
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

Faculty of Electrical Engineering and Computing
University of Zagreb, Croatia

bruno.gavranovic@fer.hr

July 23, 2018

Overview

Meta

- Generative Adversarial Networks are a new idea - June 2014

Meta

- Generative Adversarial Networks are a new idea - June 2014

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

- Generative Adversarial Networks are a new idea - June 2014

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.”

- We still don’t properly understand the mechanics

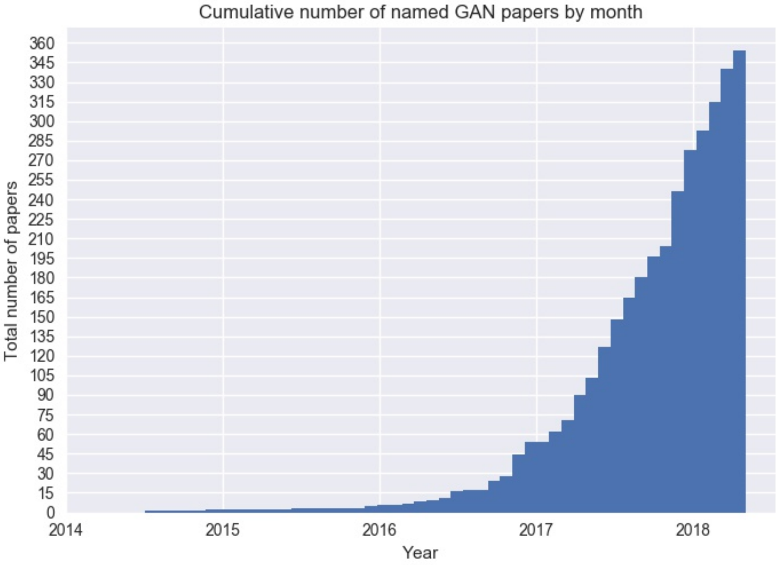
Meta

- Generative Adversarial Networks are a new idea - June 2014

Yann LeCun

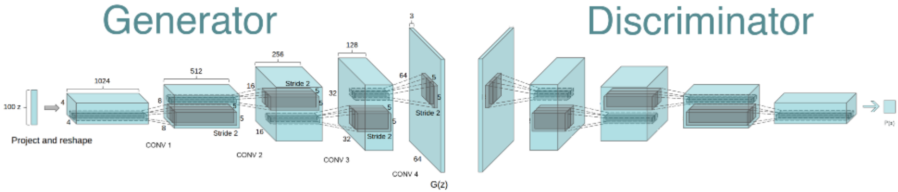
"The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks).
This, and the variations that are now being proposed is the most interesting idea in the last 10 years in ML, in my opinion."

- We still don't properly understand the mechanics
- Compared to other ML models, we're still in early stages


Cumulative number of named GAN papers by month
Total number of papers
360
345
330
315
300
285
270
255
240
225
210
195
180
165
150
135
120
105
90
75
60
45
30
15
0
2014
2015
2016
2017
2018
Year

GAN: Just tell me what it is

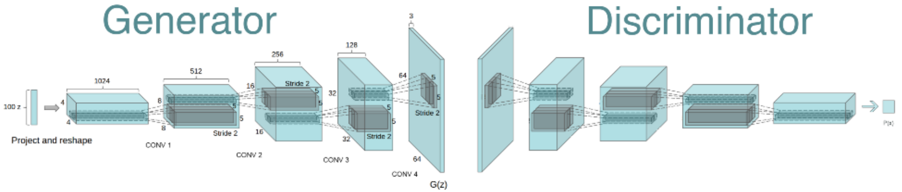
- Generative machine learning model in which two neural networks are competing against each other



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

GAN: Just tell me what it is

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

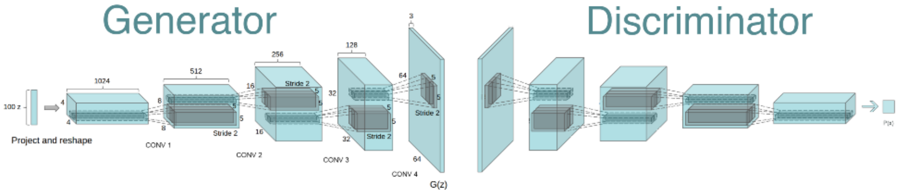


- Instead of using a standard fixed cost function, we *learn* the cost function with the neural network

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

GAN: Just tell me what it is

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

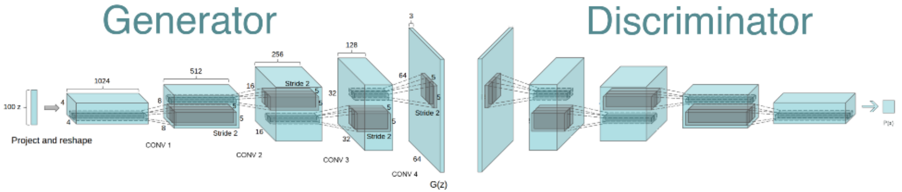


- Instead of using a standard fixed cost function, we *learn* the cost function with the neural network
- We alternate between training different parts of the network

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

GAN: Just tell me what it is

- 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
- It's difficult to train and analyze

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

Forger and the police

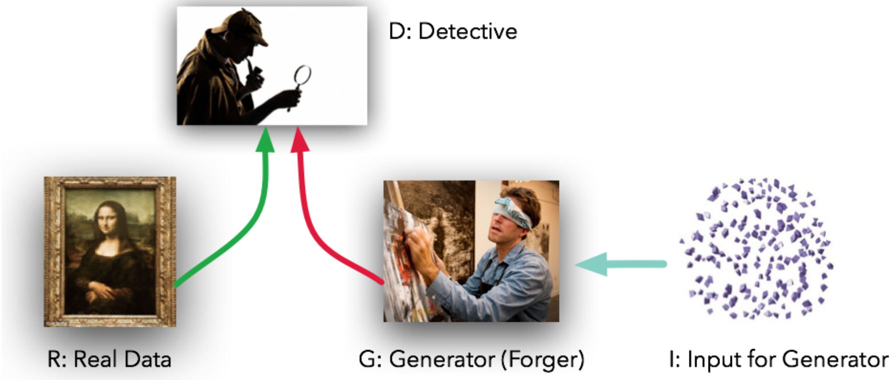


Figure: Analogy capturing the generator-discriminator dynamics

[0] https://medium.com/@devnag/generative-adversarial-networks-gans-in-50-lines-of-code-pytorch-e81b79659e3f

Figure: Which side are real images?

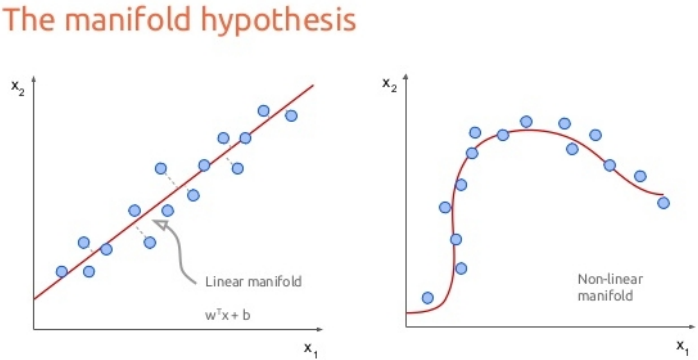
3.5 years of Progress on Faces

[0] http://maliciousaireport.com/

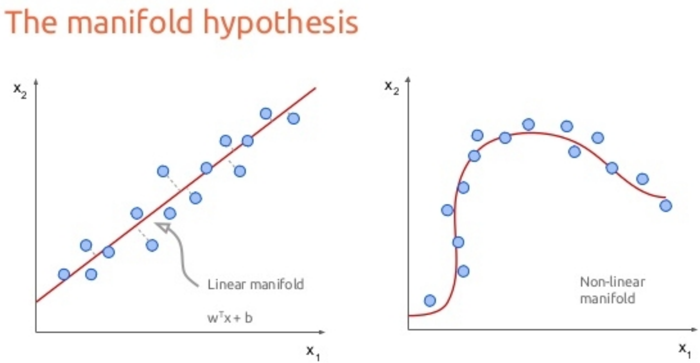
What is this talk about?

- Manifold learning
- Computational graphs
- Statistical distances
- Practical training advice

Prerequisites

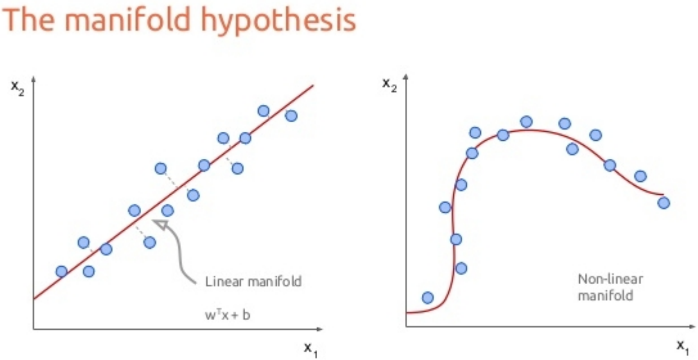


Prerequisites

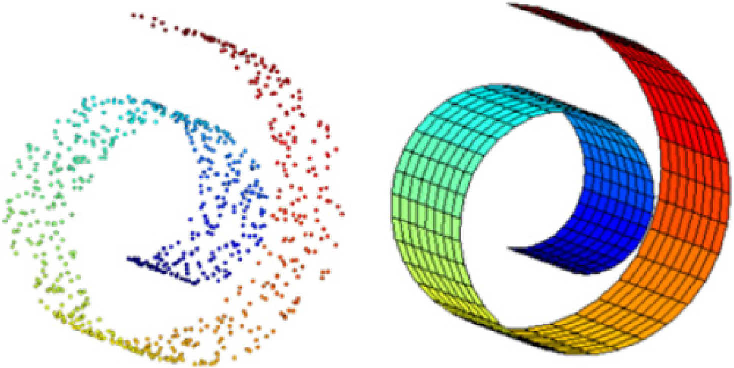


- Understanding real world data

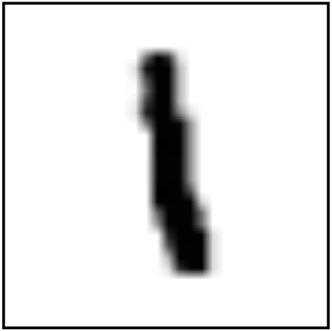
Prerequisites



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



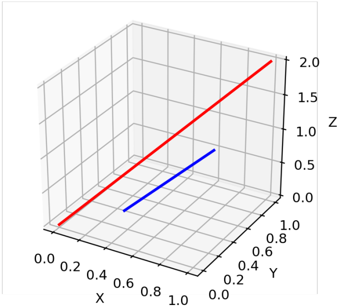
(a) (b)

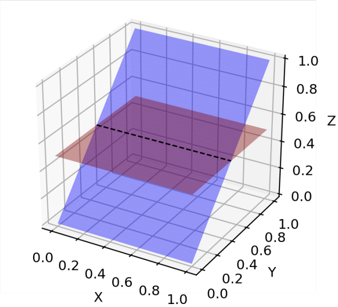


$$\cong \begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&.6&.8&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.5&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.7&0&0&0&0&0\\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

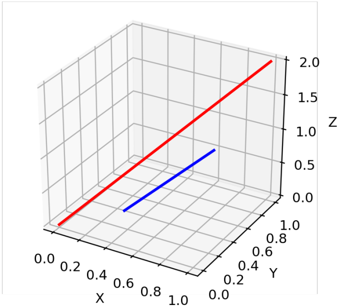
Do low-dimensional manifolds overlap?

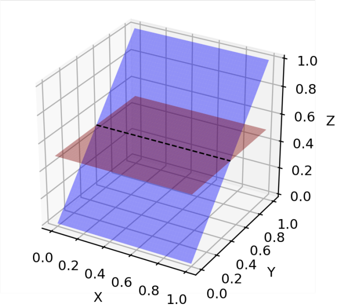
Do low-dimensional manifolds overlap?





Do low-dimensional manifolds overlap?





- Support of intersection of low-dimensional manifolds in a high-dimensional space is approximately zero!

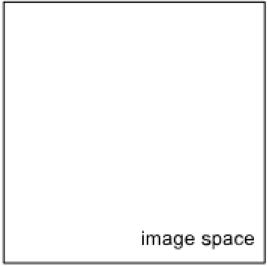


image space

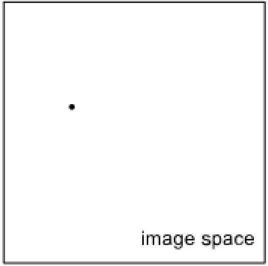


image space

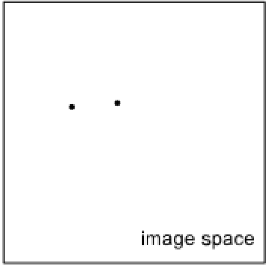


image space

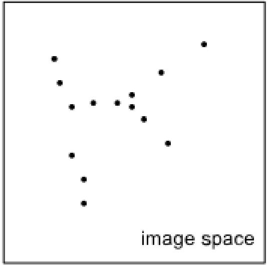
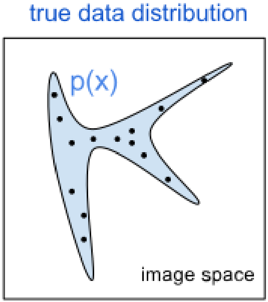
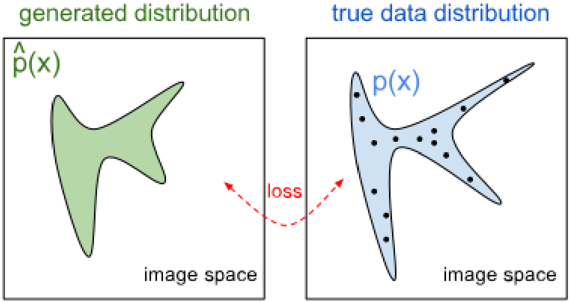
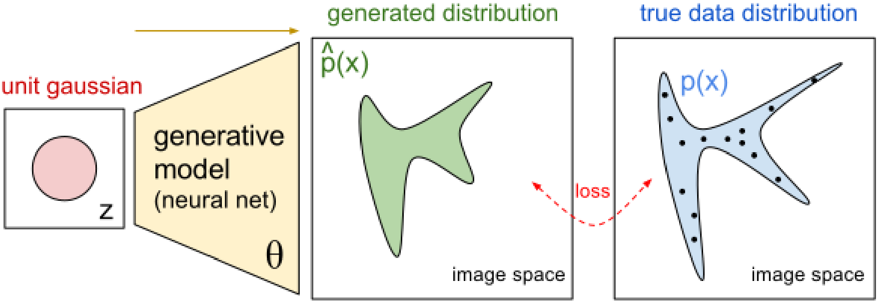


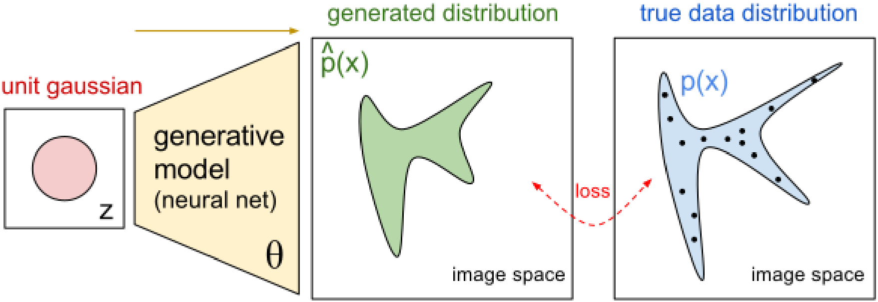
image space


true data distribution
p(x)
image space

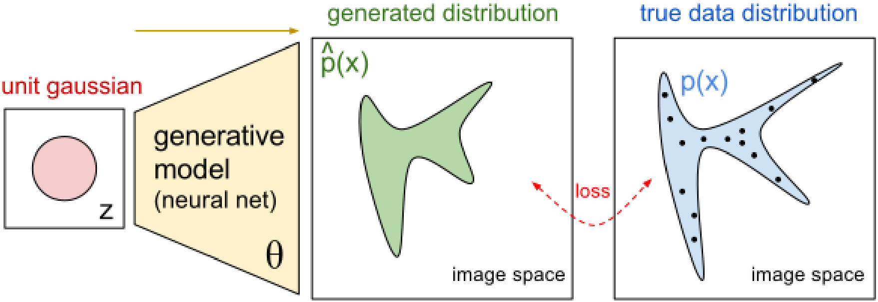

generated distribution
true data distribution
$\hat{p}(x)$
$p(x)$
loss
image space
image space



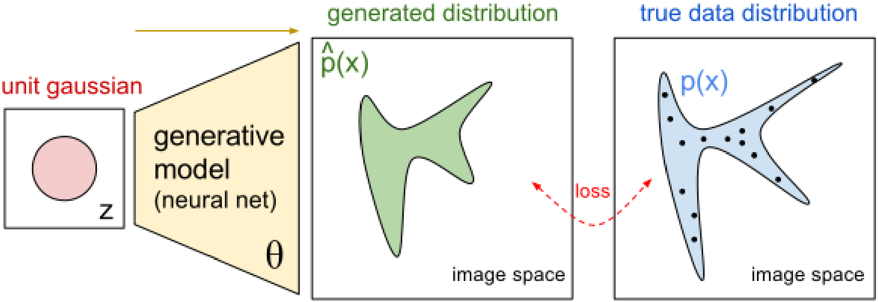
- We have a random vector $z \sim \mathcal{Z}$



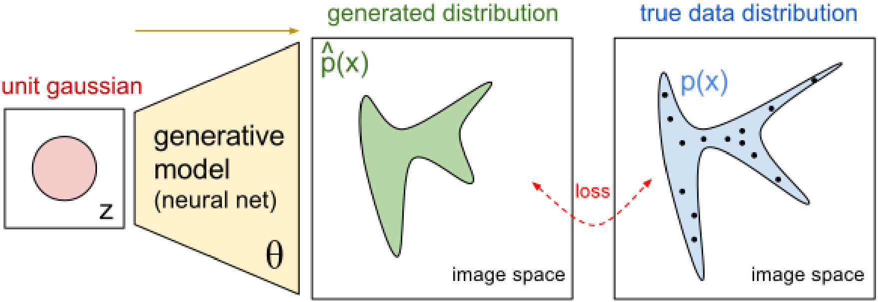
- We have a random vector $z \sim \mathcal{Z}$
- We can define a parametric function $g_\theta : \mathcal{Z} \to \mathcal{X}$ that generates samples from a certain distribution $\mathbb{P}_\theta$



- 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 $\mathbb{P}_r$

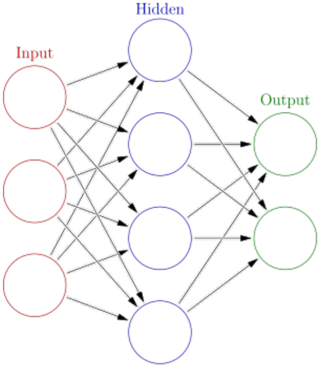


- We have a random vector $z \sim \mathcal{Z}$
- We can define a parametric function $g_\theta : \mathcal{Z} \to \mathcal{X}$ that generates samples from a certain distribution $\mathbb{P}_\theta$
- We have samples from the real data distribution $\mathbb{P}_r$
- By varying θ we can make the $\mathbb{P}_\theta$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

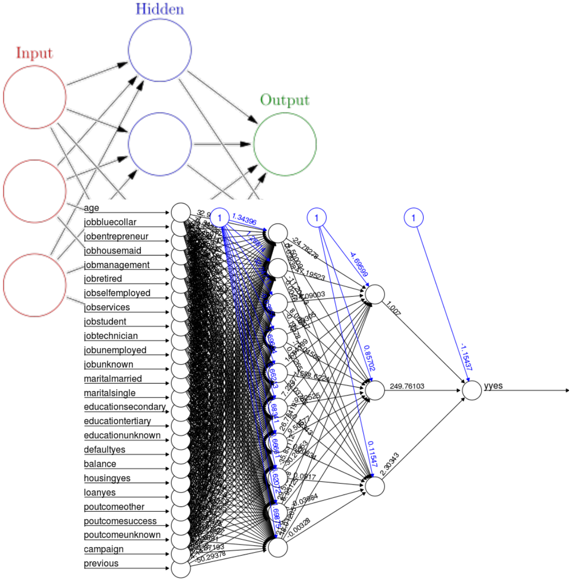
Let’s take a step back.

What is a neural network, actually?

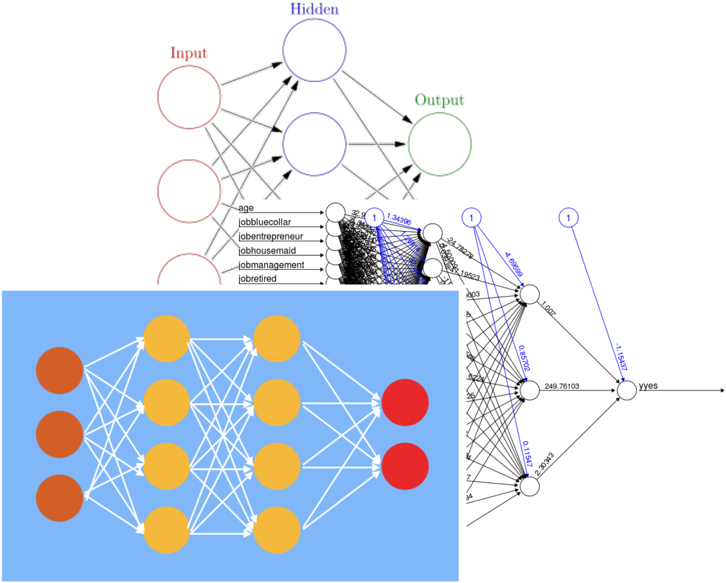
What is a neural network, actually?



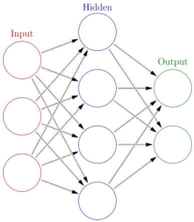
What is a neural network, actually?

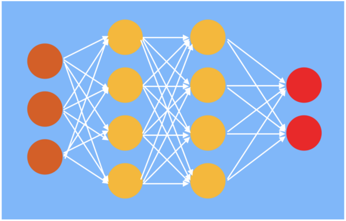


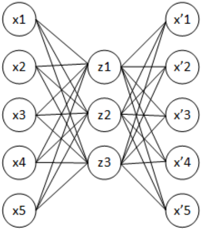
What is a neural network, actually?



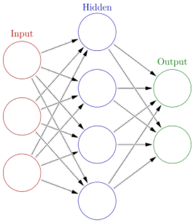
What is a neural network, actually?

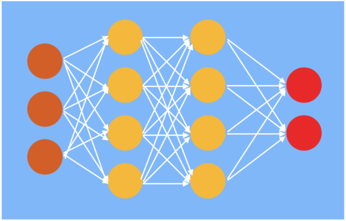


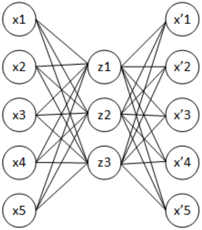




What is a neural network, actually?

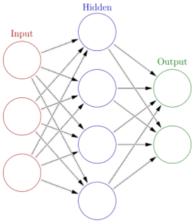


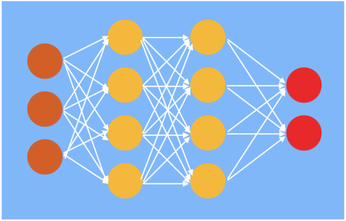


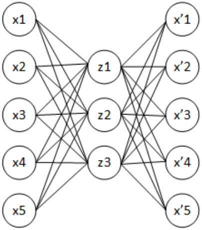


- Frameworks need to manage batches of data, a cost function, the entire training procedure

What is a neural network, actually?

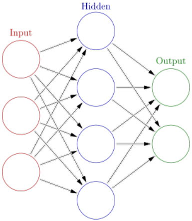


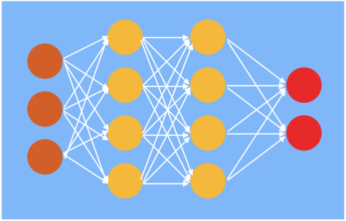


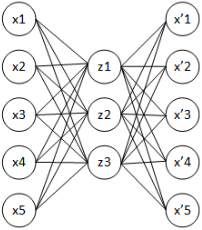


- Frameworks need to manage batches of data, a cost function, the entire training procedure
- This is not they store and manipulate neural networks!

What is a neural network, actually?

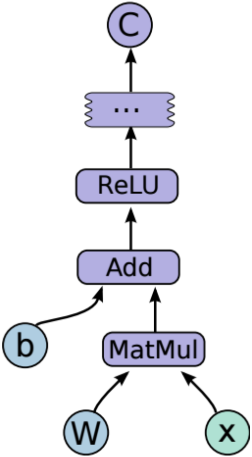






- Frameworks need to manage batches of data, a cost function, the entire training procedure
- This is not they store and manipulate neural networks!
- Internally, networks are stored as directed acyclic graphs (DAGs) - also called computational graphs

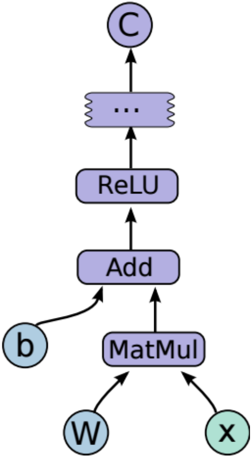
Neural networks as DAGs



- Captures full power expressable with usual diagrams + much more

[0] https://medium.com/@asjad/notes-on-tensor-flow-b90ef02b144f

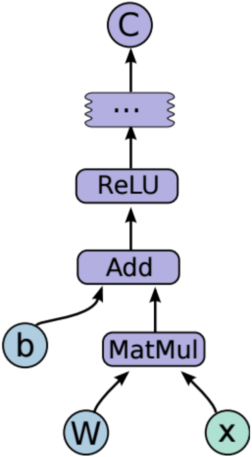
Neural networks as DAGs



- Captures full power expressable with usual diagrams + much more
- *Actual* way it's implemented in frameworks

[0] `https://medium.com/@asjad/notes-on-tensor-flow-b90ef02b144f`

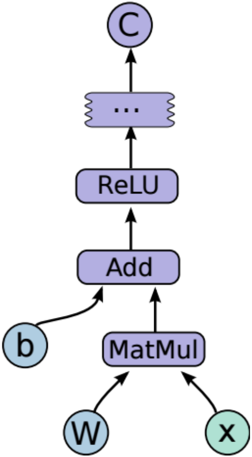
Neural networks as DAGs



- Captures full power expressable with usual diagrams + much more
- *Actual* way it's implemented in frameworks
- *The way* we need to use to understand GANs

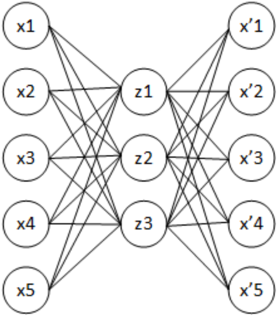
[0] https://medium.com/@asjad/notes-on-tensor-flow-b90ef02b144f

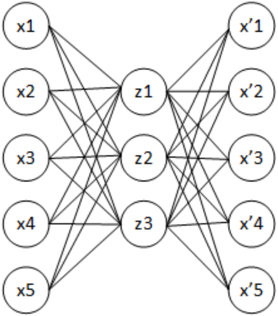
Neural networks as DAGs

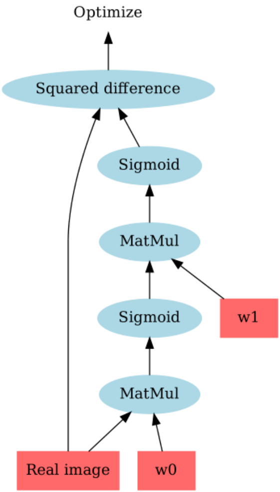


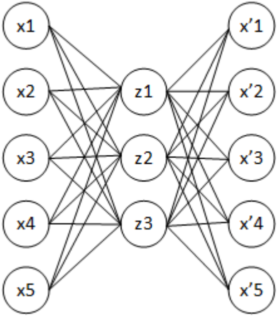
- Captures full power expressable with usual diagrams + much more
- *Actual* way it's implemented in frameworks
- *The way* we need to use to understand GANs
- Notation we use shapes the way we think

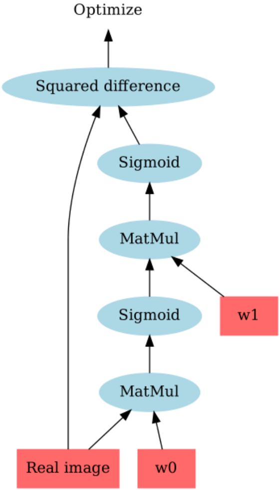
[0] https://medium.com/@asjad/notes-on-tensor-flow-b90ef02b144f

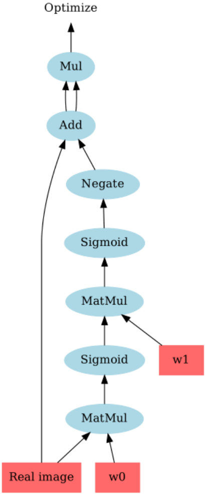

x1
x2
x3
x4
x5
z1
z2
z3
x'1
x'2
x'3
x'4
x'5

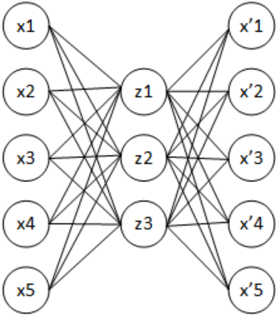

x1
x2
x3
x4
x5
z1
z2
z3
x'1
x'2
x'3
x'4
x'5

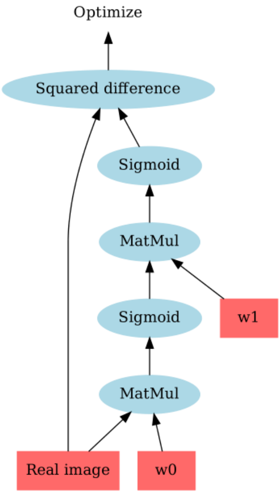

Optimize
Squared difference
Sigmoid
MatMul
Sigmoid
w1
MatMul
Real image
w0

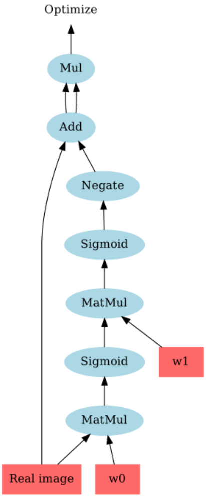

x1
x2
x3
x4
x5
z1
z2
z3
x'1
x'2
x'3
x'4
x'5


Optimize
Squared difference
Sigmoid
MatMul
w1
Sigmoid
MatMul
Real image
w0


Optimize
Mul
Add
Negate
Sigmoid
MatMul
w1
Sigmoid
MatMul
Real image
w0



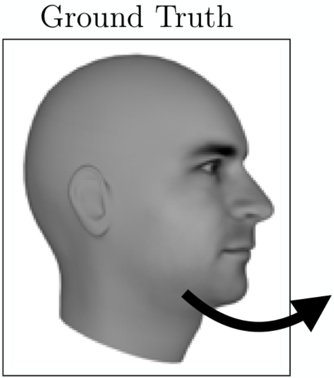




Things to note

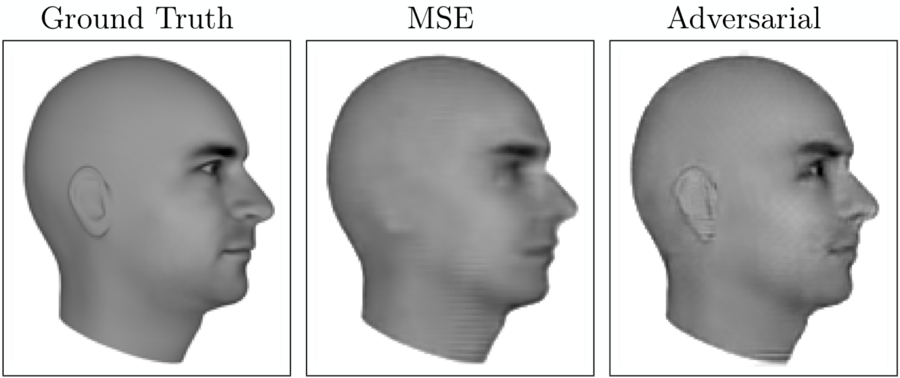
1. Cost function is not parametrized (it's fixed)
2. We train the network with just one cost function
3. We train the network monolithically

Cost function



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

Cost function

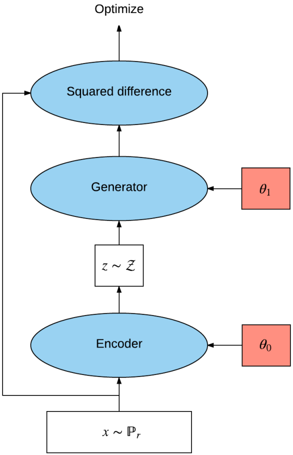


- MSE needs to average over all possibilities and choose a single answer
- Square error - a simple parabola, is it sufficient?

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

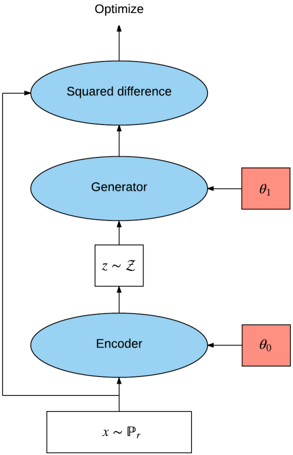
Autoencoder - perspective shift

- Autoencoder has an encoder and decoder



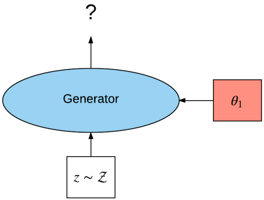
Autoencoder - perspective shift

- Autoencoder has an encoder and decoder
- Decoder = generator!



Autoencoder - perspective shift

- Autoencoder has an encoder and decoder
- Decoder = generator!



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
- Fixing a cost function has some undesired properties

The great idea

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

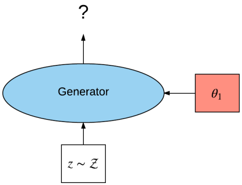
- ...
- ???

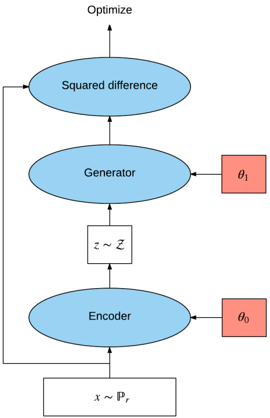
The great idea

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

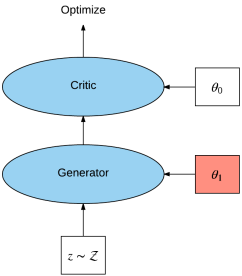
- ...
- ???
- Introduces a lot of problems

Training regime - main concept

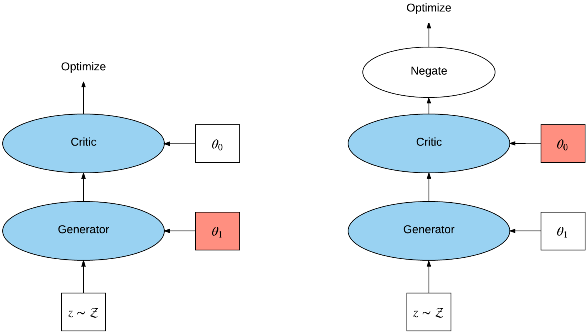



Optimize
Squared difference
Generator
θ_1
$z \sim \mathcal{Z}$
Encoder
θ_0
$x \sim \mathbb{P}_r$

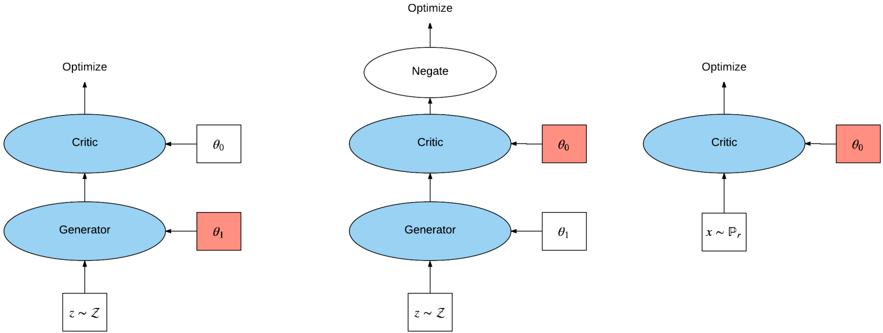
Training regime - main concept



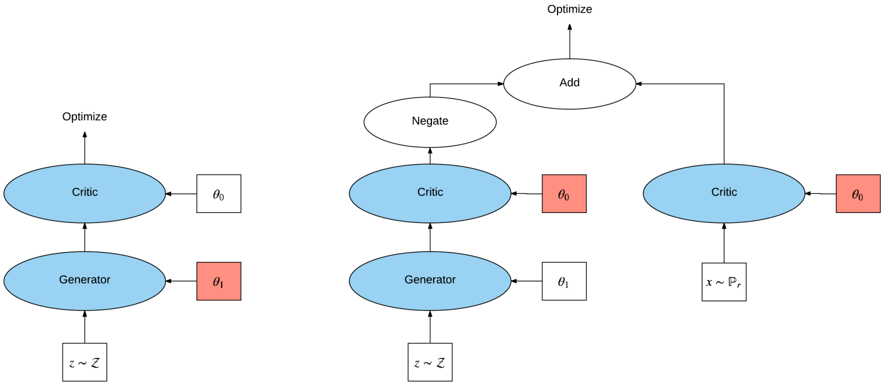
Training regime - main concept



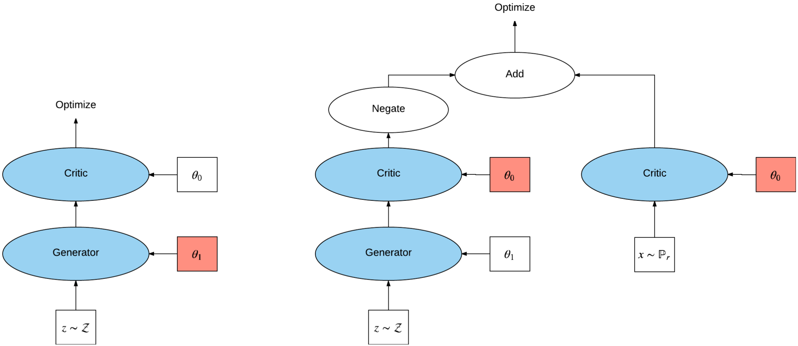
Training regime - main concept



Training regime - main concept



- Two-step optimization process



Require: initial critic parameters w_0, initial generator parameters θ_0.

while θ has not converged **do**
    for $t = 1, ..., n_{\text{critic}}$ **do**
        for $i = 1, ..., m$ **do**
            Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$.
            $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
            $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
            $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
        end for
        $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2)$
    end for
    Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^{m} \sim p(\boldsymbol{z})$.
    $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
end while

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

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

Kullback-Leibler and Jensen Shannon divergence

KL-divergence

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

- Doesn't satisfy triangle inequality and symmetry

Kullback-Leibler and Jensen Shannon divergence

KL-divergence

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

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

Kullback-Leibler and Jensen Shannon divergence

KL-divergence

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

- 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

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

- 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

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

- 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

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

- 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

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

- 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

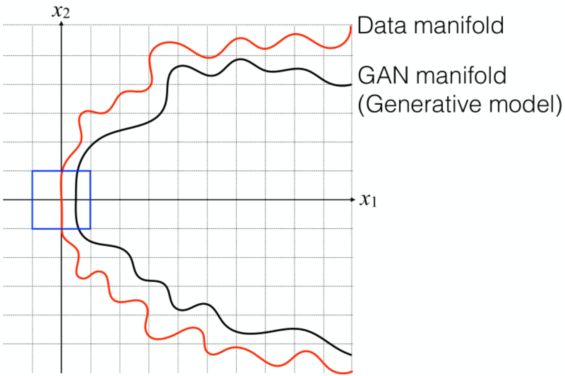
Wasserstein (Earth mover) distance

EM distance

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

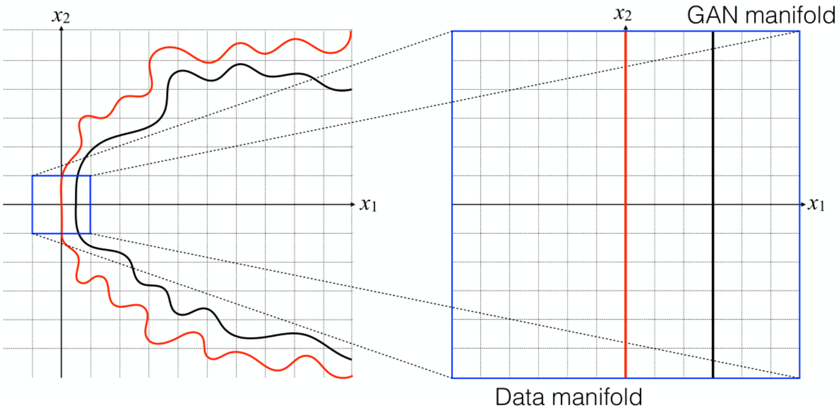
- Distance function that takes into account underlying geometry of the distributions
- Minimum cost of turning a pile of dirt into the other
- Proposed by Arjovsky *et al.* as improvement to original GAN training - Wasserstein GAN
- Intractable

Learning distributions - toy example



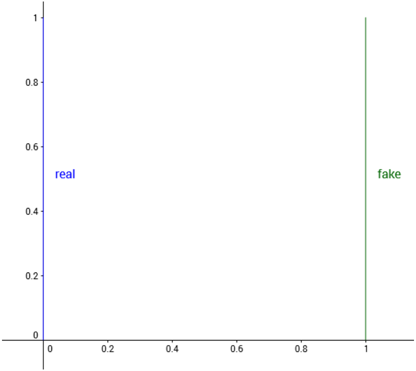
[0]MILA DLSS: https://drive.google.com/file/d/0B_wzP_JlVFcKQ21udGpTSkh0aVk/view

Learning distributions - toy example



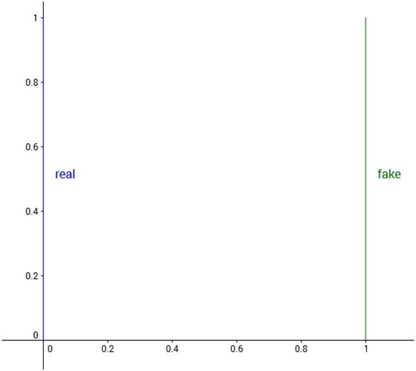
[0]MILA DLSS: https://drive.google.com/file/d/0B_wzP_JlVFcKQ21udGpTSkh0aVk/view

Learning distributions - toy example



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

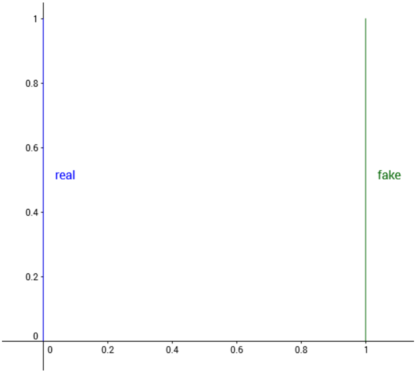
Learning distributions - toy example



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

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

Learning distributions - toy example

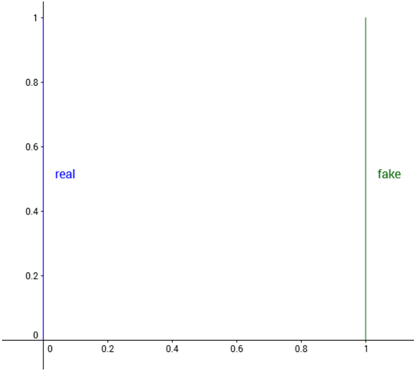


$$\mathbb{P}_r = (0, y) \qquad \mathbb{P}_g = (\theta, y)$$

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

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

Learning distributions - toy example



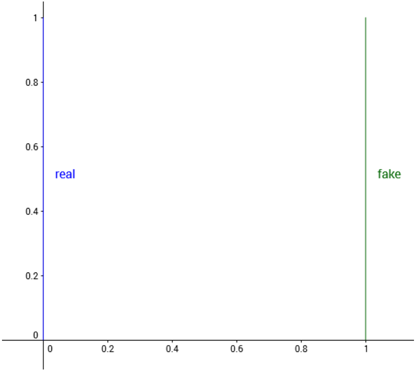
$$\mathbb{P}_r = (0, y) \qquad \mathbb{P}_g = (\theta, y)$$

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

- $KL(\mathbb{P}_r || \mathbb{P}_g) = +\infty$

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

Learning distributions - toy example



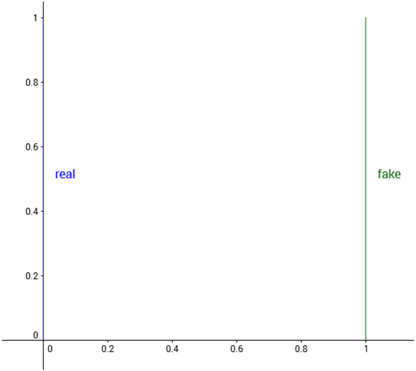
- $KL(\mathbb{P}_r||\mathbb{P}_g) = +\infty$
- $KL(\mathbb{P}_g||\mathbb{P}_r) = +\infty$

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

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

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

Learning distributions - toy example



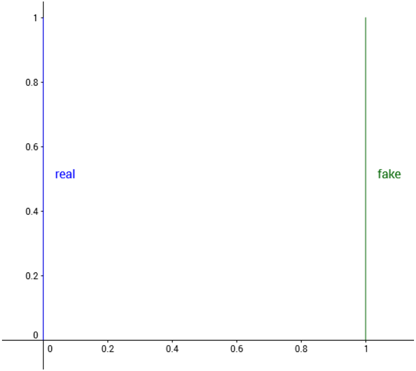
$$\mathbb{P}_r = (0, y) \qquad \mathbb{P}_g = (\theta, y)$$

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

- $KL(\mathbb{P}_r || \mathbb{P}_g) = +\infty$
- $KL(\mathbb{P}_g || \mathbb{P}_r) = +\infty$
- $JS(\mathbb{P}_r, \mathbb{P}_g) = JS(\mathbb{P}_g, \mathbb{P}_r) = \log 2$

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

Learning distributions - toy example



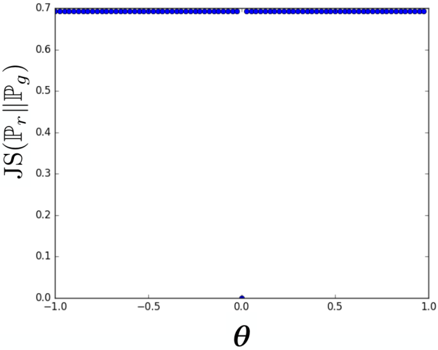
- $KL(\mathbb{P}_r||\mathbb{P}_g) = +\infty$
- $KL(\mathbb{P}_g||\mathbb{P}_r) = +\infty$
- $JS(\mathbb{P}_r, \mathbb{P}_g) = JS(\mathbb{P}_g, \mathbb{P}_r) = \log 2$
- $W(\mathbb{P}_r, \mathbb{P}_g) = W(\mathbb{P}_g, \mathbb{P}_r) = |\theta|$

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

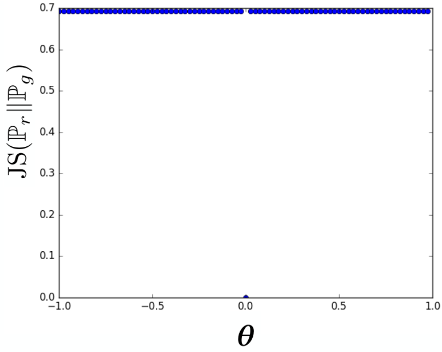
$$y \sim U[0, 1]$$

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

JS and EM distance w.r.t. θ

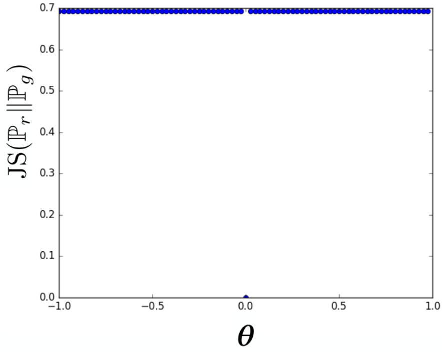


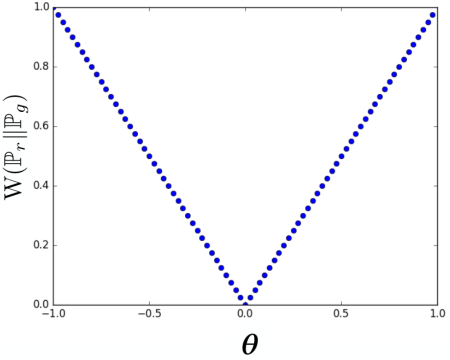
JS and EM distance w.r.t. θ



- JS-divergence gradient is zero

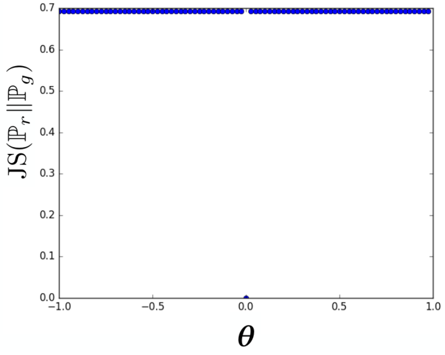
JS and EM distance w.r.t. θ

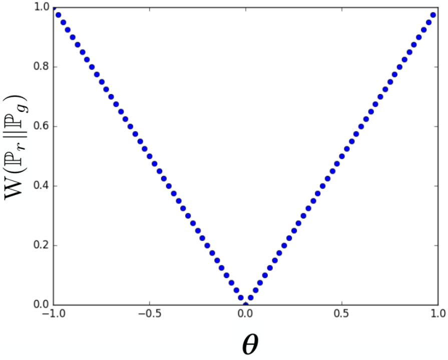




- JS-divergence gradient is zero

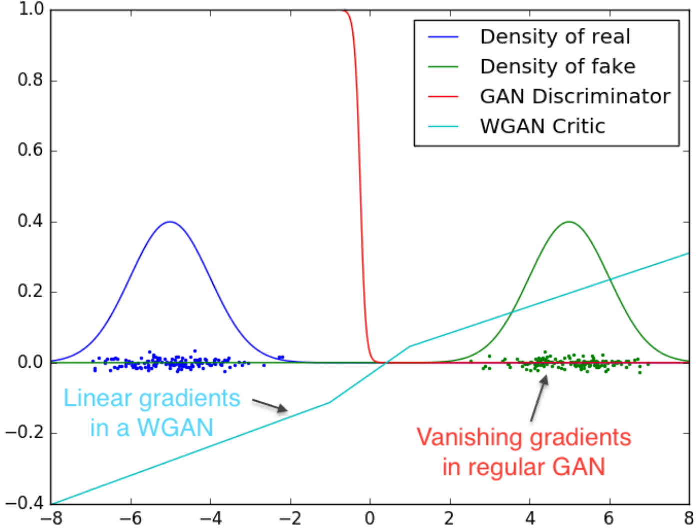
JS and EM distance w.r.t. θ



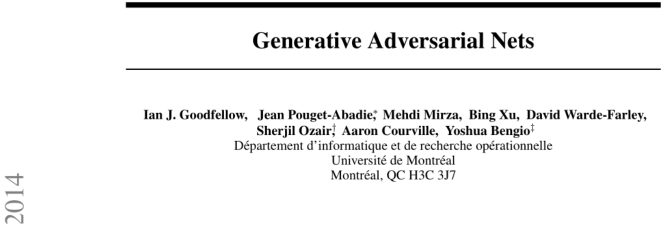


- JS-divergence gradient is zero
- Wasserstein gradient is constant

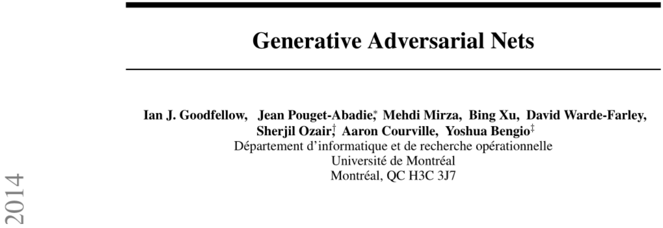
Gradients between two gaussian distributions



Short history of training GANs - July 2014



Short history of training GANs - July 2014



GAN Value function

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

- 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

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

We’re ready to put all the pieces together.

[0] https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html

- Under an optimal discriminator, GANs minimize JS-divergence.

[0] https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html

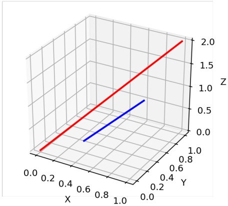
- Under an optimal discriminator, GANs minimize JS-divergence.
- If the manifolds don't overlap and we use JS-divergence, we get vanishing gradients

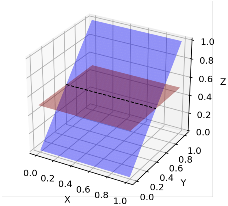
[0] https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html

- Under an optimal discriminator, GANs minimize JS-divergence.
- If the manifolds don't overlap and we use JS-divergence, we get vanishing gradients
- We're modelling natural data which is embedded on a low-dimensional manifold

[0] https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html

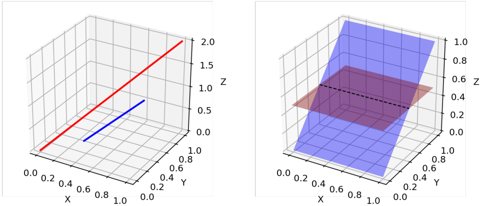
- Under an optimal discriminator, GANs minimize JS-divergence.
- If the manifolds don't overlap and we use JS-divergence, we get vanishing gradients
- We're modelling natural data which is embedded on a low-dimensional manifold
- Intersection of low dimensional manifolds in high-dimensional space has negligible support





0 https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html

- Under an optimal discriminator, GANs minimize JS-divergence.
- If the manifolds don't overlap and we use JS-divergence, we get vanishing gradients
- We're modelling natural data which is embedded on a low-dimensional manifold
- Intersection of low dimensional manifolds in high-dimensional space has negligible support



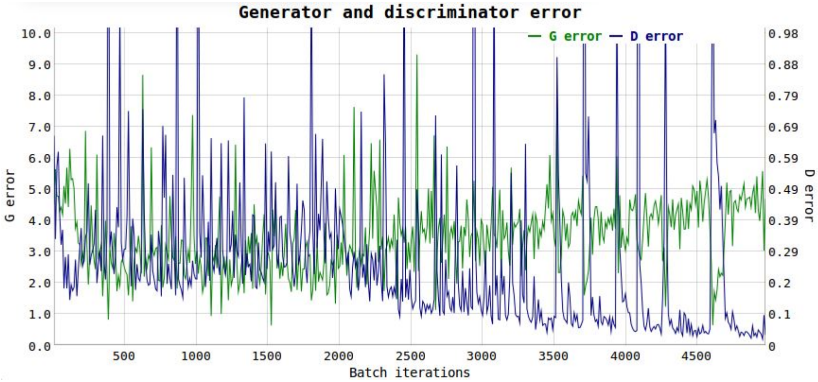
Conclusion!

Original GAN training regime on natural data leads to vanishing gradients

---

[0] https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html

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


BAG OF
TRICKS

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

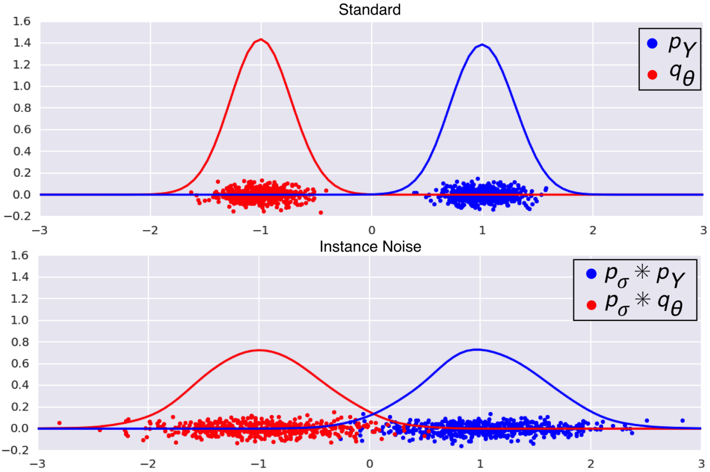
Trick - not training the discriminator until convergence

- 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

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

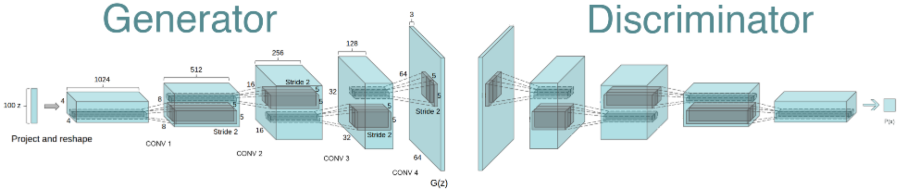
(TFD) [28], and CIFAR-10 [21]. The generator nets used a mixture of rectifier linear activations [19, 9] and sigmoid activations, while the discriminator net used maxout [10] activations. Dropout [17] was applied in training the discriminator net. While our theoretical framework permits the use of dropout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

UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS

Alec Radford & Luke Metz
indico Research



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

Trick - cherrypicking architecture - DCGAN

UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS

Alec Radford & Luke Metz
indico Research

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

UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS

Alec Radford & Luke Metz
indico Research

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.

- Authors reports that they had model generator [1]

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

Trick - cherrypicking architecture - DCGAN

UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS

Alec Radford & Luke Metz
indico Research

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.

- Authors reports that they had model generator [1]
- No explanation for model performance, very unstable

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

Short history of training GANs

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

Short history of training GANs

EM distance

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

[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?

[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

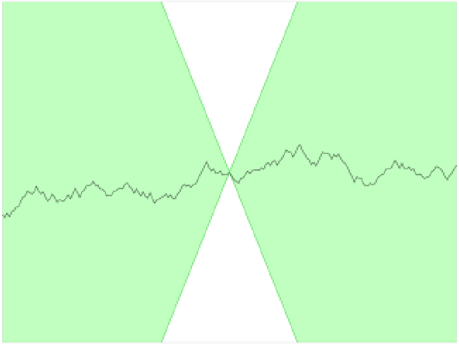
Kantorovich-Rubinstein duality

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

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

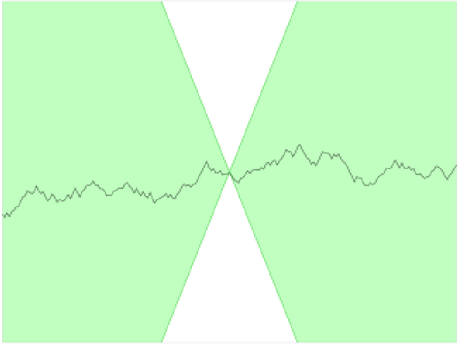
[2] `http://cedricvillani.org/wp-content/uploads/2012/08/preprint-1.pdf`

Lipschitz continuity



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

Lipschitz continuity



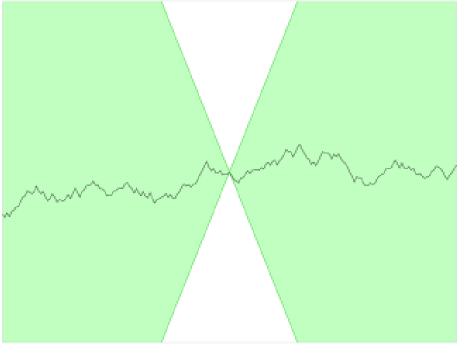
- Continuous function which is limited how fast it can change

Lipschitz continuous function

$$|f'(x)| \leq 1 \qquad \forall x \in \mathbb{R}$$

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

Lipschitz continuity



- Continuous function which is limited how fast it can change

K-Lipschitz continuous function

$$|f'(x)| \leq K \qquad \forall x \in \mathbb{R}$$

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

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 - 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

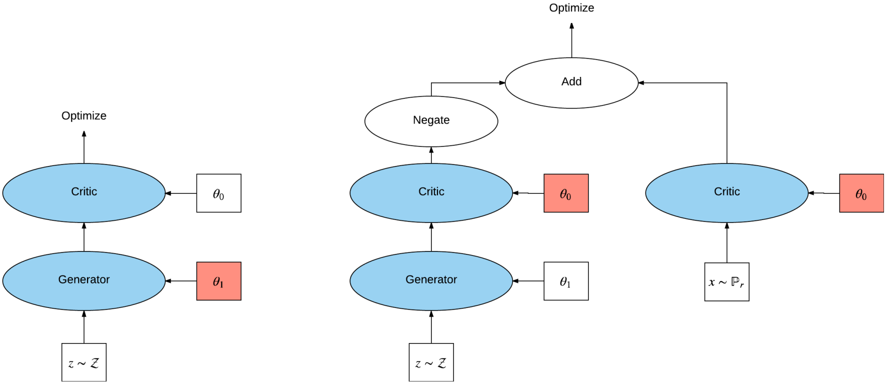
WGAN Value function

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

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

- Open question - how to effectively enforce the Lipschitz constraint?

Wasserstein GAN

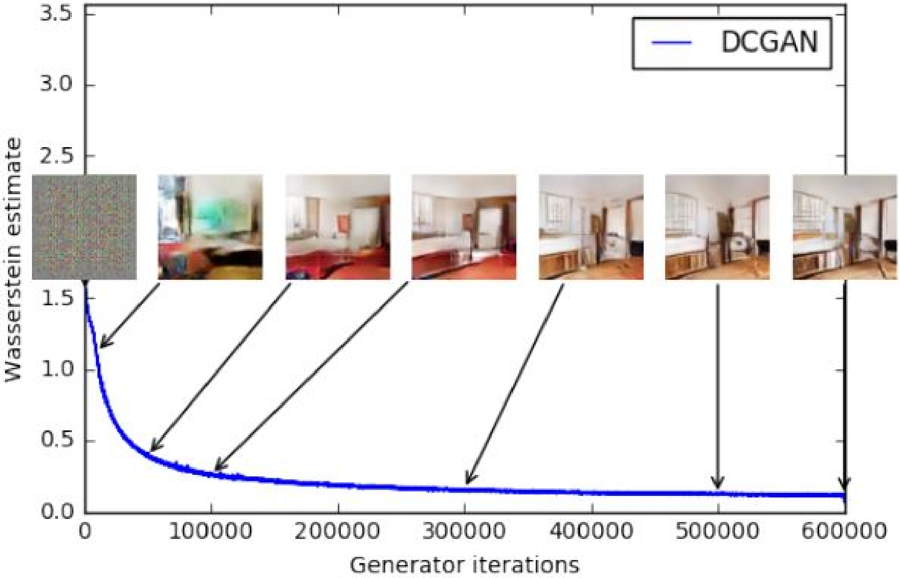


WGAN Value function

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

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

Meaningful loss function



Method #1 - Weight Clipping

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

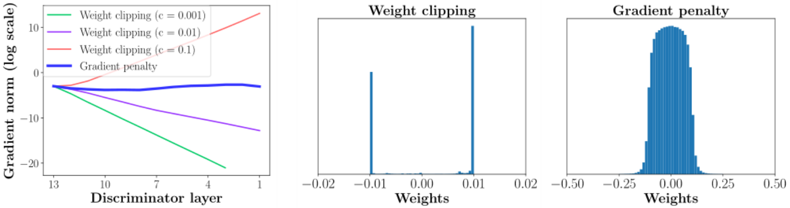
Method #1 - Weight Clipping

- 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

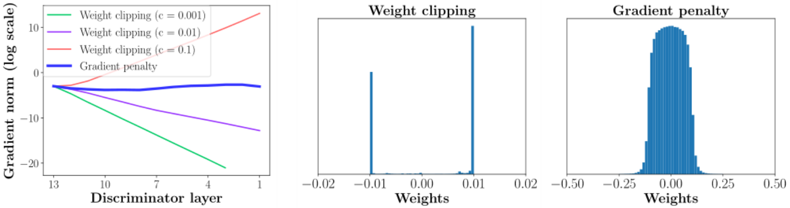
- After optimization step, clip all weights to $[-c, c]$
- Results in critic being a subset of K-Lipschitz functions, where K is a function of c and the critic's architecture



- Works! But...

Method #1 - Weight Clipping

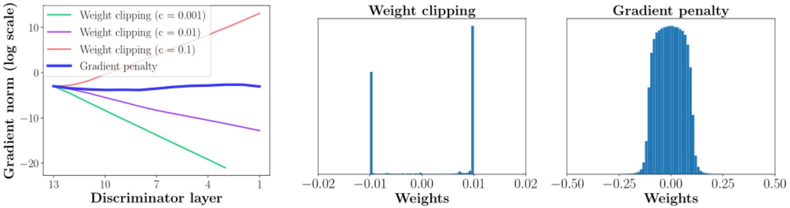
- After optimization step, clip all weights to $[-c, c]$
- Results in critic being a subset of K-Lipschitz functions, where K is a function of c and the critic's architecture



- 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

- Recall the Lipschitz constraint
  $|f'(x)| \leq 1$

Method #2 - Gradient Penalty

- Recall the Lipschitz constraint
  $|f'(x)| \leq 1$
- Optimal WGAN critic has gradient norm 1

Method #2 - Gradient Penalty

- Recall the Lipschitz constraint
  $|f'(x)| \leq 1$
- Optimal WGAN critic has gradient norm 1
- Add a regularization term to the WGAN value function

Method #2 - Gradient Penalty

- Recall the Lipschitz constraint $|f'(x)| \leq 1$
- Optimal WGAN critic has gradient norm 1
- Add a regularization term to the WGAN value function
- Sampling along straight lines

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

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

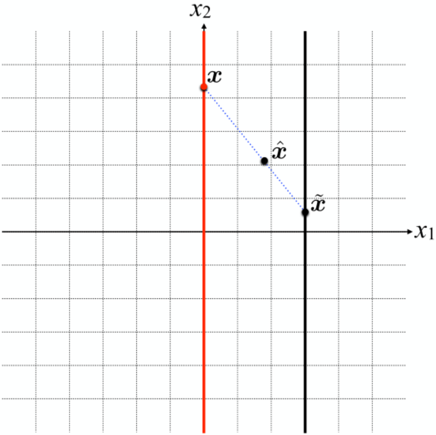
Method #2 - Gradient Penalty

- Recall the Lipschitz constraint $|f'(x)| \leq 1$
- 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



Method #2 - Gradient Penalty

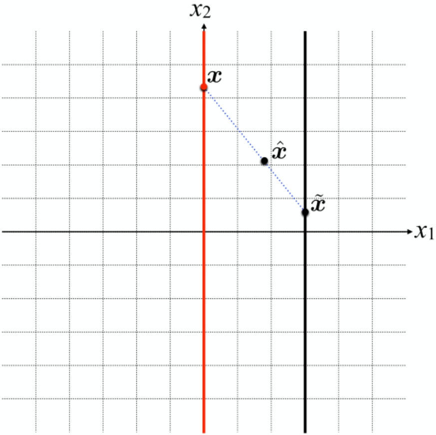
- Recall the Lipschitz constraint $|f'(x)| \leq 1$
- Optimal WGAN critic has gradient norm 1
- Add a regularization term to the WGAN value function
- Sampling along straight lines

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

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

- Gradient penalty

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



Method #2 - Gradient Penalty

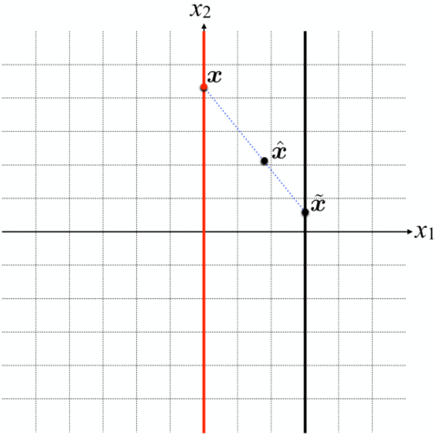
- Recall the Lipschitz constraint $|f'(x)| \leq 1$
- Optimal WGAN critic has gradient norm 1
- Add a regularization term to the WGAN value function
- Sampling along straight lines

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

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

- Gradient penalty

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



Method #2 - Gradient Penalty

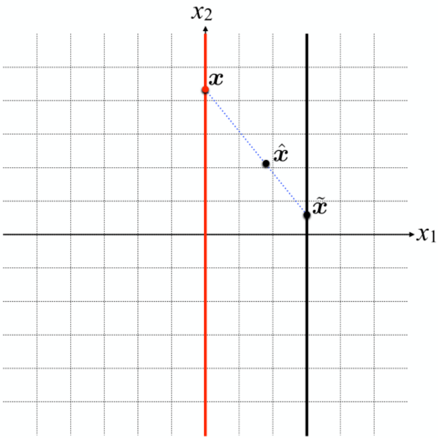
- Recall the Lipschitz constraint $|f'(x)| \leq 1$
- Optimal WGAN critic has gradient norm 1
- Add a regularization term to the WGAN value function
- Sampling along straight lines

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

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

- Gradient penalty

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



Method #2 - Gradient Penalty

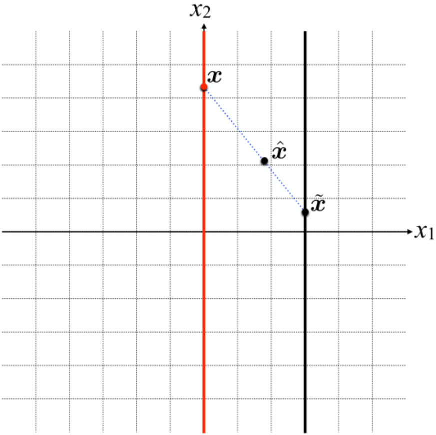
- Recall the Lipschitz constraint $|f'(x)| \leq 1$
- Optimal WGAN critic has gradient norm 1
- Add a regularization term to the WGAN value function
- Sampling along straight lines

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

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

- Gradient penalty

$$\left(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1\right)^2$$



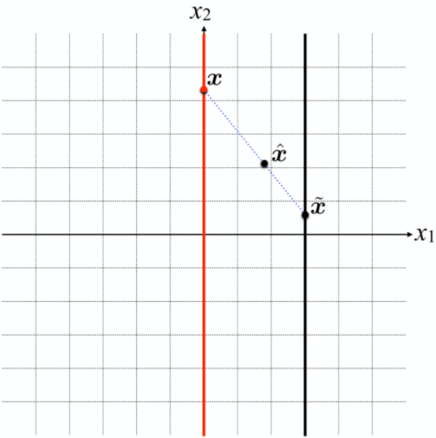
Method #2 - Gradient Penalty

- Recall the Lipschitz constraint $|f'(x)| \leq 1$
- 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 $\mathbb{E}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}}\left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2\right]$



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

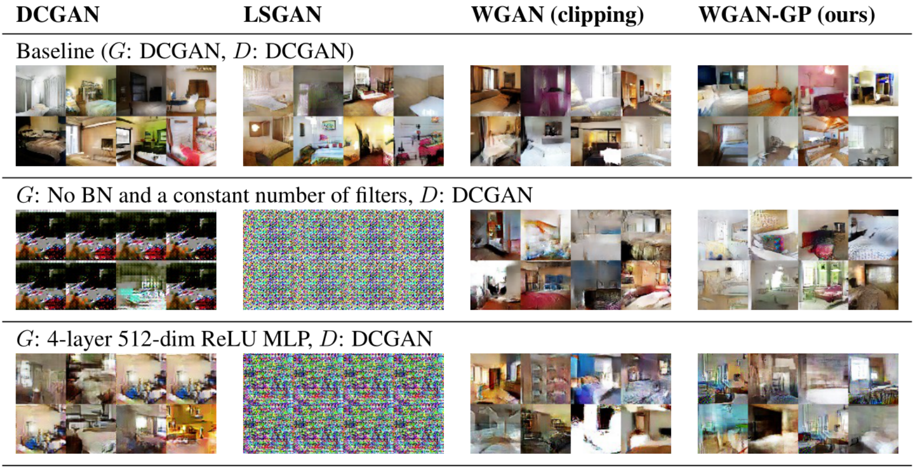
Ishaan Gulrajani[1], **Faruk Ahmed**[1], **Martin Arjovsky**[2], **Vincent Dumoulin**[1], **Aaron Courville**[1,3]
[1] Montreal Institute for Learning Algorithms
[2] Courant Institute of Mathematical Sciences
[3] CIFAR Fellow
igul222@gmail.com
{faruk.ahmed,vincent.dumoulin,aaron.courville}@umontreal.ca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

WGAN-GP Value function

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

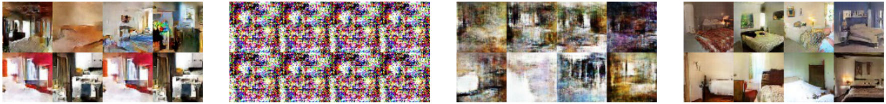
- Enforcing the Lipschitz constraint with a gradient penalty regularization term

WGAN-GP results



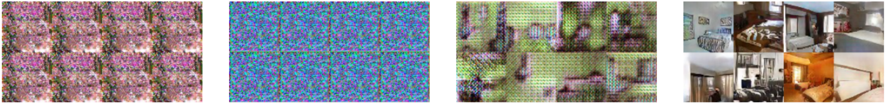
WGAN-GP results

Gated multiplicative nonlinearities everywhere in G and D

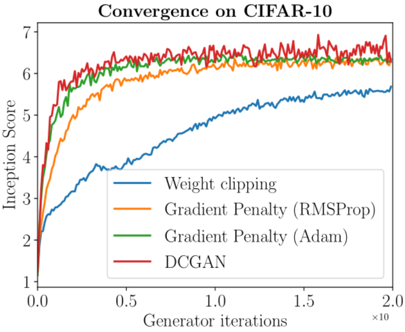


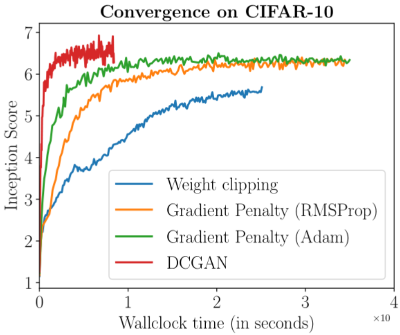
$tanh$ nonlinearities everywhere in G and D

101-layer ResNet G and D



WGAN-GP results

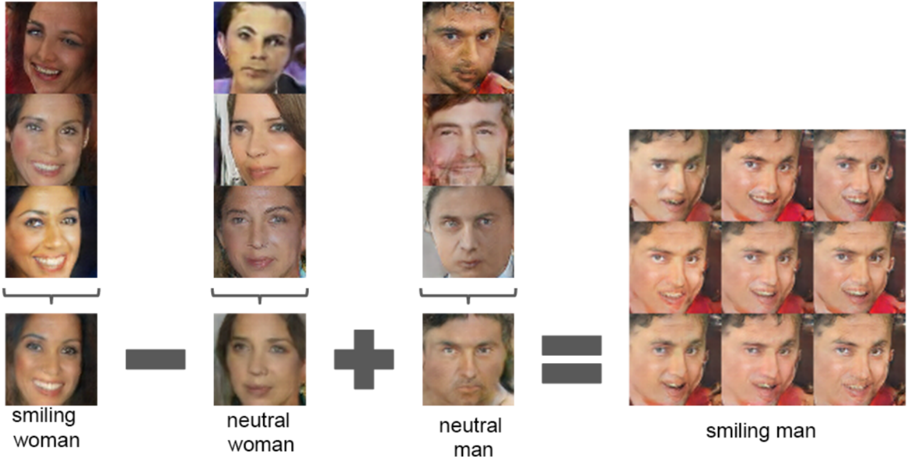




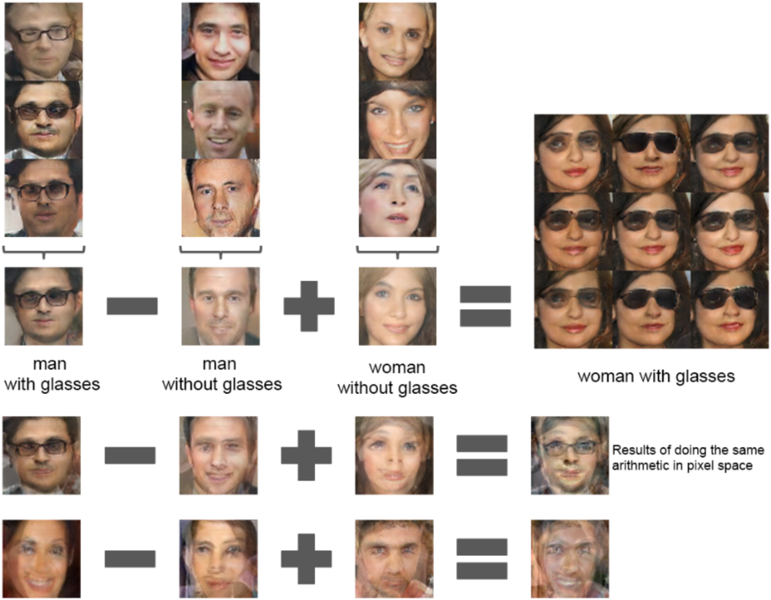
- DCGAN converges faster
- Significantly outperforms weight clipping
- **Robust to changes in model architecture**

Cool things you can do with GANs

Latent space arithmetic



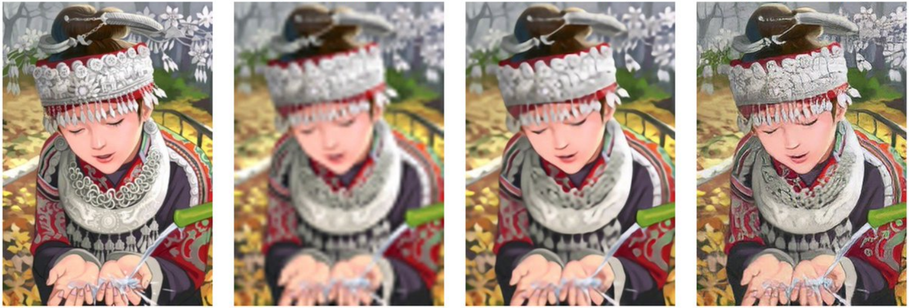
Latent space arithmetic

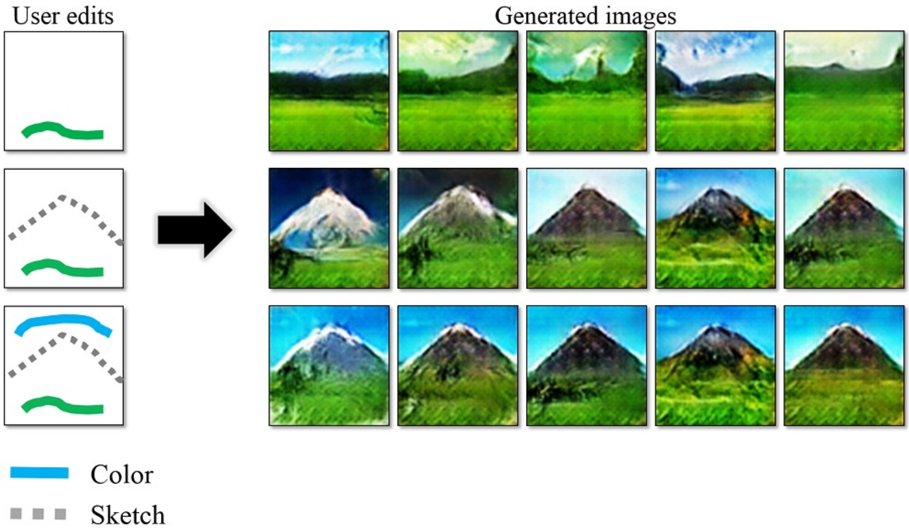


Interpolation between images

Super-Resolution

original | bicubic (21.59dB/0.6423) | SRResNet (23.44dB/0.7777) | SRGAN (20.34dB/0.6562)



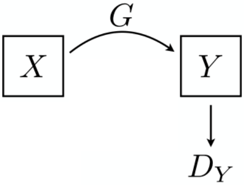


▸ Interactive GAN

Style Transfer

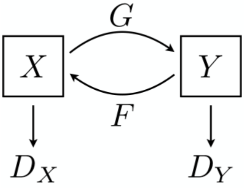
$\boxed{X}$ $\quad\quad\quad$ $\boxed{Y}$

Style Transfer

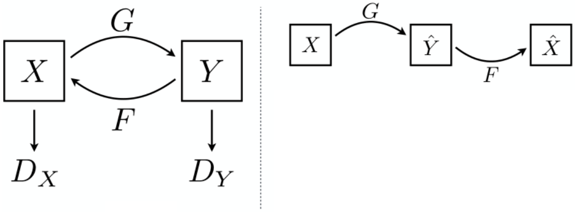


$$X \xrightarrow{G} Y \rightarrow D_Y$$

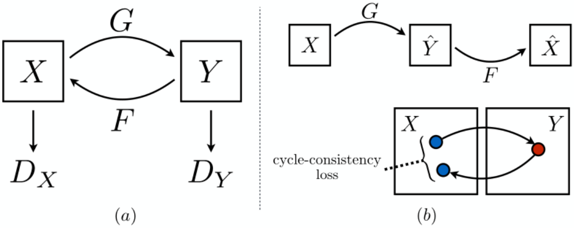
Style Transfer



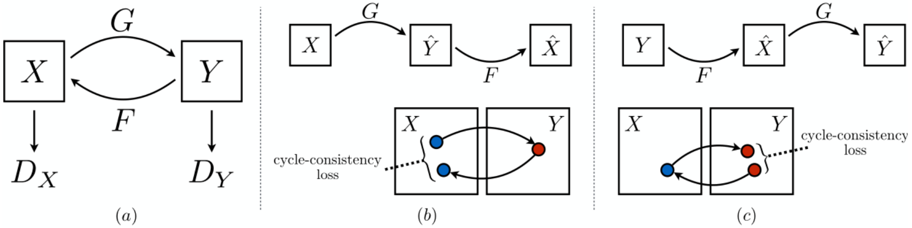
Style Transfer



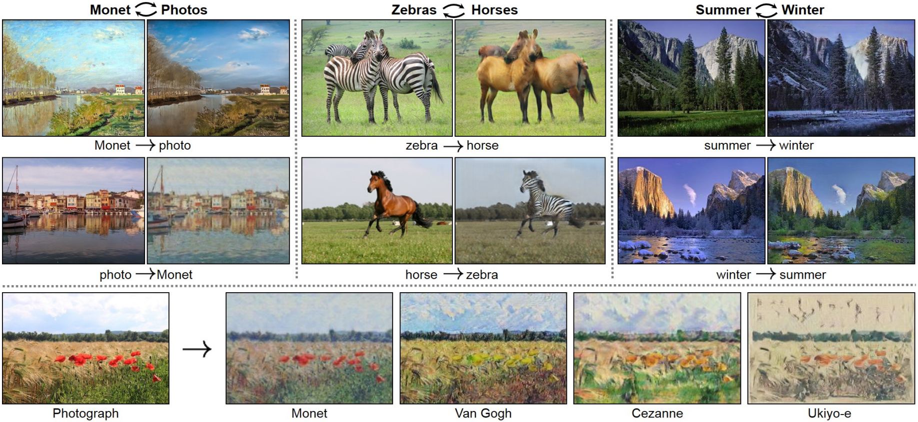
Style Transfer



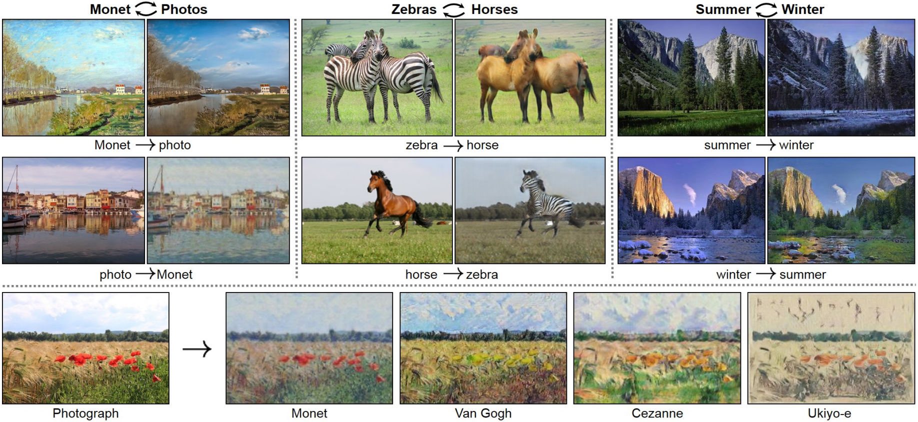
Style Transfer



Style Transfer



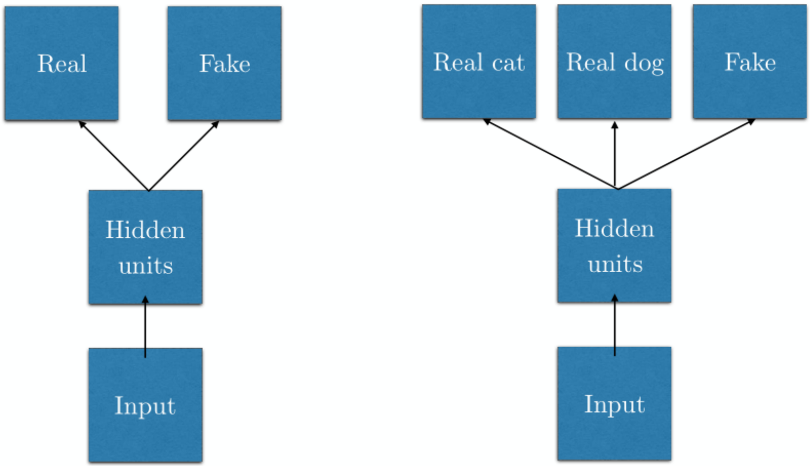
Style Transfer



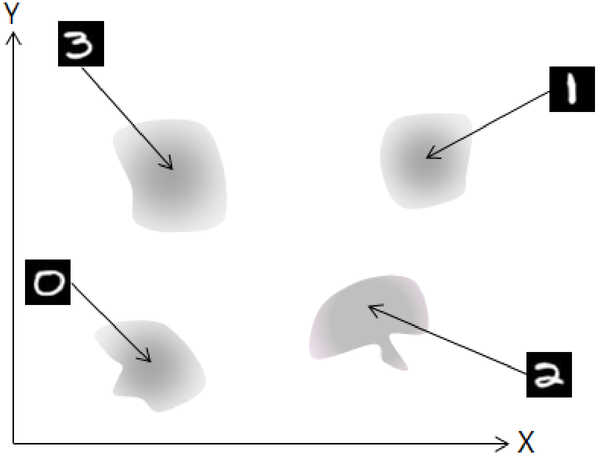
▸ CycleGAN in action ▸ Creepy CycleGAN

Failure case

Supervised discriminator



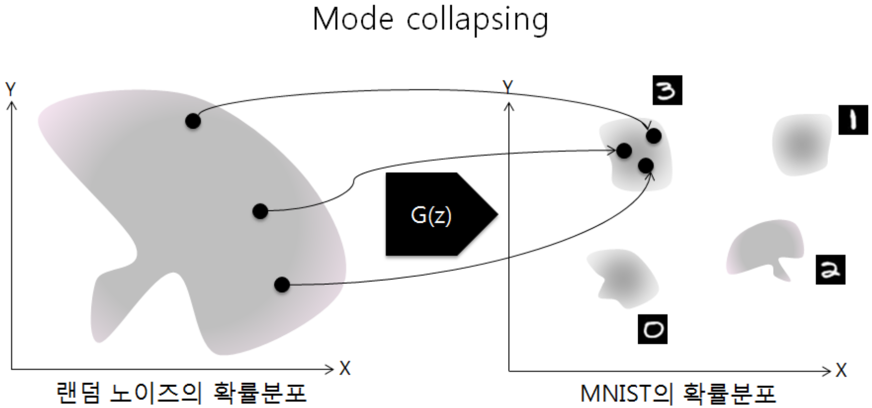
Problems with GANs - Mode collapse



---

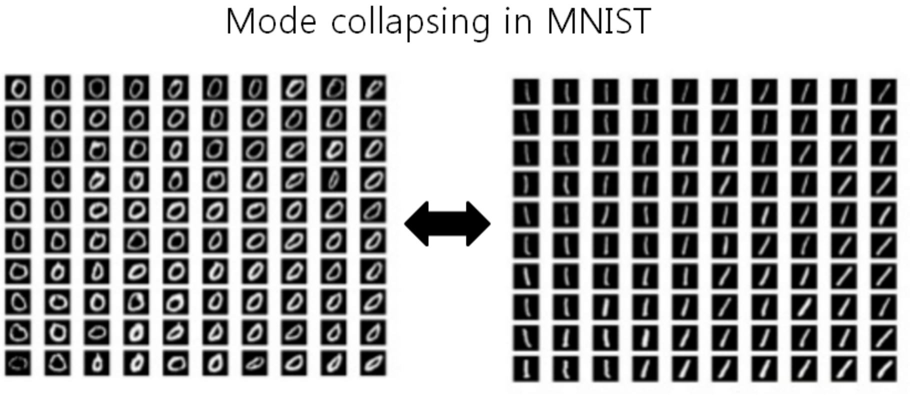
[2] http://dl-ai.blogspot.com/2017/08/gan-problems.html

Problems with GANs - Mode collapse



[2] http://dl-ai.blogspot.com/2017/08/gan-problems.html

Problems with GANs - Mode collapse



[2] http://dl-ai.blogspot.com/2017/08/gan-problems.html

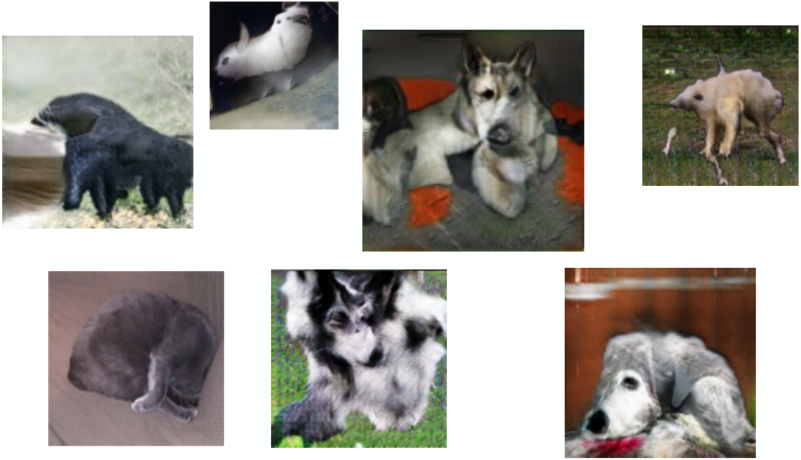
Problems with GANs - Counting






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

Problems with GANs - Perspective



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

What we didn’t talk about

- Performance measures (Inception score, FID)

What we didn't talk about

- Performance measures (Inception score, FID)
- Spectral normalization

What we didn't talk about

- Performance measures (Inception score, FID)
- Spectral normalization
- Game theoretic perspective

What we didn’t talk about

- Performance measures (Inception score, FID)
- Spectral normalization
- Game theoretic perspective
- GANs that use other divergences (Coloumb GAN, Fischer GAN, Cramer GAN...)

What we didn’t talk about

- Performance measures (Inception score, FID)
- Spectral normalization
- Game theoretic perspective
- GANs that use other divergences (Coloumb GAN, Fischer GAN, Cramer GAN...)
- Clever tricks for making your GANs work

What we didn't talk about

- Performance measures (Inception score, FID)
- Spectral normalization
- Game theoretic perspective
- GANs that use other divergences (Coloumb GAN, Fischer GAN, Cramer GAN...)
- Clever tricks for making your GANs work
- Applications of GANs

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

Cool links

- https://vincentherrmann.github.io/blog/wasserstein/
- https://www.alexirpan.com/2017/02/22/wasserstein-gan.html
- https://jeremykun.com/2018/03/05/earthmover-distance/
- https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html

Thank you!

Bruno Gavranović
Faculty of Electrical Engineering and Computing
University of Zagreb
bruno.gavranovic@fer.hr

Feel free to drop me an email with any questions!