

# Generative Adversarial Networks

Bruno Gavranović

Petnica Summer School of Machine Learning

*[bruno.gavranovic@fer.hr](mailto:bruno.gavranovic@fer.hr)*

July 19, 2017

# Overview

1 Meta

2 Prerequisites

- The manifold hypothesis

3 Comparison with autoencoders

- Computational graph perspective
- Training regime

4 Statistical distances

- Comparison of various statistical distances
- Comparison on toy example
- Short history of training GANs
- Tricks for training GANs

5 Game theory perspective

6 GANs - cool stuff

7 Problems and open questions

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.



- What is this talk about?



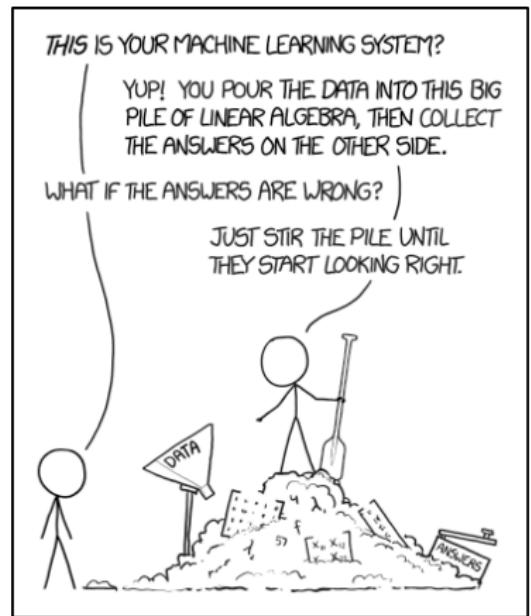
- What is this talk about?
- New idea



- What is this talk about?
- New idea
- Nobody is still sure how it works



- What is this talk about?
- New idea
- Nobody is still sure how it works
- Compared to other ML models,  
we're still in early stages



- What is this talk about?
- New idea
- 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



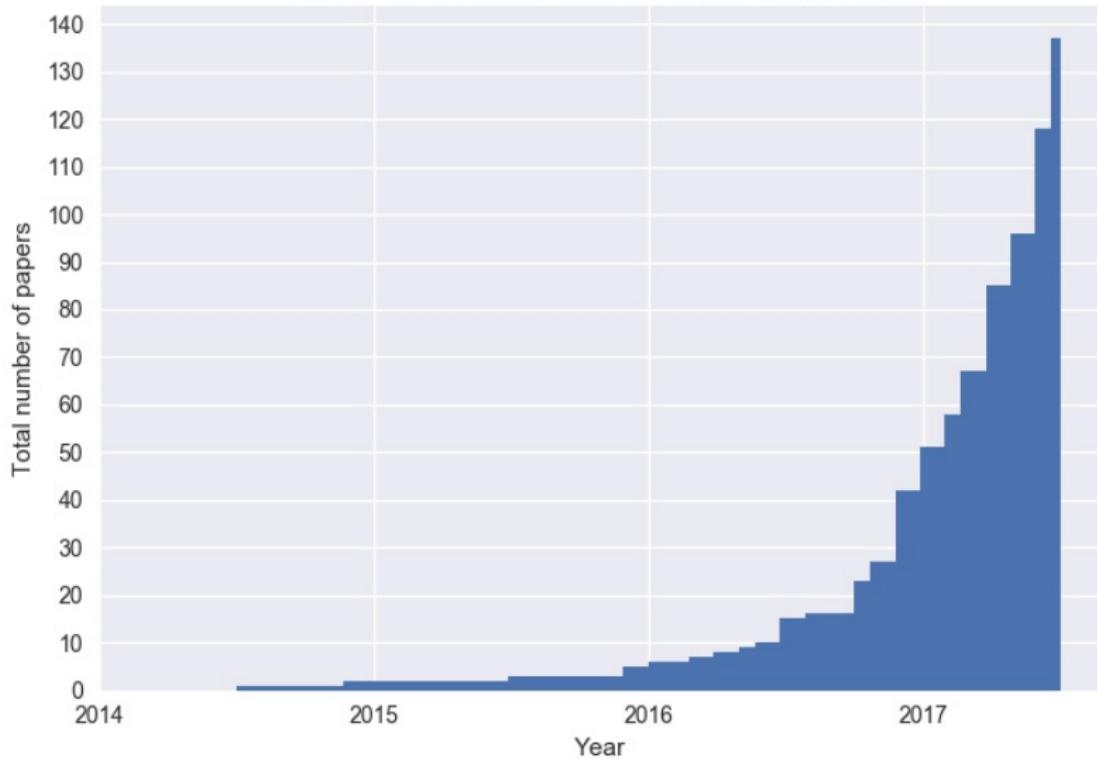
- What is this talk about?
- New idea
- Nobody is still sure how it works
- Compared to other ML models, we're still in early stages
- Our understanding of it changes from week to week



## Yann LeCun

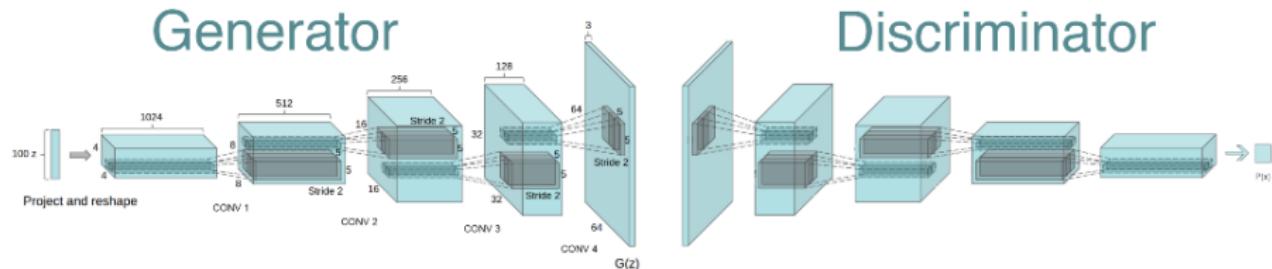
"The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks). This, and the variations that are now being proposed is the most interesting idea in the last 10 years in ML, in my opinion."

Cumulative number of named GAN papers by month



# GAN: Just tell me what it is

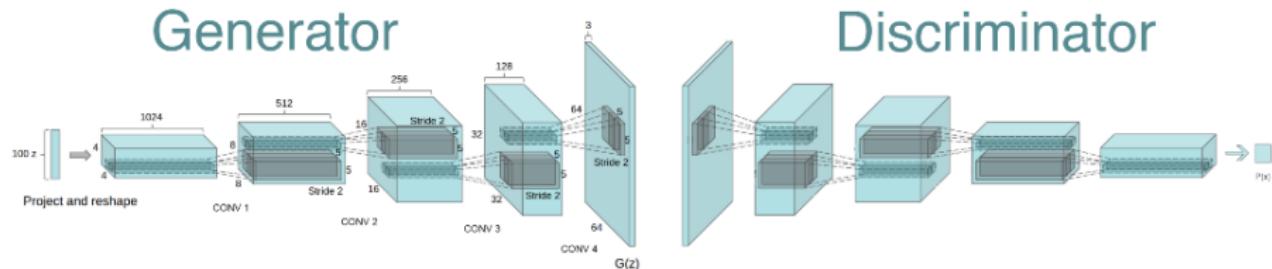
- Generative machine learning model in which two neural networks are competing against each other



<sup>0</sup><https://github.com/dmonn/GAN-face-generator>

# GAN: Just tell me what it is

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

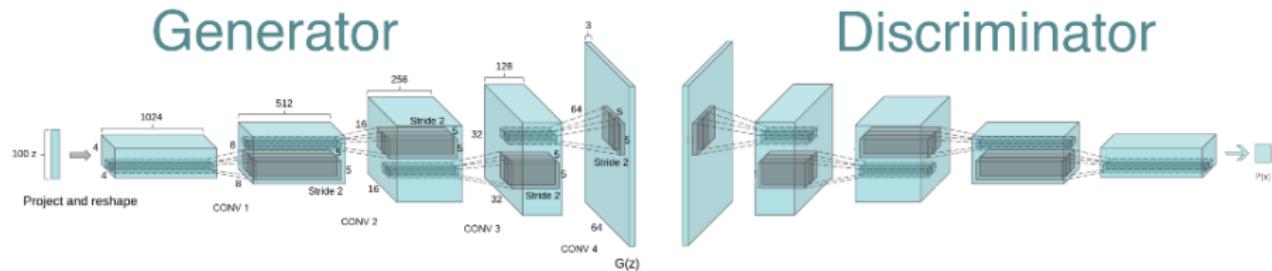


- Instead of using a standard fixed cost function, we *learn* the cost function with the neural network

<sup>0</sup><https://github.com/dmonn/GAN-face-generator>

# GAN: Just tell me what it is

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

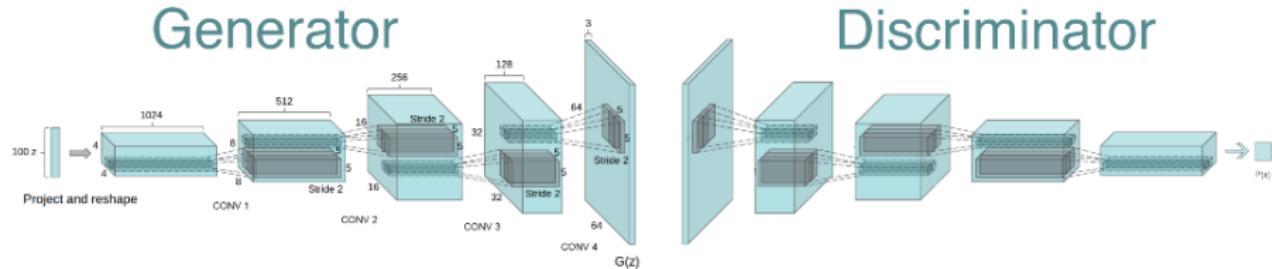


- Instead of using a standard fixed cost function, we *learn* the cost function with the neural network
- We alternate between training different parts of the network

<sup>0</sup><https://github.com/dmonn/GAN-face-generator>

# GAN: Just tell me what it is

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

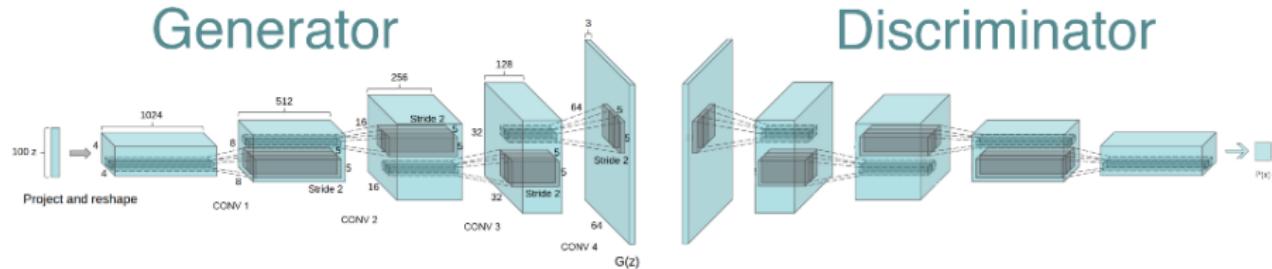


- Instead of using a standard fixed cost function, we *learn* the cost function with the neural network
- We alternate between training different parts of the network
- Forger and the police

<sup>0</sup><https://github.com/dmonn/GAN-face-generator>

# GAN: Just tell me what it is

- 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

<sup>0</sup><https://github.com/dmonn/GAN-face-generator>



Figure: Which side are real images?

	This bird is blue with white and has a very short beak	This bird has wings that are brown and has a yellow belly	A white bird with a black crown and yellow beak	This bird is white, black, and brown in color, with a brown beak	The bird has small beak, with reddish brown crown and gray belly	This is a small, black bird with a white breast and white on the wingbars.	This bird is white black and yellow in color, with a short black beak
Text description							
Stage-I images	   				  		
Stage-II images	   				  		

Figure: StackGAN





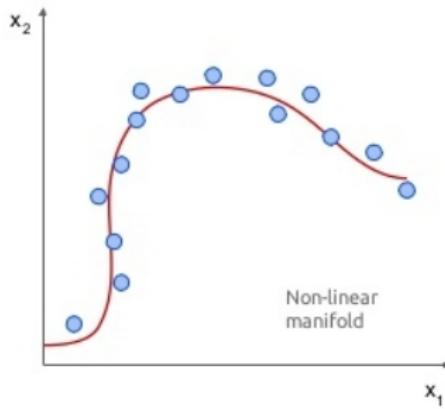
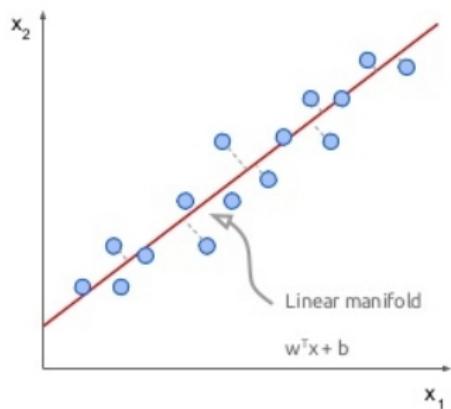


# What is this talk about?

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

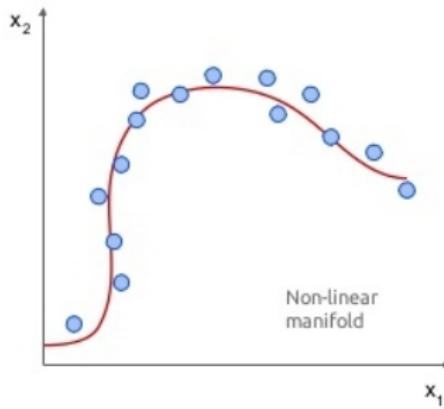
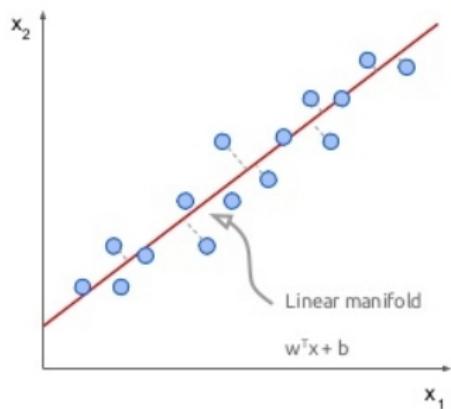
# Prerequisites

## The manifold hypothesis



# Prerequisites

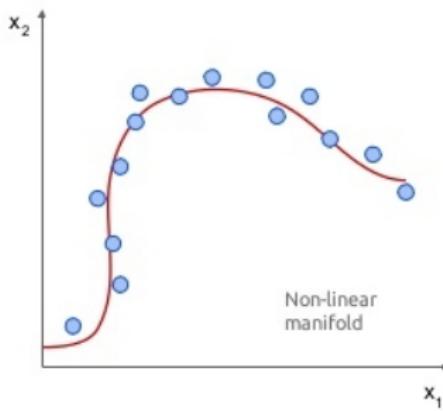
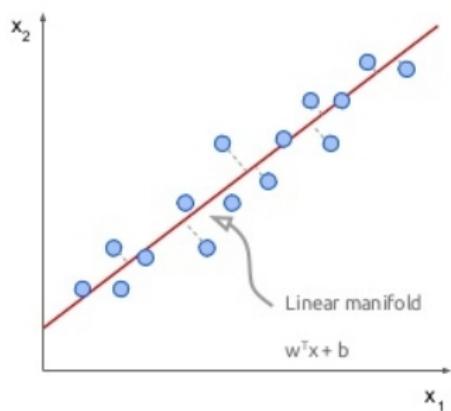
## The manifold hypothesis



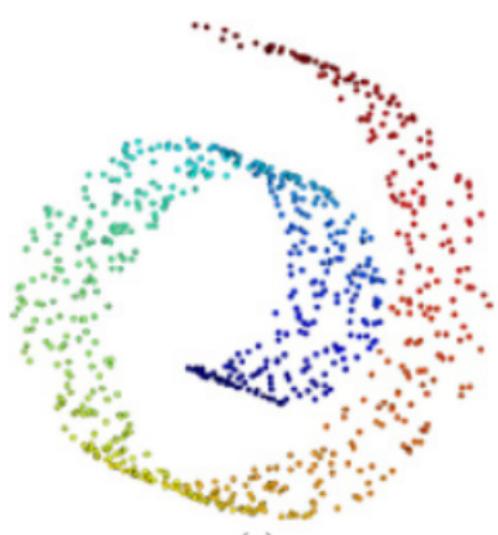
- Understanding real world data

# Prerequisites

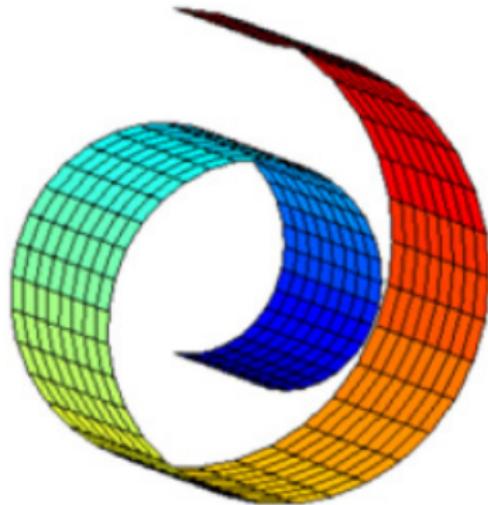
## The manifold hypothesis



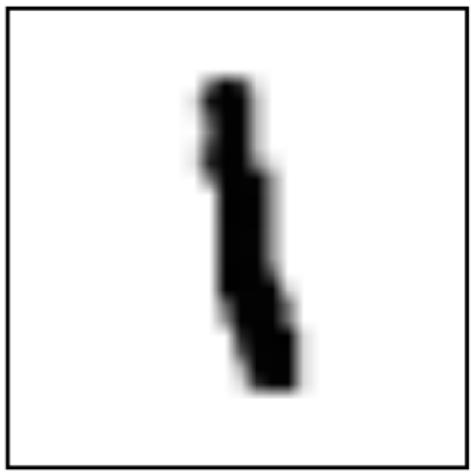
- Understanding real world data
- Natural data forms a low dimensional manifold in its embedding space



(a)

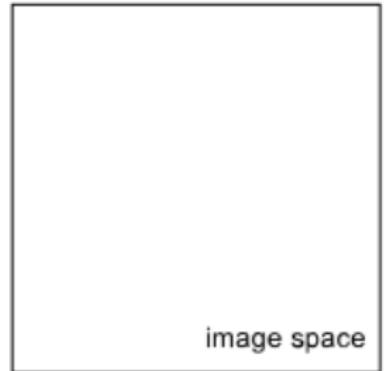


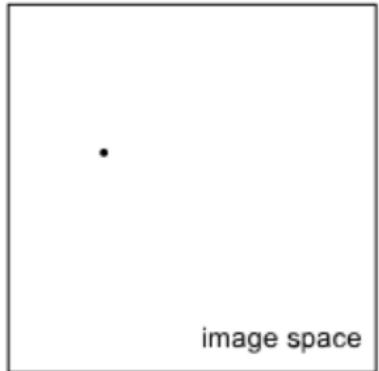
(b)

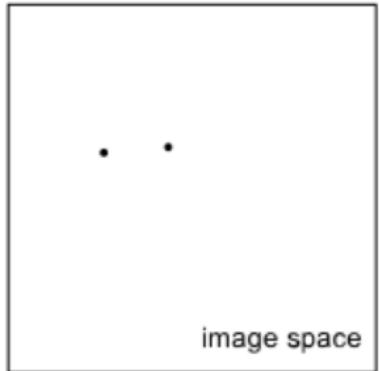


$\approx$

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	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	0	0	0	0	0	0	.7	.1	0	0	0	0	0	0	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	0	0	0	0	0	0	.5	.1	.4	0	0	0	0	0	0	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	0	0	0	0	0	0	.1	.4	0	0	0	0	0	0	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	0	0	0	0	0	0	.1	.1	0	0	0	0	0	0	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	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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

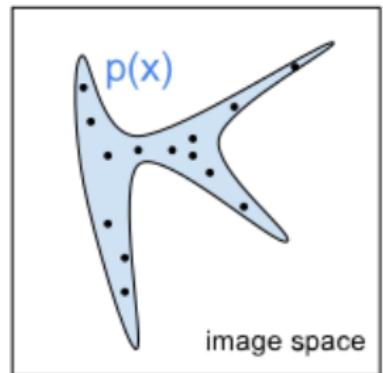




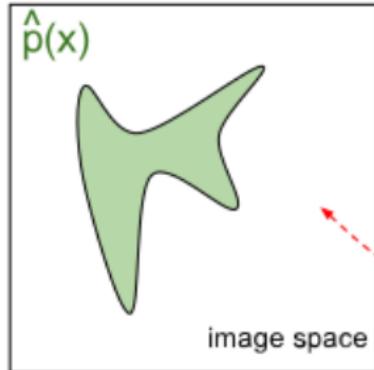




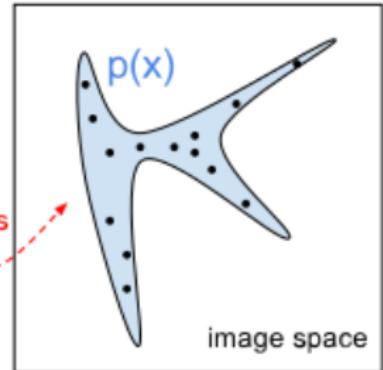
true data distribution



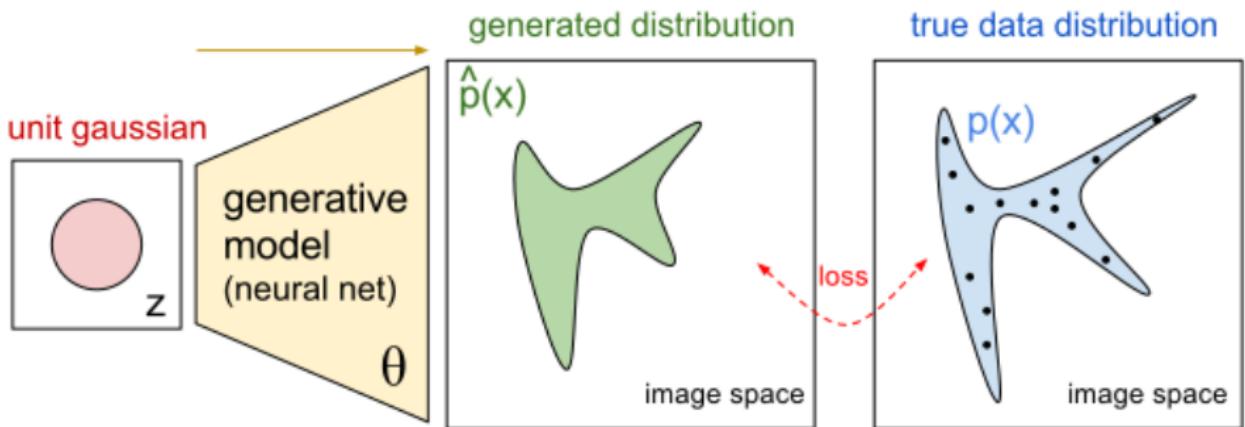
generated distribution



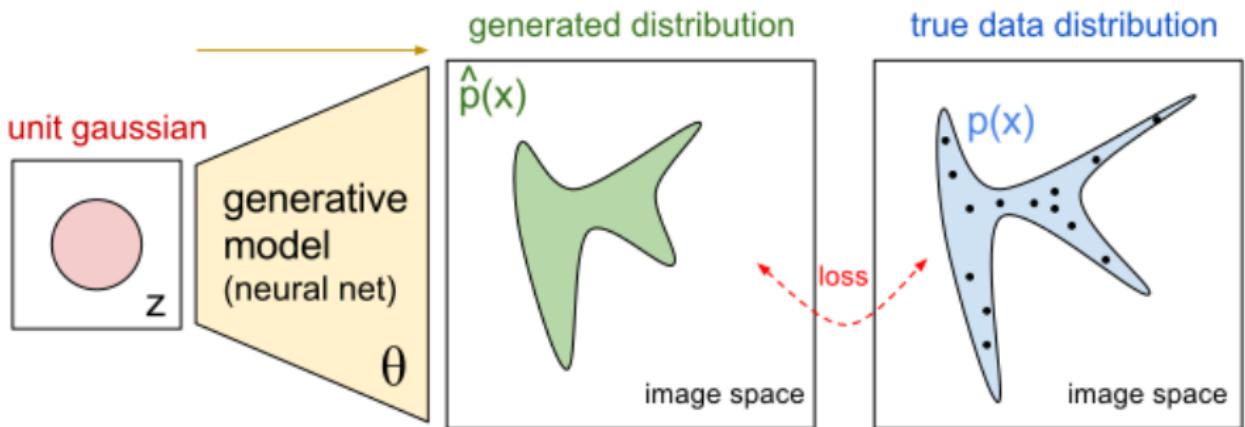
true data distribution



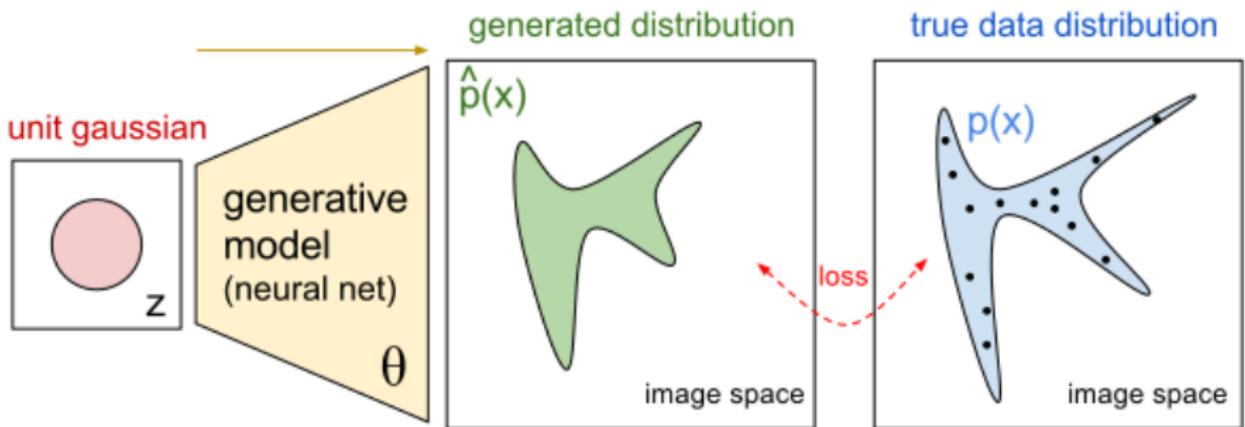
loss



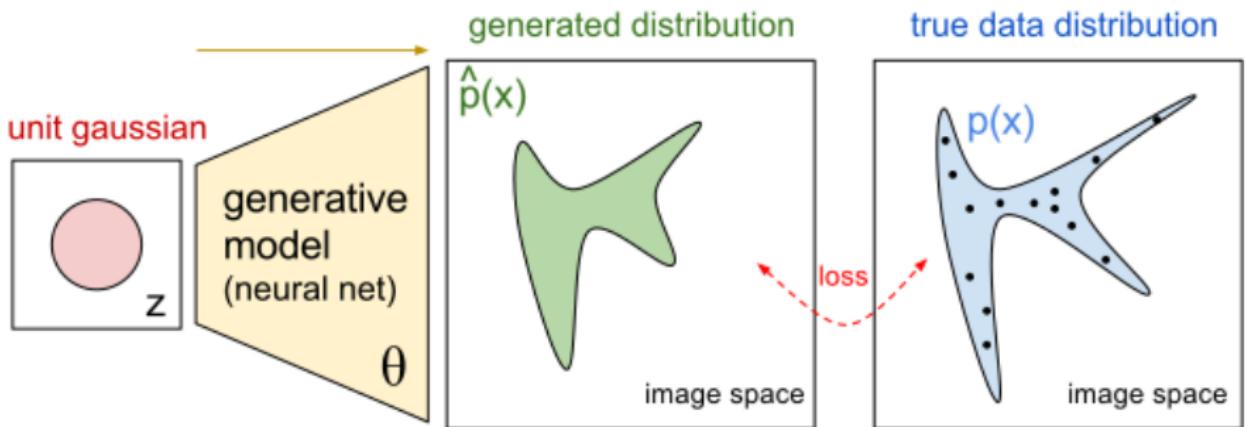
- We have a random vector  $z \sim \mathcal{Z}$



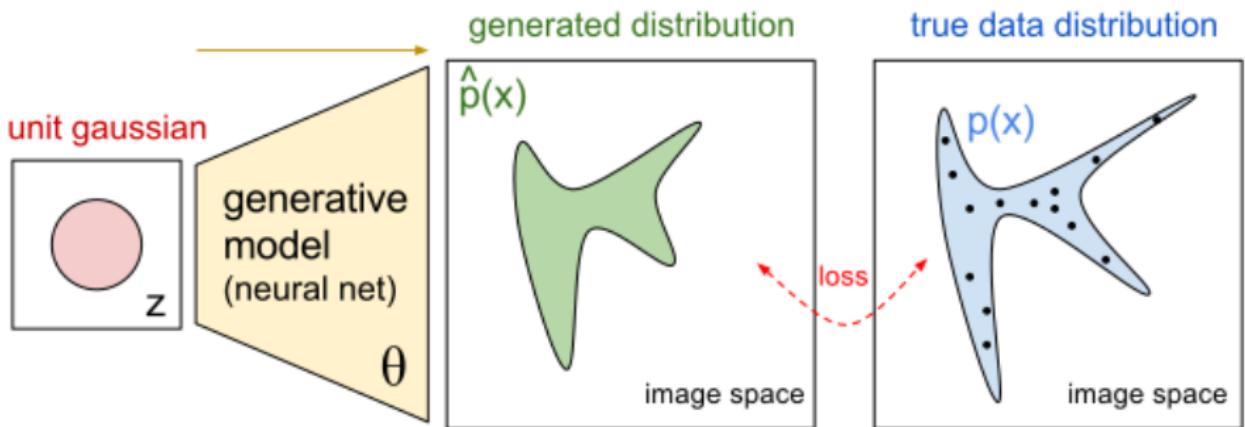
- We have a random vector  $z \sim \mathcal{Z}$
- We can define a parametric function  $g_\theta : \mathcal{Z} \rightarrow \mathcal{X}$  that generates samples from a certain distribution  $\mathbb{P}_\theta$



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



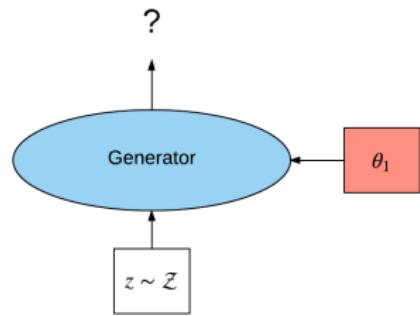
- We have a random vector  $z \sim \mathcal{Z}$
- We can define a parametric function  $g_\theta : \mathcal{Z} \rightarrow \mathcal{X}$  that generates samples from a certain distribution  $\mathbb{P}_\theta$
- We have samples from the real data distribution  $\mathbb{P}_r$
- By varying  $\theta$  we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

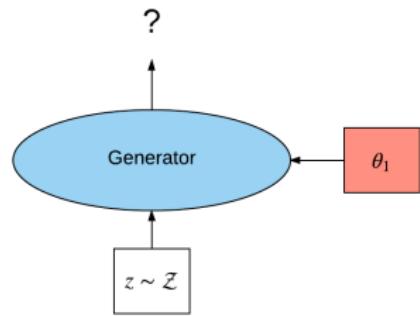
# First attempt

- How to match the distributions?



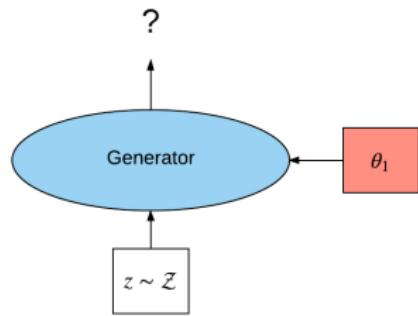
# First attempt

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



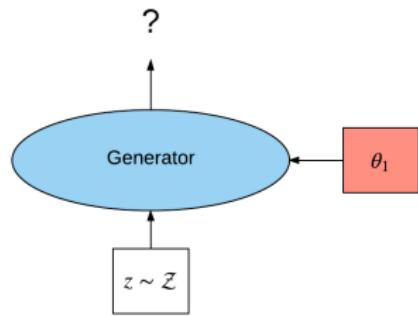
# First attempt

- How to match the distributions?
- Autoencoder as an example
- Network decomposition into basic building blocks



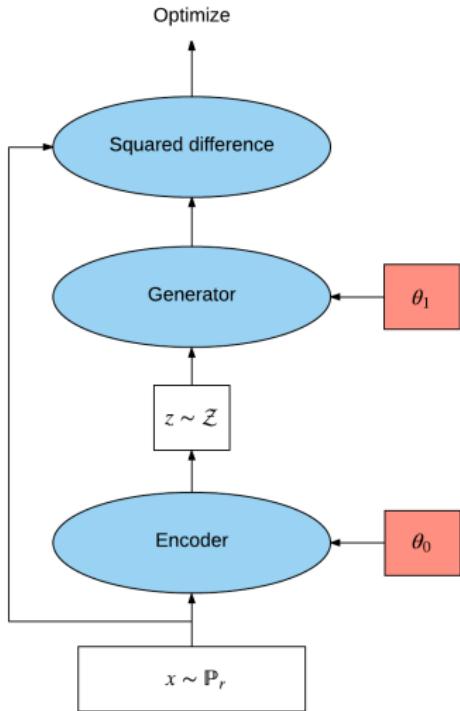
# First attempt

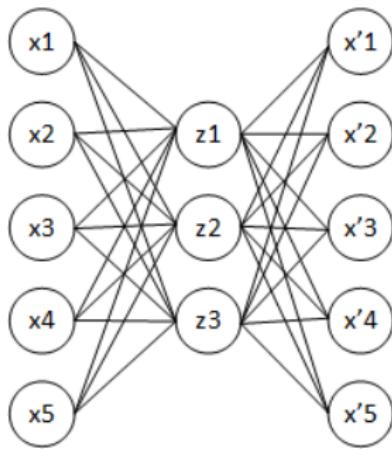
- How to match the distributions?
- Autoencoder as an example
- Network decomposition into basic building blocks
- Tools for thinking about neural networks

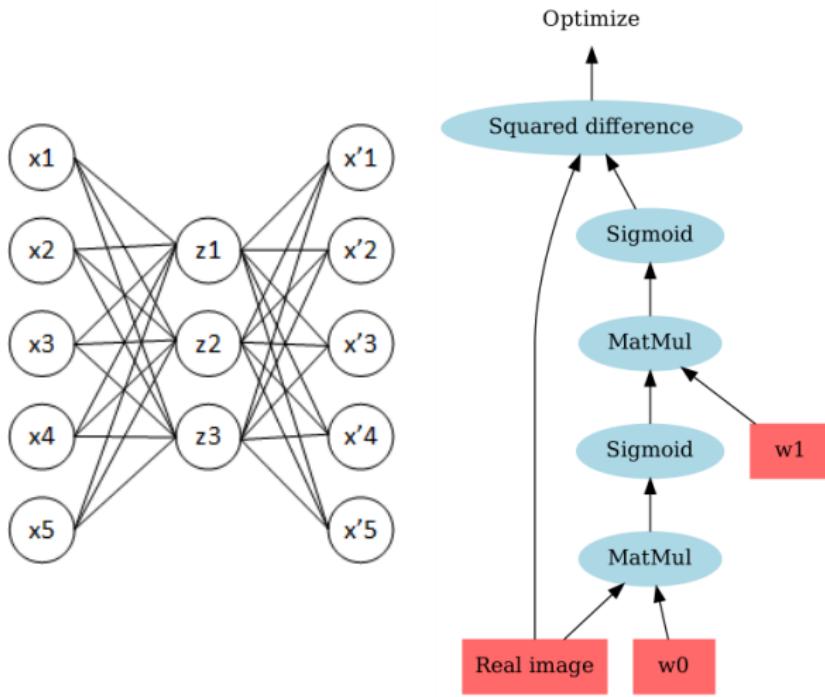


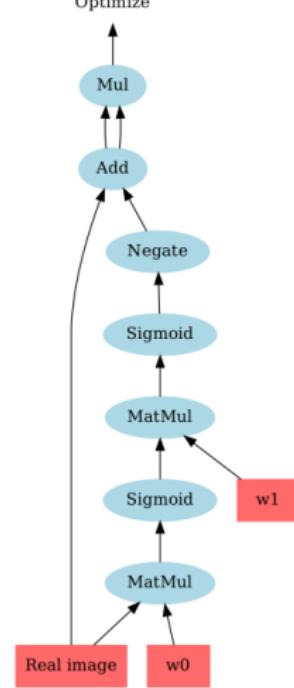
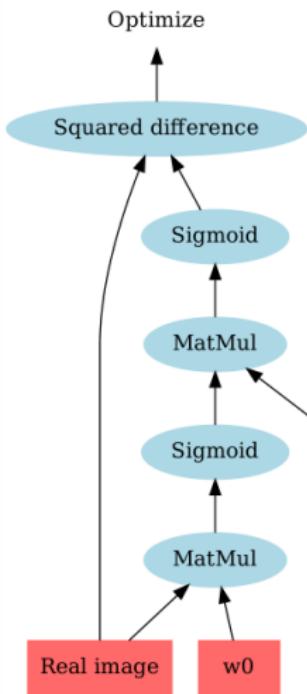
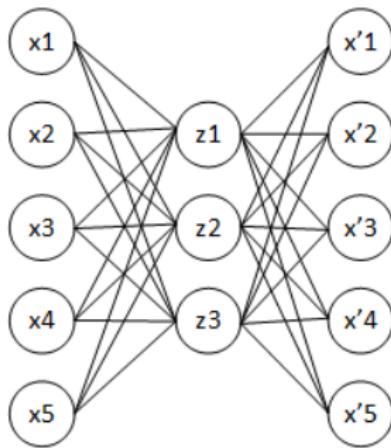
# Autoencoder

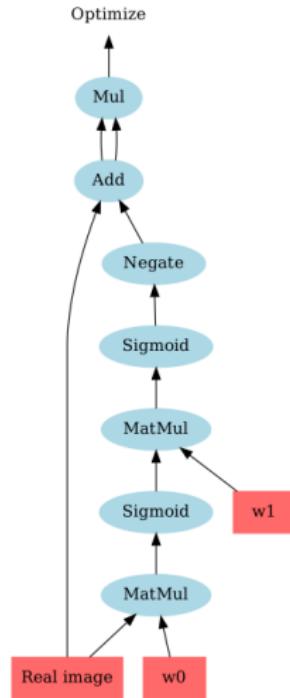
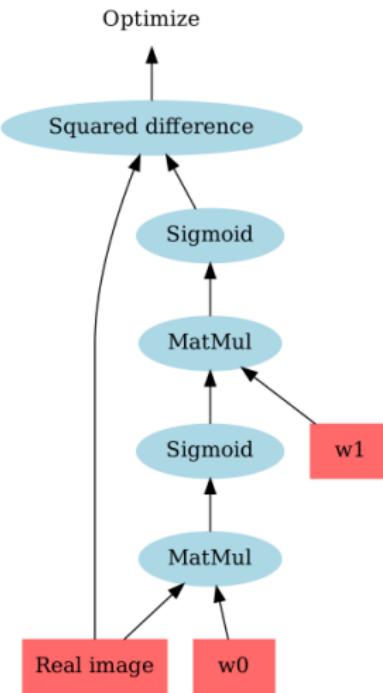
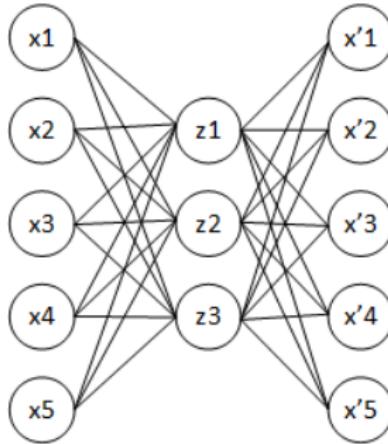
- Recreation of samples from the real distribution
- Encoding the images into the bottleneck  $\theta$
- Implicitly training the network to extract useful features









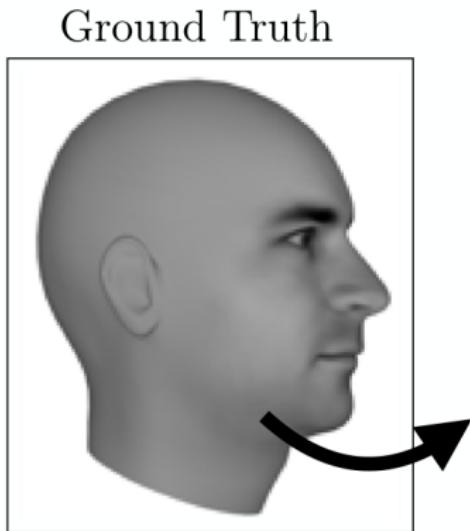


## Things to note

- ① Cost function is not parametrized (it's fixed)
- ② We train the network with just one cost function
- ③ 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?

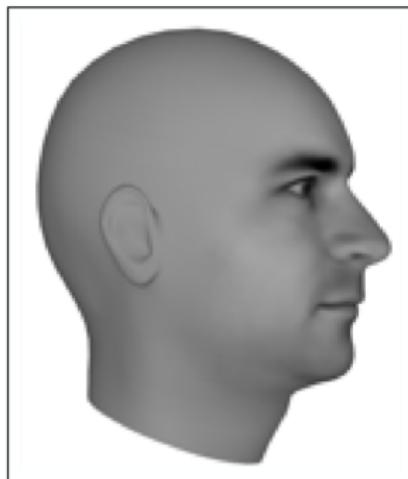


<sup>0</sup>Ian J. Goodfellow: NIPS 2016 Tutorial: Generative Adversarial Networks

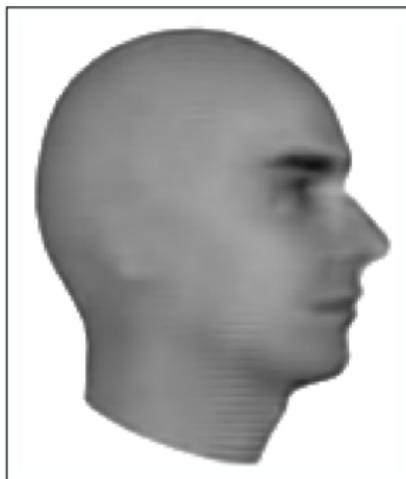
# Cost function

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

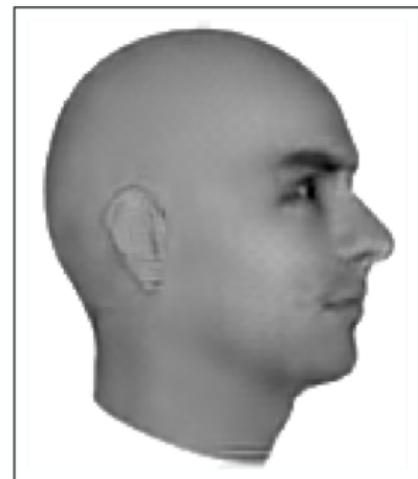
Ground Truth



MSE



Adversarial



<sup>0</sup>Ian J. Goodfellow: NIPS 2016 Tutorial: Generative Adversarial Networks

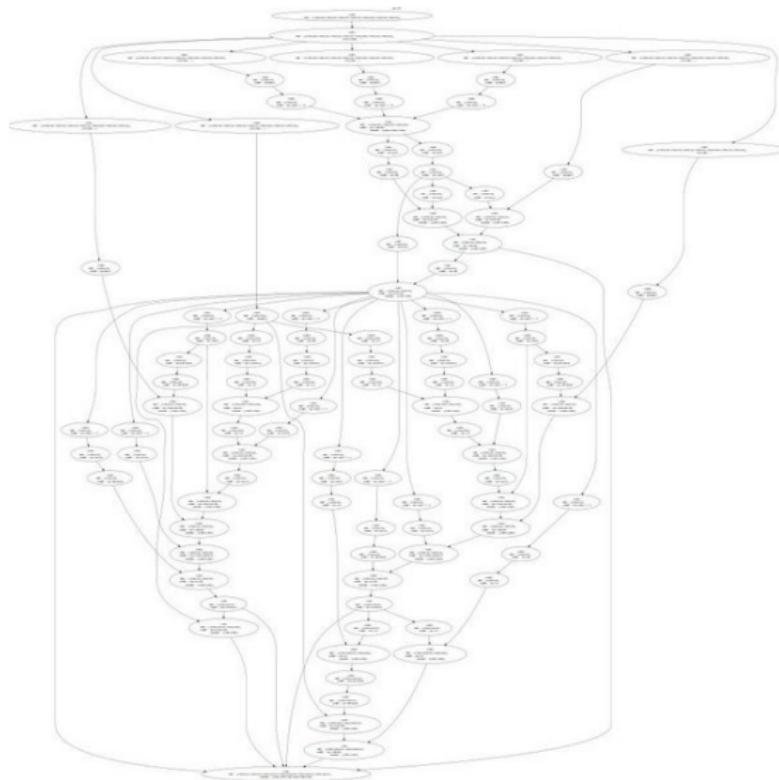


Figure: Neural turing machines computational graph

# 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?

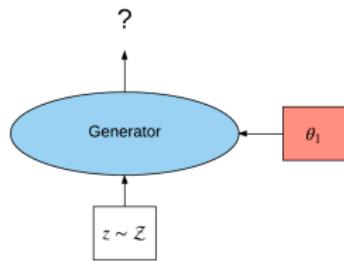
# The great idea

- 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 its behaviour?

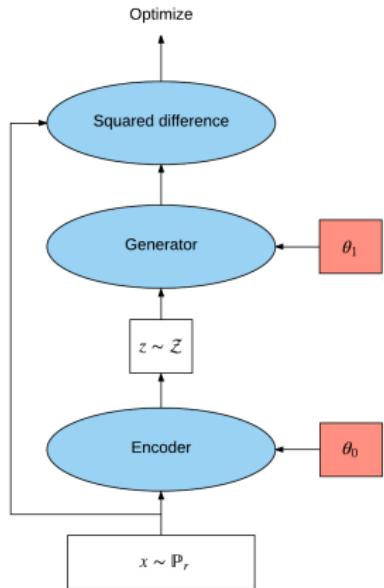
# The great idea

- 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 its behaviour?
- Partial answer to the "How do we know it's close?" question from the beginning

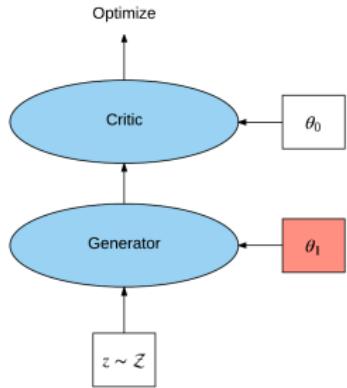
# Training regime



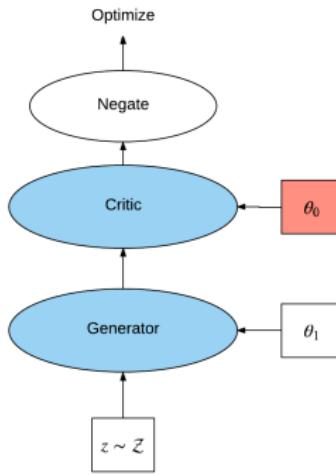
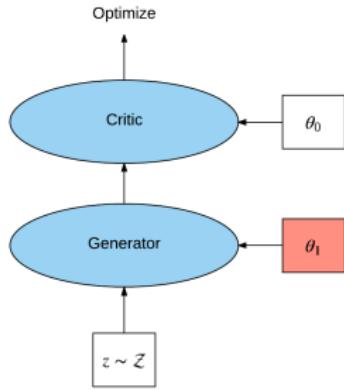
# Training regime



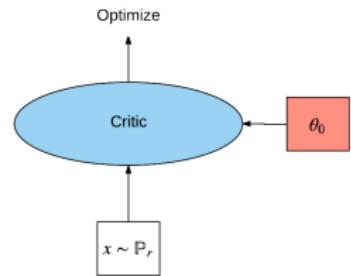
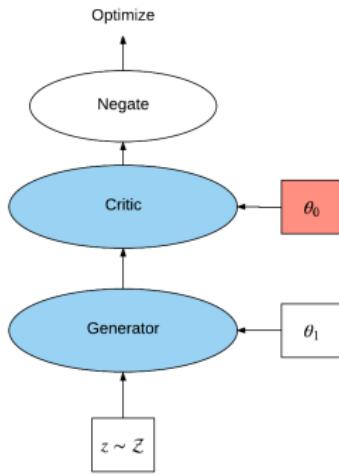
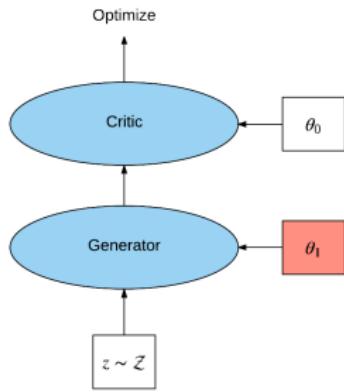
# Training regime



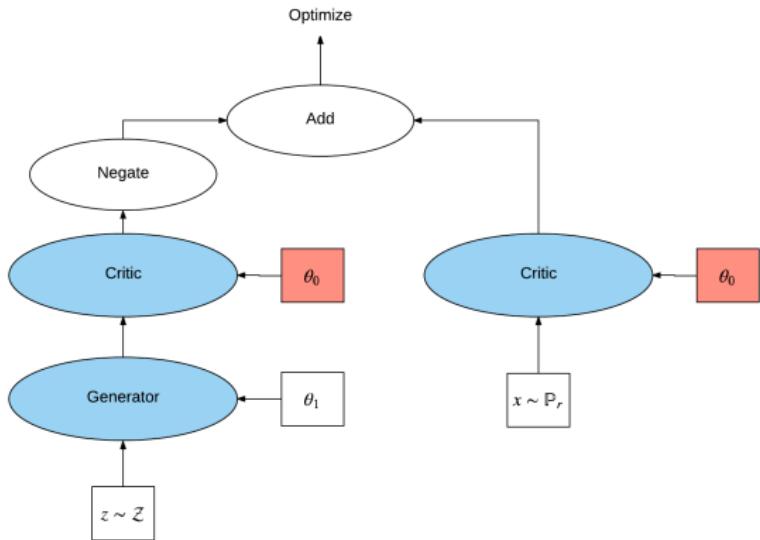
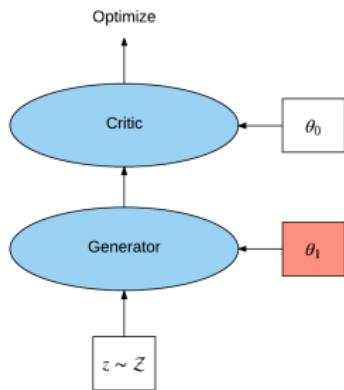
# Training regime



# Training regime



# Training regime



- Two-step optimization optimization process

# Intuitive approach

- Forger and the police

# Intuitive approach

- Forger and the police
- Forger tries to create real looking money

# Intuitive approach

- Forger and the police
- Forger tries to create real looking money
- Police tries to get better at detecting fake money

# Intuitive approach

- Forger and the police
- Forger tries to create real looking money
- Police tries to get better at detecting fake money
- They both keep improving until the difference between fake and real money is imperceptible to us

# 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
- ...

# 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

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

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

# Wasserstein (Earth mover) distance

## EM distance

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

- 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

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

- 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

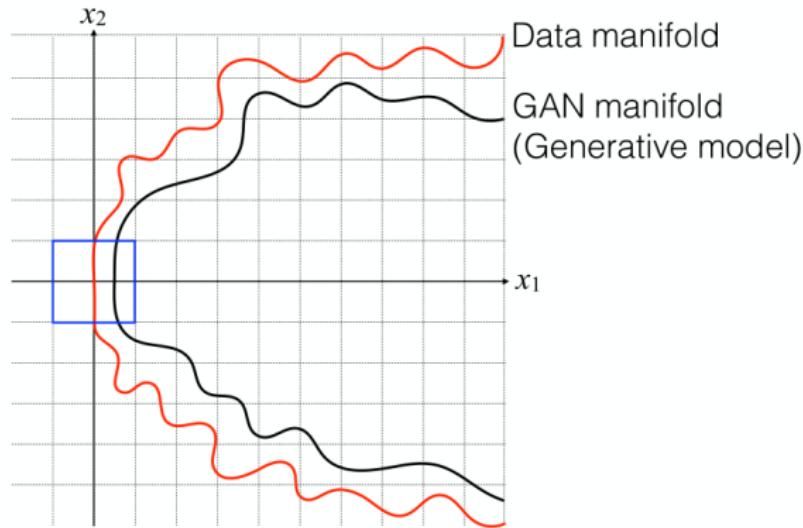
# Wasserstein (Earth mover) distance

## EM distance

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

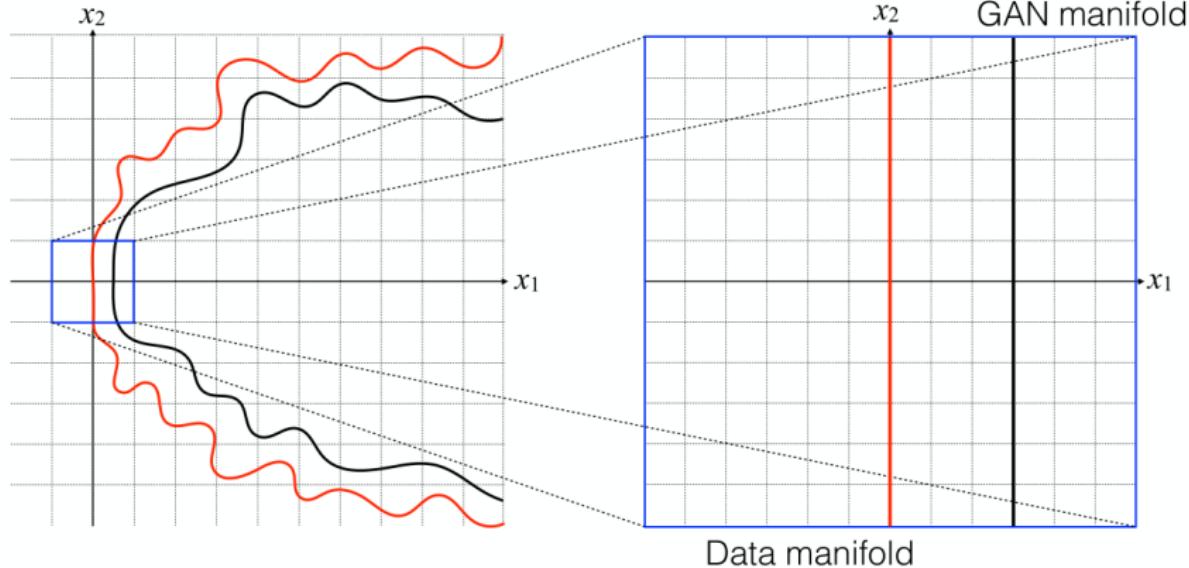
- Distance function that takes into account underlying geometry of the distributions
- Minimum cost of turning a pile of dirt into the other
- Proposed by Arjovsky *et al.* as improvement to original GAN training - Wasserstein GAN
- Intractable

# Learning distributions - toy example



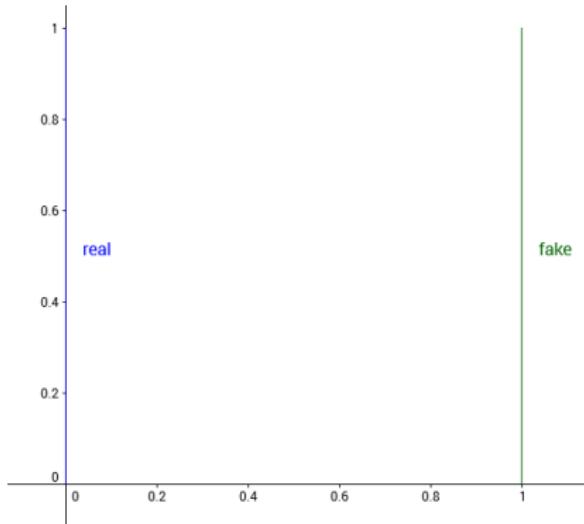
<sup>0</sup>MILA DLSS: [https://drive.google.com/file/d/0B\\_wzP\\_J1VFcKQ21udGpTSkh0aVk/view](https://drive.google.com/file/d/0B_wzP_J1VFcKQ21udGpTSkh0aVk/view)

# Learning distributions - toy example



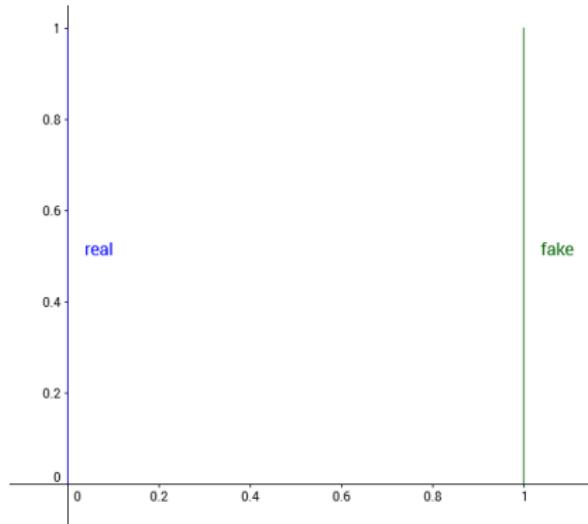
<sup>0</sup>MILA DLSS: [https://drive.google.com/file/d/0B\\_wzP\\_J1VFcKQ21udGpTSkh0aVk/view](https://drive.google.com/file/d/0B_wzP_J1VFcKQ21udGpTSkh0aVk/view)

# Learning distributions - toy example



<sup>0</sup><http://www.alexirpan.com/2017/02/22/wasserstein-gan.html>

# Learning distributions - toy example

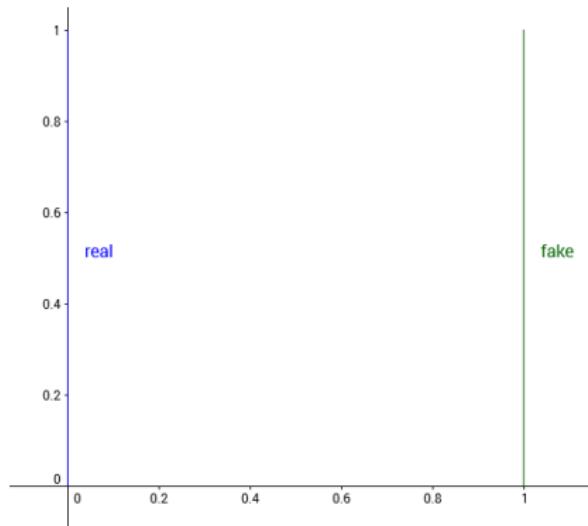


$$\mathbb{P}_r = (0, y)$$

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

<sup>0</sup><http://www.alexirpan.com/2017/02/22/wasserstein-gan.html>

# Learning distributions - toy example



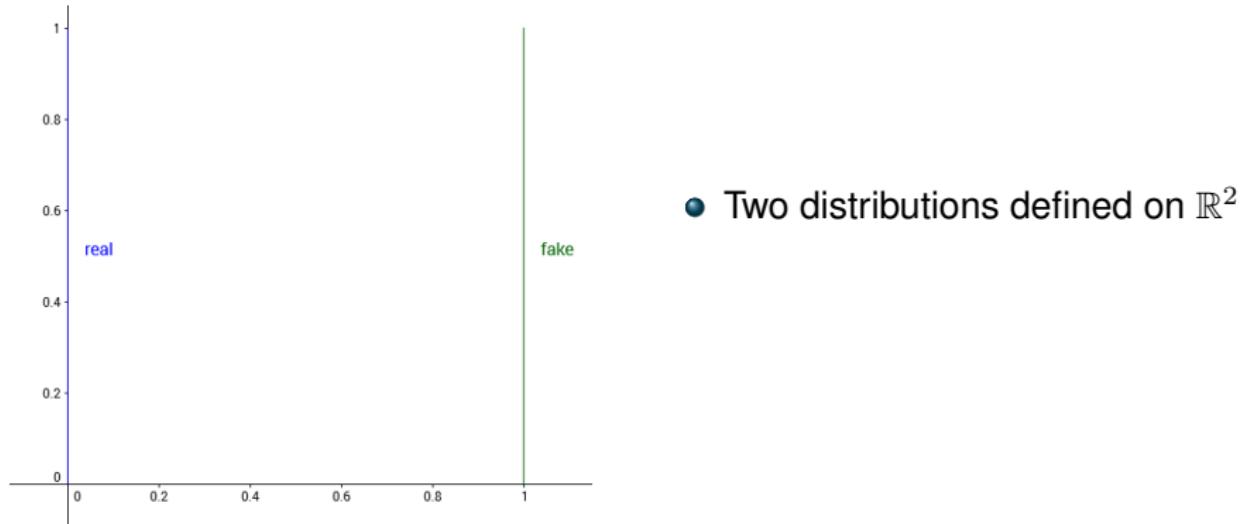
$$\mathbb{P}_r = (0, y)$$

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

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

<sup>0</sup><http://www.alexirpan.com/2017/02/22/wasserstein-gan.html>

# Learning distributions - toy example



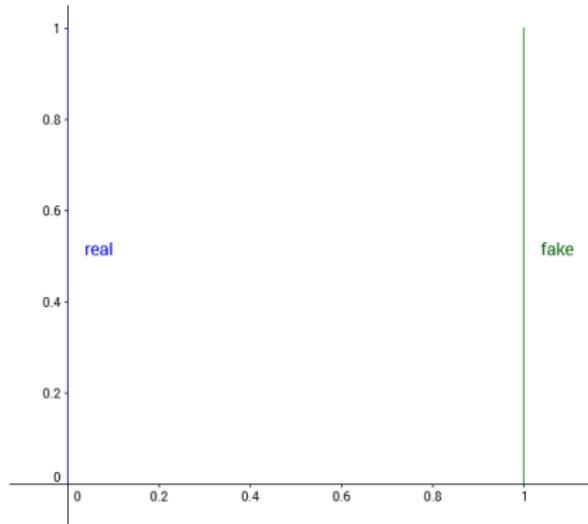
$$\mathbb{P}_r = (0, y)$$

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

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

<sup>0</sup><http://www.alexirpan.com/2017/02/22/wasserstein-gan.html>

# Learning distributions - toy example



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

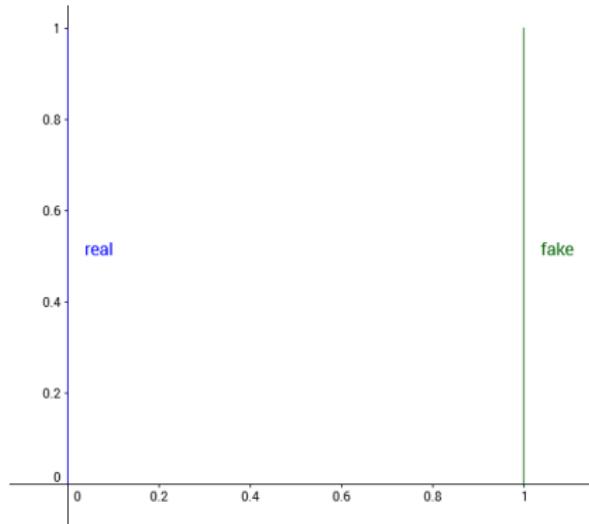
$$\mathbb{P}_r = (0, y)$$

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

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

<sup>0</sup><http://www.alexirpan.com/2017/02/22/wasserstein-gan.html>

# Learning distributions - toy example



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

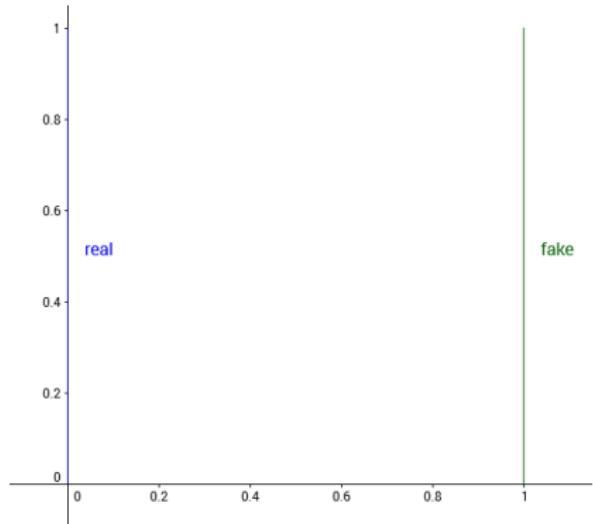
$$\mathbb{P}_r = (0, y)$$

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

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

<sup>0</sup><http://www.alexirpan.com/2017/02/22/wasserstein-gan.html>

# Learning distributions - toy example



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

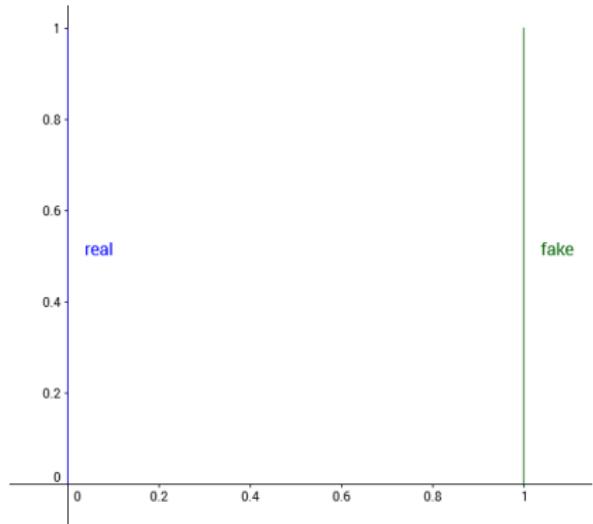
$$\mathbb{P}_r = (0, y)$$

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

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

<sup>0</sup><http://www.alexirpan.com/2017/02/22/wasserstein-gan.html>

# Learning distributions - toy example



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

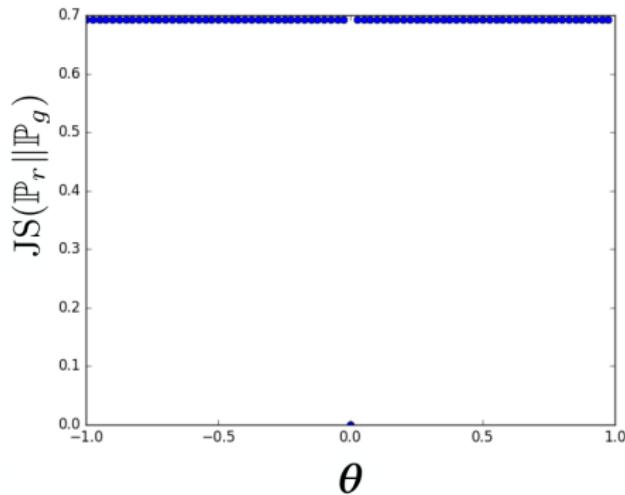
$$\mathbb{P}_g = (\theta, y)$$

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

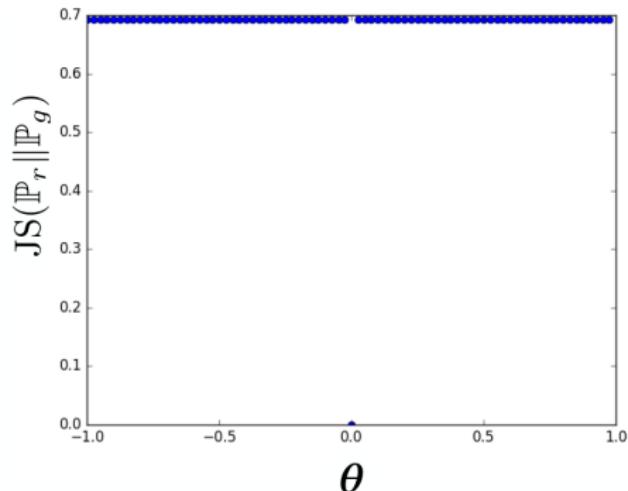
- Two distributions defined on  $\mathbb{R}^2$
- $KL(\mathbb{P}_r || \mathbb{P}_g) = +\infty$
- $KL(\mathbb{P}_g || \mathbb{P}_r) = +\infty$
- $JS(\mathbb{P}_r, \mathbb{P}_g) = JS(\mathbb{P}_g, \mathbb{P}_r) = \log 2$
- $W(\mathbb{P}_r, \mathbb{P}_g) = W(\mathbb{P}_g, \mathbb{P}_r) = |\theta|$

<sup>0</sup><http://www.alexirpan.com/2017/02/22/wasserstein-gan.html>

## JS and EM distance w.r.t. $\theta$

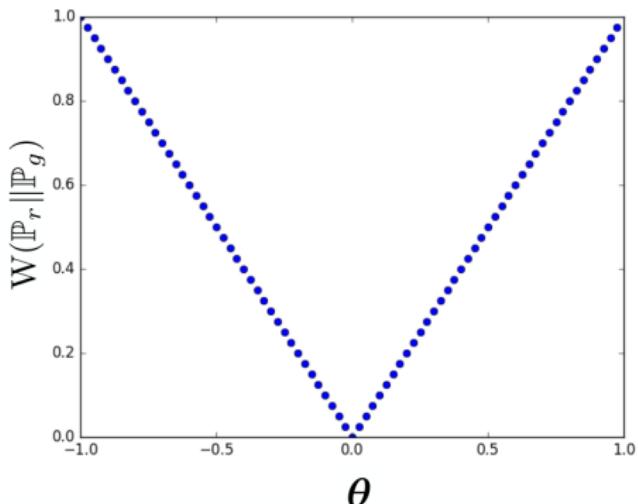
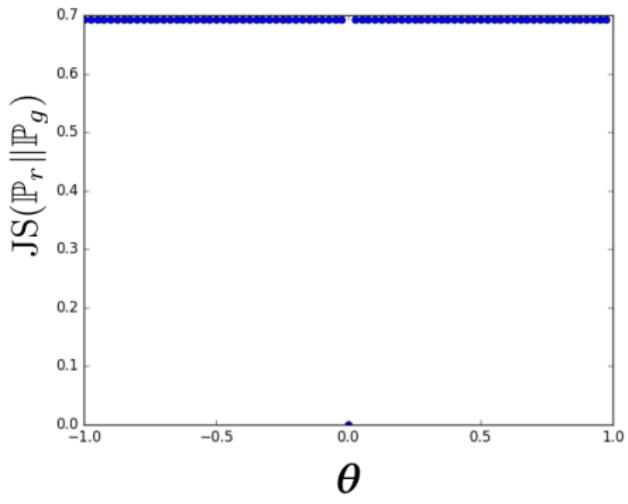


## JS and EM distance w.r.t. $\theta$



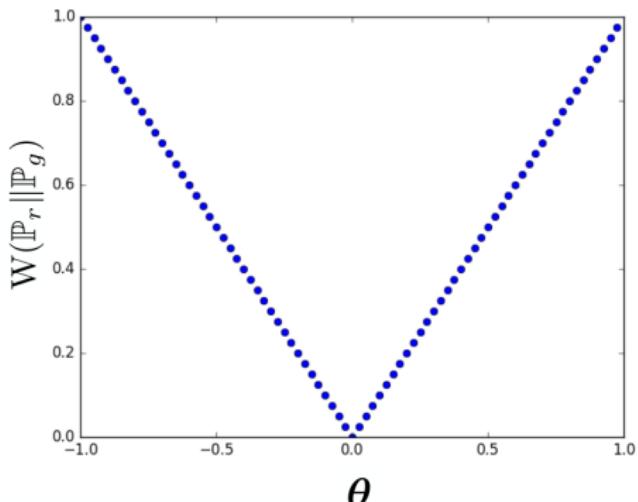
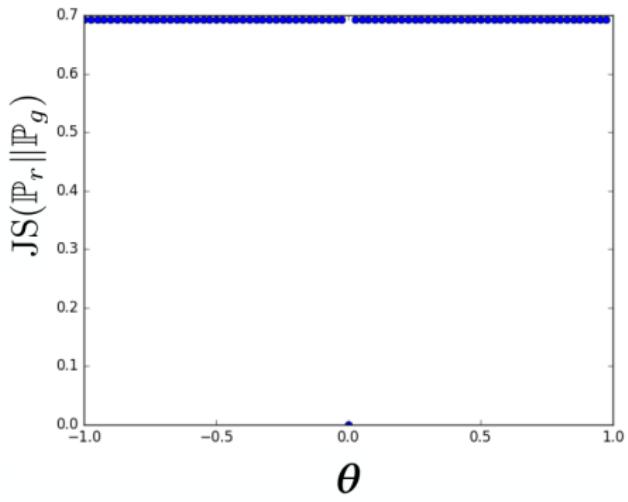
- JS-divergence gradient is zero

## JS and EM distance w.r.t. $\theta$



- JS-divergence gradient is zero

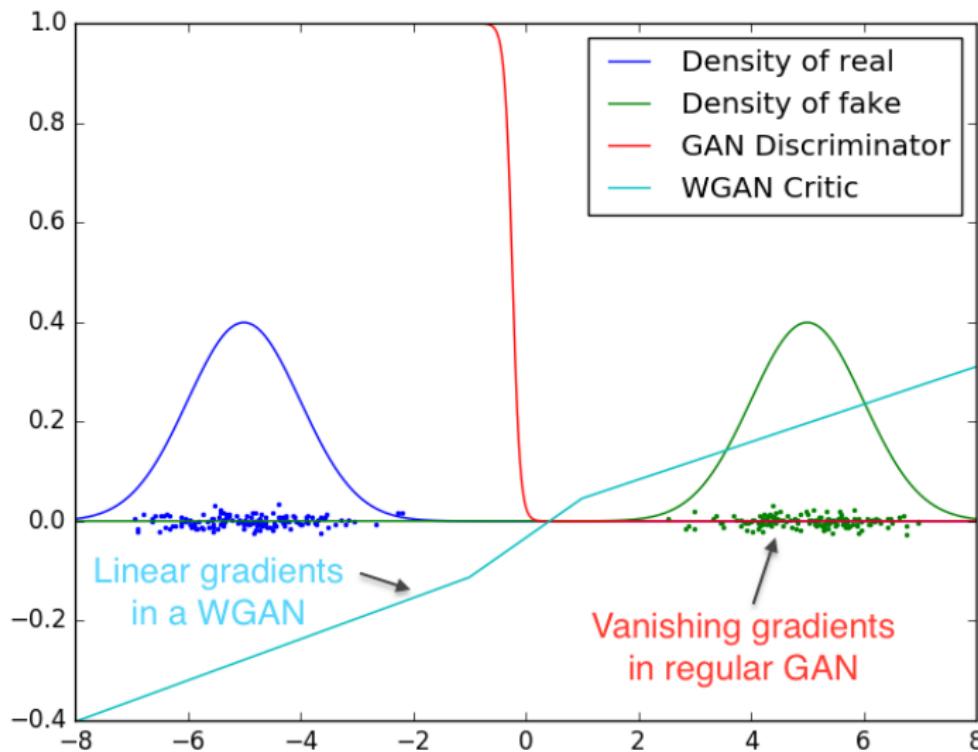
## JS and EM distance w.r.t. $\theta$



- JS-divergence gradient is zero

- Wasserstein gradient is constant

# Gradients between two gaussian distributions



2014

## Generative Adversarial Nets

Ian J. Goodfellow, Jean Pouget-Abadie\*, Mehdi Mirza, Bing Xu, David Warde-Farley,  
Sherjil Ozair,<sup>†</sup> Aaron Courville, Yoshua Bengio<sup>†</sup>

Département d'informatique et de recherche opérationnelle  
Université de Montréal  
Montréal, QC H3C 3J7

### GAN Value function

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

- 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:

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

# Short history of training GANs

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

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

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

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

# Short history of training GANs

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

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

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

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

- 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:

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

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

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

- 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:

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

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

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

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

# Short history of training GANs

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

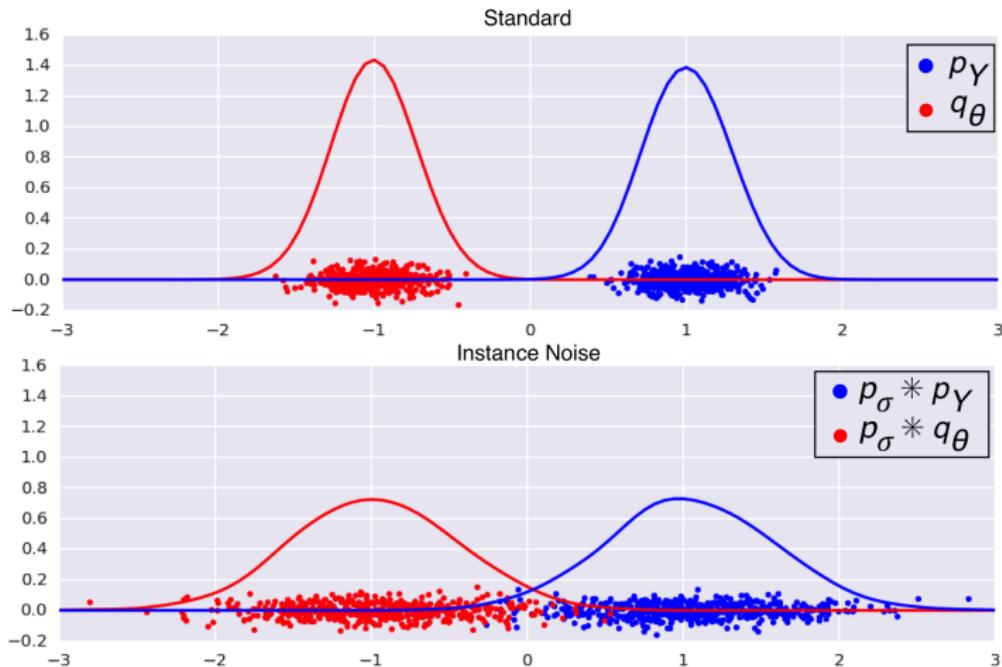
$$D_G^*(\mathbf{x}) = \frac{p_r(\mathbf{x})}{p_r(\mathbf{x}) + p_g(\mathbf{x})}$$

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

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

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

# Trick - adding noise



- Matching the noise corresponds to matching the underlying distributions

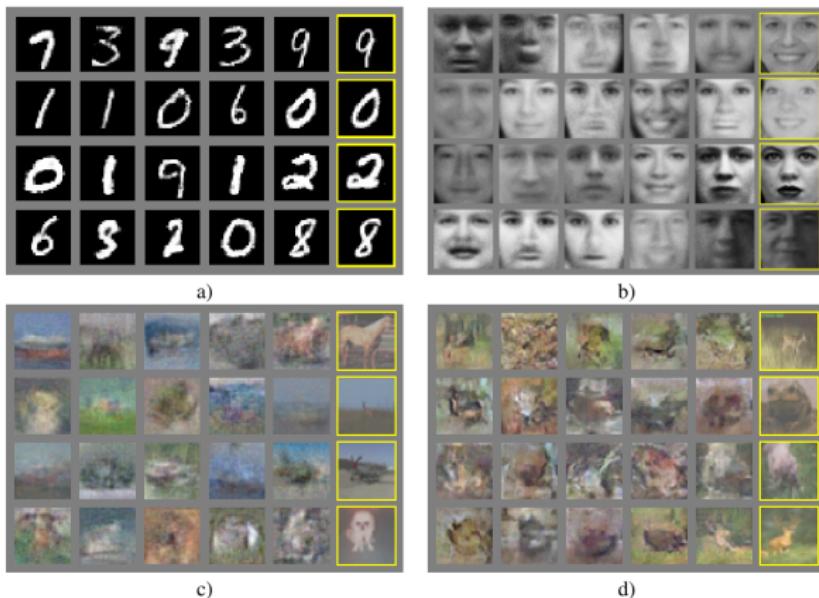
<http://www.inference.vc/instance-noise-a-trick-for-stabilising-gan-training/>

## Trick - cherrypicking architecture

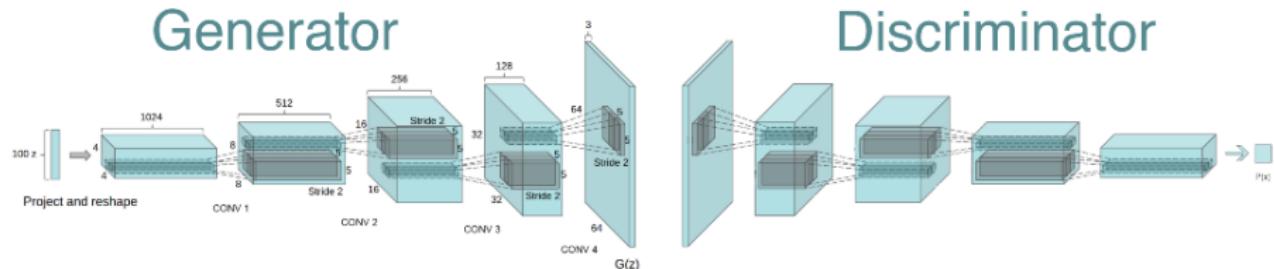
(TFD) [28], and CIFAR-10 [21]. The generator nets used a mixture of rectifier linear activations [19, 9] and sigmoid activations, while the discriminator net used maxout [10] activations. Dropout [17] was applied in training the discriminator net. While our theoretical framework permits the use of dropout 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.

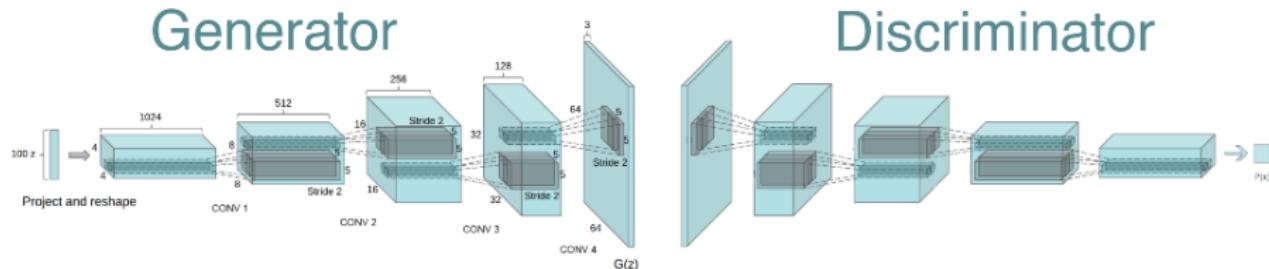


# Trick - cherrypicking architecture - DCGAN



<sup>1</sup><https://www.youtube.com/watch?v=X1mUN6dD8uEt=795s>

# Trick - cherrypicking architecture - DCGAN

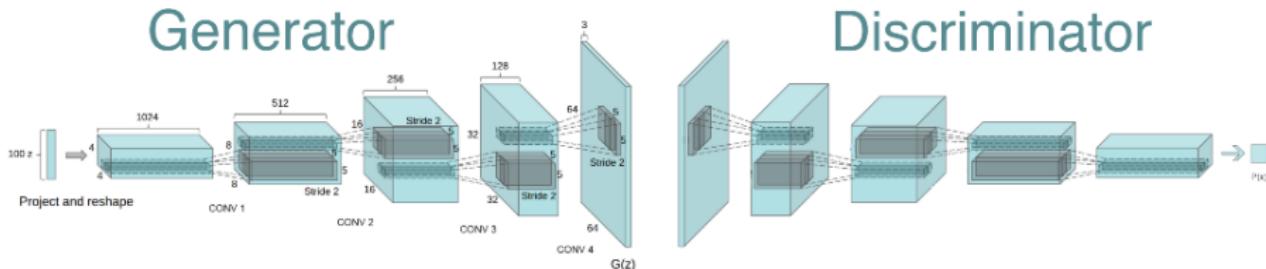


## Architecture guidelines for stable Deep Convolutional GANs

- 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.

<sup>1</sup><https://www.youtube.com/watch?v=X1mUN6dD8uEt=795s>

# Trick - cherrypicking architecture - DCGAN

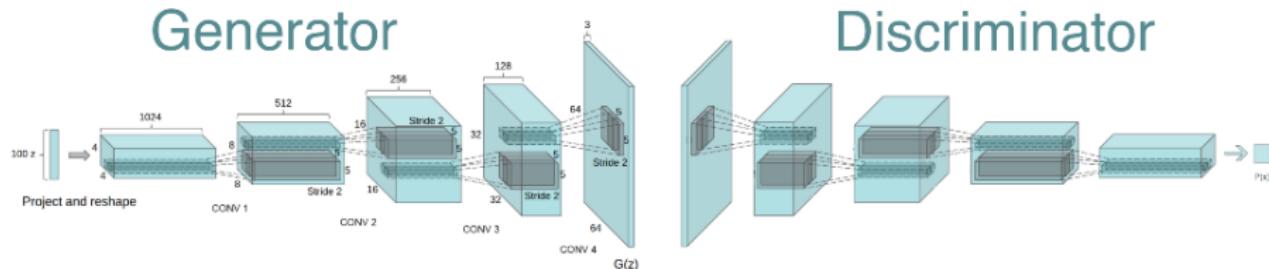


## Architecture guidelines for stable Deep Convolutional GANs

- 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.

<sup>1</sup><https://www.youtube.com/watch?v=X1mUN6dD8uEt=795s>

# Trick - cherrypicking architecture - DCGAN



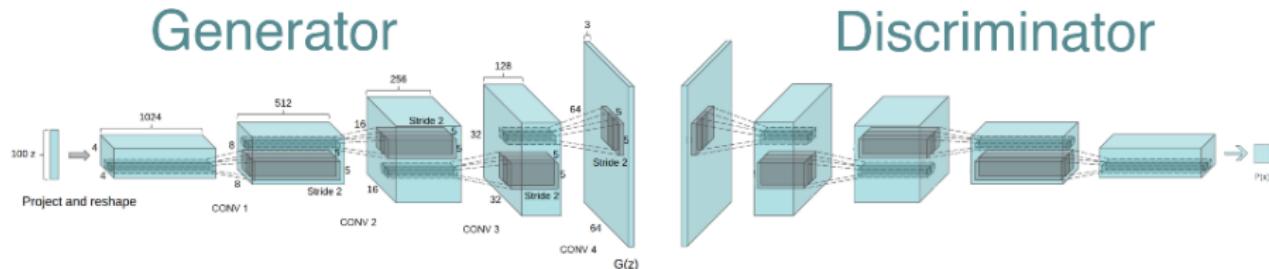
## Architecture guidelines for stable Deep Convolutional GANs

- 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 <sup>1</sup>

<sup>1</sup><https://www.youtube.com/watch?v=X1mUN6dD8uEt=795s>

# Trick - cherrypicking architecture - DCGAN



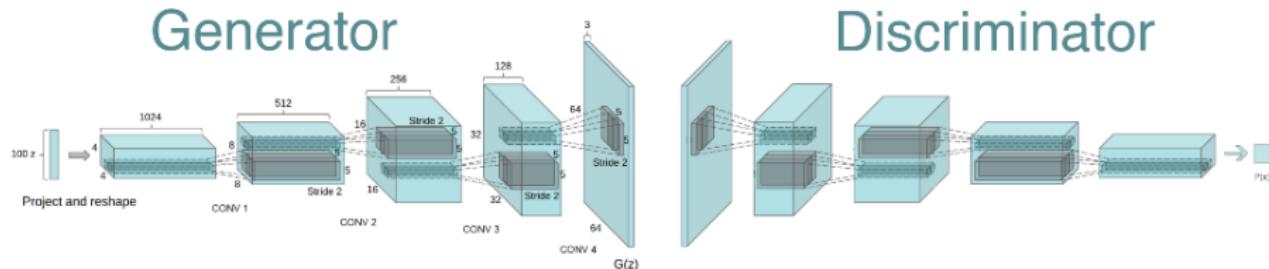
## Architecture guidelines for stable Deep Convolutional GANs

- 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 <sup>1</sup>
- No explanation for model performance

<sup>1</sup><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 <sup>1</sup>
- No explanation for model performance
- Very unstable

<sup>1</sup><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

## EM distance

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

<sup>2</sup><http://cedricvillani.org/wp-content/uploads/2012/08/preprint-1.pdf>

# Short history of training GANs

## EM distance

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

- How to compute the infimum?

<sup>2</sup><http://cedricvillani.org/wp-content/uploads/2012/08/preprint-1.pdf>

# Short history of training GANs

## EM distance

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

- How to compute the infimum?
- 1000 page book on Optimal Transport <sup>2</sup>

<sup>2</sup><http://cedricvillani.org/wp-content/uploads/2012/08/preprint-1.pdf>

# Short history of training GANs

## EM distance

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

- How to compute the infimum?
- 1000 page book on Optimal Transport <sup>2</sup>
- Kantorovich-Rubinstein duality

<sup>2</sup><http://cedricvillani.org/wp-content/uploads/2012/08/preprint-1.pdf>

# Short history of training GANs

## EM distance

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

- How to compute the infimum?
- 1000 page book on Optimal Transport <sup>2</sup>
- Kantorovich-Rubinstein duality

## Kantorovich-Rubinstein duality

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

<sup>2</sup><http://cedricvillani.org/wp-content/uploads/2012/08/preprint-1.pdf>

# Short history of training GANs

## EM distance

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

- How to compute the infimum?
- 1000 page book on Optimal Transport <sup>2</sup>
- Kantorovich-Rubinstein duality

## Kantorovich-Rubinstein duality

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

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

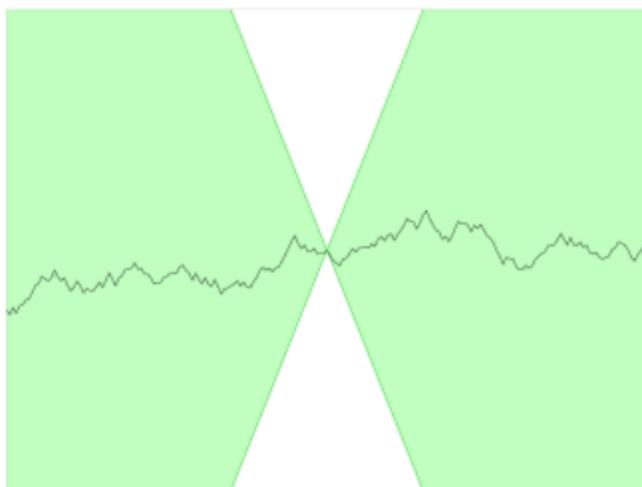
<sup>2</sup><http://cedricvillani.org/wp-content/uploads/2012/08/preprint-1.pdf>

# Lipschitz continuity

## K-Lipschitz function

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

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



<sup>2</sup>[https://en.wikipedia.org/wiki/Lipschitz\\_continuity](https://en.wikipedia.org/wiki/Lipschitz_continuity)

# Short history of training GANs - January 2017

Jan 2017

## Wasserstein GAN

Martin Arjovsky<sup>1</sup>, Soumith Chintala<sup>2</sup>, and Léon Bottou<sup>1,2</sup>

<sup>1</sup>Courant Institute of Mathematical Sciences

<sup>2</sup>Facebook AI Research

### WGAN Value function

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

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

# Short history of training GANs - January 2017

Jan 2017

## Wasserstein GAN

Martin Arjovsky<sup>1</sup>, Soumith Chintala<sup>2</sup>, and Léon Bottou<sup>1,2</sup>

<sup>1</sup>Courant Institute of Mathematical Sciences

<sup>2</sup>Facebook AI Research

### WGAN Value function

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

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

- 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

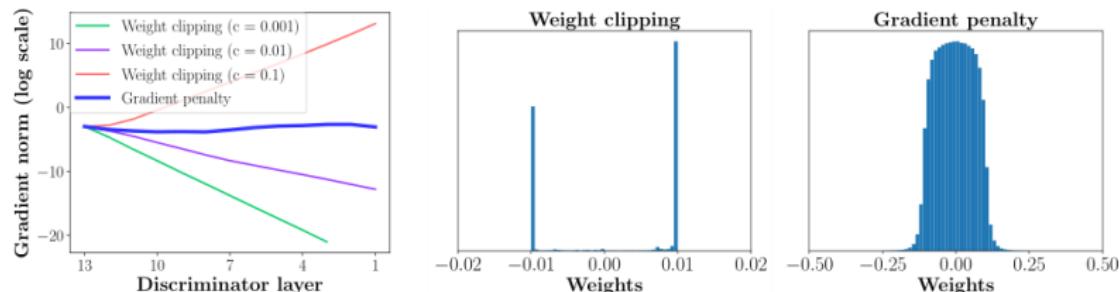
- After optimization step, clip all weights to  $[-c, c]$
- Results in critic being a subset of K-Lipschitz functions, where K is a function of c and the critic's architecture

# 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

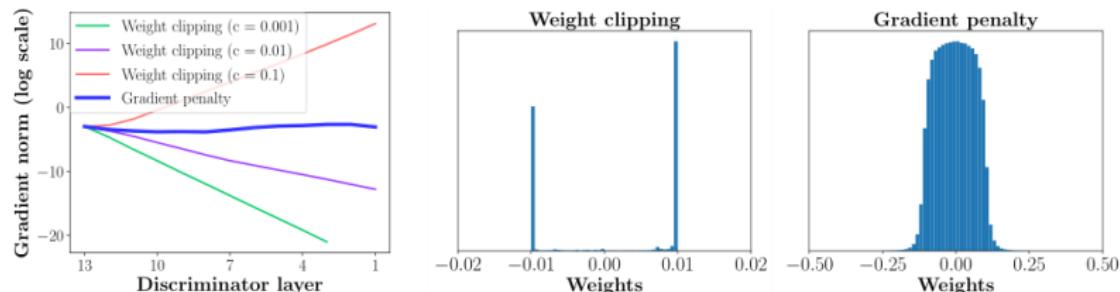
- After optimization step, clip all weights to  $[-c, c]$
- Results in critic being a subset of K-Lipschitz functions, where K is a function of c and the critic's architecture



- Works! But...

# Method #1 - Weight Clipping

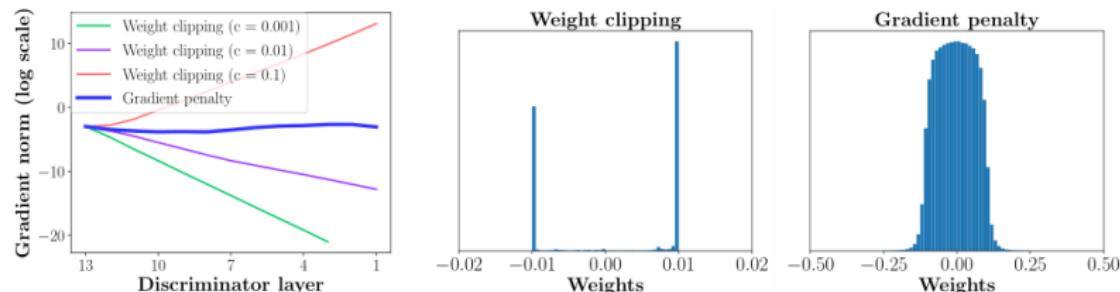
- After optimization step, clip all weights to  $[-c, c]$
- Results in critic being a subset of K-Lipschitz functions, where K is a function of c and the critic's architecture



- 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

$$\text{critic } |f(x) - f(y)| \leq |x - y|$$

## 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

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

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

## Method #2 - Gradient Penalty

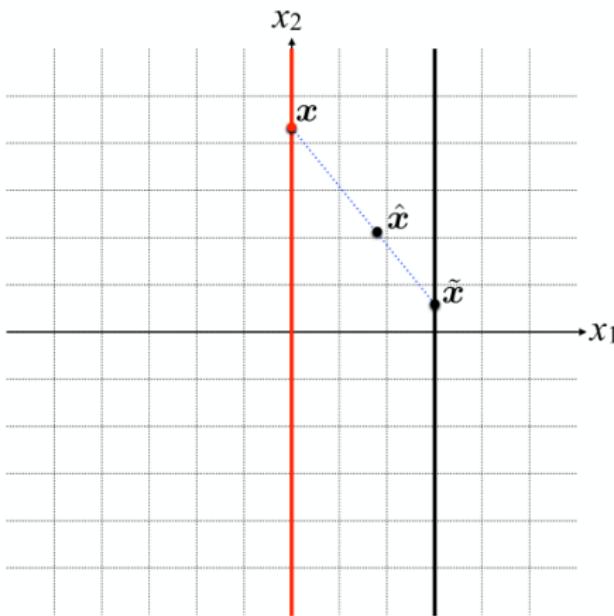
- Property of the optimal WGAN  
critic  $|f(x) - f(y)| \leq |x - y|$
- Optimal WGAN critic has gradient norm 1
- Add a regularization term to the WGAN value function
- Sampling along straight lines

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

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

- Gradient penalty

$$D(\hat{\mathbf{x}})$$



## Method #2 - Gradient Penalty

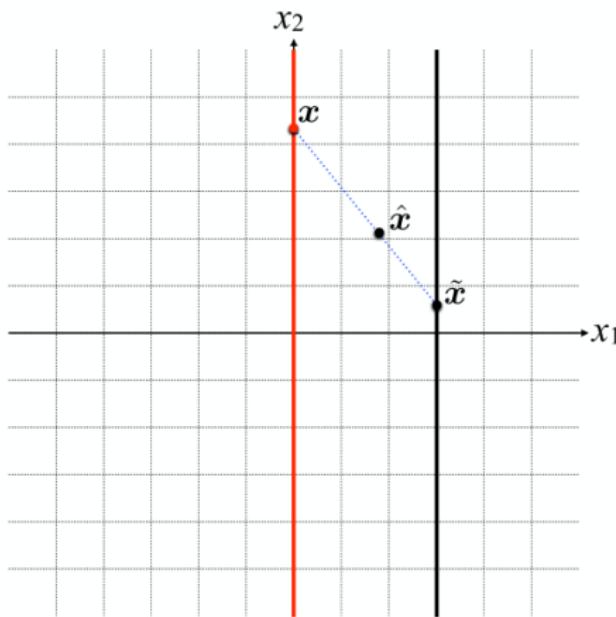
- Property of the optimal WGAN  
critic  $|f(x) - f(y)| \leq |x - y|$
- Optimal WGAN critic has gradient norm 1
- Add a regularization term to the WGAN value function
- Sampling along straight lines

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

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

- Gradient penalty

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



## Method #2 - Gradient Penalty

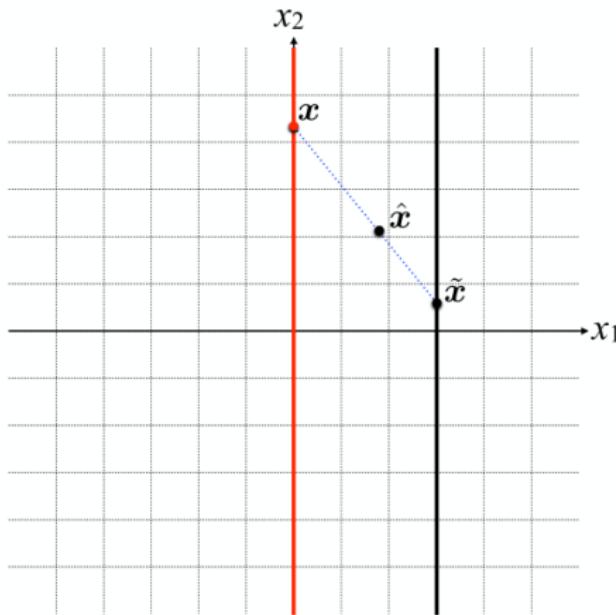
- Property of the optimal WGAN  
critic  $|f(x) - f(y)| \leq |x - y|$
- Optimal WGAN critic has gradient norm 1
- Add a regularization term to the WGAN value function
- Sampling along straight lines

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

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

- Gradient penalty

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



## Method #2 - Gradient Penalty

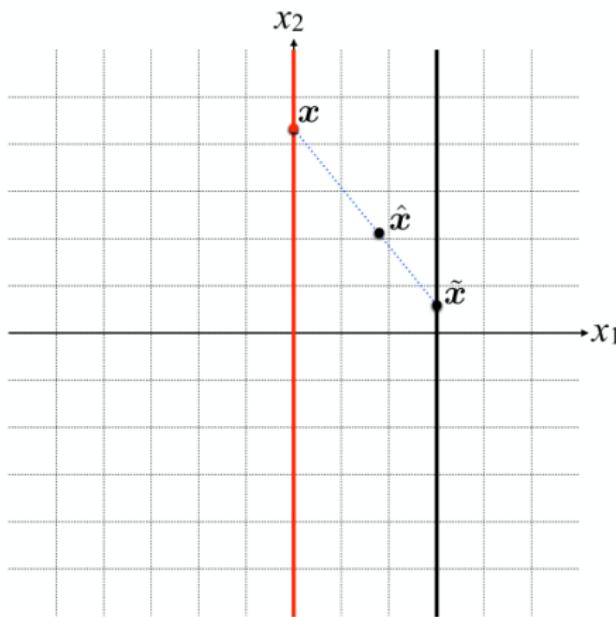
- Property of the optimal WGAN critic  $|f(x) - f(y)| \leq |x - y|$
- Optimal WGAN critic has gradient norm 1
- Add a regularization term to the WGAN value function
- Sampling along straight lines

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

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

- Gradient penalty

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



# Short history of training GANs - March 2017

Mar 2017

## Improved Training of Wasserstein GANs

Ishaan Gulrajani<sup>1</sup>, Faruk Ahmed<sup>1</sup>, Martin Arjovsky<sup>2</sup>, Vincent Dumoulin<sup>1</sup>, Aaron Courville<sup>1,3</sup>

<sup>1</sup> Montreal Institute for Learning Algorithms

<sup>2</sup> Courant Institute of Mathematical Sciences

<sup>3</sup> CIFAR Fellow

igul122@gmail.com

{faruk.ahmed,vincent.dumoulin,aaron.courville}@umontreal.ca

ma4371@nyu.edu

### WGAN-GP Value function

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

Mar 2017

## Improved Training of Wasserstein GANs

Ishaan Gulrajani<sup>1</sup>, Faruk Ahmed<sup>1</sup>, Martin Arjovsky<sup>2</sup>, Vincent Dumoulin<sup>1</sup>, Aaron Courville<sup>1,3</sup>

<sup>1</sup> Montreal Institute for Learning Algorithms

<sup>2</sup> Courant Institute of Mathematical Sciences

<sup>3</sup> CIFAR Fellow

igul122@gmail.com

{faruk.ahmed,vincent.dumoulin,aaron.courville}@umontreal.ca

ma4371@nyu.edu

### WGAN-GP Value function

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

- Enforcing the Lipschitz constraint with a gradient penalty regularization term

Mar 2017

## Improved Training of Wasserstein GANs

Ishaan Gulrajani<sup>1</sup>, Faruk Ahmed<sup>1</sup>, Martin Arjovsky<sup>2</sup>, Vincent Dumoulin<sup>1</sup>, Aaron Courville<sup>1,3</sup>

<sup>1</sup> Montreal Institute for Learning Algorithms

<sup>2</sup> Courant Institute of Mathematical Sciences

<sup>3</sup> CIFAR Fellow

igul122@gmail.com

{faruk.ahmed,vincent.dumoulin,aaron.courville}@umontreal.ca

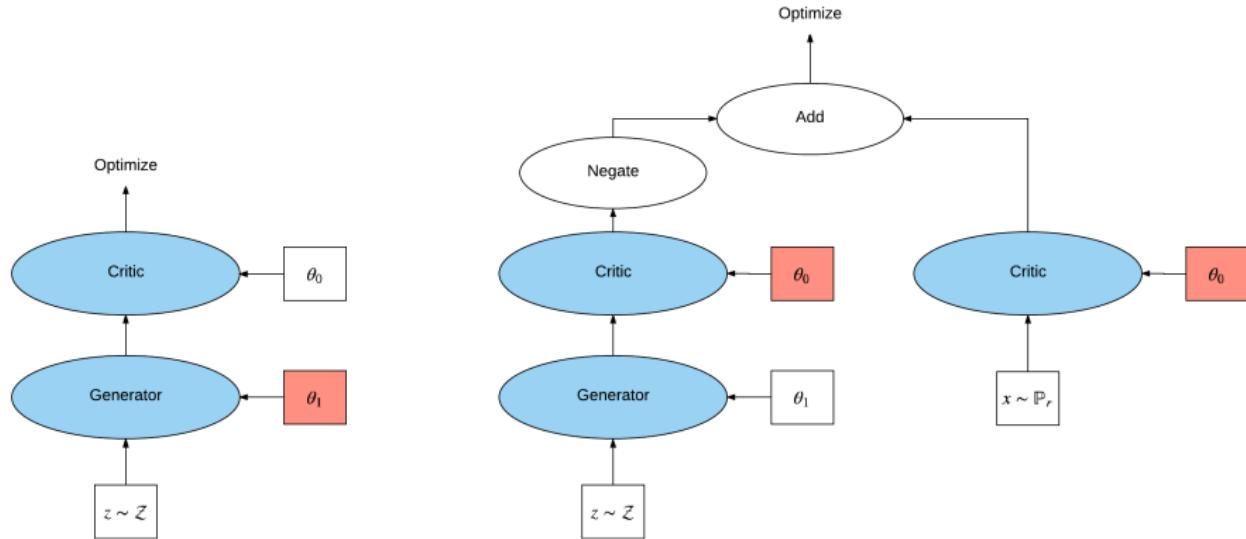
ma4371@nyu.edu

### WGAN-GP Value function

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

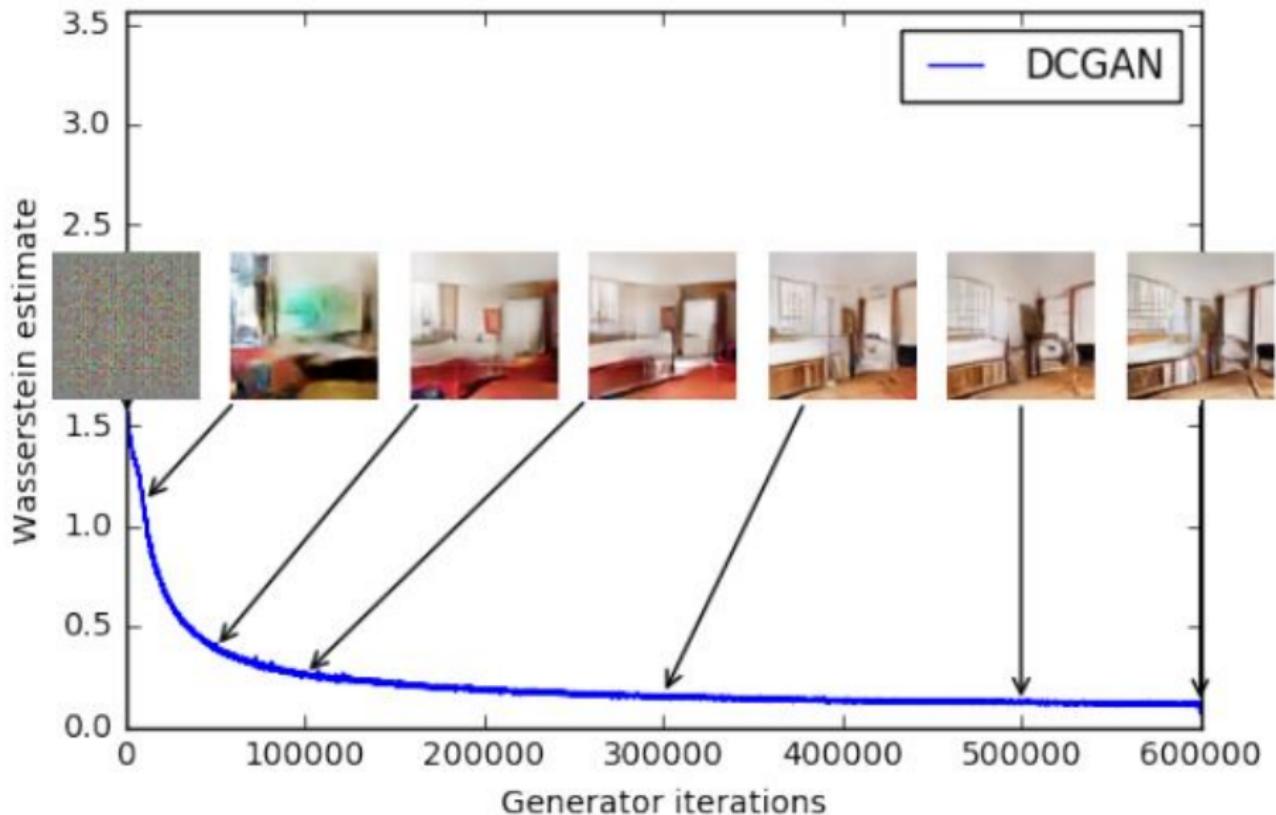
- Enforcing the Lipschitz constraint with a gradient penalty regularization term
- Improvements?

# Recap



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

# Meaningful loss function



# WGAN-GP results

DCGAN

Baseline ( $G$ : DCGAN,  $D$ : DCGAN)



LSGAN

Baseline ( $G$ : DCGAN,  $D$ : DCGAN)



WGAN (clipping)

Baseline ( $G$ : DCGAN,  $D$ : DCGAN)

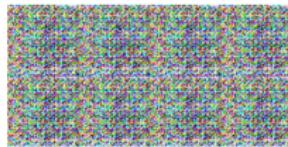


WGAN-GP (ours)

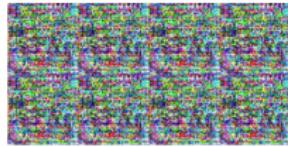
Baseline ( $G$ : DCGAN,  $D$ : DCGAN)



$G$ : No BN and a constant number of filters,  $D$ : DCGAN

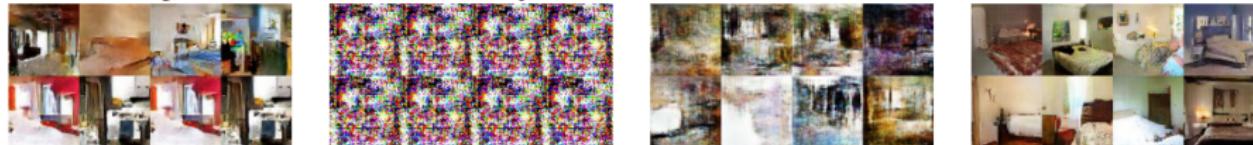


$G$ : 4-layer 512-dim ReLU MLP,  $D$ : DCGAN



# WGAN-GP results

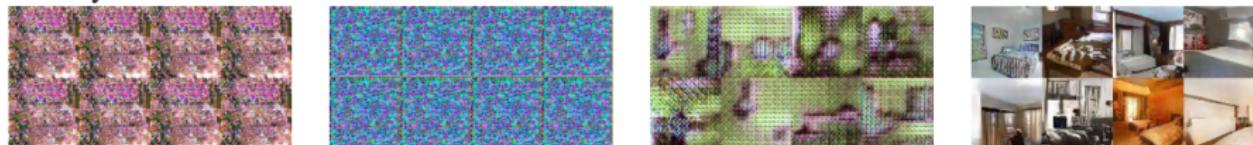
Gated multiplicative nonlinearities everywhere in  $G$  and  $D$



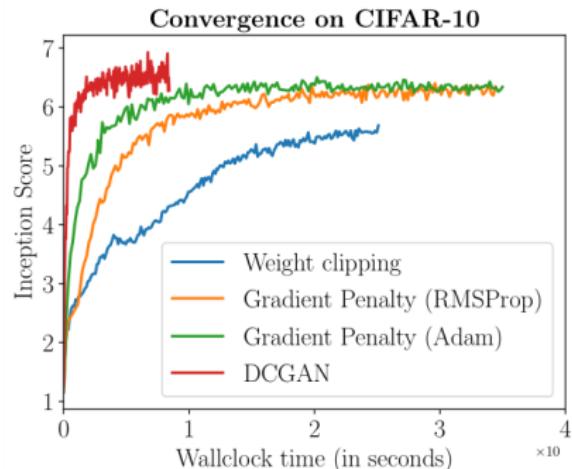
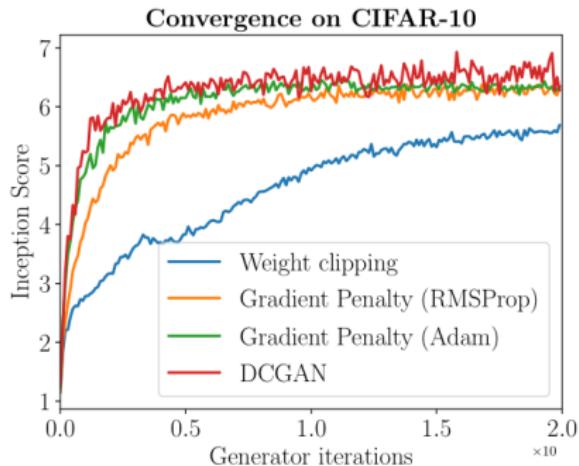
$tanh$  nonlinearities everywhere in  $G$  and  $D$



101-layer ResNet  $G$  and  $D$



# 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

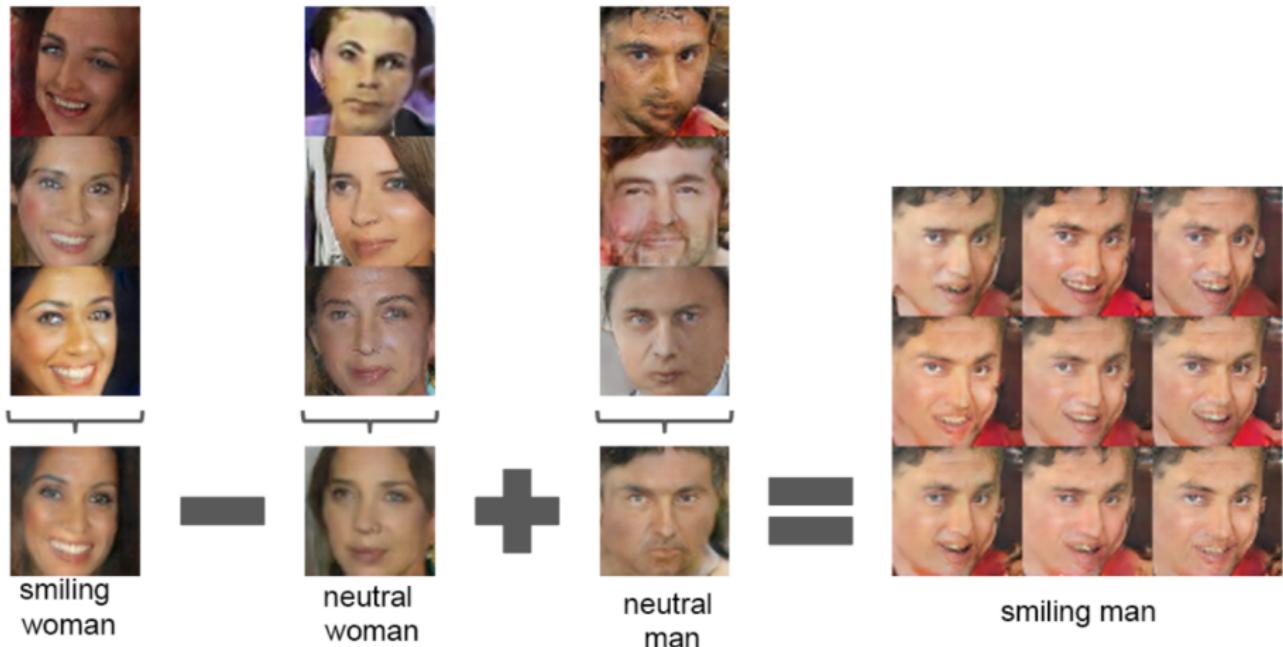
- Two neural networks are playing a zero-sum game
- Nash equilibrium
- Goal - both players adopting strategies where any deviations would be disadvantageous

# Game theory perspective

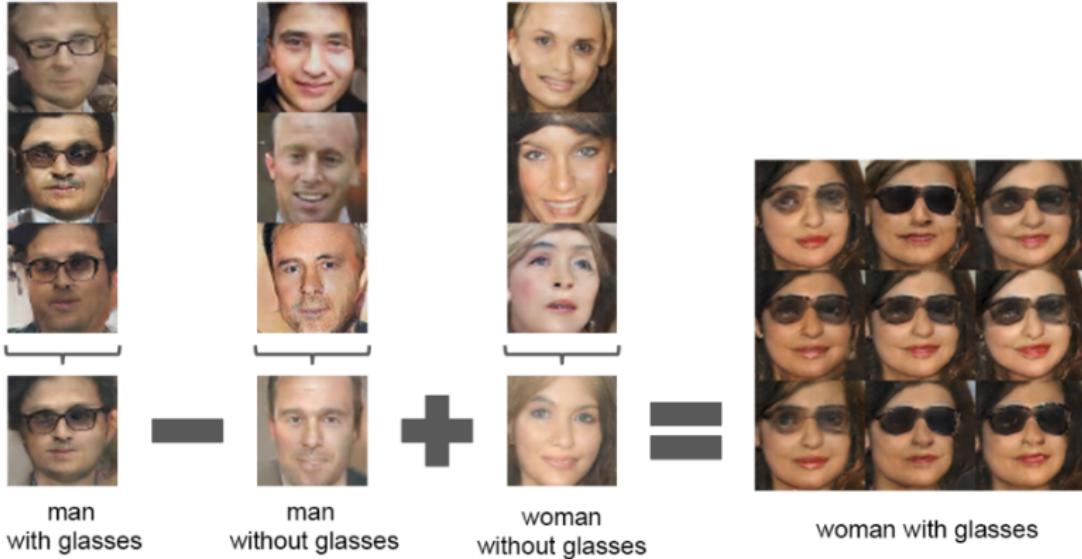
- Two neural networks are playing a zero-sum game
- Nash equilibrium
- Goal - both players adopting strategies where any deviations would be disadvantageous
- Lack of theoretical insights

# Cool things you can do with GANs

# Latent space arithmetic



# Latent space arithmetic



# Interpolation between images



# Super-Resolution

original



bicubic  
(21.59dB/0.6423)



SRResNet  
(23.44dB/0.7777)

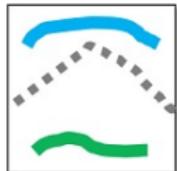
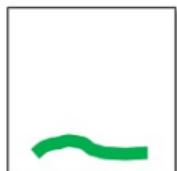


SRGAN  
(20.34dB/0.6562)



# Interactive GAN

User edits



Generated images



Color

Sketch

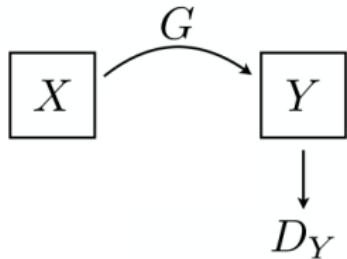
▶ Interactive GAN

# Style Transfer

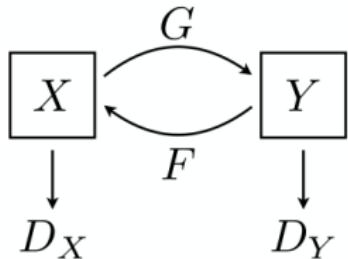
$X$

$Y$

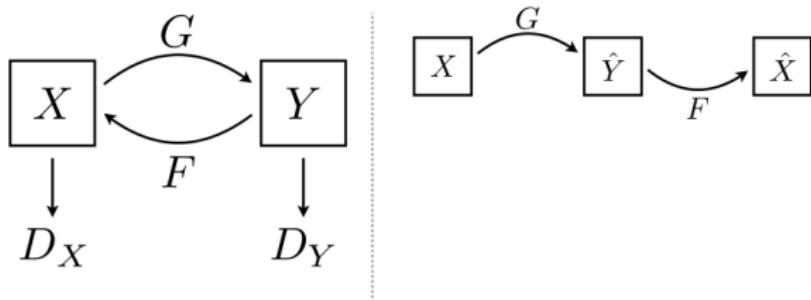
# Style Transfer



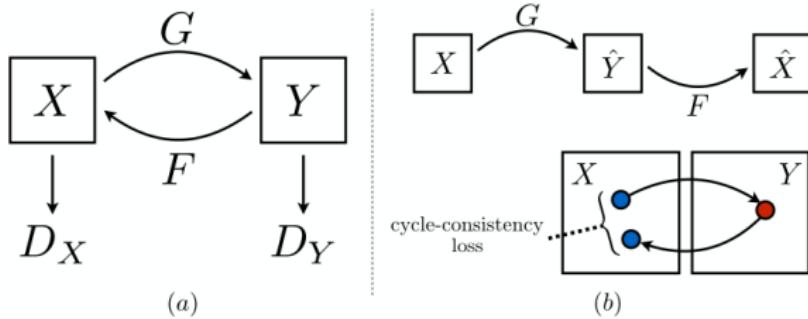
# Style Transfer



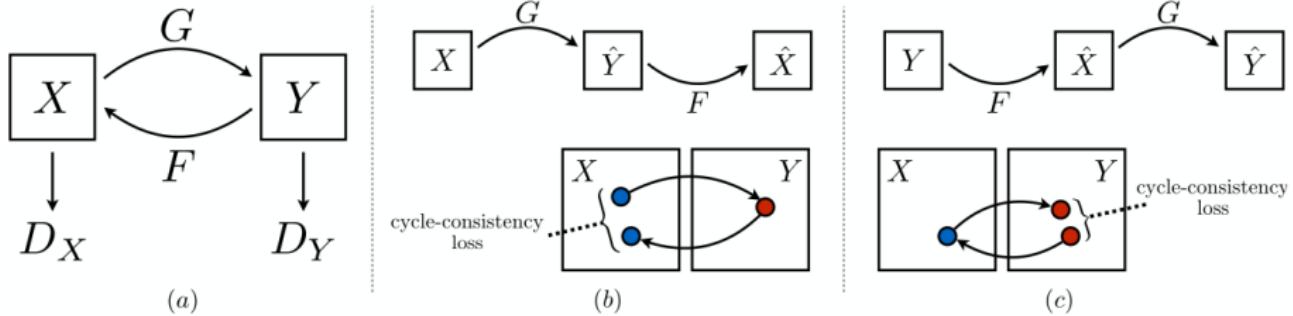
# Style Transfer



# Style Transfer



# Style Transfer



# Style Transfer

Monet ↪ Photos



Monet → photo

Zebras ↪ Horses



zebra → horse

Summer ↪ Winter



summer → winter



photo → Monet



horse → zebra



winter → summer



Photograph



Monet



Van Gogh

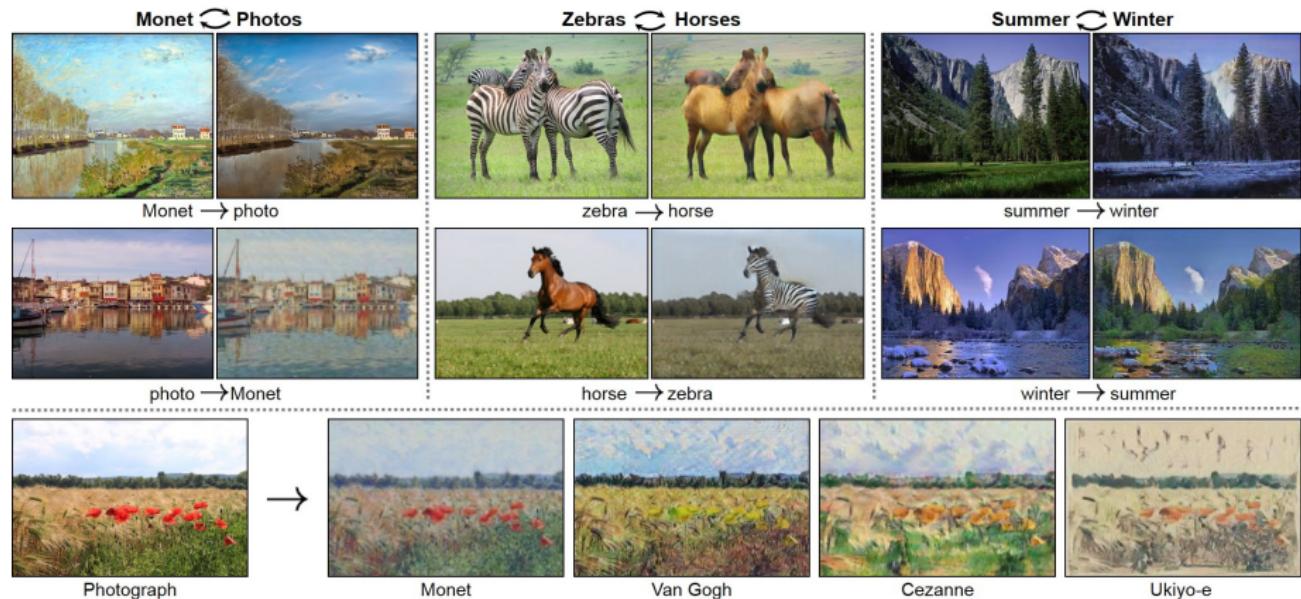


Cezanne



Ukiyo-e

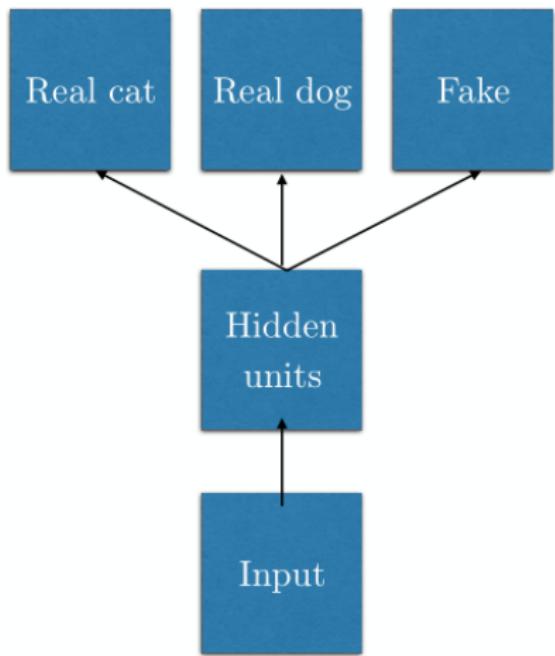
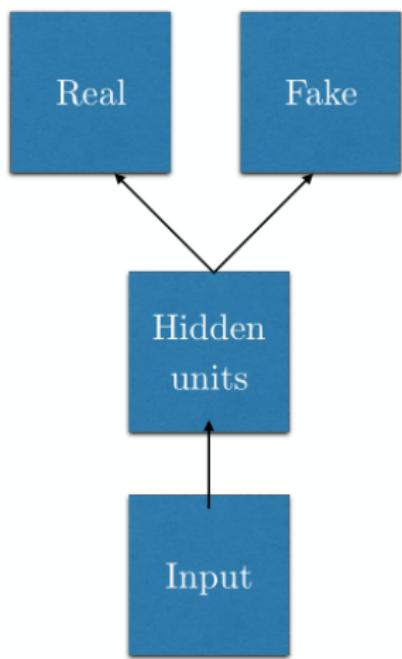
# Style Transfer



▶ CycleGAN in action

▶ Creepy CycleGAN

# 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

# Perspectives

- Matching distributions
- Neural network as a cost function
- Modular training
- Two player minimax game

# 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?
- Are GANs overfitting? How to measure it?

# 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?
- Are GANs overfitting? How to measure it?
- 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?
- Are GANs overfitting? How to measure it?
- Problems with counting
- Perspective

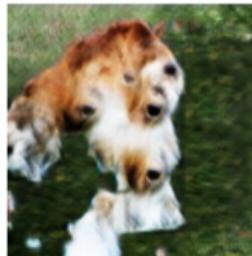
# Problems with GANs

- 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?
- Are GANs overfitting? How to measure it?
- Problems with counting
- Perspective
- High level structure

# Problems with GANs

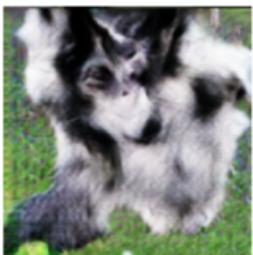
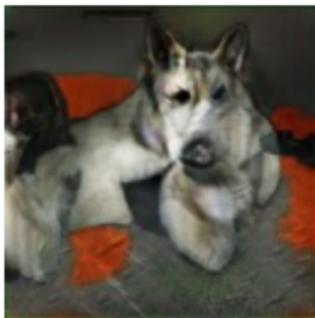
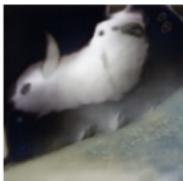
- How to evaluate GANs?
- No way to tell how good of an approximation of EM distance the critic really is
- Mode collapse
- Multi-task solving?
- Are GANs overfitting? How to measure it?
- Problems with counting
- Perspective
- High level structure
- Sequential data

# Counting



<sup>2</sup>Ian J. Goodfellow: NIPS 2016 Tutorial: Generative Adversarial Networks

# Perspective



<sup>2</sup>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!