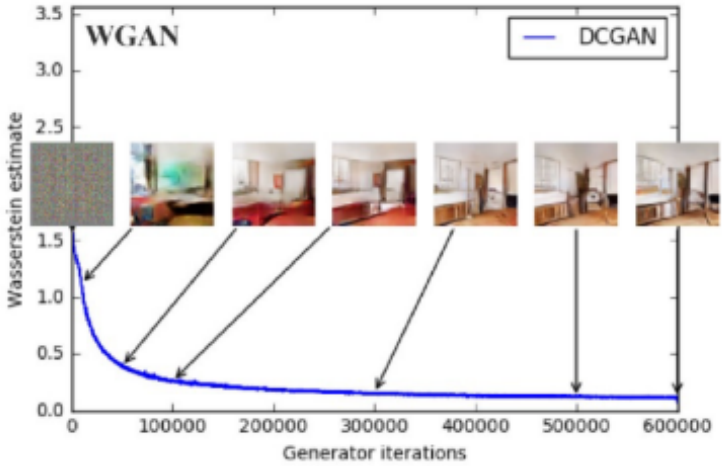
WGAN (cont.)



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



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



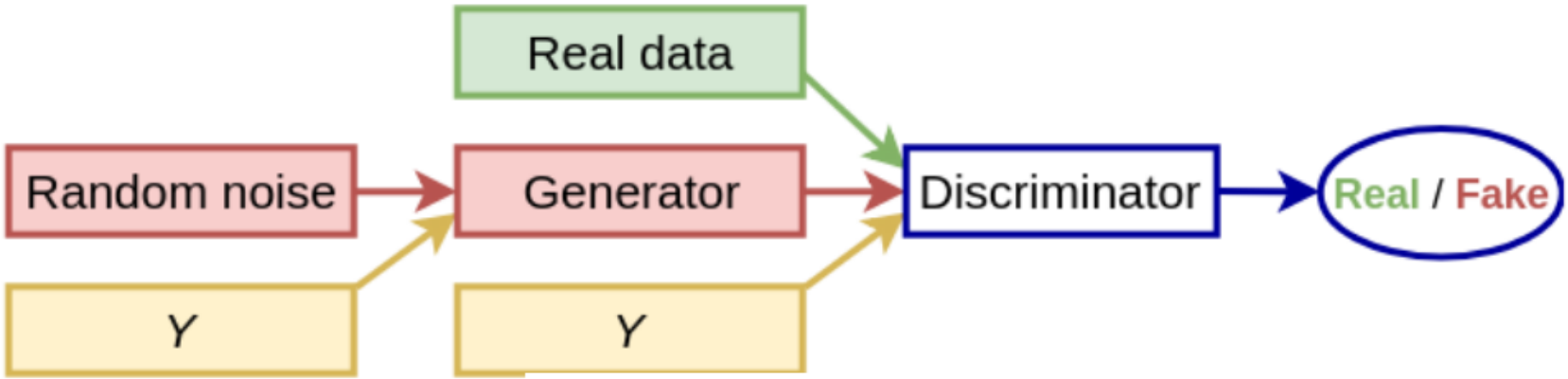
$$\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$$

WGAN (cont.)

Figure 10: WGAN generation results on bedroom images

Conditional GAN

- In addition to random noise, a condition is added to the generator input.
- The discriminator also receives this condition.



Conditional GAN (cont.)

- Discriminator Loss:

$$\mathcal{L}^{(D)}\left(\theta^{(G)}, \theta^{(D)}\right) = -\mathbb{E}_{x\sim p_{data}}\left[\log D(x|y)\right] - \mathbb{E}_{z\sim p_z(z),\ y\sim p_{data}(y)}\left[\log(1 - D(G(z|y)))\right]$$

- Generator Loss:

$$\mathcal{L}^{(G)}\left(\theta^{(G)}, \theta^{(D)}\right) = -\mathbb{E}_z\left[\log D(G(z|y))\right]$$

Conditional GAN (cont.)

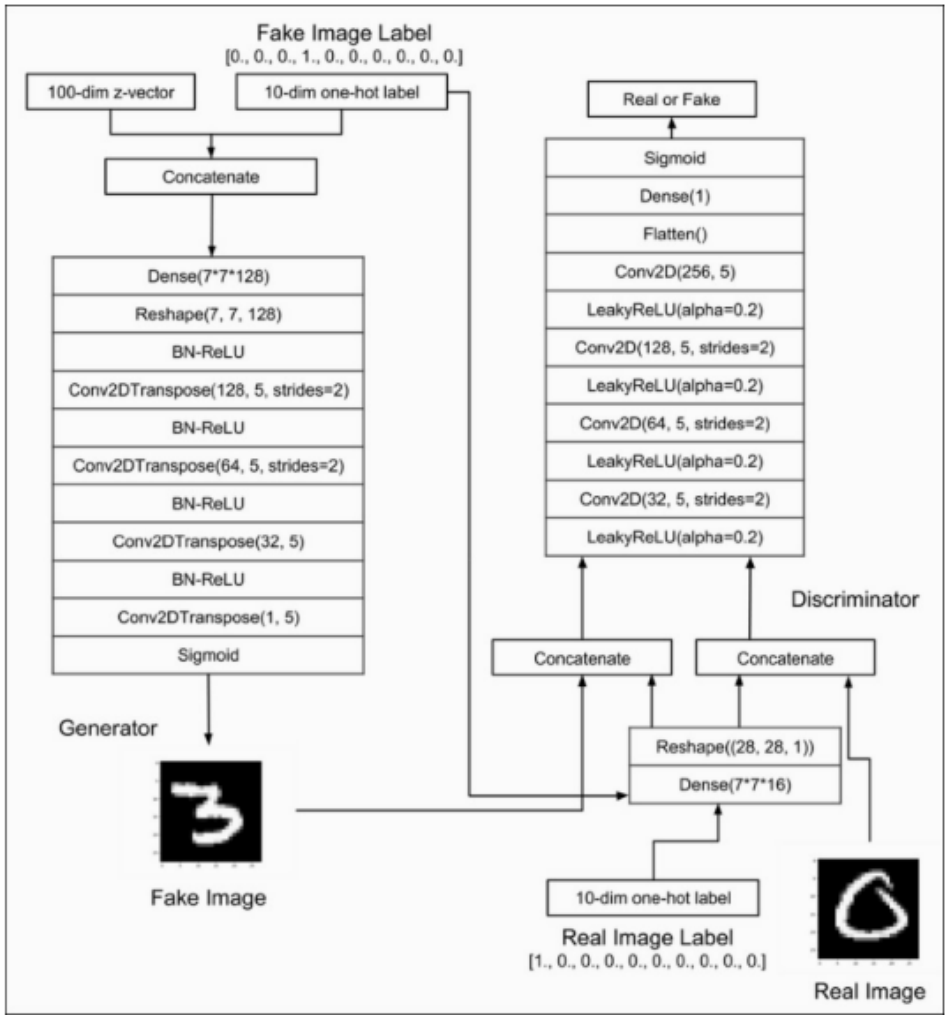


Figure 11: A conditional GAN architecture for the MNIST digits dataset.

Condional GAN - Image to Image

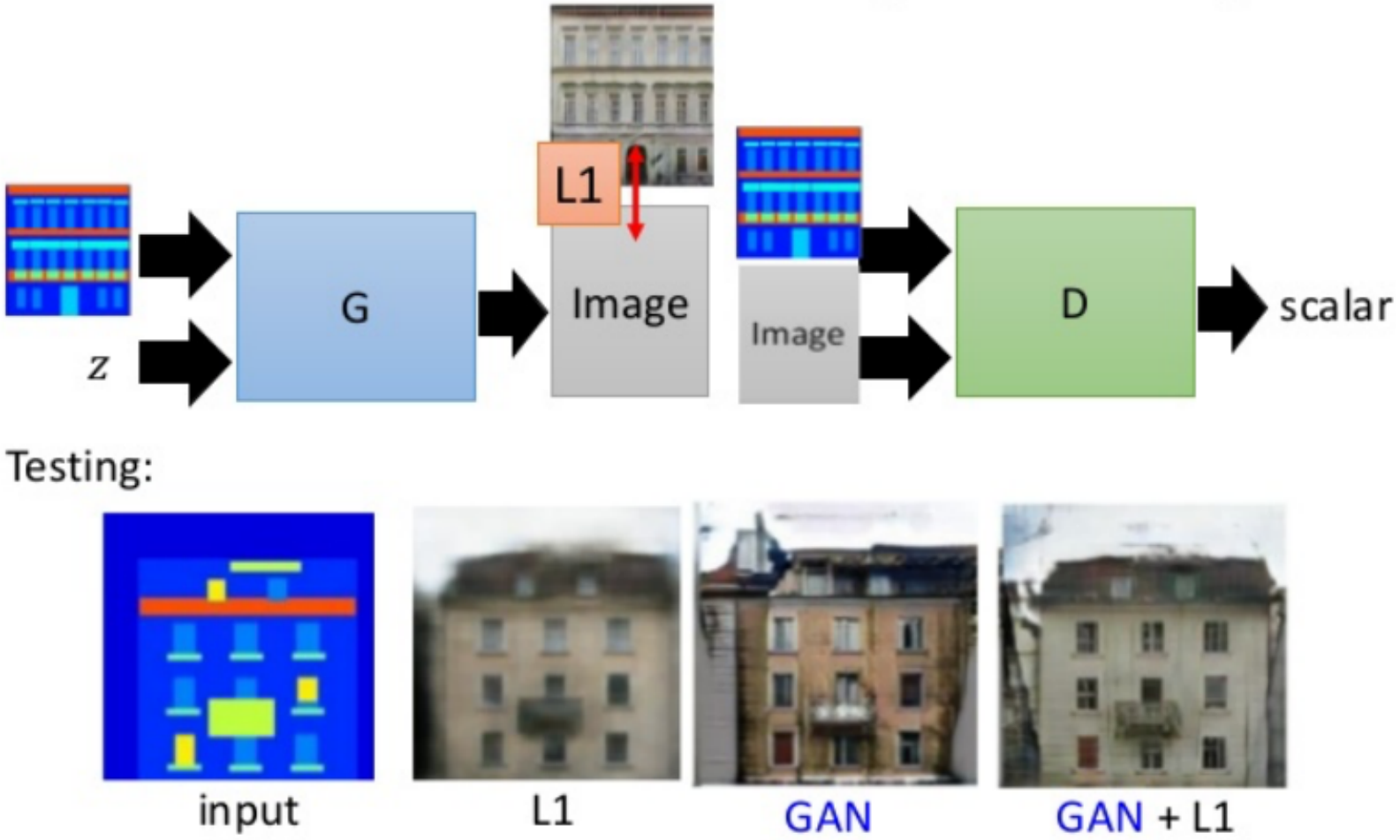


Figure 12: Using L1 loss in addition in GAN loss can help in image to image translation

Condional GAN - Image to Image (cont.)

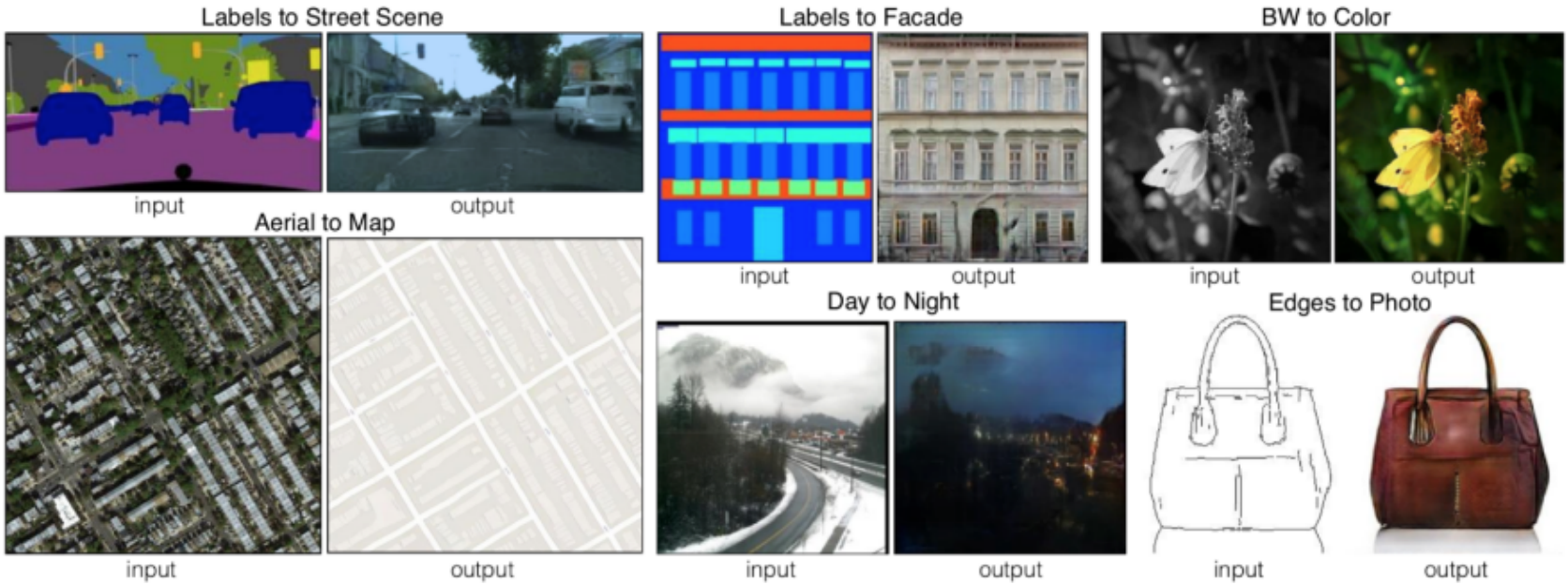
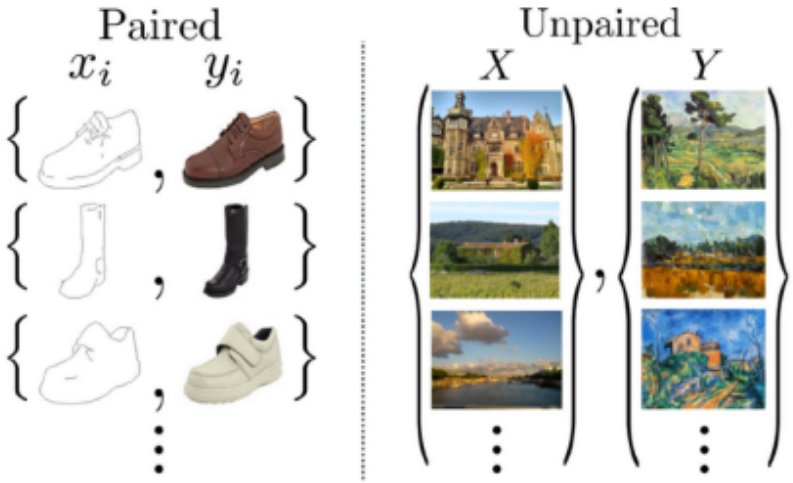
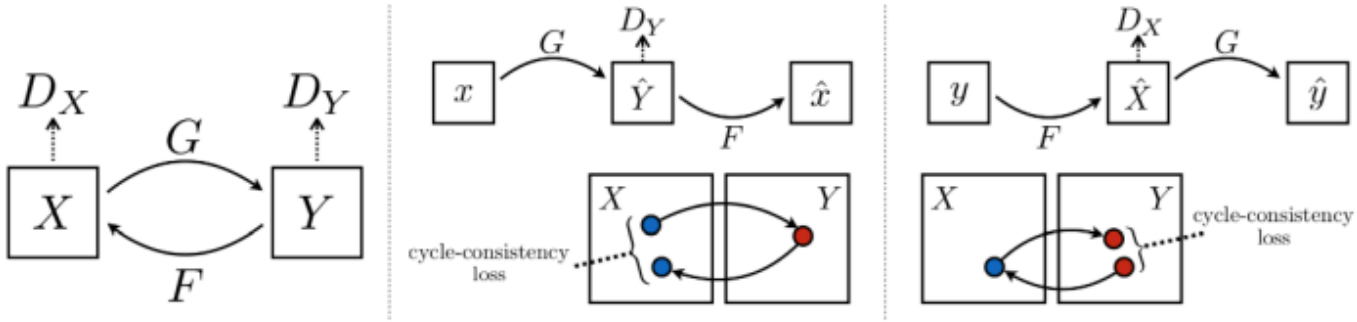


Figure 13: Image to Image translation with condional GANs

Cycle GANs

- Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, Efros, ICC7 2017





Cycle GANs (cont.)

$$C_{horse \to zebra} = horse \to G_{horse \to zebra} \to \hat{zebra} \to [D_{zebra}, G_{zebra \to horse}] \to \hat{horse}$$

Cycle GANs (cont.)

$$C_{zebra \to horse} = zebra \to G_{zebra \to horse} \to \hat{horse} \to [D_{horse}, G_{horse \to zebra}] \to \hat{zebra}$$

Cycle GANs (cont.)

Cycle GANs (cont.)

$G_{zebra \rightarrow horse}$

Cycle GANs (cont.)

$G_{horse \rightarrow zebra}$

Cycle GANs (cont.)

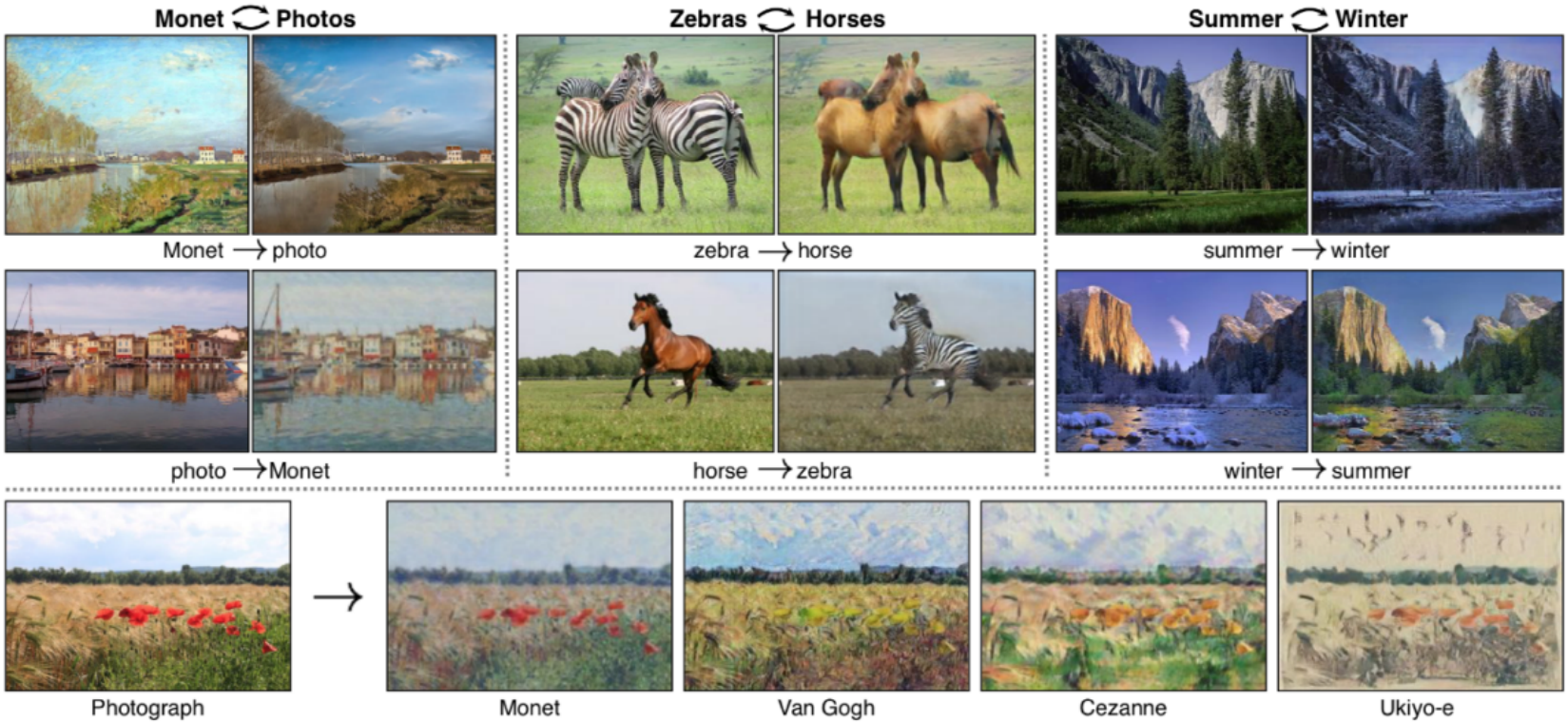
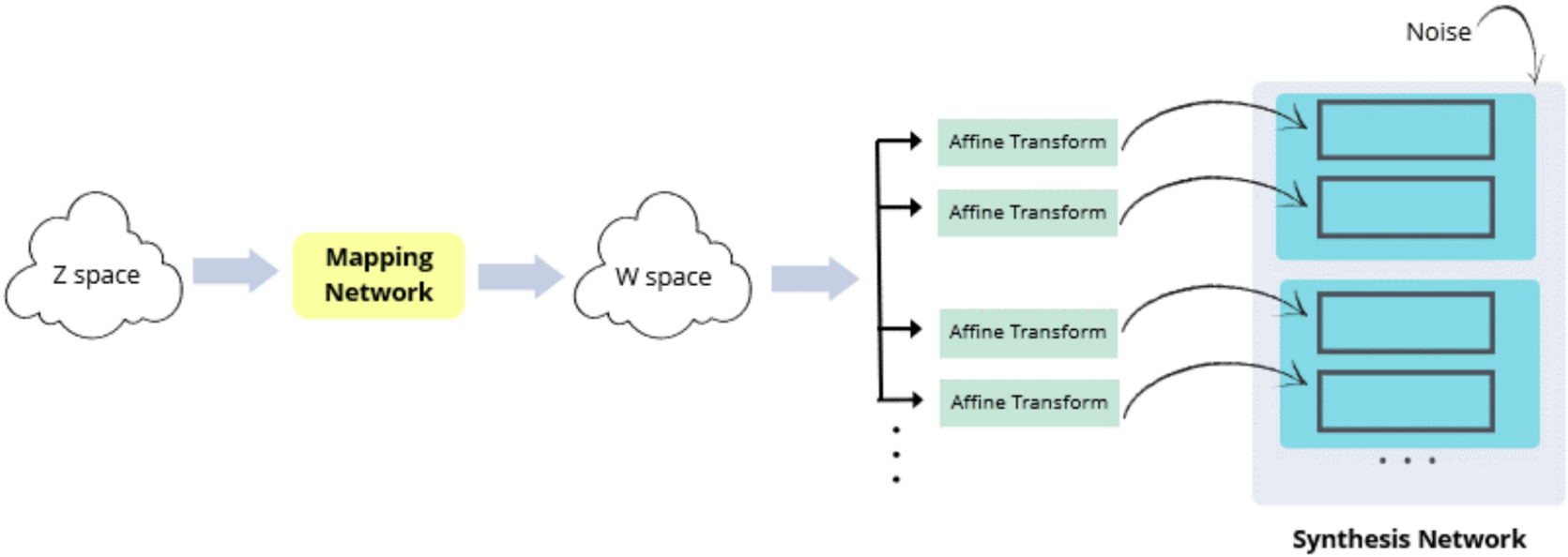


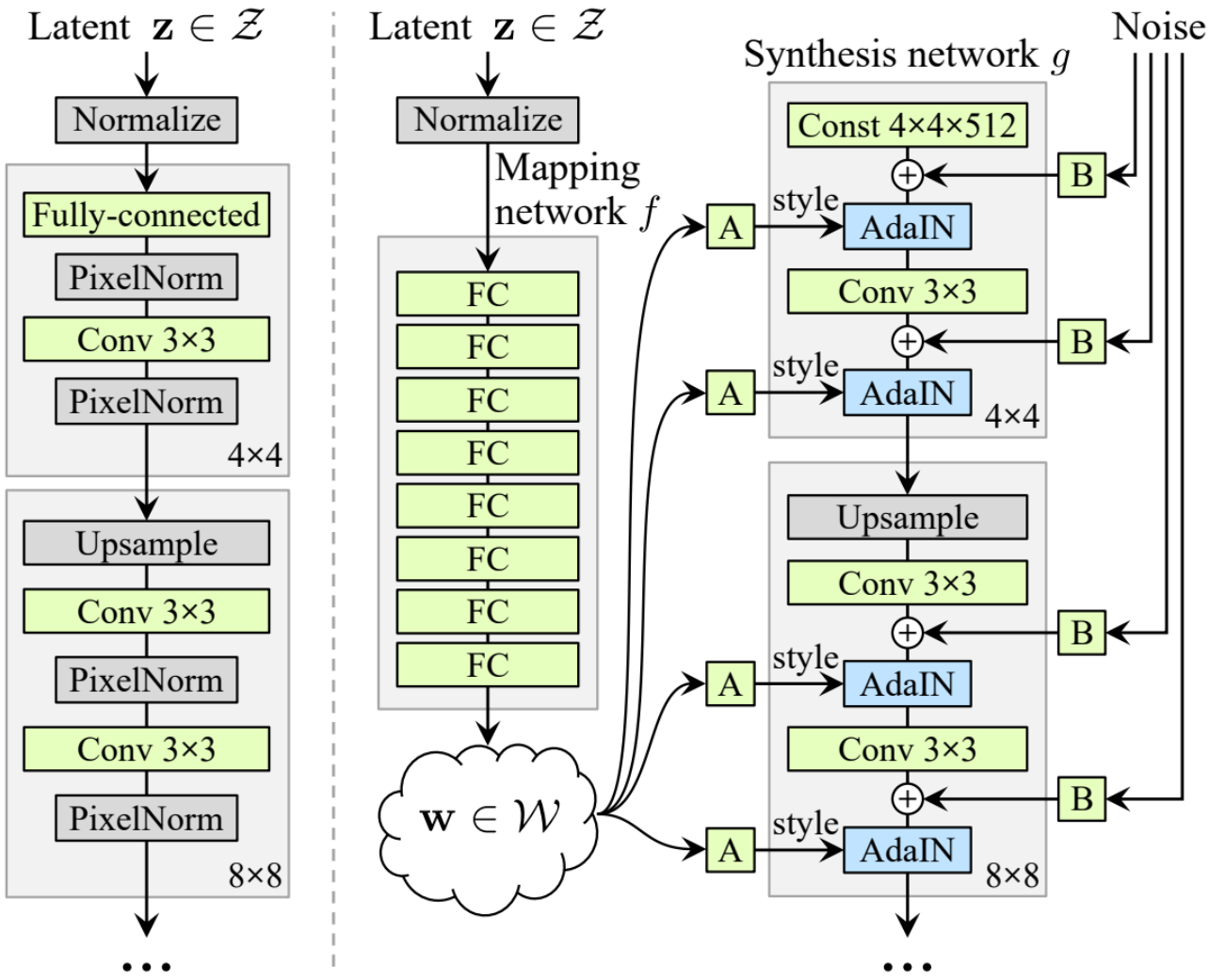
Figure 14: Image to Image translation with Cycle GANs

- Goal: Better disentanglement of features in latent space (**W** space)



[1] https://www.cs.unc.edu/ ronisen/teaching/fall2022/pdf_ectures/lecture6_gan2.pdf

Style GANs



(a) Traditional (b) Style-based generator

[1] https://arxiv.org/pdf/1812.04948

Style GANs

- **A** = learned affine transformation block for AdaIN (predicts y)
- **Ada**ptive **I**stance **N**ormalization (very effective in controlling styles)

$$\text{AdaIN}(x_i, y) = y_{s,i} \frac{x_i - u(x_i}{\sigma(x_i)} + y_{b,i}$$

- **B** = learned per-channel scaling factor for noise input.

Which Latent space to choose for embedding and editing?

أكاديمية كاوست
KAUST ACADEMY

- **Z** : 512 dimensional latent space (not good)
- **W** : 512 dimensional latent space (better but not perfect)
- **W+** : 18x512 dimensional latent space (after affine transformation **A** has been applied)
- **W** is better for editing.
- **W+** is better for reconstruction or embedding of real images.

Style GANs - Results

[1] https://arxiv.org/pdf/1812.04948

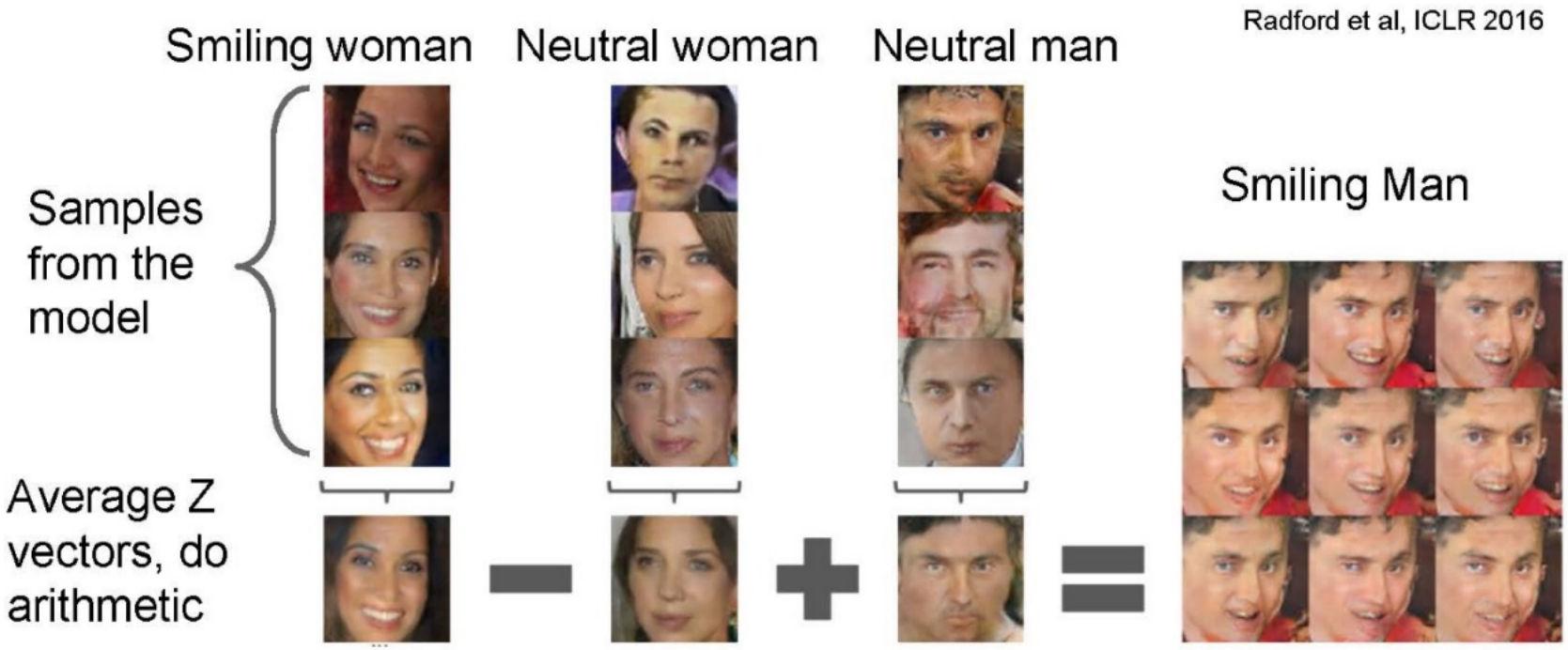


Figure 15: Latent space interploation with GANs

GANs - Latent space Interpolation (cont.)

Latent space interpolation with StyleGAN
(Demo by Xander Steenburgge)

https://colab.research.google.com/drive/1mH70YxGNInEaSOn0J8LsgkI-QOvsIb3MscrollTo=uEhxBvAR-7y3

Some more GAN results

Figure 16: Image Inpainting.
https://www.nvidia.com/en-us/research/ai-demos/

Some more GAN results (cont.)

Figure 17: Text to Image Synthesis with GANs

Some more GAN results (cont.)

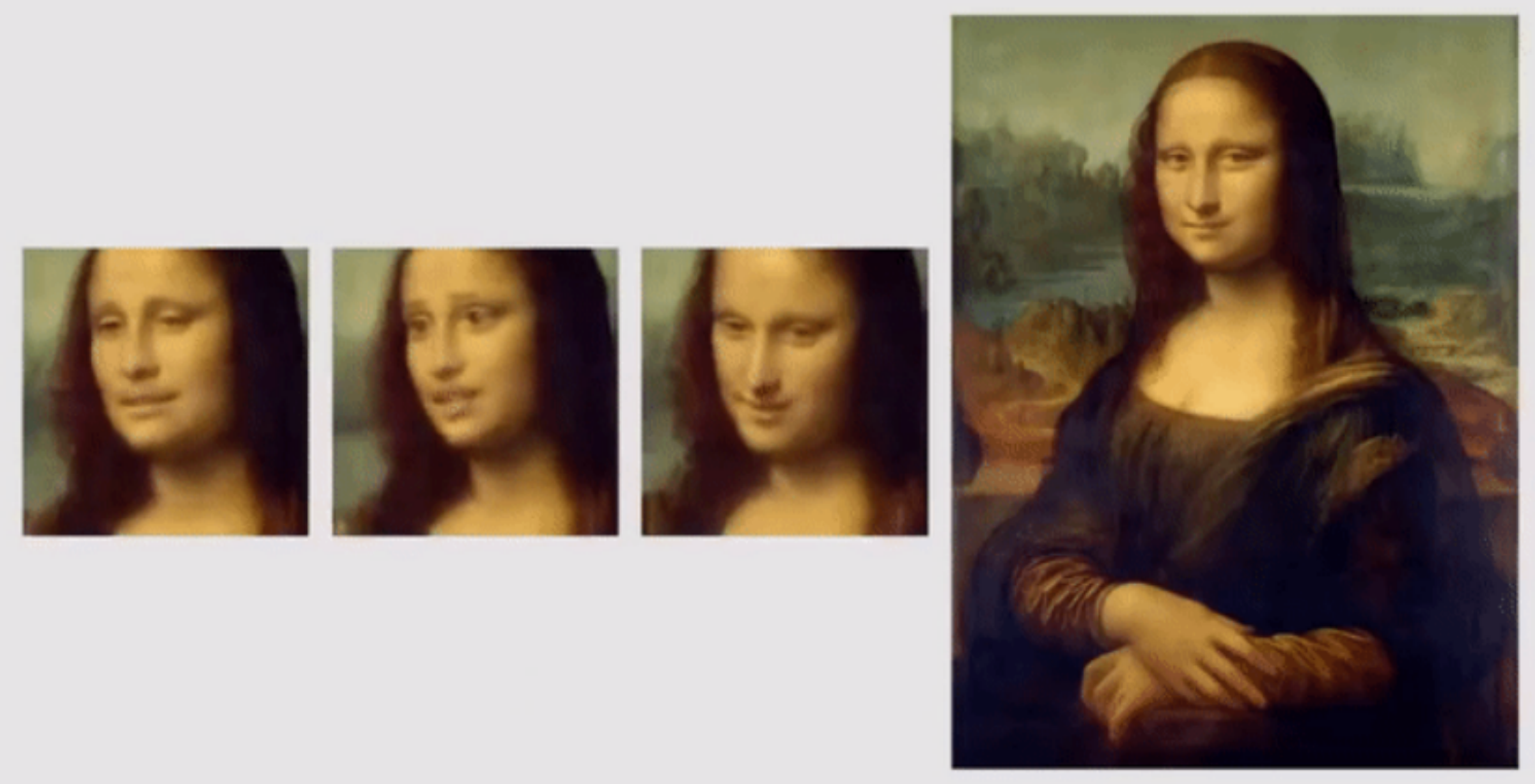


Figure 18: Living Portraits with GANs

Some more GAN results (cont.)

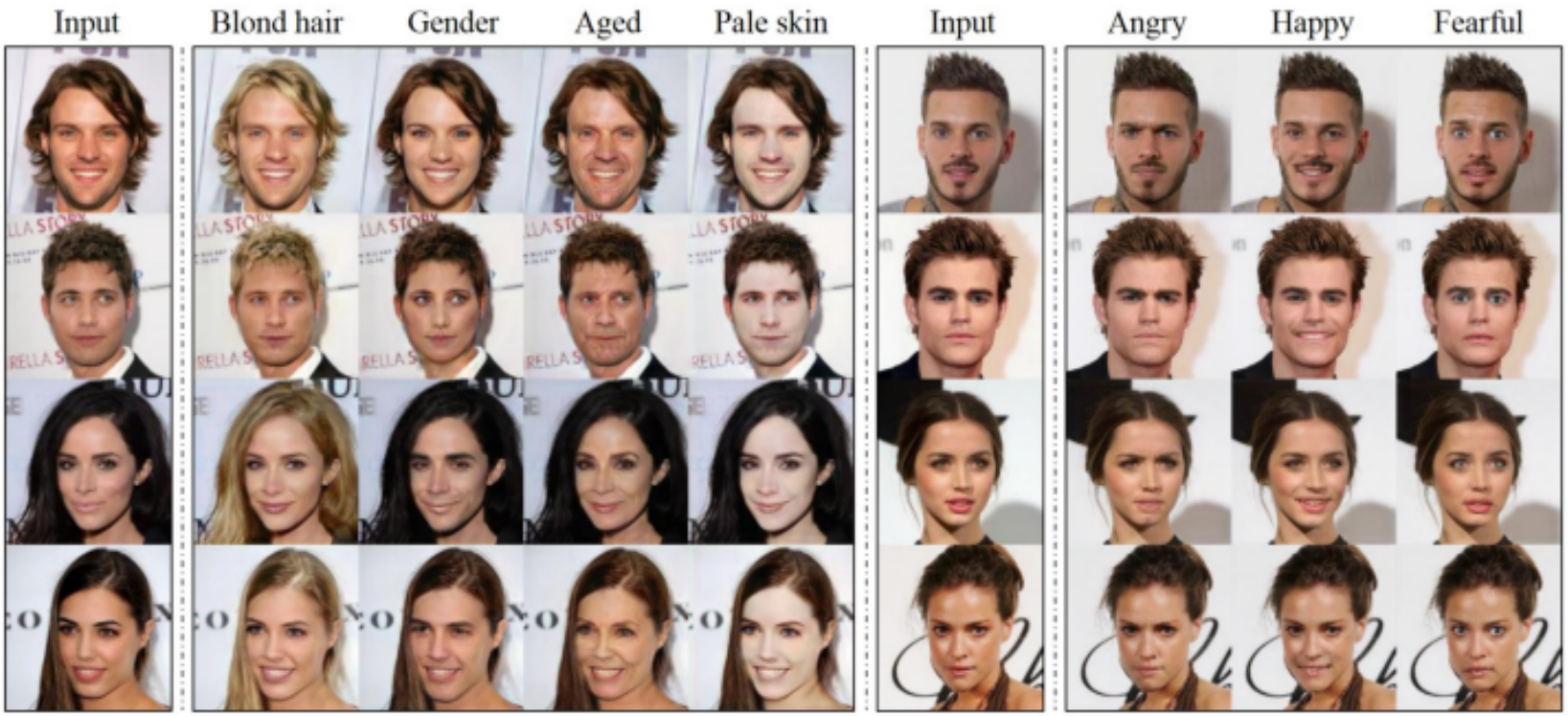


Figure 19: Image to Image translation in multiple domains with StyleGAN (Choi et al.)

Summary - GANs

- Instead of explicitly learning data probability distribution. Learn it implicitly.
- Train two networks jointly. Generator generates fake samples and aims to fool discriminator. Discriminator aims to discriminate between real and fake samples.
- GANs are notoriously difficult to train. Lots of tips and tricks available.
- Huge amount of research done on GANs and plenty of variations with interesting results.

References

Reference Slides

- Fei-Fei Li "Generative Deep Learning" CS231
- Francois Fleuret "Deep Learning" EE559
- Murtaza Taj "Deep Learning" CS437

References (Papers)

- Goodfellow et al., "Generative Adversarial Networks," 2014. arXiv:1406.2661
- Arjovsky et al., "Wasserstein GAN," 2017. arXiv:1701.07875
- Zhu et al., "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," 2017. arXiv:1703.10593
- Karras et al., "A Style-Based Generator Architecture for Generative Adversarial Networks," CVPR 2019. arXiv:1812.04948
- Creswell et al., "Generative Adversarial Networks: An Overview," IEEE Signal Processing Magazine, 2018. IEEE Xplore

GANs: Definition and Core Components:

- **Definition**: Generative Adversarial Networks (GANs) consist of two neural networks trained in opposition to each other.
- **Core Components:**
  - **Generator (G):** Takes random noise (latent vector) and produces fake data samples.
  - **Discriminator (D):** Receives real or fake data and predicts whether the input is real.
- **Mathematical Formulation:**

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

Where:

- x: Real data sample
- z: Noise vector sampled from prior p_z
- $G(z)$: Generated sample

Credits

Dr. Prashant Aparajeya

Computer Vision Scientist — Director(AISimply Ltd)

p.aparajeya@aisimply.uk

This project benefited from external collaboration, and we acknowledge their contribution with gratitude.