
THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

Generative Adversarial Networks

Bruno Gavranović

IRB Seminar

bruno.gavranovic@fer.hr

November 8, 2017

Overview

Meta

- What is this talk about?

Meta

- What is this talk about?
- New idea

Meta

- What is this talk about?
- New idea

Yann LeCun

"The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks).
This, and the variations that are now being proposed is the most interesting idea in the last 10 years in ML, in my opinion."

Meta

- What is this talk about?
- New idea

Yann LeCun

"The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks).
This, and the variations that are now being proposed is the most interesting idea in the last 10 years in ML, in my opinion.”

- Nobody is still sure how it works

Meta

- What is this talk about?
- New idea

Yann LeCun

"The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks).
This, and the variations that are now being proposed is the most interesting idea in the last 10 years in ML, in my opinion."

- Nobody is still sure how it works
- Compared to other ML models, we're still in early stages

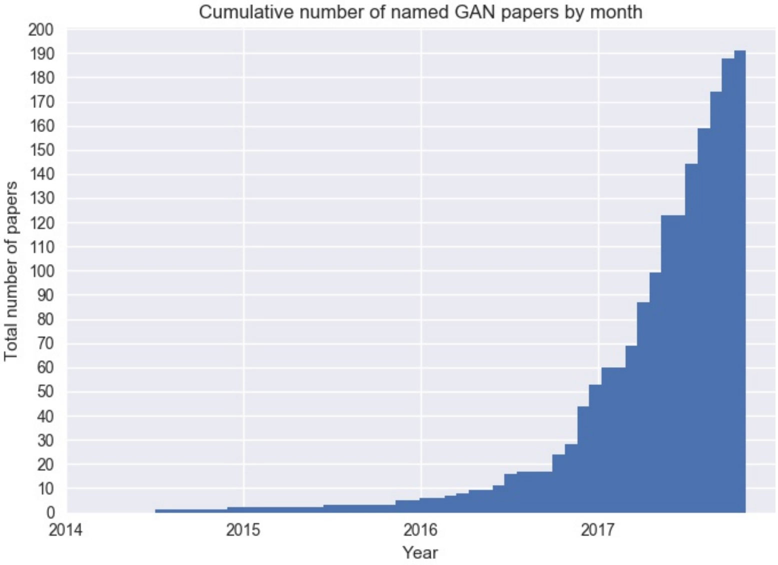
Meta

- What is this talk about?
- New idea

Yann LeCun

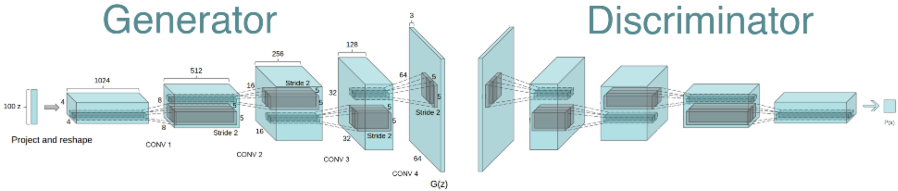
"The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks).
This, and the variations that are now being proposed is the most interesting idea in the last 10 years in ML, in my opinion."

- Nobody is still sure how it works
- Compared to other ML models, we're still in early stages
- Our understanding of it changes from week to week


Cumulative number of named GAN papers by month
Total number of papers
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160
170
180
190
200
2014
2015
2016
2017
Year

GAN: Just tell me what it is

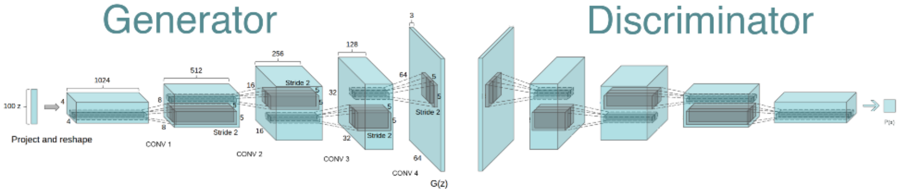
- Unsupervised generative machine learning model in which two neural networks are competing against each other



[0] https://github.com/dmonn/GAN-face-generator

GAN: Just tell me what it is

- Unsupervised generative machine learning model in which two neural networks are competing against each other

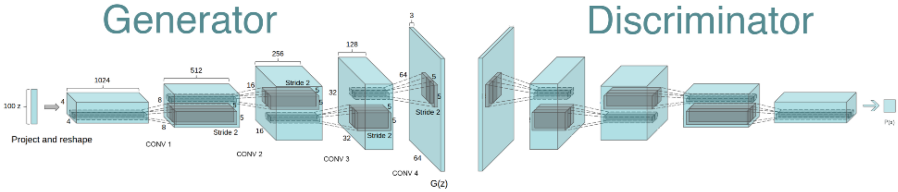


- Instead of using a standard fixed cost function, we *learn* the cost function with the neural network

[0] `https://github.com/dmonn/GAN-face-generator`

GAN: Just tell me what it is

- Unsupervised generative machine learning model in which two neural networks are competing against each other

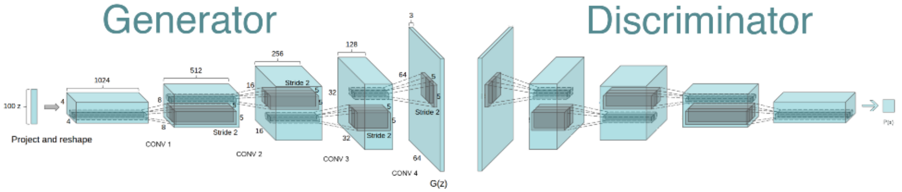


- Instead of using a standard fixed cost function, we *learn* the cost function with the neural network
- We alternate between training different parts of the network

[0] https://github.com/dmonn/GAN-face-generator

GAN: Just tell me what it is

- Unsupervised generative machine learning model in which two neural networks are competing against each other

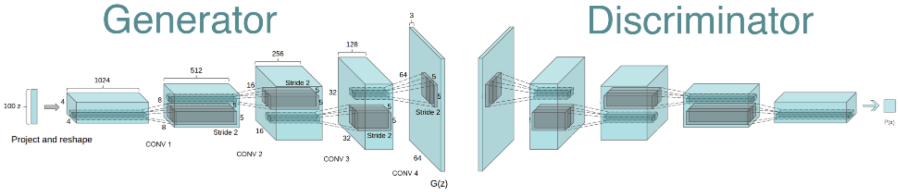


- Instead of using a standard fixed cost function, we *learn* the cost function with the neural network
- We alternate between training different parts of the network
- Forger and the police

[0] https://github.com/dmonn/GAN-face-generator

GAN: Just tell me what it is

- Unsupervised generative machine learning model in which two neural networks are competing against each other



- Instead of using a standard fixed cost function, we *learn* the cost function with the neural network
- We alternate between training different parts of the network
- Forger and the police
- It's difficult to train and analyze

[0] https://github.com/dmonn/GAN-face-generator

Figure: Which side are real images?

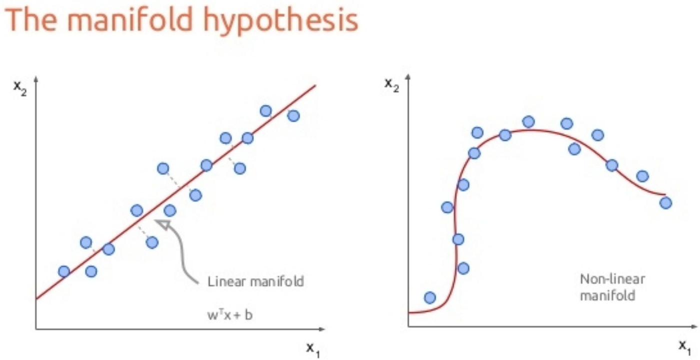
Figure: StackGAN

Are you ready?

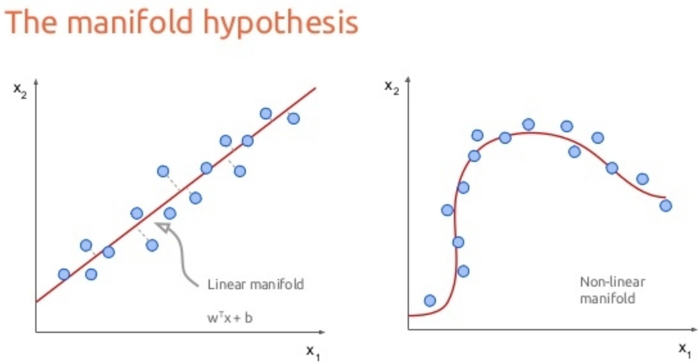
What is this talk about?

- Theory behind Generative Adversarial Networks
- Comparison of different models
- Practical advice for training
- Everything is a Work In Progress.

Prerequisites

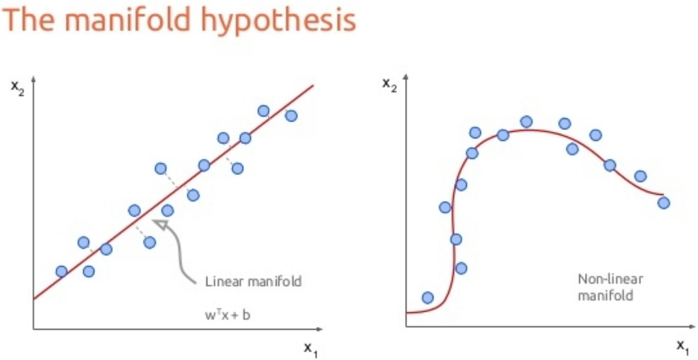


Prerequisites

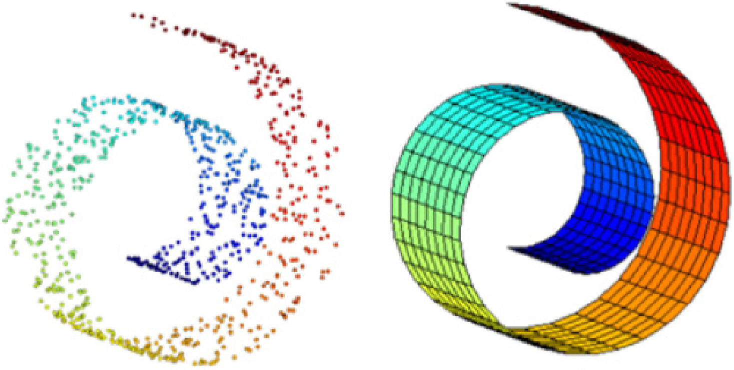


- Understanding real world data

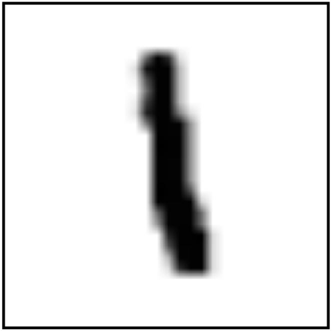
Prerequisites



- Understanding real world data
- Natural data forms a low dimensional manifold in its embedding space



(a) (b)



$$\simeq \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .6 & .8 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .5 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .7 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .9 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .3 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

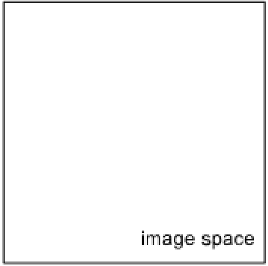


image space

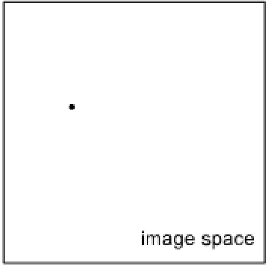


image space

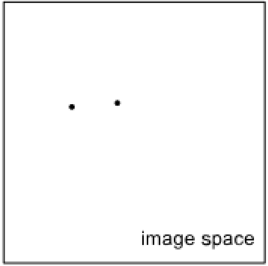


image space

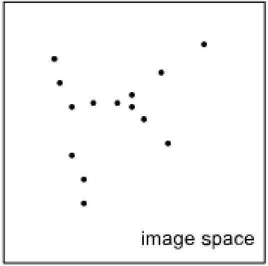
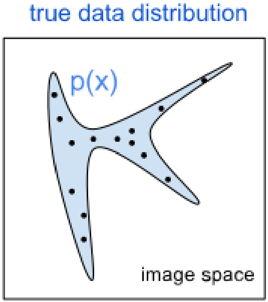
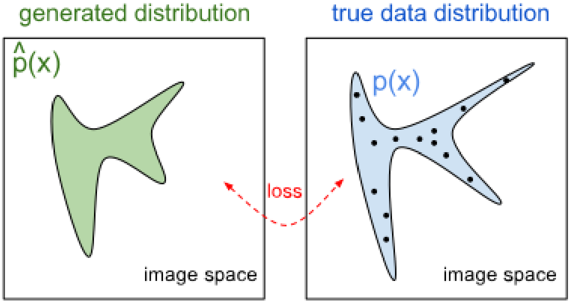
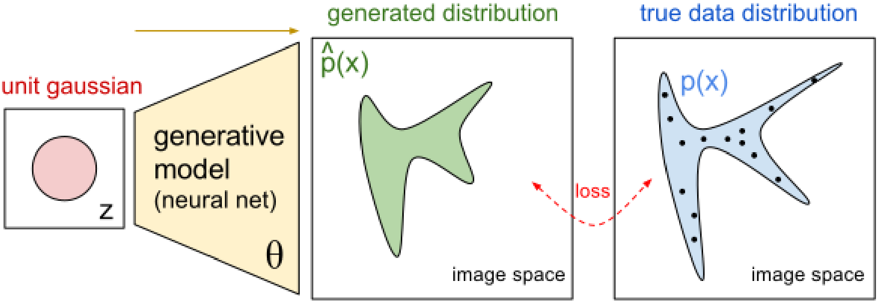


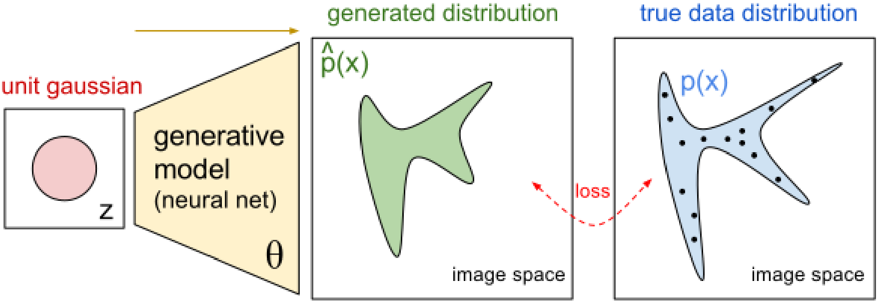
image space


true data distribution
p(x)
image space

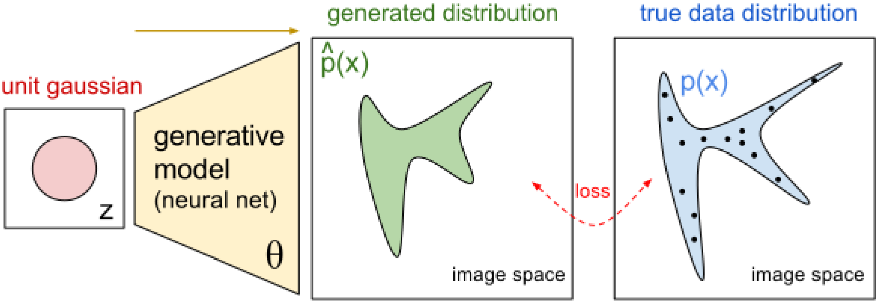

generated distribution
true data distribution
$\hat{p}(x)$
$p(x)$
loss
image space
image space



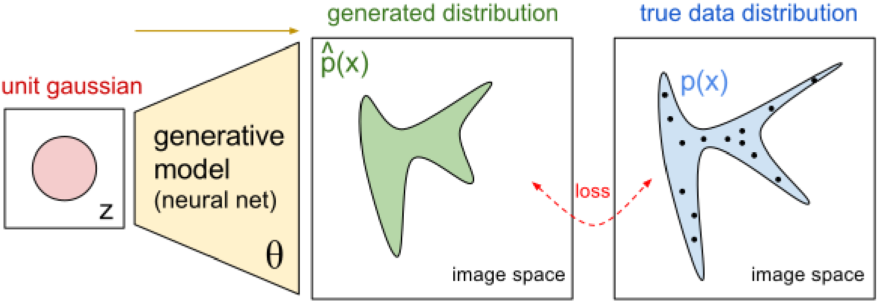
- We have a random vector $z \sim \mathcal{Z}$



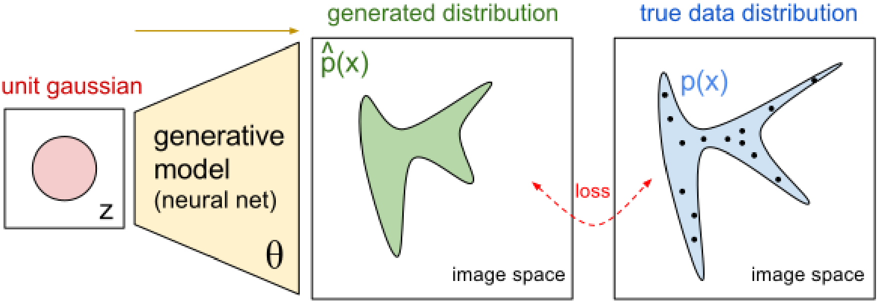
- We have a random vector $z \sim \mathcal{Z}$
- We can define a parametric function $g_\theta : \mathcal{Z} \rightarrow \mathcal{X}$ that generates samples from a certain distribution $\mathbb{P}_\theta$



- We have a random vector $z \sim \mathcal{Z}$
- We can define a parametric function $g_\theta : \mathcal{Z} \to \mathcal{X}$ that generates samples from a certain distribution $\mathbb{P}_\theta$
- We have samples from the real data distribution $\mathbb{P}_r$



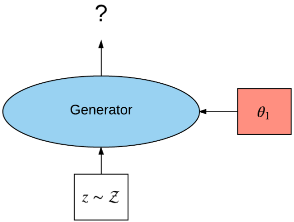
- We have a random vector $z \sim \mathcal{Z}$
- We can define a parametric function $g_\theta : \mathcal{Z} \to \mathcal{X}$ that generates samples from a certain distribution $\mathbb{P}_\theta$
- We have samples from the real data distribution $\mathbb{P}_r$
- By varying θ we can make the $\mathbb{P}_\theta$ distribution arbitrarily close to $\mathbb{P}_r$



- We have a random vector $z \sim \mathcal{Z}$
- We can define a parametric function $g_\theta : \mathcal{Z} \to \mathcal{X}$ that generates samples from a certain distribution $\mathbb{P}_\theta$
- We have samples from the real data distribution $\mathbb{P}_r$
- By varying θ we can make the $\mathbb{P}_\theta$ distribution arbitrarily close to $\mathbb{P}_r$
- How do we define “close"?

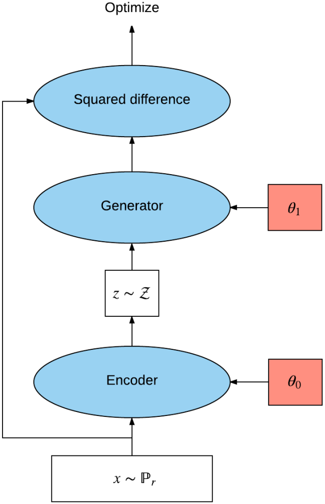
Autoencoder

- How to match the distributions?



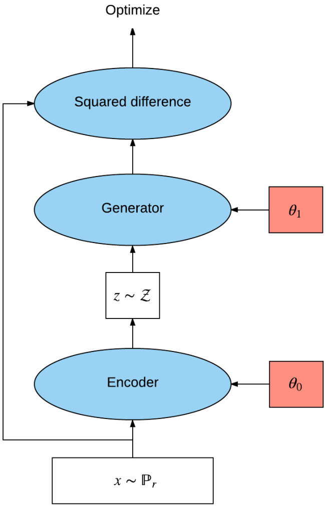
Autoencoder

- How to match the distributions?
- Autoencoder as an example



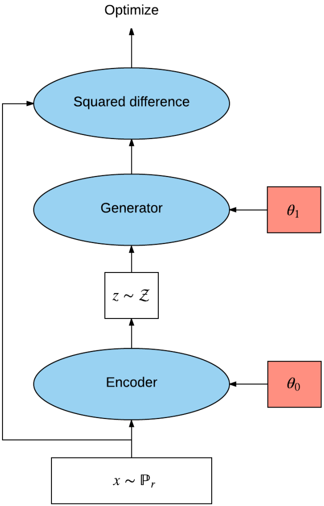
Autoencoder

- How to match the distributions?
- Autoencoder as an example
- Recreation of samples from the real distribution



Autoencoder

- How to match the distributions?
- Autoencoder as an example
- Recreation of samples from the real distribution
- Implicitly training the network to extract useful features

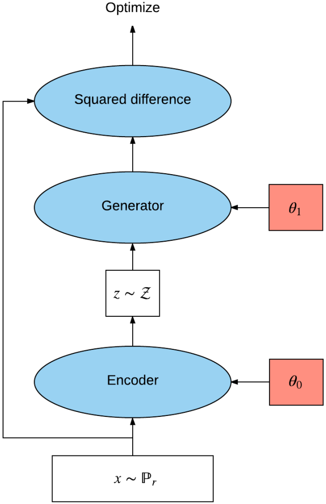


Autoencoder

- How to match the distributions?
- Autoencoder as an example
- Recreation of samples from the real distribution
- Implicitly training the network to extract useful features

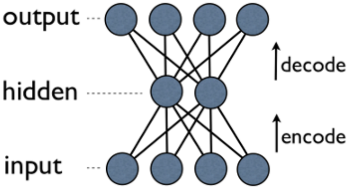
Things to note

1. Cost function is a part of the computational graph
2. Cost function is not parametrized (it's fixed)
3. We train the network with just one cost function
4. Weight freezing and sharing is fixed throughout the training



Autoencoder

- How to match the distributions?
- Autoencoder as an example
- Recreation of samples from the real distribution
- Implicitly training the network to extract useful features



Things to note

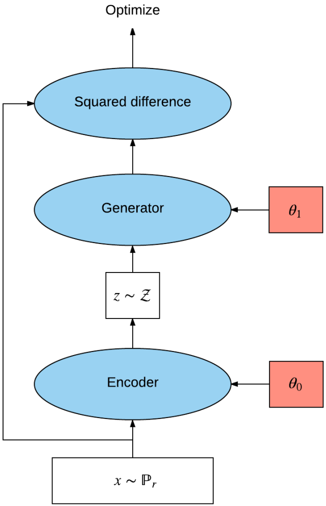
1. Cost function is a part of the computational graph
2. Cost function is not parametrized (it's fixed)
3. We train the network with just one cost function
4. Weight freezing and sharing is fixed throughout the training

Autoencoder

- How to match the distributions?
- Autoencoder as an example
- Recreation of samples from the real distribution
- Implicitly training the network to extract useful features

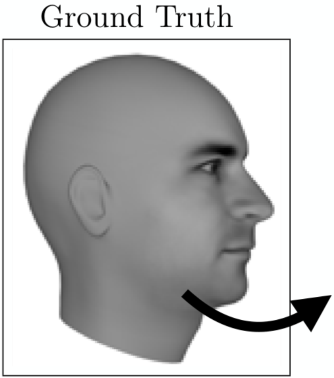
Things to note

1. Cost function is a part of the computational graph
2. Cost function is not parametrized (it's fixed)
3. We train the network with just one cost function
4. Weight freezing and sharing is fixed throughout the training



Cost function

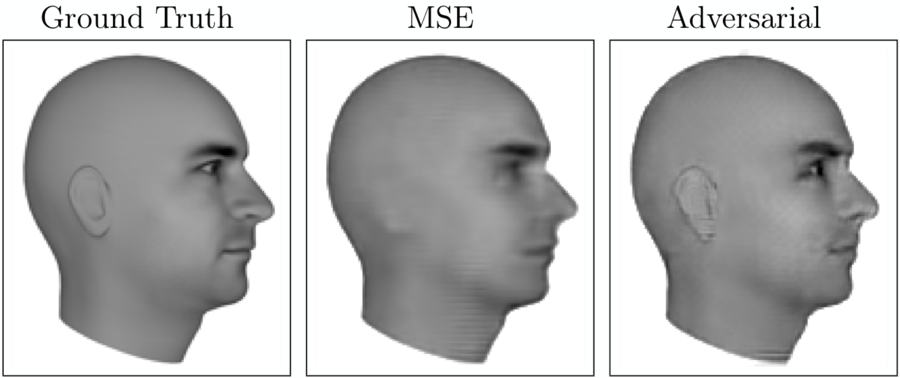
- MSE needs to average over all possibilities and choose a single answer
- Square error - a simple parabola, is it sufficient?



[0] Ian J. Goodfellow: NIPS 2016 Tutorial: Generative Adversarial Networks

Cost function

- MSE needs to average over all possibilities and choose a single answer
- Square error - a simple parabola, is it sufficient?



[0] Ian J. Goodfellow: NIPS 2016 Tutorial: Generative Adversarial Networks

The great idea

The great idea

- Neural network is a computational graph!

The great idea

- Neural network is a computational graph!
- Cost function *is a part of* that computational graph

The great idea

- Neural network is a computational graph!
- Cost function *is a part of* that computational graph
- The great idea - why don't we put a neural network as the cost function?

The great idea

- Neural network is a computational graph!
- Cost function *is a part of* that computational graph
- The great idea - why don't we put a neural network as the cost function?
- Introduces a lot of problems

The great idea

- Neural network is a computational graph!
- Cost function *is a part of* that computational graph
- The great idea - why don't we put a neural network as the cost function?
- Introduces a lot of problems
- How do we even train this thing?

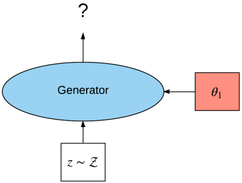
The great idea

- Neural network is a computational graph!
- Cost function *is a part of* that computational graph
- The great idea - why don't we put a neural network as the cost function?
- Introduces a lot of problems
- How do we even train this thing?
- How do we analyze it's behaviour?

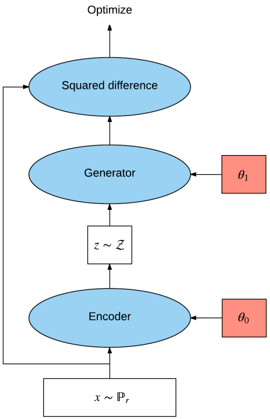
The great idea

- Neural network is a computational graph!
- Cost function *is a part of* that computational graph
- The great idea - why don't we put a neural network as the cost function?
- Introduces a lot of problems
- How do we even train this thing?
- How do we analyze it's behaviour?
- Partial answer to the "How do we know it's close?" question from the beginning

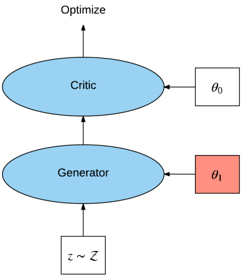
Training regime



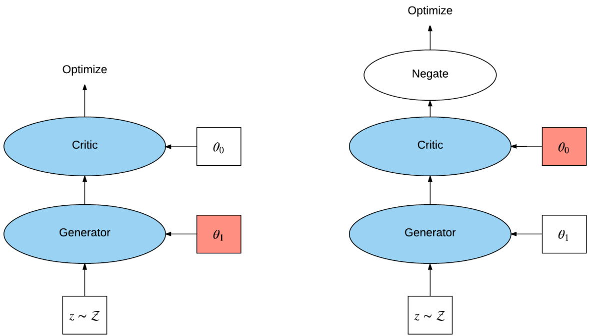
Training regime



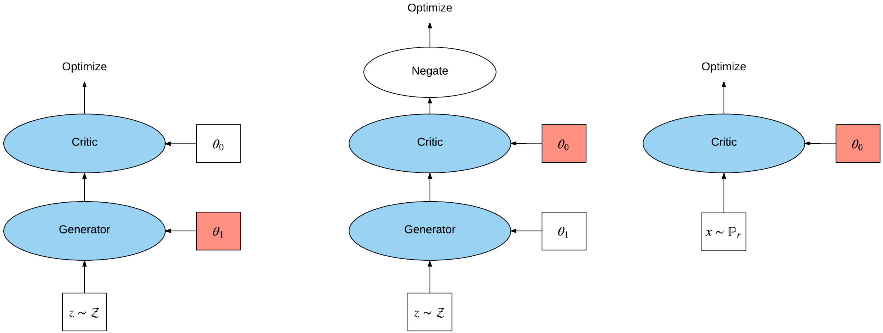
Training regime



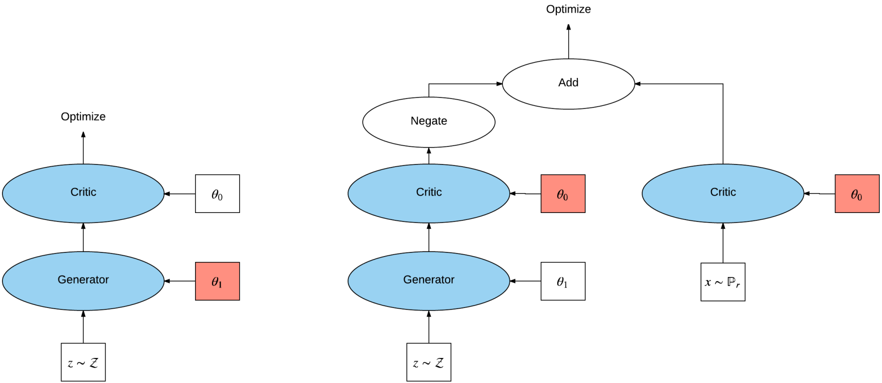
Training regime



Training regime



Training regime



- Two-step optimization optimization process

Different statistical distances

- Recall from before - we have real and fake images, we want the fake distribution to become the same as the real distribution

Different statistical distances

- Recall from before - we have real and fake images, we want the fake distribution to become the same as the real distribution
- Instead of comparing distributions in the pixel space - we use the neural network (the cost function) to transform them to a more suitable representation

Different statistical distances

- Recall from before - we have real and fake images, we want the fake distribution to become the same as the real distribution
- Instead of comparing distributions in the pixel space - we use the neural network (the cost function) to transform them to a more suitable representation
- But we still have to compare those distributions

Different statistical distances

- Recall from before - we have real and fake images, we want the fake distribution to become the same as the real distribution
- Instead of comparing distributions in the pixel space - we use the neural network (the cost function) to transform them to a more suitable representation
- But we still have to compare those distributions
- Defining distance between two points in Euclidean space is intuitive

Different statistical distances

- Recall from before - we have real and fake images, we want the fake distribution to become the same as the real distribution
- Instead of comparing distributions in the pixel space - we use the neural network (the cost function) to transform them to a more suitable representation
- But we still have to compare those distributions
- Defining distance between two points in Euclidean space is intuitive
- How do we define distances between distributions?

Various statistical distances - divergences

- KL-divergence
- JS-divergence
- Earth-mover distance (Wasserstein distance)
- Total variation distance
- Hellinger distance
- Mahalanobis distance
- Bhattacharyya distance
- Energy distance
- ...

Kullback-Leibler and Jensen Shannon divergence

KL-divergence

$$KL(\mathbb{P}||\mathbb{Q}) = \mathbb{E}_{x\sim\mathbb{P}}\left[\log\frac{P(x)}{Q(x)}\right]$$

Kullback-Leibler and Jensen Shannon divergence

KL-divergence

$$KL(\mathbb{P}||\mathbb{Q}) = \mathbb{E}_{x\sim\mathbb{P}}\left[\log\frac{P(x)}{Q(x)}\right] = \int_x \log\frac{P(x)}{Q(x)}P(x)dx$$

Kullback-Leibler and Jensen Shannon divergence

KL-divergence

$$KL(\mathbb{P}||\mathbb{Q}) = \mathbb{E}_{x\sim\mathbb{P}}\left[\log\frac{P(x)}{Q(x)}\right] = \int_x \log\frac{P(x)}{Q(x)}P(x)dx$$

- Doesn't satisfy triangle inequality and symmetry

Kullback-Leibler and Jensen Shannon divergence

KL-divergence

$$KL(\mathbb{P}||\mathbb{Q}) = \mathbb{E}_{x\sim\mathbb{P}}\left[\log\frac{P(x)}{Q(x)}\right] = \int_x \log\frac{P(x)}{Q(x)}P(x)dx$$

- Doesn't satisfy triangle inequality and symmetry
- Origins in information theory

Kullback-Leibler and Jensen Shannon divergence

KL-divergence

$$KL(\mathbb{P}||\mathbb{Q}) = \mathbb{E}_{x\sim\mathbb{P}}\left[\log\frac{P(x)}{Q(x)}\right] = \int_x \log\frac{P(x)}{Q(x)}P(x)dx$$

- Doesn't satisfy triangle inequality and symmetry
- Origins in information theory
- Information gained when revising from prior Q to posterior P

Kullback-Leibler and Jensen Shannon divergence

KL-divergence

$$KL(\mathbb{P}||\mathbb{Q}) = \mathbb{E}_{x\sim\mathbb{P}}\left[\log\frac{P(x)}{Q(x)}\right] = \int_x \log\frac{P(x)}{Q(x)}P(x)dx$$

- Doesn't satisfy triangle inequality and symmetry
- Origins in information theory
- Information gained when revising from prior Q to posterior P

JS-divergence

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

Kullback-Leibler and Jensen Shannon divergence

KL-divergence

$$KL(\mathbb{P}||\mathbb{Q}) = \mathbb{E}_{x\sim\mathbb{P}}\left[\log\frac{P(x)}{Q(x)}\right] = \int_x \log\frac{P(x)}{Q(x)}P(x)dx$$

- Doesn't satisfy triangle inequality and symmetry
- Origins in information theory
- Information gained when revising from prior Q to posterior P

JS-divergence

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

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

Kullback-Leibler and Jensen Shannon divergence

KL-divergence

$$KL(\mathbb{P}||\mathbb{Q}) = \mathbb{E}_{x\sim\mathbb{P}}\left[\log\frac{P(x)}{Q(x)}\right] = \int_x \log\frac{P(x)}{Q(x)}P(x)dx$$

- Doesn't satisfy triangle inequality and symmetry
- Origins in information theory
- Information gained when revising from prior Q to posterior P

JS-divergence

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

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

- Symmetric and bounded

Kullback-Leibler and Jensen Shannon divergence

KL-divergence

$$KL(\mathbb{P}||\mathbb{Q}) = \mathbb{E}_{x\sim\mathbb{P}}\left[\log\frac{P(x)}{Q(x)}\right] = \int_x \log\frac{P(x)}{Q(x)}P(x)dx$$

- Doesn't satisfy triangle inequality and symmetry
- Origins in information theory
- Information gained when revising from prior Q to posterior P

JS-divergence

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

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

- Symmetric and bounded
- Original GAN paper minimizes JS-divergence

Wasserstein (Earth mover) distance

EM distance

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

- Distance function that takes into account underlying geometry of the distributions

Wasserstein (Earth mover) distance

EM distance

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

- Distance function that takes into account underlying geometry of the distributions
- Minimum cost of turning a pile of dirt into the other

Wasserstein (Earth mover) distance

EM distance

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

- Distance function that takes into account underlying geometry of the distributions
- Minimum cost of turning a pile of dirt into the other
- Proposed by Arjovsky *et al.* as improvement to original GAN training - Wasserstein GAN

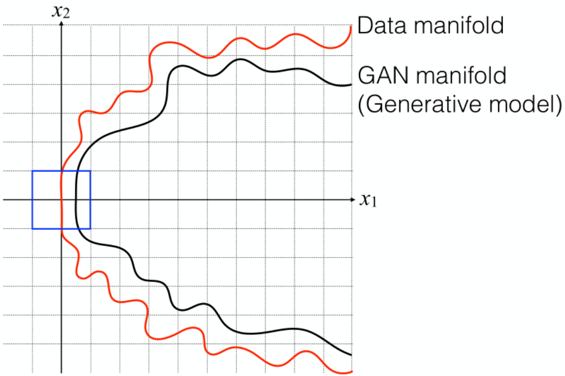
Wasserstein (Earth mover) distance

EM distance

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

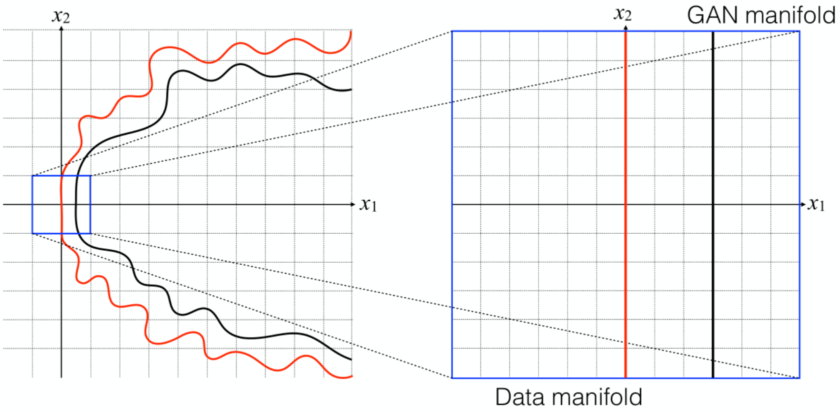
- Distance function that takes into account underlying geometry of the distributions
- Minimum cost of turning a pile of dirt into the other
- Proposed by Arjovsky *et al.* as improvement to original GAN training - Wasserstein GAN
- Intractable

Learning distributions - toy example



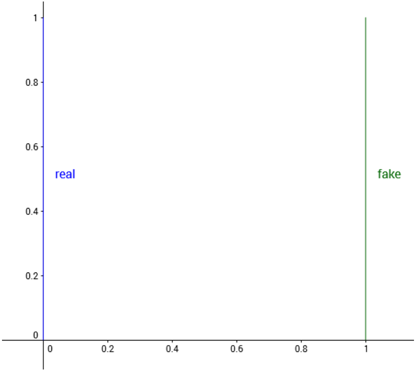
[0] MILA DLSS: `https://drive.google.com/file/d/0B_wzP_JlVFcKQ21udGpTSkh0aVk/view`

Learning distributions - toy example



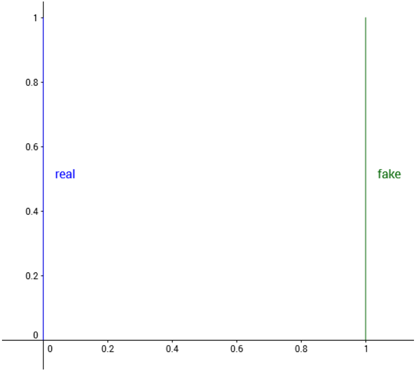
[0] MILA DLSS: `https://drive.google.com/file/d/0B_wzP_JlVFcKQ21udGpTSkh0aVk/view`

Learning distributions - toy example



[0] http://www.alexirpan.com/2017/02/22/wasserstein-gan.html

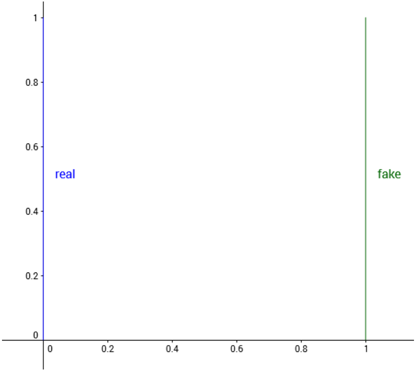
Learning distributions - toy example



$\mathbb{P}_r = (0, y)$ $\mathbb{P}_g = (\theta, y)$

[0] `http://www.alexirpan.com/2017/02/22/wasserstein-gan.html`

Learning distributions - toy example

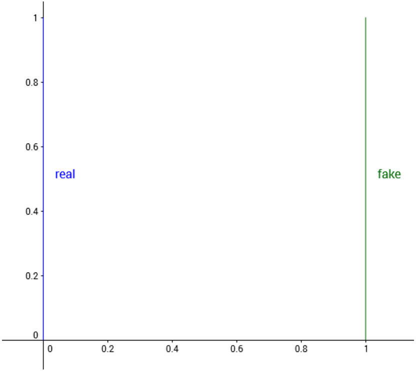


$$\mathbb{P}_r = (0, y) \qquad \mathbb{P}_g = (\theta, y)$$

$$y \sim U[0, 1]$$

[0] http://www.alexirpan.com/2017/02/22/wasserstein-gan.html

Learning distributions - toy example



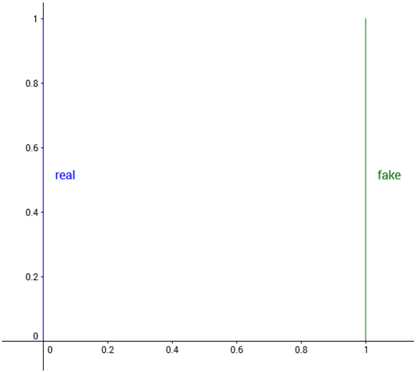
$$\mathbb{P}_r = (0, y) \qquad \mathbb{P}_g = (\theta, y)$$

$$y \sim U\left[0, 1\right]$$

- Two distributions defined on $\mathbb{R}^2$

[0] http://www.alexirpan.com/2017/02/22/wasserstein-gan.html

Learning distributions - toy example



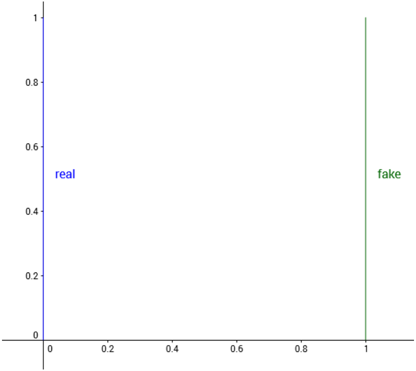
- Two distributions defined on $\mathbb{R}^2$
- $KL(\mathbb{P}_r||\mathbb{P}_g) = +\infty$

$$\mathbb{P}_r = (0, y) \qquad \mathbb{P}_g = (\theta, y)$$

$$y \sim U\big[0, 1\big]$$

[0] http://www.alexirpan.com/2017/02/22/wasserstein-gan.html

Learning distributions - toy example



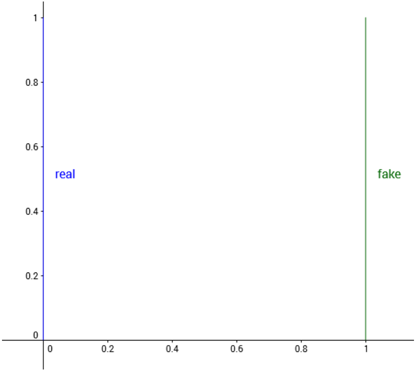
- Two distributions defined on $\mathbb{R}^2$
- $KL(\mathbb{P}_r||\mathbb{P}_g) = +\infty$
- $KL(\mathbb{P}_g||\mathbb{P}_r) = +\infty$

$$\mathbb{P}_r = (0, y) \qquad \mathbb{P}_g = (\theta, y)$$

$$y \sim U[0, 1]$$

[0] `http://www.alexirpan.com/2017/02/22/wasserstein-gan.html`

Learning distributions - toy example



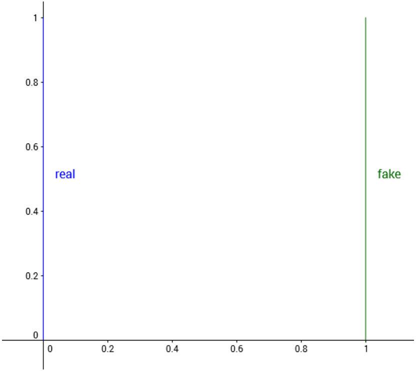
$$\mathbb{P}_r = (0, y) \qquad\qquad \mathbb{P}_g = (\theta, y)$$

$$y \sim U\left[0, 1\right]$$

- Two distributions defined on $\mathbb{R}^2$
- $KL(\mathbb{P}_r||\mathbb{P}_g) = +\infty$
- $KL(\mathbb{P}_g||\mathbb{P}_r) = +\infty$
- $JS(\mathbb{P}_r, \mathbb{P}_g) = JS(\mathbb{P}_g, \mathbb{P}_r) = \log 2$

[0] `http://www.alexirpan.com/2017/02/22/wasserstein-gan.html`

Learning distributions - toy example



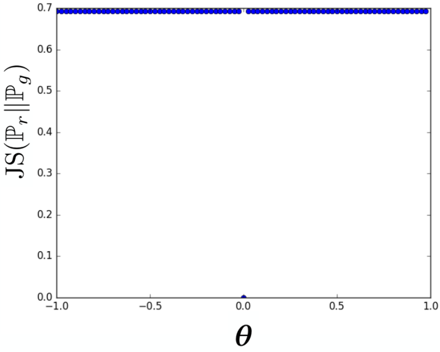
$$\mathbb{P}_r = (0, y) \qquad \mathbb{P}_g = (\theta, y)$$

$$y \sim U[0, 1]$$

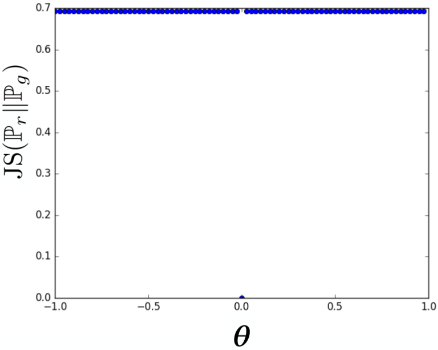
- Two distributions defined on $\mathbb{R}^2$
- $KL(\mathbb{P}_r || \mathbb{P}_g) = +\infty$
- $KL(\mathbb{P}_g || \mathbb{P}_r) = +\infty$
- $JS(\mathbb{P}_r, \mathbb{P}_g) = JS(\mathbb{P}_g, \mathbb{P}_r) = \log 2$
- $W(\mathbb{P}_r, \mathbb{P}_g) = W(\mathbb{P}_g, \mathbb{P}_r) = |\theta|$

[0] http://www.alexirpan.com/2017/02/22/wasserstein-gan.html

JS and EM distance w.r.t. θ

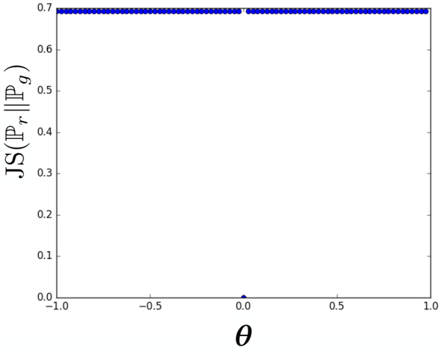


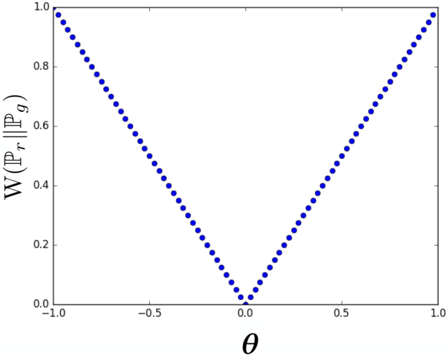
JS and EM distance w.r.t. θ



- JS-divergence gradient is zero

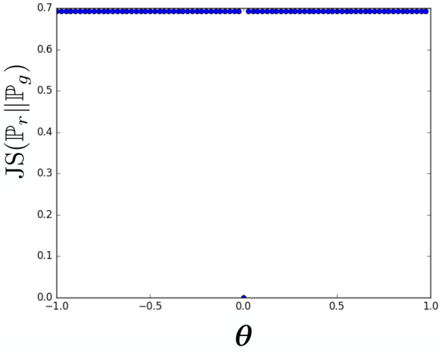
JS and EM distance w.r.t. θ

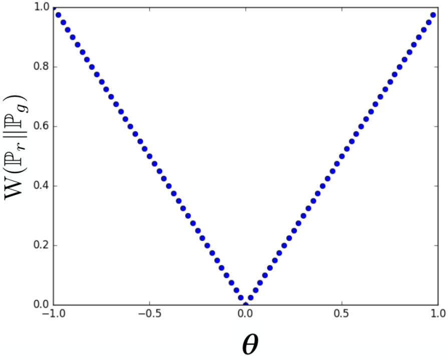




- JS-divergence gradient is zero

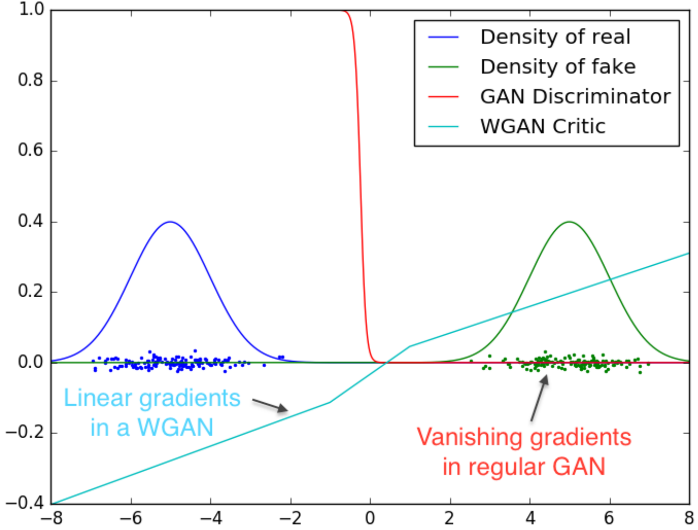
JS and EM distance w.r.t. θ





- JS-divergence gradient is zero
- Wasserstein gradient is constant

Gradients between two gaussian distributions



Short history of training GANs - July 2014

Generative Adversarial Nets

Ian J. Goodfellow, Jean Pouget-Abadie,[*] Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair,[†] Aaron Courville, Yoshua Bengio[‡]
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

2014

GAN Value function

$$\min_{G} \max_{D} \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_r} [\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim \mathcal{Z}} [\log(1 - D(G(\boldsymbol{z})))]$$

- Discriminator output is probability of image being real (1) or fake (0)

Short history of training GANs

- For G fixed, the optimal discriminator D^* is:

$$D^*_G(\boldsymbol{x}) = \frac{p_r(\boldsymbol{x})}{p_r(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

Short history of training GANs

- For G fixed, the optimal discriminator D^* is:

$$D^*_G(\boldsymbol{x}) = \frac{p_r(\boldsymbol{x})}{p_r(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

- Under an optimal discriminator, generator minimizes the JS-divergence

$$JS(\mathbb{P}_r||\mathbb{P}_g) = \frac{1}{2}KL\left(\mathbb{P}_r \middle|\middle| \frac{\mathbb{P}_r + \mathbb{P}_g}{2}\right) + \frac{1}{2}KL\left(\mathbb{P}_g \middle|\middle| \frac{\mathbb{P}_r + \mathbb{P}_g}{2}\right)$$

Short history of training GANs

- For G fixed, the optimal discriminator D^* is:

$$D^*_G(\boldsymbol{x}) = \frac{p_r(\boldsymbol{x})}{p_r(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

- Under an optimal discriminator, generator minimizes the JS-divergence

$$JS(\mathbb{P}_r||\mathbb{P}_g) = \frac{1}{2}KL\left(\mathbb{P}_r \middle|\middle| \frac{\mathbb{P}_r + \mathbb{P}_g}{2}\right) + \frac{1}{2}KL\left(\mathbb{P}_g \middle|\middle| \frac{\mathbb{P}_r + \mathbb{P}_g}{2}\right)$$

- We want the discriminator (cost function) to be as good as possible

Short history of training GANs

- For G fixed, the optimal discriminator D^* is:

$$D_G^*(\boldsymbol{x}) = \frac{p_r(\boldsymbol{x})}{p_r(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

- Under an optimal discriminator, generator minimizes the JS-divergence

$$JS(\mathbb{P}_r||\mathbb{P}_g) = \frac{1}{2}KL\left(\mathbb{P}_r \middle\| \frac{\mathbb{P}_r + \mathbb{P}_g}{2}\right) + \frac{1}{2}KL\left(\mathbb{P}_g \middle\| \frac{\mathbb{P}_r + \mathbb{P}_g}{2}\right)$$

- We want the discriminator (cost function) to be as good as possible
- With JS-divergence good discriminator leads to vanishing gradients

Short history of training GANs

- For G fixed, the optimal discriminator D^* is:

$$D^*_G(\boldsymbol{x}) = \frac{p_r(\boldsymbol{x})}{p_r(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

- Under an optimal discriminator, generator minimizes the JS-divergence

$$JS(\mathbb{P}_r || \mathbb{P}_g) = \frac{1}{2} KL\left(\mathbb{P}_r \middle\| \frac{\mathbb{P}_r + \mathbb{P}_g}{2}\right) + \frac{1}{2} KL\left(\mathbb{P}_g \middle\| \frac{\mathbb{P}_r + \mathbb{P}_g}{2}\right)$$

- We want the discriminator (cost function) to be as good as possible
- With JS-divergence good discriminator leads to vanishing gradients
- Disciminator should be good but not too good?

Short history of training GANs

- For G fixed, the optimal discriminator D^* is:

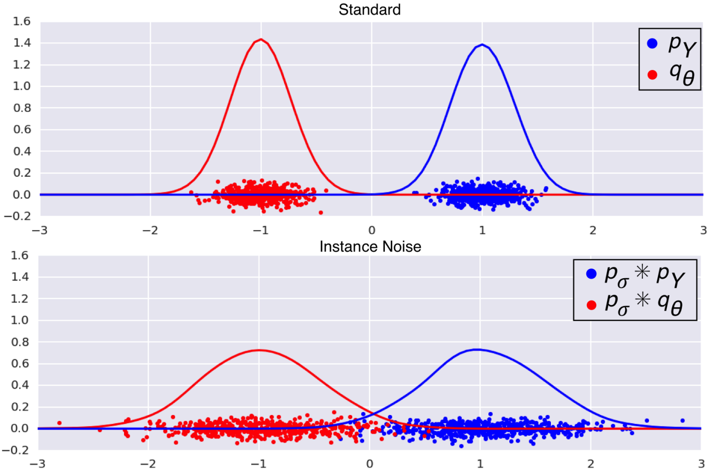
$$D_G^*(\boldsymbol{x}) = \frac{p_r(\boldsymbol{x})}{p_r(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

- Under an optimal discriminator, generator minimizes the JS-divergence

$$JS(\mathbb{P}_r||\mathbb{P}_g) = \frac{1}{2}KL\left(\mathbb{P}_r \middle|\middle| \frac{\mathbb{P}_r + \mathbb{P}_g}{2}\right) + \frac{1}{2}KL\left(\mathbb{P}_g \middle|\middle| \frac{\mathbb{P}_r + \mathbb{P}_g}{2}\right)$$

- We want the discriminator (cost function) to be as good as possible
- With JS-divergence good discriminator leads to vanishing gradients
- Disciminator should be good but not too good?
- Bag of tricks

Trick - adding noise



- Matching the noise corresponds to matching the underlying distributions

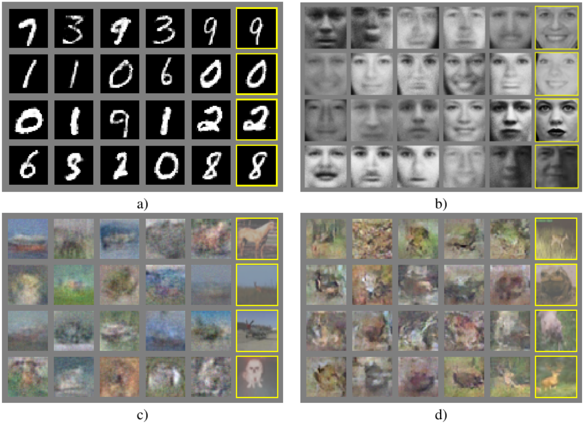
[0] http://www.inference.vc/instance-noise-a-trick-for-stabilising-gan-training/

Trick - cherrypicking architecture

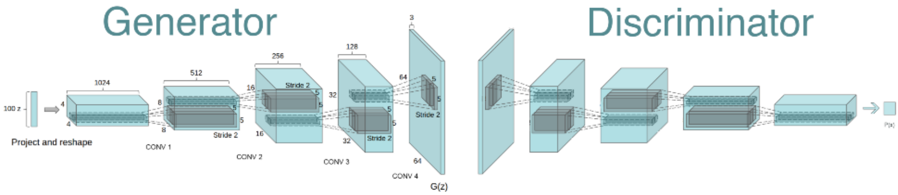
(TFD) [28], and CIFAR-10 [21]. The generator nets used a mixture of rectifier linear activations [19, 9] and sigmoid activations, while the discriminator net used maxout [10] activations. Dropout [17] was applied in training the discriminator net. While our theoretical framework permits the use of dropout and other noise at intermediate layers of the generator, we used noise as the input to only the bottommost layer of the generator network.

Trick - cherrypicking architecture

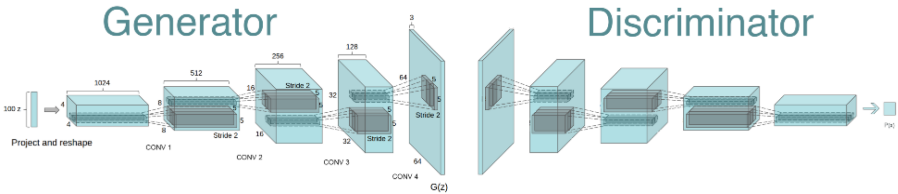
(TFD) [28], and CIFAR-10 [21]. The generator nets used a mixture of rectifier linear activations [19, 9] and sigmoid activations, while the discriminator net used maxout [10] activations. Dropout [17] was applied in training the discriminator net. While our theoretical framework permits the use of dropout and other noise at intermediate layers of the generator, we used noise as the input to only the bottommost layer of the generator network.



Trick - cherrypicking architecture - DCGAN



[1] https://www.youtube.com/watch?v=X1mUN6dD8uEt=795s

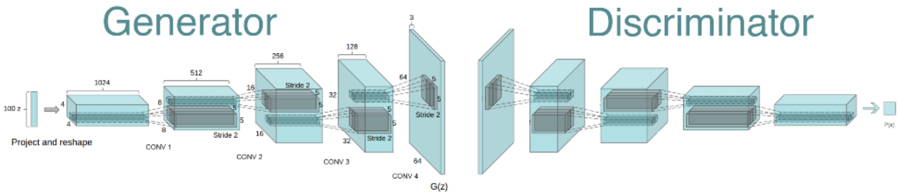


Architecture guidelines for stable Deep Convolutional GANs

- 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.

[1] https://www.youtube.com/watch?v=X1mUN6dD8uEt=795s

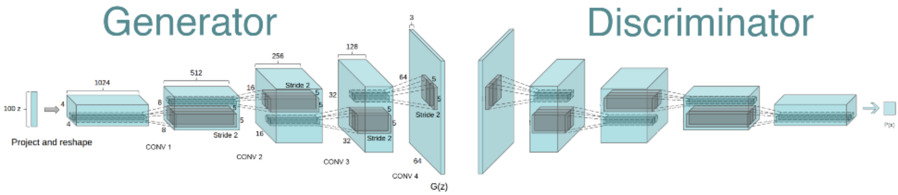
Trick - cherrypicking architecture - DCGAN



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.

[1] https://www.youtube.com/watch?v=X1mUN6dD8uEt=795s



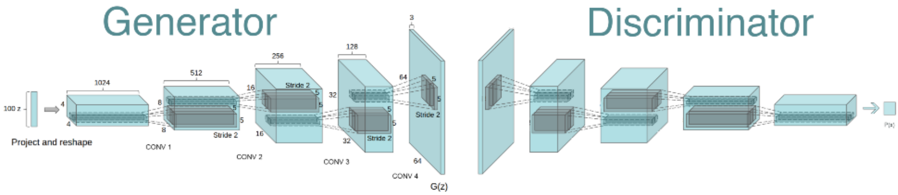
Architecture guidelines for stable Deep Convolutional GANs

- 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.

- Chintala reports that they had model generator [1]

[1] https://www.youtube.com/watch?v=X1mUN6dD8uEt=795s

Trick - cherrypicking architecture - DCGAN



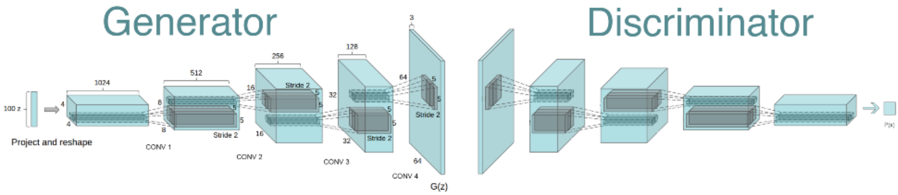
Architecture guidelines for stable Deep Convolutional GANs

- 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.

- Chintala reports that they had model generator [1]
- No explanation for model performance

[1] https://www.youtube.com/watch?v=X1mUN6dD8uEt=795s

Trick - cherrypicking architecture - DCGAN



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.

- Chintala reports that they had model generator [1]
- No explanation for model performance
- Very unstable

[1] https://www.youtube.com/watch?v=X1mUN6dD8uEt=795s

Trick - not training the discriminator until convergence

- Making sure the discriminator is not "too far ahead of" the generator

Trick - not training the discriminator until convergence

- Making sure the discriminator is not "too far ahead of" the generator
- Each step trains the generator once and discriminator once

Trick - not training the discriminator until convergence

- Making sure the discriminator is not "too far ahead of" the generator
- Each step trains the generator once and discriminator once
- Many alternative training plans with questionable efficiency

Trick - not training the discriminator until convergence

- Making sure the discriminator is not "too far ahead of" the generator
- Each step trains the generator once and discriminator once
- Many alternative training plans with questionable efficiency
- Has a nice side effect of speeding up the training time

But there's a bigger problem...

- **Value of the loss doesn't tell us anything!**
- No correlation between loss and image quality
- Problem stems from mentioned undesired properties of JS-divergence

Short history of training GANs - January 2017

Wasserstein GAN

Martin Arjovsky[1], Soumith Chintala[2], and Léon Bottou[1,2]

[1]Courant Institute of Mathematical Sciences
[2]Facebook AI Research

Jan 2017

WGAN Value function

$$\min_{G} \max_{D \in \mathcal{D}} \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_r} \left[D(\boldsymbol{x}) \right] - \mathbb{E}_{\boldsymbol{z} \sim \mathcal{Z}} \left[D(G(\boldsymbol{z})) \right]$$

$\mathcal{D}$ - set of all K-Lipschitz functions

Short history of training GANs

EM distance

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

[2] http://cedricvillani.org/wp-content/uploads/2012/08/preprint-1.pdf

EM distance

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

- How to compute the infimum?

[2] http://cedricvillani.org/wp-content/uploads/2012/08/preprint-1.pdf

Short history of training GANs

EM distance

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

- How to compute the infimum?
- 1000 page book on Optimal Transport [2]

[2] http://cedricvillani.org/wp-content/uploads/2012/08/preprint-1.pdf

Short history of training GANs

EM distance

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

- How to compute the infimum?
- 1000 page book on Optimal Transport [2]
- Kantorovich-Rubinstein duality

[2] http://cedricvillani.org/wp-content/uploads/2012/08/preprint-1.pdf

Short history of training GANs

EM distance

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

- How to compute the infimum?
- 1000 page book on Optimal Transport [2]
- Kantorovich-Rubinstein duality

Kantorovich-Rubinstein duality

$$K \cdot W(\mathbb{P}_r, \mathbb{P}_g) = \sup_{||f||_L \leq K} \mathbb{E}_{x\sim\mathbb{P}_r}\left[f(x)\right] - \mathbb{E}_{x\sim\mathbb{P}_g}\left[f(x)\right]$$

[2] http://cedricvillani.org/wp-content/uploads/2012/08/preprint-1.pdf

Short history of training GANs

EM distance

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

- How to compute the infimum?
- 1000 page book on Optimal Transport [2]
- Kantorovich-Rubinstein duality

Kantorovich-Rubinstein duality

$$K \cdot W(\mathbb{P}_r, \mathbb{P}_g) = \sup_{||f||_L \le K} \mathbb{E}_{x\sim\mathbb{P}_r} \left[f(x) \right] - \mathbb{E}_{x\sim\mathbb{P}_g} \left[f(x) \right]$$

- Supremum is the norm over all K-Lipschitz functions $f : \mathcal{X} \to \mathbb{R}$

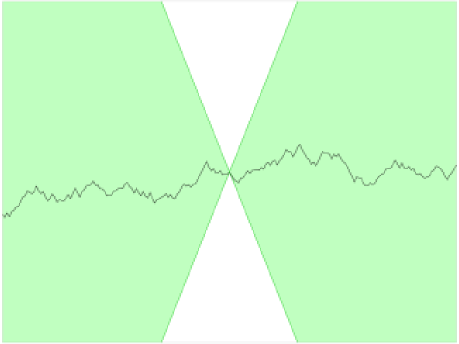
[2] `http://cedricvillani.org/wp-content/uploads/2012/08/preprint-1.pdf`

Lipschitz continuity

K-Lipschitz function

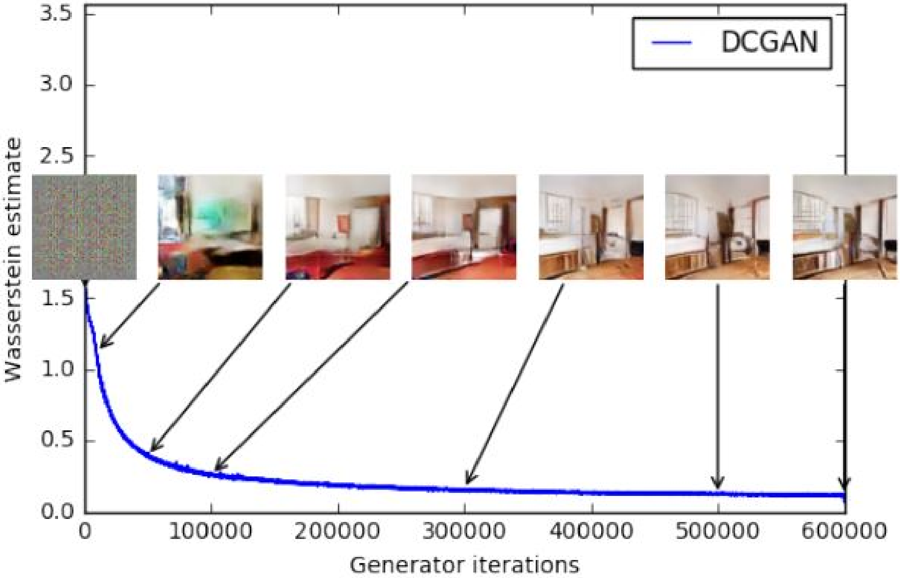
$$|f(x_1) - f(x_2)| \leq K \cdot |x_1 - x_2|, \qquad \forall x_1, x_2$$

- Continuous function which is limited how fast it can change
- Every function that has a bounded first derivative is Lipschitz

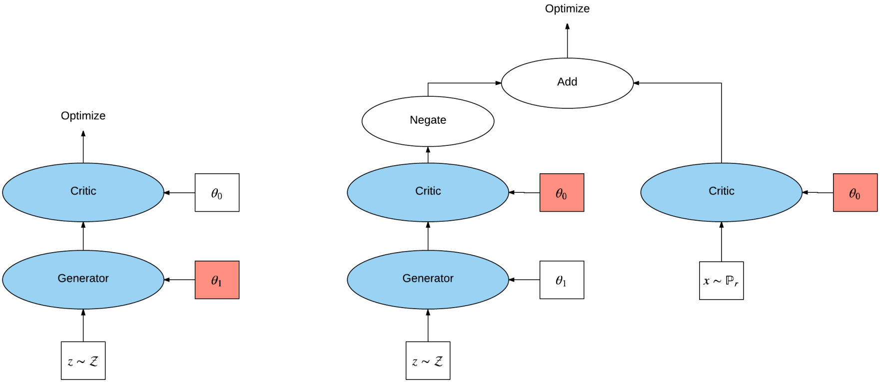


[2] https://en.wikipedia.org/wiki/Lipschitz_continuity

Meaningful loss function

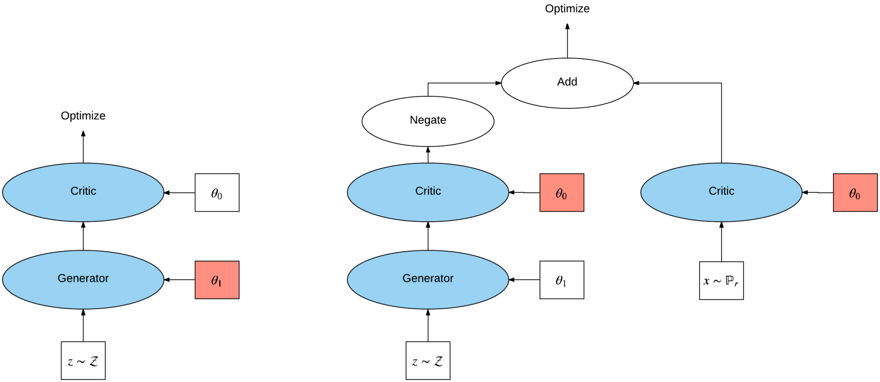


Wasserstein GAN



- Using EM distance instead of JS-divergence leads to robust models
- Theoretical and empirical data
- No need for tricks

Wasserstein GAN



- Using EM distance instead of JS-divergence leads to robust models
- Theoretical and empirical data
- No need for tricks
- Open question - how to effectively enforce the Lipschitz constraint?

Method #1 - Weight Clipping

- After optimization step, clip all weights to $[-c, c]$

Method #1 - Weight Clipping

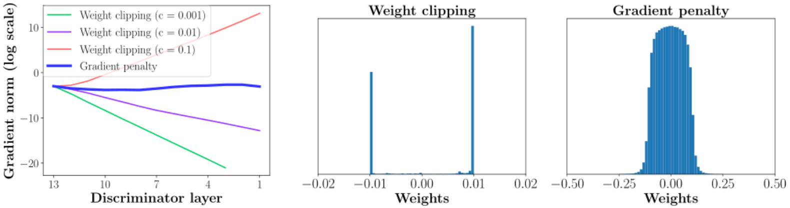
- After optimization step, clip all weights to $[-c, c]$
- Results in critic being a subset of K-Lipschitz functions, where K is a function of c and the critic's architecture

Method #1 - Weight Clipping

- After optimization step, clip all weights to $[-c, c]$
- Results in critic being a subset of K-Lipschitz functions, where K is a function of c and the critic's architecture

Method #1 - Weight Clipping

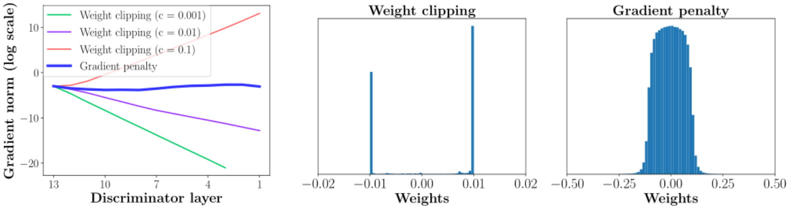
- After optimization step, clip all weights to $[-c, c]$
- Results in critic being a subset of K-Lipschitz functions, where K is a function of c and the critic's architecture



- Works! But...

Method #1 - Weight Clipping

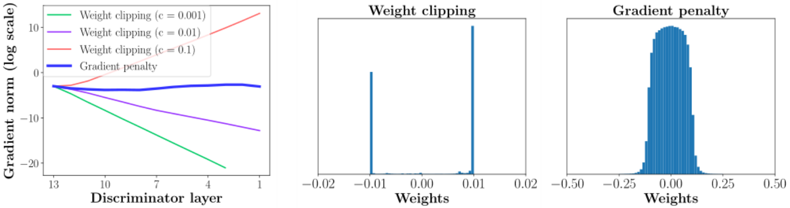
- After optimization step, clip all weights to $[-c, c]$
- Results in critic being a subset of K-Lipschitz functions, where K is a function of c and the critic's architecture



- Works! But...
- Capacity underuse

Method #1 - Weight Clipping

- After optimization step, clip all weights to $[-c, c]$
- Results in critic being a subset of K-Lipschitz functions, where K is a function of c and the critic's architecture



- Works! But...
- Capacity underuse
- Exploding and vanishing gradients

Method #2 - Gradient Penalty

- Property of the optimal WGAN critic $|f(x) - f(y)| \leq |x - y|$

Method #2 - Gradient Penalty

- Property of the optimal WGAN critic $|f(x) - f(y)| \leq |x - y|$
- Optimal WGAN critic has gradient norm 1

Method #2 - Gradient Penalty

- Property of the optimal WGAN critic $|f(x) - f(y)| \leq |x - y|$
- Optimal WGAN critic has gradient norm 1
- Add a regularization term to the WGAN value function

Method #2 - Gradient Penalty

- Property of the optimal WGAN critic $|f(x) - f(y)| \leq |x - y|$
- Optimal WGAN critic has gradient norm 1
- Add a regularization term to the WGAN value function
- Sampling along straight lines

$$\epsilon \sim U\left[0, 1\right], \boldsymbol{x} \sim \mathbb{P}_g, \tilde{\boldsymbol{x}} \sim \mathbb{P}_r$$

$$\hat{\boldsymbol{x}} = t\boldsymbol{x} + (1 - t)\,\tilde{\boldsymbol{x}}$$

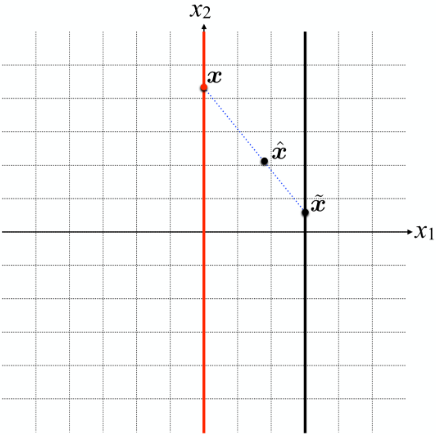
Method #2 - Gradient Penalty

- Property of the optimal WGAN critic $|f(x) - f(y)| \leq |x - y|$
- Optimal WGAN critic has gradient norm 1
- Add a regularization term to the WGAN value function
- Sampling along straight lines

$$\epsilon \sim U[0,1], \boldsymbol{x} \sim \mathbb{P}_g, \tilde{\boldsymbol{x}} \sim \mathbb{P}_r$$

$$\hat{\boldsymbol{x}} = t\boldsymbol{x} + (1-t)\tilde{\boldsymbol{x}}$$

- Gradient penalty



Method #2 - Gradient Penalty

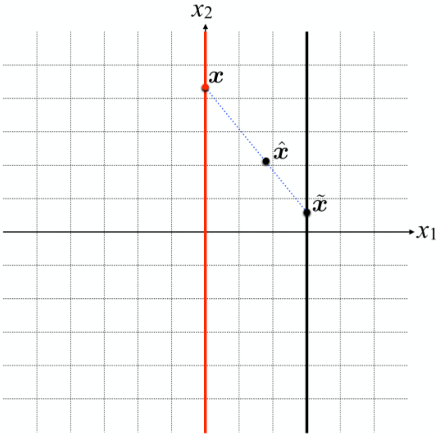
- Property of the optimal WGAN critic $|f(x) - f(y)| \leq |x - y|$
- Optimal WGAN critic has gradient norm 1
- Add a regularization term to the WGAN value function
- Sampling along straight lines

$$\epsilon \sim U\left[0, 1\right], \boldsymbol{x} \sim \mathbb{P}_g, \tilde{\boldsymbol{x}} \sim \mathbb{P}_r$$

$$\hat{\boldsymbol{x}} = t\boldsymbol{x} + (1 - t)\,\tilde{\boldsymbol{x}}$$

- Gradient penalty

$$\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})$$



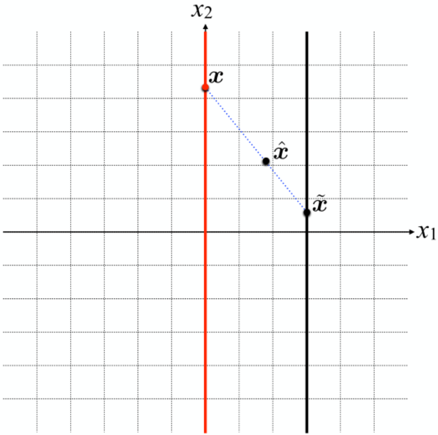
- Property of the optimal WGAN critic $|f(x) - f(y)| \leq |x - y|$
- Optimal WGAN critic has gradient norm 1
- Add a regularization term to the WGAN value function
- Sampling along straight lines

$$\epsilon \sim U[0,1], \boldsymbol{x} \sim \mathbb{P}_g, \tilde{\boldsymbol{x}} \sim \mathbb{P}_r$$

$$\hat{\boldsymbol{x}} = t\boldsymbol{x} + (1-t)\tilde{\boldsymbol{x}}$$

- Gradient penalty

$$\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2$$



Method #2 - Gradient Penalty

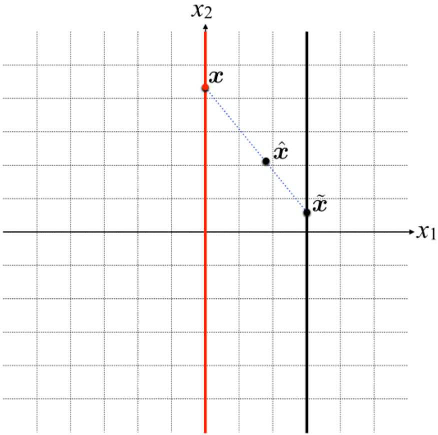
- Property of the optimal WGAN critic $|f(x) - f(y)| \leq |x - y|$
- Optimal WGAN critic has gradient norm 1
- Add a regularization term to the WGAN value function
- Sampling along straight lines

$$\epsilon \sim U[0,1], \boldsymbol{x} \sim \mathbb{P}_g, \tilde{\boldsymbol{x}} \sim \mathbb{P}_r$$

$$\hat{\boldsymbol{x}} = t\boldsymbol{x} + (1-t)\tilde{\boldsymbol{x}}$$

- Gradient penalty

$$\mathbb{E}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}} \left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2 \right]$$



Short history of training GANs - March 2017

Improved Training of Wasserstein GANs

Ishaan Gulrajani[1], **Faruk Ahmed**[1], **Martin Arjovsky**[2], **Vincent Dumoulin**[1], **Aaron Courville**[1,3]
[1] Montreal Institute for Learning Algorithms
[2] Courant Institute of Mathematical Sciences
[3] CIFAR Fellow
igul222@gmail.com
{faruk.ahmed,vincent.dumoulin,aaron.courville}@umontreal.ca
ma4371@nyu.edu

WGAN-GP Value function

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_r}[D(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z} \sim \mathcal{Z}}[D(G(\boldsymbol{z}))]$$

Short history of training GANs - March 2017

Improved Training of Wasserstein GANs

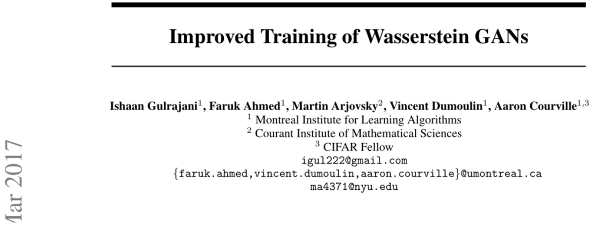
Ishaan Gulrajani[1], **Faruk Ahmed**[1], **Martin Arjovsky**[2], **Vincent Dumoulin**[1], **Aaron Courville**[1,3]
[1] Montreal Institute for Learning Algorithms
[2] Courant Institute of Mathematical Sciences
[3] CIFAR Fellow
igul222@gmail.com
{faruk.ahmed,vincent.dumoulin,aaron.courville}@umontreal.ca
ma4371@nyu.edu

WGAN-GP Value function

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}[D(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z}\sim\mathcal{Z}}[D(G(\boldsymbol{z}))] + \lambda \mathbb{E}_{\hat{\boldsymbol{x}}\sim\mathbb{P}_{\hat{\boldsymbol{x}}}}\left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2\right]$$

- Enforcing the Lipschitz constraint with a gradient penalty regularization term

Short history of training GANs - March 2017



Improved Training of Wasserstein GANs

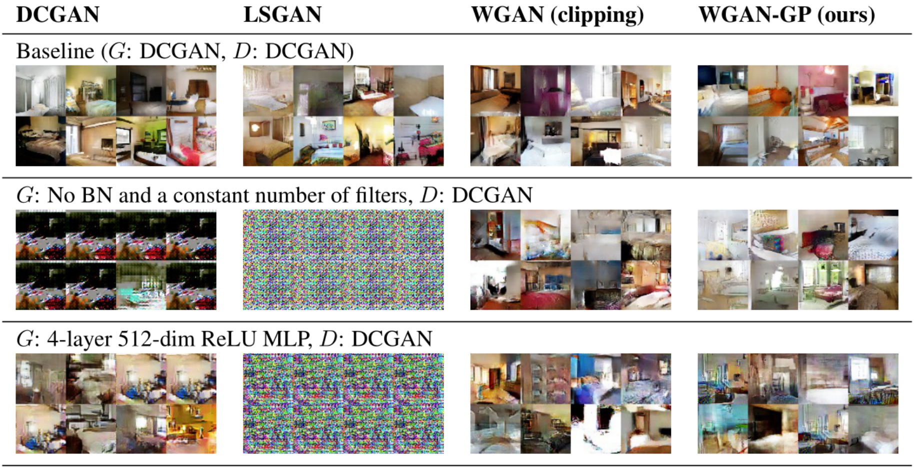
Ishaan Gulrajani[1], **Faruk Ahmed**[1], **Martin Arjovsky**[2], **Vincent Dumoulin**[1], **Aaron Courville**[1,3]
[1] Montreal Institute for Learning Algorithms
[2] Courant Institute of Mathematical Sciences
[3] CIFAR Fellow
igul222@gmail.com
{faruk.ahmed,vincent.dumoulin,aaron.courville}@umontreal.ca
ma4371@nyu.edu

WGAN-GP Value function

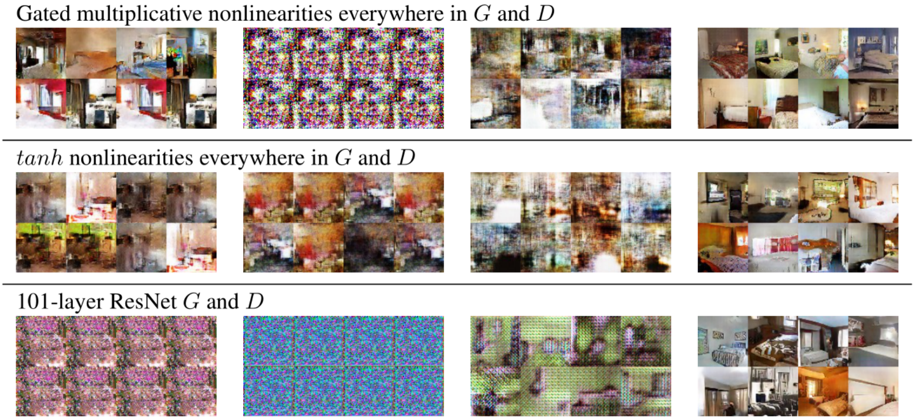
$$\min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}[D(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z}\sim\mathcal{Z}}[D(G(\boldsymbol{z}))] + \lambda \mathbb{E}_{\hat{\boldsymbol{x}}\sim\mathbb{P}_{\hat{\boldsymbol{x}}}}\left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2\right]$$

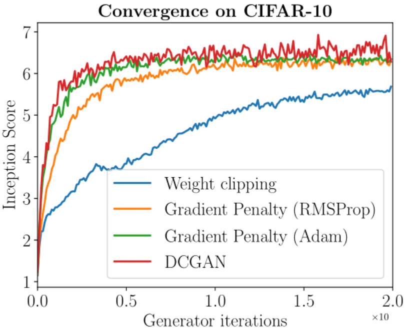
- Enforcing the Lipschitz constraint with a gradient penalty regularization term
- Improvements?

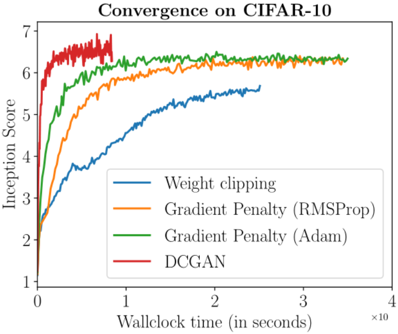
WGAN-GP results



WGAN-GP results







- DCGAN converges faster
- Significantly outperforms weight clipping
- **Robust to changes in model architecture**

Game theory perspective

- Two neural networks are playing a zero-sum game

Game theory perspective

- Two neural networks are playing a zero-sum game
- Nash equilibrium

Game theory perspective

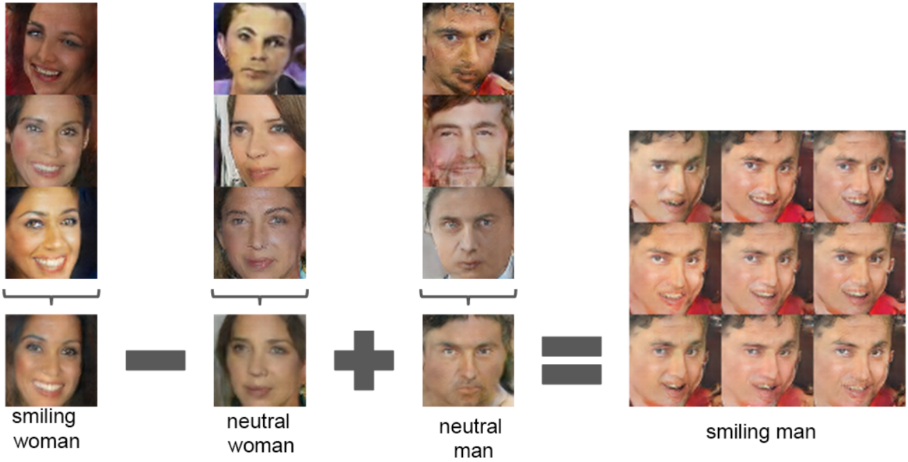
- Two neural networks are playing a zero-sum game
- Nash equilibrium
- Goal - both players adopting strategies where any deviations would be disadvantageous

Game theory perspective

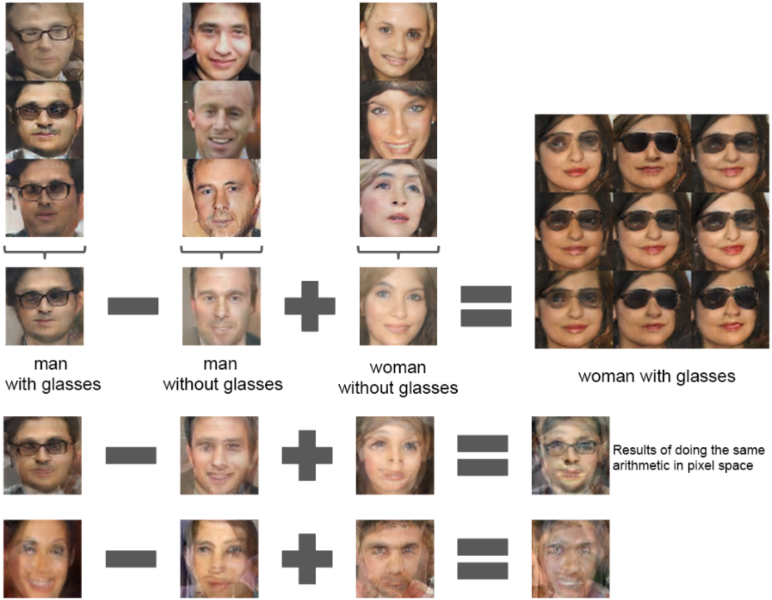
- Two neural networks are playing a zero-sum game
- Nash equilibrium
- Goal - both players adopting strategies where any deviations would be disadvantageous
- Lack of theoretical insights

Cool things you can do with GANs

Latent space arithmetic

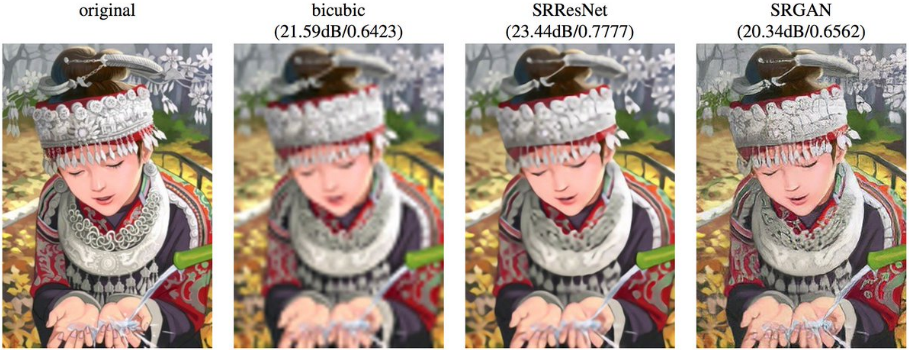


Latent space arithmetic

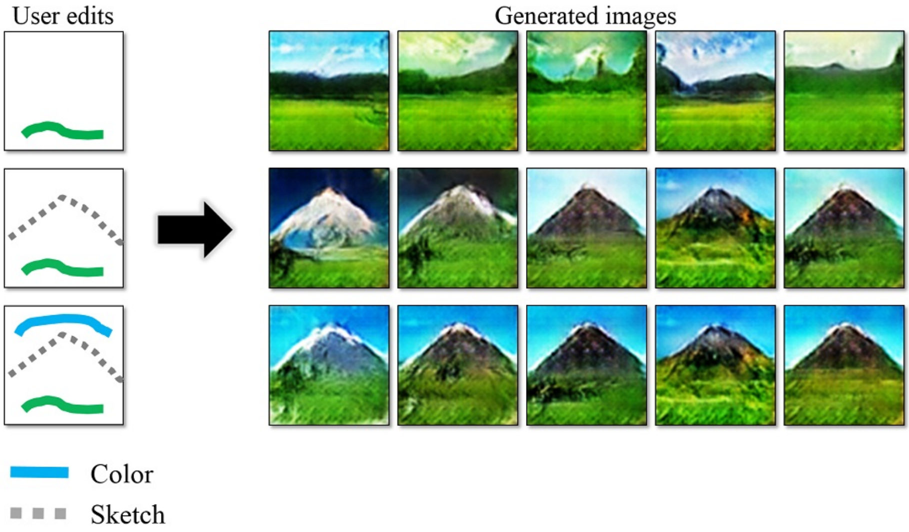


Interpolation between images

Super-Resolution



Interactive GAN

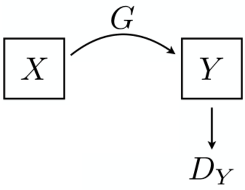


▸ Interactive GAN

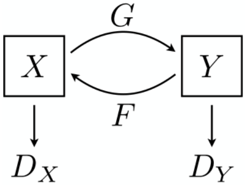
Style Transfer

$\boxed{X}$ $\qquad\qquad$ $\boxed{Y}$

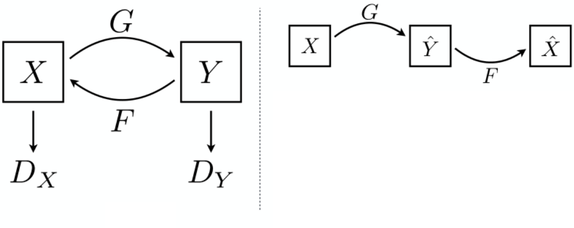
Style Transfer



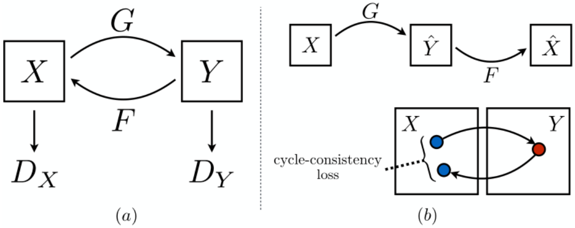
Style Transfer



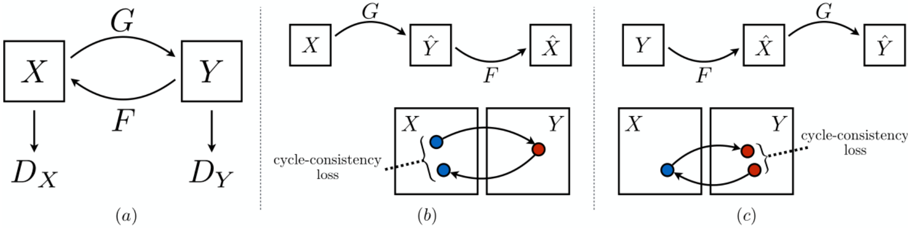
Style Transfer



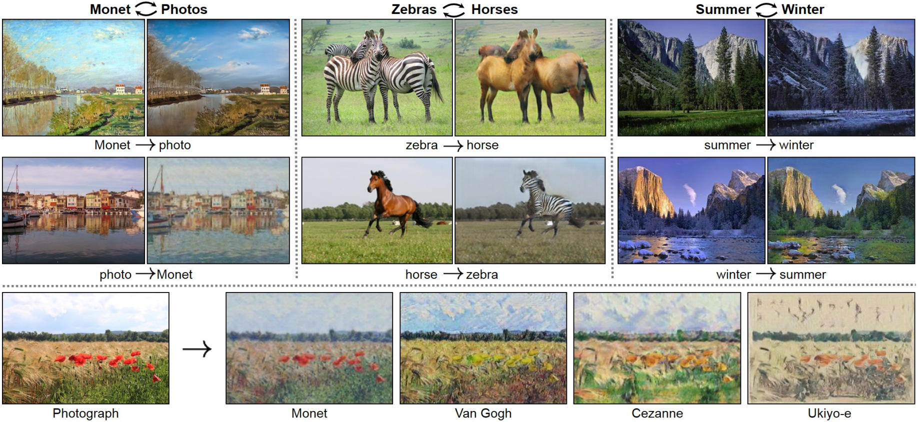
Style Transfer



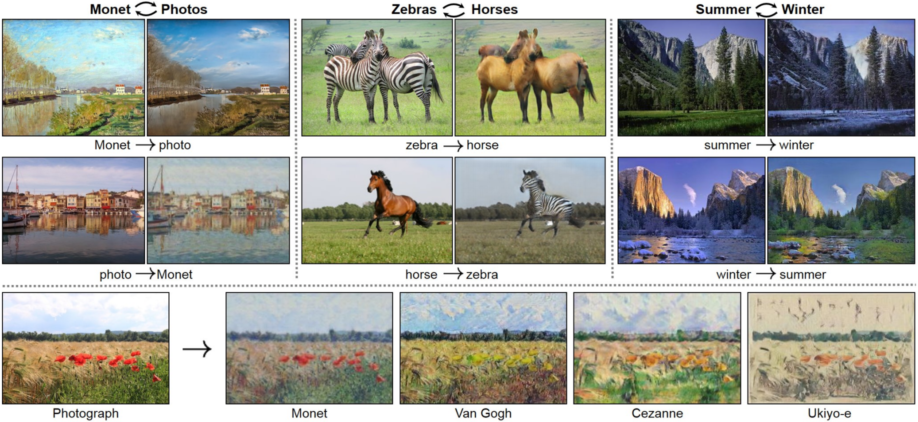
Style Transfer



Style Transfer



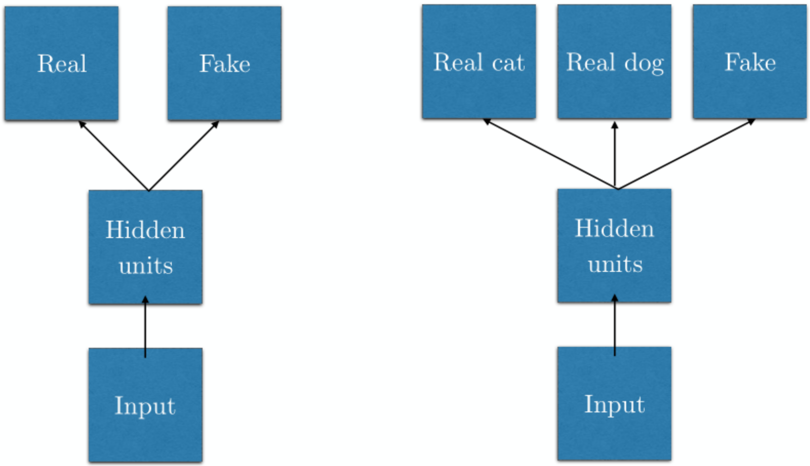
Style Transfer



- CycleGAN in action
- Creepy CycleGAN

Failure case

Supervised discriminator



Heuristics for training WGANs

- Make sure your critic is "ahead of" the generator!
- 5-10x more than the generator reduces oscillatory behaviour
- Should work without any hyperparameter tuning

Problems with GANs

- How to evaluate GANs?

Problems with GANs

- How to evaluate GANs?
- No way to tell how good of an approximation of EM distance the critic really is

Problems with GANs

- How to evaluate GANs?
- No way to tell how good of an approximation of EM distance the critic really is
- Mode collapse

Problems with GANs

- How to evaluate GANs?
- No way to tell how good of an approximation of EM distance the critic really is
- Mode collapse
- Multi-task solving?

Problems with GANs

- How to evaluate GANs?
- No way to tell how good of an approximation of EM distance the critic really is
- Mode collapse
- Multi-task solving?
- Problems with counting

Problems with GANs

- How to evaluate GANs?
- No way to tell how good of an approximation of EM distance the critic really is
- Mode collapse
- Multi-task solving?
- Problems with counting
- Perspective

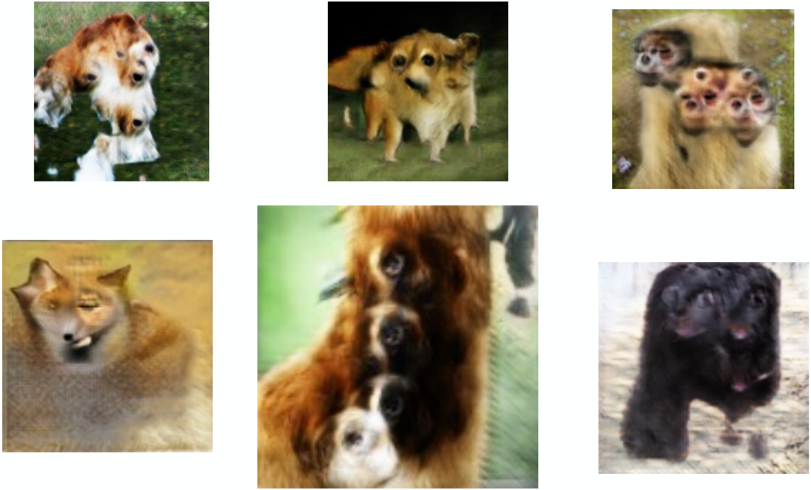
Problems with GANs

- How to evaluate GANs?
- No way to tell how good of an approximation of EM distance the critic really is
- Mode collapse
- Multi-task solving?
- Problems with counting
- Perspective
- High level structure

Problems with GANs

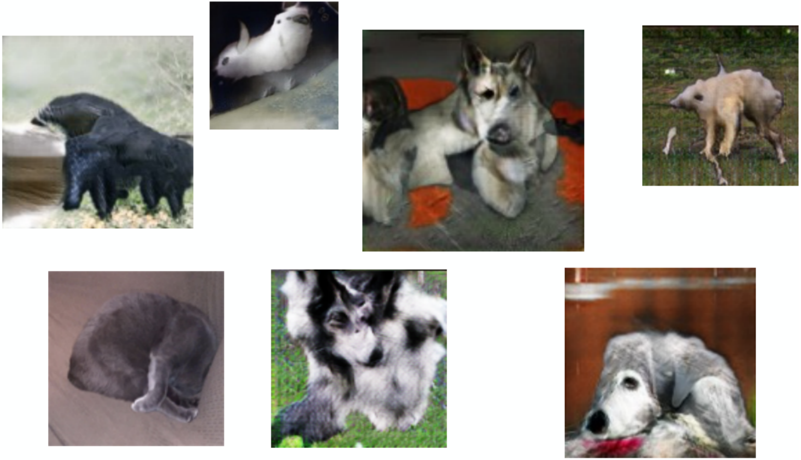
- How to evaluate GANs?
- No way to tell how good of an approximation of EM distance the critic really is
- Mode collapse
- Multi-task solving?
- Problems with counting
- Perspective
- High level structure
- Sequential data

Counting



[2] Ian J. Goodfellow: NIPS 2016 Tutorial: Generative Adversarial Networks

Perspective



[2] Ian J. Goodfellow: NIPS 2016 Tutorial: Generative Adversarial Networks

Future of GANs

- Which statistical distance to use?

Future of GANs

- Which statistical distance to use?
- Best training regime?

Future of GANs

- Which statistical distance to use?
- Best training regime?
- Convergence conditions?

Future of GANs

- Which statistical distance to use?
- Best training regime?
- Convergence conditions?
- Solving mode collapse?

Future of GANs

- Which statistical distance to use?
- Best training regime?
- Convergence conditions?
- Solving mode collapse?
- Understanding models with multiple optimization criteria?

Future of GANs

- Which statistical distance to use?
- Best training regime?
- Convergence conditions?
- Solving mode collapse?
- Understanding models with multiple optimization criteria?

- Sepp Hochreiter - proof of GAN convergence

Future of GANs

- Which statistical distance to use?
- Best training regime?
- Convergence conditions?
- Solving mode collapse?
- Understanding models with multiple optimization criteria?

- Sepp Hochreiter - proof of GAN convergence
- *Multiplayer* GAN?

Future of GANs

- Which statistical distance to use?
- Best training regime?
- Convergence conditions?
- Solving mode collapse?
- Understanding models with multiple optimization criteria?

- Sepp Hochreiter - proof of GAN convergence
- *Multiplayer* GAN?
- Exchanging discriminators?

Future of GANs

- Which statistical distance to use?
- Best training regime?
- Convergence conditions?
- Solving mode collapse?
- Understanding models with multiple optimization criteria?

- Sepp Hochreiter - proof of GAN convergence
- *Multiplayer* GAN?
- Exchanging discriminators?
- Cramer GAN - new statistical distance?

Future of GANs

- Which statistical distance to use?
- Best training regime?
- Convergence conditions?
- Solving mode collapse?
- Understanding models with multiple optimization criteria?

- Sepp Hochreiter - proof of GAN convergence
- *Multiplayer* GAN?
- Exchanging discriminators?
- Cramer GAN - new statistical distance?
- Many, many more papers...

History of great ideas in deep learning

History of great ideas in deep learning

- Need for feature engineering before applying ML model!

History of great ideas in deep learning

- Need for feature engineering before applying ML model!
- **Oh, neural networks can do that.** Too bad we still have to figure out which cost function to use...

History of great ideas in deep learning

- Need for feature engineering before applying ML model!
- **Oh, neural networks can do that.** Too bad we still have to figure out which cost function to use...
- **Oh, GANs can figure out the cost function by themselves!** Too bad we still have to find the correct way to do the optimization...

History of great ideas in deep learning

- Need for feature engineering before applying ML model!
- **Oh, neural networks can do that.** Too bad we still have to figure out which cost function to use...
- **Oh, GANs can figure out the cost function by themselves!** Too bad we still have to find the correct way to do the optimization...
- **Oh, the neural network can optimize itself...?**

History of great ideas in deep learning

- Need for feature engineering before applying ML model!
- **Oh, neural networks can do that.** Too bad we still have to figure out which cost function to use...
- **Oh, GANs can figure out the cost function by themselves!** Too bad we still have to find the correct way to do the optimization...
- **Oh, the neural network can optimize itself...?**
- Learning to learn

Thank you!