Introduction to Deep Learning

Generative Adversarial Networks

Andres Mendez-Vazquez

June 25, 2025

Outline

Outline

Generative Models

We have seen how probability models can be used to generate samples

- Basically

$$data \sim p_{data}\left(x|\Theta\right)$$

Generative Models

We have seen how probability models can be used to generate samples

- Basically

$$data \sim p_{data}\left(x|\Theta\right)$$

Thus, we use a function $p_{model}\left(x|\Theta\right)$

- And a loss function for minimization

$$\min \mathcal{L}\left(p_{data}, p_{model}\right)$$

Generative Models

We have seen how probability models can be used to generate samples

- Basically

$$data \sim p_{data}\left(x|\Theta\right)$$

Thus, we use a function $p_{model}\left(x|\Theta\right)$

- And a loss function for minimization

$$\min \mathcal{L}\left(p_{data}, p_{model}\right)$$

Such, we can sample from the model

$$data \sim p_{model}\left(x|\Theta\right)$$

Why the generative model?

Realistic samples generation and handling of missing data.

- Known as data augmentation

Why the generative model?

Realistic samples generation and handling of missing data.

- Known as data augmentation

Address the density estimation problem in unsupervised learning

- Yes unsupervised learning depends a lot on knn properties

Why the generative model?

Realistic samples generation and handling of missing data.

- Known as data augmentation

Address the density estimation problem in unsupervised learning

- Yes unsupervised learning depends a lot on knn properties

It solves the problem of generating new data

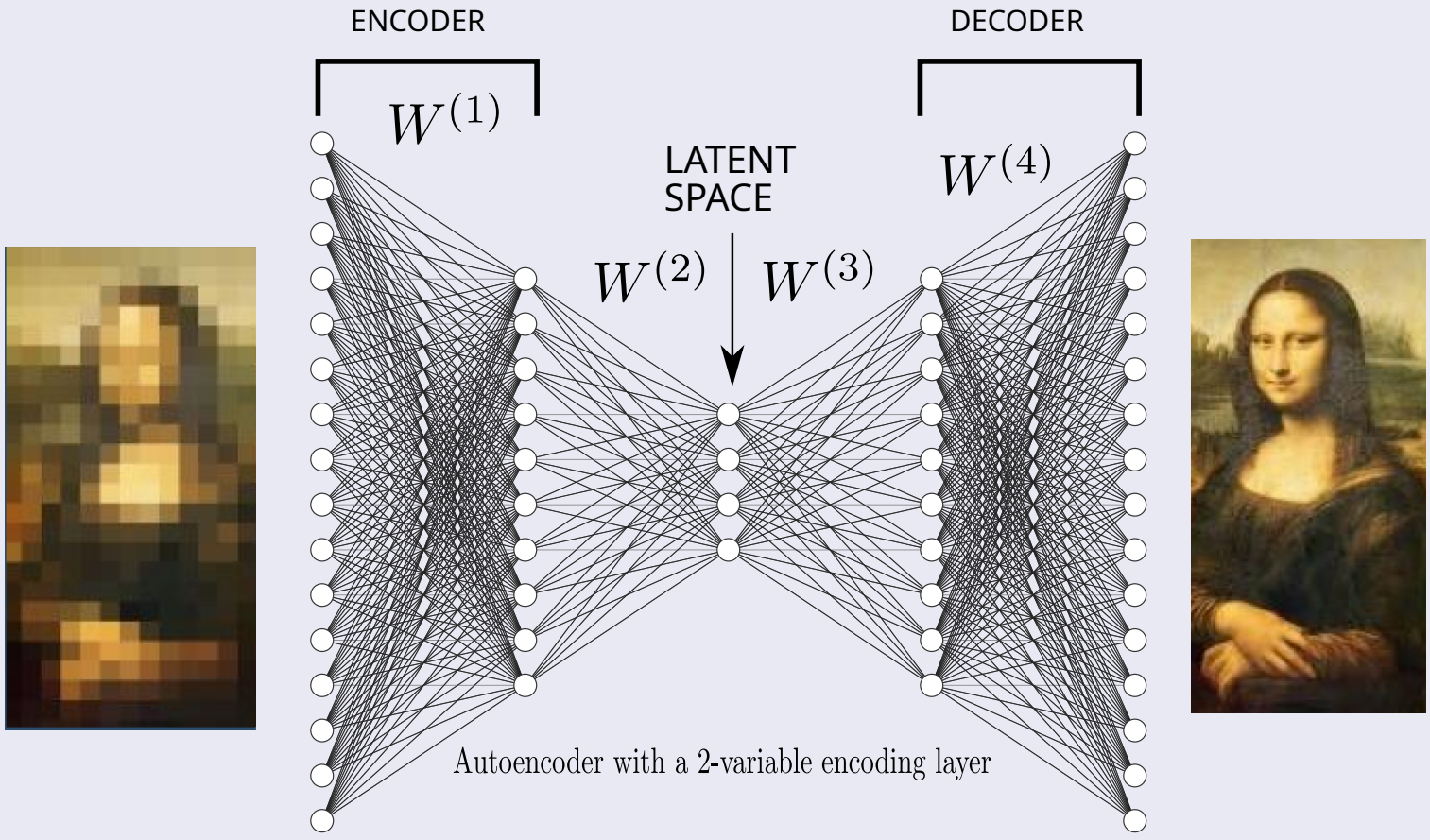
- Generate new data without human intervention

Outline

Remember Autoencoders

Do you remember?

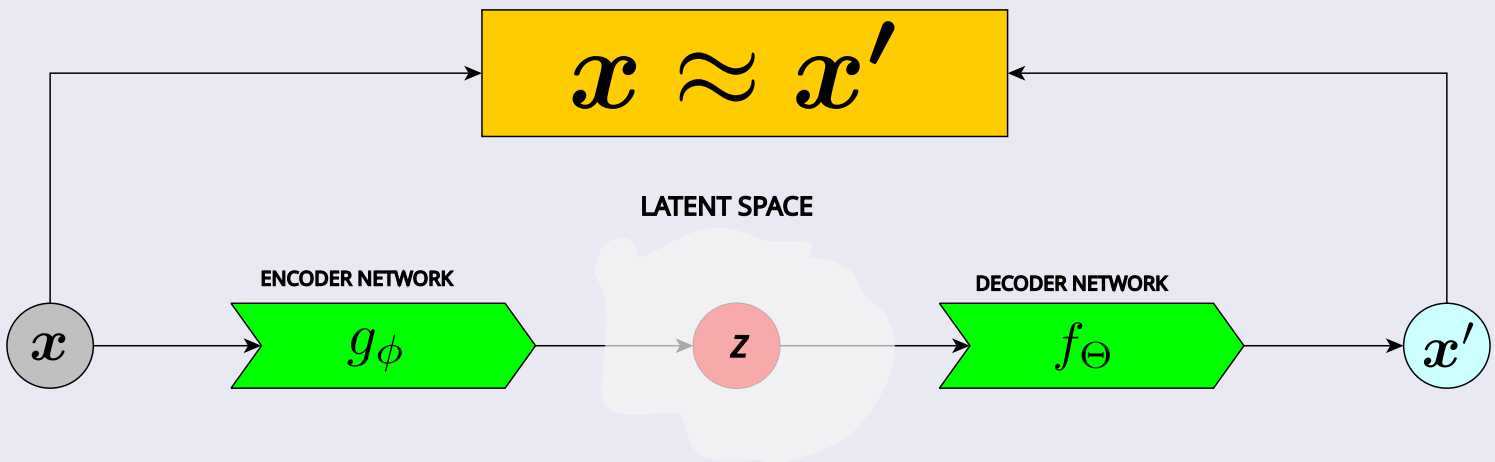
- "An autoencoder is a specific type of a neural network, which is mainly designed to encode the input into a compressed and meaningful representation, and then decode it back such that the reconstructed input is similar as possible to the original one."



Here, we have

With the following loss function

$$\mathcal{L}_{AE} = \frac{1}{n}\sum_{i=1}^{n} \|x - f_\Theta\left(g_\phi\left(x\right)\right)\|^2$$



Problem with this model

Samples, although reduced, are coming from a complex distribution

$$z \sim P_{complex}\left(z|\Theta\right)$$

Problem with this model

Samples, although reduced, are coming from a complex distribution

$$z \sim P_{complex}\left(z|\Theta\right)$$

Thus, we need as Markov Chain Monte Carlo Methods [1]

- In the case of the Cumulative Distribution Probability of $P_{complex}\left(z|\Theta\right)$ is not invertible or too complex
  - Which is almost always the case!!!

Problem with this model

Samples, although reduced, are coming from a complex distribution

$$z \sim P_{complex}\left(z|\Theta\right)$$

Thus, we need as Markov Chain Monte Carlo Methods [1]

- In the case of the Cumulative Distribution Probability of $P_{complex}\left(z|\Theta\right)$ is not invertible or too complex
  - Which is almost always the case!!!

Thus, Generative Adversarial Networks

- A escape way beyond Markov Chains [1]

Outline

This comes from the following paper

An notable work

- I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in Advances in neural information processing systems, pp. 2672–2680, 2014.

This comes from the following paper

An notable work

- I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in Advances in neural information processing systems, pp. 2672–2680, 2014.

A expansion beyond Monte Carlo

- Given than GAN's do not depend on them

In addition

We are grateful to the notes by

- Scott Rome
  - "An Annotated Proof of Generative Adversarial Networks with Implementation Notes"
  - https://srome.github.io/An-Annotated-Proof-of-Generative-Adversarial-Networks-with-Implementation-Notes/

Generative Adversarial Networks (GAN)

Something Notable

- Generative Adversarial Networks (GAN) are an unsupervised Deep Learning Method using a min-max game between two-players

Generative Adversarial Networks (GAN)

Something Notable

- Generative Adversarial Networks (GAN) are an unsupervised Deep Learning Method using a min-max game between two-players

One player is called the generator network (G)

- At the initial work it was a MLP given that its inpiration was Variational Autoencoders

Generative Adversarial Networks (GAN)

Something Notable

- Generative Adversarial Networks (GAN) are an unsupervised Deep Learning Method using a min-max game between two-players

One player is called the generator network (G)

- At the initial work it was a MLP given that its inpiration was Variational Autoencoders

The other is called the discriminator network (D)

- Also a MLP given that its inpiration was Variational Autoencoders

The Game

Something Notable

- The first player tries to generate a sample able to fool the discriminator player

The Game

Something Notable

- The first player tries to generate a sample able to fool the discriminator player

This is done by using a random latent vector z

- The Generator works by trying to generate better fake results

The Game

Something Notable

- The first player tries to generate a sample able to fool the discriminator player

This is done by using a random latent vector z

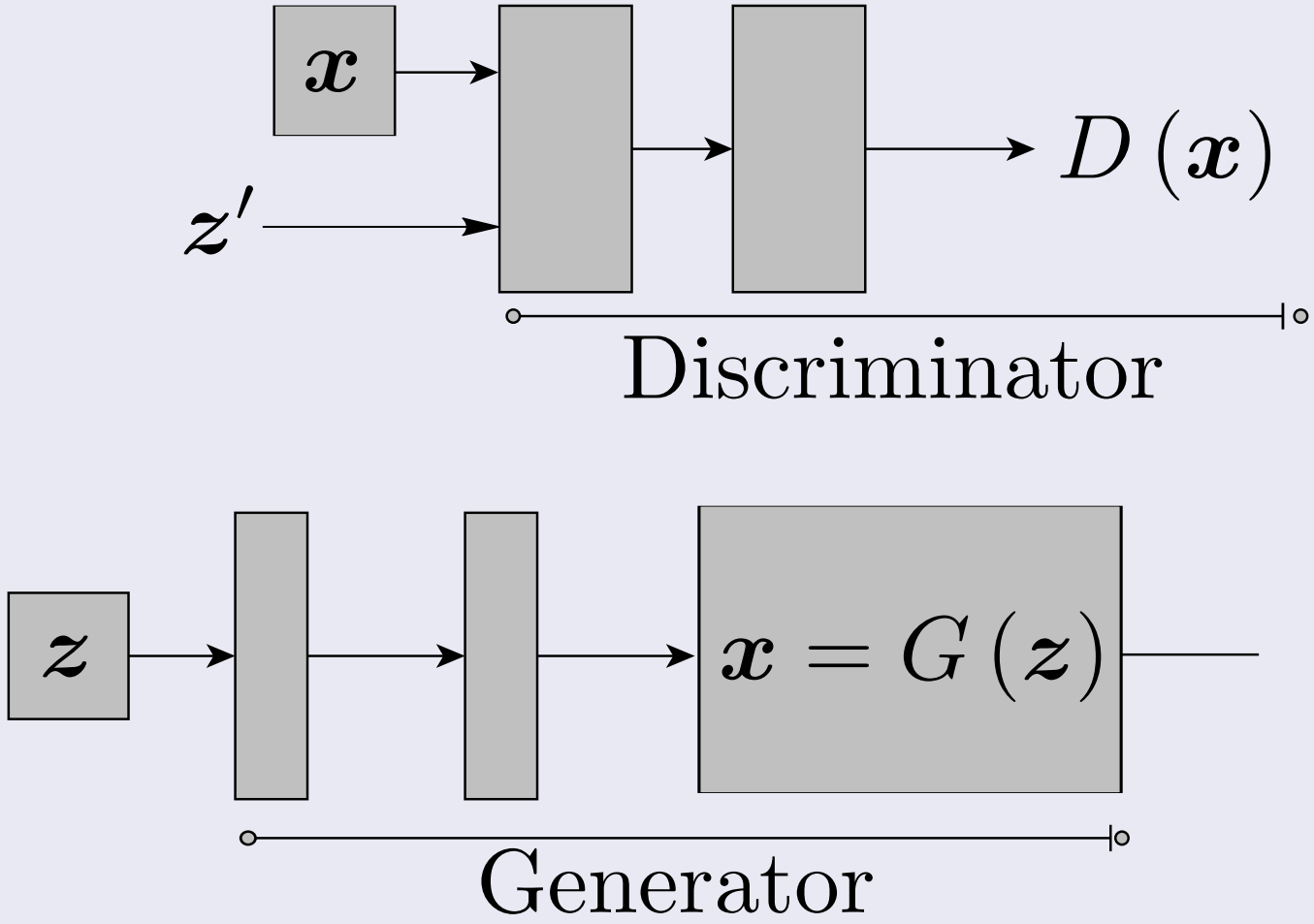
- The Generator works by trying to generate better fake results

The Discriminator tries to be better at finding

- if you have a real sample or not

Graphically

We have the following Basic Model



Here

There is a need to join both functions

- So, we can use the idea of Backpropagation to obtain the desired minimization.

Here

There is a need to join both functions

- So, we can use the idea of Backpropagation to obtain the desired minimization.

How can we do this?

- We can define a sensible learning criterion when the dataset is not linearly separable

Here

There is a need to join both functions

- So, we can use the idea of Backpropagation to obtain the desired minimization.

How can we do this?

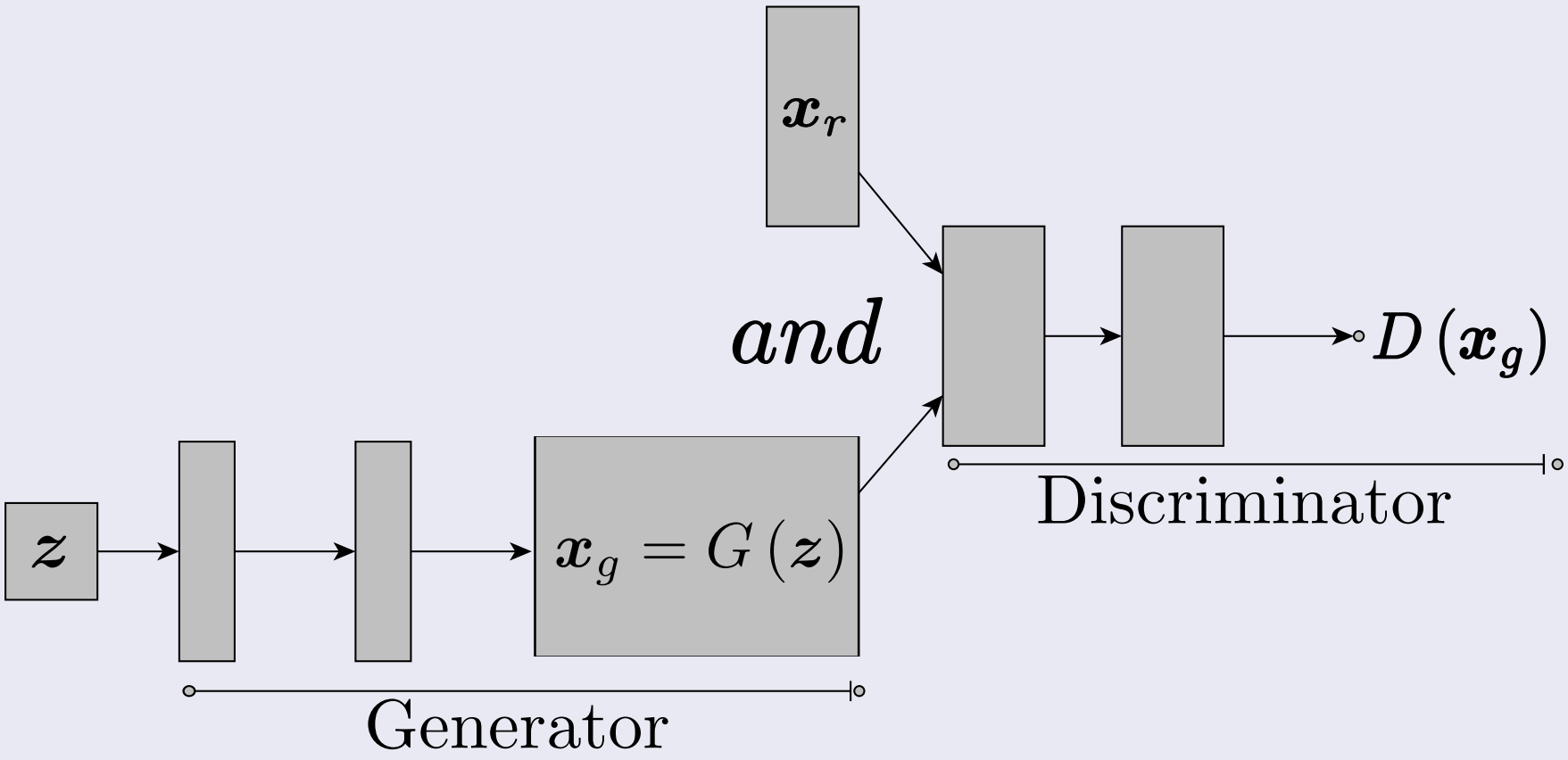
- We can define a sensible learning criterion when the dataset is not linearly separable

For this, we can use the **logistic cross-entropy loss** (We will explain more about this later)

$$\mathcal{L}_{LCE}(z,t) = L_{CE}(\sigma(z),t) = t\log(\sigma(z)) + (1-t)\log(1-\sigma(z))$$

Therefore, we have

The following architecture use this idea



Thus, we need to express this mathematically

Basically, we have [2]

$$E_{x \sim p_{data}(x)} \left[\log \left(D \left(x \right) \right) \right]$$

Thus, we need to express this mathematically

Basically, we have [2]

$$E_{x \sim p_{data}(x)} \left[\log \left(D \left(x \right) \right) \right]$$

Something Notable

- This term comes from the "positive class" of the log-loss function.

Thus, we need to express this mathematically

Basically, we have [2]

$$E_{x \sim p_{data}(x)} \left[\log \left(D \left(x \right) \right) \right]$$

Something Notable

- This term comes from the "positive class" of the log-loss function.

Properties

- Maximizing this term corresponds to D being able to precisely predict $D(x) = 1$ when $x \sim p_{data} \left(x \right)$

Now for the generator

The next term has to do with the Generator G tricking the discriminator.

- In particular, the term comes from the "negative class" of the log-loss function

$$E_{z \sim p_z(z)} \left[\log \left(1 - D \left(G \left(z \right) \right) \right) \right]$$

Now for the generator

The next term has to do with the Generator G tricking the discriminator.

- In particular, the term comes from the "negative class" of the log-loss function

$$E_{z \sim p_z(z)} \left[\log \left(1 - D \left(G \left(z \right) \right) \right) \right]$$

If this value is maximized because $\log(x) < 0$ when $x < 1$

- Meaning that $D(G(z)) \approx 0$ and G is NOT tricking D.

The Discriminator Goal

To combine these two concepts, the Discriminator's goal is to maximize

$$\max_D \left\{ E_{x \sim p_{data}(x)} \left[\log \left(D \left(x \right) \right) \right] + E_{z \sim p_z(z)} \left[\log \left(1 - D \left(G \left(z \right) \right) \right) \right] \right\}$$

The Discriminator Goal

To combine these two concepts, the Discriminator's goal is to maximize

$$\max_D \left\{ E_{x \sim p_{data}(x)} \left[\log \left(D \left(x \right) \right) \right] + E_{z \sim p_z(z)} \left[\log \left(1 - D \left(G \left(z \right) \right) \right) \right] \right\}$$

Given G

- which means that D properly identifies real and fake data points

The Discriminator Goal

To combine these two concepts, the Discriminator's goal is to maximize

$$\max_D \left\{ E_{x \sim p_{data}(x)} \left[\log \left(D \left(x \right) \right) \right] + E_{z \sim p_z(z)} \left[\log \left(1 - D \left(G \left(z \right) \right) \right) \right] \right\}$$

Given G

- which means that D properly identifies real and fake data points

Thus

- The optimal Discriminator in the sense of the above equation for a given G will be denoted D_G^*.

Meaning of this equation

Discriminant Training

- We have two parts the fake images and the real images, thus we can use $\log\left(D\left(x\right)\right)$ for the real images and the $\log\left(1-D\left(G\left(z\right)\right)\right)$ for the fake images

Meaning of this equation

Discriminant Training

- We have two parts the fake images and the real images, thus we can use $\log\left(D\left(x\right)\right)$ for the real images and the $\log\left(1-D\left(G\left(z\right)\right)\right)$ for the fake images

Generative Training

- We need to generate images that need to pass fake images for real by using $\log\left(D\left(x\right)\right)$

Meaning of this equation

Discriminant Training

- We have two parts the fake images and the real images, thus we can use $\log\left(D\left(x\right)\right)$ for the real images and the $\log\left(1-D\left(G\left(z\right)\right)\right)$ for the fake images

Generative Training

- We need to generate images that need to pass fake images for real by using $\log\left(D\left(x\right)\right)$

For this we can use the following cost function BCE

$$-x\log\left(\sigma\left(z\right)\right)-\left(1-x\right)\log\left(1-\sigma\left(z\right)\right)$$

Thus the final cost function $V(G, D)$

We have that

$$V(G, D) = E_{x \sim p_{data}(x)} [\log (D(x))] + E_{z \sim p_z(z)} [\log (1 - D(G(z)))]$$

Thus the final cost function $V(G, D)$

We have that

$$V(G, D) = E_{x \sim p_{data}(x)} [\log (D(x))] + E_{z \sim p_z(z)} [\log (1 - D(G(z)))]$$

Then, we have that

$$D_G^* = \arg \max_D V(G, D)$$

Now, G's aim is the reverse

The optimal G minimizes the previous equation when

- $D = D^*_G$

Now, G's aim is the reverse

The optimal G minimizes the previous equation when

- $D = D^*_G$

At the original work

- They use the concept of a minimax game between discriminator and generator

Now, G's aim is the reverse

The optimal G minimizes the previous equation when

- $D = D_G^*$

At the original work

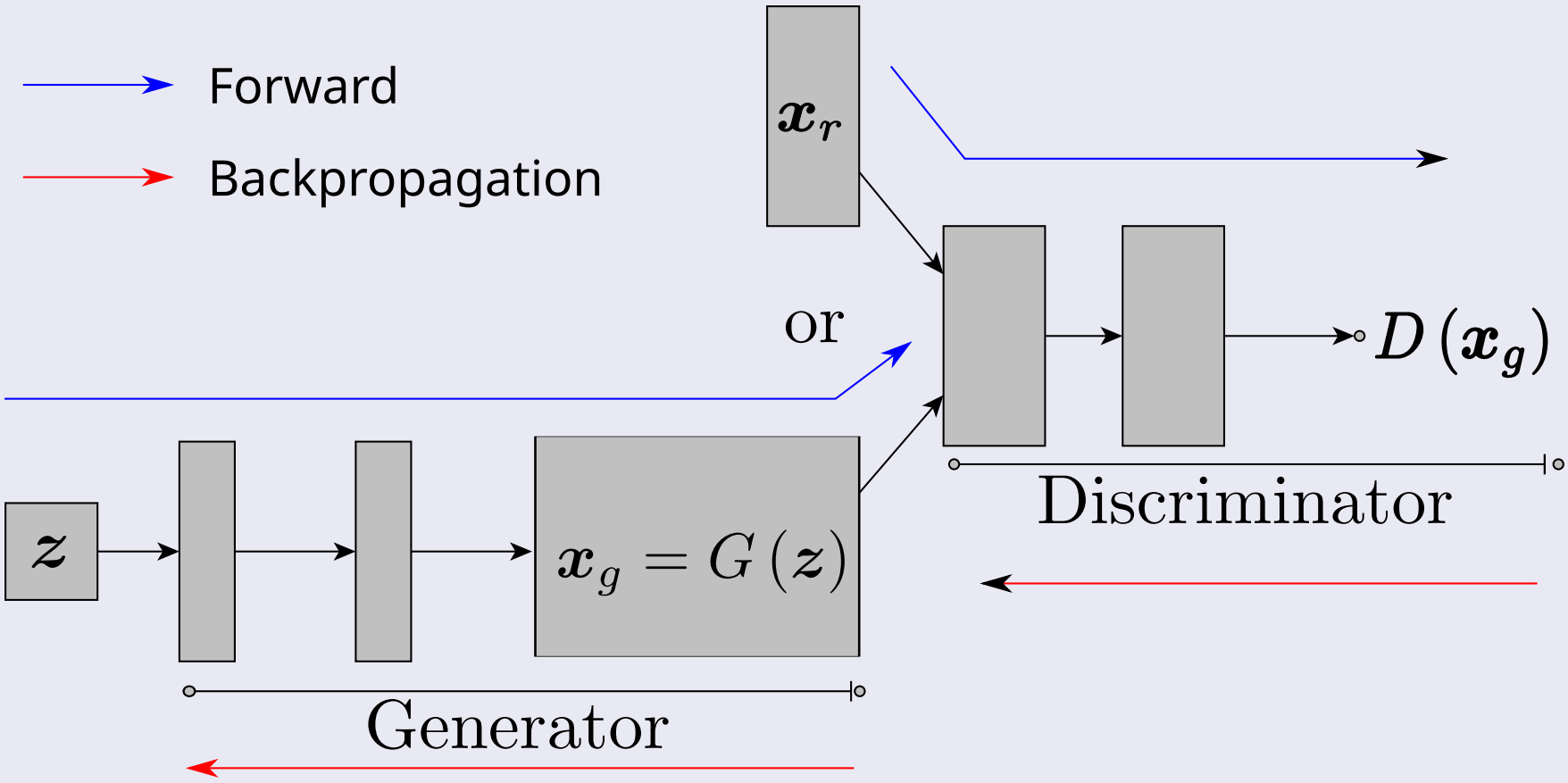
- They use the concept of a minimax game between discriminator and generator

Basically

$$G^* = \arg\min_G V\left(G, D_G^*\right)$$

Therefore, we have two updates

First update the Discriminator



Outline

Now

we must show that this optimization problem has a unique solution G^*

- And that we have $p_G = p_{data}$

Now

we must show that this optimization problem has a unique solution G^*

- And that we have $p_G = p_{data}$

One big idea from the GAN paper

- We have that G does not need to be invertible

Now

we must show that this optimization problem has a unique solution G^*

- And that we have $p_G = p_{data}$

One big idea from the GAN paper

- We have that G does not need to be invertible

A really problematic proof

- We could try to replicate the proof incorrectly by using the classic change of variables of statistics
  - Needing an invertible G!!! Not a good idea

Yes, the following theorem does not work for GAN's

Theorem

- Suppose X is continuous with probability density function $f_x(x)$. Let $y = h(x)$ with h a strictly increasing continuously differentiable function with **inverse** $\boldsymbol{x = g(y)}$. Then $Y = h(X)$ defined by

$$P(c \leq Y \leq d) = P(a \leq Y \leq b),$$

where $a = g(c)$ and $b = g(d)$, is continuous with probability density function $f_Y(y)$ given by $f_Y(y) = f_X(g(y))\, g'(y)$.

Yes, the following theorem does not work for GAN's

Theorem

- Suppose X is continuous with probability density function $f_x(x)$. Let $y = h(x)$ with h a strictly increasing continuously differentiable function with **inverse** $\boldsymbol{x = g(y)}$. Then $Y = h(X)$ defined by

$$P(c \leq Y \leq d) = P(a \leq Y \leq b),$$

where $a = g(c)$ and $b = g(d)$, is continuous with probability density function $f_Y(y)$ given by $f_Y(y) = f_X(g(y))\, g'(y)$.

Clearly, we cannot use the previous theorem given its need of an inverse

- Not only that but the fact that h a strictly increasing continuously differentiable
  - This is too limiting!!!

Actually, you need the Law of the Unconscious Statistician

AKA LOTUS

- Given a random variable x with continuous distribution f_x then the expected value of a function $g : \mathbb{R} \to \mathbb{R}$ evaluated in x is

$$E_h\left[g\left(x\right)\right] = \int_{-\infty}^{\infty} g\left(x\right) f_x\left(x\right) dx$$

- h is the distribution of $g\left(x\right)$ which has support in the Radon–Nikodym Theorem [3].

Actually, we can apply it

Under continuous distribution and assuming $G(z) = x$ or $x \sim G$

$$E_{z \sim p_z(z)} \left[\log \left(1 - D \left(G \left(z \right) \right) \right) \right] = \int_x p_G \left(x \right) \log \left(1 - D \left(x \right) \right) dx$$

Actually, we can apply it

Under continuous distribution and assuming $G(z) = x$ or $x \sim G$

$$E_{z \sim p_z(z)} \left[\log \left(1 - D \left(G \left(z \right) \right) \right) \right] = \int_x p_G \left(x \right) \log \left(1 - D \left(x \right) \right) dx$$

Thus, we can do the following

$$\begin{aligned} V \left(G, D \right) &= \int_x p_{data} \left(x \right) \log D \left(x \right) dx + \int_z p \left(z \right) \log \left(1 - D \left(x \right) \right) dx \\ &= \int_x p_{data} \left(x \right) \log D \left(x \right) dx + \int_x p_G \left(x \right) \log \left(1 - D \left(x \right) \right) dx \\ &= \int_x \left[p_{data} \left(x \right) \log D \left(x \right) + p_G \left(x \right) \log \left(1 - D \left(x \right) \right) \right] dx \end{aligned}$$

Comments

If anybody wants to look at a great proof of Radon-Nikodym Theorem

- https://www.youtube.com/watch?v=sKQ6XLPymCg&t=2585s&ab_channel=InstitutodeMatem%C3%A1ticaPuraeAplicada

Comments

If anybody wants to look at a great proof of Radon-Nikodym Theorem

- https://www.youtube.com/watch?v=sKQ6XLPymCg&t=2585s&ab_channel=InstitutodeMatem%C3%A1ticaPuraeAplicada

A surprising property

- This makes the GAN not dependent on Markov Chains
  - MCMC's depend on having irreducible and aperiodic Markov Chains as targets [4]

Comments

If anybody wants to look at a great proof of Radon-Nikodym Theorem

- https://www.youtube.com/watch?v=sKQ6XLPymCg&t=2585s&ab_channel=InstitutodeMatem%C3%A1ticaPuraeAplicada

A surprising property

- This makes the GAN not dependent on Markov Chains
  - MCMC's depend on having irreducible and aperiodic Markov Chains as targets [4]

GAN's do not talk at all about such chains

- It could be included if necessary [5, 6]

But as the Bard comments

As always Horatio our philosophy falls shorts of our dreams

- "There are more things in heaven and Earth, Horatio, than are dreamt of in your philosophy."

But as the Bard comments

As always Horatio our philosophy falls shorts of our dreams

- "There are more things in heaven and Earth, Horatio, than are dreamt of in your philosophy."

Yes, it is possible that the Universe is full of non-Markovian Probabilites

- As they say, we are the beach of the beginning of knowing the Universe

But as the Bard comments

As always Horatio our philosophy falls shorts of our dreams

- "There are more things in heaven and Earth, Horatio, than are dreamt of in your philosophy."

Yes, it is possible that the Universe is full of non-Markovian Probabilites

- As they say, we are the beach of the beginning of knowing the Universe

Another Beautiful Pebble has been lifted

- And it will take us to another new wonderful place

Now, which discriminator?

Given that the function can be rewritten as

$$f(y) = a \log y + b \log (1 - y)$$

Now, which discriminator?

Given that the function can be rewritten as

$$f(y) = a \log y + b \log (1 - y)$$

How?

- First Derivative
- And Second Derivative Test

In this way, we have

Something Notable

$$V(G,D) \leq \int \max_{y} \left\{ p_{data}(x) \log y + p_G(x) \log(1-y) \right\} dx$$

In this way, we have

Something Notable

$$V(G,D) \leq \int \max_y \left\{ p_{data}(x) \log y + p_G(x) \log(1-y) \right\} dx$$

Under such bound, we have that

- If you let $D(x) = \frac{p_{data}(x)}{p_{data}(x)+p_G(x)}$ you achieve the maximum.

In this way, we have

Something Notable

$$V(G,D) \leq \int \max_y \left\{ p_{data}(x) \log y + p_G(x) \log(1-y) \right\} dx$$

Under such bound, we have that

- If you let $D(x) = \frac{p_{data}(x)}{p_{data}(x)+p_G(x)}$ you achieve the maximum.

Properties

- D is unique as well, as no other choice of D will achieve the maximum.

Outline

First than anything

The generator obtains the best when $p_G = p_{data}$

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

- This means that the Discriminator is completely confused.

First than anything

The generator obtains the best when $p_G = p_{data}$

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

- This means that the Discriminator is completely confused.

The authors then using this idea

- That such G is the solution of the minimax game

We have that

Theorem

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

We have that

Theorem

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

Proof - Backward Direction

- Assuming $p_G = p_{data}$

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

We have that

Theorem

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

Proof - Backward Direction

- Assuming $p_G = p_{data}$

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

Then, we have

$$\begin{aligned} V(G, D_G^*) &= -\log 2 \int_x p_{data}(x)\, dx - \log 2 \int_x p_G(x)\, dx \\ &= -2 \log 2 = -\log 4 \end{aligned}$$

A candidate for the minimum

We need to prove that is always the minimum

- For this, we drop the assumption that $p_G = p_{data}$.

A candidate for the minimum

We need to prove that is always the minimum

- For this, we drop the assumption that $p_G = p_{data}$.

Observe that for any G

- We can plug D_G^* into $C(G) = \max_D V(G, D)$

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

Because

We have that

$$1 - D_G^* (x) = 1 - \frac{p_{data}(x)}{p_{data}(x) + p_G(x)}$$
$$= \frac{p_G(x)}{p_{data}(x) + p_G(x)}$$

Because

We have that

$$1 - D_G^* (x) = 1 - \frac{p_{data}(x)}{p_{data}(x) + p_G(x)}$$
$$= \frac{p_G(x)}{p_{data}(x) + p_G(x)}$$

Now knowing that the minimum is $-\log 4$

$$C(G) = \int_x \left[\underbrace{(\log 2 - \log 2) p_{data}(x)}_{0} + p_{data}(x) \log \left(\frac{p_{data}(x)}{p_{data}(x) + p_G(x)} \right) \right.$$
$$\left. + \underbrace{(\log 2 - \log 2) p_G(x)}_{0} + p_G(x) \log \left(\frac{p_G(x)}{p_{data}(x) + p_G(x)} \right) \right] dx$$

We have the following

We ca rewrite $C(G)$

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

We have the following

We ca rewrite $C(G)$

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

The first part of the equation

$$-\log 2 \left[\int_x \left[p_G(x) + p_{data}(x)\right] dx\right] = -2 \log 2 = -\log 4$$

Now, we have that

The following equality

$$\log 2 + \log\left[\frac{p_{data}(x)}{p_G(x) + p_{data}(x)}\right] = \log\left[2\frac{p_{data}(x)}{p_G(x) + p_{data}(x)}\right]$$
$$= \log\left[\frac{p_{data}(x)}{\frac{p_G(x)+p_{data}(x)}{2}}\right]$$

Now, we have that

The following equality

$$\log 2 + \log\left[\frac{p_{data}(x)}{p_G(x) + p_{data}(x)}\right] = \log\left[2\frac{p_{data}(x)}{p_G(x) + p_{data}(x)}\right]$$
$$= \log\left[\frac{p_{data}(x)}{\frac{p_G(x)+p_{data}(x)}{2}}\right]$$

Therefore, we have

$$C(G) = -\log 4 + \int_x p_{data}(x) \log\left[\frac{p_{data}(x)}{\frac{p_G(x)+p_{data}(x)}{2}}\right] dx + \int_x p_G(x) \log\left[\frac{p_G(x)}{\frac{p_G(x)+p_{data}(x)}{2}}\right] dx$$

A Series of Kullback-Leiber Divergences

We have that

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

A Series of Kullback-Leiber Divergences

We have that

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

The Kullback-Leibler divergence is always non-negative

- we immediately see that $-\log 4$ is the global minimum of $C(G)$

Final Steps

If we prove only one G acheives this

- Because $p_G = p_{data}$ would be the unique point where $C(G) = -\log 4$.

Final Steps

If we prove only one G acheives this

- Because $p_G = p_{data}$ would be the unique point where $C(G) = -\log 4$.

we have to notice a second identity

- The Jenson-Shannon divergence

Jensen-Shannon Divergence

The Jensen–Shannon Divergence (JSD)

- $JSD : \mathcal{M}_+^1(A) \times \mathcal{M}_+^1(A) \to [0, \infty)$ is defined as

$$JSD(P||Q) = \frac{1}{2} KL(P||M) + \frac{1}{2} KL(Q||M)$$

- With $M = \frac{1}{2}(Q + P)$

Jensen-Shannon Divergence

The Jensen–Shannon Divergence (JSD)

- $JSD : \mathcal{M}_+^1(A) \times \mathcal{M}_+^1(A) \to [0, \infty)$ is defined as

$$JSD(P||Q) = \frac{1}{2} KL(P||M) + \frac{1}{2} KL(Q||M)$$

- With $M = \frac{1}{2}(Q + P)$

Property

$$0 \leq JSD(P||Q) \leq 1$$

Jensen-Shannon Divergence

The Jensen–Shannon Divergence (JSD)

- $JSD : \mathcal{M}_+^1(A) \times \mathcal{M}_+^1(A) \to [0, \infty)$ is defined as

$$JSD(P||Q) = \frac{1}{2} KL(P||M) + \frac{1}{2} KL(Q||M)$$

- With $M = \frac{1}{2}(Q + P)$

Property

$$0 \leq JSD(P||Q) \leq 1$$

$JSD(P||Q) = 0$

- When $P = Q$

Therefore, we have that

Thus the minimum $-\log 4$ happens

- When $p_G = p_{data}$ as

$$KL\left[p_{data}||p_{data}\right] + KL\left[p_G||p_G\right] = 0$$

Now, the Last Almost Theorem

Theorem

- If G and D have enough capacity, and at each step of the minimax game, the discriminator is allowed to reach its optimum D^* given G, and p_g is updated so as to improve the criterion

$$V\left(G, D^*\right) = E_{x \sim p_{data}(x)}\left[\log\left(D_G^*\left(x\right)\right)\right] + E_{x \sim p_g}\left[\log\left(1 - D_G^*\left(x\right)\right)\right]$$

then p_g converges to p_{data}.

Proof

Consider $V(G, D) = U(p_g, D)$ as a function of p_g as done in the above criterion.

- Note that $U(p_g, D)$ is convex in p_g

Proof

Consider $V(G,D) = U(p_g, D)$ as a function of p_g as done in the above criterion.

- Note that $U(p_g, D)$ is convex in p_g

How this happens? Given two different probability generators p_{g_1} and p_{g_2}

- Given $\lambda \in (0,1)$, we have that $\lambda p_{g_1} + (1-\lambda) p_{g_2}$

Proof

Consider $V(G,D) = U(p_g, D)$ as a function of p_g as done in the above criterion.

- Note that $U(p_g, D)$ is convex in p_g

How this happens? Given two different probability generators p_{g_1} and p_{g_2}

- Given $\lambda \in (0,1)$, we have that $\lambda p_{g_1} + (1-\lambda) p_{g_2}$

Then

$$U(\lambda p_{g_1} + (1-\lambda) p_{g_2}, D) = \int (\lambda p_{g_1} + (1-\lambda) p_{g_2}) \log(1 - D_G(x)) + const$$

Here

We can see that for U, $E_{x \sim p_{data}(x)} \left[\log \left(D_G \left(x \right) \right) \right]$ is a constant or convex by definition

$$const = E_{x \sim p_{data}(x)} \left[\log \left(D_G \left(x \right) \right) \right]$$

Here

We can see that for U, $E_{x\sim p_{data}(x)}\left[\log\left(D_G\left(x\right)\right)\right]$ is a constant or convex by definition

$$const = E_{x\sim p_{data}(x)}\left[\log\left(D_G\left(x\right)\right)\right]$$

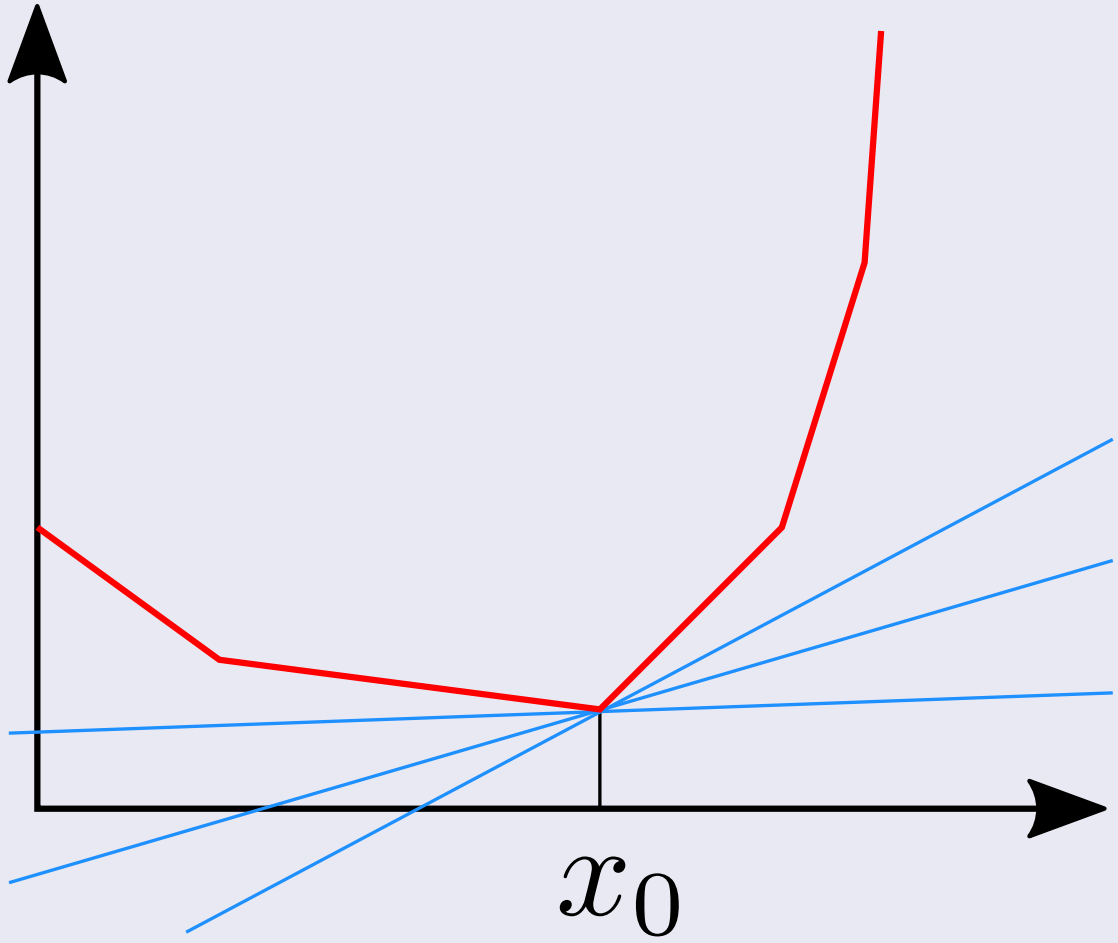
Then, we have that

$$\int \left(\lambda p_{g_1} + \left(1-\lambda\right)p_{g_2}\right)\log\left(1-D_G\left(x\right)\right) = \lambda E_{p_{g_1}}\left[1-D_G\left(x\right)\right] + ...$$
$$\left(1-\lambda\right)E_{p_{g_2}}\left[1-D_G\left(x\right)\right]$$

- Or a convex or concave function

Here, we introduce the concept of subderivatives

Basically used when convex functions are not derivable



Or a more mathematical version

Definition

- A subderivative of a convex function $f : I \to \mathbb{R}$ at a point x_0 in the open interval I is a real number c such that

$$f(x) - f(x_0) \geq c(x - x_0)$$

for all $x \in I$

Or a more mathematical version

Definition

- A subderivative of a convex function $f : I \to \mathbb{R}$ at a point x_0 in the open interval I is a real number c such that

$$f(x) - f(x_0) \geq c(x - x_0)$$

for all $x \in I$

Therefore if we want to improve p_g we improve the generator

- Actually we are looking for the $\arg\inf U(p_g, D^*)$

Or a more mathematical version

Definition

- A subderivative of a convex function $f : I \to \mathbb{R}$ at a point x_0 in the open interval I is a real number c such that

$$f(x) - f(x_0) \geq c(x - x_0)$$

for all $x \in I$

Therefore if we want to improve p_g we improve the generator

- Actually we are looking for the $\arg\inf U(p_g, D^*)$

Important!!!

- Here, $U(p_g, D^*)$ is convex and also concave at the same time

Therefore, for this special case

When looking at infimum, we use the supremum, remember $\min_G \max_D V(G, D)$

- And we know that

$$\sup \max_D V(G, D)$$

Therefore, for this special case

When looking at infimum, we use the supremum, remember $\min_G \max_D V(G, D)$

- And we know that

$$\sup \max_D V(G, D)$$

Therefore, we define the following function

$$f(x) = \sup_{p_g} U(p_g(x), D^*)$$

- It is a convex function

Then

we can look at the argument of the sup for a fixed x

- Then, when getting the optimal argument
  $p_g^* (x) = \arg\sup_{p_g} U (p_g (x), D^*)$

Then

we can look at the argument of the sup for a fixed x

- Then, when getting the optimal argument $p_g^*(x) = \arg\sup_{p_g} U(p_g(x), D^*)$

Properties of the subdifferential

- A point $p_g^*(x)$ is a global minimum of a convex function f if and only if $\partial U\left(p_g^*(x), D^*\right) = 0$ is contained in the subdifferential set $\partial f(x)$

$$\partial U\left(p_g^*(x), D^*\right) = 0 \in \partial f(x)$$

Which is true because

$$\partial U\left(p_g^*(x), D^*\right) = 0$$

- Which happens given that at the optimal $U\left(p_g^*(x), D^*\right) = -\log 4$

Which is true because

$\partial U\left(p_g^*(x), D^*\right) = 0$

- Which happens given that at the optimal $U\left(p_g^*(x), D^*\right) = -\log 4$

Now a big problem we cannot equate this to a convergence by gradient descent

- Actually, we can only use the sub-gradient method

Sub-gradient method

Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function with domain $\mathbb{R}^n$

- The subgradient method uses the iteration

$$x_{k+1} = x_k - \alpha_k g_k$$

  - Where g_k are sub-gradient of $f(x)$

Theorem

We have that

- For constant step size and constant step length, the subgradient method is guaranteed to converge to within some range of the optimal value

$$\lim f_{best}^{k} - U\left(p_{g}^{*}\left(x\right), D^{*}\right) < \epsilon$$

Theorem

We have that

- For constant step size and constant step length, the subgradient method is guaranteed to converge to within some range of the optimal value

$$\lim f_{best}^{k} - U\left(p_g^*\left(x\right), D^*\right) < \epsilon$$

For the diminishing step size rule

- The algorithm is guaranteed to o converge to the optimal value i.e.

$$\lim_{k->\infty} f\left(x_k\right) = U\left(p_g^*\left(x\right), D^*\right)$$

Theorem

We have that

- For constant step size and constant step length, the subgradient method is guaranteed to converge to within some range of the optimal value

$$\lim f_{best}^{k} - U\left(p_g^*\left(x\right), D^*\right) < \epsilon$$

For the diminishing step size rule

- The algorithm is guaranteed to o converge to the optimal value i.e.

$$\lim_{k->\infty} f\left(x_k\right) = U\left(p_g^*\left(x\right), D^*\right)$$

The meaning of the reason of saying

- "with sufficiently small updates of p_g , p_g converges to p_x"

Here, we have a small situation

We have that the last theorem still needs more work

- Because different types of $U(p_g, D)$ will imply different results

Here, we have a small situation

We have that the last theorem still needs more work

- Because different types of $U(p_g, D)$ will imply different results

As they say no free lunch

- More study is needed before saying more about this

Outline

Implementation

Something Notable

- Most implementations do not explicitly perform the minimax optimization

Implementation

Something Notable

- Most implementations do not explicitly perform the minimax optimization

First, we define two models

- The first is the Discriminator $D(x)$
- The second is the Adversarial net $A(z) = D(G(z))$

Implementation

Something Notable

- Most implementations do not explicitly perform the minimax optimization

First, we define two models

- The first is the Discriminator $D(x)$
- The second is the Adversarial net $A(z) = D(G(z))$

Note the following

- Notice, the adversarial net is also a classifier!!!

Thus

You can define

- 1 to be a **real image**
- 0 to be a **fake image**

Thus

You can define

- 1 to be a **real image**
- 0 to be a **fake image**

Train is done in two steps

- Train the Discriminator D to discriminate between real images and fake generated images via a standard 0-1 classification loss function.

Thus

You can define

- 1 to be a **real image**
- 0 to be a **fake image**

Train is done in two steps

- Train the Discriminator D to discriminate between real images and fake generated images via a standard 0-1 classification loss function.

Then,

- Freeze the weights of D and train the adversarial network A with generated images with their labels forced to be 1.

GAN original algorithm

Minibatch stochastic gradient descent training

- **for** training iterations do
- **for** k steps **do**
- Sample minibatch of m noise samples $z_i \sim p_g(z)$
- Sample minibatch of m examples $x_i \sim p_{data}(x)$
- update by stochastic ascent
- $\nabla_{\theta_D} \frac{1}{m} \sum_{i=1}^{m} \left[\log\left(D\left(x_i\right)\right) + \log\left(1 - D\left(G\left(z_i\right)\right)\right) \right]$
- Sample minibatch of m noise samples $z_i \sim p_g(z)$
- Update generator
- $$\nabla_{\theta_G} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(z_i\right)\right)\right)$$

Clearly, there are other methods

Please take a look at

- https://github.com/soumith/ganhacks
  - How to Train a GAN? Tips and tricks to make GANs work
- Jabbar, Abdul, Xi Li, and Bourahla Omar. "A survey on generative adversarial networks: Variants, applications, and training." ACM Computing Surveys (CSUR) 54.8 (2021): 1-49.

Clearly, there are other methods

Please take a look at

- https://github.com/soumith/ganhacks
  - How to Train a GAN? Tips and tricks to make GANs work
- Jabbar, Abdul, Xi Li, and Bourahla Omar. "A survey on generative adversarial networks: Variants, applications, and training." ACM Computing Surveys (CSUR) 54.8 (2021): 1-49.

Last Word

- But not worry they are difficult to train!!!

Outline

Conditional GAN (CGAN) [7]

The objective function of a two-player minimax game

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x|y)] + E_{z\sim p_z(z)}[1-\log D(G(z|y))]$$

Conditional GAN (CGAN) [7]

The objective function of a two-player minimax game

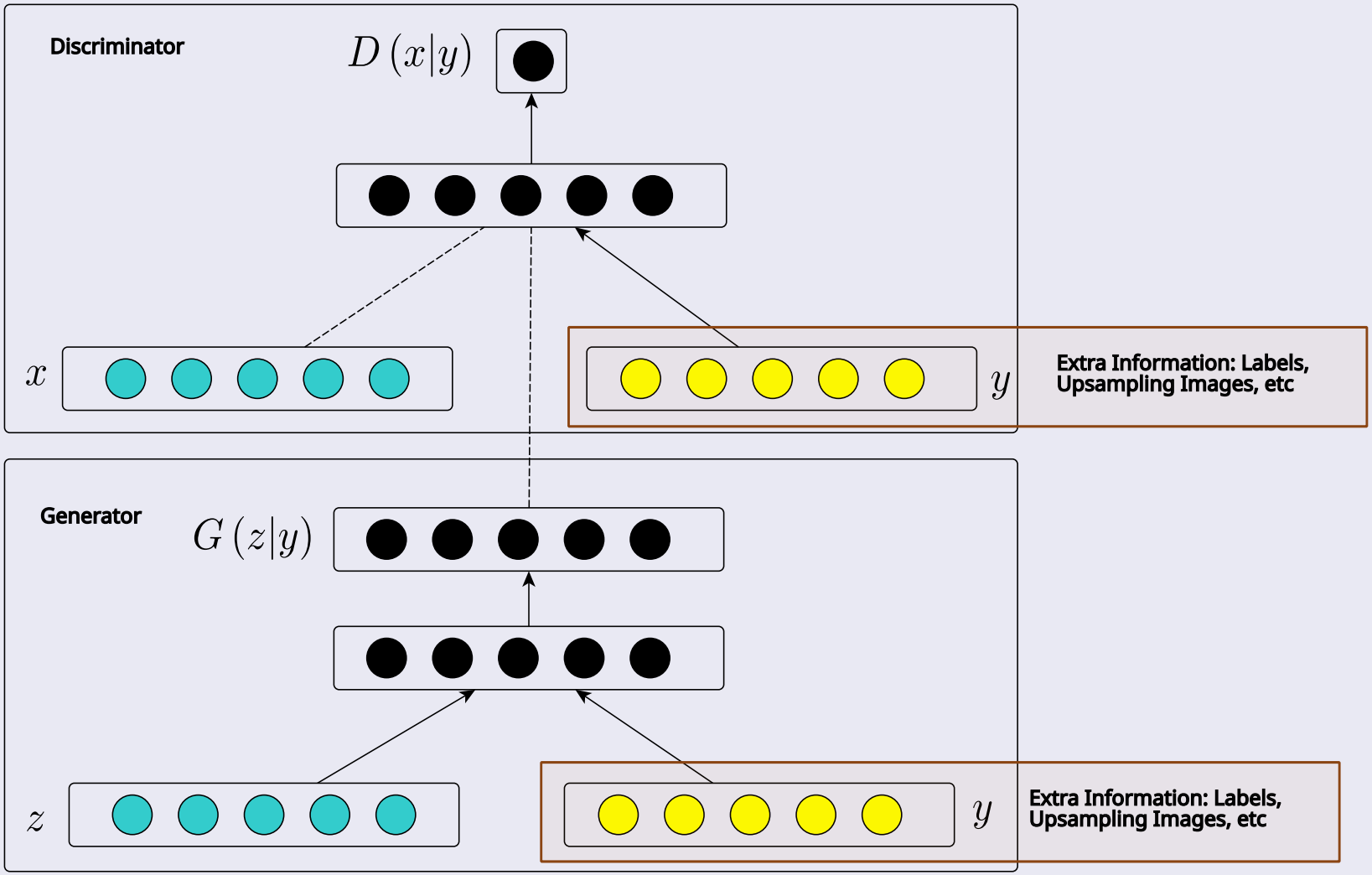
$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x|y)] + E_{z\sim p_z(z)}[1-\log D(G(z|y))]$$

This is extra information

- y could be any kind of auxiliary information, such as
  - class labels or
  - data from other modalities.

Graphical Representation

We have the following representation



Outline

Deep Convolutional GAN (DCGAN) [8]

DCGAN was the first structure that practiced de-convolutional neural networks (de-CNN)

- For stabilization purposes... remember the transpose Convolution

Deep Convolutional GAN (DCGAN) [8]

DCGAN was the first structure that practiced de-convolutional neural networks (de-CNN)

- For stabilization purposes... remember the transpose Convolution

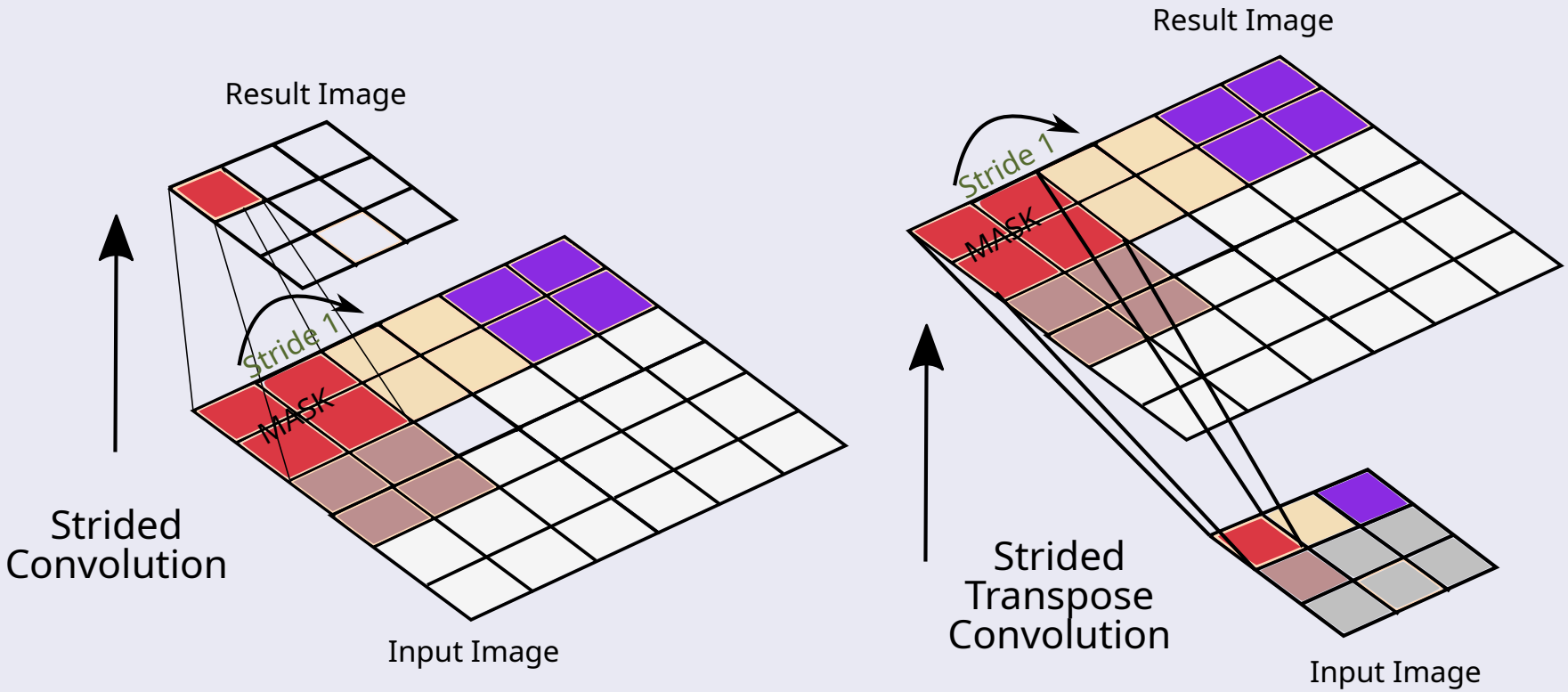
These frameworks consist of two networks

- One network works as a CNN called the generator
- The other network works as a de-CNN called discriminator

Thus

We have the following Architecture

- Architecture guidelines for stable Deep Convolutional GANs
  - Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).



Furthermore

Batch Normalization

- Use batch normalization in both the generator and the discriminator.

Furthermore

Batch Normalization

- Use batch normalization in both the generator and the discriminator.

Finally

- 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.

Outline

Laplacian GAN [9]

A Combination of

- The CGAN
- Laplacian Pyramid

Laplacian GAN [9]

A Combination of

- The CGAN
- Laplacian Pyramid

Laplacian Pyramid

- Somewhat convoluted process
  - First a Gaussian Pyramid is built

$$G(I) = [I_0, I_1, ..., I_K]$$

Here, we have that

$I_0 = I$ the original image

- Here, we use a down sampling operation d which blurs and decimates a $\frac{j}{2} \times \frac{j}{2}$image I,
  - We could use a Convolution

Here, we have that

$I_0 = I$ the original image

- Here, we use a down sampling operation d which blurs and decimates a $\frac{j}{2} \times \frac{j}{2}$image I,
  - We could use a Convolution

Where, we have that the next image I_i is a $\frac{j}{2} \times \frac{j}{2}$

- We also need an up sampling operator u which smooths and expand an image from $\frac{j}{2} \times \frac{j}{2}$ to $\frac{j}{2} \times \frac{j}{2}$
  - Maybe a Deconvolution!!!

Here, we have that

$I_0 = I$ the original image

- Here, we use a down sampling operation d which blurs and decimates a $\frac{j}{2} \times \frac{j}{2}$image I,
  - We could use a Convolution

Where, we have that the next image I_i is a $\frac{j}{2} \times \frac{j}{2}$

- We also need an up sampling operator u which smooths and expand an image from $\frac{j}{2} \times \frac{j}{2}$ to $\frac{j}{2} \times \frac{j}{2}$
  - Maybe a Deconvolution!!!

Using the down sampling, it is easy to build

$$G(I) = [I_0, I_1, ..., I_K]$$

Now, Laplacian Pyramid $\mathscr{L}(\mathscr{I})$

They are constructed by taking the difference between adjacent levels in the Gaussian pyramid

$$h_k = \mathscr{L}_k(I) = \mathcal{G}_k(I) - u(\mathcal{G}_{k+1}(I)) = I_k - u(I_{k+1})$$

- Intuitively, each level captures image structure present at a particular scale.

Now, Laplacian Pyramid $\mathscr{L}(\mathscr{I})$

They are constructed by taking the difference between adjacent levels in the Gaussian pyramid

$$h_k = \mathscr{L}_k(I) = \mathcal{G}_k(I) - u(\mathcal{G}_{k+1}(I)) = I_k - u(I_{k+1})$$

- Intuitively, each level captures image structure present at a particular scale.

The final level of the Laplacian pyramid h_K is not a difference image

- but a Low-Frequency residual equal to the final Gaussian pyramid, $h_K = I_K$

And here comes Deep Learning

Basically, we generate two sequences

- $\{G_0, G_1, ..., G_K\}$
- $\{D_0, D_1, ..., D_K\}$

And here comes Deep Learning

Basically, we generate two sequences

- $\{G_0, G_1, ..., G_K\}$
- $\{D_0, D_1, ..., D_K\}$

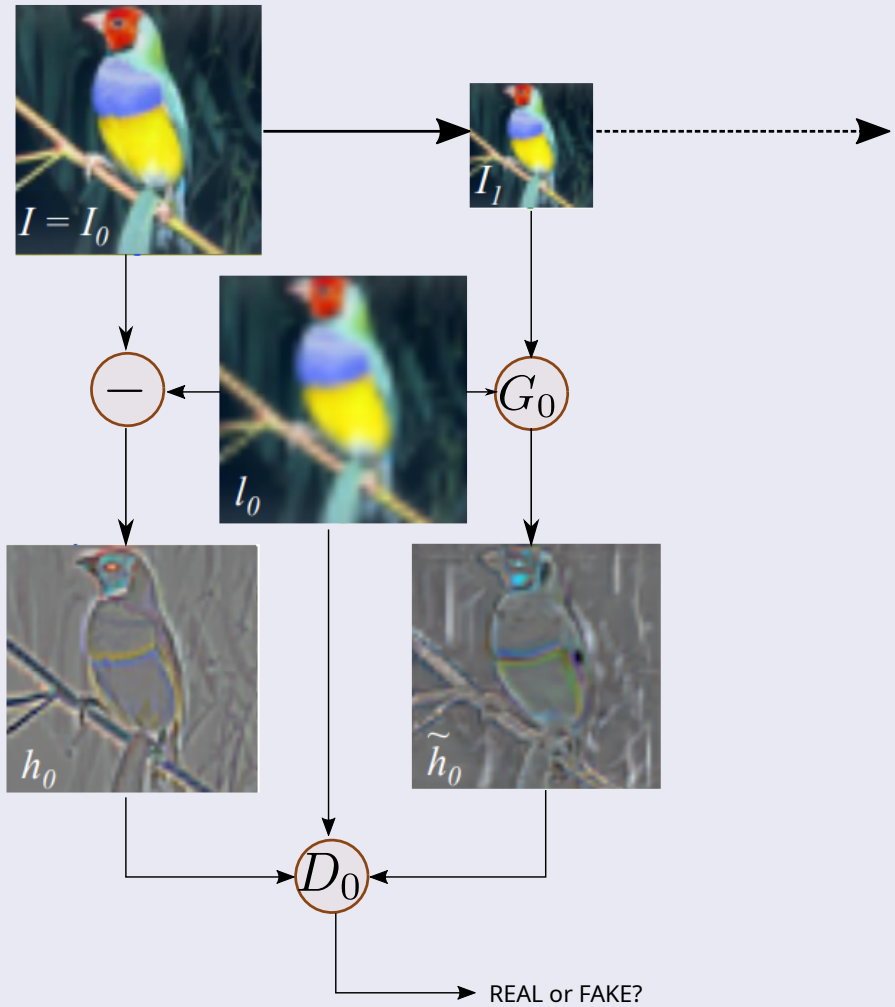
When reconstructing the image or sampling from the generatives

$$\begin{aligned}\widetilde{I}_k =& u\left(\widetilde{I}_{k+1}\right) + \widetilde{h}_k \\ =& u\left(\widetilde{I}_{k+1}\right) + G_k\left(z_k, u\left(\widetilde{I}_{k+1}\right)\right)\end{aligned}$$

- Where $G_k\left(z_k, u\left(\widetilde{I}_{k+1}\right)\right) = G\left(z_k | u\left(\widetilde{I}_{k+1}\right)\right)$

Training of Generative Model

Starting at $k = 0$



A Note

They used a validation set

- Model selection was performed using a combination of visual inspection and a heuristic based on l_2 norm

A Note

They used a validation set

- Model selection was performed using a combination of visual inspection and a heuristic based on l_2 norm

The heuristic computes the error for a given validation image at level k

$$L_k\left(I_k\right) = \min_{\{z_j\}} \left\| G_k\left(z_k, u\left(I_{k+1}\right)\right) - h_k \right\|_2$$

A Note

They used a validation set

- Model selection was performed using a combination of visual inspection and a heuristic based on l_2 norm

The heuristic computes the error for a given validation image at level k

$$L_k\left(I_k\right) = \min_{\{z_j\}} \left\| G_k\left(z_k, u\left(I_{k+1}\right)\right) - h_k \right\|_2$$

In other words, the heuristic is asking

- **Are any of the generated residual images close to the ground truth?**

However

The cost functions used at each D_k and G_k

- It is the Binary Cross Entropy - Fake or Real

However

The cost functions used at each D_k and G_k

- It is the Binary Cross Entropy - Fake or Real

Therefore

- The Laplacian GAN is trained using Binary Cross Entropy
  - But validation used the previous heuristic

However

The cost functions used at each D_k and G_k

- It is the Binary Cross Entropy - Fake or Real

Therefore

- The Laplacian GAN is trained using Binary Cross Entropy
  - But validation used the previous heuristic

However

- We want to train all that in an automatic fashion

The Possible Loss Function

The Final Loss Function

$$
\begin{aligned}
L = & E_{x \sim p_{data}(x)} \left[\log D \left(\widetilde{I}_k | u \left(I_{k+1} \right) \right) \right] + ... \\
& E_{z \sim p_z(z)} \left[1 - \log D \left(G \left(z | u \left(I_{k+1} \right) \right) \right) \right] + ... \\
& \sum \min_{\{z_j\}} \left\| G_k \left(z_k | u \left(\widetilde{I}_{k+1} \right) \right) - h_k \right\|_2
\end{aligned}
$$

Outline

Information Maximizing GAN [10]

We have case where

- We do not know what the optimal representations are for solving a specific problem.
- Sometimes the labeling of the data would be too difficult or expensive.

Information Maximizing GAN [10]

We have case where

- We do not know what the optimal representations are for solving a specific problem.
- Sometimes the labeling of the data would be too difficult or expensive.

Thus, we have that

- We can use unsupervised learning for representation learning.

Information Maximizing GAN [10]

We have case where

- We do not know what the optimal representations are for solving a specific problem.
- Sometimes the labeling of the data would be too difficult or expensive.

Thus, we have that

- We can use unsupervised learning for representation learning.

Basically

- We are learning reusable feature representations

Therefore, InfoGa

Tries to decompose the noise z in two parts

- z, which is treated as source of incompressible noise
- c, which we will call the latent code

Therefore, InfoGa

Tries to decompose the noise z in two parts

- z, which is treated as source of incompressible noise
- c, which we will call the latent code

In this way

- c will target the salient structured semantic features of the data distribution.

Therefore

We denote the set of structured latent variables by

- $c_1, c_2, ..., c_L$ with $P(c_1, c_2, ..., c_L) = \prod_{i=1}^{L} P(c_i)$ or you can get each of them independently

Therefore

We denote the set of structured latent variables by

- $c_1, c_2, ..., c_L$ with $P(c_1, c_2, ..., c_L) = \prod_{i=1}^{L} P(c_i)$ or you can get each of them independently

Thus, the proposal

- To have a generator with two variables $G(z, c)$

Therefore

We denote the set of structured latent variables by

- $c_1, c_2, ..., c_L$ with $P(c_1, c_2, ..., c_L) = \prod_{i=1}^{L} P(c_i)$ or you can get each of them independently

Thus, the proposal

- To have a generator with two variables $G(z, c)$

The proposal is to find codes with high information between c and $G(z, c)$

- For this, we can look at mutual information

Here

We have that

$$I(X;Y) = H(X) - H(X|Y)$$

- Where $\mathrm{H}(X) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log(x_i)$ and

$$\mathrm{H}(X|Y) = -\sum_{i=1}^{n} \mathrm{P}(x_i|y_i) \log(x_i|y_i)$$

Here

We have that

$$I(X;Y) = H(X) - H(X|Y)$$

- Where $\mathrm{H}(X) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log(x_i)$ and

$$\mathrm{H}(X|Y) = -\sum_{i=1}^{n} \mathrm{P}(x_i|y_i) \log(x_i|y_i)$$

Which has the following intuition

- If X and Y are independent, then $I(X;Y) = 0$,
  - because knowing one variable reveals nothing about the other

Here

We have that

$$I(X;Y) = H(X) - H(X|Y)$$

- Where $\mathrm{H}(X) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log(x_i)$ and

$$\mathrm{H}(X|Y) = -\sum_{i=1}^{n} \mathrm{P}(x_i|y_i) \log(x_i|y_i)$$

Which has the following intuition

- If X and Y are independent, then $I(X;Y) = 0$,
  - because knowing one variable reveals nothing about the other

If X and Y are related by a deterministic, invertible function

- Then maximal mutual information is attained.

Given that $I(X;Y) \geq 0$

Consequently

- $H(X) \geq H(X|Y)$

Given that $I(X;Y) \geq 0$

Consequently

- $H(X) \geq H(X|Y)$

Or if you want

- $-\sum_{i=1}^{n} \mathrm{P}(x_i) \log(x_i) \geq -\sum_{i=1}^{n} \mathrm{P}(x_i|y_i) \log(x_i|y_i)$

Using the previous interpretation

This interpretation makes it easy to formulate a cost

- Given any $x \sim P_G(x)$ we want $P_G(c|x)$ to have small entropy

Using the previous interpretation

This interpretation makes it easy to formulate a cost

- Given any $x \sim P_G(x)$ we want $P_G(c|x)$ to have small entropy

In other words

- The information in the latent code c should not be lost in the generation process.

This allows to develop a new cost function

The new cost function

$$\min_{G} \max_{D} V_1(D, G) = V(D, G) - \lambda I(c, G(z, c))$$

This allows to develop a new cost function

The new cost function

$$\min_G \max_D V_1(D,G) = V(D,G) - \lambda I(c, G(z,c))$$

However

- The mutual information term $I(c, G(z,c))$ is hard to maximize as you need $P(c|x)$

This allows to develop a new cost function

The new cost function

$$\min_G \max_D V_1(D,G) = V(D,G) - \lambda I(c, G(z,c))$$

However

- The mutual information term $I(c, G(z,c))$ is hard to maximize as you need $P(c|x)$

Thus, we use a variational approach

- For this, we define an auxiliary distribution $Q(c|x)$ to approximate $P(c|x)$

We have this

The following equation

$$
\begin{aligned}
I\left(c; G\left(z, c\right)\right) =& H\left(c\right) - H\left(c|G\left(z, c\right)\right) \\
=& \mathbb{E}_{x \sim G(z,c)} \left[\mathbb{E}_{c' \sim P(c|x)} \log P\left(c'|x\right)\right] + H\left(c\right) \\
=& \mathbb{E}_{x \sim G(z,c)} \left[\underbrace{D_{KL}\left(P\left(\cdot|x\right)\right) \| Q\left(\cdot|x\right)}_{\geq 0} + \mathbb{E}_{c' \sim P(c|x)} \log Q\left(c'|x\right)\right] + H\left(c\right) \\
\geq& \mathbb{E}_{x \sim G(z,c)} \left[\mathbb{E}_{c' \sim P(c|x)} \log Q\left(c'|x\right)\right] + H\left(c\right)
\end{aligned}
$$

They proved the following lemma

Lemma

- For random variables X, Y and function $f(x, y)$ under suitable regularity conditions:

$$\mathbb{E}_{x\sim X, y\sim Y|x}\left[f\left(x, y\right)\right] = \mathbb{E}_{x\sim X, y\sim Y|x, x'\sim X|y}\left[f\left(x', y\right)\right]$$

They proved the following lemma

Lemma

- For random variables X, Y and function $f(x, y)$ under suitable regularity conditions:

$$\mathbb{E}_{x\sim X, y\sim Y|x}\left[f\left(x, y\right)\right] = \mathbb{E}_{x\sim X, y\sim Y|x, x'\sim X|y}\left[f\left(x', y\right)\right]$$

Thus, we can prove that

$$\begin{aligned} L_I\left(G, Q\right) =& \mathbb{E}_{c\sim P(x), x\sim G(z,c)}\left[\log Q\left(c|x\right)\right] + H\left(c\right) \\ =& \mathbb{E}_{x\sim G(z,c)}\left[\mathbb{E}_{c'\sim P(c|x)}\left[\log Q\left(c'|x\right)\right]\right] + H\left(c\right) \\ \leq& I\left(c; G\left(z, c\right)\right) \end{aligned}$$

They proved the following lemma

Lemma

- For random variables X, Y and function $f(x, y)$ under suitable regularity conditions:

$$\mathbb{E}_{x\sim X, y\sim Y|x}\left[f\left(x, y\right)\right] = \mathbb{E}_{x\sim X, y\sim Y|x, x'\sim X|y}\left[f\left(x', y\right)\right]$$

Thus, we can prove that

$$\begin{aligned} L_I\left(G, Q\right) =& \mathbb{E}_{c\sim P(x), x\sim G(z,c)}\left[\log Q\left(c|x\right)\right] + H\left(c\right) \\ =& \mathbb{E}_{x\sim G(z,c)}\left[\mathbb{E}_{c'\sim P(c|x)}\left[\log Q\left(c'|x\right)\right]\right] + H\left(c\right) \\ \leq& I\left(c; G\left(z, c\right)\right) \end{aligned}$$

This is easy to approximate using Monte Carlo simulation

- In particular, $L_I\left(G, Q\right)$ can be maximized w.r.t. Q directly and w.r.t. G via the re-parametrization trick.

In this way

We have that

$$\min_{G,Q} \max_{D} V_{InfoGAN}(D,G,Q) = V(D,G) - \lambda L_I(G,Q)$$

Outline

They asked [11]

What is the meaning of learning a distribution?

- We always go for a parametric family of densities $(P_\theta)_{\theta \in \mathbb{R}^d}$ when we have a data set $\{x_i\}_{i=1}^N$

$$\max_{\theta \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} \log P_\theta (x_i)$$

They asked [11]

What is the meaning of learning a distribution?

- We always go for a parametric family of densities $(P_\theta)_{\theta \in \mathbb{R}^d}$ when we have a data set $\{x_i\}_{i=1}^N$

$$\max_{\theta \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} \log P_\theta (x_i)$$

Thus, given the $\mathbb{P}_{data}$ (Over all possible data) and $\mathbb{P}_\theta$ final min result

- We want to

$$\min_{\mathbb{P}_\theta} KL \left[\mathbb{P}_{data} || \mathbb{P}_\theta \right]$$

They asked the following

How to improve

- on the various ways to define a distance or divergence $\rho\left(\mathbb{P}_{data}, \mathbb{P}_{\theta}\right)$

They asked the following

How to improve

- on the various ways to define a distance or divergence $\rho\left(\mathbb{P}_{data}, \mathbb{P}_{\theta}\right)$

Therefore, they asked for a continous mapping between

$$\mathscr{M} : \mathbb{R}^{d} \rightarrow \mathbb{P}_{\theta}, \mathscr{M}\left(\theta\right) = \mathbb{P}_{\theta}$$

They asked the following

How to improve

- on the various ways to define a distance or divergence $\rho\left(\mathbb{P}_{data}, \mathbb{P}_{\theta}\right)$

Therefore, they asked for a continous mapping between

$$\mathscr{M} : \mathbb{R}^{d} \to \mathbb{P}_{\theta}, \mathscr{M}\left(\theta\right) = \mathbb{P}_{\theta}$$

Given a sequence $\{\theta_t\}$ with limit θ

- Continuity means that when a sequence of parameters θ_t converges to θ, the distributions $\mathbb{P}_{\theta_t}$ also converge to $\mathbb{P}_{\theta}$.

Thus, they asked for

A weaker distance

- The weaker this distance, the easier it is to define a continuous mapping from θ-space to $\mathbb{P}_\theta$-space

Thus, they asked for

A weaker distance

- The weaker this distance, the easier it is to define a continuous mapping from θ-space to $\mathbb{P}_\theta$-space

After all given a weaker distance ρ

- We would love to have the following loss function

$$\mathscr{L} : \mathbb{R}^d \to \mathbb{R}, \mathscr{L}(\theta) = \rho(\mathbb{P}_\theta, \mathbb{P}_{data})$$

Then, they studied elementary distances and divergences

The Total Variation (TV) distance

$$\delta\left(\mathbb{P}_\theta, \mathbb{P}_{data}\right) = \sup_{A \in \Sigma} \left|\mathbb{P}_\theta\left(A\right) - \mathbb{P}_{data}\left(A\right)\right|$$

- Actually over a measurable space (X, Σ)

Then, they studied elementary distances and divergences

The Total Variation (TV) distance

$$\delta\left(\mathbb{P}_\theta, \mathbb{P}_{data}\right) = \sup_{A \in \Sigma} \left|\mathbb{P}_\theta\left(A\right) - \mathbb{P}_{data}\left(A\right)\right|$$

- Actually over a measurable space (X, Σ)

The Kullback-Leibler (KL) divergence

$$KL\left(\mathbb{P}_{data} || \mathbb{P}_\theta\right) = \int \log\left(\frac{\mathbb{P}_{data}\left(x\right)}{\mathbb{P}_\theta\left(x\right)}\right) \mathbb{P}_{data}\left(x\right) d\mu\left(x\right)$$

Then, they studied elementary distances and divergences

The Total Variation (TV) distance

$$\delta\left(\mathbb{P}_\theta, \mathbb{P}_{data}\right) = \sup_{A \in \Sigma} \left|\mathbb{P}_\theta\left(A\right) - \mathbb{P}_{data}\left(A\right)\right|$$

- Actually over a measurable space (X, Σ)

The Kullback-Leibler (KL) divergence

$$KL\left(\mathbb{P}_{data} || \mathbb{P}_\theta\right) = \int \log\left(\frac{\mathbb{P}_{data}\left(x\right)}{\mathbb{P}_\theta\left(x\right)}\right) \mathbb{P}_{data}\left(x\right) d\mu\left(x\right)$$

The Jensen-Shannon (JS) divergence

$$JS\left(\mathbb{P}_{data}, \mathbb{P}_\theta\right) = KL\left(\mathbb{P}_{data} || \mathbb{P}_m\right) + KL\left(\mathbb{P}_\theta || \mathbb{P}_m\right)$$

- where $\mathbb{P}_m = \frac{\mathbb{P}_\theta + \mathbb{P}_{data}}{2}$

Finally

The Earth-Mover (EM) distance or Wasserstein-1

$$W\left(\mathbb{P}_{data}, \mathbb{P}_{\theta}\right) = \inf_{\gamma \in \prod\left(\mathbb{P}_{data}, \mathbb{P}_{\theta}\right)} \mathbb{E}_{(x,y)\sim\gamma}\left[\|x - y\|\right]$$

- where $\prod\left(\mathbb{P}_{data}, \mathbb{P}_{\theta}\right)$ is the set of all joint distributions $\gamma\left(x, y\right)$ whose marginals are respectively $\mathbb{P}_{data}, \mathbb{P}_{\theta}$.

Finally

The Earth-Mover (EM) distance or Wasserstein-1

$$W\left(\mathbb{P}_{data}, \mathbb{P}_{\theta}\right) = \inf_{\gamma \in \prod\left(\mathbb{P}_{data}, \mathbb{P}_{\theta}\right)} \mathbb{E}_{(x,y)\sim\gamma}\left[\|x - y\|\right]$$

- where $\prod\left(\mathbb{P}_{data}, \mathbb{P}_{\theta}\right)$ is the set of all joint distributions $\gamma\left(x, y\right)$ whose marginals are respectively $\mathbb{P}_{data}, \mathbb{P}_{\theta}$.

Something Notable

- Intuitively, $\gamma\left(x, y\right)$ indicates how much "mass" must be transported from x to y in order to transform the distributions $\mathbb{P}_{data}$ into the distribution $\mathbb{P}_{\theta}$.

Here they did a proof pointing to the Wasserstein-1

Theorem

- Let $\mathbb{P}$ be a distribution on a compact space $\mathcal{X}$ and $(\mathbb{P}_n)_{n\in\mathbb{N}}$ be a sequence of distributions on $\mathcal{X}$. Then, considering all limits as $n \to \infty$

Here they did a proof pointing to the Wasserstein-1

Theorem

- Let $\mathbb{P}$ be a distribution on a compact space $\mathcal{X}$ and $(\mathbb{P}_n)_{n\in\mathbb{N}}$ be a sequence of distributions on $\mathcal{X}$. Then, considering all limits as $n \to \infty$

The following statements are equivalent

1. $W(\mathbb{P}_n, \mathbb{P}_0) \to 0$
2. $\mathbb{P}_n \xrightarrow{\mathcal{D}} \mathbb{P}_0$ where $\xrightarrow{\mathcal{D}}$ represents convergence in distribution for random variables.

In this way, we have

The Kantorovich-Rubinstein duality [12]

$$
\begin{aligned}
W(P,Q) &= \inf_{\pi \in \prod(P,Q)} \mathbb{E}\left[\|x-y\|_2\right] \\
&= \sup_{\|h\|_L \leq 1} \left[\mathbb{E}_{x\sim P}\left[h(x)\right] - \mathbb{E}_{x\sim Q}\left[h(x)\right]\right]
\end{aligned}
$$

In this way, we have

The Kantorovich-Rubinstein duality [12]

$$W(P,Q) = \inf_{\pi \in \prod(P,Q)} \mathbb{E}\left[\|x-y\|_2\right] = \sup_{\|h\|_L \leq 1} \left[\mathbb{E}_{x\sim P}\left[h(x)\right] - \mathbb{E}_{x\sim Q}\left[h(x)\right]\right]$$

Applied to the GAN problem

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

In this way, we have

The Kantorovich-Rubinstein duality [12]

$$W(P,Q) = \inf_{\pi \in \prod(P,Q)} \mathbb{E}\left[\|x-y\|_2\right] = \sup_{\|h\|_L \leq 1} \left[\mathbb{E}_{x\sim P}\left[h(x)\right] - \mathbb{E}_{x\sim Q}\left[h(x)\right]\right]$$

Applied to the GAN problem

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

Where, we have that

- $f : \mathcal{X} \to \mathbb{R}$ a 1-Lipschitz function or Lipschitz continous with derative bounded by 1

Therefore

Therefore, if we have a parameterized family of functions $\{f_w\}_{w \in \mathcal{W}}$

- that are all K-Lipschitz for some K

Therefore

Therefore, if we have a parameterized family of functions $\{f_w\}_{w \in \mathcal{W}}$

- that are all K-Lipschitz for some K

we could consider solving the problem

$$\max_{w \in \mathcal{W}} \left[\mathbb{E}_{x \sim \mathbb{P}_{data}} \left[f_w \left(x \right) \right] - \mathbb{E}_{z \sim p(z)} \left[f_w \left(G_\theta \left(z \right) \right) \right] \right]$$

Therefore, we have that

Theorem

- Let $\mathbb{P}_{data}$ be any distribution. Let $\mathbb{P}_\theta$ be the distribution of $G_\theta(Z)$ with Z a random variable with density p and G_θ a function satisfying If it is continuous in θ, thus it is $W(\mathbb{P}_{data}, \mathbb{P}_\theta)$. Then, there is a solution $f : \mathcal{X} \to \mathbb{R}$ to the problem

$$\max_{w \in \mathcal{W}} \left[\mathbb{E}_{x \sim \mathbb{P}_{data}} [f_w(x)] - \mathbb{E}_{z \sim p(z)} [f_w(G_\theta(z))] \right]$$

Therefore, we have that

Theorem

- Let $\mathbb{P}_{data}$ be any distribution. Let $\mathbb{P}_\theta$ be the distribution of $G_\theta(Z)$ with Z a random variable with density p and G_θ a function satisfying If it is continuous in θ, thus it is $W(\mathbb{P}_{data}, \mathbb{P}_\theta)$. Then, there is a solution $f : \mathcal{X} \to \mathbb{R}$ to the problem

$$\max_{w \in \mathcal{W}} \left[\mathbb{E}_{x \sim \mathbb{P}_{data}} [f_w(x)] - \mathbb{E}_{z \sim p(z)} [f_w(G_\theta(z))] \right]$$

And we have that

$$\nabla_\theta W(\mathbb{P}_{data}, \mathbb{P}_\theta) = -\mathbb{E}_{z \sim p(z)} [\nabla_\theta f(G_\theta(z))]$$

Therefore

We have an algorithm

- That allows to obtain an approximation to the $\mathbb{P}_{data}$ using the Wasserstein loss W

For more in the subject

Take a look at

- Jabbar, Abdul, Xi Li, and Bourahla Omar. "A survey on generative adversarial networks: Variants, applications, and training." ACM Computing Surveys (CSUR) 54.8 (2021): 1-49.

Outline

Anime Character Generation

At Computer Games

- The creation of animated characters in computer games is expensive

Anime Character Generation

At Computer Games

- The creation of animated characters in computer games is expensive

There have great results using

- Progressive Structure-Conditional GAN (PS-CGAN) [13] generates high resolution (e.g.1024 X 1024), and full-body anime characters with particular sequences of poses.

Outline

Image Super-Resolution

The purpose of image super-resolution (SR) is to up-scale the images of low-resolution (LR)

- because images with high-resolution (HR) have better visual qualities.

Image Super-Resolution

The purpose of image super-resolution (SR) is to up-scale the images of low-resolution (LR)

- because images with high-resolution (HR) have better visual qualities.

Examples of this techniques are

- Deep Enhanced SRGAN (ESRGAN) [14]
- Super Resolution Dual-GAN (SRDGAN) [15]
- etc

Example

We have that from [14]

Figure 7: (a) Ground truth, (b) SRGAN result, and (c) ESRGAN result are shown in the figure, which shows that ESRGAN delivers better visual performance than SRGAN. Images from [100].

Outline

Physics-Informed Deep Learning (PIDL)

Recetly, we have that [16]

- Given a labeled data set $\{(x_u, y_u)\}_{i=1}^{N_u}$

Physics-Informed Deep Learning (PIDL)

Recetly, we have that [16]

- Given a labeled data set $\{(x_u, y_u)\}_{i=1}^{N_u}$

We are interested in learning a neural network model, $\widehat{y} = f_\theta(x)$

- We also want to ensure that $\widehat{y}$ satisfies K physics-based equations, $\left\{R^{(k)}(x, y) = 0\right\}_{k=1}^{K}$
  - on a larger set of unlabeled instances $\left\{x_{u_j}\right\}_{j=1}^{N_f}$ with $N_f \gg N_u$

Using a Physics-Based Loss

We define the following

$$\mathcal{L}(\theta) = \frac{1}{N_u} \sum_{i=1}^{N_u} \| y_{u_i} - \widehat{y}_{u_i} \| + \frac{\lambda}{N_f} \sum_{j=1}^{N_f} \sum_{k=1}^{K} R^{(k)} \left(x_{f_j}, \widehat{y}_{f_j} \right)^2$$

Furthermore

Using the previous equation, we have

$$\mathcal{L}_G(\theta) = \frac{1}{N_u}\sum_{i=1}^{N_u} D\left(x_{u_i}, \widehat{y}_{u_i}\right) + \frac{\lambda}{N_f}\sum_{j=1}^{N_f}\sum_{k=1}^{K} R^{(k)}\left(x_{f_j}, \widehat{y}_{f_j}\right)^2$$

$$\mathcal{L}_D(\theta) = \frac{1}{N_u}\sum_{i=1}^{N_u} \log D\left(x_{u_i}, \widehat{y}_{u_i}\right) + \frac{1}{N_u}\sum_{i=1}^{N_u} \log\left[1 - D\left(x_{u_i}, y_{u_i}\right)\right]$$

Furthermore

Using the previous equation, we have

$$\mathcal{L}_G(\theta) = \frac{1}{N_u}\sum_{i=1}^{N_u} D\left(x_{u_i}, \widehat{y}_{u_i}\right) + \frac{\lambda}{N_f}\sum_{j=1}^{N_f}\sum_{k=1}^{K} R^{(k)}\left(x_{f_j}, \widehat{y}_{f_j}\right)^2$$

$$\mathcal{L}_D(\theta) = \frac{1}{N_u}\sum_{i=1}^{N_u} \log D\left(x_{u_i}, \widehat{y}_{u_i}\right) + \frac{1}{N_u}\sum_{i=1}^{N_u} \log\left[1 - D\left(x_{u_i}, y_{u_i}\right)\right]$$

Where

- $\widehat{y}_{u_i} = G\left(x_{u_i}, z_{u_i}\right)$ and $\widehat{y}_{f_j} = G\left(x_{f_j}, z_{f_j}\right)$

Example

From https://physicsbaseddeeplearning.org/intro.html

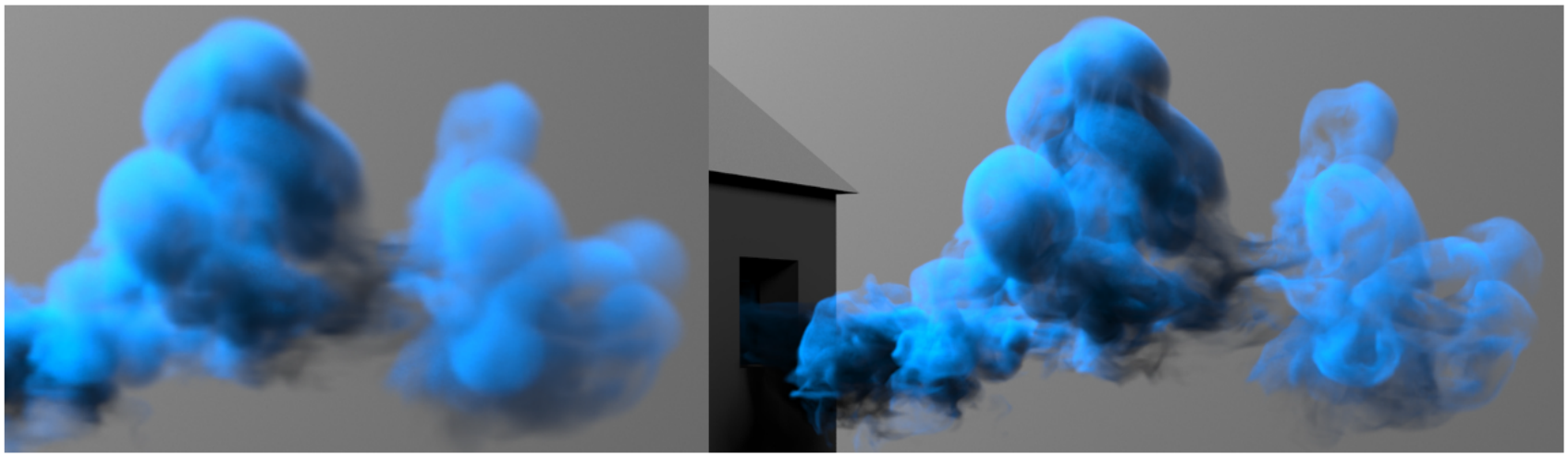


Fig. 31 GANs were shown to work well for tasks such as the inference of super-resolution solutions where the range of possible results can be highly ambiguous.

Conclusion

The explosion of GAN

- It has been incredible

Conclusion

The explosion of GAN

- It has been incredible

And Recently with the definition of

- Out Distribution and Adversarial Attacks

Conclusion

The explosion of GAN

- It has been incredible

And Recently with the definition of

- Out Distribution and Adversarial Attacks

We are going on the direction of

- Synthetic Data!!!


W. R. Gilks, S. Richardson, and D. Spiegelhalter, *Markov chain Monte Carlo in practice*.
CRC press, 1995.


I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.


P. Billingsley, *Probability and measure*.
A Wiley-Interscience publication, New York [u.a.]: Wiley, 3. ed ed., 1995.


S. M. Ross, *Introduction to Probability Models*.
San Diego, CA, USA: Academic Press, sixth ed., 1997.


W. Zhou and M. A. Anastasio, "Markov-chain monte carlo approximation of the ideal observer using generative adversarial networks," in *Medical Imaging 2020: Image Perception, Observer Performance, and Technology Assessment*, vol. 11316, pp. 46–52, SPIE, 2020.


R. Turner, J. Hung, E. Frank, Y. Saatchi, and J. Yosinski, "Metropolis-hastings generative adversarial networks," in *International Conference on Machine Learning*, pp. 6345–6353, PMLR, 2019.


M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.


A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.


E. L. Denton, S. Chintala, a. szlam, and R. Fergus, "Deep generative image models using a ï¿Œlaplacian pyramid of adversarial networks," in *Advances in Neural Information Processing Systems* (C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds.), vol. 28, Curran Associates, Inc., 2015.


X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," *Advances in neural information processing systems*, vol. 29, 2016.


M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 214–223, PMLR, 06–11 Aug 2017.


D. A. Edwards, "On the kantorovich–rubinstein theorem," *Expositiones Mathematicae*, vol. 29, no. 4, pp. 387–398, 2011.


K. Hamada, K. Tachibana, T. Li, H. Honda, and Y. Uchida, "Full-body high-resolution anime generation with progressive structure-conditional generative adversarial networks," *CoRR*, vol. abs/1809.01890, 2018.


X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy, "Esrgan: Enhanced super-resolution generative adversarial networks," in *Proceedings of the European conference on computer vision (ECCV) workshops*, pp. 0–0, 2018.


J. Guan, C. Pan, S. Li, and D. Yu, "SRDGAN: learning the noise prior for super resolution with dual generative adversarial networks," *CoRR*, vol. abs/1903.11821, 2019.


A. Daw, M. Maruf, and A. Karpatne, "Pid-gan: A gan framework based on a physics-informed discriminator for uncertainty quantification with physics," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 237–247, 2021.