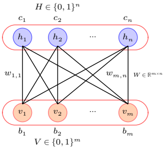
Generative Adversarial Networks (GANs)

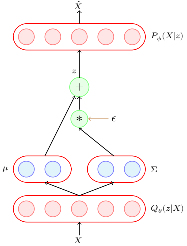
ML Instruction Team, Fall 2022

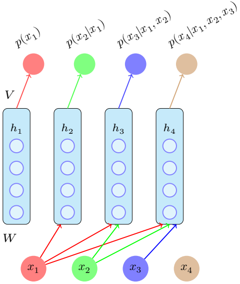
CE Department
Sharif University of Technology

Module 23.1: Generative Adversarial Networks - The intuition

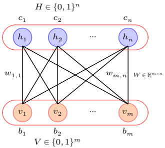
- So far we have looked at generative models which explicitly model the joint probability distribution or conditional probability distribution

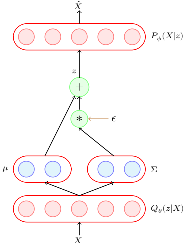


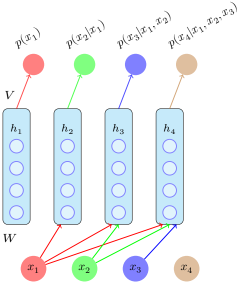




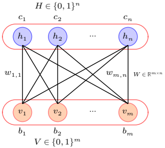
- So far we have looked at generative models which explicitly model the joint probability distribution or conditional probability distribution
- For example, in RBMs we learn $P(X, H)$, in VAEs we learn $P(z|X)$ and $P(X|z)$ whereas in AR models we learn $P(X)$

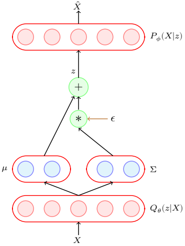


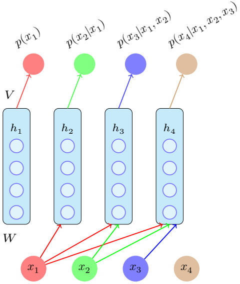




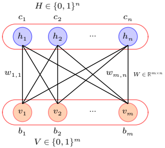
- So far we have looked at generative models which explicitly model the joint probability distribution or conditional probability distribution
- For example, in RBMs we learn $P(X, H)$, in VAEs we learn $P(z|X)$ and $P(X|z)$ whereas in AR models we learn $P(X)$
- What if we are only interested in sampling from the distribution and don't really care about explicit density function $P(X)$?

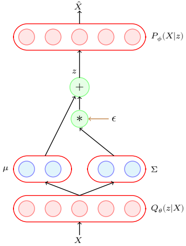


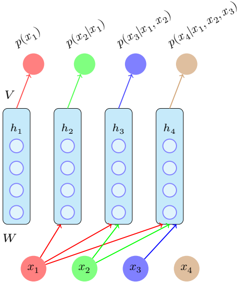




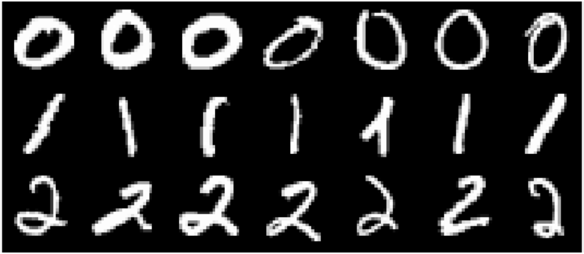
- So far we have looked at generative models which explicitly model the joint probability distribution or conditional probability distribution
- For example, in RBMs we learn $P(X, H)$, in VAEs we learn $P(z|X)$ and $P(X|z)$ whereas in AR models we learn $P(X)$
- What if we are only interested in sampling from the distribution and don't really care about explicit density function $P(X)$?
- What does this mean?



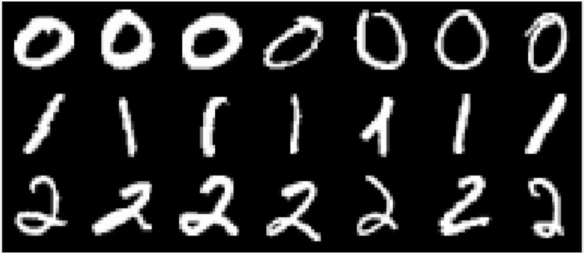




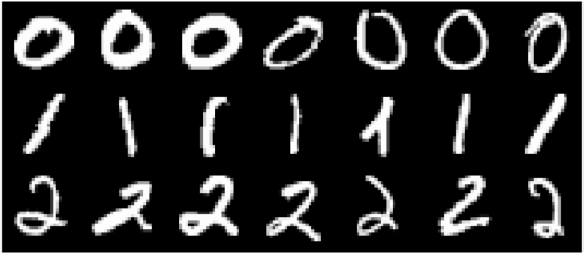
- So far we have looked at generative models which explicitly model the joint probability distribution or conditional probability distribution
- For example, in RBMs we learn $P(X, H)$, in VAEs we learn $P(z|X)$ and $P(X|z)$ whereas in AR models we learn $P(X)$
- What if we are only interested in sampling from the distribution and don't really care about explicit density function $P(X)$?
- What does this mean? Let us see



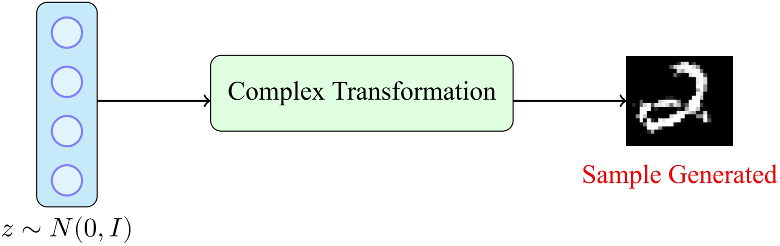
- As usual we are given some training data (say, MNIST images) which obviously comes from some underlying distribution



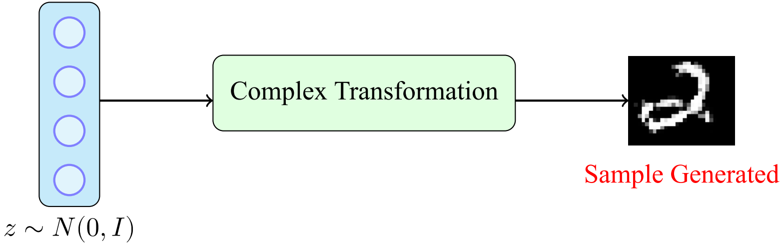
- As usual we are given some training data (say, MNIST images) which obviously comes from some underlying distribution
- Our goal is to generate more images from this distribution (*i.e.*, create images which look similar to the images from the training data)



- As usual we are given some training data (say, MNIST images) which obviously comes from some underlying distribution
- Our goal is to generate more images from this distribution (*i.e.*, create images which look similar to the images from the training data)
- In other words, we want to sample from a complex high dimensional distribution which is intractable (recall RBMs, VAEs and AR models deal with this intractability in their own way)



- 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

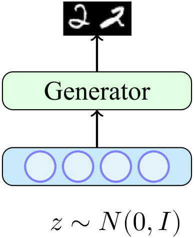
- What can we use for such a complex transformation?

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

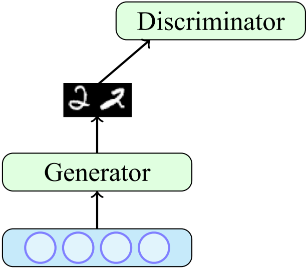
- What can we use for such a complex transformation? A Neural Network
- How do you train such a neural network?

- 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:

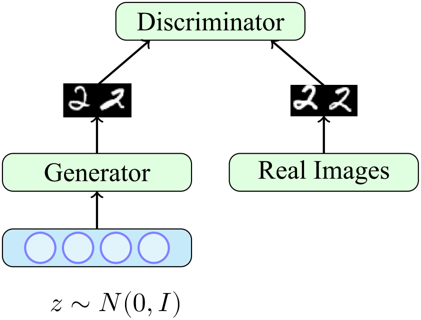


- What can we use for such a complex transformation? A Neural Network
- How do you train such a neural network? Using a two player game
- There are two players in the game: a generator

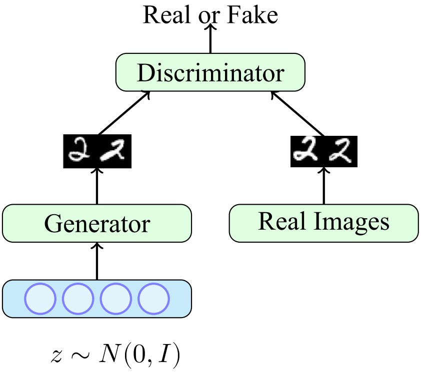


$z \sim N(0, I)$

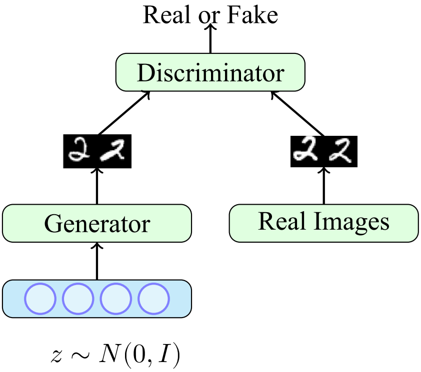
- What can we use for such a complex transformation? A Neural Network
- How do you train such a neural network? Using a two player game
- There are two players in the game: a generator and a discriminator



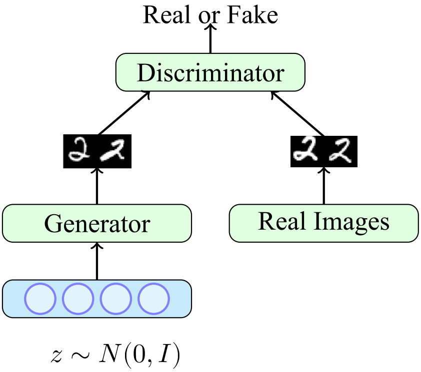
- What can we use for such a complex transformation? A Neural Network
- How do you train such a neural network? Using a two player game
- There are two players in the game: a generator and a discriminator
- The job of the generator is to produce images which look so natural that the discriminator thinks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



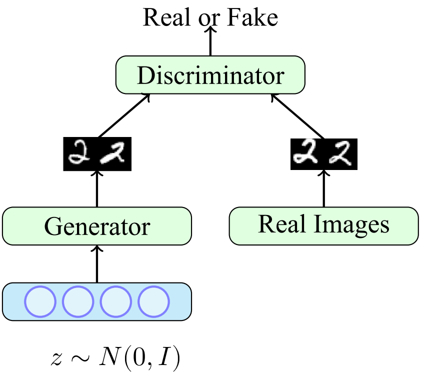
- What can we use for such a complex transformation? A Neural Network
- How do you train such a neural network? Using a two player game
- There are two players in the game: a generator and a discriminator
- The job of the generator is to produce images which look so natural that the discriminator thinks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



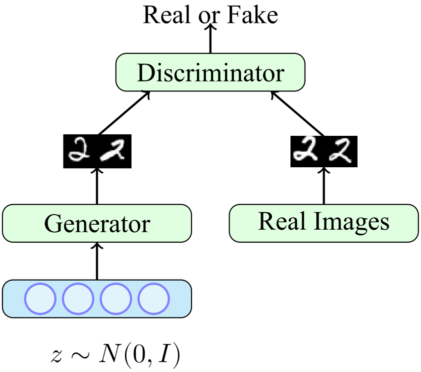
- So let's look at the full picture



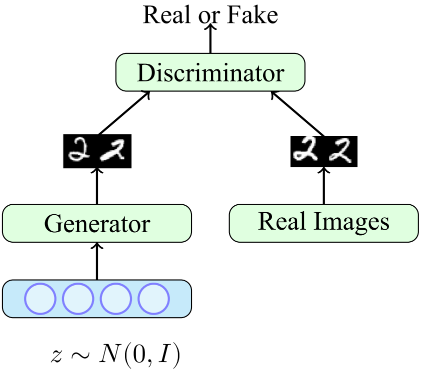
- So let's look at the full picture
- Let G_ϕ be the generator and D_θ be the discriminator (ϕ and θ are the parameters of G and D, respectively)



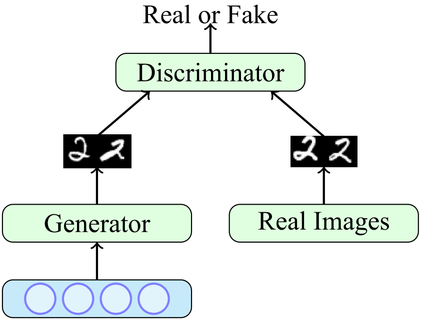
- So let's look at the full picture
- Let G_ϕ be the generator and D_θ be the discriminator (ϕ and θ are the parameters of G and D, respectively)
- We have a neural network based generator which takes as input a noise vector $z \sim N(0, I)$ and produces $G_\phi(z) = X$



- 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 $G_\phi(z) = X$
- We have a neural network based discriminator which could take as input a real X or a generated $X = G_\phi(z)$ and classify the input as real/fake

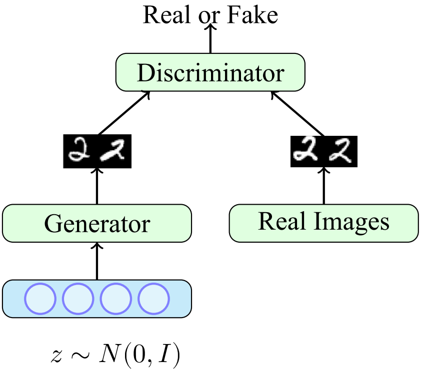


- What should be the objective function of the overall network?

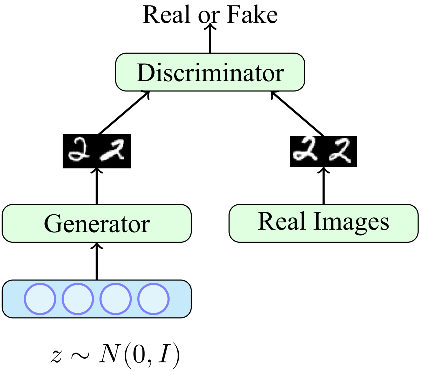


$z \sim N(0, I)$

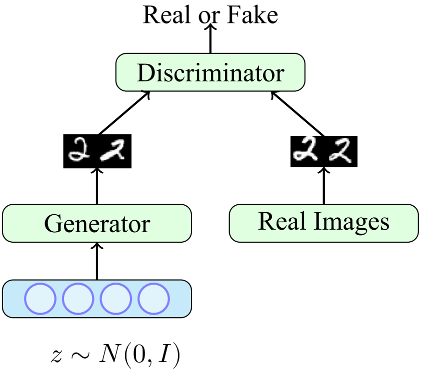
- What should be the objective function of the overall network?
- Let's look at the objective function of the generator first



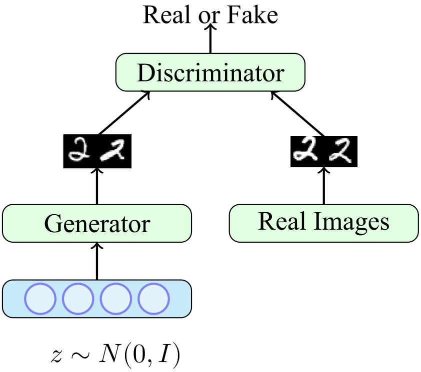
- What should be the objective function of the overall network?
- Let's look at the objective function of the generator first
- Given an image generated by the generator as $G_\phi(z)$ the discriminator assigns a score $D_\theta(G_\phi(z))$ to it



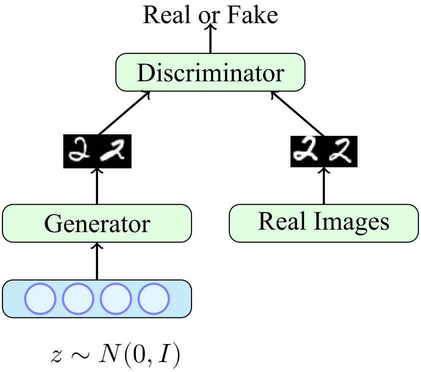
- What should be the objective function of the overall network?
- Let's look at the objective function of the generator first
- Given an image generated by the generator as $G_\phi(z)$ the discriminator assigns a score $D_\theta(G_\phi(z))$ to it
- This score will be between 0 and 1 and will tell us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

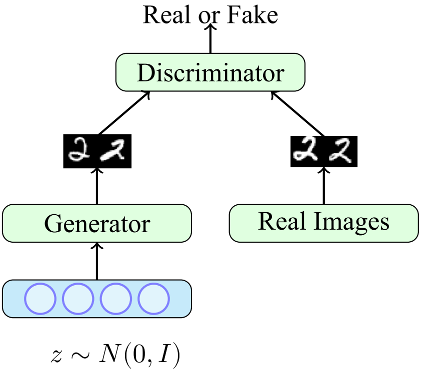


- This is just for a single z and the generator would like to do this for all possible values of z,



- 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

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



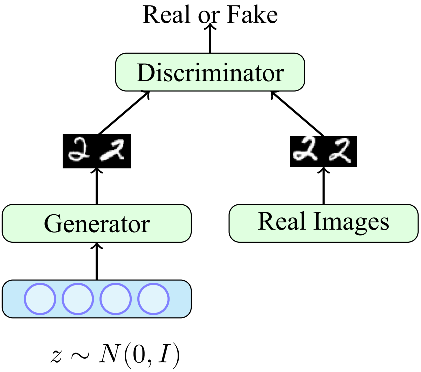
- This is just for a single z and the generator would like to do this for all possible values of z,
- For example, if z was discrete and drawn from a uniform distribution (*i.e.*, $p(z) = \frac{1}{N}\ \forall z$) then the generator's objective function would be

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

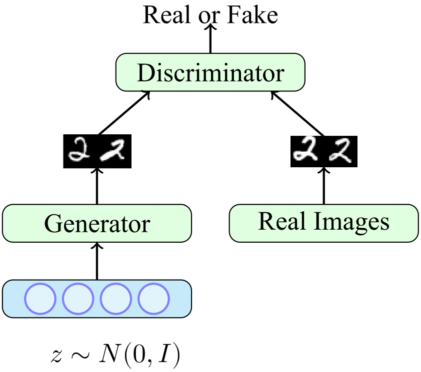
- However, in our case, z is continuous and not uniform ($z \sim N(0, I)$) so the equivalent objective function would be

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

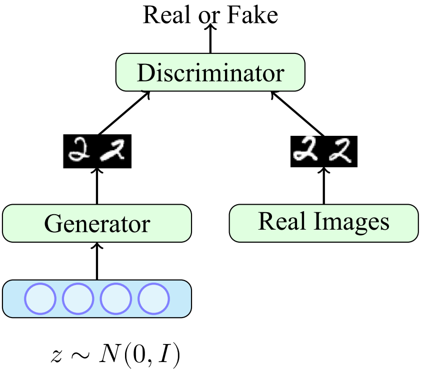
$$\min_{\phi} E_{z \sim p(z)}[\log(1 - D_\theta(G_\phi(z)))]$$



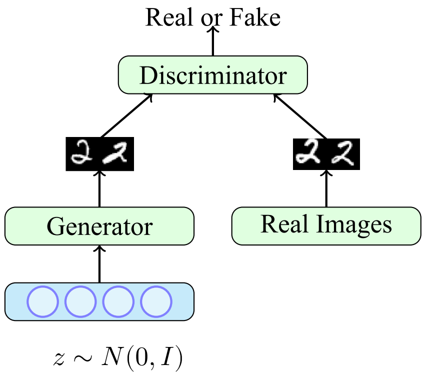
- Now let's look at the discriminator



- 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

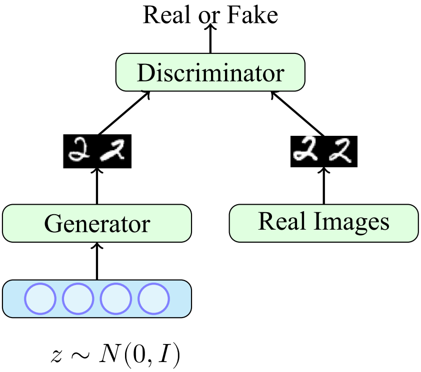


- Now let's look at the discriminator
- The task of the discriminator is to assign a high score to real images and a low score to fake images
- And it should do this for all possible real images and all possible fake images



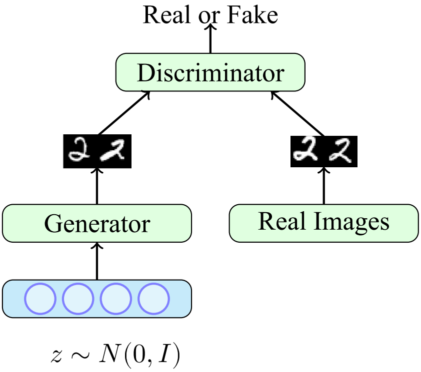
- Now let's look at the discriminator
- The task of the discriminator is to assign a high score to real images and a low score to fake images
- And it should do this for all possible real images and all possible fake images
- In other words, it should try to maximize the following objective function

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



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

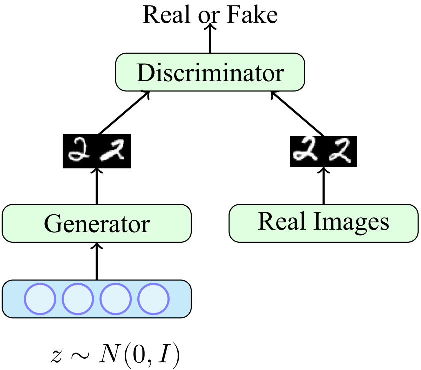
$$\min_{\phi} \; \max_{\theta} \; [\mathbb{E}_{x \sim p_{data}} \, log \, D_\theta(x) + \mathbb{E}_{z \sim p(z)} \, log(1 - D_\theta(G_\phi(z))$$



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

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

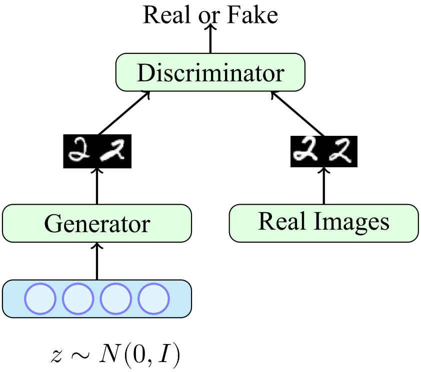
- The first term in the objective is only w.r.t. the parameters of the discriminator (θ)



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

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

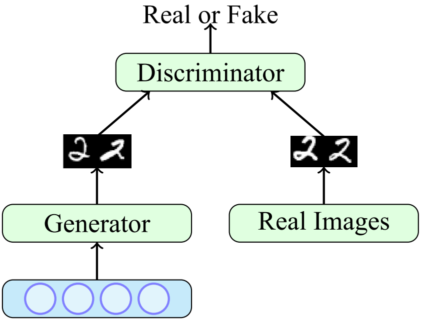
- The first term in the objective is only w.r.t. the parameters of the discriminator (θ)
- The second term in the objective is w.r.t. the parameters of the generator (ϕ) as well as the discriminator (θ)



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

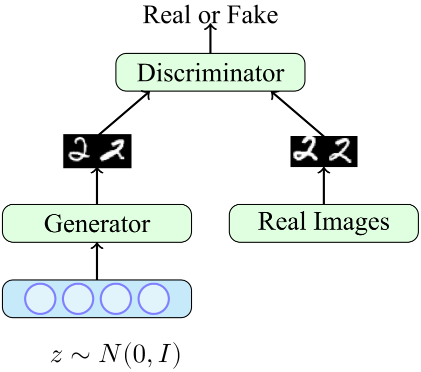
$$\min_{\phi} \max_{\theta} \quad [\mathbb{E}_{x \sim p_{data}} \log D_\theta(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_\theta(G_\phi(z)))$$

- 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)



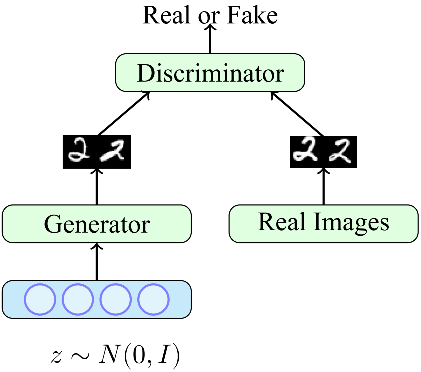
$z \sim N(0, I)$

- 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

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

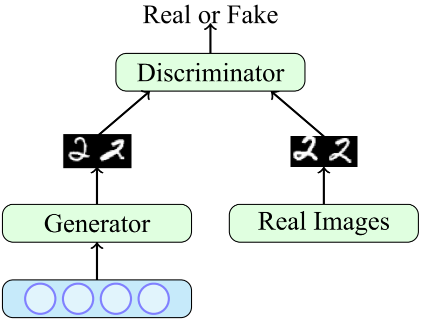


- So the overall training proceeds by alternating between these two step
- **Step 1:** Gradient Ascent on Discriminator

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

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

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



$z \sim N(0, I)$

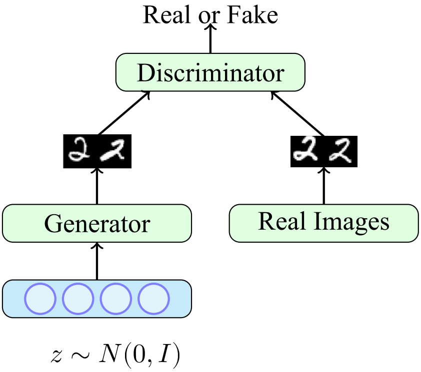
- So the overall training proceeds by alternating between these two step
- **Step 1:** Gradient Ascent on Discriminator

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

- **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

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

- **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

Thank You!

Any Question?