Generative Models II

Dr. Parlett-Pelleriti

Outline

- Generative Adversarial Networks Overview
- Training GANs
- Issues with Training GANs
- Conditional GANs

Generative Adversarial Networks

Generative **Adversarial** Networks

Generator

From: https://www.youtube.com/watch?v=JBIm4wnjNMY

Discriminator

From: https://www.youtube.com/watch?v=JBIm4wnjNMY

Adversarial Training

From: https://www.youtube.com/watch?v=JBlm4wnjNMY

Adversarial Training

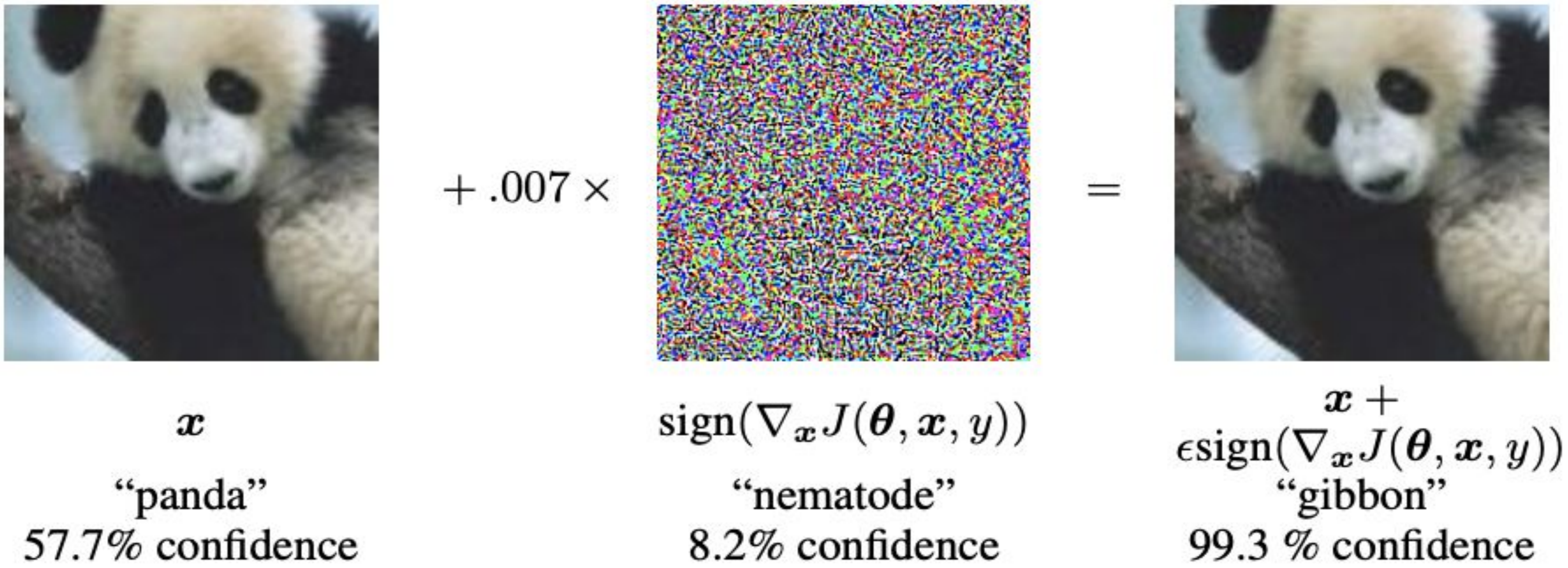
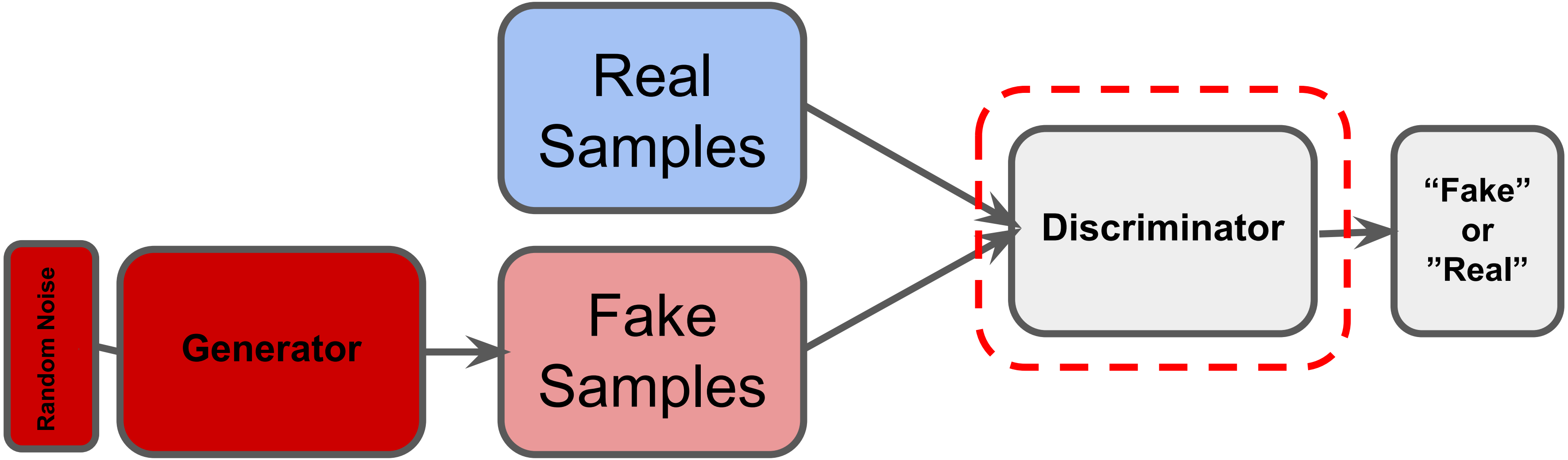


Image from: Explaining And Harnessing Adversarial Examples Goodfellow (2015)

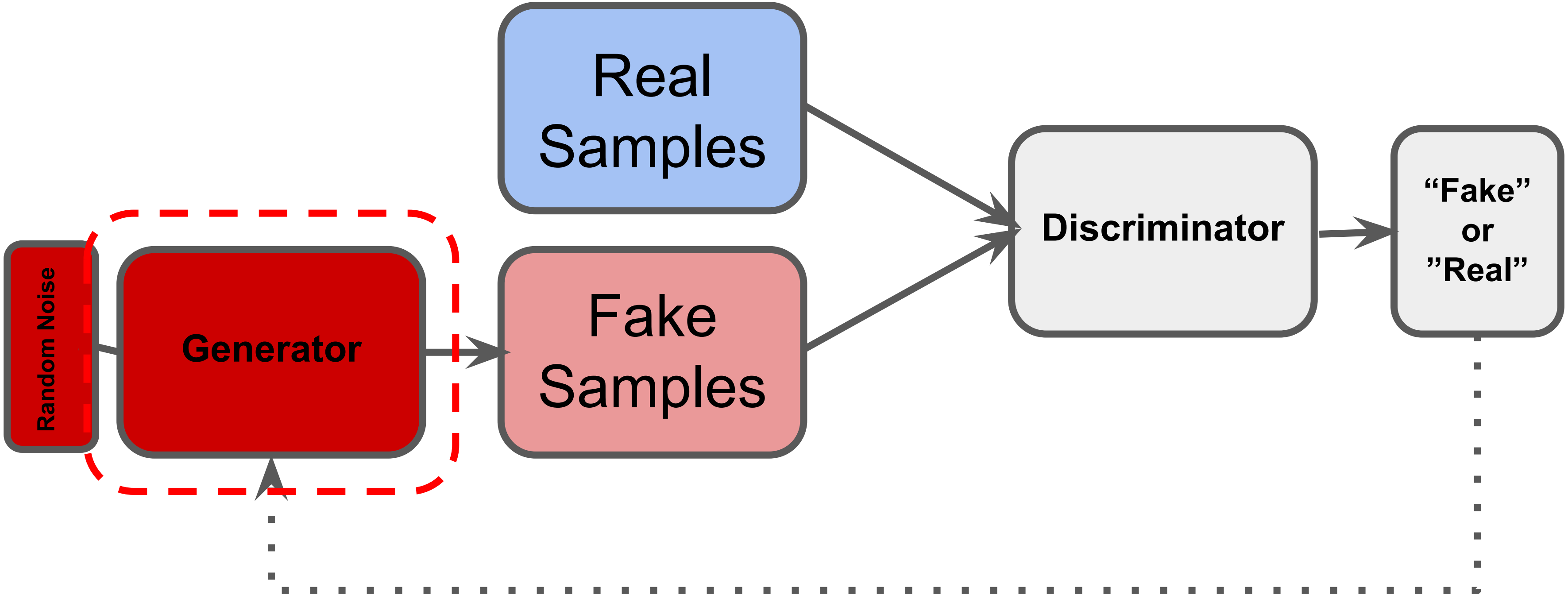
Adversarial Training

ZeroGPT

The Discriminator



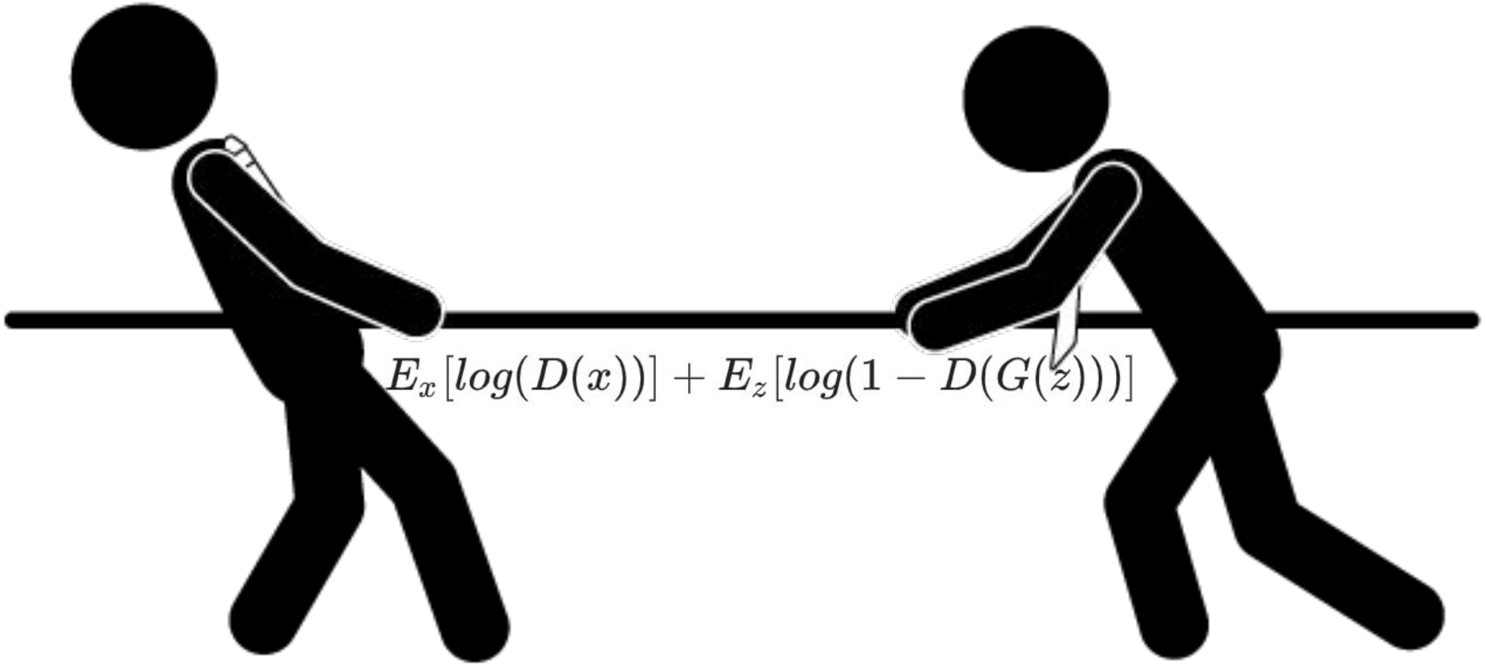
The Generator



Training GANs

1. Train the **discriminator** holding **generator** constant
2. Train the **generator** holding the **discriminator** constant
3. **Repeat** until convergence

GAN Loss Functions



Generator

Discriminator

GAN Loss Functions

$$E_x[log(\boxed{D(x)})] + E_z[log(1 - D(G(z)))]$$

Discriminator's
estimate that real
sample is real

GAN Loss Functions

$$E_x[log(D(x))] + E_z[log(1 - \boxed{D(G(z))})]$$

Discriminator's
estimate that fake
sample is real

GAN Loss Functions

Discriminator wants to maximize

$$E_x[log(\boxed{D(x)})] + E_z[log(1 - \boxed{D(G(z))})]$$

Discriminator's estimate that real sample is real

Discriminator's estimate that fake sample is real

GAN Loss Functions

Generator wants to minimize

$$E_x[log(\boxed{D(x)})] + E_z[log(1 - \boxed{D(G(z))})]$$

Discriminator’s estimate that real sample is real

Discriminator’s estimate that fake sample is real

Training Loop

```
for i in range(training_iteration):
    # hold generator constant
    for k in range(discrim_steps):
        x   = sample_real_inputs(num = m)
        gz1 = generate_fake_inputs(num = m)
        discrim_params = gradient_ascent(x, gz)
    # hold discriminator constant
    gz2 = generate_fake_inputs(num = m)
    gener_params = gradient_descent(gz2)
```

Training Loop

```
for i in range(training_iteration):
    # hold generator constant
    for k in range(discrim_steps):
        x   = sample_real_inputs(num = m)
        gz1 = generate_fake_inputs(num = m)
        discrim_params = gradient_ascent(x, gz)
    # hold discriminator constant
    gz2 = generate_fake_inputs(num = m)
    gener_params = gradient_descent(gz2)
```

Training Loop

```
for i in range(training_iteration):
    # hold generator constant
    for k in range(discrim_steps):
        x   = sample_real_inputs(num = m)
        gz1 = generate_fake_inputs(num = m)
        discrim_params = gradient_ascent(x, gz)
    # hold discriminator constant
    gz2 = generate_fake_inputs(num = m)
    gener_params = gradient_descent(gz2)
```

Training GANs

1. Train the **discriminator** holding **gene**
2. Train the **generator** holding the
3. **Repeat** until convergence

What if the Discriminator is too good?

In the minimax game, the discriminator <u>minimizes</u> a cross-entropy, but the generator <u>maximizes</u> the same cross-entropy. This is unfortunate for the generator, because when the discriminator successfully rejects generator samples with high confidence, the ***generator's gradient vanishes.***

-Ian Goodfellow (https://arxiv.org/pdf/1701.00160.pdf)

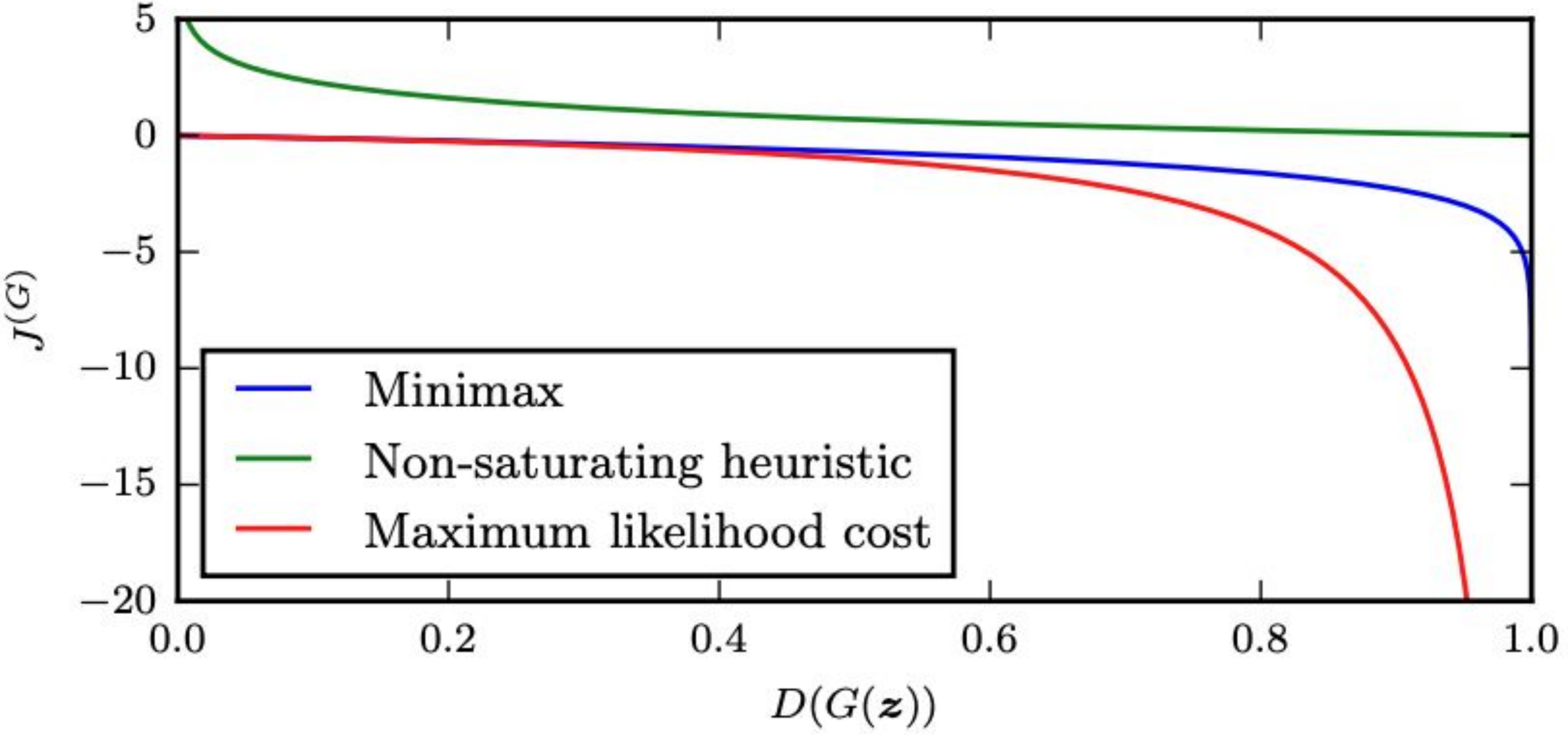


Image from: Ian Goodfellow (https://arxiv.org/pdf/1701.00160.pdf)

Non-Saturating GAN Loss

Generator wants to maximize

$$-log(D(G(z)))$$

instead of minimizing

$$log(1 - D(G(z)))$$

Mode Collapse

Image from: https://neptune.ai/blog/gan-loss-functions

Mode Collapse

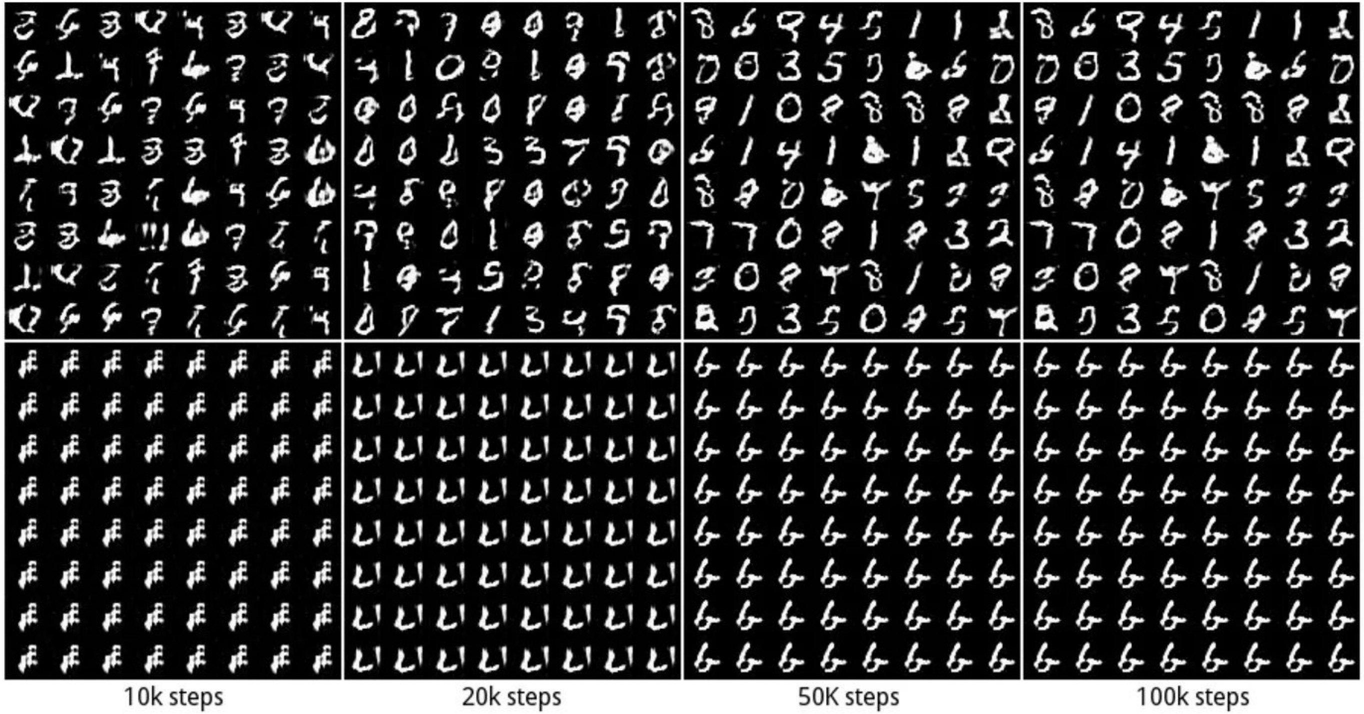
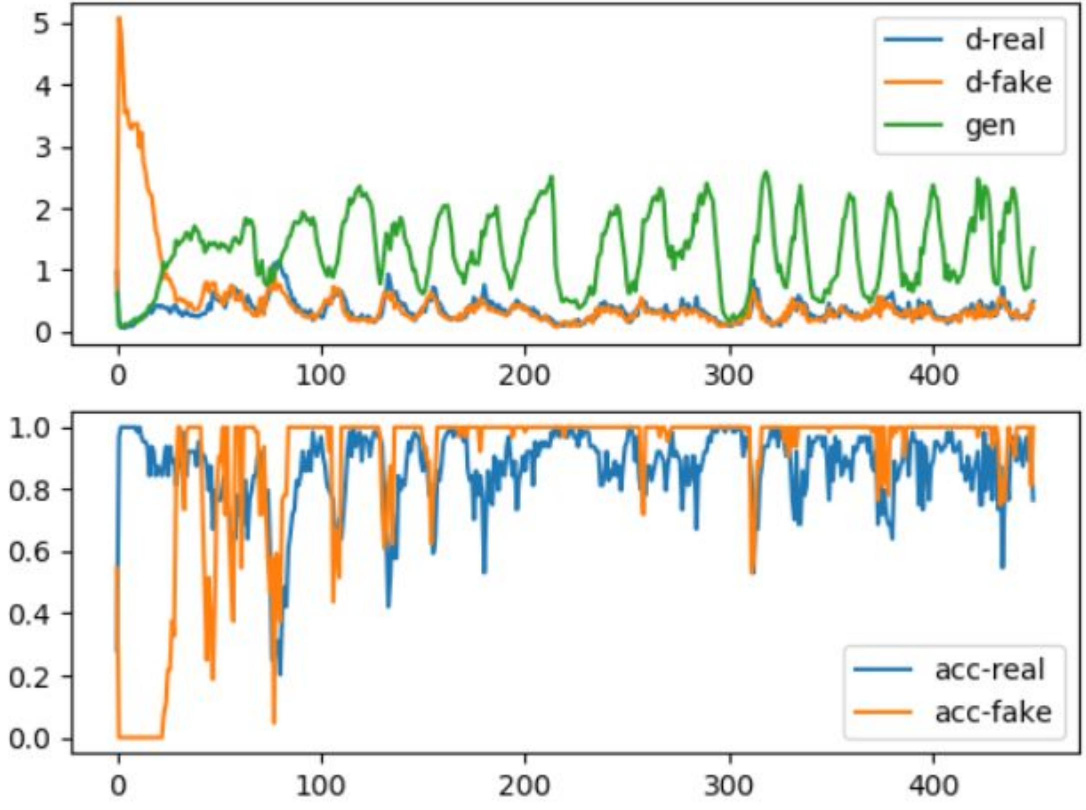


Image from: https://arxiv.org/pdf/1611.02163.pdf

Convergence



Sometimes the two players eventually reach an equilibrium, but in other scenarios they repeatedly undo each others' progress without arriving anywhere useful.

-Ian Goodfellow
(https://arxiv.org/pdf/1701.00160.pdf)

Image from: https://neptune.ai/blog/gan-loss-functions

Wasserstein GANs

Discriminator Maximizes:

$$D(x) - D(G(z))$$

Generator Maximizes:

$$D(G(z))$$

Wasserstein GANs

"The Critic" Maximizes:

$$D(x) - D(G(z))$$

Generator Maximizes:

$$D(G(z))$$

Conditional GANs

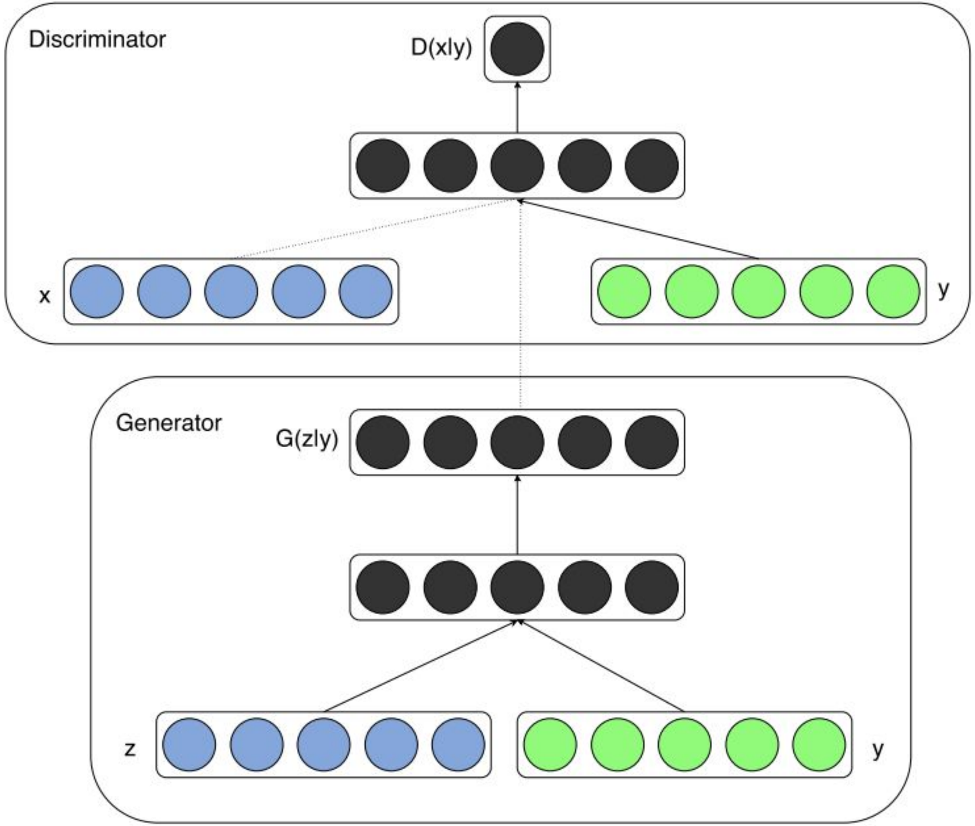


Figure 1: Conditional adversarial net

Image from: (Mirza and Osindero, 2014)

Conditional GAN