

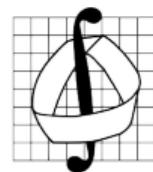
# Introduction to Machine and Deep Learning Theory

## Generative Adversarial Networks

Aleksandr Petiushko

Lomonosov MSU  
Faculty of Mechanics and Mathematics

February 22, 2023



# Content

- ➊ Purpose of a generative model
- ➋ Discriminate vs generative models
- ➌ GANs — Generative Adversarial Networks
- ➍ DCGAN
- ➎ Wasserstein's metric
- ➏ Conditional GAN

# Motivation

## Problem1

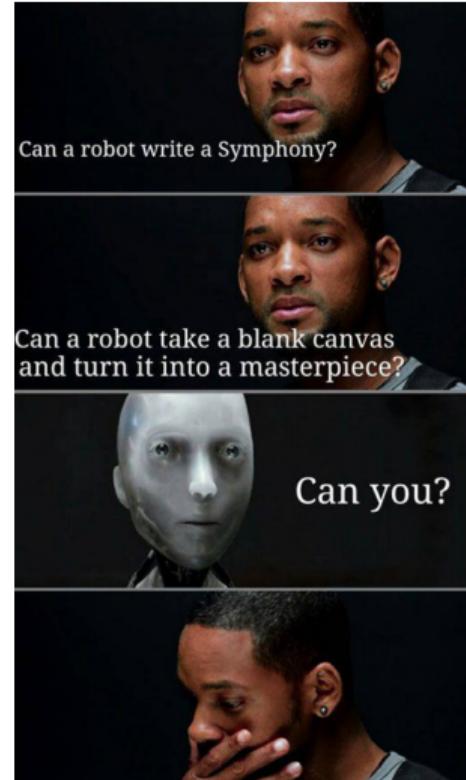
For supervised learning and/or robust evaluation of modern AI solutions we need an enormous amount of data (and very clean markup of this data!). E.g. it is millions of images, hundreds and thousands of hours of videos.

- Every new task usually requires the collection of the new dataset (as there is no the universal one), or adding the new attributes markup using the old raw data — and the cost of such procedures is huge (\$\$\$)
- Also some companies can have such volumes of data due to their business model (Google, Facebook, Twitter etc)

# Motivation

## Problem2

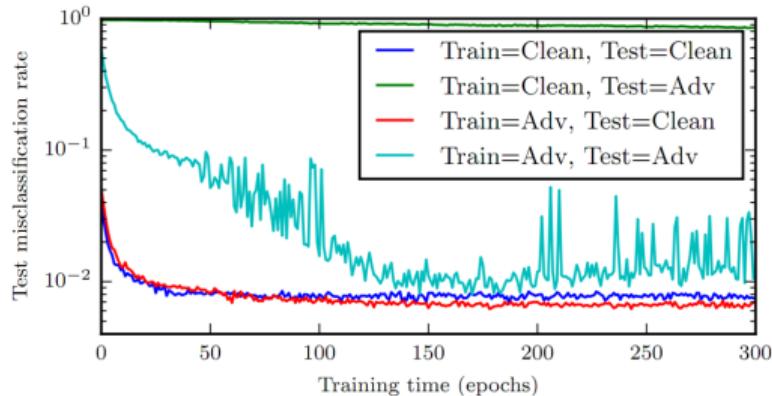
Why robots are not making music, poems, paintings?



# Preferred outcome

- Having solved **Problem1** our wish could be: adding the generated data into the training data improves the eval metrics
- Having solved **Problem2** our wish could be: generate the data indistinguishable from real world data

## Training on Adversarial Examples



# Idea

## Assumption

Suppose having some real data from distribution  $p_{data}$ . Probably the number of examples in it is not enough for training a good model (e.g., a classifier), but at the same we can assume, that this data sample reflects to some extent the target data distribution.

## Suggestion

Let us use so called “generative models” which are capable of creating the data from the real data distribution  $p_{data}$ .

# About generative models functionality

## Important notes:

- ① Let us consider image data
- ② Image distribution is defined on a *discrete* support (assuming 8-bit color encoding) of **enormous cardinality**  $\Rightarrow$  the common procedure of sampling from distribution  $p_{data}$  is not realistic (or even inferring it using some analytical formulas)
- ③ Diversity of generated data is coming from the means of generative process: the generative model will be using as an input the vector of random values from some very simple distribution (e.g.  $z \sim N(0, 1)$ )

# Generative and Discriminate models

- ① **Discriminate** model — the model providing the conditional probability of label (e.g. whether the data is real or not)  $y$  conditioned on data  $x$  (e.g. generated  $x \sim p_{gen}$  and real  $x \sim p_{data}$ ):  $P(Y|X = x)$
- ② **Generative** model — the model of joint probability of data and its label:  $P(X, Y)$

**Remark.** Apply the conditional probability formula:

$$\arg \max_x P(X, Y = y) = \arg \max_x P(X|Y = y)P(Y = y) = \arg \max_x P(X|Y = y)$$

⇒ For generative model the equivalent model (considering the fixed labels):  $P(X|Y = y)$ .

# Variety of generative models

Main representatives of generative models:

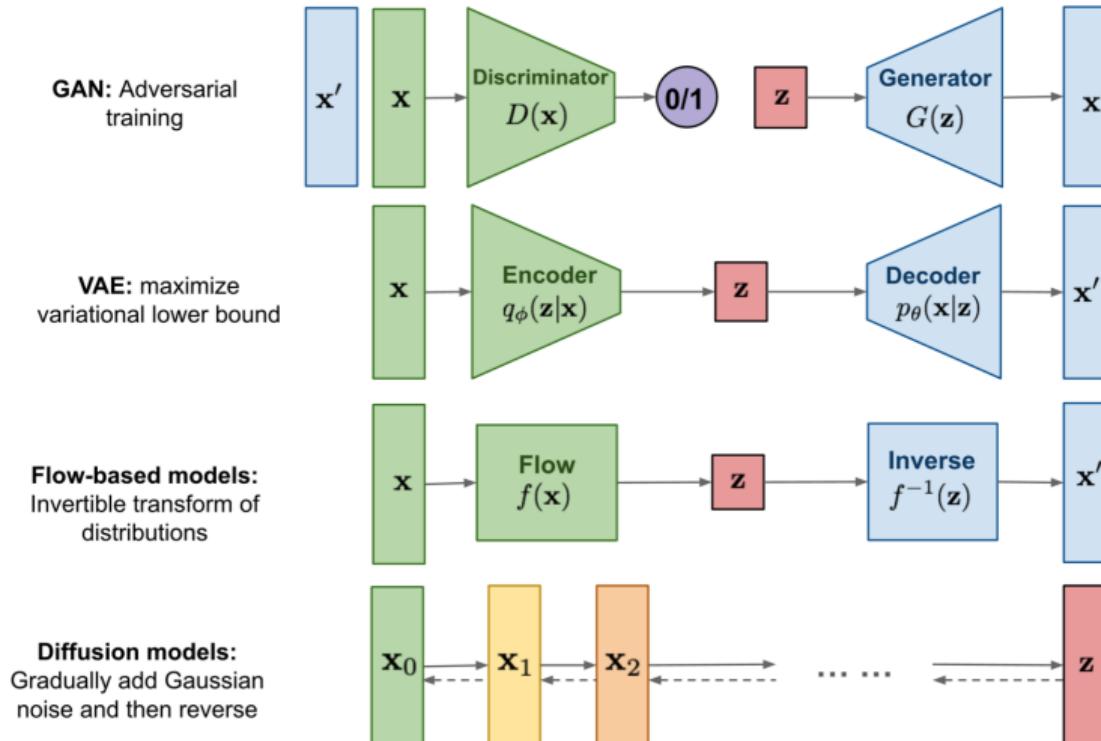
- Mixture models (e.g., GMM)
- Hidden Markov Models (HMMs)
- Bayesian Network (and Autoregressive Models)
- Flow-based Models
- Energy-based Models (EBMs)
- Diffusion Models
- Variational Autoencoder (VAE)
- Generative Adversarial Networks (GANs)

# Variety of generative models

Main representatives of generative models:

- Mixture models (e.g., GMM)
- Hidden Markov Models (HMMs)
- Bayesian Network (and Autoregressive Models)
- Flow-based Models
- Energy-based Models (EBMs)
- Diffusion Models
- Variational Autoencoder (VAE)
- **Generative Adversarial Networks (GANs)**

# Variety of generative models: illustration<sup>1</sup>



<sup>1</sup>Lil'log blog

# Generative Adversarial Network

## Main idea

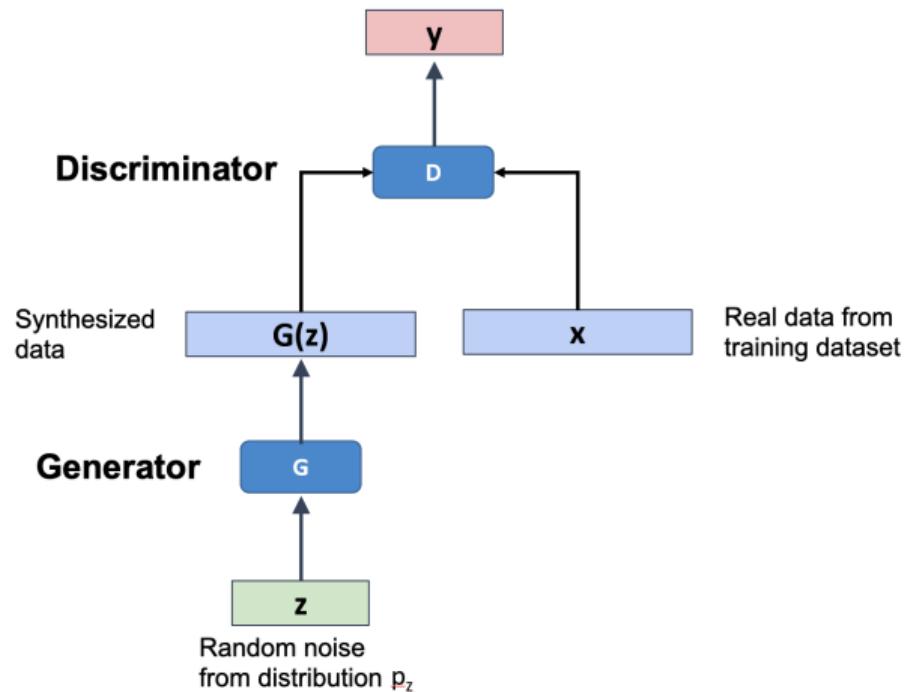
- No any explicit check on latent distribution of  $z$  (important: during train and eval the distribution of  $z$  should be the same)
- Additional module named **Discriminator**  $D$  is introduced
- Main aim of  $D$ : to check the realism of its input data (input to  $D$  can be either real or the output of the decoder called here **Generator**  $G(z)$ )

# Generative Adversarial Network

One more time, “GAN” (generative adversarial networks) — the (**G**enerative) model, which:

- Consists of 2 neural **N**ets having the opposite (**A**dversarial) goals
- Discriminator  $D$  is trying to check whether its input is “real” (i.e. very similar in distribution to the examples from training dataset) or “fake” (synthesized)
- Generator  $G$  is trying to synthesize the examples as close as possible in distribution to training examples

# GAN<sup>2</sup> scheme



Discriminator's goal:

- $D(x) \rightarrow 1$
- $D(G(z)) \rightarrow 0$

Generator's goal:

- $D(G(z)) \rightarrow 1$

<sup>2</sup>I. Goodfellow et al. “Generative Adversarial Networks”. 2014

# GAN's Discriminator

Discriminator  $D$ :

- Suppose Computer Vision usecase: usually the well-known classification neural net (VGG, ResNet, transformer) is used as a basis
- Discriminator's output  $y = D(x)$  — the value from the range  $[0, 1]$  showing the “trust” of discriminator in input data realism, i.e.
  - ▶  $D(x) = 1$  means that discriminator is 100% sure in realism of  $x$
  - ▶  $D(x) = 0$  means that discriminator is 100% sure in the synthetic nature of  $x$

# GAN's Generator

Generator  $G$ :

- Most interesting and non-trivial part
- Suppose Computer Vision usecase:  $G$  — convolutional neural net / transformer:
  - ▶ Input: random vector (not even image)  $z \in R^N$  from some simple distribution
  - ▶ Then by Fully-Connected / Reshape the input is transformed in 2D-image of much smaller size than training examples
  - ▶ After some layers increasing the spatial size are consequently applied (e.g., transposed convolutions)
  - ▶ Generator's  $G(z)$  output: tensor of the **exactly same size** as the real input  $x$

# Loss function in GAN

For GAN training the minimax optimization of the following loss function is used:

$$V(D, G) = \log D(x) \cdot (1 - D(G(z)))$$

## Why minimax?

- We would like the parameters of  $D$  to be of the following properties: on real data the  $D$  score is large, and on synthetic data the  $D$  score is small  $\Rightarrow \max_D V$
- We would like the parameters of  $G$  to be of the following property: on synthetic data the  $D$  score is large  $\Rightarrow \min_G V$

As a result, we have the following optimization task:

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

# GAN optimization procedure

## Minimax task solution

- Let us use staged training: on the first step to update the parameters of  $D$  by backpropagation, on the second step to update the parameters of  $G$ , then again  $D$ , then one more time  $G$  and so on
- For discriminator parameters update we need both real and synthetic data. At the same time the discriminator is updated by multiple iterations in a row
- Discriminator parameters update: by **gradient ascend** (because  $\max!$ )
- For generator parameters update we need only synthetic data, and loss function for the corresponding iteration is simpler:  $\min_G \mathbb{E}_{z \sim p_z} \log(1 - D(G(z)))$
- Generator parameters update: by **gradient descend**

# Original GAN training algorithm

---

**Algorithm 1** Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator,  $k$ , is a hyperparameter. We used  $k = 1$ , the least expensive option, in our experiments.

---

**for** number of training iterations **do**

**for**  $k$  steps **do**

- Sample minibatch of  $m$  noise samples  $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$  from noise prior  $p_g(\mathbf{z})$ .
- Sample minibatch of  $m$  examples  $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$  from data generating distribution  $p_{\text{data}}(\mathbf{x})$ .
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[ \log D(\mathbf{x}^{(i)}) + \log (1 - D(G(\mathbf{z}^{(i)}))) \right].$$

**end for**

- Sample minibatch of  $m$  noise samples  $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$  from noise prior  $p_g(\mathbf{z})$ .
- Update the generator by descending its stochastic gradient:

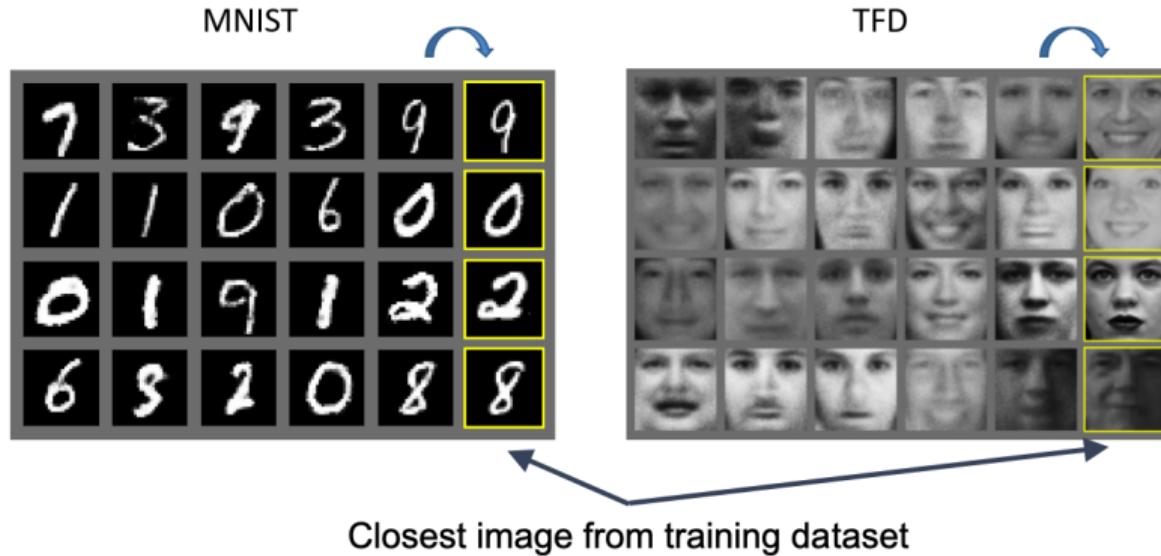
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log (1 - D(G(\mathbf{z}^{(i)}))).$$

**end for**

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

---

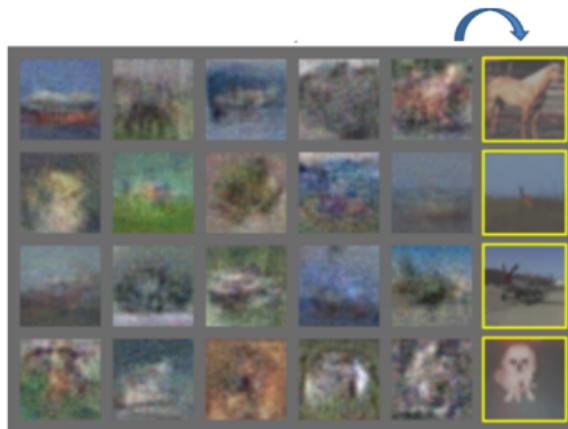
# Original GAN results on MNIST and faces<sup>3</sup>



<sup>3</sup>Toronto Face Dataset

# Original GAN results on CIFAR-10

Fully Connected D and G



D – CNN

G – Transposed convolutions



Closest image from training dataset

# Original GAN: remarks

- Quite unusual minimax algorithm
- Very unstable and unpredictable training
- Experiments show that GAN does not simply remember the training dataset
- Experiments show that (in case of images and convolutions) it is better to use transposed convolutions and not fully connected layers

# DCGAN<sup>5</sup>

DCGAN (**D**eep **C**onvolutional GAN) — one of the first successful GAN implementations<sup>4</sup>, providing the baseline for many projects:

- Discriminator: CNN with convolutions using strides  $s > 1$  (no maxpooling)
- Generator: CNN using transposed convolutions
- Batch normalization for both  $D$  and  $G$
- Generator uses activation function  $ReLU$ , discriminator uses  $Leaky\ ReLU$

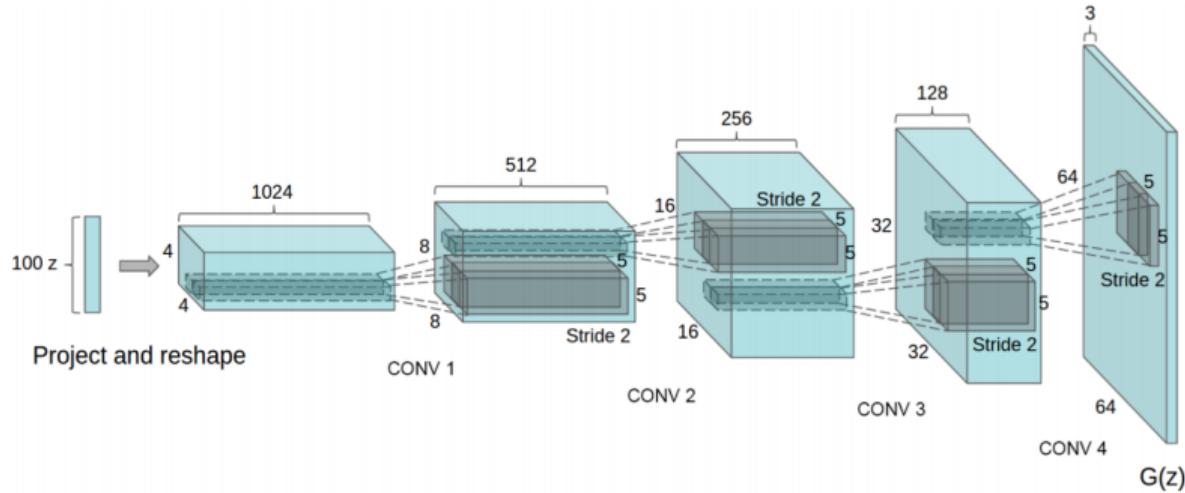


---

<sup>4</sup><https://github.com/carpedm20/DCGAN-tensorflow>

<sup>5</sup>A. Radford et al. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. 2015

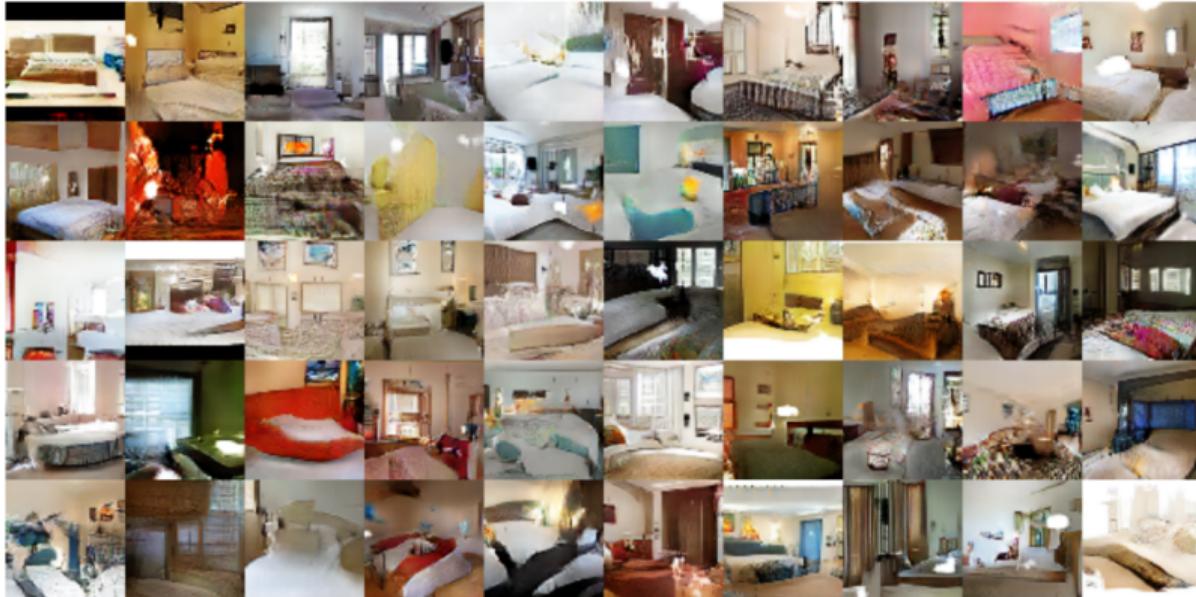
# DCGAN Generator



- Input: 100-dimensional noise vector  $z \sim N(0, 1)$  (or  $U[0, 1]$ ), output: RGB image  $64 \times 64$
- Scaling rule (lately to be adopted by many approaches, not only GAN): increasing by  $k$  times the spatial size leads to decreasing by  $k$  times the number of features maps

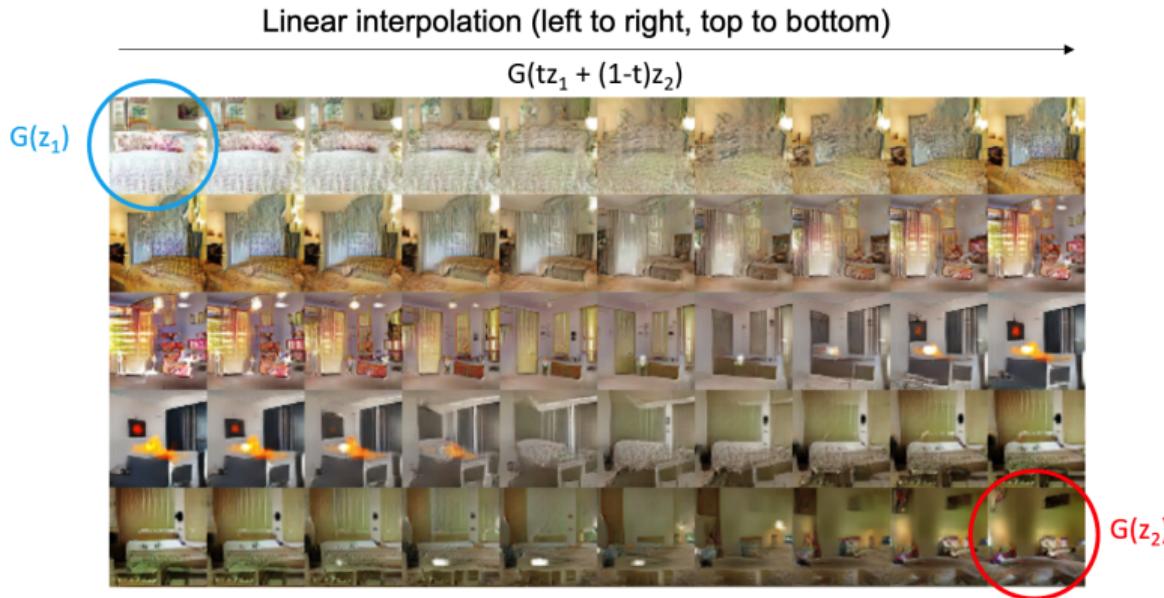
# DCGAN results

Results of training on LSUN<sup>6</sup>



<sup>6</sup><https://www.yf.io/p/lsun>

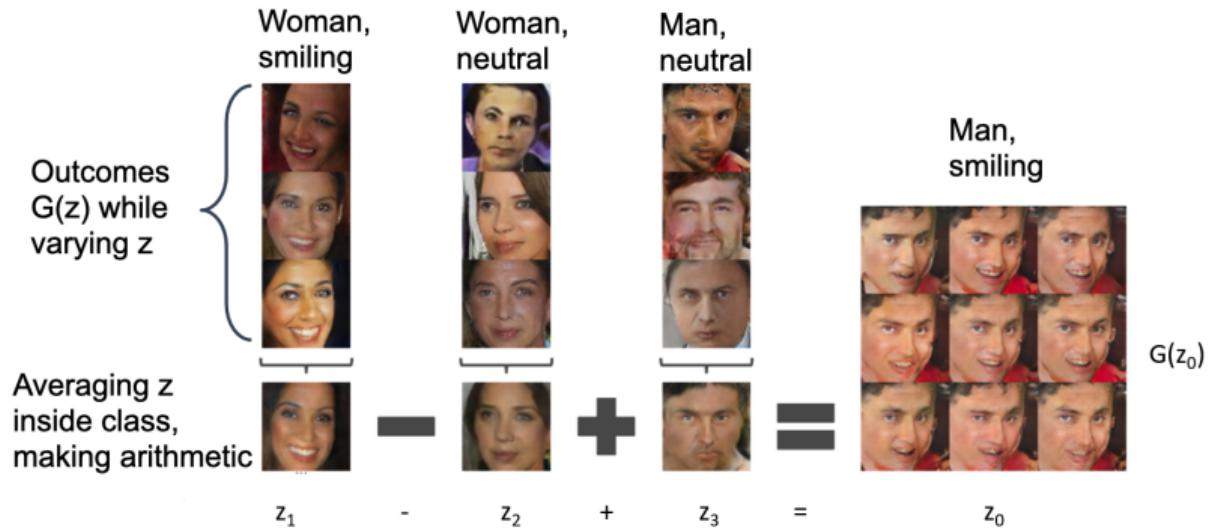
# Hidden (latent) representation interpolation



$$t \in [0, 1]$$

AP

# Hidden (latent) representation arithmetic



# GAN and optimal distributions (1)

Let us denote  $p_g$  as the distribution of  $G(z)$ . Then the original GAN optimization function

$$\min_G \max_D V(D, G) = \min_G \max_D [\mathbb{E}_{x \sim p_{data}} \log D(x) + \mathbb{E}_{z \sim p_g} \log(1 - D(G(z)))]$$

can be re-written as

$$\min_G \max_D V(D, G) = \min_G \max_D [\mathbb{E}_{x \sim p_{data}} \log D(x) + \mathbb{E}_{x \sim p_g} \log(1 - D(x))]$$

## Theorem

$$\arg \max_D V(D, G) = D_G = \frac{p_{data}}{p_{data} + p_g}.$$

**Proof.**  $V(D, G) = \mathbb{E}_{x \sim p_{data}} \log D(x) + \mathbb{E}_{x \sim p_g} \log(1 - D(x)) = \int_x (p_{data}(x) \log D(x) + p_g(x) \log(1 - D(x))) dx.$

## GAN and optimal distributions (2)

Theorem

$$\arg \max_D V(D, G) = D_G = \frac{p_{data}}{p_{data} + p_g}.$$

**Proof.** Let us find the extremum point of a function  $f(s) = p_{data} \log s + p_g \log(1 - s)$ :

- $f'(s) = \frac{p_{data}}{s} - \frac{p_g}{1-s}$ ,
- $f'(s) = 0 \Leftrightarrow p_{data}(1 - s) - p_g s = 0 \Leftrightarrow s = \frac{p_{data}}{p_{data} + p_g}$ .

At the same time  $s = \frac{p_{data}}{p_{data} + p_g}$  the extremum point of a function  $f(s)$  (the simple check:  
e.g.,  $\lim_{s \rightarrow 0+} f'(s) > 0, \lim_{s \rightarrow 1-} f'(s) < 0$ ).

Thus  $\arg \max_D V(D, G) = D_G = \frac{p_{data}}{p_{data} + p_g}$ . ■

# GAN and optimal distributions (3)

Let us  $C(G) = \max_D V(D, G)$

## Theorem

$$\arg \min_G C(G) = p_{data}, C(p_{data}) = -\log 4$$

- $C(G) = \mathbb{E}_{x \sim p_{data}} \log D_G(x) + \mathbb{E}_{x \sim p_g} \log(1 - D_G(x)) = \mathbb{E}_{x \sim p_{data}} \log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} + \mathbb{E}_{x \sim p_g} \log \frac{p_g(x)}{p_{data}(x) + p_g(x)},$
- $\mathbb{E}_{x \sim p_{data}} \log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} = \mathbb{E}_{x \sim p_{data}} \log \frac{p_{data}(x)}{\frac{p_{data}(x) + p_g(x)}{2}} - \mathbb{E}_{x \sim p_{data}} \log 2 = D_{KL}(p_{data} \parallel \frac{p_{data}(x) + p_g(x)}{2}) - \log 2,$
- $\mathbb{E}_{x \sim p_g} \log \frac{p_g(x)}{p_{data}(x) + p_g(x)} = D_{KL}(p_g \parallel \frac{p_{data}(x) + p_g(x)}{2}) - \log 2,$
- $\Rightarrow C(G) = D_{KL}(p_{data} \parallel \frac{p_{data}(x) + p_g(x)}{2}) + D_{KL}(p_g \parallel \frac{p_{data}(x) + p_g(x)}{2}) - 2 \log 2,$

# GAN and optimal distributions (4)

## Theorem

$$\arg \min_G C(G) = p_{data}, C(p_{data}) = -\log 4.$$

## Proof.

- $\Rightarrow C(G) = 2D_{JS}(p_{data}||p_g) - \log 4 \geq -\log 4,$
- Wherein  $C(G) = -\log 4 \Leftrightarrow D_{JS}(p_{data}||p_g) = 0 \Leftrightarrow p_{data} = p_g.$  ■

# GAN: why multiple iterations of D update in comparison to G update

## Theorem

$$\arg \min_G C(G) = p_{data}, C(p_{data}) = -\log 4.$$

**Remark.** Thus generator  $G$  will try to completely mirror the input real data distribution:  $p_g = p_{data}$ .

## BUT

The Theorem above is true only for **already** converged discriminator  $D_G \Rightarrow$   
That's why we try to put into discriminator training a little bit more iterations.  
At the same time, training discriminator w/o taking into account the untrained noisy  
generator will lead at the end to a very simple generator training capable of fooling  
discriminator 100% of cases.

# Main problems with GANs

## GANs problems

- Unstable training (e.g. by vanishing gradients for JS due to disjoint supports)
- Mode collapse (no any prevent for Generator to fall into few modes encouraged by Discriminator)
- Lack of explicit eval metric (when to stop? how to compare?)

Let us look into the first issue.

# Wasserstein (Kantorovich-Rubinstein)<sup>8</sup> metric

- Even the theoretical foundations of GAN have some flaws
- Proposal: to use so-called Wasserstein-1 **metric** (also known as Earth Mover Distance) for usage with distributions having different supports:

$$W_1(P, Q) = \inf_{\gamma \in \Pi(P, Q)} \mathbb{E}_{(x, y) \sim \gamma} [|x - y|_1]$$

where  $\Pi(P, Q)$  is the set of all joint distributions  $\gamma(x, y)$  so as its marginal distributions are  $P$  и  $Q$  correspondingly

- But this definition of  $W_1(P, Q)$  doesn't provide tractable solution. Recently the dual Theorem of Kantorovich-Rubinstein has been proved<sup>7</sup>:

$$W_1(P, Q) = \sup_{\|f\|_L \leq 1} (\mathbb{E}_{x \sim P}[f(x)] - \mathbb{E}_{x \sim Q}[f(x)])$$

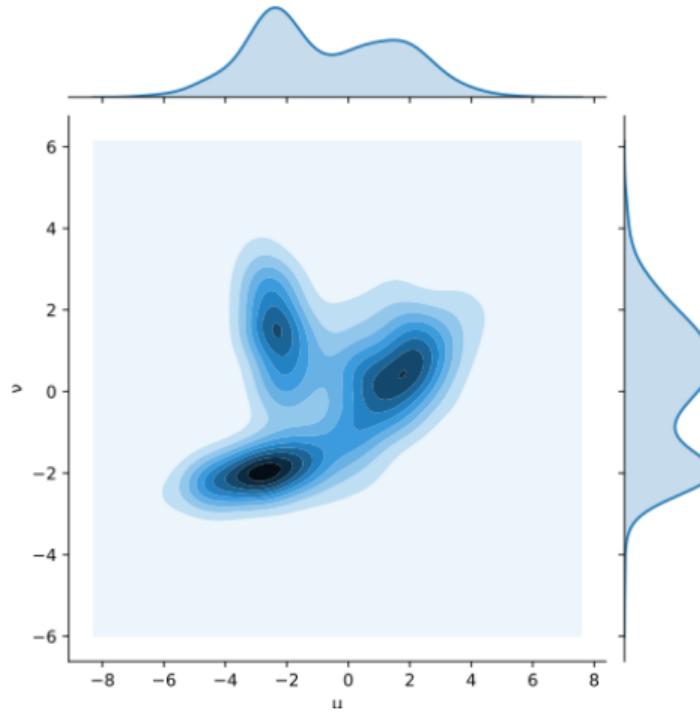
---

<sup>7</sup>Cedric Villani. Optimal Transport: Old and New. 2009

<sup>8</sup>L. Kantorovich. "Mathematical methods of organizing and planning production." 1939

# Wasserstein metric illustration

- What is joint distribution  
 $\gamma(x, y) \in \Pi(P, Q)?$
- $\sum_x \gamma(x, y) = Q(y)$
- $\sum_y \gamma(x, y) = P(x)$
- There could be multiple  
 $\gamma(x, y) \in \Pi(P, Q)$  minimizing  $W_1(P, Q)$



# Wasserstein metric and disjoint supports

- Suppose that  $\forall(x, y) \in P: x = 0, y \sim U(0, 1)$
- $\forall(x, y) \in Q: x = \theta, \theta \in [0, 1], y \sim U(0, 1)$
- When  $\theta \neq 0$ :  $D_{KL}(P||Q) = D_{KL}(Q||P) = +\infty$ ,  $D_{JS}(P||Q) = \log 2$ ,  $W_1(P, Q) = \theta$
- When  $\theta = 0$ :  $D_{KL}(P||Q) = D_{KL}(Q||P) = D_{JS}(P||Q) = 0$ ,  $W_1(P, Q) = \theta = 0$

## Conclusion

- $D_{KL}$  gives infinity on disjoint supports
- $D_{JS}$  gives discontinuity at  $\theta = 0$ , and vanishing gradients outside
- $W_1$  gives a smooth value and more stable training

**Exercise:** Prove it.

# On Lipschitz functions

- Condition  $\|f\|_L \leq 1$  means that  $f$  is 1-Lipschitz function
- **1-Lipschitz function**, or so-called **metric map**: a continuous mapping between metric spaces which doesn't increase distances.

## Metric map

If  $X, Y$  – metric spaces,  $f : X \rightarrow Y$  – mapping function,  $d_X(\cdot, \cdot), d_Y(\cdot, \cdot)$  – distance functions on  $X$  and  $Y$  correspondingly, then  $f$  – **metric map** iff for all points  $a, b \in X$  the following is true:  $d_Y(f(a), f(b)) \leq d_X(a, b)$ .

# Wasserstein metric and GAN

$$W_1(P, Q) = \sup_{\|f\|_L \leq 1} (\mathbb{E}_{x \sim P}[f(x)] - \mathbb{E}_{x \sim Q}[f(x)])$$

- Let us inject<sup>9</sup> Wasserstein metric into original GAN:
  - As function  $f$  we use discriminator  $D$
  - As distribution  $P$  – real data distribution  $p_{data}$
  - As distribution  $Q$  – generated data distribution  $p_g$  (distribution  $G(z)$ )
- Then functional to optimize:

$$\min_G W_1(p_{data}, p_g) = \min_G \max_{\|D\|_L \leq 1} (\mathbb{E}_{x \sim p_{data}}[D(x)] - \mathbb{E}_{z \sim p_z}[D(G(z))])$$

<sup>9</sup>M. Arjovsky et al. “Wasserstein GAN”. 2017

# WGAN training algorithm

**Algorithm 1** WGAN, our proposed algorithm. All experiments in the paper used the default values  $\alpha = 0.00005$ ,  $c = 0.01$ ,  $m = 64$ ,  $n_{\text{critic}} = 5$ .

**Require:** :  $\alpha$ , the learning rate.  $c$ , the clipping parameter.  $m$ , the batch size.  $n_{\text{critic}}$ , the number of iterations of the critic per generator iteration.

**Require:** :  $w_0$ , initial critic parameters.  $\theta_0$ , initial generator's parameters.

```
1: while  $\theta$  has not converged do
2:   for  $t = 0, \dots, n_{\text{critic}}$  do
3:     Sample  $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$  a batch from the real data.
4:     Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of priors.
5:      $g_w \leftarrow \nabla_w [\frac{1}{m} \sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))]$ 
6:      $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$ 
7:      $w \leftarrow \text{clip}(w, -c, c)$ 
8:   end for
9:   Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
10:   $g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))$ 
11:   $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$ 
12: end while
```

- Metric map condition in case of discriminator is relaxed by its weights clipping:  $w_L = \min(\max(-c, w), c)$ , where  $c \in \mathbb{R}_+$  — some quite small number
- Discriminator is called **critic**
- No logarithms inside optimization function, the output of critic is used (e.g., by global average layer)
- Every critic update iteration its weights are clipped by interval  $[-c, c]$

# DCGAN vs WGAN



- DCGAN: **left**, WGAN: **right**
- WGAN training is approximately 25% faster than DCGAN
- Quality is similar

# WGAN-GP<sup>10</sup>: further development of WGAN

- Let us rewrite the condition  $\|f\|_L \leq 1$  as:
- $|f(x) - f(y)| \leq \|x - y\|$
- $\frac{|f(x) - f(y)|}{\|x - y\|} \leq 1$
- Taking into account  $x \rightarrow y$  we get:  $\|\nabla_x f(x)\| \leq 1$

Thus we can inject so-called **Gradient Penalty**:

## WGAN-GP

Let us  $x \sim p_{data}$ ,  $y = G(z) \sim p_g$ ,  $s = tx + (1 - t)y$ ,  $t \in [0, 1]$ . Functional to optimize:

$$\max_G \min_D \left( \mathbb{E}_{z \sim p_z} [D(G(z))] - \mathbb{E}_{x \sim p_{data}} [D(x)] + \lambda \mathbb{E}_{s \sim p_s} [(\|\nabla_s D(s)\| - 1)^2] \right)$$

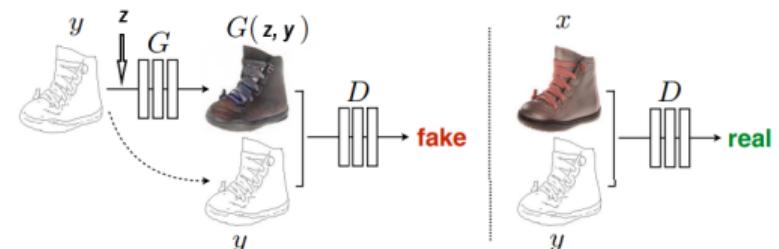
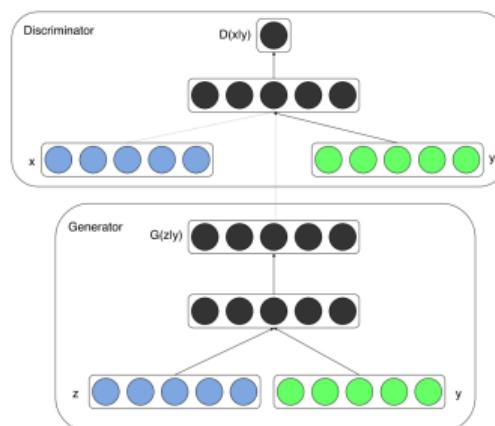
**Remark.** Need of  $s$  usage instead of  $x$  is proved in the corresponding paper (idea: along the interpolation line  $s = tx + (1 - t)y$  the gradient has a unit norm)

<sup>10</sup>Gulrajani I. et al. "Improved training of wasserstein gans". 2017.

# Conditional GAN

Suppose we would like to put some conditions into generation process<sup>11,12</sup>: e.g., the object class  $y$ . Then the function to optimize for cGAN (conditional GAN):

$$\min_G \max_D [\mathbb{E}_{x \sim p_{data}, y \sim p_y} \log D(x, y) + \mathbb{E}_{z \sim p_z, y \sim p_y} \log(1 - D(G(z, y), y))]$$



<sup>11</sup>Mirza, Mehdi, and Simon Osindero. “Conditional generative adversarial nets.” 2014

<sup>12</sup>Isola, Phillip, et al. “Image-to-image translation with conditional adversarial networks.” 2016

# GANs: a grain of salt

Google Brain<sup>13</sup>

Finally, we did not find evidence that any of the tested algorithms consistently outperforms the original one.

Two major claims from it:

- When the budget is limited, any statistically significant comparison of the models is unattainable
- Algorithmic differences in state-of-the-art GANs become less relevant, as the computational budget increases

---

<sup>13</sup>M. Lucic et al. “Are gans created equal? a large-scale study.” 2018.

# Takeaway notes

- GAN can be used for synthesis of data close in distribution to real data
- Minimax GAN training is super unstable
- Some improvements of GAN training are connected to Wasserstein metric<sup>14</sup>
- GAN publication number: a couple of years ago was huge spike, now decreasing
- Every year the quality of synthetic data becomes better and better
- We are entering the DeepFake<sup>15</sup> era

---

<sup>14</sup>Gulrajani, Ishaan, et al. “Improved training of wasserstein gans.” 2017.

<sup>15</sup><https://deepfakedetectionchallenge.ai/>

Thank you!