

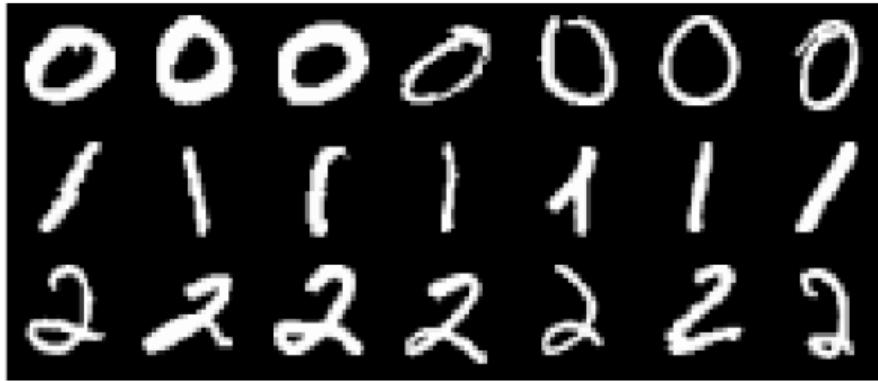
CS7015 (Deep Learning) : Lecture 23

Generative Adversarial Networks (GANs)

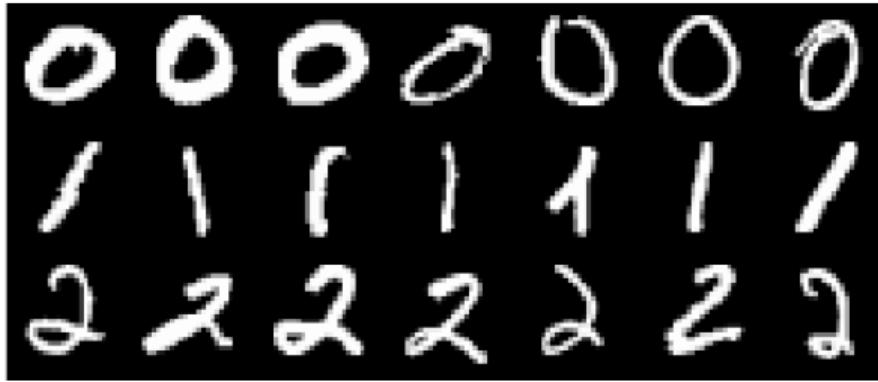
Mitesh M. Khapra

Department of Computer Science and Engineering
Indian Institute of Technology Madras

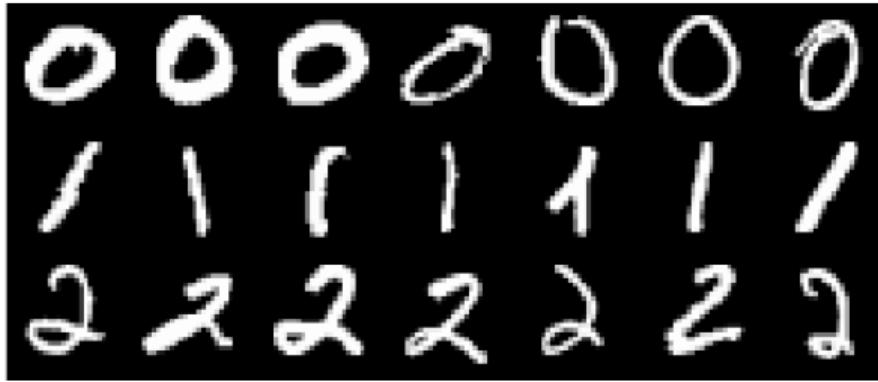
Module 23.1: Generative Adversarial Networks - The intuition



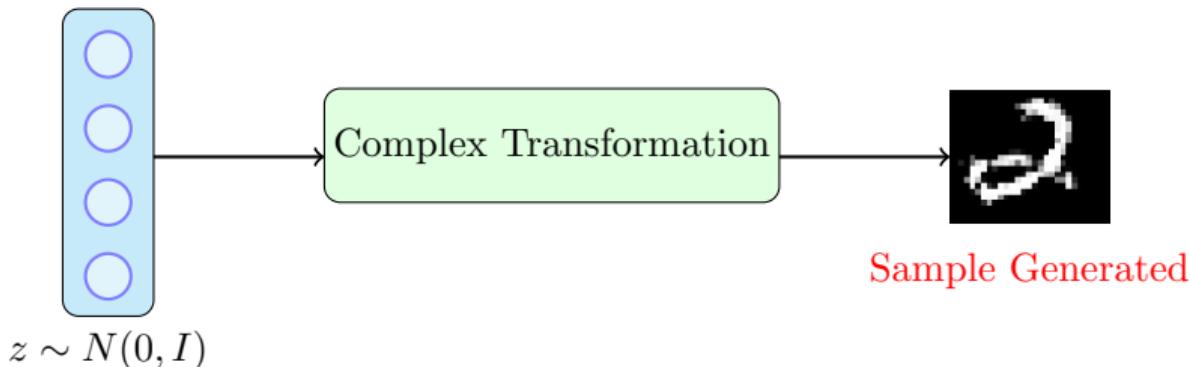
- As usual we are given some training data (say, MNIST images) which obviously comes from some underlying distribution



- 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?

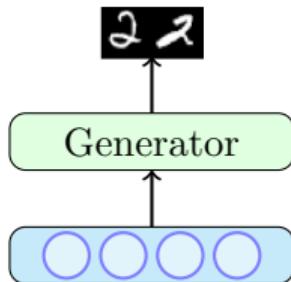
- What can we use for such a complex transformation? **A Neural Network**

- 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**

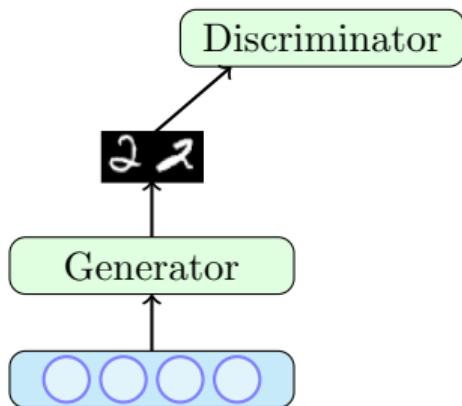
- What can we use for such a complex transformation? A Neural Network
- How do you train such a neural network? Using a two player game
- There are two players in the game:

- 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**



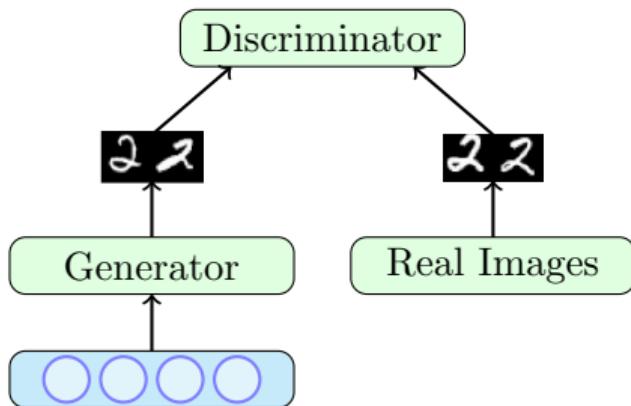
$$z \sim N(0, I)$$

- 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**

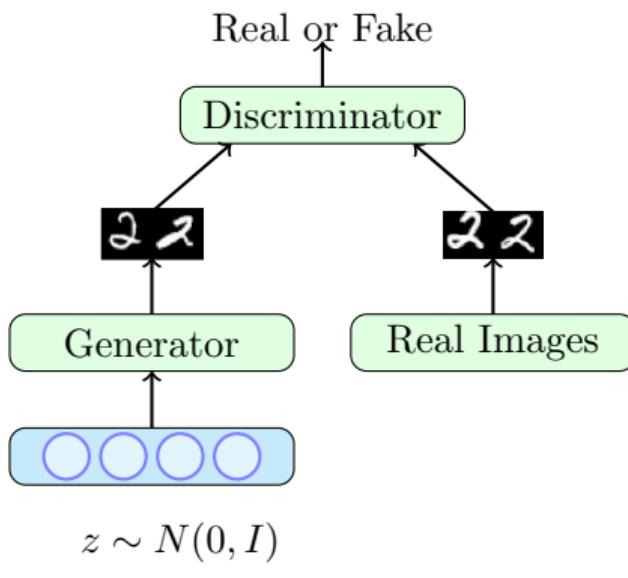


$$z \sim N(0, I)$$

- 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
 - 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

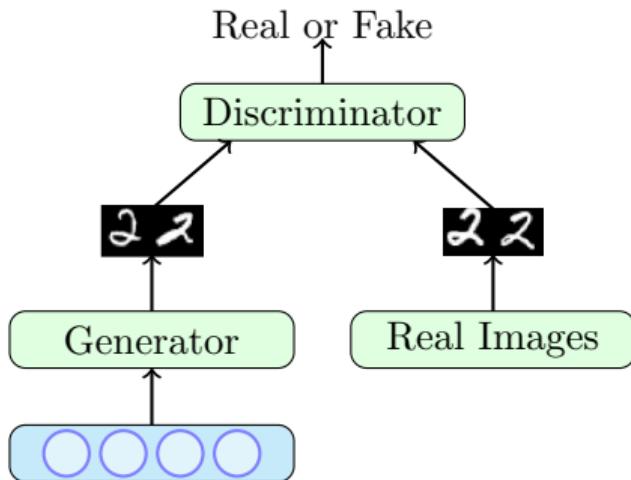


$$z \sim N(0, I)$$

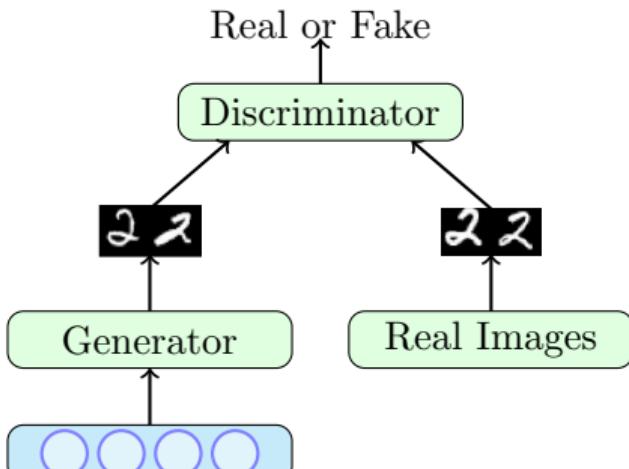


- What can we use for such a complex transformation? A Neural Network
- How do you train such a neural network? Using a two player game
- There are two players in the game: a generator and a discriminator
- The job of the generator is to produce images which look so natural that the discriminator thinks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

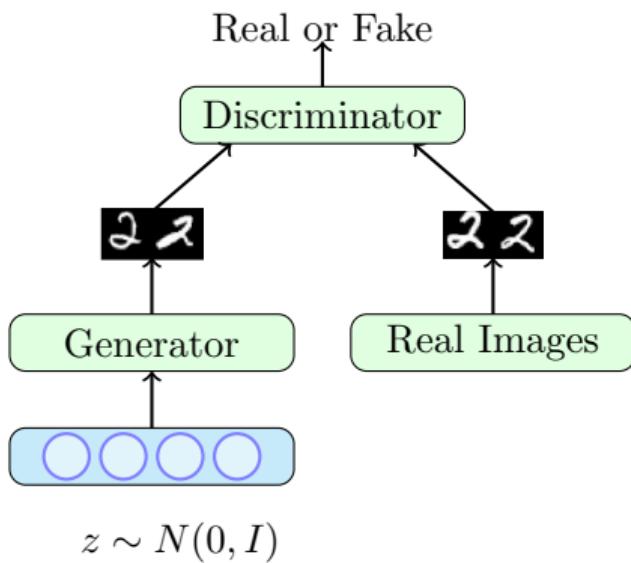
- So let's look at the full picture

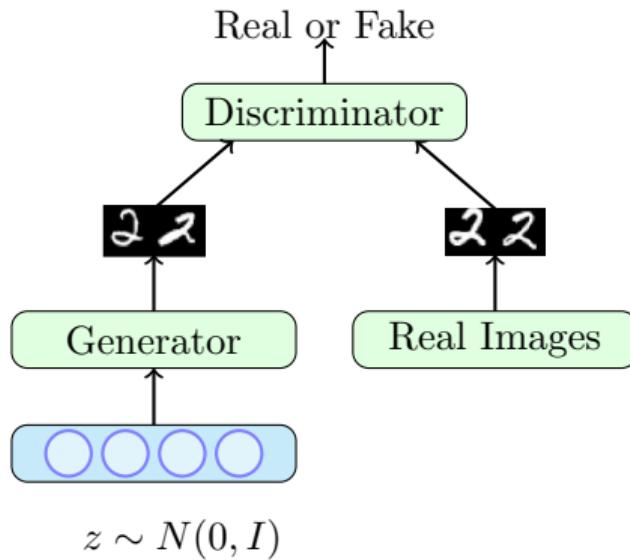


- So let's look at the full picture
- Let G_ϕ be the generator and D_θ be the discriminator (ϕ and θ are the parameters of G and D , respectively)



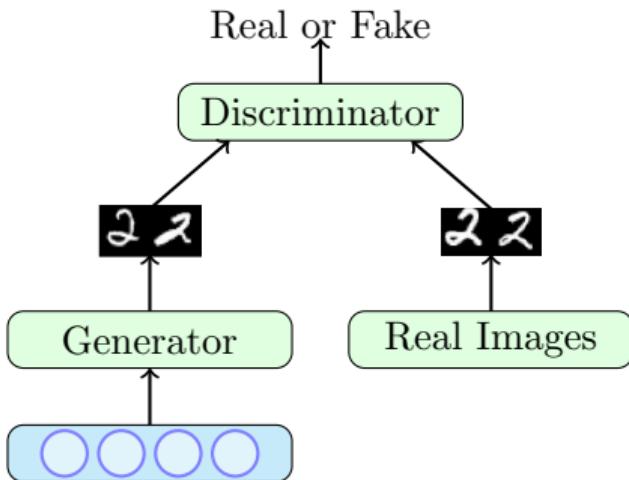
- So let's look at the full picture
 - Let G_ϕ be the generator and D_θ be the discriminator (ϕ and θ are the parameters of G and D , respectively)
 - We have a neural network based generator which takes as input a noise vector $z \sim N(0, I)$ and produces $G_\phi(z) = X$



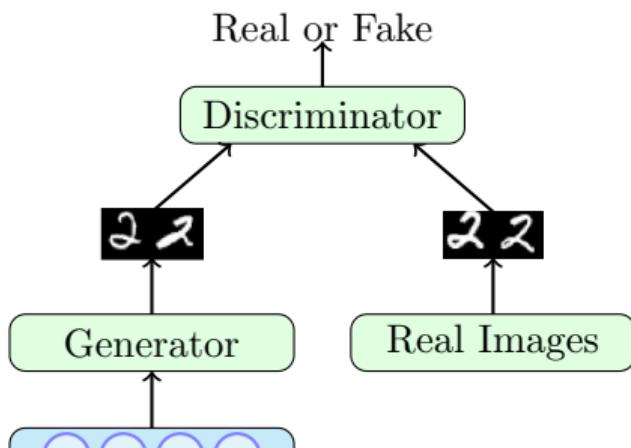


- So let's look at the full picture
- Let G_ϕ be the generator and D_θ be the discriminator (ϕ and θ are the parameters of G and D , respectively)
- We have a neural network based generator which takes as input a noise vector $z \sim N(0, I)$ and produces $G_\phi(z) = X$
- We have a neural network based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

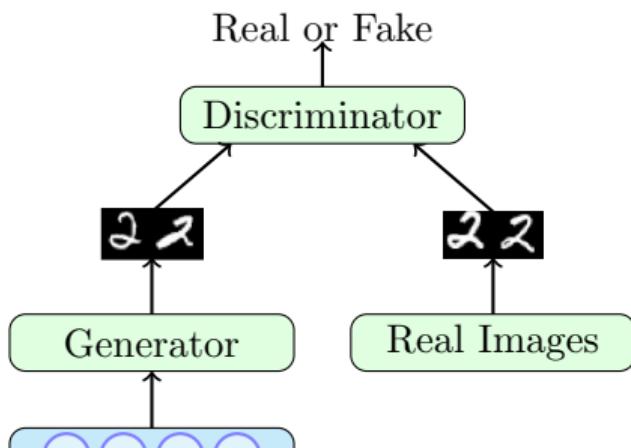
- What should be the objective function of the overall network?



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

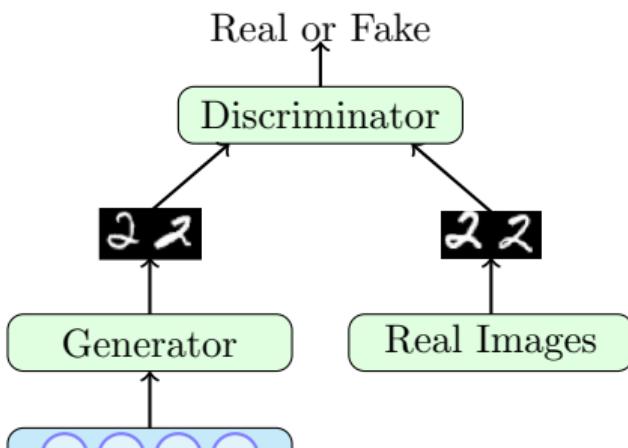


- What should be the objective function of the overall network?
- Let's look at the objective function of the generator first
- Given an image generated by the generator as $G_\phi(z)$ the discriminator assigns a score $D_\theta(G_\phi(z))$ to it

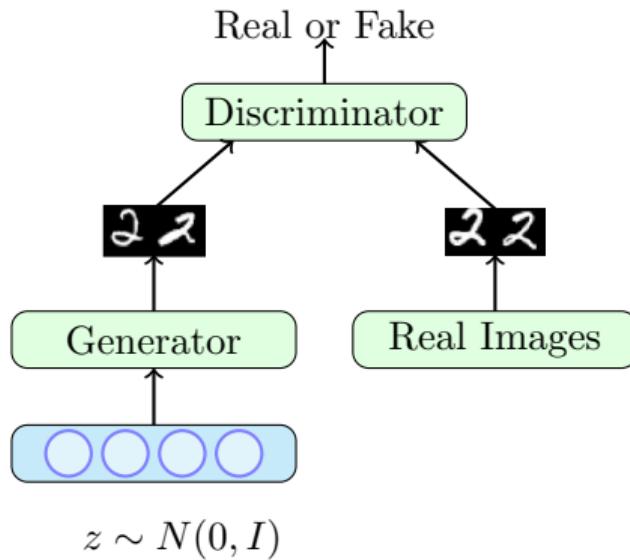


$$z \sim N(0, I)$$

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

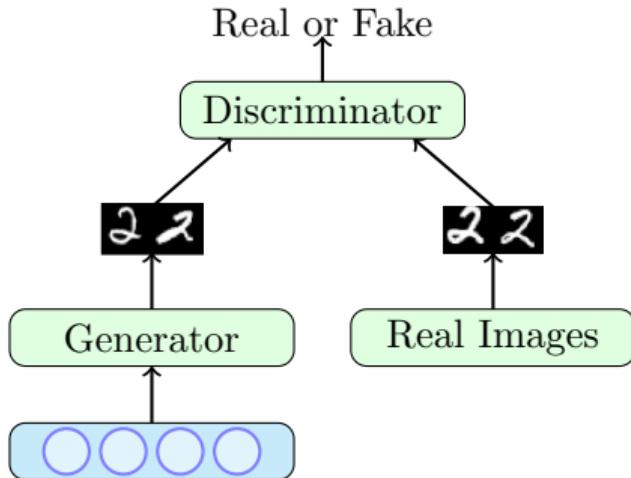


- Let's look at the objective function of the generator first
- Given an image generated by the generator as $G_\phi(z)$ the discriminator assigns a score $D_\theta(G_\phi(z))$ to it
- This score will be between 0 and 1 and will tell us the probability of the image being real or fake



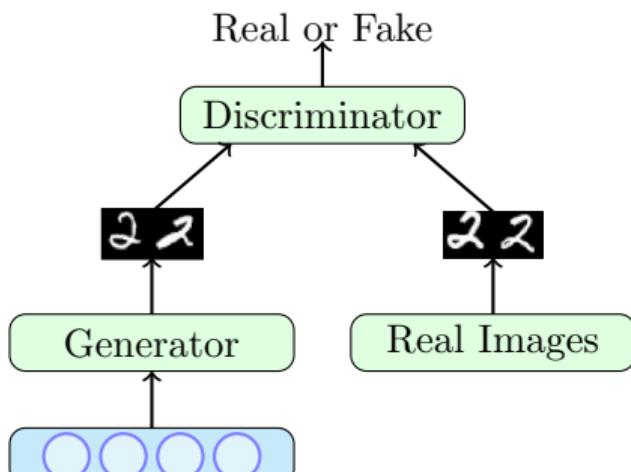
- What should be the objective function of the overall network?
- Let's look at the objective function of the generator first
- Given an image generated by the generator as $G_\phi(z)$ the discriminator assigns a score $D_\theta(G_\phi(z))$ to it
- This score will be between 0 and 1 and will tell us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

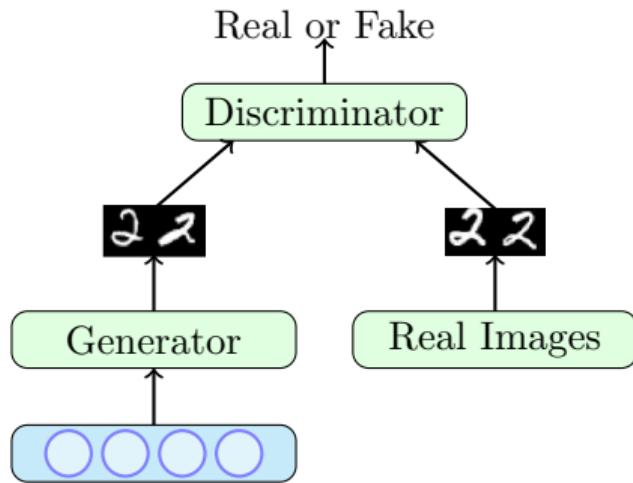


- This is just for a single z and the generator would like to do this for all possible values of z ,
- For example, if z was discrete and drawn from a uniform distribution (*i.e.*, $p(z) = \frac{1}{N} \forall z$) then the generator's objective function would be

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



$$z \sim N(0, I)$$



- 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

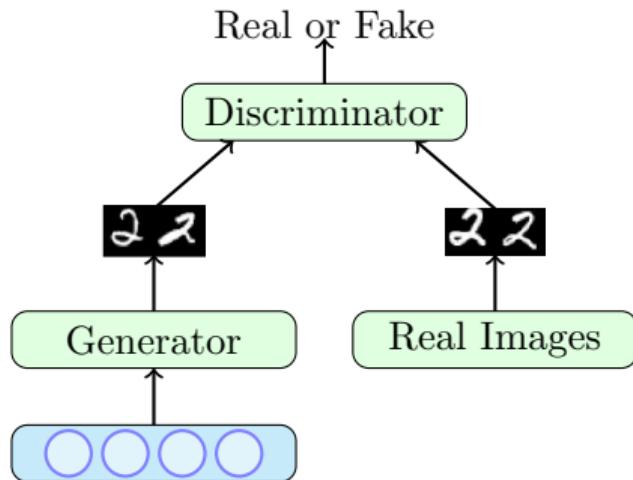
$$\min_{\phi} \sum_{i=1}^N \frac{1}{N} \log(1 - D_{\theta}(G_{\phi}(z)))$$

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

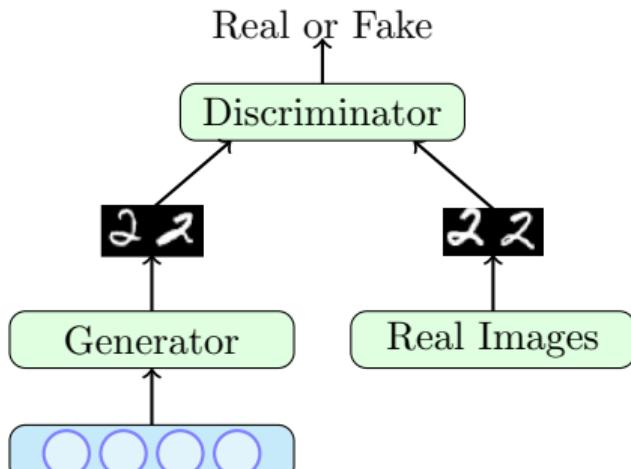
$$\min_{\phi} \int p(z) \log(1 - D_{\theta}(G_{\phi}(z)))$$

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

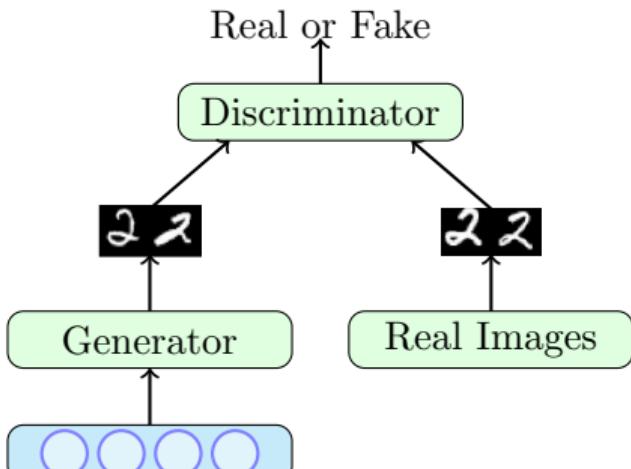
- Now let's look at the discriminator

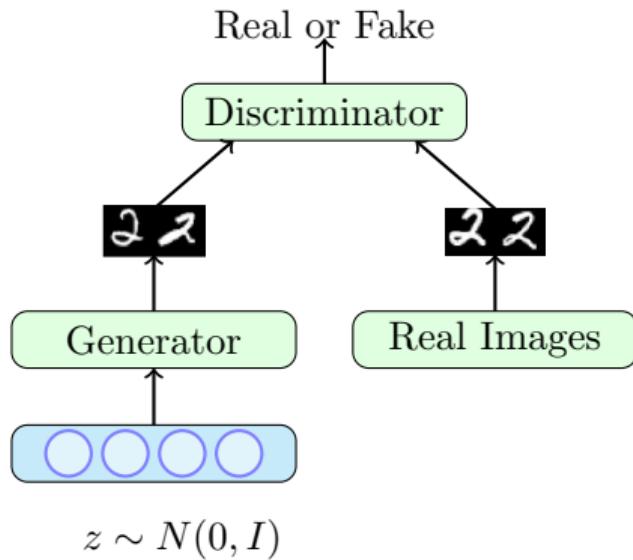


- Now let's look at the discriminator
- The task of the discriminator is to assign a high score to real images and a low score to fake images



- 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

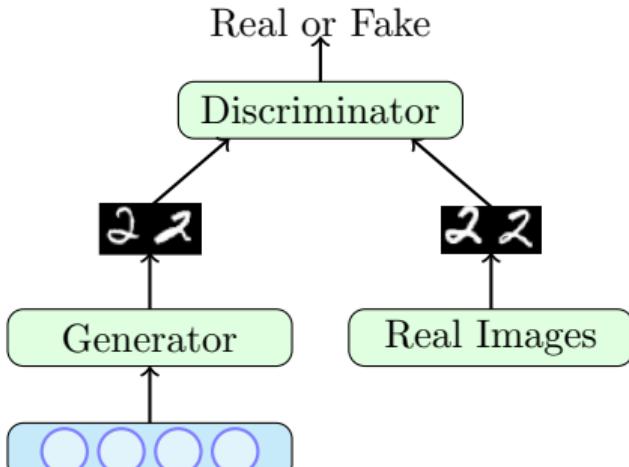




- Now let's look at the discriminator
- The task of the discriminator is to assign a high score to real images and a low score to fake images
- And it should do this for all possible real images and all possible fake images
- In other words, it should try to maximize the following objective function

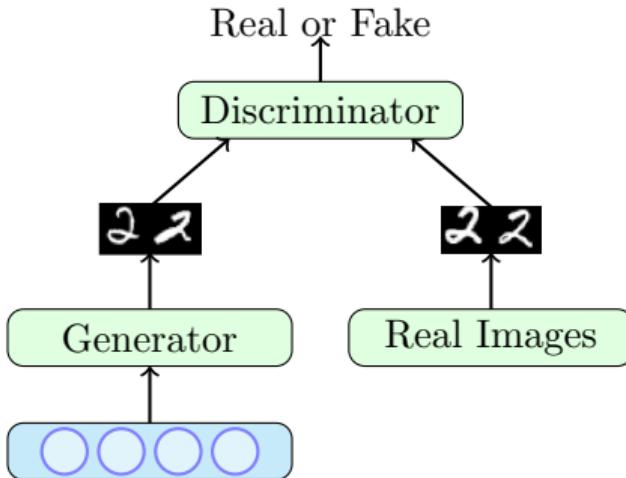
$$\max_{\theta} E_{x \sim p_{data}} [\log D_{\theta}(x)] + E_{z \sim p(z)} [\log(1 - D_{\theta}(G_{\phi}(z)))]$$

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



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

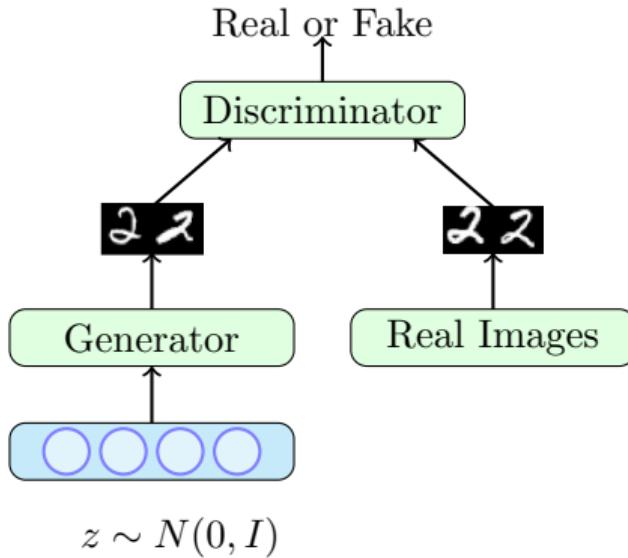
- If we put the objectives of the generator and discriminator together we get a minimax game



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

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

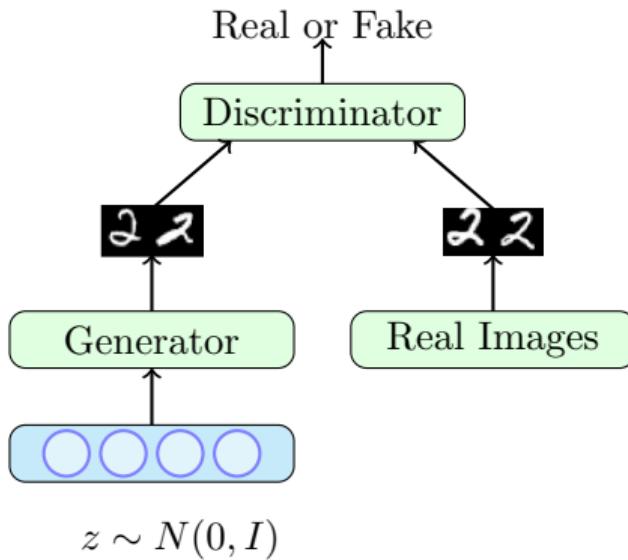
- If we put the objectives of the generator and discriminator together we get a minimax game



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

- 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)

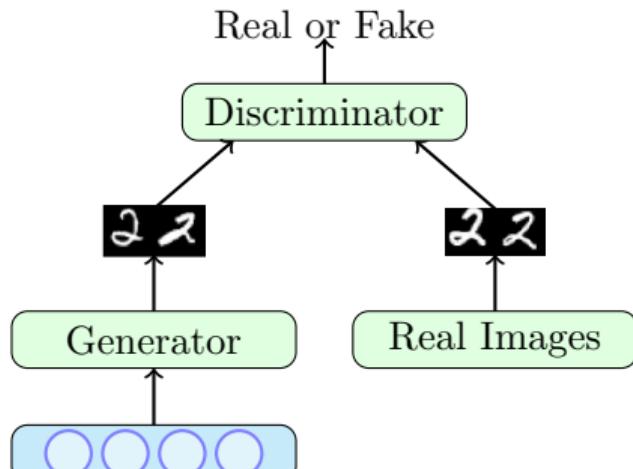
- If we put the objectives of the generator and discriminator together we get a minimax game



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

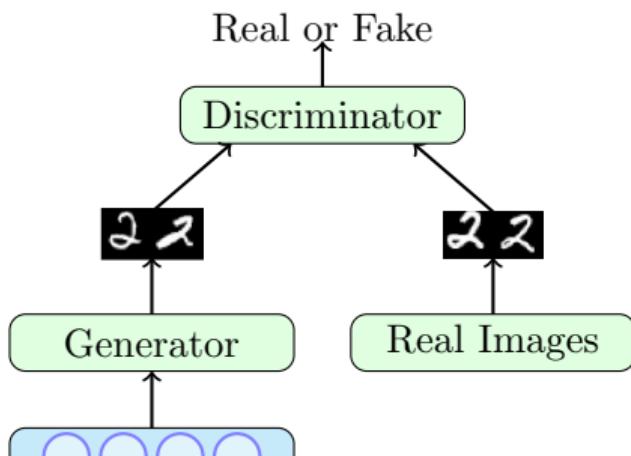
- The first term in the objective is only w.r.t. the parameters of the discriminator (θ)
- The second term in the objective is w.r.t. the parameters of the generator (ϕ) as well as the discriminator (θ)
- The discriminator wants to maximize the second term whereas the generator wants to minimize it (hence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

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



$$z \sim N(0, I)$$

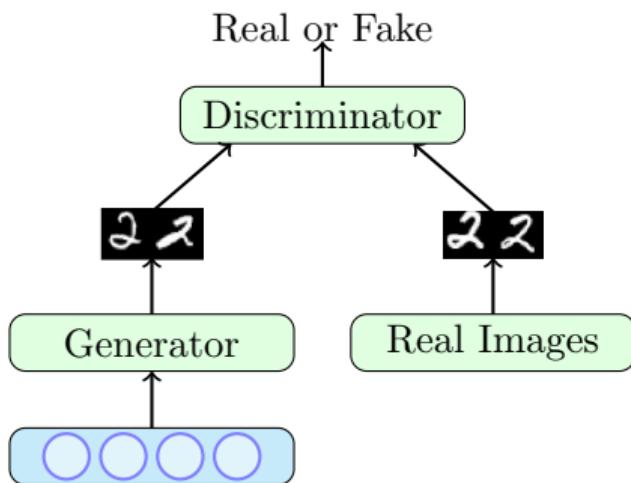
- So the overall training proceeds by alternating between these two step

- **Step 1:** Gradient Ascent on Discriminator

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

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

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



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

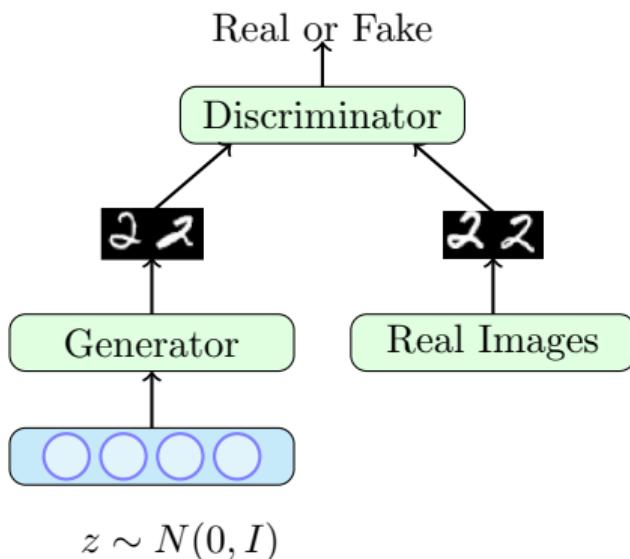
- Step 1:** Gradient Ascent on Discriminator

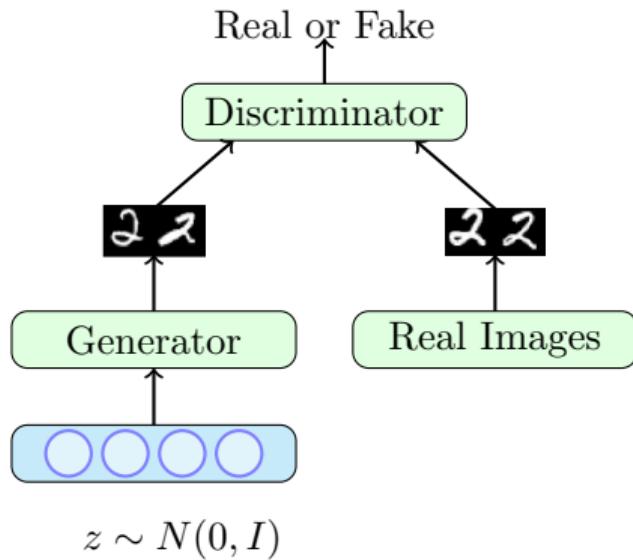
$$\max_{\theta} [\mathbb{E}_{x \sim p_{data}} \log D_{\theta}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta}(G_{\phi}(z)))]$$

- 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 steps

- Step 1:** Gradient Ascent on Discriminator

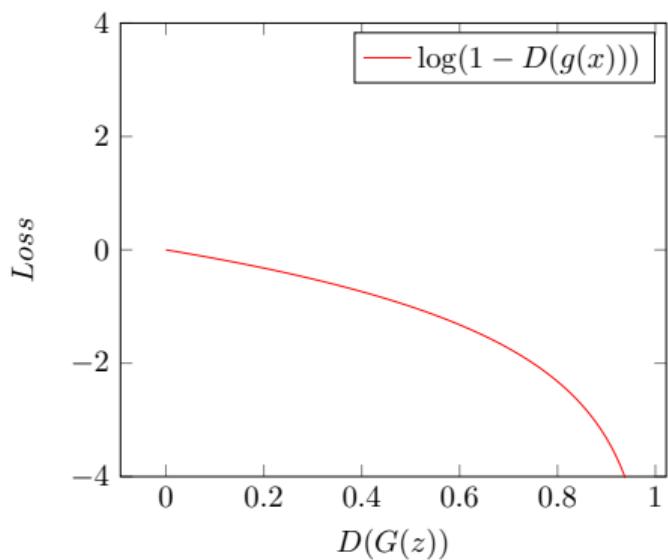
$$\max_{\theta} [\mathbb{E}_{x \sim p_{data}} \log D_{\theta}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta}(G_{\phi}(z)))]$$

- Step 2:** Gradient Descent on Generator

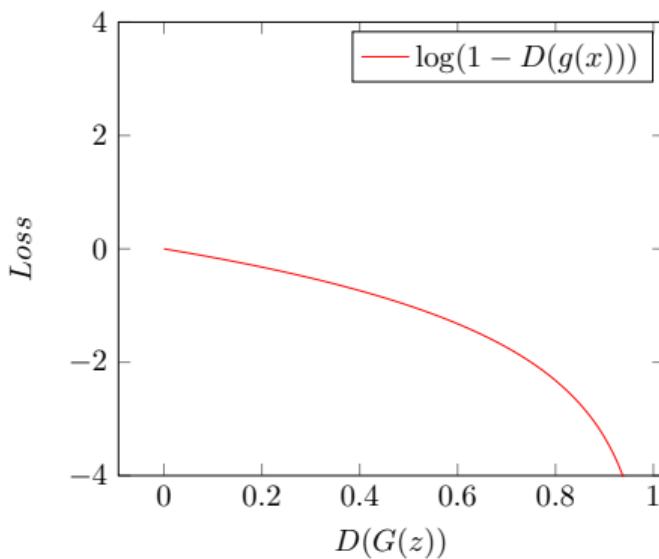
$$\min_{\phi} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta}(G_{\phi}(z)))$$

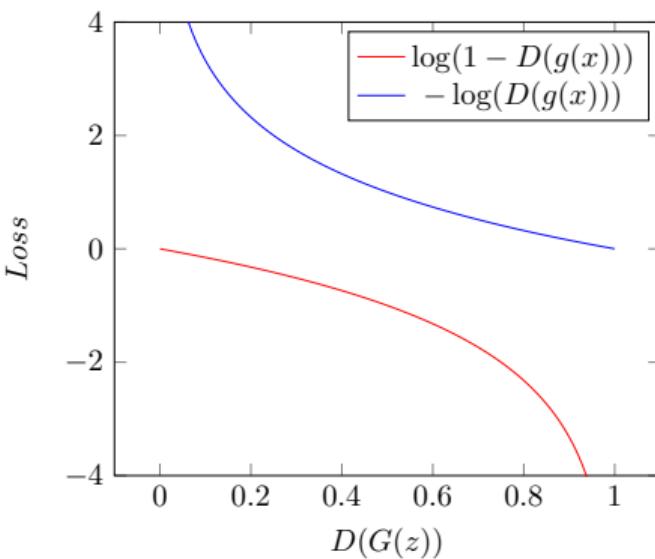
- In practice, the above generator objective does not work well and we use a slightly modified objective
- Let us see why

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

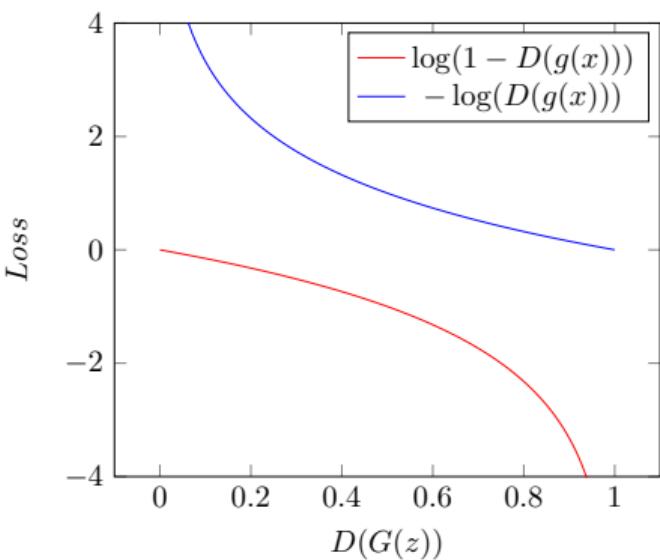


- When the sample is likely fake, we want to give a feedback to the generator (using gradients)
- However, in this region where $D(G(z))$ is close to 0, the curve of the loss function is very flat and the gradient would be close to 0





- 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

1: **procedure** GAN TRAINING

11: **end procedure**

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

```
1: procedure GAN TRAINING
2:   for number of training iterations do
10:  end for
11: end procedure
```

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

```
1: procedure GAN TRAINING
2:   for number of training iterations do
3:     for k steps do
7:       end for
10:      end for
11: end procedure
```

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

```
1: procedure GAN TRAINING
2:   for number of training iterations do
3:     for k steps do
4:       • Sample minibatch of  $m$  noise samples  $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$  from noise prior  $p_g(\mathbf{z})$ 
7:     end for
10:   end for
11: end procedure
```

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

```
1: procedure GAN TRAINING
2:   for number of training iterations do
3:     for k steps do
4:       • Sample minibatch of  $m$  noise samples  $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$  from noise prior  $p_g(\mathbf{z})$ 
5:       • Sample minibatch of  $m$  examples  $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$  from data generating distribution  $p_{data}(\mathbf{x})$ 
6:
7:     end for
8:
9:
10:   end for
11: end procedure
```

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

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

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

7:     end for
8:
9:
10:   end for
11: end procedure
```

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

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

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

7:     end for
8:     • Sample minibatch of  $m$  noise samples  $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$  from noise prior  $p_g(\mathbf{z})$ 
9:
10:    end for
11: end procedure
```

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

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

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

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

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

10:   end for
11: end procedure
```

Module 23.2: Generative Adversarial Networks - Architecture

- We will now look at one of the popular neural networks used for the generator and discriminator (Deep Convolutional GANs)

- We will now look at one of the popular neural networks used for the generator and discriminator (Deep Convolutional GANs)
- For discriminator, any CNN based classifier with 1 class (real) at the output can be used (e.g. VGG, ResNet, etc.)

- We will now look at one of the popular neural networks used for the generator and discriminator (Deep Convolutional GANs)
- For discriminator, any CNN based classifier with 1 class (real) at the output can be used (e.g. VGG, ResNet, etc.)

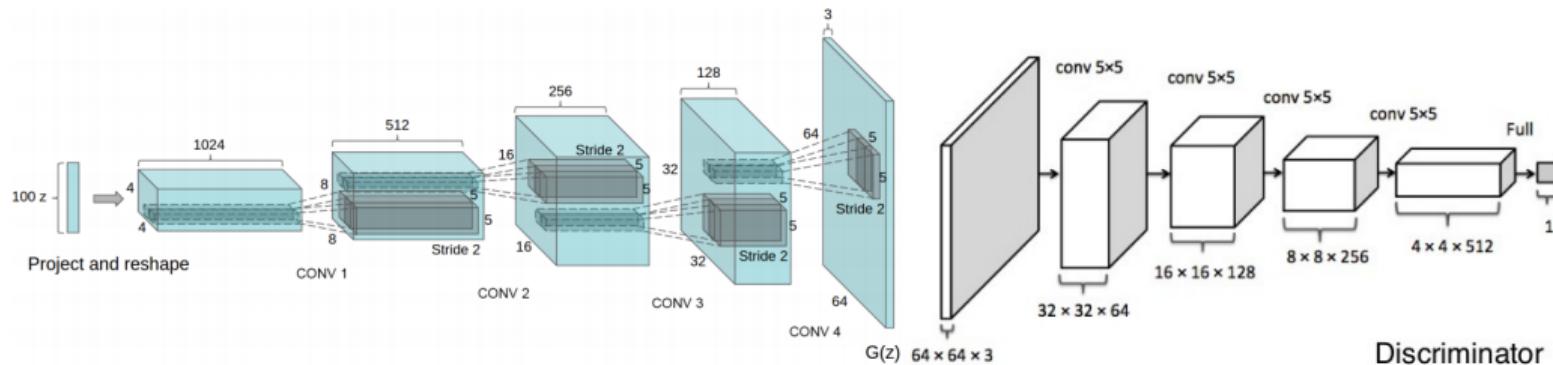


Figure: Generator (Redford et al 2015) (left) and discriminator (Yeh et al 2016) (right) used in DCGAN

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses tanh.

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses tanh.
- Use LeakyReLU activation in the discriminator for all layers

Module 23.3: Generative Adversarial Networks - The Math Behind it

- 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** $p_G = p_{data}$

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

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

- First let us look at the objective function again

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

- We will expand it to its integral form

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

- First let us look at the objective function again

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

- 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

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

- First let us look at the objective function again

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

- 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

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

- The above replacement follows from the *law of the unconscious statistician* ([click to link of wikipedia page](#))

- Okay, so our revised objective is given by

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

- Okay, so our revised objective is given by

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

- Given a generator G, we are interested in finding the optimum discriminator D which will maximize the above objective function

- Okay, so our revised objective is given by

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

- 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 $\forall x$

- Okay, so our revised objective is given by

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

- 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

- Okay, so our revised objective is given by

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

- 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

$$\frac{d}{d(D_{\theta}(x))} (p_{data}(x) \log D_{\theta}(x) + p_G(x) \log(1 - D_{\theta}(x))) = 0$$

- Okay, so our revised objective is given by

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

- 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

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

- Okay, so our revised objective is given by

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

- 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

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

- Okay, so our revised objective is given by

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

- 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

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

- Okay, so our revised objective is given by

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

- 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

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

- This means for any given generator

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

- 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 $p_G = p_{data}$ ”

- 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

- 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

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

- 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

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

$$V(G, D_G^*) = \int_x p_{data}(x) \log D(x) + p_G(x) \log (1 - D(x)) dx$$

- 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

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

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

- 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

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

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

- 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

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

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

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 $p_G = p_{data}$

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 $-\log 4$

- 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

- 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 $-\log 4$

- 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 $p_G = p_{data}$

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

$$C(G) = \int_x \left[p_{data}(x) \log \left(\frac{p_{data}(x)}{p_G(x) + p_{data}(x)} \right) + p_G(x) \log \left(1 - \frac{p_{data}(x)}{p_G(x) + p_{data}(x)} \right) \right] dx$$

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

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

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

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

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

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

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

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

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 $p_G = p_{data}$

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

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

- Okay, so we have

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

- We know that KL divergence is always ≥ 0

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

- Okay, so we have

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

- We know that KL divergence is always ≥ 0

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

- Hence the minimum possible value of $C(G)$ is $-\log 4$

- Okay, so we have

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

- 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)

- Okay, so we have

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

- 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 $p_G = p_{data}$

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
 $p_G = p_{data}$

- 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

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

- 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

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

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

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

- 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

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

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

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

- This will happen only when $p_G = p_{data}$ (you can prove this easily)

- 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

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

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

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

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

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

- 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

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

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

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

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

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

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

Module 23.4: Generative Adversarial Networks - Some Cool Stuff and Applications



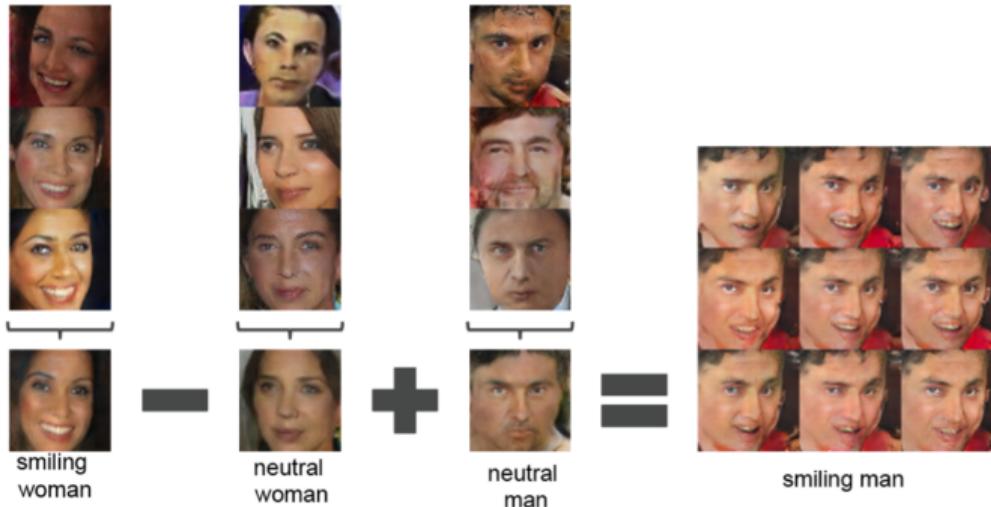
- In each row the first image was generated by the network by taking a vector z_1 as the input and the last images were generated by a vector z_2 as the input



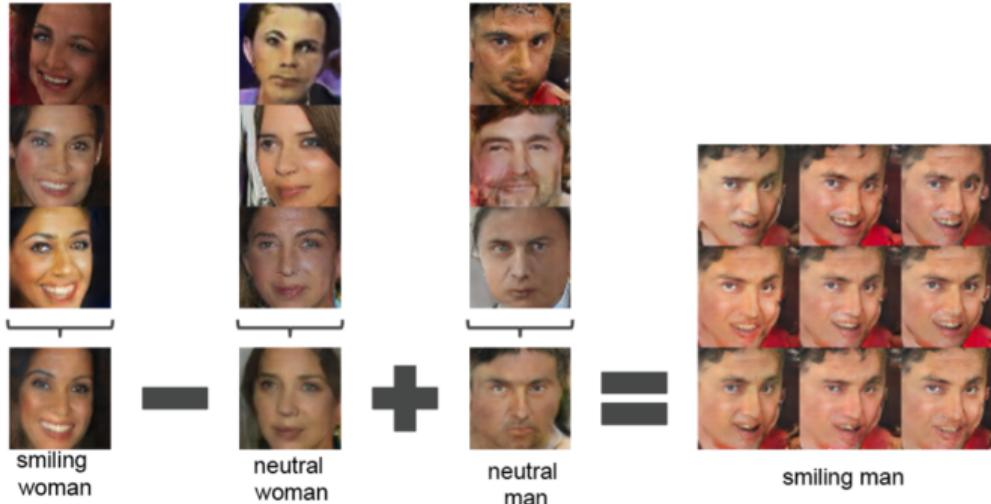
- In each row the first image was generated by the network by taking a vector z_1 as the input and the last images was generated by a vector z_2 as the input
- All intermediate images were generated by feeding z 's which were obtained by interpolating z_1 and z_2 ($z = \lambda z_1 + (1 - \lambda)z_2$)



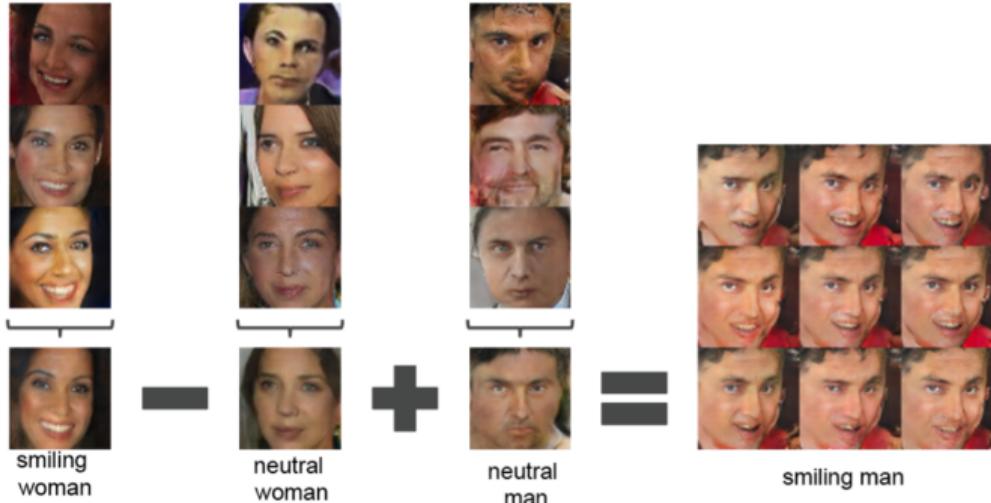
- In each row the first image was generated by the network by taking a vector z_1 as the input and the last images was generated by a vector z_2 as the input
- All intermediate images were generated by feeding z 's which were obtained by interpolating z_1 and z_2 ($z = \lambda z_1 + (1 - \lambda)z_2$)
- As we transition from z_1 to z_2 in the input space there is a corresponding smooth transition in the image space also



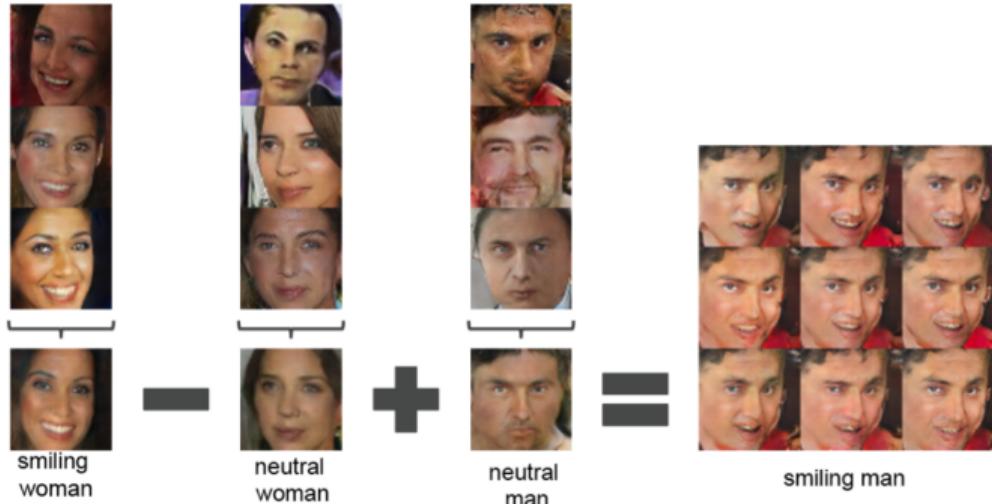
- The first 3 images in the first column were generated by feeding some z_{11}, z_{12}, z_{13} respectively as the input to the generator



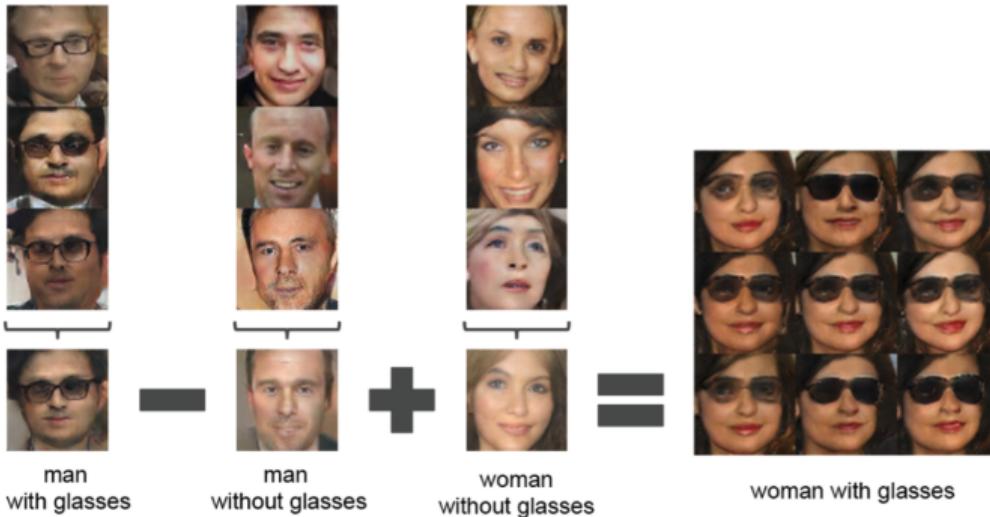
- The first 3 images in the first column were generated by feeding some z_{11}, z_{12}, z_{13} respectively as the input to the generator
- The fourth image was generated by taking an average of $z_1 = z_{11}, z_{12}, z_{13}$ and feeding it to the generator



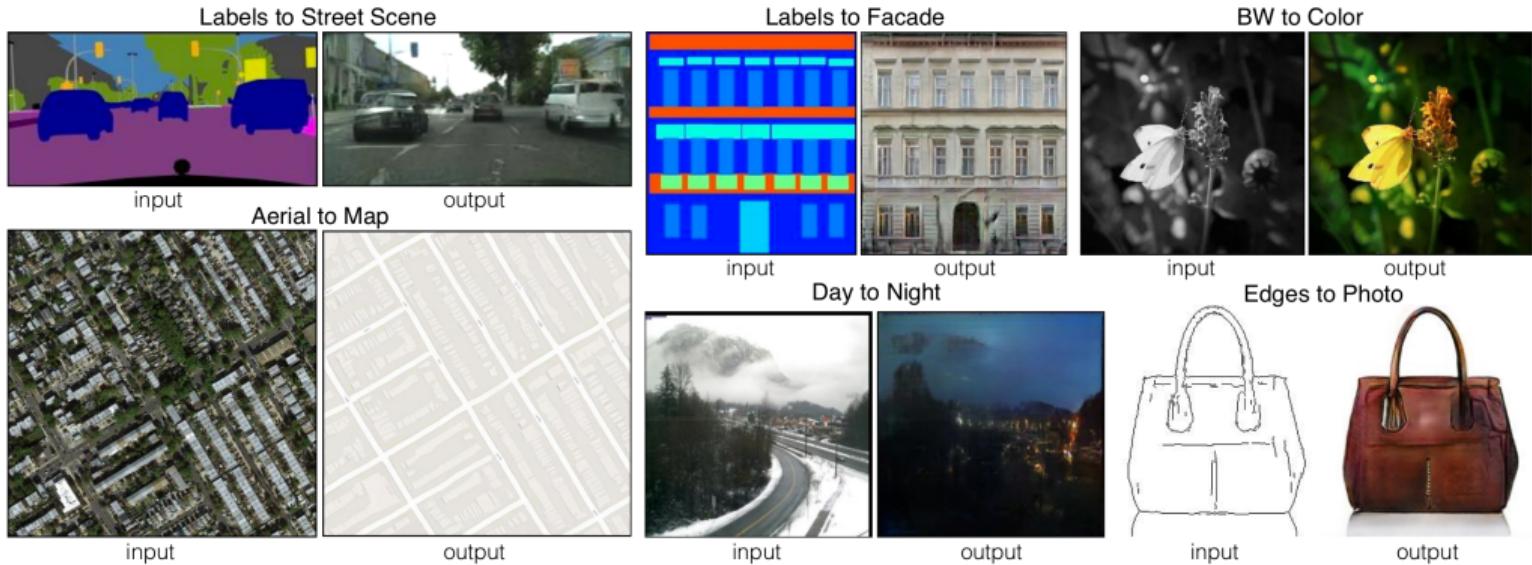
- The first 3 images in the first column were generated by feeding some z_{11}, z_{12}, z_{13} respectively as the input to the generator
- The fourth image was generated by taking an average of $z_1 = z_{11}, z_{12}, z_{13}$ and feeding it to the generator
- Similarly we obtain the average vectors z_2 and z_3 for the 2nd and 3rd columns



- The first 3 images in the first column were generated by feeding some z_{11}, z_{12}, z_{13} respectively as the input to the generator
- The fourth image was generated by taking an average of $z_1 = z_{11}, z_{12}, z_{13}$ and feeding it to the generator
- Similarly we obtain the average vectors z_2 and z_3 for the 2nd and 3rd columns
- If we do a simple vector arithmetic on these averaged vectors then we see the corresponding effect in the generated images



- The first 3 images in the first column were generated by feeding some z_{11}, z_{12}, z_{13} respectively as the input to the generator
- The fourth image was generated by taking an average of $z_1 = z_{11}, z_{12}, z_{13}$ and feeding it to the generator
- Similarly we obtain the average vectors z_2 and z_3 for the 2nd and 3rd columns
- If we do a simple vector arithmetic on these averaged vectors then we see the corresponding effect in the generated images



[Phillip Isola](#), [Jun-Yan Zhu](#), [Tinghui Zhou](#), [Alexei A. Efros](#), Image-to-Image Translation with Conditional Adversarial Networks, CVPR, 2017.

Module 23.5: Bringing it all together (the deep generative summary)

	RBM	VAE	AR models	GANs
Abstraction	Yes	Yes	No	No

Table: Comparison of Generative Models

	RBM	VAE	AR models	GANs
Abstraction	Yes	Yes	No	No
Generation	Yes	Yes	Yes	Yes

Table: Comparison of Generative Models

	RBM	VAE	AR models	GANs
Abstraction	Yes	Yes	No	No
Generation	Yes	Yes	Yes	Yes
Compute $P(X)$	Intractable	Intractable	Tractable	No

Table: Comparison of Generative Models

	RBM	VAE	AR models	GANs
Abstraction	Yes	Yes	No	No
Generation	Yes	Yes	Yes	Yes
Compute $P(X)$	Intractable	Intractable	Tractable	No
Sampling	MCMC	Fast	Slow	Fast

Table: Comparison of Generative Models

	RBM	VAE	AR models	GANs
Abstraction	Yes	Yes	No	No
Generation	Yes	Yes	Yes	Yes
Compute $P(X)$	Intractable	Intractable	Tractable	No
Sampling	MCMC	Fast	Slow	Fast
Type of GM	Undirected	Directed	Directed	Directed

Table: Comparison of Generative Models

	RBM	VAE	AR models	GANs
Abstraction	Yes	Yes	No	No
Generation	Yes	Yes	Yes	Yes
Compute $P(X)$	Intractable	Intractable	Tractable	No
Sampling	MCMC	Fast	Slow	Fast
Type of GM	Undirected	Directed	Directed	Directed
Loss	KL-divergence	KL-divergence	KL-divergence	JSD

Table: Comparison of Generative Models

	RBM	VAE	AR models	GANs
Abstraction	Yes	Yes	No	No
Generation	Yes	Yes	Yes	Yes
Compute $P(X)$	Intractable	Intractable	Tractable	No
Sampling	MCMC	Fast	Slow	Fast
Type of GM	Undirected	Directed	Directed	Directed
Loss	KL-divergence	KL-divergence	KL-divergence	JSD
Assumptions	X independent given z	X independent given z	None	N.A.

Table: Comparison of Generative Models

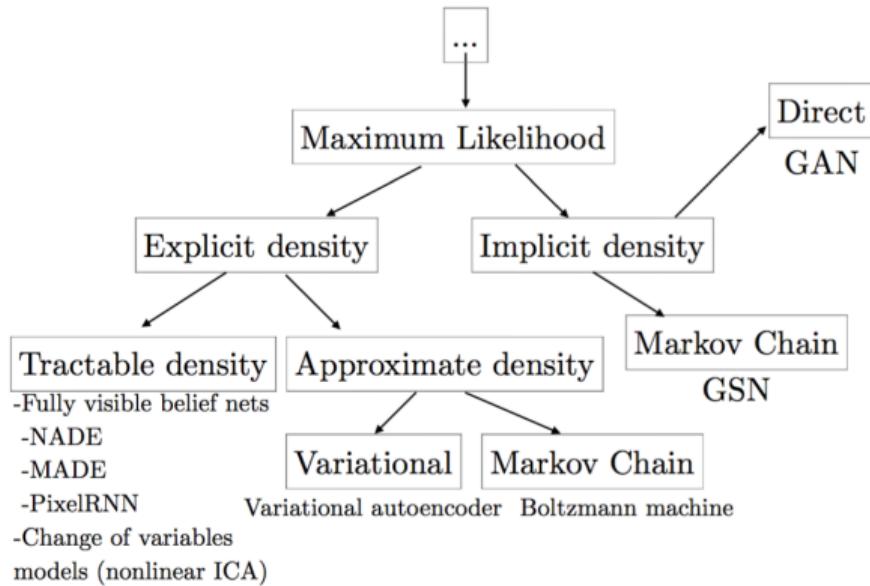
	RBM	VAE	AR models	GANs
Abstraction	Yes	Yes	No	No
Generation	Yes	Yes	Yes	Yes
Compute $P(X)$	Intractable	Intractable	Tractable	No
Sampling	MCMC	Fast	Slow	Fast
Type of GM	Undirected	Directed	Directed	Directed
Loss	KL-divergence	KL-divergence	KL-divergence	JSD
Assumptions	X independent given z	X independent given z	None	N.A.
Samples	Bad	Ok	Good	Good (best)

Table: Comparison of Generative Models

	RBM	VAE	AR models	GANs
Abstraction	Yes	Yes	No	No
Generation	Yes	Yes	Yes	Yes
Compute $P(X)$	Intractable	Intractable	Tractable	No
Sampling	MCMC	Fast	Slow	Fast
Type of GM	Undirected	Directed	Directed	Directed
Loss	KL-divergence	KL-divergence	KL-divergence	JSD
Assumptions	X independent given z	X independent given z	None	N.A.
Samples	Bad	Ok	Good	Good (best)

Table: Comparison of Generative Models

Recent works look at combining these methods: e.g. Adversarial Autoencoders (Makhzani 2015), PixelVAE (Gulrajani 2016) and PixelGAN Autoencoders (Makhzani 2017)



Source: Ian Goodfellow, NIPS 2016 Tutorial: Generative Adversarial Networks