CS349: Generative Adversarial Networks (GANs)

Asif Ekbal

Department of Computer Science and Engineering

Indian Institute of Technology Patna

Generative Adversarial Network (GAN)

- **Generative**
  - Learn a generative model

- **Adversarial**
  - Trained in an adversarial setting

- **Networks**
  - Use Deep Neural Networks

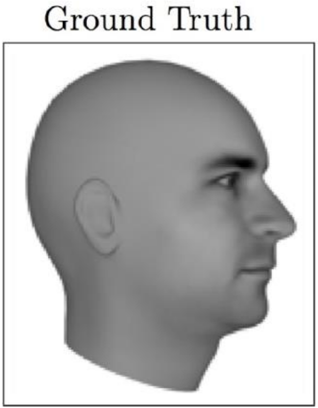
Why Generative Models?

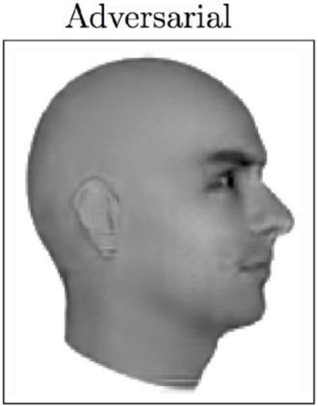
- **We've only seen discriminative models so far**
  - Given an image **X**, predict a label **Y**
  - Estimates **P(Y|X)**

- **Discriminative models have several key limitations**
  - Can't model **P(X)**, i.e. the probability of seeing a certain image
  - Thus, can't sample from **P(X)**, i.e. **can't generate new images**

- **Generative models (in general) cope with all of above**
  - Can model **P(X)**
  - Can generate new images/any other data sample

Magic of GANs...





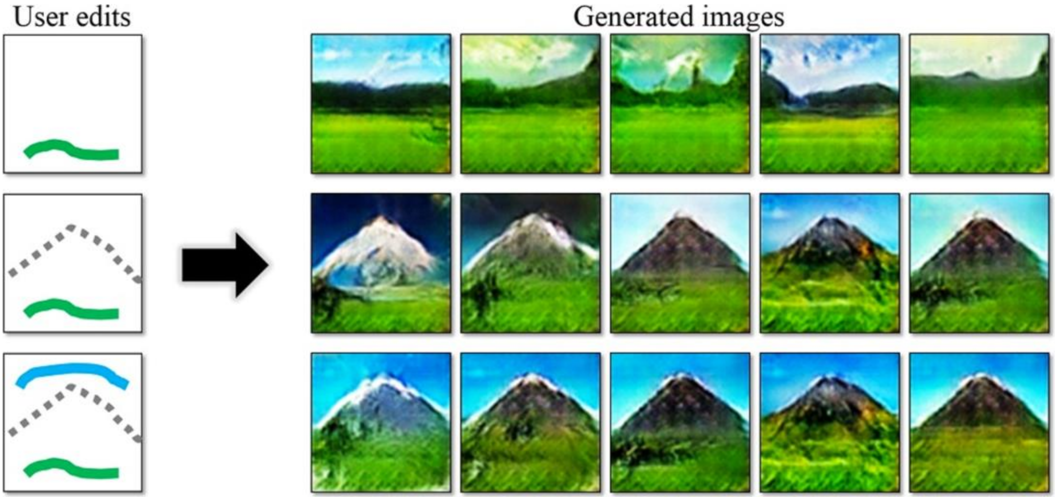
Lotter, William, Gabriel Kreiman, and David Cox. "Unsupervised learning of visual structure using predictive generative networks." *arXiv preprint arXiv:1511.06380* (2015).

Magic of GANs...

Which one is Computer generated?

Ledig, Christian, et al. "Photo-realistic single image super-resolution using a generative adversarial network." *arXiv preprint arXiv:1609.04802* (2016).

Magic of GANs...



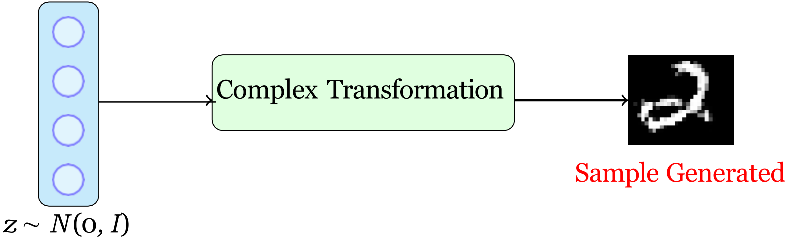
http://people.eecs.berkeley.edu/~junyanz/projects/gvm/

Adversarial Training

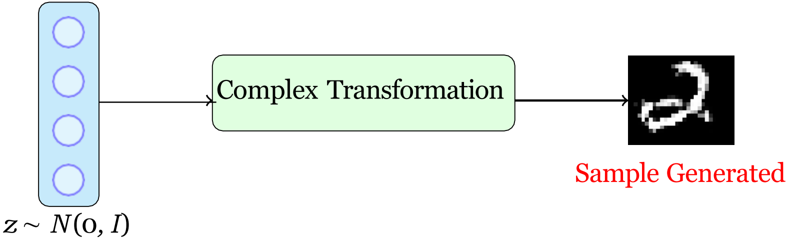
- **Important points**
  - We can generate adversarial samples to fool a discriminative model
  - We can use those adversarial samples to make models robust
  - We then require more effort to generate adversarial samples
  - Repeat this and we get better discriminative model

- **GANs extend that idea to generative models**
  - **Generator**: generate fake samples, tries to fool the *Discriminator*
  - **Discriminator**: tries to distinguish between *real and fake samples*
  - Train them against each other
  - Repeat this and we get better *Generator* and *Discriminator*

- So far we have looked at generative models which explicitly model the joint probability distribution or conditional probability distribution



- GANs take a different approach to this problem where the idea is to sample from a simple tractable distribution (say, $z \sim N(0, I)$) and then learn a complex transformation from this to the training distribution



- GANs take a different approach to this problem where the idea is to sample from a simple tractable distribution (say, $z \sim N(0, I)$) and then learn a complex transformation from this to the training distribution
- In other words, we will take a $z \sim N(0, I)$, learn to make a series of complex transformations on it so that the output looks as if it came from our training distribution

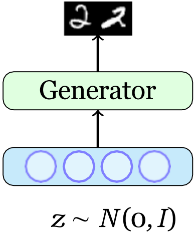
- What can we use for such a complex transformation?

- What can we use for such a complex transformation? A Neural Network

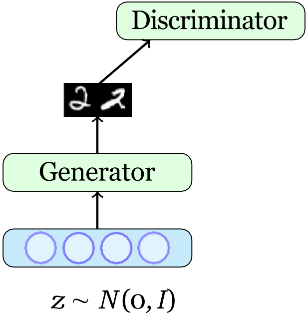
- What can we use for such a complex transformation? A Neural Network
- How do you train such a neural network?

- What can we use for such a complex transformation? A Neural Network
- How do you train such a neural network? Using a two player game

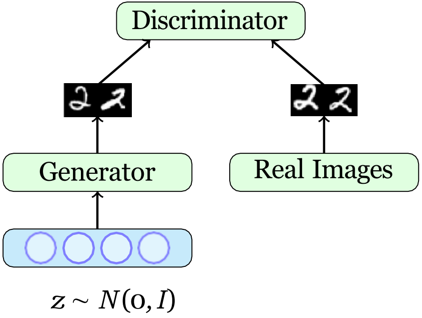
- What can we use for such a complex transformation? A Neural Network
- How do you train such a neural network? Using a two player game
- There are two players in the game:



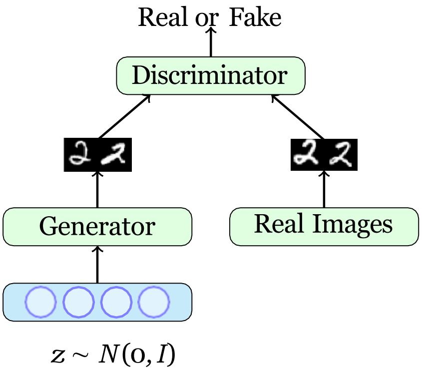
- What can we use for such a complex transformation? A Neural Network
- How do you train such a neural network? Using a two player game
- There are two players in the game: a generator



- What can we use for such a complex transformation? A Neural Network
- How do you train such a neural network? Using a two player game
- There are two players in the game: a generator and a discriminator



- What can we use for such a complex transformation? A Neural Network
- How do you train such a neural network? Using a two player game
- There are two players in the game: a generator and a discriminator
- The job of the generator is to produce images which look so natural that the discriminator thinks that the images came from the real data distribution



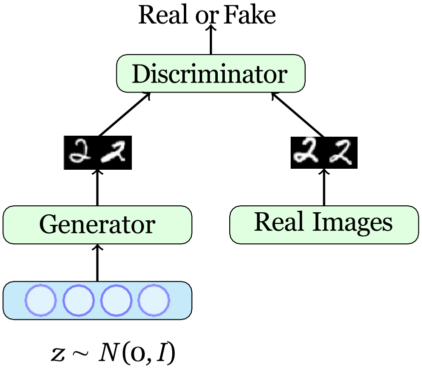
- What can we use for such a complex transformation? A Neural Network
- How do you train such a neural network? Using a two player game
- There are two players in the game: a generator and a discriminator
- The job of the generator is to produce images which look so natural that the discriminator thinks that the images came from the real data distribution
- The job of the discriminator is to get better and better at distinguishing between true images and generated (fake) images

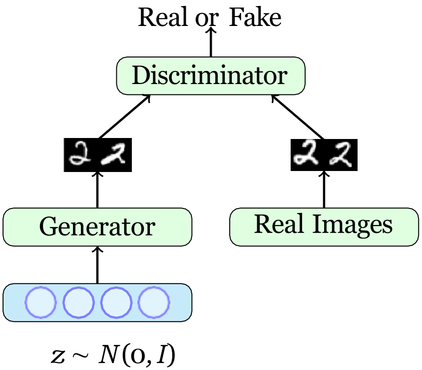
View of GAN

- The simplest way of looking at a GAN is as a *generator network* that is trained to produce realistic samples by introducing an adversary i.e. the *discriminator network*, whose job is to detect if a given sample is “real” or “fake”

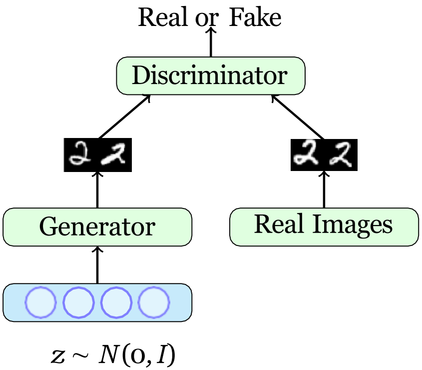
- Discriminator is a dynamically-updated evaluation metric for the tuning of the generator
- Both, the generator and discriminator continuously improve until an equilibrium point is reached
- **Generator**
    - improves as it receives feedback as to how well its generated samples managed to fool the discriminator
- **Discriminator**
    - improves by being shown not only the “fake” samples generated by the generator, but also “real” samples drawn from a real-life distribution
    - learns what generated samples look like and what real samples look like, thus enabling it to give better feedback to the generator

- So let's look at the full picture

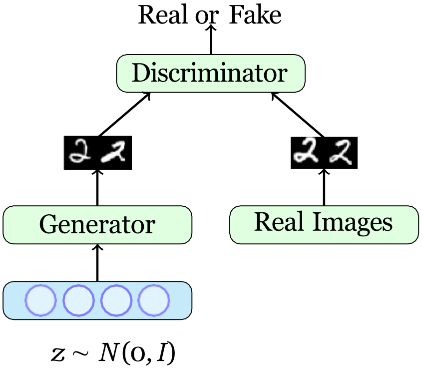




- So let's look at the full picture
- Let G_φ be the generator and D_θ be the discriminator (φ and θ are the parameters of G and D, respectively)

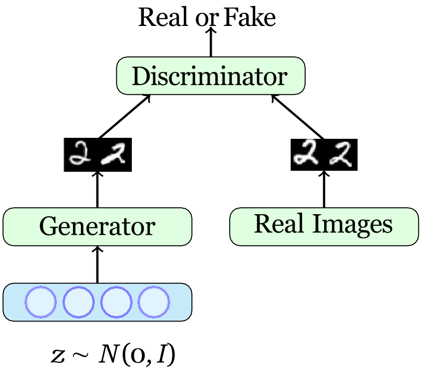


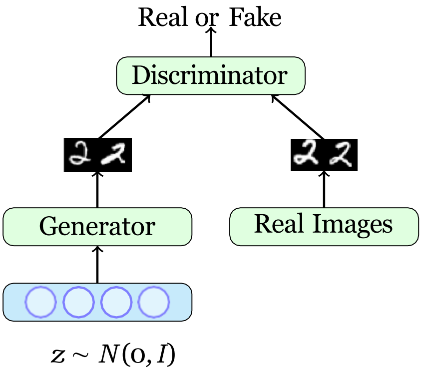
- So let's look at the full picture
- Let G_φ be the generator and D_θ be the discriminator (φ and θ are the parameters of G and D, respectively)
- We have a neural network based generator which takes as input a noise vector $z \sim N(0, I)$ and produces $G_\varphi(z) = X$



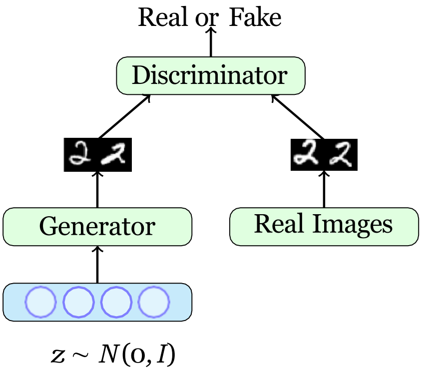
- So let's look at the full picture
- Let G_φ be the generator and D_θ be the discriminator (φ and θ are the parameters of G and D, respectively)
- We have a neural network based generator which takes as input a noise vector $z \sim N(0, I)$ and produces $G_\varphi(z) = X$
- We have a neural network based discriminator which could take as input a real X or a generated $X = G_\varphi(z)$ and classify the input as real/fake

- What should be the objective function of the overall network?

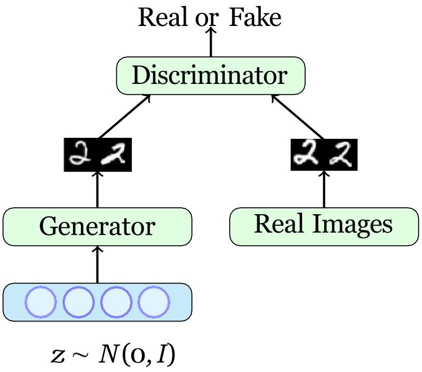




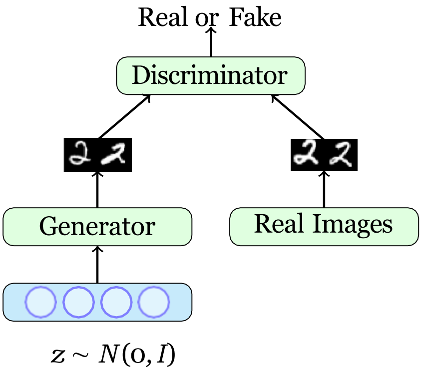
- What should be the objective function of the overall network?
- Let's look at the objective function of the generator first



- What should be the objective function of the overall network?
- Let's look at the objective function of the generator first
- Given an image generated by the generator as $G_\varphi(z)$ the discriminator assigns a score $D_\theta(G_\varphi(z))$ to it

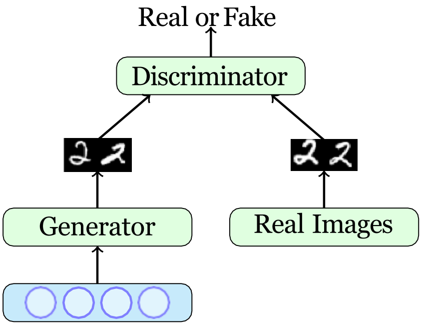


- What should be the objective function of the overall network?
- Let's look at the objective function of the generator first
- Given an image generated by the generator as $G_\varphi(z)$ the discriminator assigns a score $D_\theta(G_\varphi(z))$ to it
- This score will be between 0 and 1 and will tell us the probability of the image being real or fake

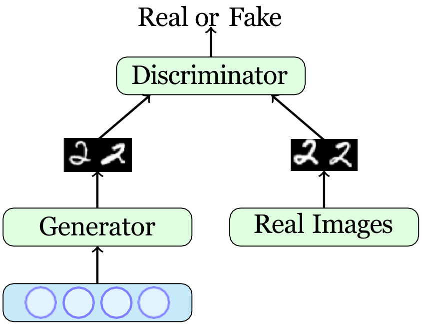


- What should be the objective function of the overall network?
- Let's look at the objective function of the generator first
- Given an image generated by the generator as $G_\varphi(z)$ the discriminator assigns a score $D_\theta\,(G_\varphi(z))$ to it
- This score will be between 0 and 1 and will tell us the probability of the image being real or fake
- For a given z, the generator would want to maximize $\log\ D_\theta\ (G_\varphi(z))$ (log likelihood) or minimize $\log(1 - D_\theta(G_\varphi(z)))$

- This is just for a single z and the generator would like to do this for all possible values of z,



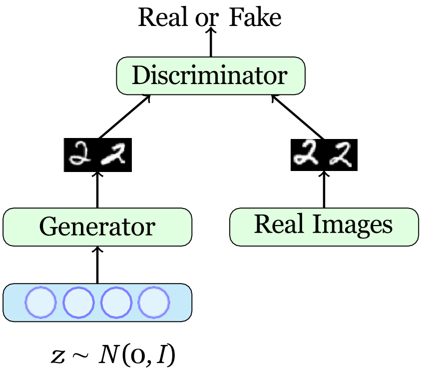
$z \sim N(0, I)$



$z \sim N(0, I)$

- This is just for a single z and the generator would like to do this for all possible values of z,
- For example, if z was discrete and drawn from a uniform distribution ($i.e.$, $p(z) = \frac{1}{N}\ \forall z$) then the generator's objective function would be

$$\min_{\varphi} \sum_{i=1}^{N} \frac{1}{N} \log(1 - D_\theta(G_\varphi(z)))$$



- This is just for a single z and the generator would like to do this for all possible values of z,
- For example, if z was discrete and drawn from a uniform distribution (*i.e.*, $p(z) = \frac{1}{N}$ $\forall z$)then the generator's objective function would be

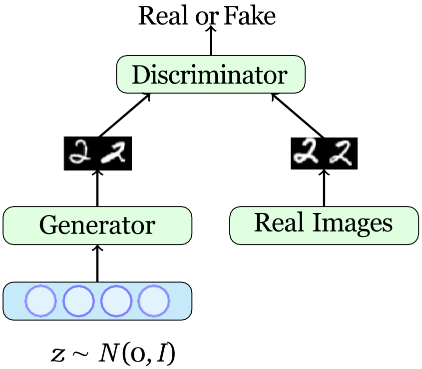
$$\min_{\varphi} \sum_{i=1}^{N} \frac{1}{N} \log(1 - D_\theta(G_\varphi(z)))$$

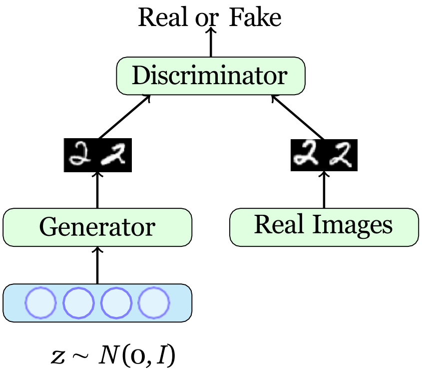
- However, in our case, z is continuous and not uniform ($z \sim N(0, I)$) so the equivalent objective function would be

$$\min_{\varphi} \int p(z) \log(1 - D_\theta(G_\varphi(z)))$$

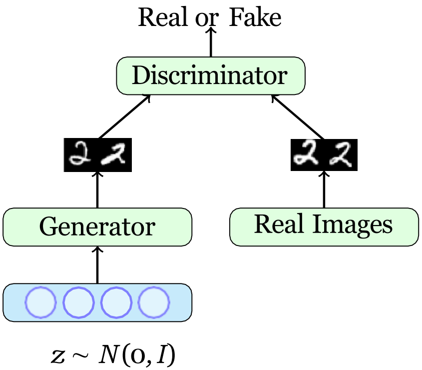
$$\min_{\varphi} E_{z \sim p(z)} [\log(1 - D_\theta(G_\varphi(z)))]$$

- Now let's look at the discriminator

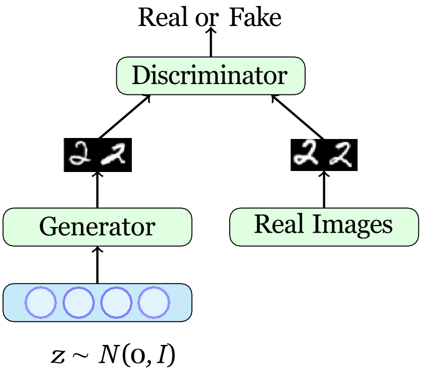




- Now let's look at the discriminator
- The task of the discriminator is to assign a high score to real images and a low score to fake images

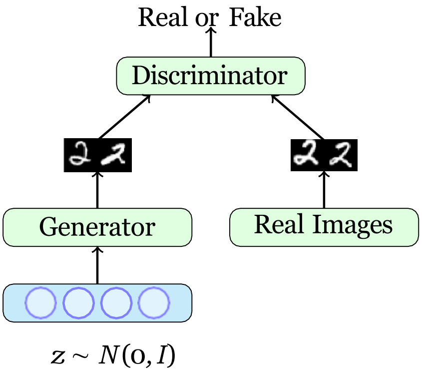


- Now let's look at the discriminator
- The task of the discriminator is to assign a high score to real images and a low score to fake images
- And it should do this for all possible real images and all possible fake images



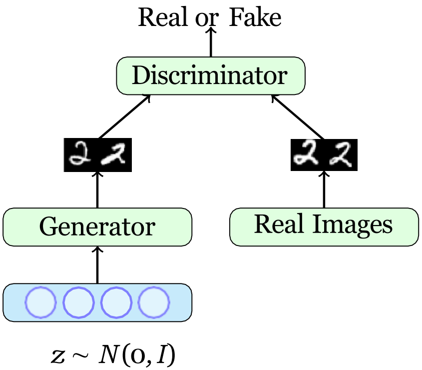
- Now let's look at the discriminator
- The task of the discriminator is to assign a high score to real images and a low score to fake images
- And it should do this for all possible real images and all possible fake images
- In other words, it should try to maximize the following objective function

$$\max_{\theta} E_{x \sim p_{data}} [\log D_\theta(x)] + E_{z \sim p(z)}[\log(1 - D_\theta(G_\phi(z)))]$$



- If we put the objectives of the generator and discriminator together we get a minimax game

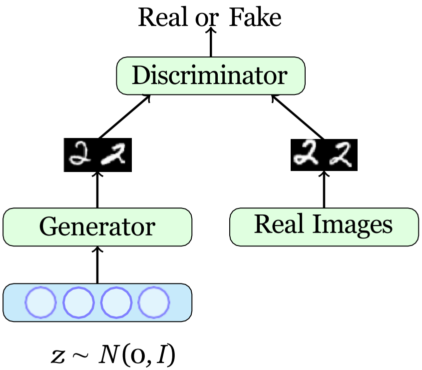
$$\min_{\varphi} \max_{\theta} \; [\mathbb{E}_{x \sim p_{data}} \log D_\theta(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_\theta(G_\varphi(z)))]$$



- If we put the objectives of the generator and discriminator together we get a minimax game

$$\min_{\varphi} \max_{\theta} \; [\mathbb{E}_{x\sim p_{data}} \log D_\theta(x) + \mathbb{E}_{z\sim p(z)} \log(1 - D_\theta(G_\varphi(z)))]$$

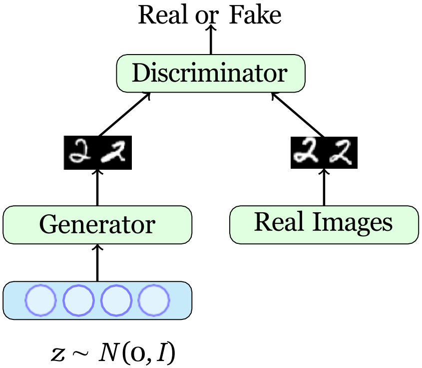
- The first term in the objective is only w.r.t. the parameters of the discriminator (θ)



- If we put the objectives of the generator and discriminator together we get a minimax game

$$\min_{\varphi} \max_{\theta} \; [\mathbb{E}_{x \sim p_{data}} \log D_\theta(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_\theta(G_\varphi(z)))]$$

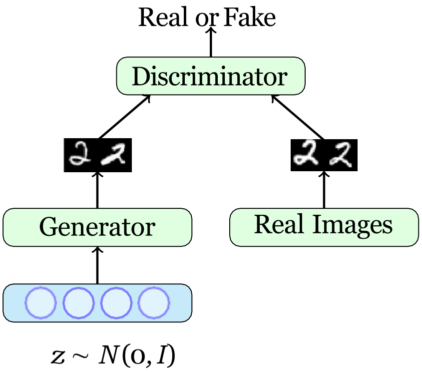
- The first term in the objective is only w.r.t. the parameters of the discriminator (θ)
- The second term in the objective is w.r.t. the parameters of the generator (φ) as well as the discriminator (θ)



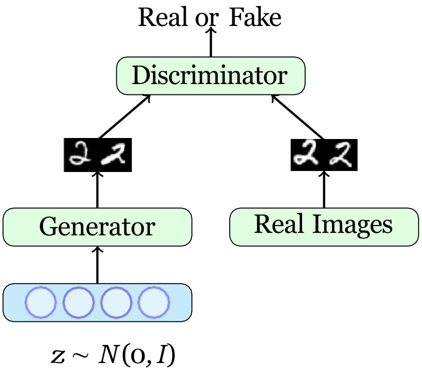
- If we put the objectives of the generator and discriminator together we get a minimax game

$$\min_{\phi} \max_{\theta} \; [\mathbb{E}_{x \sim p_{data}} \log D_\theta(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_\theta(G_\phi(z)))]$$

- The first term in the objective is only w.r.t. the parameters of the discriminator (θ)
- The second term in the objective is w.r.t. the parameters of the generator (ϕ) as well as the discriminator (θ)
- The discriminator wants to maximize the second term whereas the generator wants to minimize it (hence it is a two-player game)

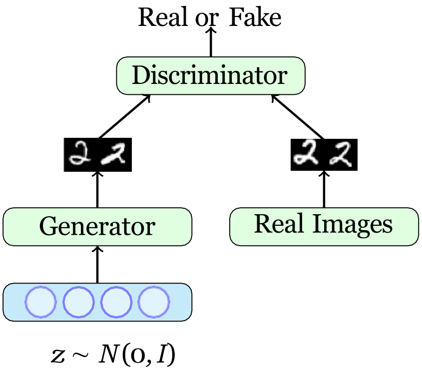


- So the overall training proceeds by alternating between these two step



- So the overall training proceeds by alternating between these two step
- **Step 1:** Gradient Ascent on Discriminator

$$\max_{\theta} \; [\mathbb{E}_{x \sim p_{data}} \log D_\theta(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_\theta(G_\phi(z)))]$$

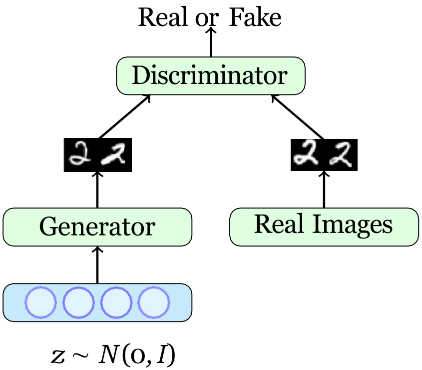


- So the overall training proceeds by alternating between these two step
- **Step 1:** Gradient Ascent on Discriminator

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

- **Step 2:** Gradient Descent on Generator

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



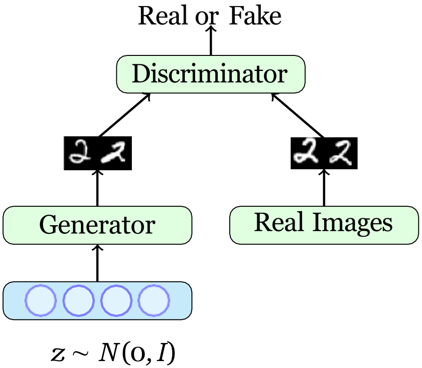
- So the overall training proceeds by alternating between these two step
- **Step 1:** Gradient Ascent on Discriminator

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

- **Step 2:** Gradient Descent on Generator

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

- In practice, the above generator objective does not work well and we use a slightly modified objective



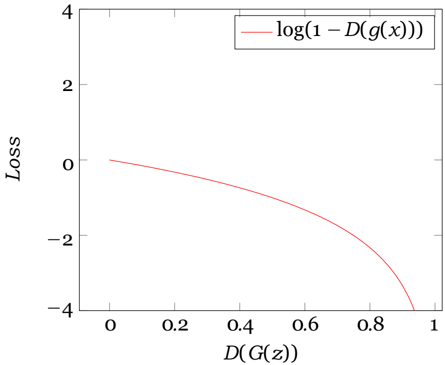
- So the overall training proceeds by alternating between these two step
- **Step 1:** Gradient Ascent on Discriminator

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

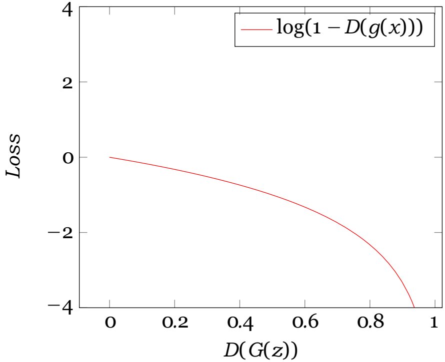
- **Step 2:** Gradient Descent on Generator

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

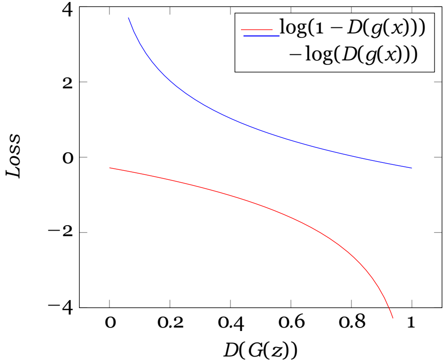
- In practice, the above generator objective does not work well and we use a slightly modified objective
- Let us see why



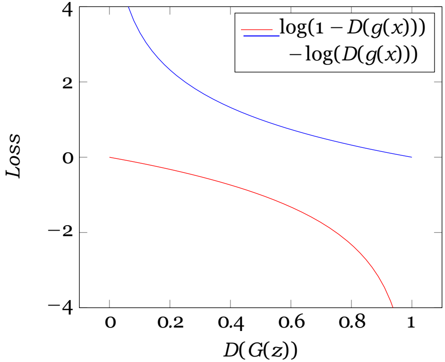
- When the sample is likely fake, we want to give a feedback to the generator (using gradients)



- When the sample is likely fake, we want to give a feedback to the generator (using gradients)
- However, in this region where $D(G(z))$ is close to 0, the curve of the loss function is very flat and the gradient would be close to 0



- When the sample is likely fake, we want to give a feedback to the generator (using gradients)
- However, in this region where $D(G(z))$ is close to 0, the curve of the loss function is very flat and the gradient would be close to 0
- Trick: **Instead of minimizing the likelihood of the discriminator being correct, maximize the likelihood of the discriminator being wrong**



- When the sample is likely fake, we want to give a feedback to the generator (using gradients)
- However, in this region where $D(G(z))$ is close to 0, the curve of the loss function is very flat and the gradient would be close to 0
- Trick: Instead of minimizing the likelihood of the discriminator being correct, maximize the likelihood of the discriminator being wrong
- In effect, the objective remains the same but the gradient signal becomes better

With that we are now ready to see the full algorithm for training GANs

procedure GAN TRAINING
  for number of training iterations **do**
    for k steps **do**
      • Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, .., \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$
      • Sample minibatch of m examples $\{\mathbf{x}^{(1)}, .., \mathbf{x}^{(m)}\}$ from data generating distribution $p_{data}(\mathbf{x})$
      • Update the discriminator by ascending its stochastic gradient:

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

    end for
    • Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, .., \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$
    • Update the generator by ascending its stochastic gradient

$$\nabla_\phi \frac{1}{m} \sum_{i=1}^{m} \left[\log \left(D_\theta \left(G_\phi \left(z^{(i)} \right) \right) \right) \right]$$

  end for
end procedure

- We will now delve a bit deeper into the objective function used by GANs and see what it implies

- We will now delve a bit deeper into the objective function used by GANs and see what it implies
- Suppose we denote the true data distribution by $p_{data}(x)$ and the distribution of the data generated by the model as $p_G(x)$

- We will now delve a bit deeper into the objective function used by GANs and see what it implies
- Suppose we denote the true data distribution by $p_{data}(x)$ and the distribution of the data generated by the model as $p_G(x)$
- What do we wish should happen at the end of training?

- We will now delve a bit deeper into the objective function used by GANs and see what it implies
- Suppose we denote the true data distribution by $p_{data}(x)$ and the distribution of the data generated by the model as $p_G(x)$
- What do we wish should happen at the end of training?

$$p_G(x) = p_{data}(x)$$

- We will now delve a bit deeper into the objective function used by GANs and see what it implies
- Suppose we denote the true data distribution by $p_{data}(x)$ and the distribution of the data generated by the model as $p_G(x)$
- What do we wish should happen at the end of training?

$$p_G(x) = p_{data}(x)$$

- Can we prove this formally even though the model is not explicitly computing this density?

- We will now delve a bit deeper into the objective function used by GANs and see what it implies
- Suppose we denote the true data distribution by $p_{data}(x)$ and the distribution of the data generated by the model as $p_G(x)$
- What do we wish should happen at the end of training?

$$p_G(x) = p_{data}(x)$$

- Can we prove this formally even though the model is not explicitly computing this density?
- We will try to prove this over the next few slides

Theorem

The global minimum of the virtual training criterion $C(G) = \max_D V(G, D)$ is achieved **if and only if** $p_G = p_{data}$

Theorem

The global minimum of the virtual training criterion $C(G) = \max_D V(G, D)$ is achieved **if and only if** $p_G = p_{data}$

is equivalent to

Theorem

The global minimum of the virtual training criterion $C(G) = \max_D V(G, D)$ is achieved **if and only if** $p_G = p_{data}$

is equivalent to

Theorem

1. **If** $p_G = p_{data}$ then the global minimum of the virtual training criterion $C(G) = \max_D V(G, D)$ is achieved **and**

Theorem

The global minimum of the virtual training criterion $C(G) = \max_D V(G, D)$ is achieved **if and only if** $p_G = p_{data}$

is equivalent to

Theorem

1. **If** $p_G = p_{data}$ then the global minimum of the virtual training criterion $C(G) = \max_D V(G, D)$ is achieved **and**
2. The global minimum of the virtual training criterion $C(G) = \max_D V(G, D)$ is achieved **only if** $p_G = p_{data}$

Outline of the Proof

The 'if' part: The global minimum of the virtual training criterion $C(G) = \max_D V(G, D)$ is achieved **if** $p_G = p_{data}$

The 'only if' part: The global minimum of the virtual training criterion $C(G) = \max_D V(G, D)$ is achieved **only if** $p_G = p_{data}$

Outline of the Proof

The 'if' part: The global minimum of the virtual training criterion $C(G) = \max_D V(G, D)$ is achieved **if** $p_G = p_{data}$

(a) Find the value of $V(D, G)$ when the generator is optimal *i.e.*, when $p_G = p_{data}$

The 'only if' part: The global minimum of the virtual training criterion $C(G) = \max_D V(G, D)$ is achieved **only if** $p_G = p_{data}$

Outline of the Proof

The 'if' part: The global minimum of the virtual training criterion $C(G) = \max_D V(G, D)$ is achieved **if** $p_G = p_{data}$

(a) Find the value of $V(D, G)$ when the generator is optimal *i.e.*, when $p_G = p_{data}$

(b) Find the value of $V(D, G)$ for other values of the generator *i.e.*, for any p_G such that $p_G \neq p_{data}$

The 'only if' part: The global minimum of the virtual training criterion $C(G) = \max_D V(G, D)$ is achieved **only if** $p_G = p_{data}$

Outline of the Proof

The 'if' part: The global minimum of the virtual training criterion $C(G) = \max_D V(G, D)$ is achieved **if** $p_G = p_{data}$

(a) Find the value of $V(D, G)$ when the generator is optimal *i.e.*, when $p_G = p_{data}$

(b) Find the value of $V(D, G)$ for other values of the generator *i.e.*, for any p_G such that $p_G \neq p_{data}$

(c) Show that $a < b \; \forall p_G \neq p_{data}$ (and hence the minimum $V(D, G)$ is achieved when $p_G = p_{data}$)

The 'only if' part: The global minimum of the virtual training criterion $C(G) = \max_D V(G, D)$ is achieved **only if** $p_G = p_{data}$

- Show that when $V(D, G)$ is minimum then $p_G = p_{data}$

Outline of the Proof

The 'if' part: The global minimum of the virtual training criterion $C(G) = \max_D V(G, D)$ is achieved **if** $p_G = p_{data}$

(a) Find the value of $V(D, G)$ when the generator is optimal *i.e.*, when $p_G = p_{data}$

(b) Find the value of $V(D, G)$ for other values of the generator *i.e.*, for any p_G such that $p_G \neq p_{data}$

(c) Show that $a < b \; \forall p_G \neq p_{data}$ (and hence the minimum $V(D, G)$ is achieved when $p_G = p_{data}$)

The 'only if' part: The global minimum of the virtual training criterion $C(G) = \max_D V(G, D)$ is achieved **only if** $p_G = p_{data}$

- Show that when $V(D, G)$ is minimum then $p_G = p_{data}$

- First let us look at the objective function again

$$\min_{\phi} \max_{\theta} \quad [\mathbb{E}_{x \sim p_{data}} \log D_\theta(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_\theta(G_\phi(z)))]$$

- We will expand it to its integral form

$$\min_{\phi} \max_{\theta} \int_x p_{data}(x) \log D_\theta(x) + \int_z p(z) \log(1 - D_\theta(G_\phi(z)))$$

- Let $p_G(X)$ denote the distribution of the X's generated by the generator and since X is a function of z we can replace the second integral as shown below

$$\min_{\phi} \max_{\theta} \int_x p_{data}(x) \log D_\theta(x) + \int_x p_G(x) \log(1 - D_\theta(x))$$

- Okay, so our revised objective is given by

$$\min_{\phi}\ \max_{\theta}\ \int_x \left(p_{data}(x)\log D_\theta(x) + p_G(x)\log(1 - D_\theta(x))\right) dx$$

- Given a generator G, we are interested in finding the optimum discriminator D which will maximize the above objective function
- The above objective will be maximized when the quantity inside the integral is maximized $\forall x$
- To find the optima we will take the derivative of the term inside the integral w.r.t. D and set it to zero

$$\begin{aligned}
\frac{d}{d(D_\theta(x))}\left(p_{data}(x)\log D_\theta(x) + p_G(x)\log(1 - D_\theta(x))\right) &= 0\\
p_{data}(x)\frac{1}{D_\theta(x)} + p_G(x)\frac{1}{1 - D_\theta(x)}(-1) &= 0\\
\frac{p_{data}(x)}{D_\theta(x)} &= \frac{p_G(x)}{1 - D_\theta(x)}\\
(p_{data}(x))(1 - D_\theta(x)) &= (p_G(x))(D_\theta(x))\\
D_\theta(x) &= \frac{p_{data}(x)}{p_G(x) + p_{data}(x)}
\end{aligned}$$

- This means for any given generator

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

- Now the if part of the theorem says "if $p_G = p_{data}$ ...."
- So let us substitute $p_G = p_{data}$ into $D_G^*(G(x))$ and see what happens to the loss functions

$$D_G^* = \frac{p_{data}}{p_{data} + p_G} = \frac{1}{2}$$

$$\begin{aligned} V(G, D_G^*) &= \int_x p_{data}(x) \log D(x) + p_G(x) \log\left(1 - D(x)\right) dx \\ &= \int_x p_{data}(x) \log \frac{1}{2} + p_G(x) \log\left(1 - \frac{1}{2}\right) dx \\ &= \log 2 \int_x p_G(x) dx - \log 2 \int_x p_{data}(x) dx \\ &= -2 \log 2 \qquad = -\log 4 \end{aligned}$$

Outline of the Proof

The 'if' part: The global minimum of the virtual training criterion $C(G) = \max_D V(G, D)$ is achieved **if** $p_G = p_{data}$

(a) Find the value of $V(D, G)$ when the generator is optimal *i.e.*, when $p_G = p_{data}$

(b) Find the value of $V(D, G)$ for other values of the generator *i.e.*, for any p_G such that $p_G \neq p_{data}$

(c) Show that $a < b \ \forall \ p_G \neq p_{data}$(and hence the minimum $V(D, G)$ is achieved when $p_G = p_{data}$)

The 'only if' part: The global minimum of the virtual training criterion $C(G) = \max_D V(G, D)$ is achieved **only if** $p_G = p_{data}$

- Show that when $V(D, G)$ is minimum then $p_G = p_{data}$

- So what we have proved so far is that if the generator is optimal ($p_G = p_{data}$) the discriminator's loss value is $-\log 4$
- We still haven't proved that this is the minima
- For example, it is possible that for some $p_G \neq p_{data}$, the discriminator's loss value is lower than $-\log 4$
- To show that the discriminator achieves its lowest value "if $p_G = p_{data}$", we need to show that for all other values of p_G the discriminator's loss value is greater than $-\log 4$

- To show this we will get rid of the assumption that $p_G = p_{data}$

$$
\begin{aligned}
C(G) &= \int_x \left[p_{data}(x) \log \left(\frac{p_{data}(x)}{p_G(x) + p_{data}(x)} \right) + p_G(x) \log \left(1 - \frac{p_{data}(x)}{p_G(x) + p_{data}(x)} \right) \right] dx \\
&= \int_x \left[p_{data}(x) \log \left(\frac{p_{data}(x)}{p_G(x) + p_{data}(x)} \right) + p_G(x) \log \left(\frac{p_G(x)}{p_G(x) + p_{data}(x)} \right) + (\log 2 - \log 2)(p_{data} + p_G) \right] \\
&= -\log 2 \int_x (p_G(x) + p_{data}(x))\, dx \\
&\quad + \int_x \left[p_{data}(x) \left(\log 2 + \log \left(\frac{p_{data}(x)}{p_G(x) + p_{data}(x)} \right) \right) + p_G(x) \left(\log 2 + \log \left(\frac{p_G(x)}{P p_G(x) + p_{data}(x)} \right) \right) \right] dx \\
&= -\log 2(1 + 1) \\
&\quad + \int_x \left[p_{data}(x) \log \left(\frac{p_{data}(x)}{\frac{p_G(x)+p_{data}(x)}{2}} \right) + p_G(x) \log \left(\frac{p_G(x)}{\frac{p_G(x)+p_{data}(x)}{2}} \right) \right] dx \\
&= -\log 4 + KL \left(p_{data} \| \frac{p_G(x) + p_{data}(x)}{2} \right) + KL \left(p_G \| \frac{p_G(x) + p_{data}(x)}{2} \right)
\end{aligned}
$$

Outline of the Proof

The 'if' part: The global minimum of the virtual training criterion $C(G) = \max_D V(G, D)$ is achieved **if** $p_G = p_{data}$

(a) Find the value of $V(D, G)$ when the generator is optimal *i.e.*, when $p_G = p_{data}$

(b) Find the value of $V(D, G)$ for other values of the generator *i.e.*, for any p_G such that $p_G \neq p_{data}$

(c) Show that $a < b \ \forall \ p_G \neq p_{data}$(and hence the minimum $V(D, G)$ is achieved when $p_G = p_{data}$)

The 'only if' part: The global minimum of the virtual training criterion $C(G) = \max_D V(G, D)$ is achieved **only if** $p_G = p_{data}$

- Show that when $V(D, G)$ is minimum then $p_G = p_{data}$

- Okay, so we have

$$C(G) = -\log 4 + KL\left(p_{data}||\frac{p_{data}+p_g}{2}\right) + KL\left(p_G||\frac{p_{data}+p_G}{2}\right)$$

- We know that KL divergence is always ≥ 0

$$\therefore C(G) \geq -\log 4$$

- Hence the minimum possible value of $C(G)$ is $-\log 4$
- But this is the value that $C(G)$ achieves when $p_G = p_{data}$ (and this is exactly what we wanted to prove)
- We have, thus, proved the **if part** of the theorem

Outline of the Proof

The 'if' part: The global minimum of the virtual training criterion $C(G) = \max_D V(G, D)$ is achieved **if** $p_G = p_{data}$

(a) Find the value of $V(D, G)$ when the generator is optimal *i.e.*, when $p_G = p_{data}$

(b) Find the value of $V(D, G)$ for other values of the generator *i.e.*, for any p_G such that $p_G \neq p_{data}$

(c) Show that $a < b \; \forall \; p_G \neq p_{data}$(and hence the minimum $V(D, G)$ is achieved when $p_G = p_{data}$)

The 'only if' part: The global minimum of the virtual training criterion $C(G) = \max_D V(G, D)$ is achieved **only if** $p_G = p_{data}$

- Show that when $V(D, G)$ is minimum then $p_G = p_{data}$

- Now let's look at the other part of the theorem
  If the global minimum of the virtual training criterion $C(G) = \max_D V(G, D)$ is achieved then $p_G = p_{data}$
- We know that

$$C(G) = -\log 4 + KL\left(p_{data} \| \frac{p_{data} + p_g}{2}\right) + KL\left(p_G \| \frac{p_{data} + p_G}{2}\right)$$

- If the global minima is achieved then $C(G) = -\log 4$ which implies that

$$KL\left(p_{data} \| \frac{p_{data} + p_g}{2}\right) + KL\left(p_G \| \frac{p_{data} + p_G}{2}\right) = 0$$

- This will happen only when $p_G = p_{data}$ (you can prove this easily)
- In fact $KL\left(p_{data} \| \frac{p_{data}+p_g}{2}\right) + KL\left(p_G \| \frac{p_{data}+p_G}{2}\right)$ is the Jenson-Shannon divergence between p_G and p_{data}

$$KL\left(p_{data} \| \frac{p_{data} + p_g}{2}\right) + KL\left(p_G \| \frac{p_{data} + p_G}{2}\right) = JSD(p_{data} \| p_G)$$

which is minimum only when $p_G = p_{data}$