CSCI 1470

Eric Ewing

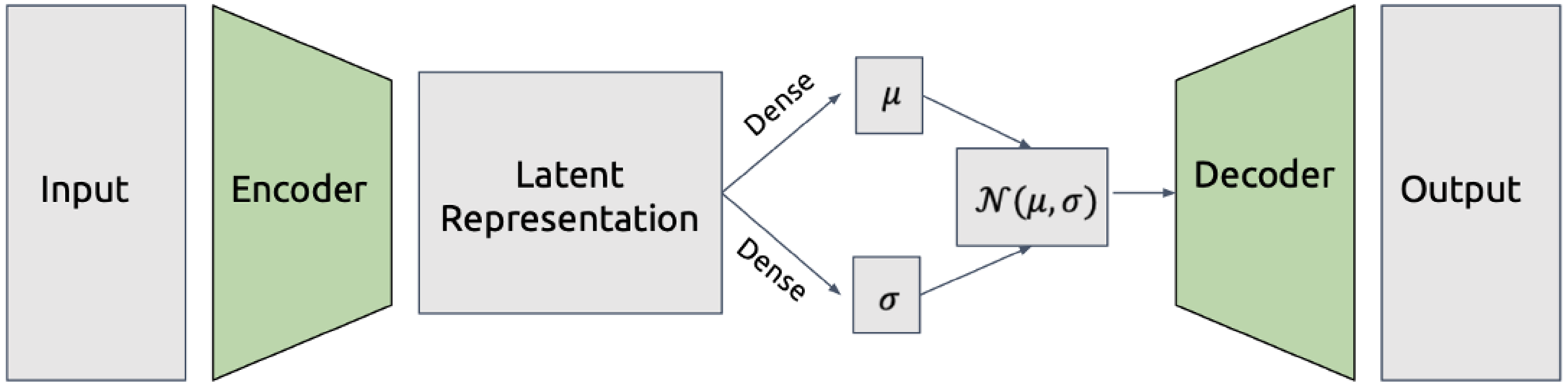
Friday,
4/4/25

Deep Learning

Day 27: GANs

Background generated by Lightweight GAN: https://arxiv.org/pdf/2111.11578

VAEs Review

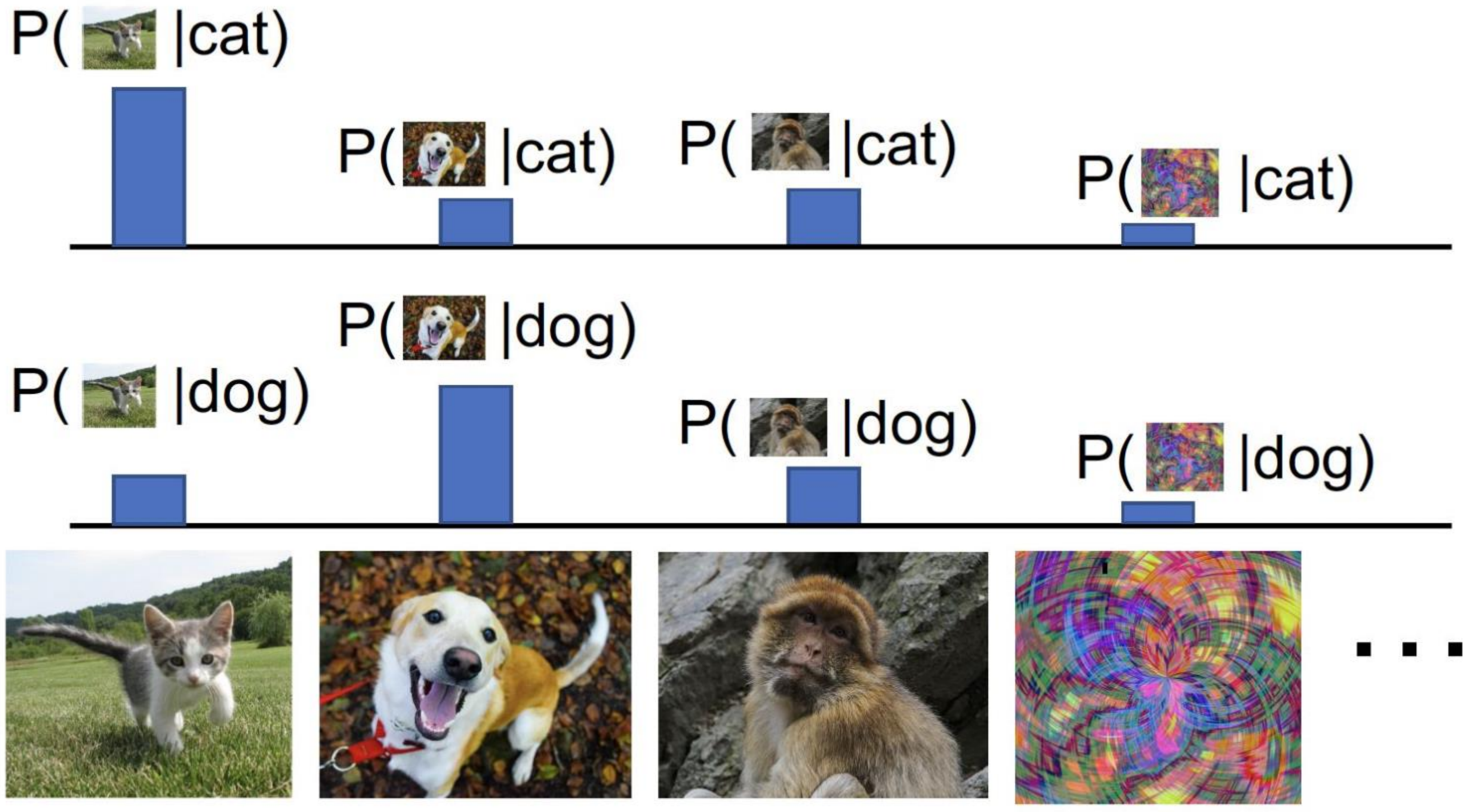

Input
Encoder
Latent Representation
Dense
Dense
μ
σ
$\mathcal{N}(\mu, \sigma)$
Decoder
Output

Discriminative vs Generative Models

Discriminative Model: Learn a probability distribution p(y|x)

Generative Model: Learn a probability distribution p(x)

Conditional Generative Model: Learn p(x|y)

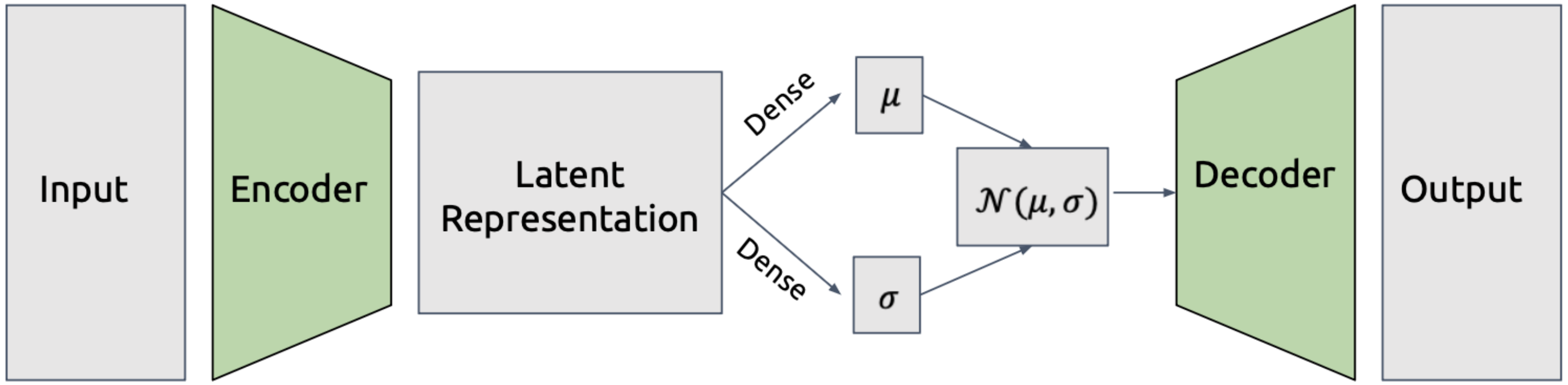


Conditional Generative Model: Each possible label induces a competition among all images

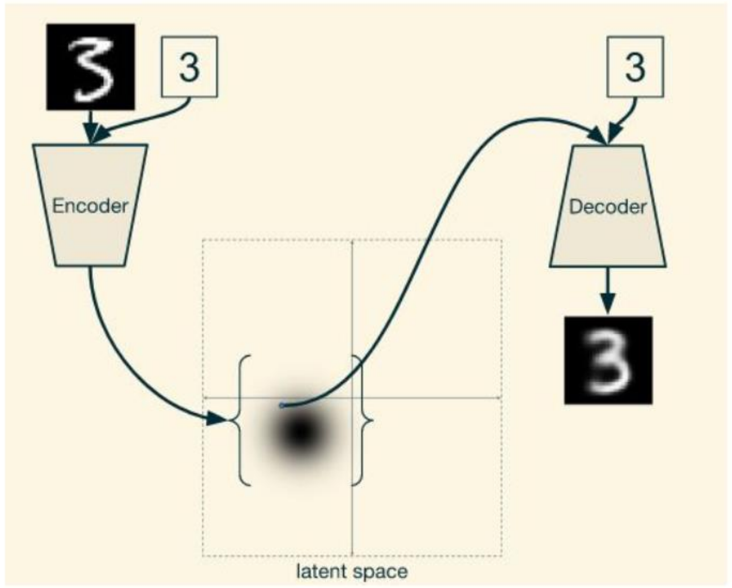
Credit: UMich EECS498

Conditional VAE

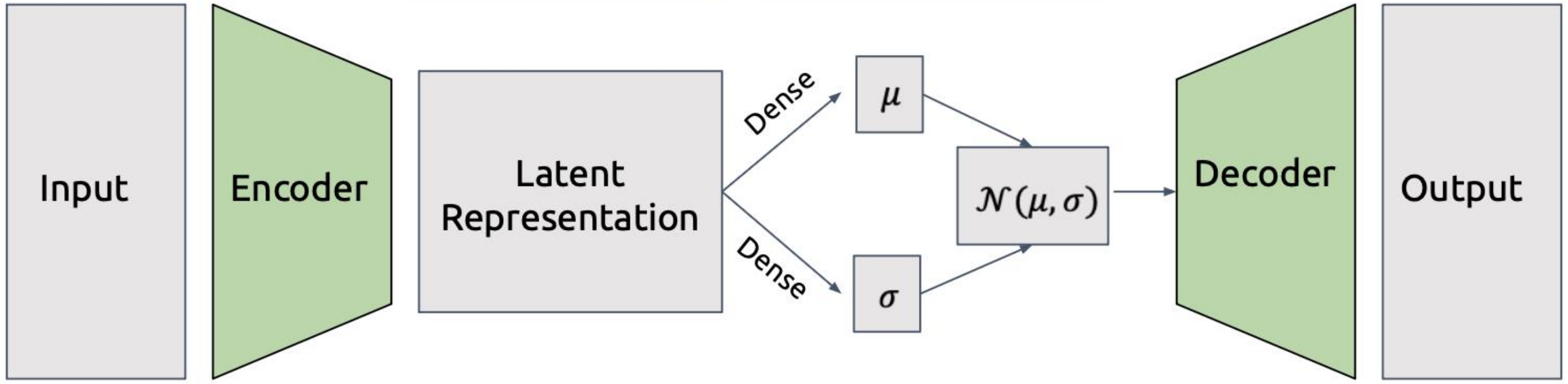
Any ideas?



Conditional VAE



Encoder generates $P(z|c)$
Generator generates image $P(\hat{x}|z, c)$



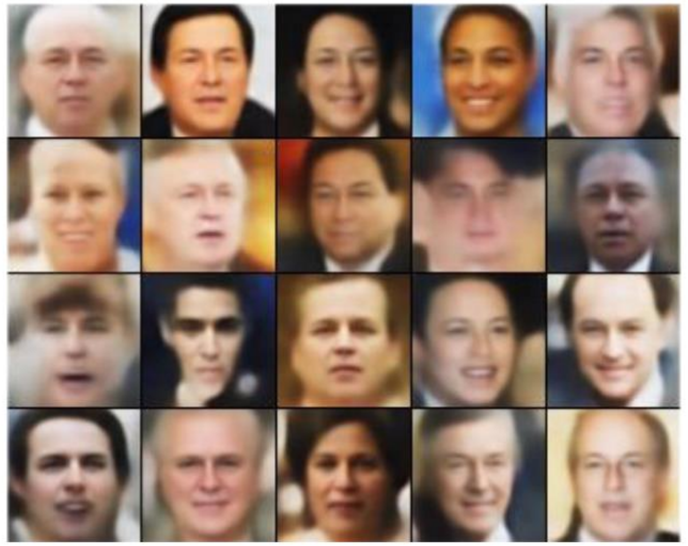
Why are VAE samples blurry?

- Our reconstruction loss is the culprit
- Mean Square Error (MSE) loss looks at each pixel in isolation
- If no pixel is too far from its target value, the loss won't be too bad
- Individual pixels look OK, but larger-scale features in the image aren't recognizable

- ***Solutions?***
  - Let's choose a different reconstruction loss!

Input

VAE reconstruction



https://towardsdatascience.com/what-the-heck-are-vae-gans-17b86023588a

Generative Adversarial Networks (GANs)

Does nefarious things

They are adversaries

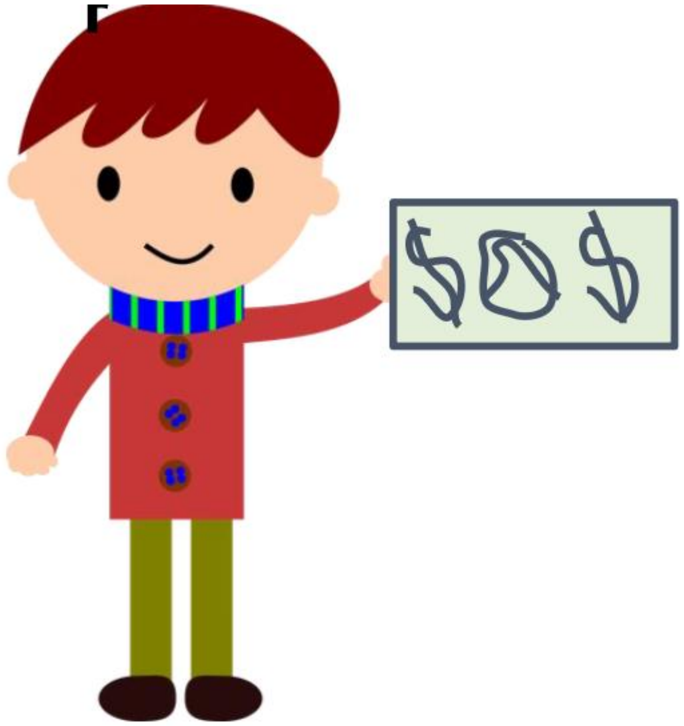
Investigates crimes

Moriarty

But how did they get so good at what they do?

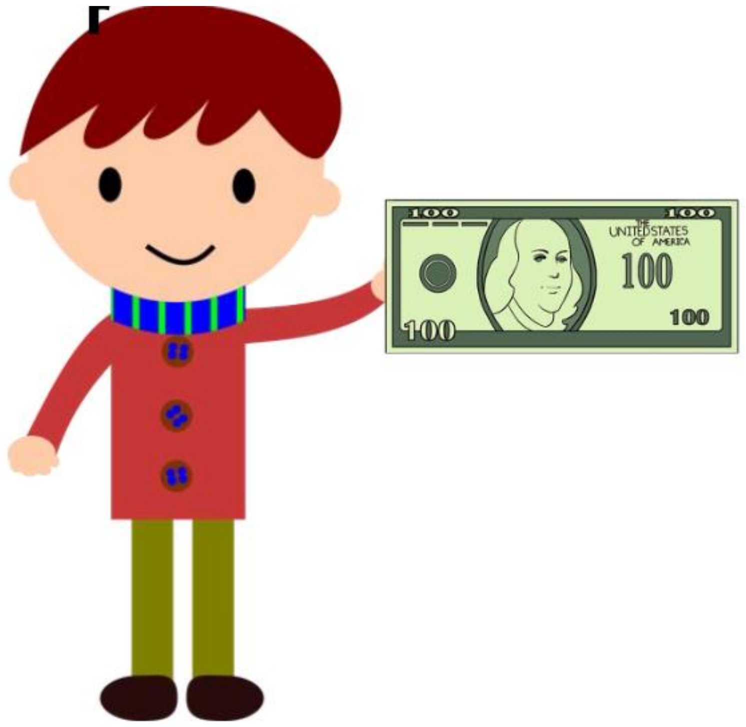
Sherlock

Moriarty started out as a child, trying to forge bills (he wasn't very good at it)



Young Sherlock would look at the bills and try to figure out if they were forged or not (Sherlock was also not very good at it)

Moriarty used the feedback to get even better at forging

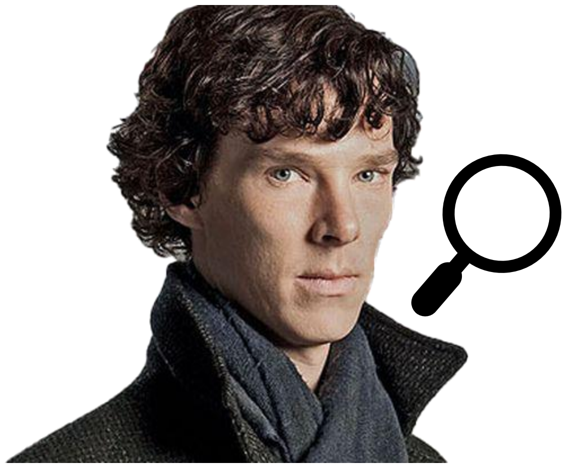


Sherlock also improved as Moriarty started producing better bills

A little bit of competition can help you grow

To try and stay ahead, Moriarty further improves

And Sherlock gets even better to match the competition



Generative Adversarial Networks

- GANs use these ideas to train a pair of networks together
- These networks are called the Generator and Discriminator

Neural network to generate data

Neural network to ”guess” if that data is real

Generator
Discriminator

GANs: Training the Discriminator

Discriminator wants to maximize:

Expectation over fake images (z is vector in latent space)

$$\underbrace{E_x}_{\text{Expectation over real images x}}\left[\log\left(\underbrace{D(x)}_{\text{The log probability predicted by Discriminator}}\right)\right] + \overset{\text{Expectation over fake images (z is vector in latent space)}}{E_z}\left[1 - \log\left(\underbrace{G(z)}_{\text{1-log probability of fake image made by generator}}\right)\right]$$

The log probability predicted by Discriminator

Expectation over real images x

1-log probability of fake image made by generator

What loss function does this remind you of?

This is BCE, if real data are “class 1” and fake data are “class 0”

But…

The discriminator wants to **MAXIMIZE:**

$$E_x\left[\log\left(D(x)\right)\right] + E_z[1 - \log\left(G(z)\right)]$$

That’s not actually a problem, we can minimize the negative…

What is a problem, is that the generator wants to minimize the original function

GANs as a Game

Overall Problem:

$$\min_{G} \quad \max_{D} \quad E_x\left[\log\left(D(x)\right)\right] + E_z[1 - \log\left(G(z)\right)]$$

Training GANs can be modeled as a 2-player zero-sum adversarial game

In an adversarial game, players have opposing goals (not everyone can win)

Zero-sum means that their goals are exactly opposite

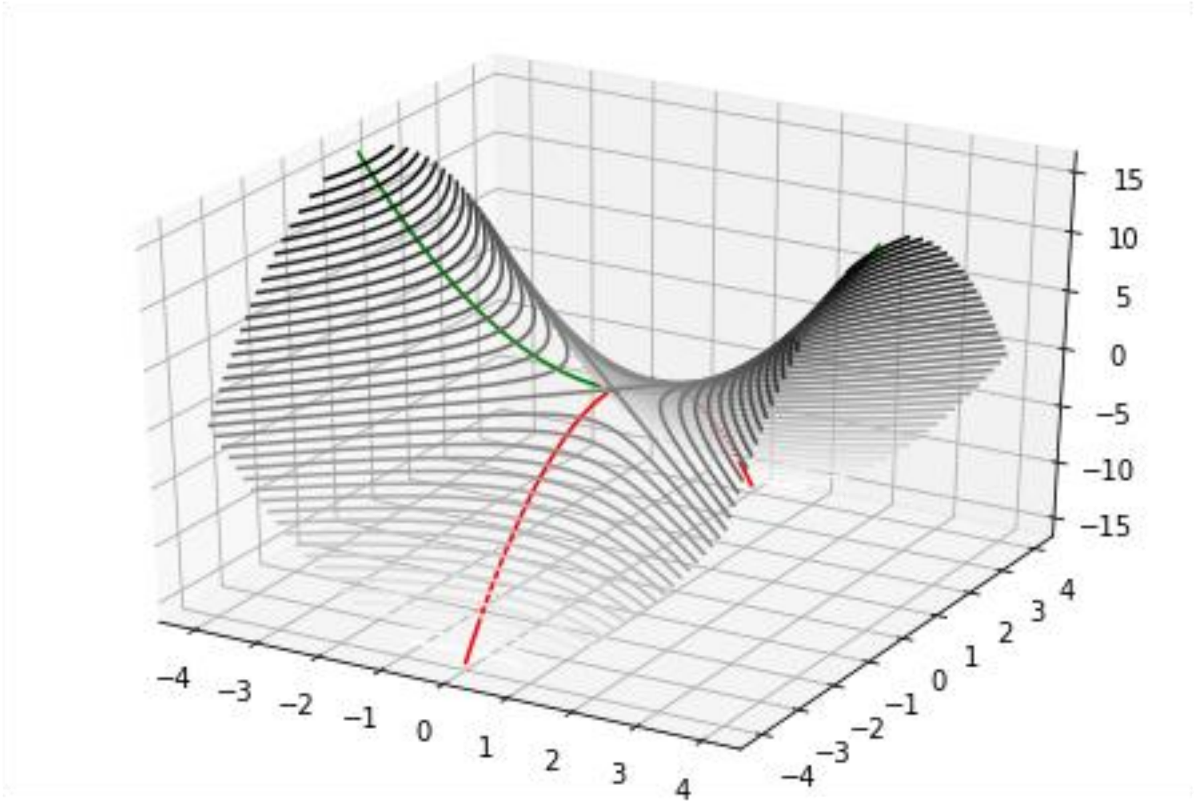
Nash Equilibrium

- A Nash Equilibrium is a solution concept for zero sum games
- Player A looks for a strategy such that no matter what player B chooses to do, player A will do no worse
- In the case of GANs, strategies are network parameter settings
- In a Nash Equilibrium, the generator will not be able to find different parameter settings such that it can do any better (and vice versa)

How do we find Nash Equilibria

- Game theory provides lots of tools, but the easiest to implement is Gradient Ascent Descent

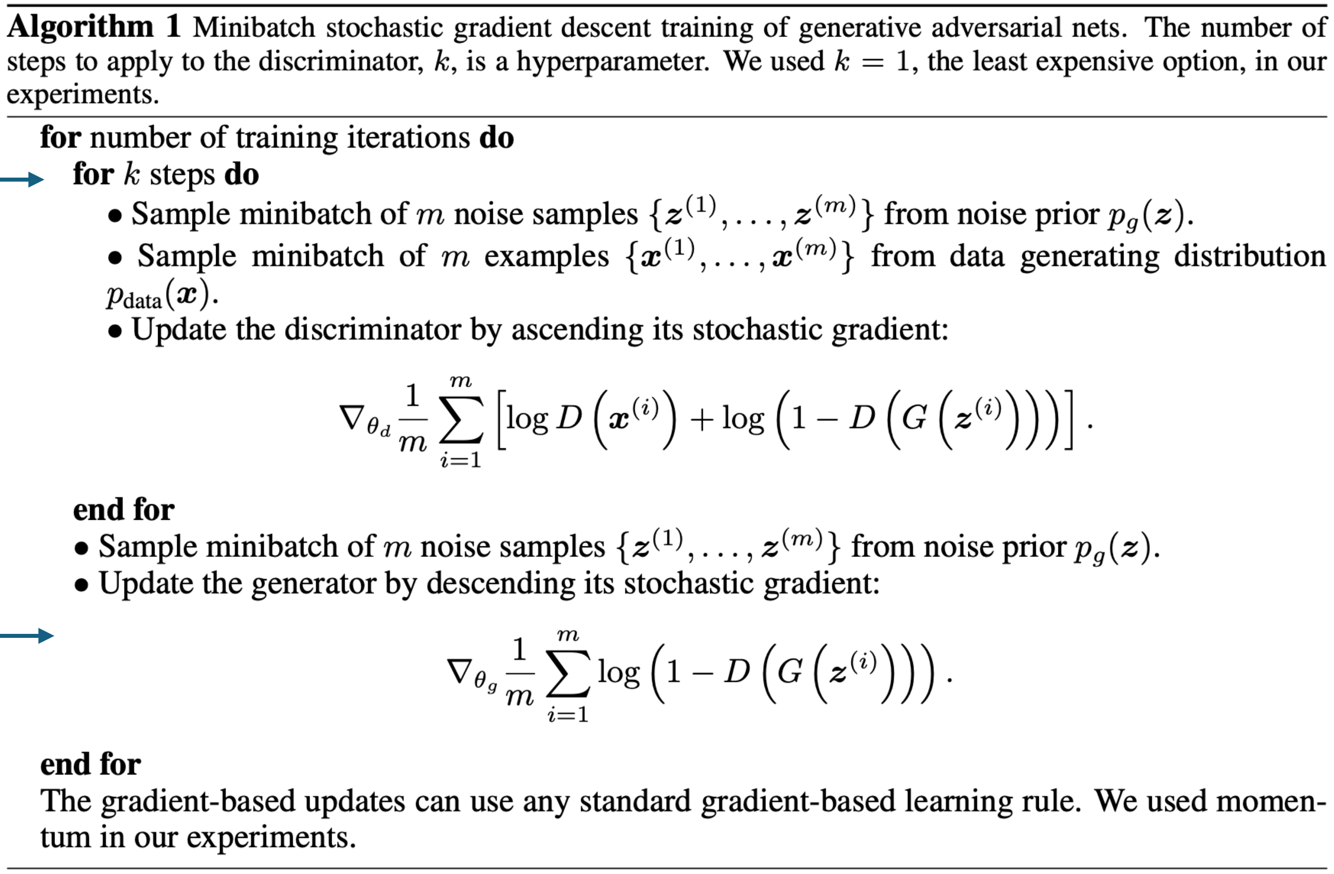
Alternate steps of gradient ascent (using gradient of maximizer) with steps of gradient descent (using gradient of minimizer)



GAN Training Algorithm

Inner loop to train D $\longrightarrow$

Outer loop to train G $\longrightarrow$



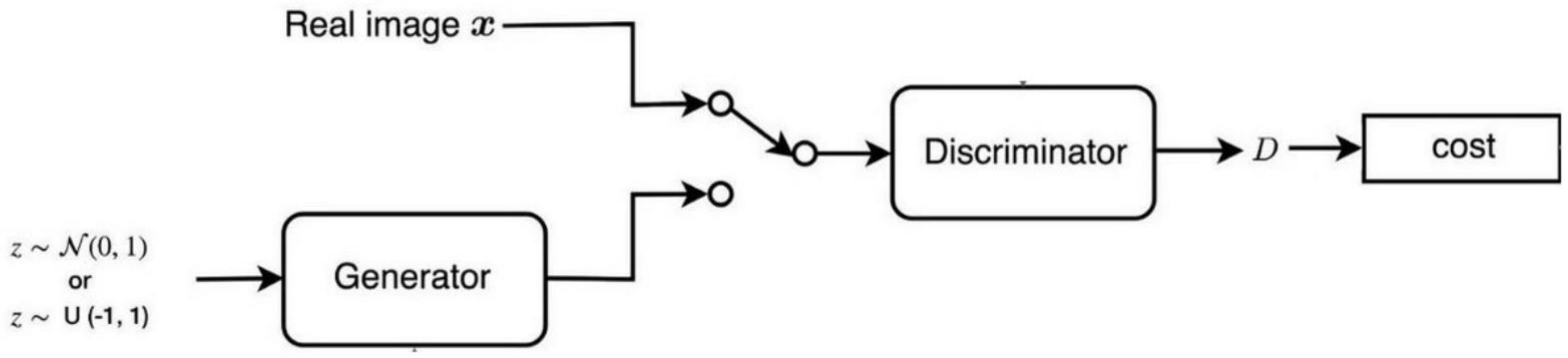
Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

```
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:
            $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$
    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 descending its stochastic gradient:
        $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$
end for
```

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

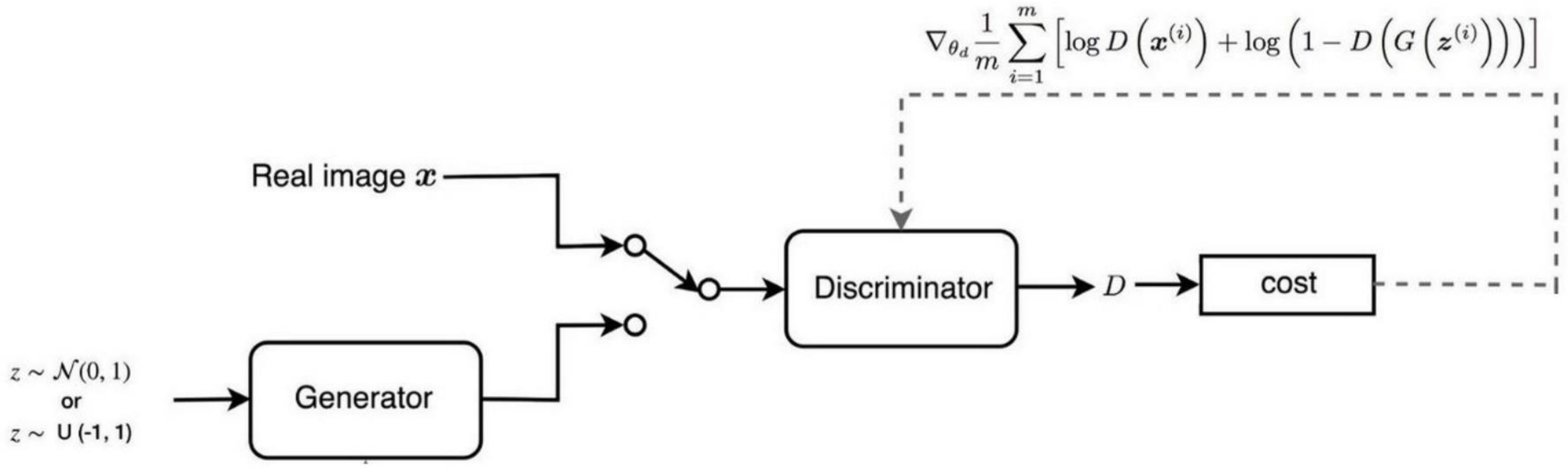
Goodfellow et al. “Generative Adversarial Nets”, 2014

GAN Loss



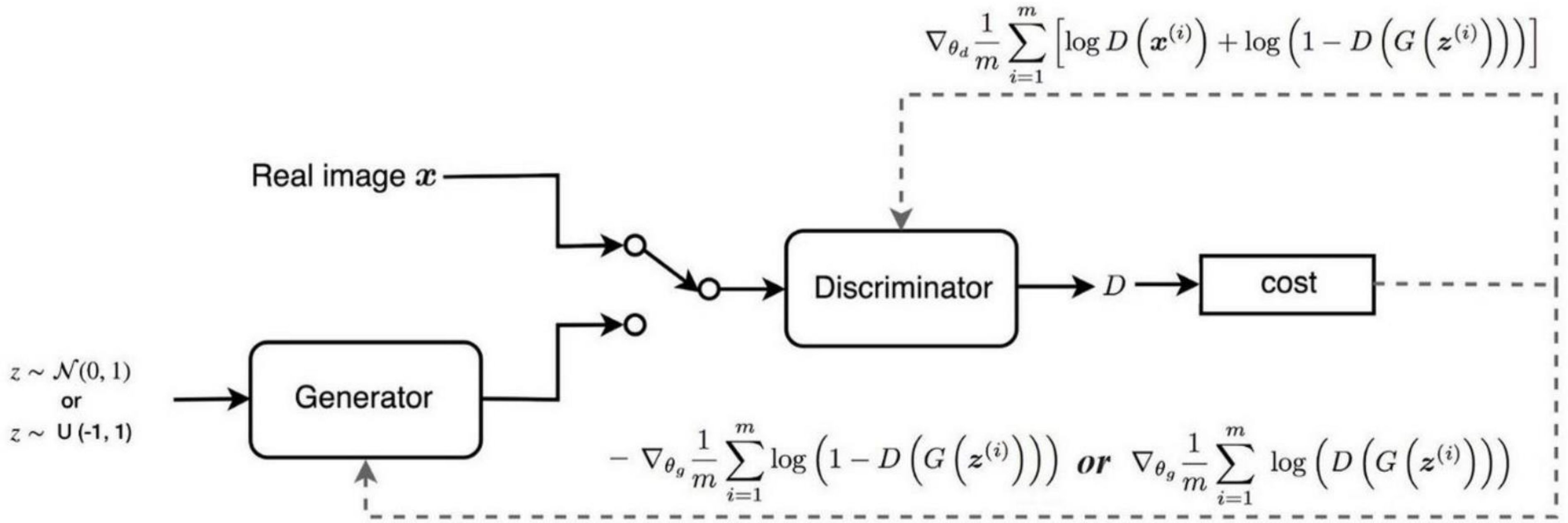
https://jonathan-hui.medium.com/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490

GAN Loss


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

Any questions?

GAN Loss



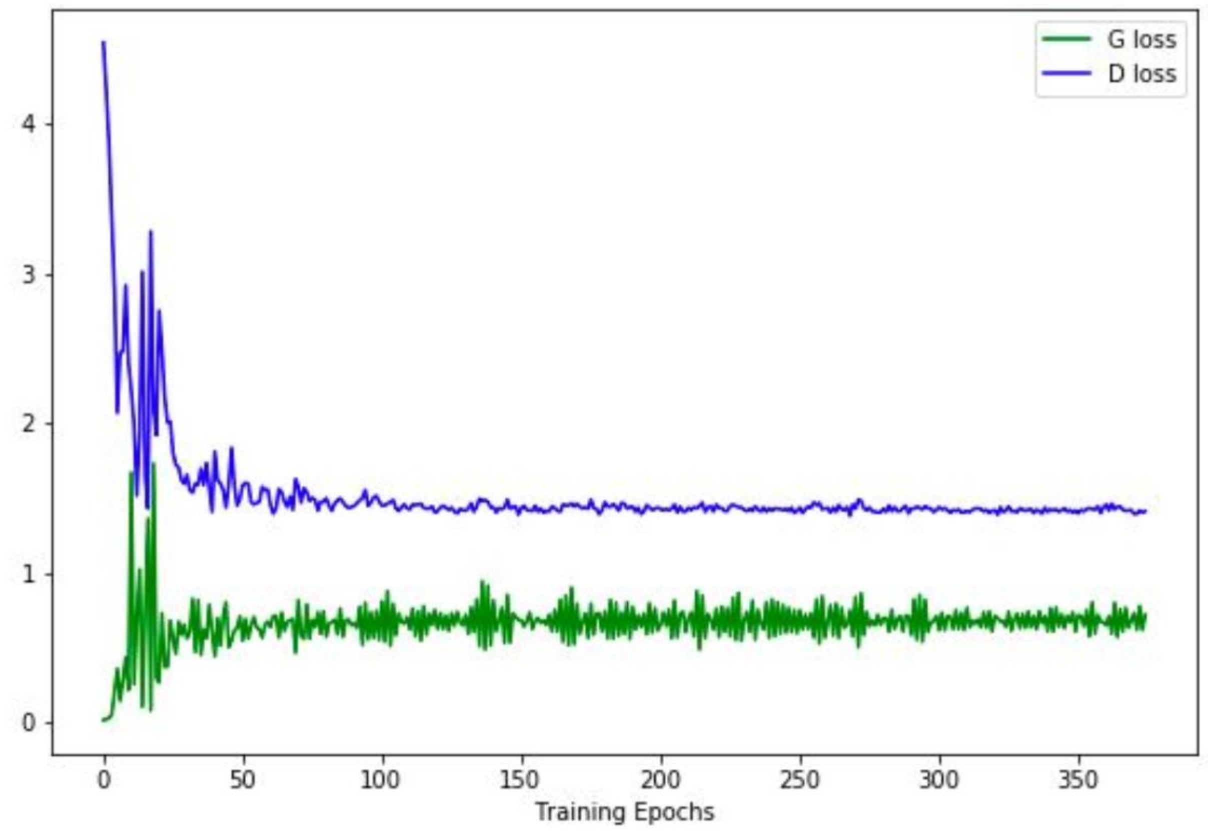
$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right]$
Real image $\boldsymbol{x}$
Discriminator
D
cost
$z \sim \mathcal{N}(0, 1)$
or
$z \sim$ U (-1, 1)
Generator
$-\nabla_{\theta_g} \frac{1}{m}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

https://jonathan-hui.medium.com/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490

Training Visualization

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

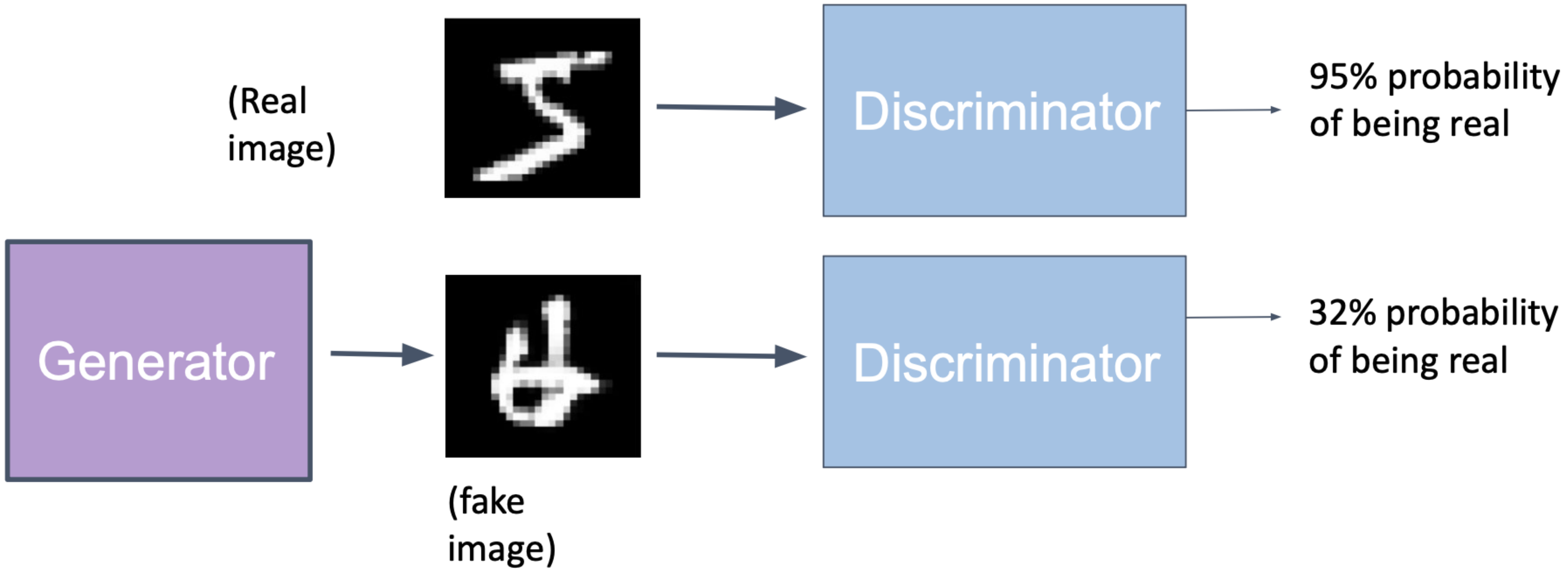
GAN Training Dynamics



- Does not exhibit the typical "training loss continues to go down" behavior
- ***Why?***
  - Training a GAN is a "stalemate" – G and D continually adjust to each other's improvements
  - More formally, training a GAN to convergence is attempting to find an equilibrium of a two-player minimax game

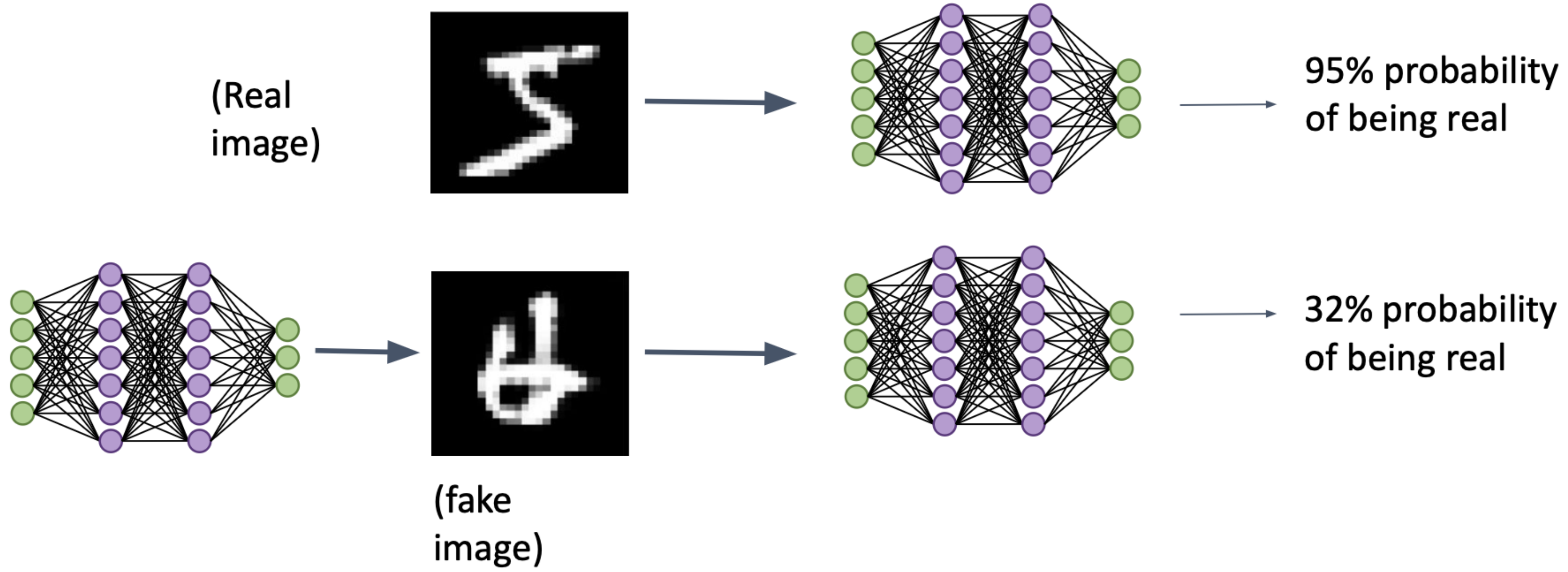
What do G and D look like inside?

- Architecture of the networks determined by problem



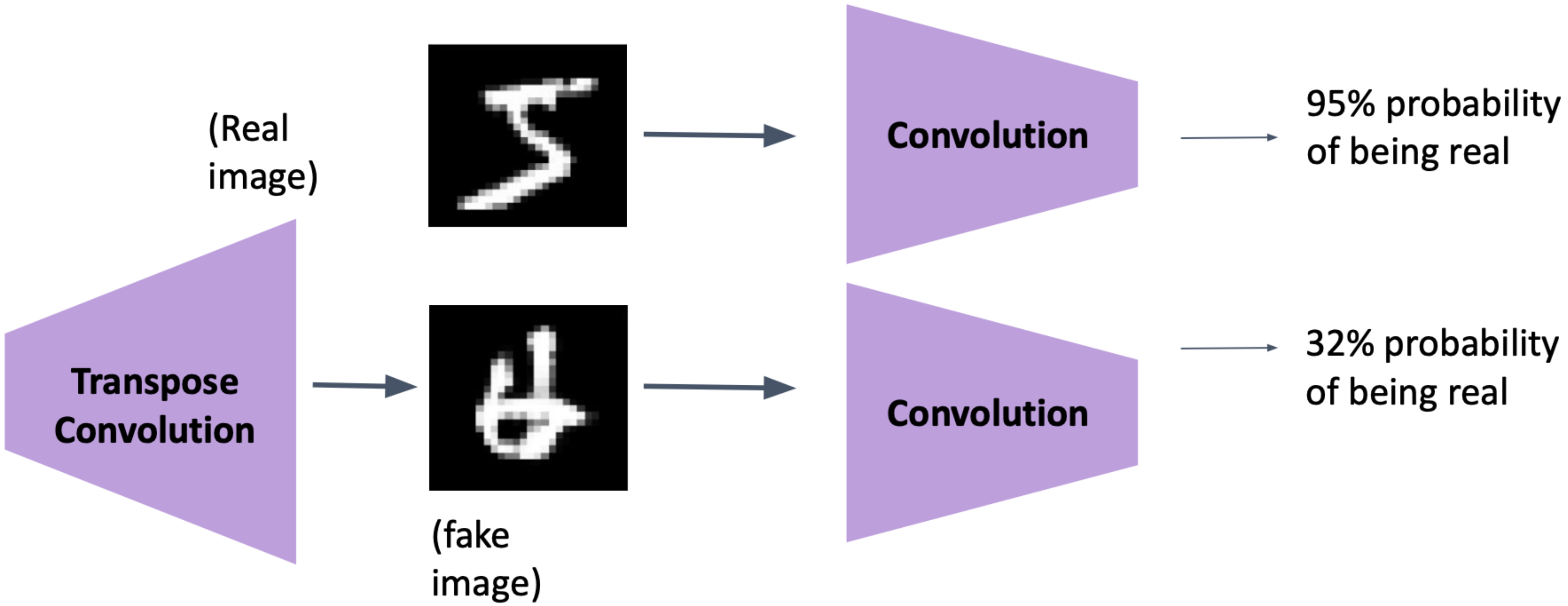
What do G and D look like inside?

- Architecture of the networks determined by problem
- **Fully connected**



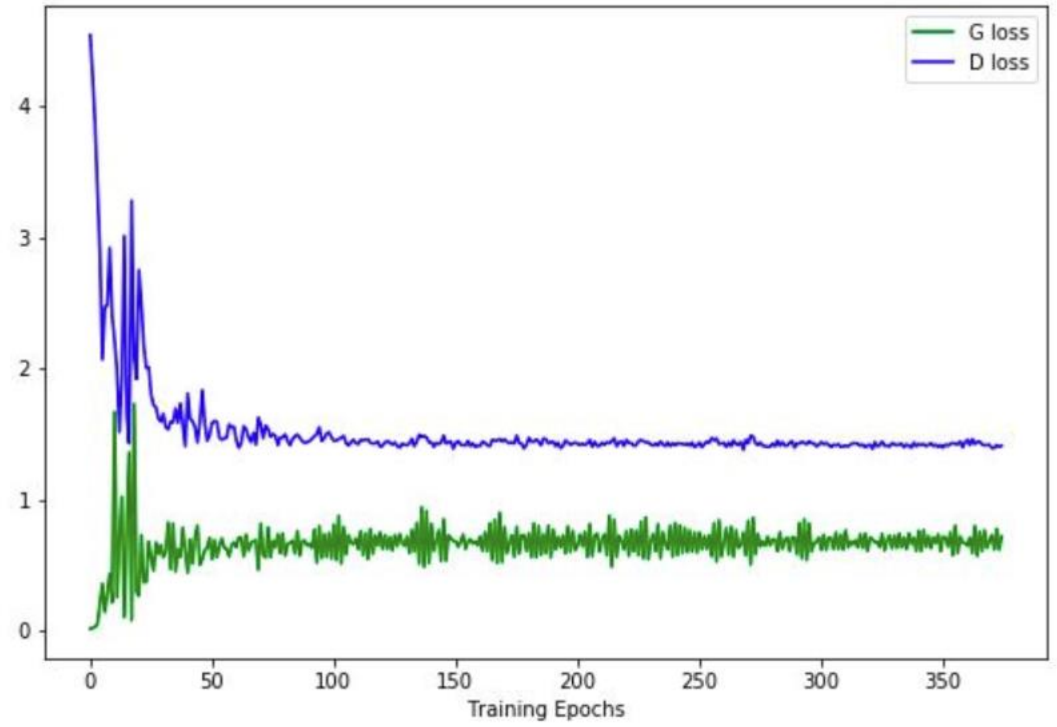
What do G and D look like inside?

- Architecture of the networks determined by problem
- **Convolutional / Transpose convolutional**



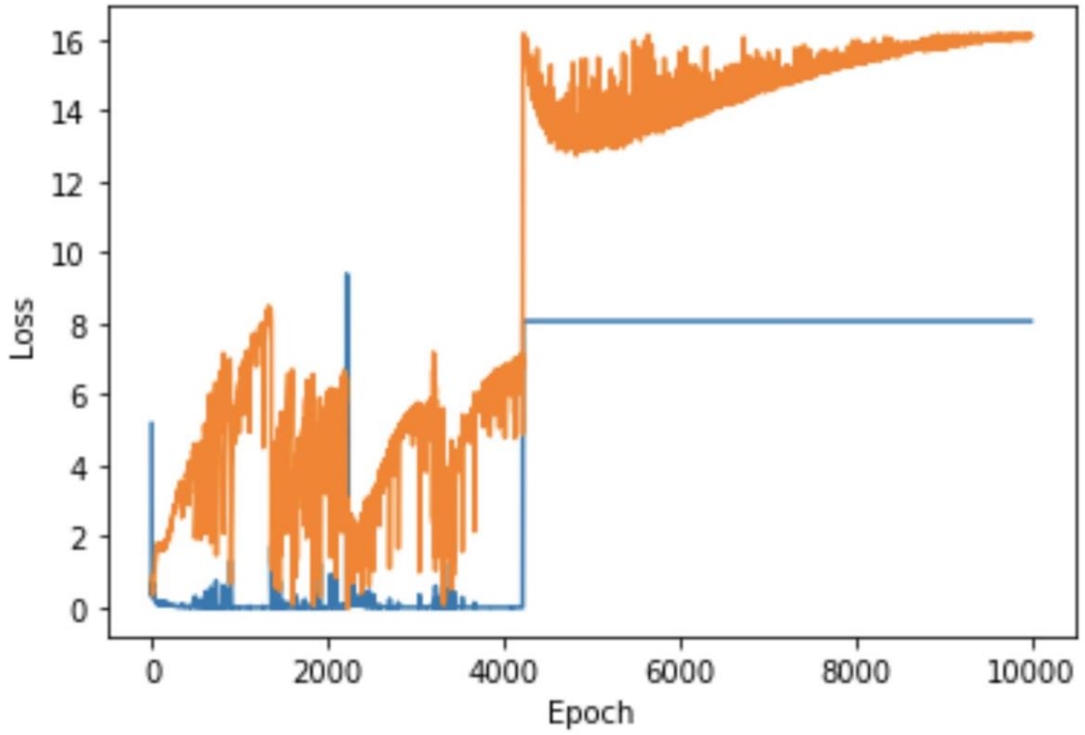
Problems with GANs

Training GANs is ***very*** unstable



- This is what it looks like when it's working…
- Turns out Equilibria are hard to find
  - With other networks, if the loss is going down, we know the network is doing better
  - Here, we have a moving target (G and D keep changing)
- These curves can oscillate a lot

Training GANs is ***very*** unstable



What happens if the discriminator ever becomes perfect at detecting the generator's fakes?

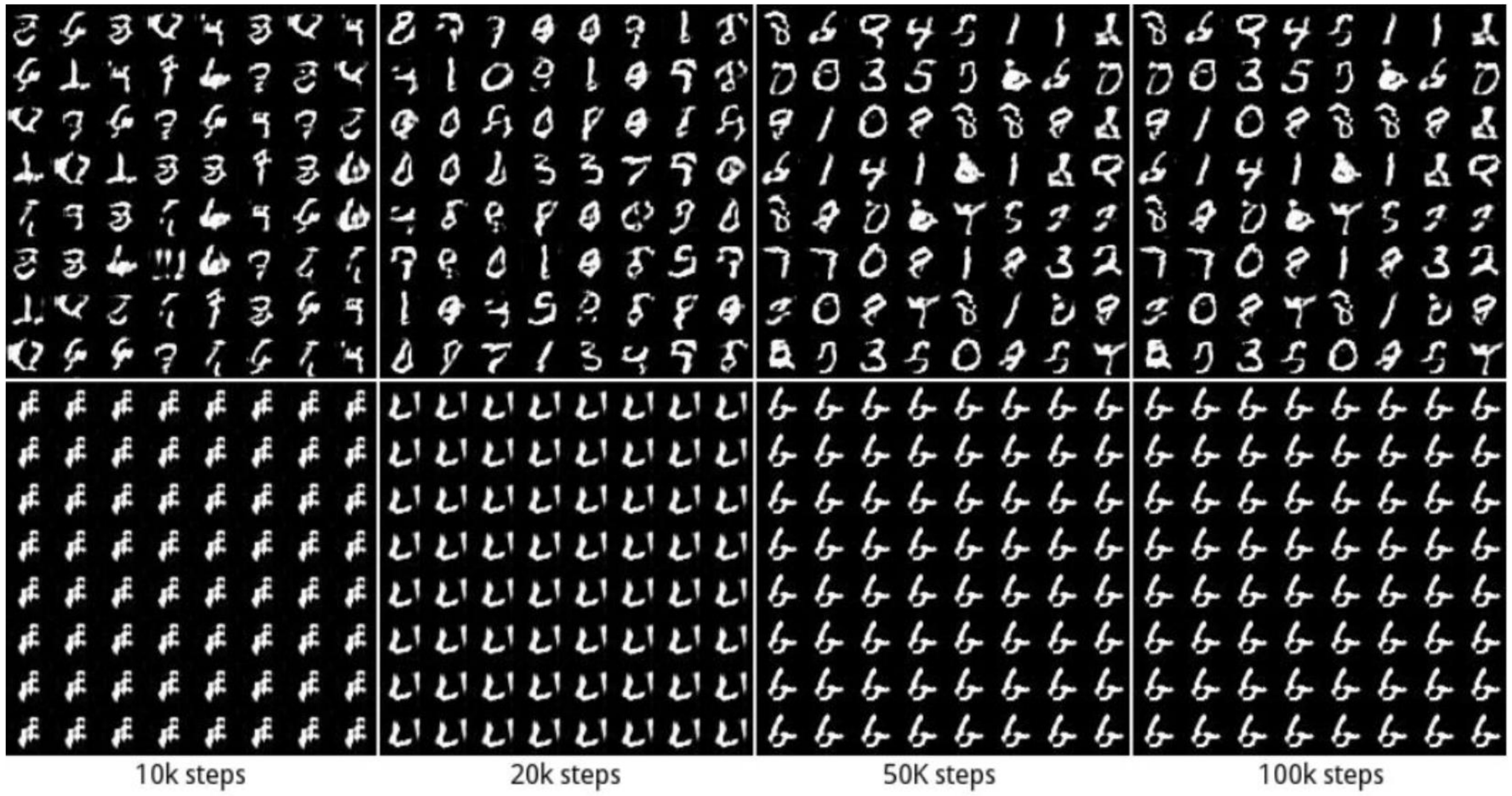
- Discriminator always returns p=0
- Since D always returns a constant, the gradient is constant
- The generator stops training

Mode Collapse

- Generator loss says: “generate an output that looks real”
- It does not say: “generate ***every*** output that looks real”
- The generator can “cheat” by finding one output / a few outputs that reliably fool the discriminator (the specific one(s) it finds can shift over training)

How do we fix this?

Mode Collapse



Output from a healthy GAN

Output from a GAN with mode collapse. All outputs from GAN, regardless of random input noise, are the same.

Balancing the Discriminator is hard

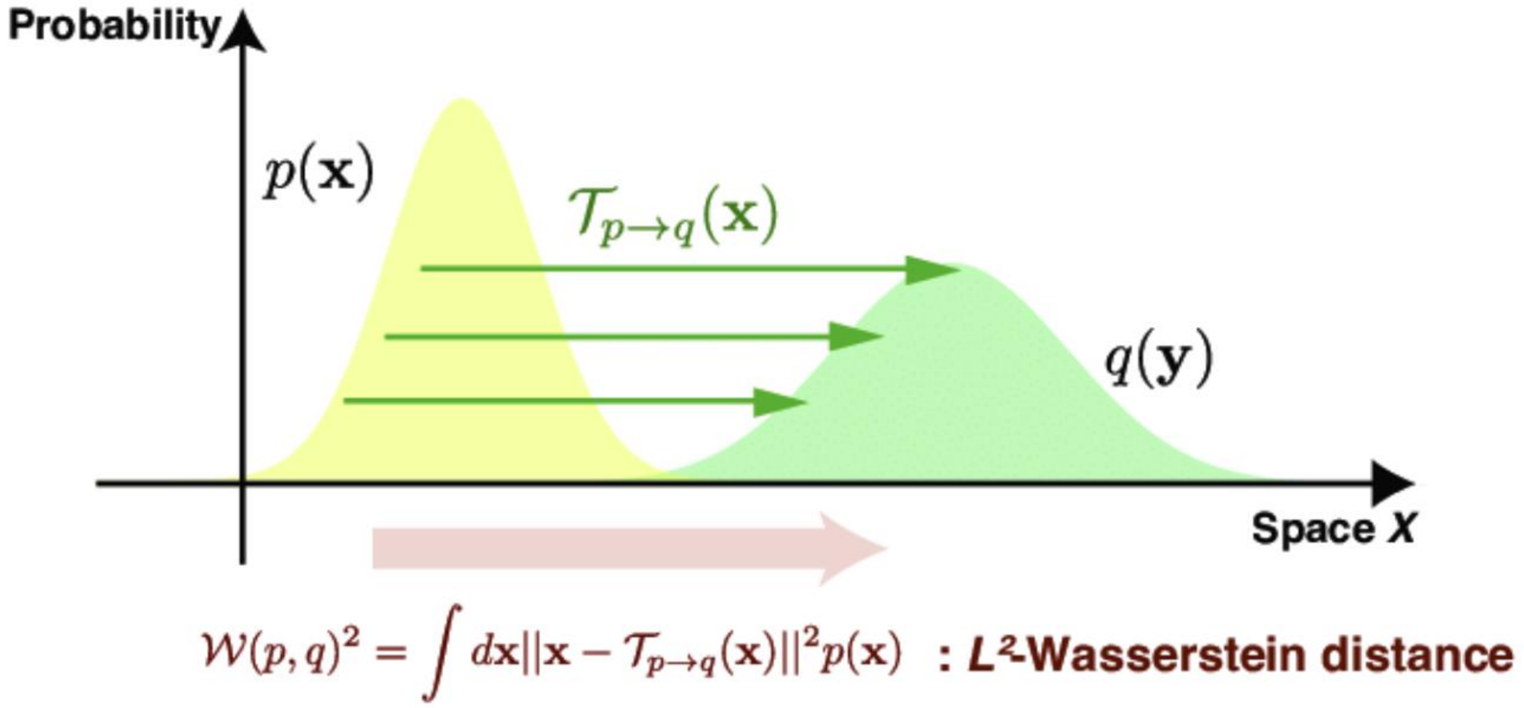
- We control how much to update the discriminator in each training loop
- The discriminator has the easier problem, classification is much easier than generation
- When the discriminator gets too good, gradients vanish and the generator stops updating
- When the discriminator is bad, the generator can start producing the same output for every input (mode collapse)

Wasserstein GANs

- What if we could train the discriminator to convergence and still be able to train our generator?
- We want a function that can tell us how likely it is an image came from the real distribution, but not have vanishing gradients
- Solution: Use a “critic” instead of a discriminator
- The critic outputs a score (value) for the generator rather than a probability (i.e., don’t use a sigmoid for activation…)

Wasserstein Distance

Wasserstein (Earth Mover's) Distance: How much work is it to move one probability distribution p, to another distribution q.



Probability
$p(\mathbf{x})$
$\mathcal{T}_{p\to q}(\mathbf{x})$
$q(\mathbf{y})$
Space *X*
$\mathcal{W}(p,q)^2 = \int d\mathbf{x}||\mathbf{x} - \mathcal{T}_{p\to q}(\mathbf{x})||^2 p(\mathbf{x})$
: L^2-Wasserstein distance

Source: https://www.researchgate.net/figure/Schematic-of-the-L-2-Wasserstein-distance-We-here-consider-optimal-transport-from-the_fig1_349704621

Wasserstein Distance

- Integrals are often hard to compute…
- Use Kantorovich–Rubinstein duality to simplify computation

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$

For the real and generated distributions

f is any function that is 1-Lipschitz continuous

Sup is the supremum, or the lowest upper bound

Looks a lot like our old loss function!

Derivation: https://abdulfatir.com/blog/2020/Wasserstein-Distance/

Lipschitz Continuity

- For this method of calculating Wasserstein Distance, we need our critic to be Lipschitz continuous (with c=1)
- That means that maximum gradient has to be 1
- There are fancy ways to do this (i.e., spectral normalization), but what if just *clip* the gradient
- If the gradient is larger than 1, just clip the value to be 1.
- tf.clip_by_value(grads, -1, 1)

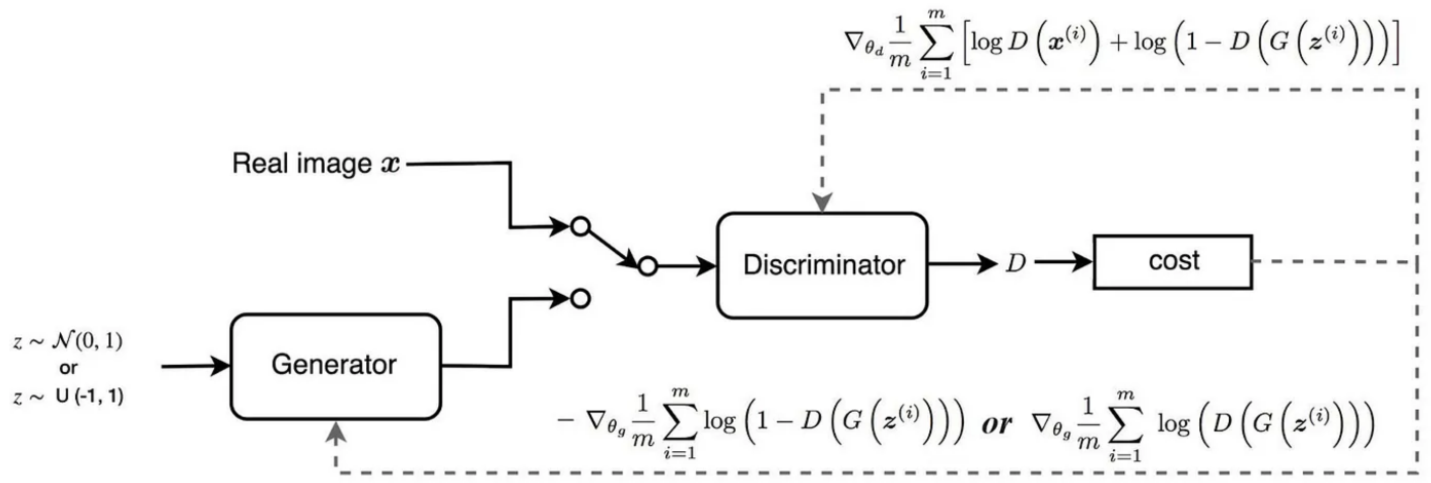
Any questions?

Key difference for WGANs:

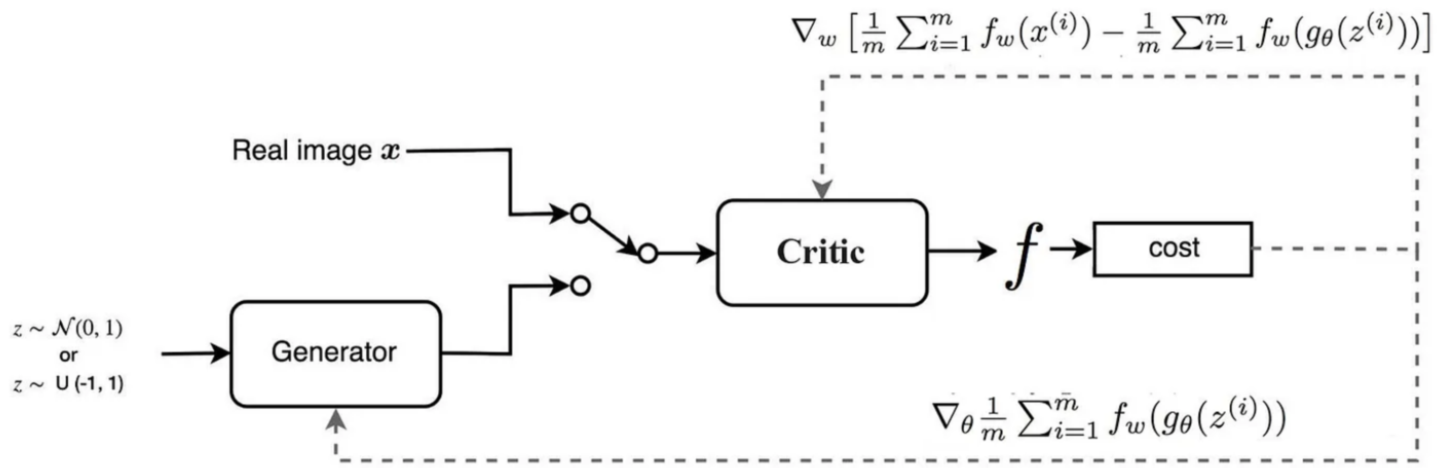
Instead of using the log probability of the output of a discriminator, use the output of the critic

Wasserstein GANs are more stable than vanilla GANs because it’s less susceptible to vanishing gradients from the discriminator

GAN:


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

WGAN

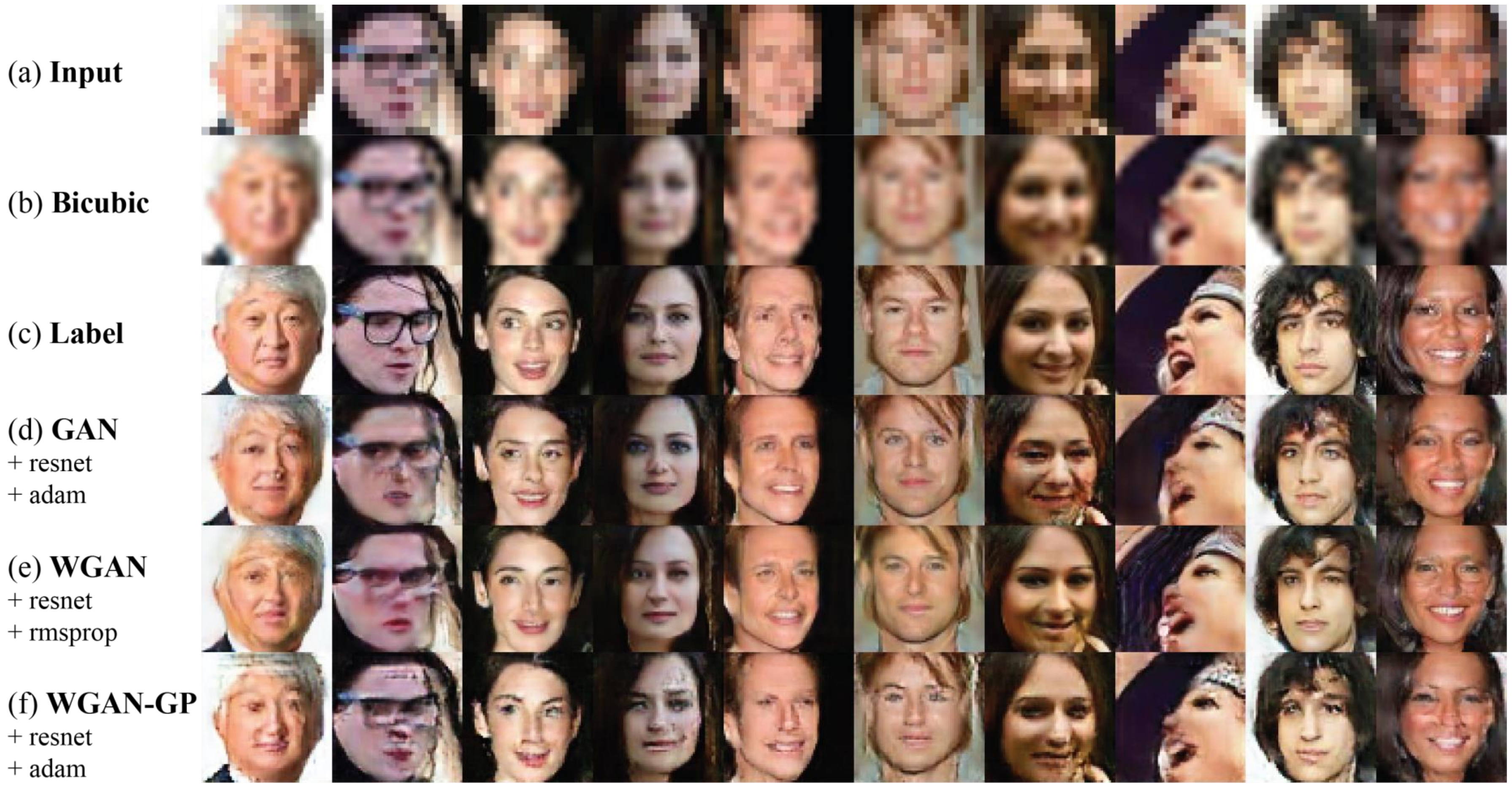

$\nabla_w \left[\frac{1}{m} \sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$
Real image $\boldsymbol{x}$
Critic
f
cost
$z \sim \mathcal{N}(0,1)$ or $z \sim$ U (-1, 1)
Generator
$\nabla_\theta \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))$

Other Tasks

- What if instead of generating images, we wanted to do another task
  - Denoising Images: Given a corrupted or noisy image, can you remove the noise?
  - Colorization: Given a black and white image, can you produce a color image?
  - Super-resolution: given an image of low resolution, can you produce an image of higher resolution (i.e., more pixels)?

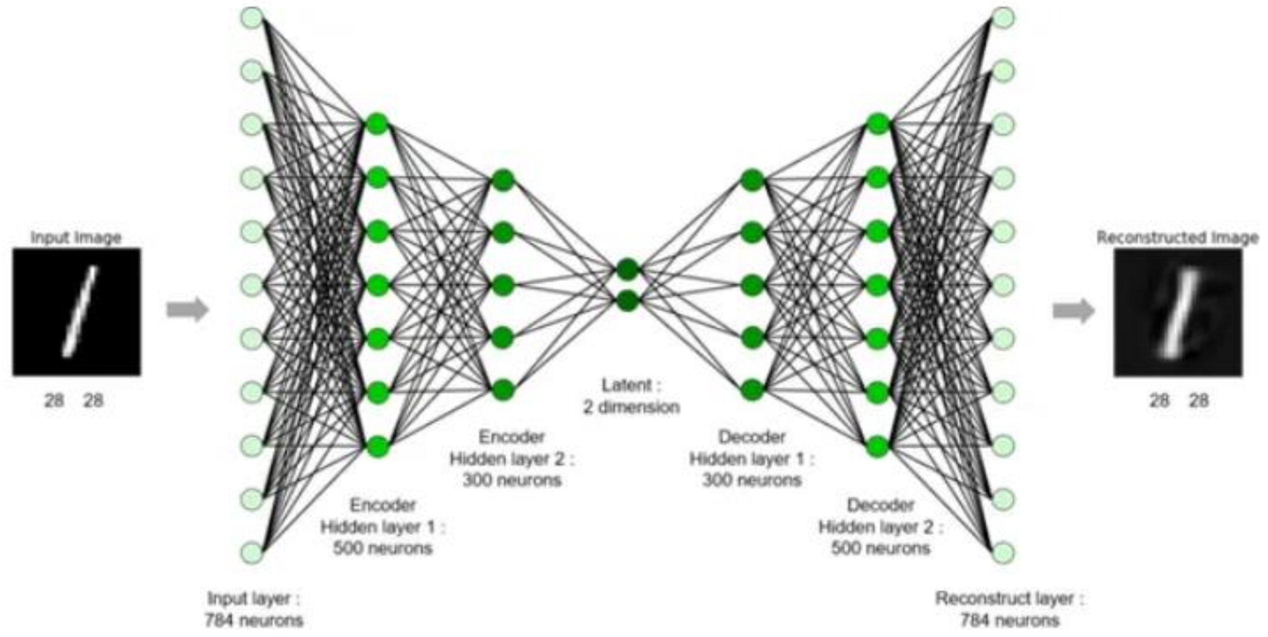
How would you frame each of these problems and train a GAN (or VAE)?

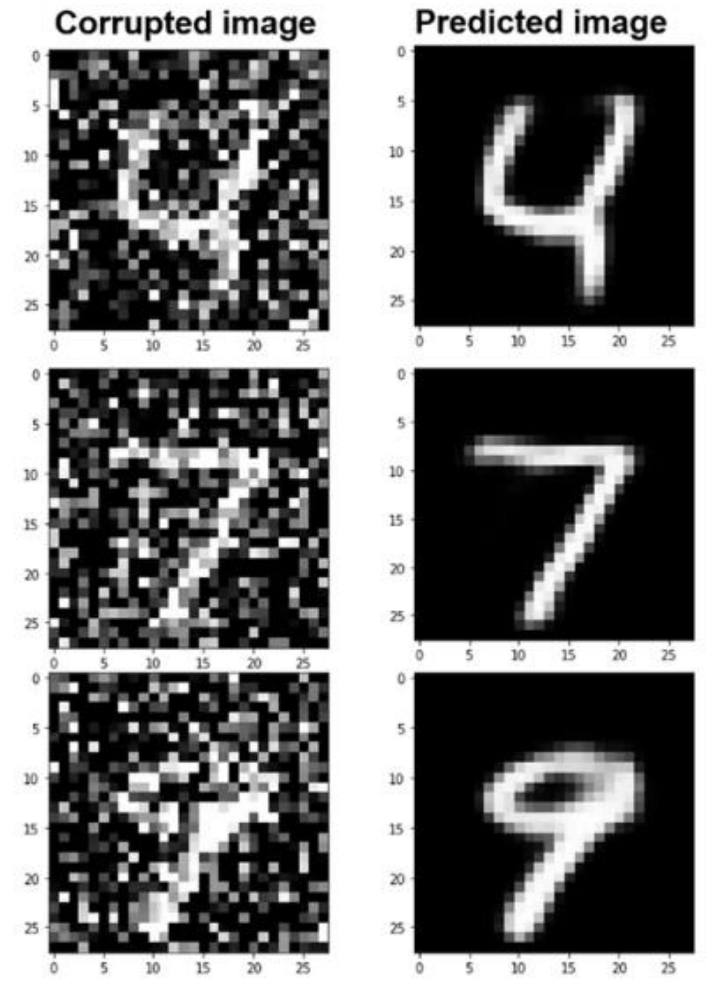
Input low resolution image to generator, have it output a higher resolution image



Source: https://ar5iv.labs.arxiv.org/html/1705.02438

Denoising Auto Encoders (DAEs)

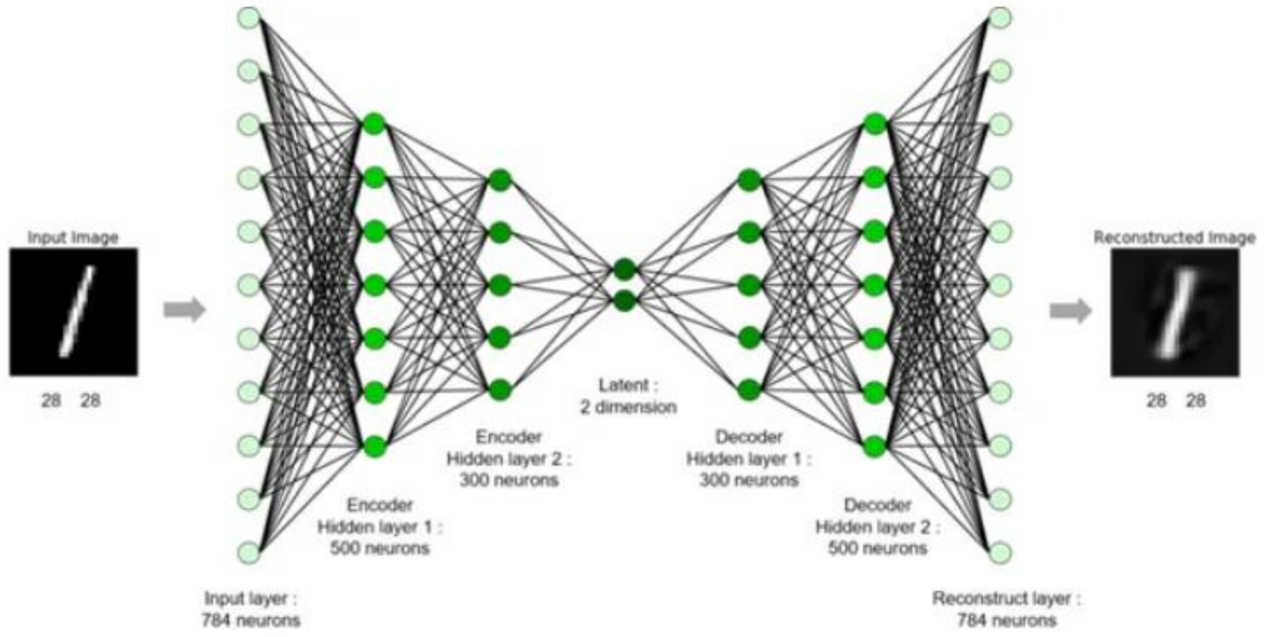

Input Image
28 28
Input layer : 784 neurons
Encoder Hidden layer 1 : 500 neurons
Encoder Hidden layer 2 : 300 neurons
Latent : 2 dimension
Decoder Hidden layer 1 : 300 neurons
Decoder Hidden layer 2 : 500 neurons
Reconstruct layer : 784 neurons
Reconstructed Image
28 28

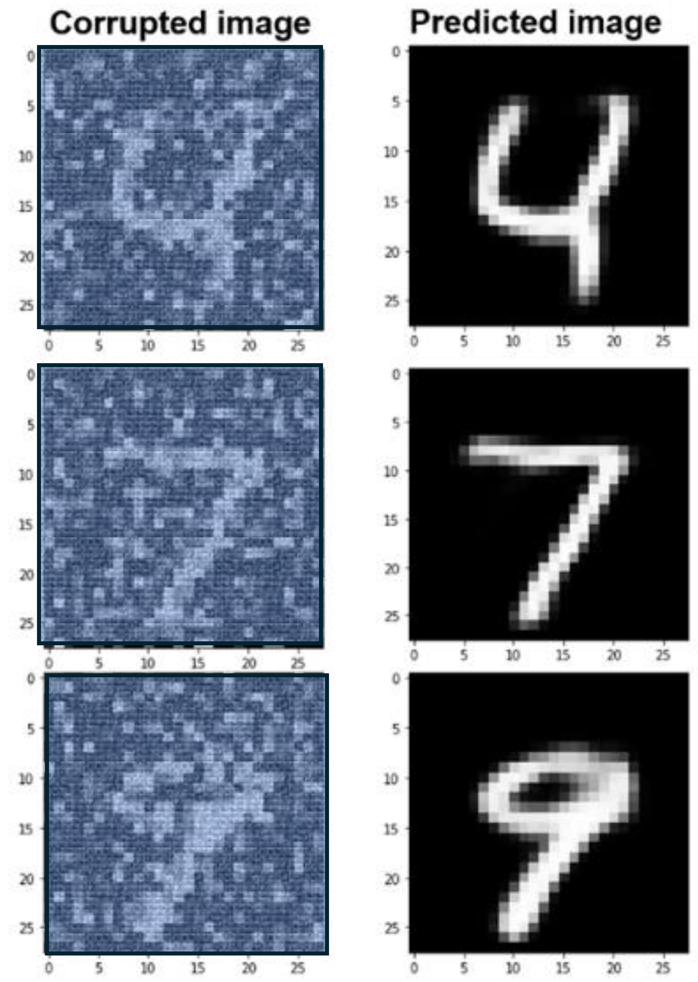


https://www.omdena.com/blog/denoising-autoencoders

Denoising Auto Encoders (DAEs)

What if our images were even noisier?


Input Image
28 28
Input layer : 784 neurons
Encoder Hidden layer 1 : 500 neurons
Encoder Hidden layer 2 : 300 neurons
Latent : 2 dimension
Decoder Hidden layer 1 : 300 neurons
Decoder Hidden layer 2 : 500 neurons
Reconstruct layer : 784 neurons
Reconstructed Image
28 28



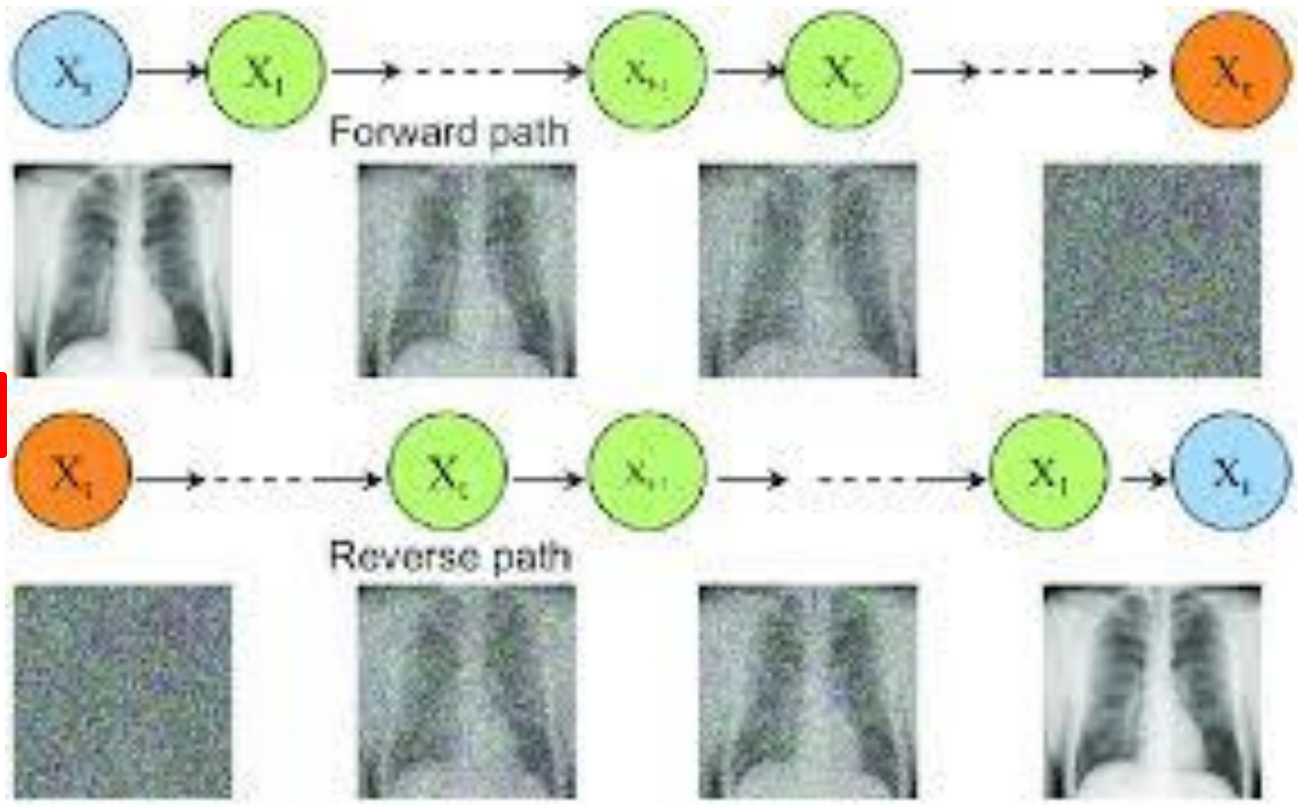
https://www.omdena.com/blog/denoising-autoencoders

Denoising Auto Encoders

Denoising all the noise in one step may be too hard to learn.

What if we added and removed noise incrementally?

Monday: Diffusion Models



Source: https://www.researchgate.net/figure/Schematic-of-the-diffusion-model-training-process_fig1_383920783

GAN Variants

There are many variations on GANs… (these are just some of the ones with architectural differences)

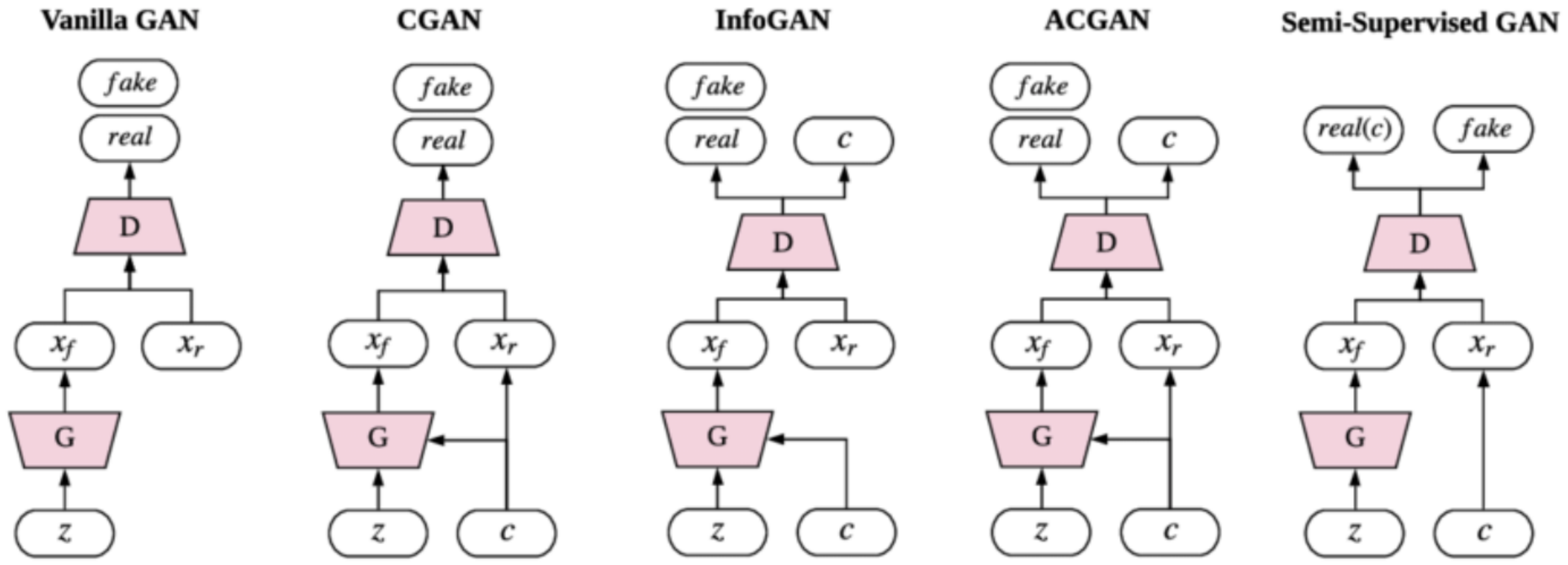


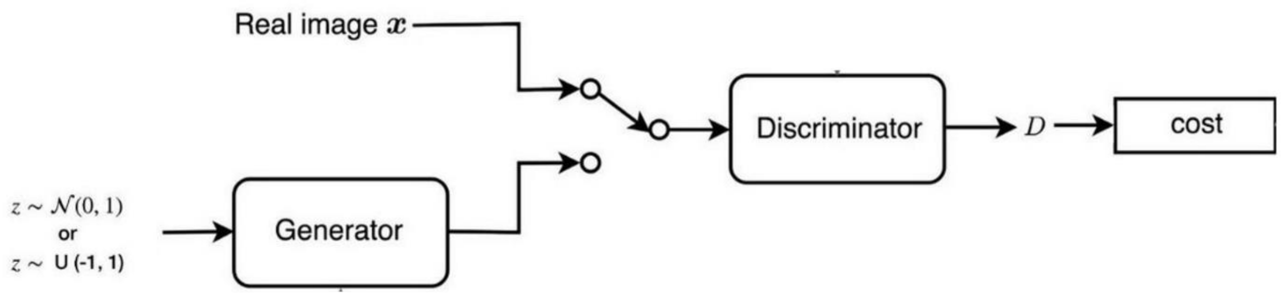
Image source: https://www.researchgate.net/figure/A-view-of-variants-of-GAN-G-represents-the-generator-network-D-is-the-discriminator_fig2_343786287

Recap

GANs

WGANs

Adapting GANs for things other than just generation


Real image x
$z \sim \mathcal{N}(0,1)$
or
$z \sim$ U (-1, 1)
Generator
Discriminator
D
cost