

Generative Adversarial Networks

Machine Learning and Data Mining, 2023

Presented by: Abdalaziz Al-Maeeni

Prepared by: Artem Maevskiy

National Research University Higher School of Economics



LAMBDA • HSE

Dec 16, 2023

Generative modelling



Problem setup

- ▶ Training data – a set of objects, e.g.:
 - Photos of animals / people faces / rooms / whatever
 - Text
 - Audio of speech / music / whatever
 - Signals from a high energy physics experiment detector
 - ▶ Goal: build a model to sample similar data
- Set of objects:
 $\{x_i \mid i = 1, \dots, N\}$

Problem setup

- ▶ Training data – a set of objects, e.g.:
 - Photos of animals / people faces / rooms / whatever
 - Text
 - Audio of speech / music / whatever
 - Signals from a high energy physics experiment detector
 - ▶ Goal: build a model to sample similar data
- Set of objects:
 $\{x_i \mid i = 1, \dots, N\}$
- Population PDF:
 $p(x)$
- I.e. $\{x_i\}$ are i.i.d.
sampled from $p(x)$

Problem setup

- ▶ Training data – a set of objects, e.g.:
 - Photos of animals / people faces / rooms / whatever
 - Text
 - Audio of speech / music / whatever
 - Signals from a high energy physics experiment detector
- ▶ Goal: build a model to sample similar data
 - Learn the **population distribution** to sample more objects from it
 - (may be done implicitly, i.e. when we can't evaluate the probability density, yet can sample from it)

Set of objects:

$$\{x_i \mid i = 1, \dots, N\}$$

Population PDF:

$$p(x)$$

i.e. $\{x_i\}$ are i.i.d.
sampled from $p(x)$

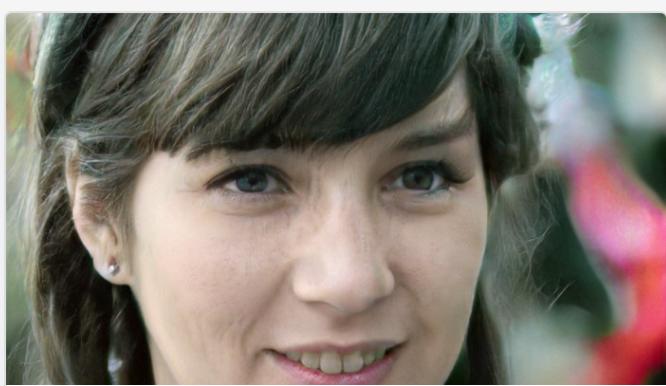
Learn $q(x) \sim p(x)$ to
sample x' from $q(x)$

Possible applications

To name a few:

- ▶ Creative professions, e.g.
 - Generating textures for 3D modelling, audio samples for music etc.
 - Style transfer
- ▶ Data privacy
 - Publishing samples without disclosing private information from the original data
- ▶ Physics and engineering
 - Building a deep learning model to reproduce the stochastic output of an accurate but slow simulator algorithm

This X does not exist



This Person Does Not Exist

The site that started it all, with the name that says it all. Created using a style-based generative adversarial network (StyleGAN), this website had the tech community buzzing with excitement and intrigue and inspired many more sites.

Created by Phillip Wang.



This Cat Does Not Exist

These purr-fect GAN-made cats will freshen your feeline-gs and make you wish you could reach through your screen and cuddle them. Once in a while the cats have visual deformities due to imperfections in the model – beware, they can cause nightmares.

Created by Ryan Hoover.



This Rental Does Not Exist

Why bother trying to look for the perfect home when you can create one instead? Just find a listing you like, buy some land, build it, and then enjoy the rest of your life.

Created by Christopher Schmidt.

<https://thisxdoesnotexist.com/>

Style transfer



<https://junyanz.github.io/CycleGAN/>

Generative models progress



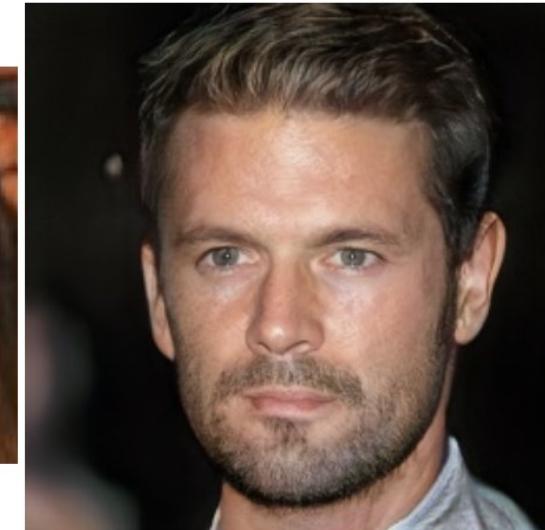
2014



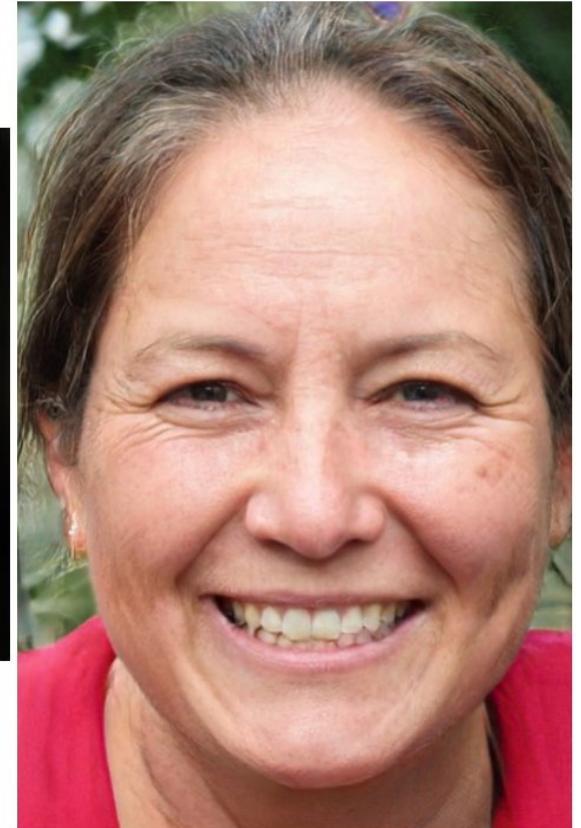
2015



2016



2017



2018

https://twitter.com/goodfellow_ian/status/1084973596236144640

Connection to supervised learning



Supervised learning \leftrightarrow Generative modelling

- ▶ Supervised: learning relation f between features x and targets y :

$$y = f(x)$$

- ▶ f is typically stochastic:

$$y | x \sim p_f(y | x)$$

- ▶ Typically, we model $p_f(y | x)$ in a very simple form like:

$$p_\theta(y | x) = \mathcal{N}(y | \mu = f_\theta(x), \sigma^2),$$

which leads to optimizing MSE

- ▶ Generative modelling \approx learning a generic distribution $p_f(y | x)$

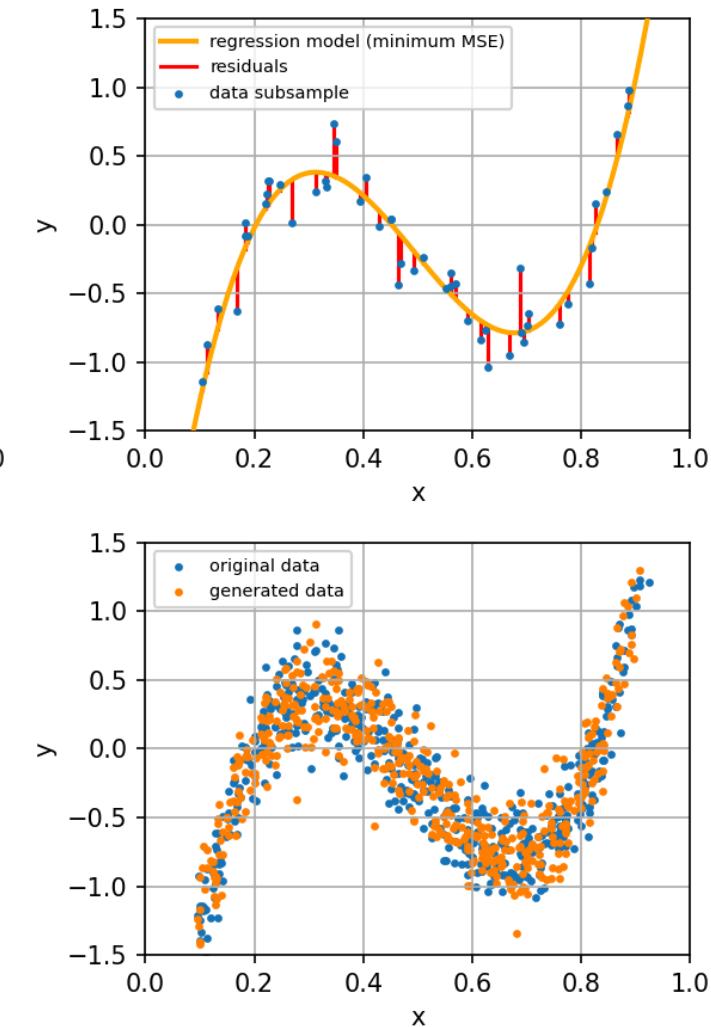
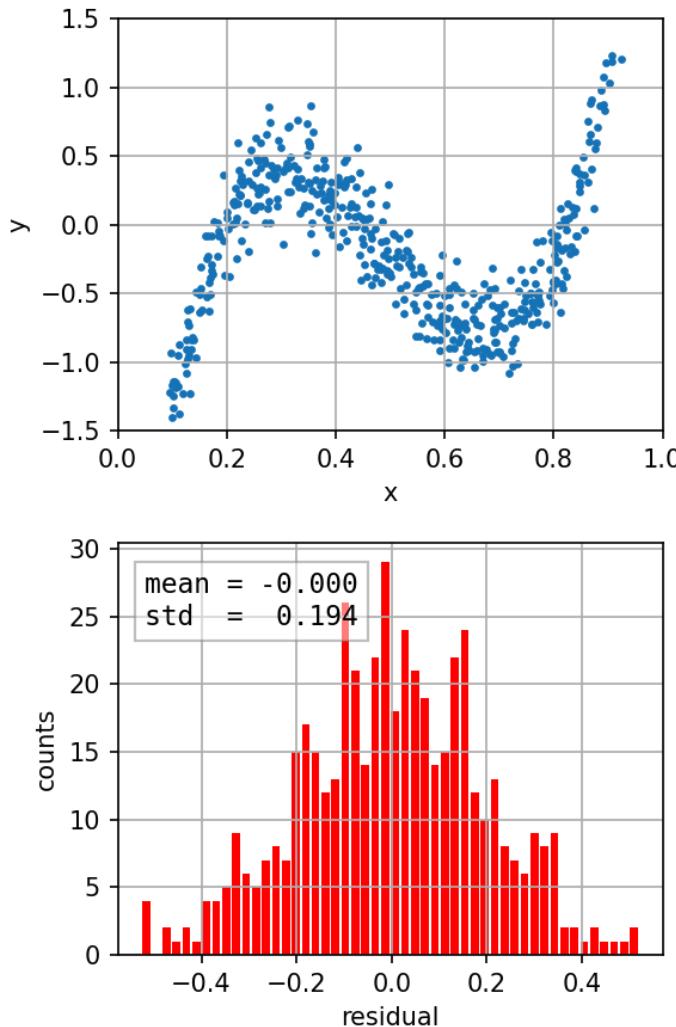
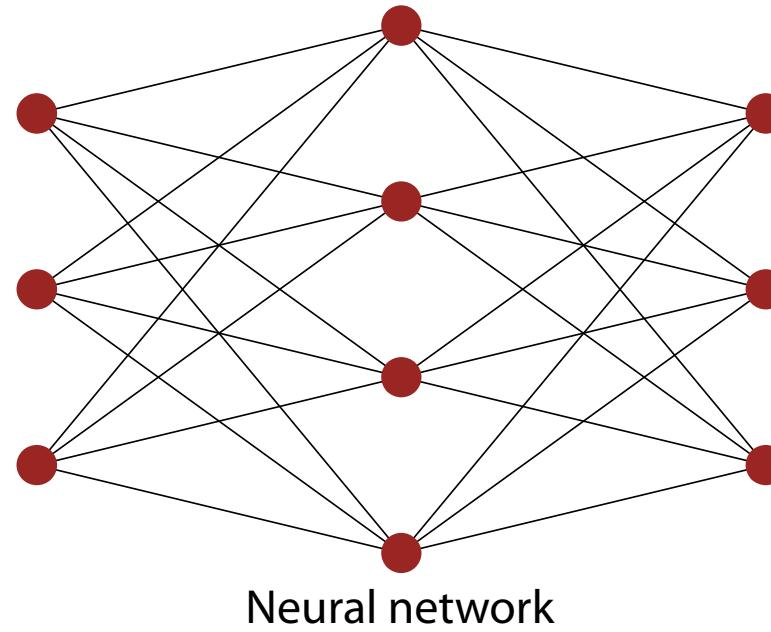


Image reference: <https://en.pelican.study/>
<https://alemira.com/examples/>

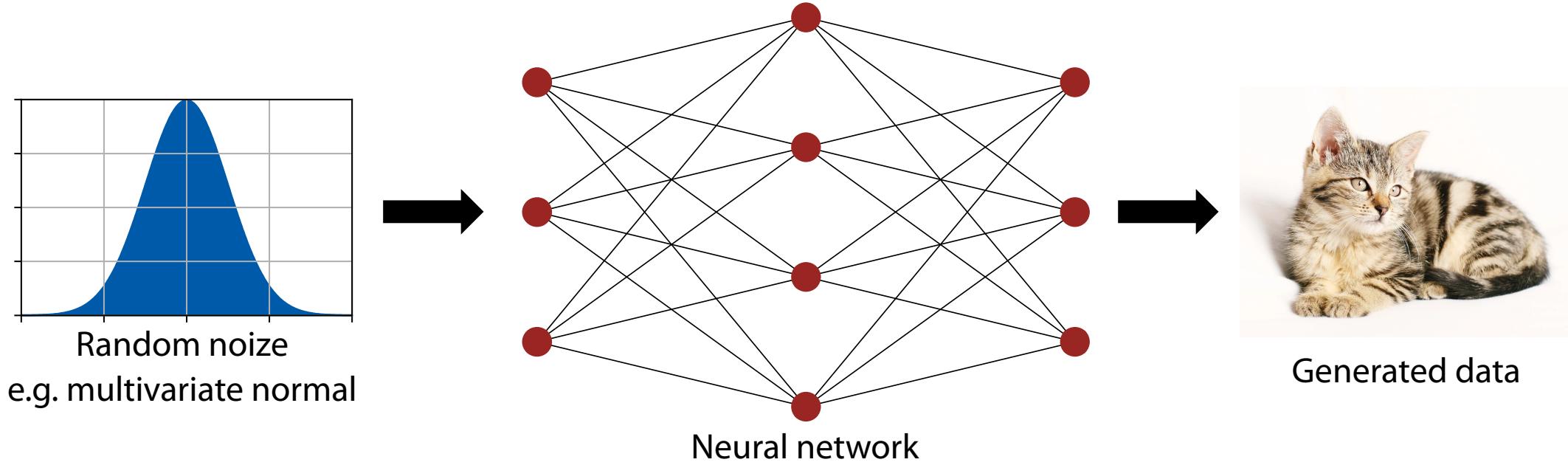
Generative Adversarial Networks



How can a neural network generate data?

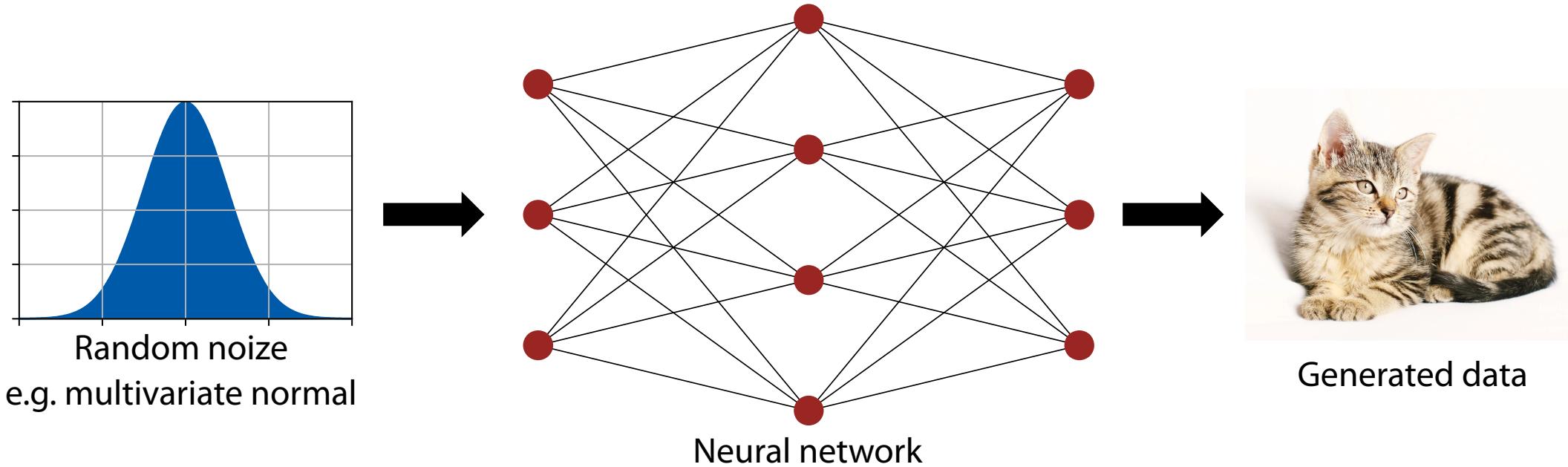


How can a neural network generate data?



Cat image attribution: <https://pixabay.com/users/chiemsee2016-1892688/>

How can a neural network generate data?



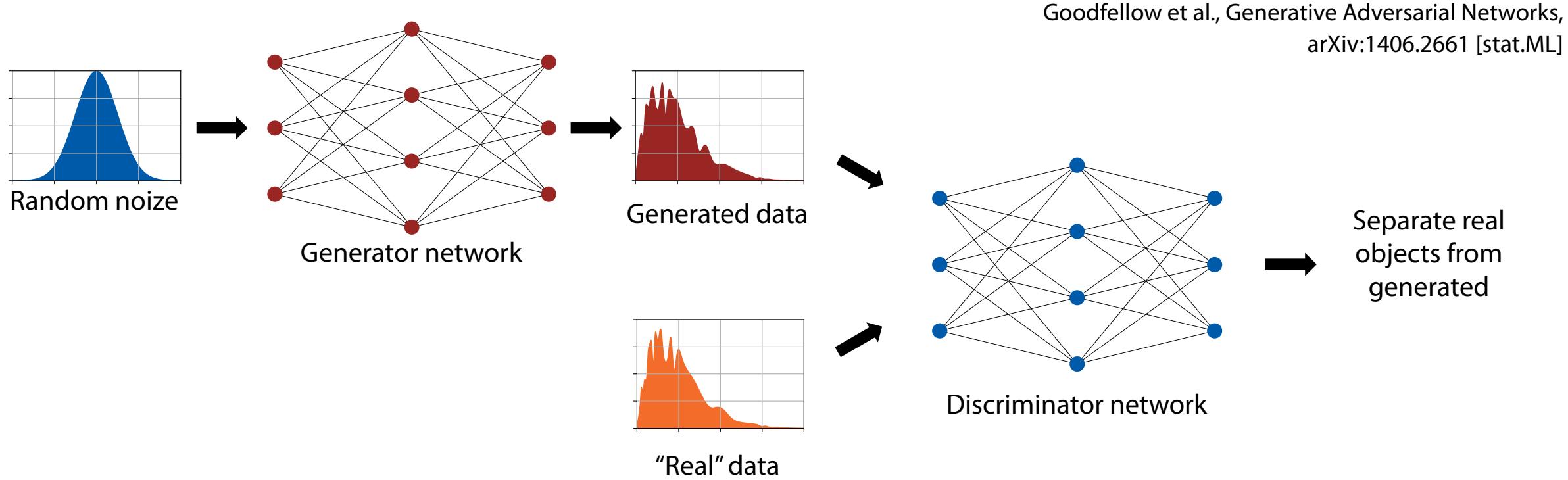
- ▶ This makes the generated object being a **differentiable function** of the network parameters

Cat image attribution: <https://pixabay.com/users/chiemsee2016-1892688/>

How to train such a generator?

- ▶ Generated object is a differentiable function of the network parameters
- ▶ Need a differentiable **measure of similarity** between the sets of generated objects and real ones
 - Can optimize with gradient descent
- ▶ How to find such a measure?

Adversarial approach



- ▶ Measure of similarity: how well can another neural network (discriminator) tell the generated objects apart from the real ones

Let's put it in formulas

- ▶ Noise samples:

$$z_i \sim p_z(z)$$

where p_z is some simple PDF we can sample from, e.g. $\mathcal{N}(0, \mathbb{I})$.

Let's put it in formulas

- ▶ Noise samples:

$$z_i \sim p_z(z)$$

where p_z is some simple PDF we can sample from, e.g. $\mathcal{N}(0, \mathbb{I})$.

- ▶ Generated samples:

$$x'_i = G_\theta(z_i)$$

where G_θ is the generator network with parameters θ .

Let's put it in formulas

- ▶ Noise samples:

$$z_i \sim p_z(z)$$

where p_z is some simple PDF we can sample from, e.g. $\mathcal{N}(0, \mathbb{I})$.

- ▶ Generated samples:

$$x'_i = G_\theta(z_i)$$

where G_θ is the generator network with parameters θ .

- ▶ Discriminator network (with parameters ϕ):

$$D_\phi(x)$$

returns the probability for x being a real sample rather than a generated one

Let's put it in formulas

- ▶ Noise samples:

$$z_i \sim p_z(z)$$

where p_z is some simple PDF we can sample from, e.g. $\mathcal{N}(0, \mathbb{I})$.

- ▶ Generated samples:

$$x'_i = G_\theta(z_i)$$

where G_θ is the generator network with parameters θ .

- ▶ Discriminator network (with parameters ϕ):

$$D_\phi(x)$$

returns the probability for x being a real sample rather than a generated one

- ▶ Measure of similarity between the generated and real samples:

$$L_G = \max_{\phi} \left[\mathbb{E}_{x \sim p(x)} [\log D_\phi(x)] + \mathbb{E}_{z \sim p_z(z)} [\log (1 - D_\phi(G_\theta(z)))] \right] \rightarrow \min_{\theta}$$

Let's put it in formulas

- ▶ Noise samples:

$$z_i \sim p_z(z)$$

where p_z is some simple PDF we can sample from, e.g. $\mathcal{N}(0, \mathbb{I})$.

- ▶ Generated samples:

$$x'_i = G_\theta(z_i)$$

where G_θ is the generator network with parameters θ .

- ▶ Discriminator network (with parameters ϕ):

$$D_\phi(x)$$

returns the probability for x being a real sample rather than a generated one

- ▶ Measure of similarity between the generated and real samples: Probability the sample was generated

$$L_G = \max_{\phi} \left[\mathbb{E}_{x \sim p(x)} [\log D_\phi(x)] + \mathbb{E}_{z \sim p_z(z)} \left[\log \left(1 - D_\phi(G_\theta(z)) \right) \right] \right] \rightarrow \min_{\theta}$$

Let's put it in formulas

- ▶ Noise samples:

$$z_i \sim p_z(z)$$

where p_z is some simple PDF we can sample from, e.g. $\mathcal{N}(0, \mathbb{I})$.

- ▶ Generated samples:

$$x'_i = G_\theta(z_i)$$

where G_θ is the generator network with parameters θ .

- ▶ Discriminator network (with parameters ϕ):

$$D_\phi(x)$$

returns the probability for x being a real sample rather than a generated one

- ▶ Measure of similarity between the generated and real samples: Probability the sample was generated

$$L_G = \max_{\phi} \left[\mathbb{E}_{x \sim p(x)} [\log D_\phi(x)] + \mathbb{E}_{z \sim p_z(z)} \left[\log \left(1 - D_\phi(G_\theta(z)) \right) \right] \right] \rightarrow \min_{\theta}$$

Log-likelihood for discriminator prediction

Let's put it in formulas

- ▶ Noise samples:

$$z_i \sim p_z(z)$$

where p_z is some simple PDF we can sample from, e.g. $\mathcal{N}(0, \mathbb{I})$.

- ▶ Generated samples:

$$x'_i = G_\theta(z_i)$$

where G_θ is the generator network with parameters θ .

- ▶ Discriminator network (with parameters ϕ):

$$D_\phi(x)$$

returns the probability for x being a real sample rather than a generated one

- ▶ Measure of similarity between the generated and real samples: Probability the sample was generated

$$L_G = \max_{\phi} \left[\mathbb{E}_{x \sim p(x)} [\log D_\phi(x)] + \mathbb{E}_{z \sim p_z(z)} [\log (1 - D_\phi(G_\theta(z)))] \right] \rightarrow \min_{\theta}$$

Log-likelihood for discriminator prediction

Training the networks

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.

<https://arxiv.org/abs/1406.2661>

A view on the adversarial loss

- ▶ Evolution from classical machine learning to deep learning:
 - “Let’s build a model that automatically finds the best features for us”
- ▶ Evolution to adversarial loss:
 - “Let’s build a model that **learns the loss function** for us”

$$L_G = \max_{\phi} \left[\mathbb{E}_{x \sim p(x)} [\log D_\phi(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D_\phi(G_\theta(z)))] \right] \rightarrow \min_{\theta}$$


Generator loss is learned by optimising the discriminator

Problems with GANs



The optimal discriminator solution

- If we re-write the loss using $p_{\text{gen},\theta}(x)$ – the distribution of $x' = G_\theta(z)$, and expand the expectations as integrals:

$$L_G = \max_{\phi} \int_x \left[p(x) \log(D_\phi(x)) + p_{\text{gen},\theta}(x) \log(1 - D_\phi(x)) \right] dx$$

The optimal discriminator solution

- ▶ If we re-write the loss using $p_{\text{gen},\theta}(x)$ – the distribution of $x' = G_\theta(z)$, and expand the expectations as integrals:

$$L_G = \max_{\phi} \int \left[p(x) \log(D_\phi(x)) + p_{\text{gen},\theta}(x) \log(1 - D_\phi(x)) \right] dx$$

- ▶ it's easy to show that max is obtained at $\phi^*(\theta)$ with:

$$D_{\phi^*(\theta)}(x) = \frac{p(x)}{p(x) + p_{\text{gen},\theta}(x)}$$

The optimal discriminator solution

- ▶ If we re-write the loss using $p_{\text{gen},\theta}(x)$ – the distribution of $x' = G_\theta(z)$, and expand the expectations as integrals:

$$L_G = \max_{\phi} \int \left[p(x) \log(D_\phi(x)) + p_{\text{gen},\theta}(x) \log(1 - D_\phi(x)) \right] dx$$

- ▶ it's easy to show that max is obtained at $\phi^*(\theta)$ with:

$$D_{\phi^*(\theta)}(x) = \frac{p(x)}{p(x) + p_{\text{gen},\theta}(x)}$$

- ▶ So the objective becomes:

$$L_G = \mathbb{E}_{x \sim p(x)} \left[\log \frac{p(x)}{p(x) + p_{\text{gen},\theta}(x)} \right] + \mathbb{E}_{x \sim p_{\text{gen},\theta}(x)} \left[\log \frac{p_{\text{gen},\theta}(x)}{p(x) + p_{\text{gen},\theta}(x)} \right]$$

The optimal discriminator solution

- ▶ If we re-write the loss using $p_{\text{gen},\theta}(x)$ – the distribution of $x' = G_\theta(z)$, and expand the expectations as integrals:

$$L_G = \max_{\phi} \int \left[p(x) \log(D_\phi(x)) + p_{\text{gen},\theta}(x) \log(1 - D_\phi(x)) \right] dx$$

- ▶ it's easy to show that max is obtained at $\phi^*(\theta)$ with:

$$D_{\phi^*(\theta)}(x) = \frac{p(x)}{p(x) + p_{\text{gen},\theta}(x)}$$

- ▶ So the objective becomes:

$$\begin{aligned} L_G &= \mathbb{E}_{x \sim p(x)} \left[\log \frac{p(x)}{p(x) + p_{\text{gen},\theta}(x)} \right] + \mathbb{E}_{x \sim p_{\text{gen},\theta}(x)} \left[\log \frac{p_{\text{gen},\theta}(x)}{p(x) + p_{\text{gen},\theta}(x)} \right] \\ &= -\log 4 + JSD(p \parallel p_{\text{gen},\theta}) \end{aligned}$$

Jensen–Shannon divergence

Vanishing gradients

- ▶ In case p and $p_{\text{gen},\theta}$ have non-overlapping support:

$$L_G = \mathbb{E}_{x \sim p(x)} \left[\log \frac{p(x)}{p(x) + p_{\text{gen},\theta}(x)} \right] + \mathbb{E}_{x \sim p_{\text{gen},\theta}(x)} \left[\log \frac{p_{\text{gen},\theta}(x)}{p(x) + p_{\text{gen},\theta}(x)} \right]$$

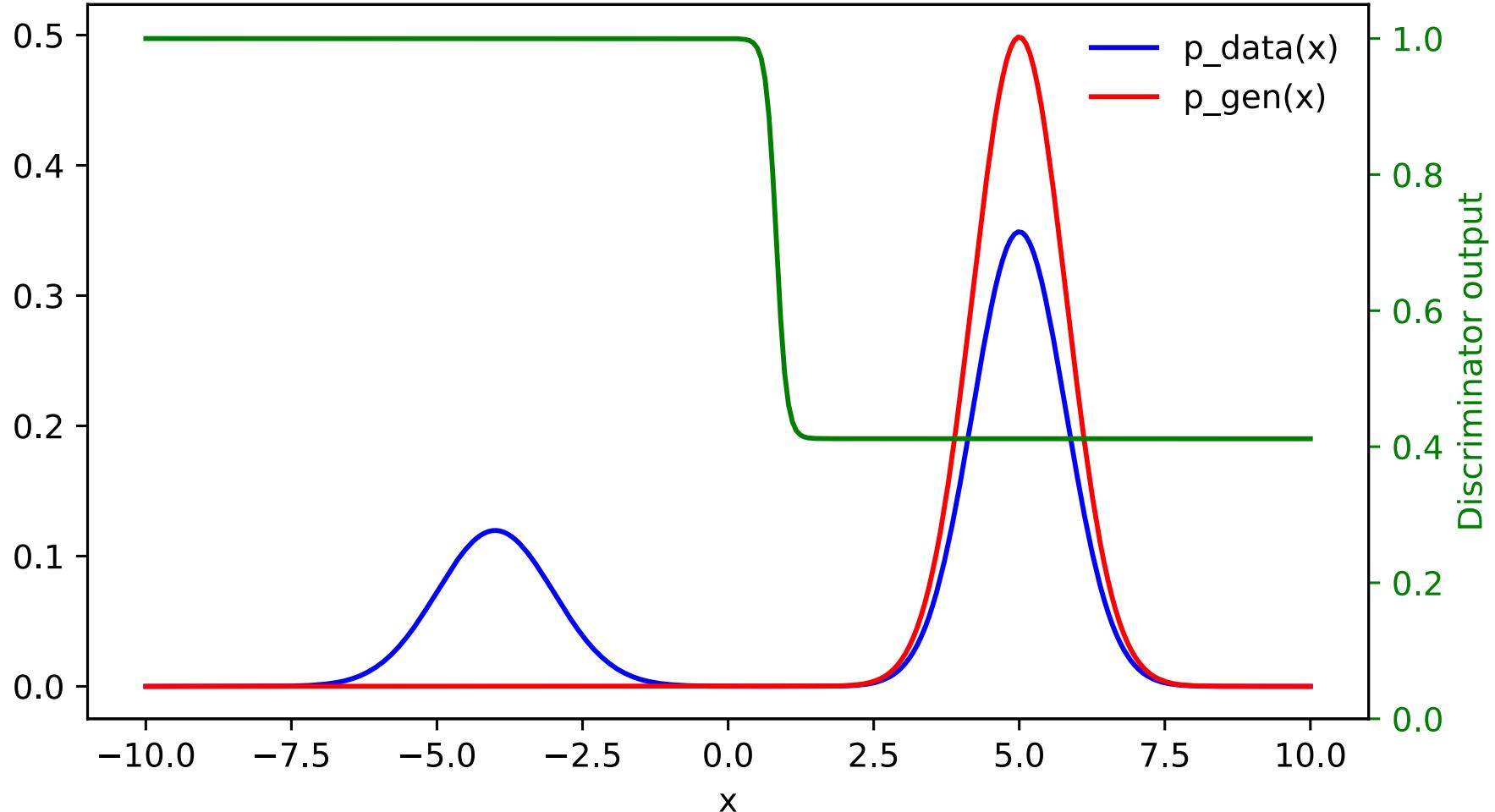
$$= \mathbb{E}_{x \sim p(x)} \left[\log \frac{p(x)}{p(x)} \right] + \mathbb{E}_{x \sim p_{\text{gen},\theta}(x)} \left[\log \frac{p_{\text{gen},\theta}(x)}{p_{\text{gen},\theta}(x)} \right] = 0 = \text{const}$$

- ▶ No meaningful gradient, can't learn

Mode collapse

- ▶ Assume at some point the generator has learned one of the modes
- ▶ No meaningful gradients to drive the solution towards covering the other modes

```
x = np.linspace(-10, 10, 300)
p_data = 0.7 * normal(x, 5, 0.8) + 0.3 * normal(x, -4, 1)
p_gen = 1.0 * normal(x, 5, 0.8)
D = p_data / (p_data + p_gen)
```



Wasserstein GAN



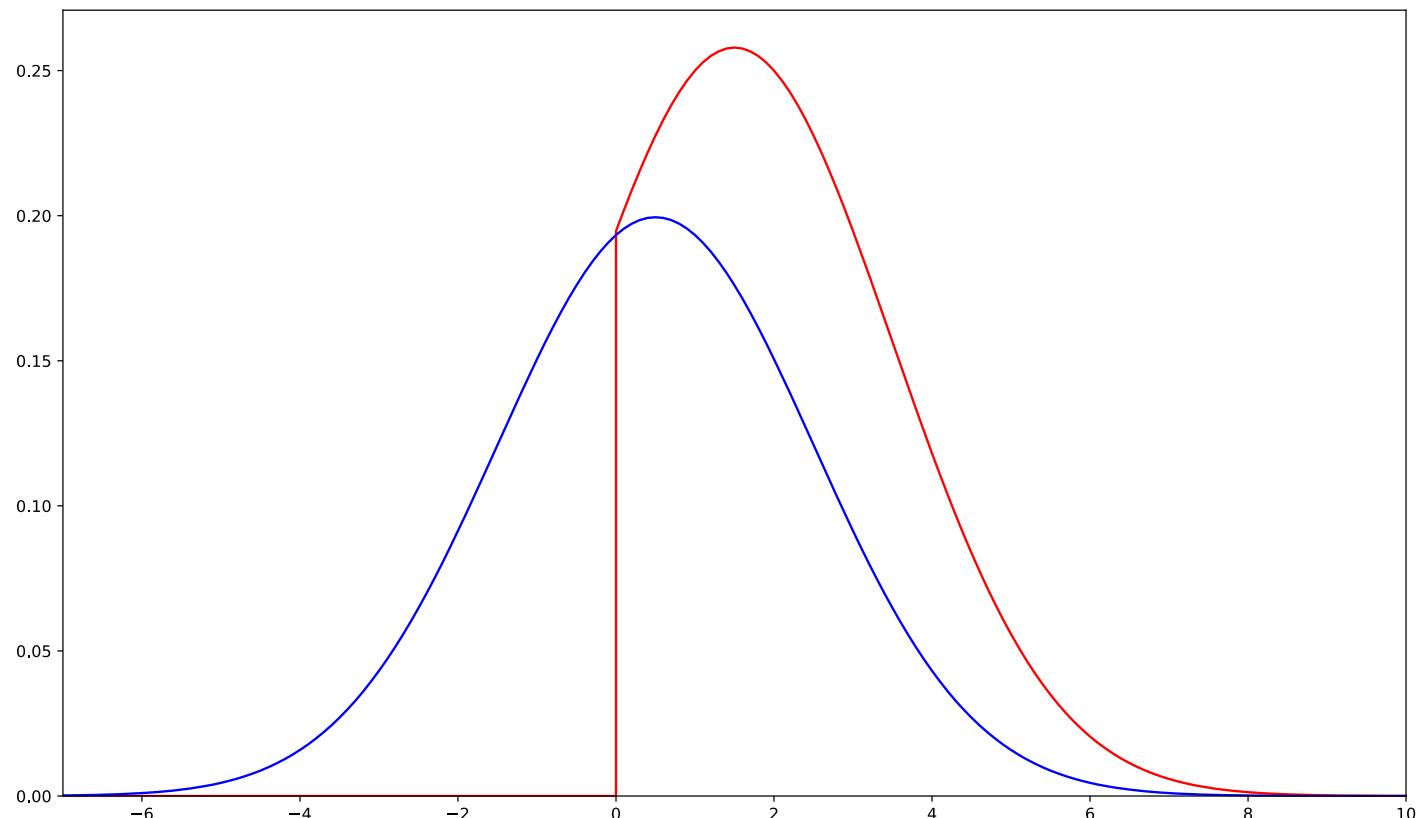
Alternative distance measure

- ▶ The problems with GANs are mainly due to Jensen–Shannon divergence providing problematic gradients
- ▶ What if we try to find some other measure of distance between real and generated distributions that doesn't have these problems?

Wasserstein distance

Also called “Earth mover’s distance” (EMD)

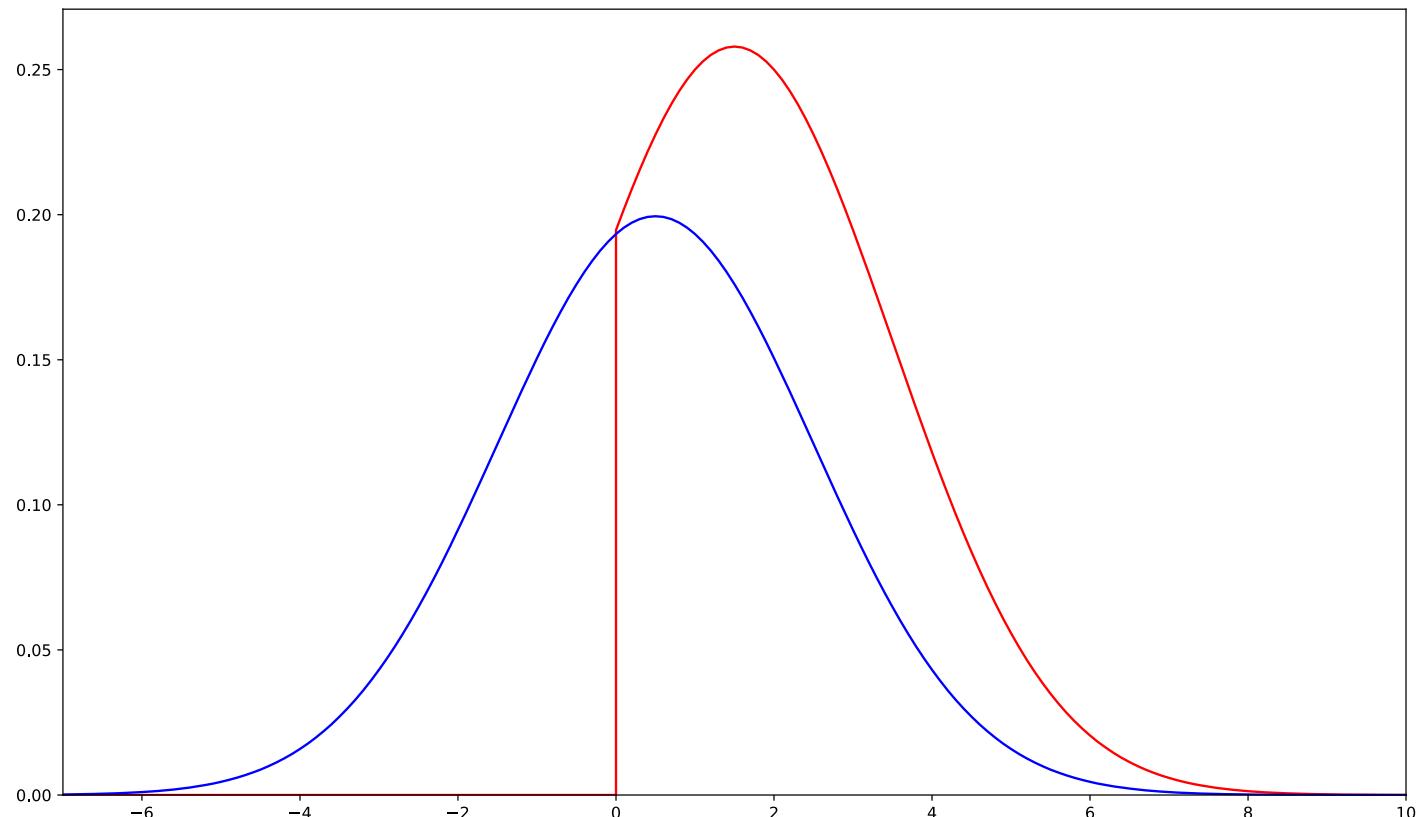
- Distributions $P(x)$ and $Q(x)$ are viewed as describing the amounts of “dirt” at point x



Wasserstein distance

Also called “Earth mover’s distance” (EMD)

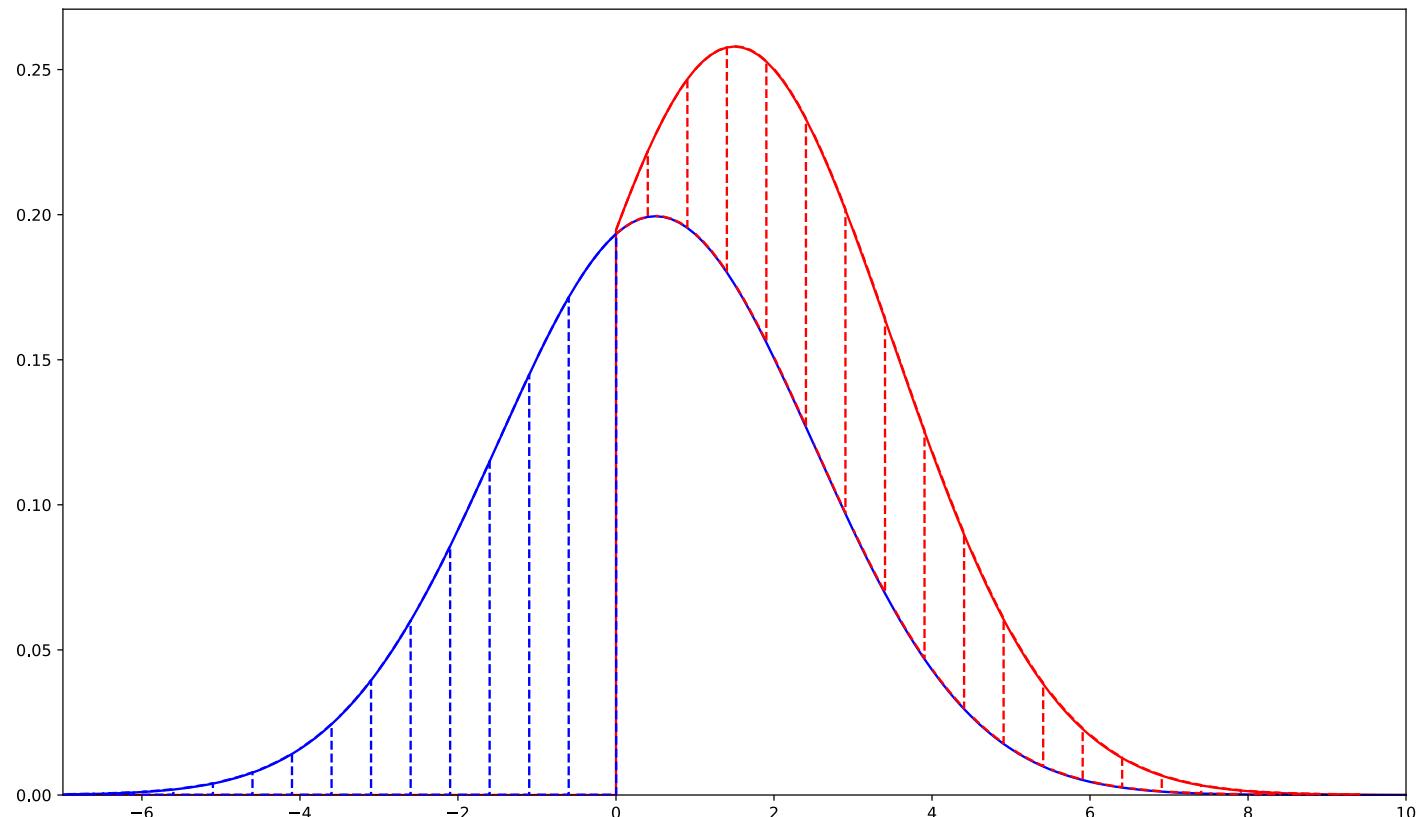
- ▶ Distributions $P(x)$ and $Q(x)$ are viewed as describing the amounts of “dirt” at point x
- ▶ We want to convert one distribution into the other by moving around some amounts of dirt



Wasserstein distance

Also called “Earth mover’s distance” (EMD)

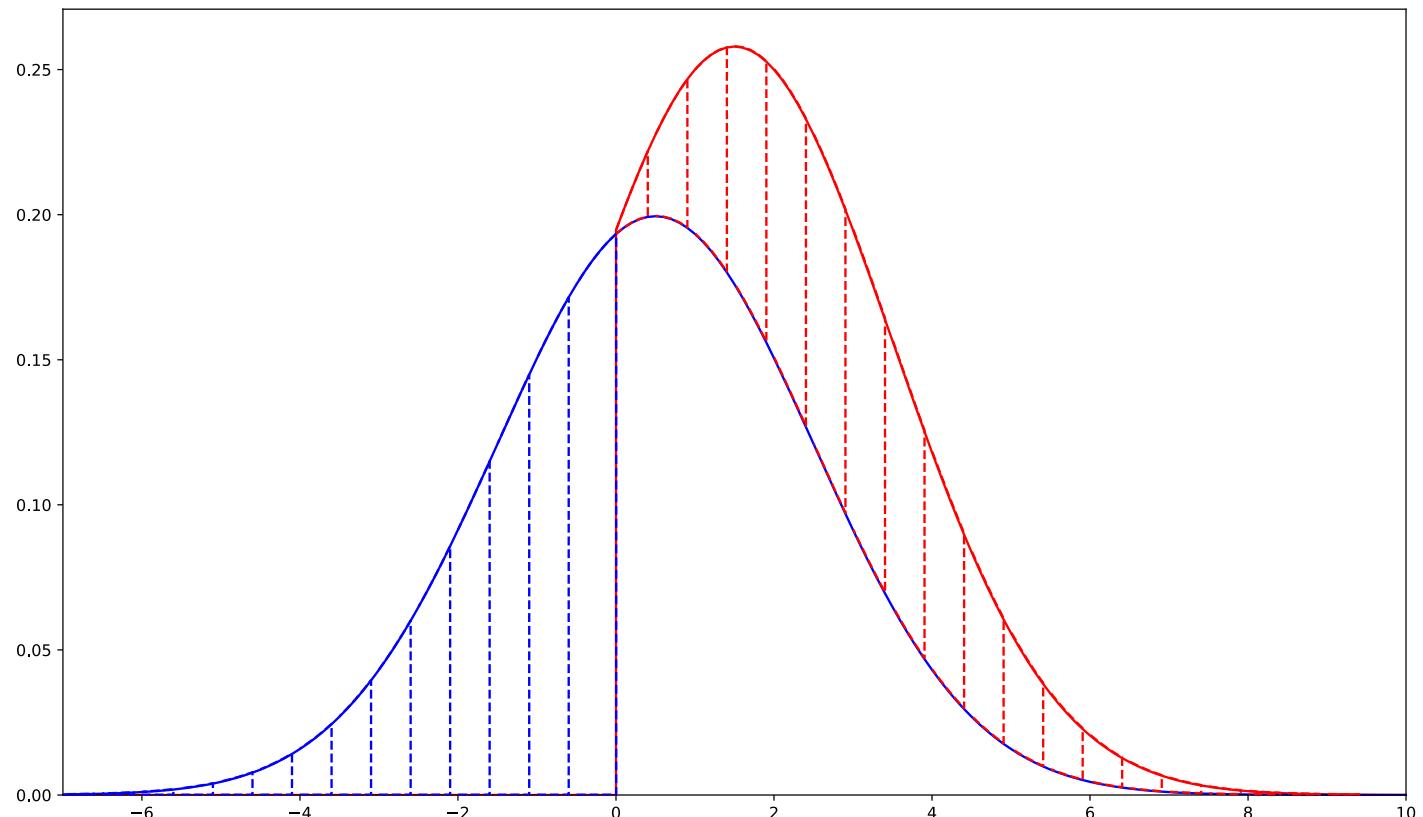
- ▶ Distributions $P(x)$ and $Q(x)$ are viewed as describing the amounts of “dirt” at point x
- ▶ We want to convert one distribution into the other by moving around some amounts of dirt



Wasserstein distance

Also called “Earth mover’s distance” (EMD)

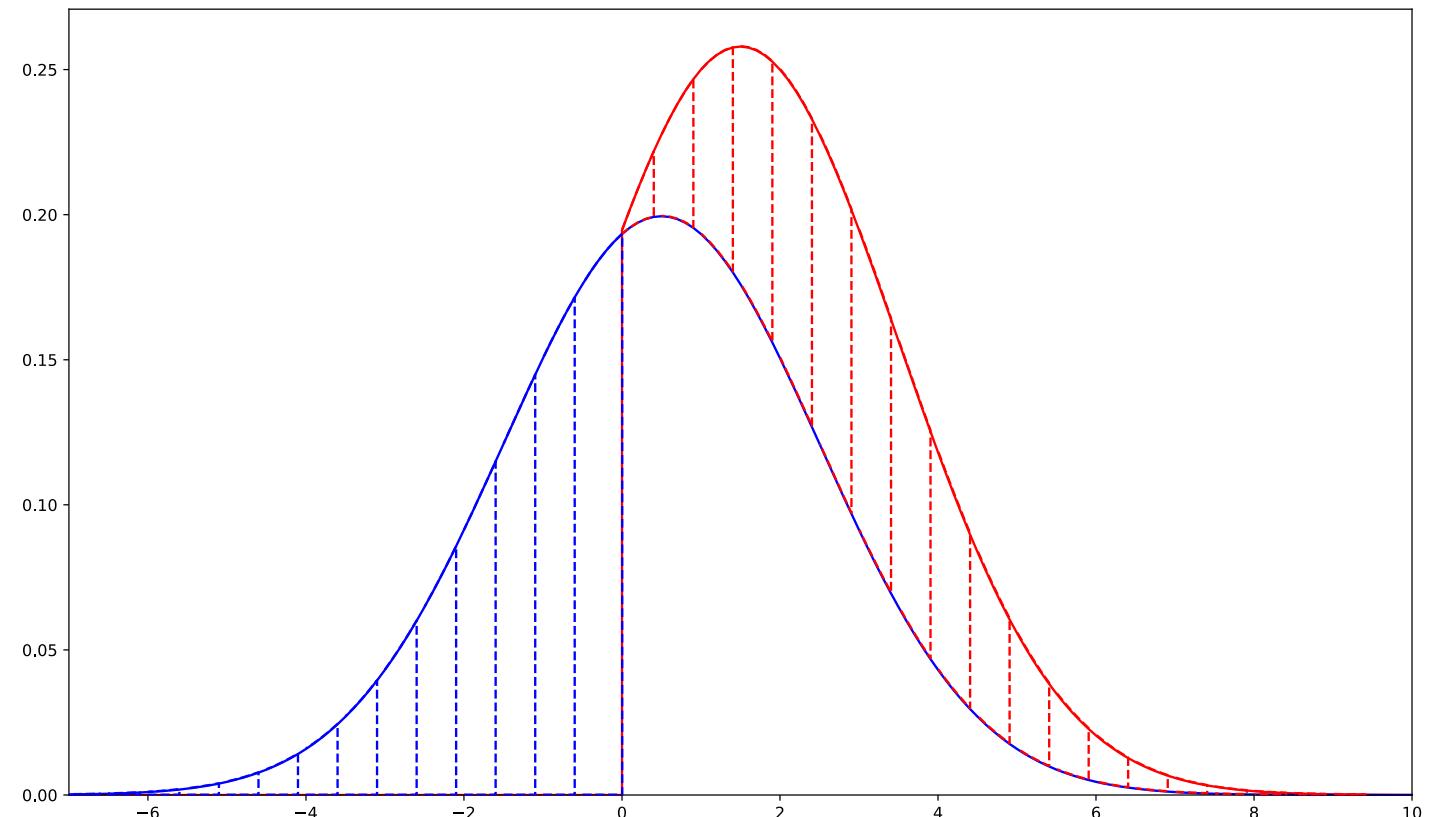
- ▶ Distributions $P(x)$ and $Q(x)$ are viewed as describing the amounts of “dirt” at point x
- ▶ We want to convert one distribution into the other by moving around some amounts of dirt



Wasserstein distance

Also called “Earth mover’s distance” (EMD)

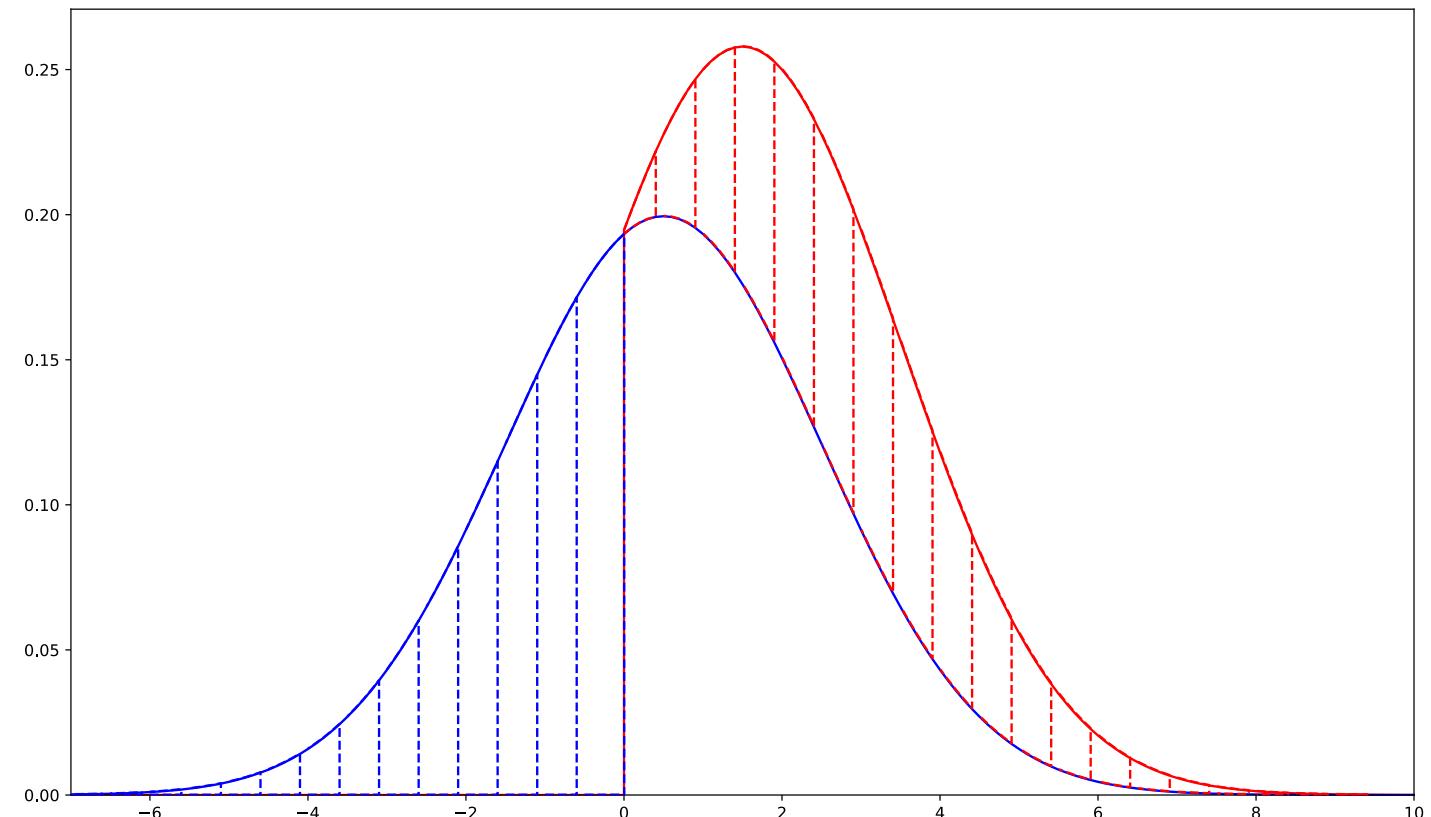
- ▶ Distributions $P(x)$ and $Q(x)$ are viewed as describing the amounts of “dirt” at point x
- ▶ We want to convert one distribution into the other by moving around some amounts of dirt
- ▶ The cost of moving an amount m from x_1 to x_2 is $m \times \|x_2 - x_1\|$



Wasserstein distance

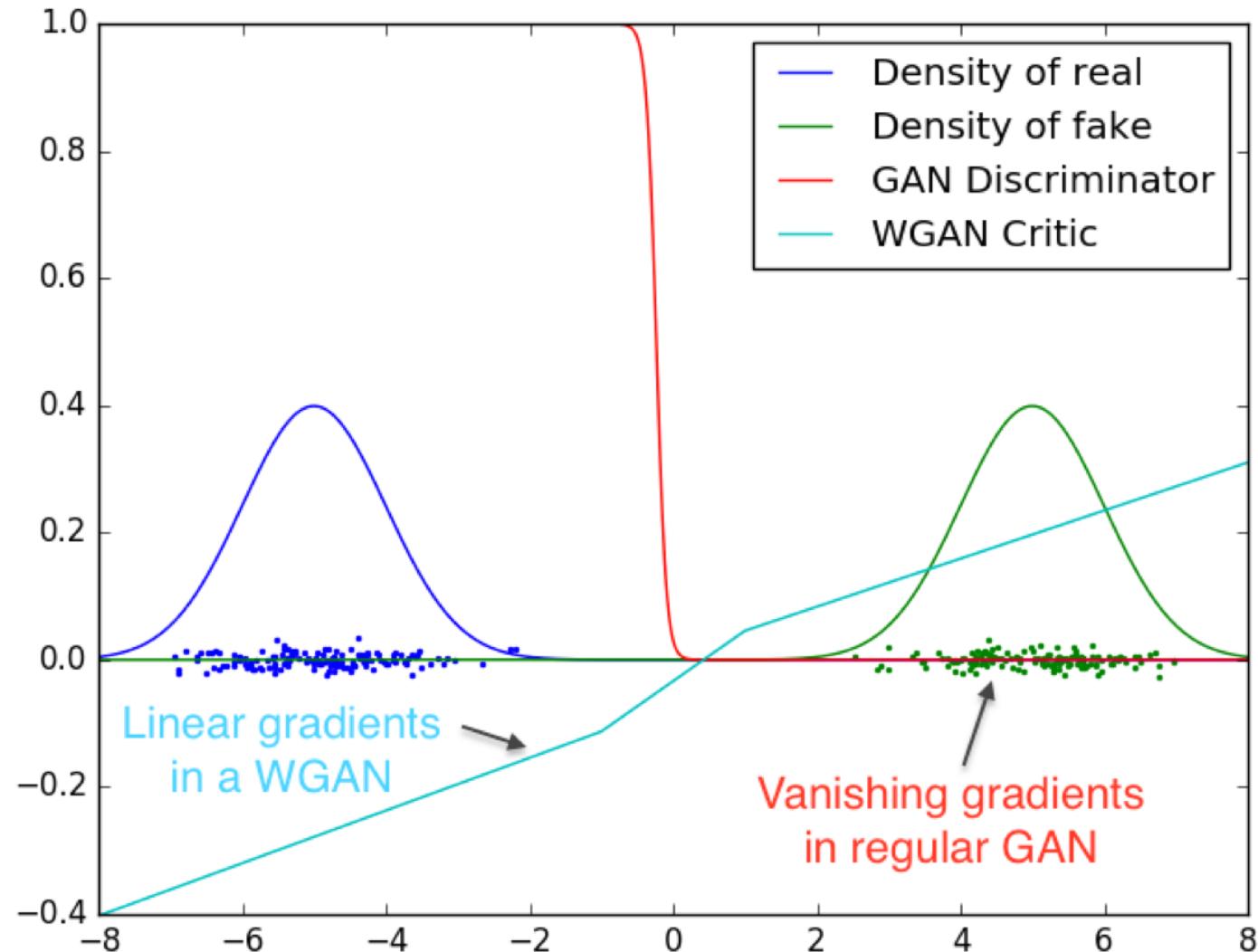
Also called “Earth mover’s distance” (EMD)

- ▶ Distributions $P(x)$ and $Q(x)$ are viewed as describing the amounts of “dirt” at point x
- ▶ We want to convert one distribution into the other by moving around some amounts of dirt



- ▶ The cost of moving an amount m from x_1 to x_2 is $m \times \|x_2 - x_1\|$
- ▶ $\text{EMD}(P, Q) = \text{minimum total cost of converting } P \text{ into } Q$

Why is it better?



<https://arxiv.org/abs/1701.07875>

Formal definition

- ▶ Say, we have a moving plan $\gamma(x_1, x_2) \geq 0$:

$\gamma(x_1, x_2) dx_1 dx_2$ – how much dirt we're moving from
 $[x_1, x_1 + dx_1]$ to $[x_2, x_2 + dx_2]$

Formal definition

- ▶ Say, we have a moving plan $\gamma(x_1, x_2) \geq 0$:
 $\gamma(x_1, x_2) dx_1 dx_2$ – how much dirt we're moving from
 $[x_1, x_1 + dx_1]$ to $[x_2, x_2 + dx_2]$
- ▶ Then, the cost of moving from $[x_1, x_1 + dx_1]$ to $[x_2, x_2 + dx_2]$ is:

$$\|x_2 - x_1\| \cdot \gamma(x_1, x_2) dx_1 dx_2$$

Formal definition

- ▶ Say, we have a moving plan $\gamma(x_1, x_2) \geq 0$:

$\gamma(x_1, x_2) dx_1 dx_2$ – how much dirt we're moving from
[$x_1, x_1 + dx_1$] to [$x_2, x_2 + dx_2$]

- ▶ Then, the cost of moving from [$x_1, x_1 + dx_1$] to [$x_2, x_2 + dx_2$] is:

$$\|x_2 - x_1\| \cdot \gamma(x_1, x_2) dx_1 dx_2$$

- ▶ and the total cost is:

$$C = \int_{x_1, x_2} \|x_2 - x_1\| \cdot \gamma(x_1, x_2) dx_1 dx_2$$

Formal definition

- ▶ Say, we have a moving plan $\gamma(x_1, x_2) \geq 0$:

$\gamma(x_1, x_2) dx_1 dx_2$ – how much dirt we're moving from
 $[x_1, x_1 + dx_1]$ to $[x_2, x_2 + dx_2]$

- ▶ Then, the cost of moving from $[x_1, x_1 + dx_1]$ to $[x_2, x_2 + dx_2]$ is:

$$\|x_2 - x_1\| \cdot \gamma(x_1, x_2) dx_1 dx_2$$

- ▶ and the total cost is:

$$C = \int_{x_1, x_2} \|x_2 - x_1\| \cdot \gamma(x_1, x_2) dx_1 dx_2 = \mathbb{E}_{x_1, x_2 \sim \gamma(x_1, x_2)} \|x_2 - x_1\|$$

Interpreting γ as a PDF

Formal definition

- ▶ Say, we have a moving plan $\gamma(x_1, x_2) \geq 0$:

$\gamma(x_1, x_2) dx_1 dx_2$ – how much dirt we're moving from
[$x_1, x_1 + dx_1$] to [$x_2, x_2 + dx_2$]

- ▶ Then, the cost of moving from [$x_1, x_1 + dx_1$] to [$x_2, x_2 + dx_2$] is:

$$\|x_2 - x_1\| \cdot \gamma(x_1, x_2) dx_1 dx_2$$

- ▶ and the total cost is:

$$C = \int_{x_1, x_2} \|x_2 - x_1\| \cdot \gamma(x_1, x_2) dx_1 dx_2 = \mathbb{E}_{x_1, x_2 \sim \gamma(x_1, x_2)} \|x_2 - x_1\|$$

Interpreting γ as a PDF

- ▶ Since we want to convert P to Q , the plan has to satisfy:

$$\int_{x_1} \gamma(x_1, x_2) dx_1 = Q(x_2),$$

$$\int_{x_2} \gamma(x_1, x_2) dx_2 = P(x_1)$$

Formal definition

- ▶ Let π be the set of all plans that convert P to Q , i.e.:

$$\pi = \left\{ \gamma: \quad \gamma \geq 0, \quad \int \gamma(x_1, x_2) dx_1 = Q(x_2), \quad \int \gamma(x_1, x_2) dx_2 = P(x_1) \right\}$$

- ▶ Then, the Wasserstein distance between P and Q is:

$$\text{EMD}(P, Q) = \inf_{\gamma \in \pi} \mathbb{E}_{x_1, x_2 \sim \gamma} \|x_2 - x_1\|$$

Formal definition

- ▶ Let π be the set of all plans that convert P to Q , i.e.:

$$\pi = \left\{ \gamma: \quad \gamma \geq 0, \quad \int \gamma(x_1, x_2) dx_1 = Q(x_2), \quad \int \gamma(x_1, x_2) dx_2 = P(x_1) \right\}$$

- ▶ Then, the Wasserstein distance between P and Q is:

$$\text{EMD}(P, Q) = \inf_{\gamma \in \pi} \mathbb{E}_{x_1, x_2 \sim \gamma} \|x_2 - x_1\|$$

Optimization over all transport plans – not too friendly

Formal definition

- ▶ Let π be the set of all plans that convert P to Q , i.e.:

$$\pi = \left\{ \gamma: \quad \gamma \geq 0, \quad \int \gamma(x_1, x_2) dx_1 = Q(x_2), \quad \int \gamma(x_1, x_2) dx_2 = P(x_1) \right\}$$

- ▶ Then, the Wasserstein distance between P and Q is:

$$\text{EMD}(P, Q) = \inf_{\gamma \in \pi} \mathbb{E}_{x_1, x_2 \sim \gamma} \|x_2 - x_1\|$$

Optimization over all transport plans – not too friendly

- ▶ Dual form (Kantorovich-Rubinstein duality):

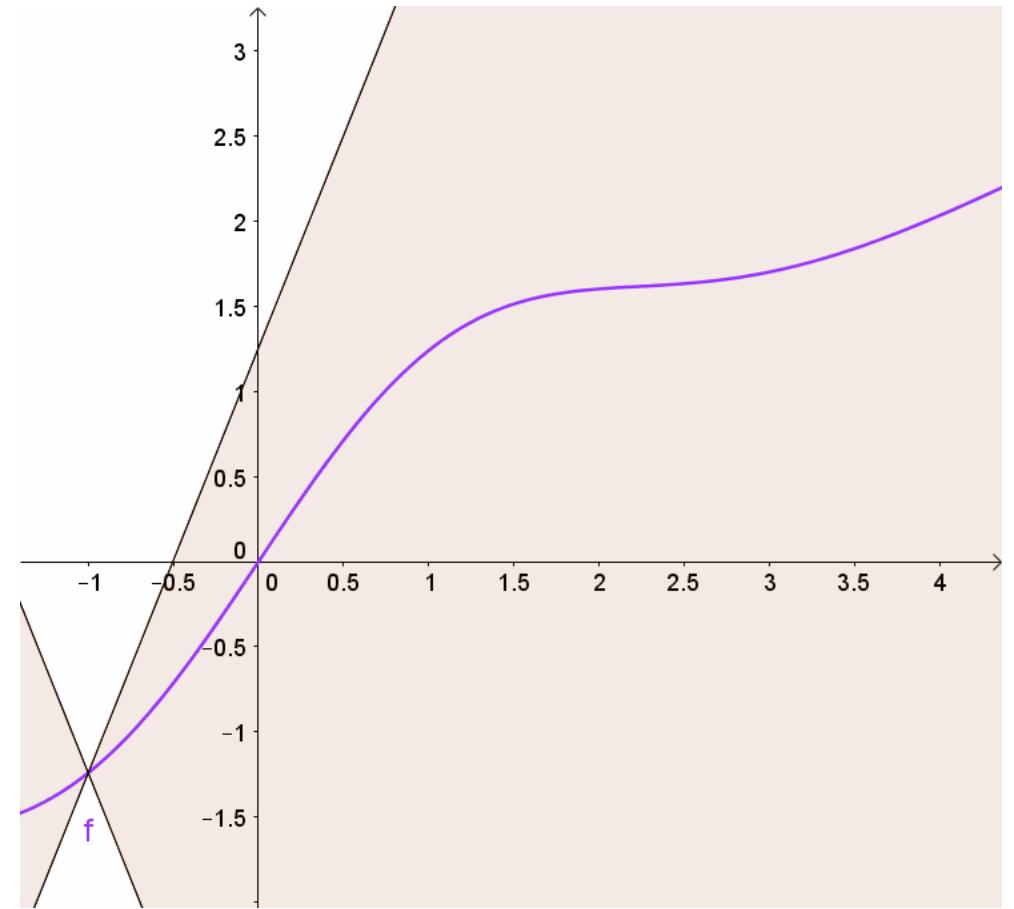
$$\text{EMD}(P, Q) = \sup_{\|f\|_L \leq 1} [\mathbb{E}_{x \sim P} f(x) - \mathbb{E}_{x \sim Q} f(x)]$$

Optimization over Lipschitz-1 continuous functions acting in $\mathcal{X} \rightarrow \mathbb{R}$

Lipschitz continuity

- ▶ f is Lipschitz- k continuous if
- ▶ there exists a constant $k \geq 0$, such that for all x_1 and x_2 :

$$|f(x_1) - f(x_2)| \leq k \cdot \|x_1 - x_2\|$$



img from https://en.wikipedia.org/wiki/Lipschitz_continuity

[intuition behind the dual form]

disclaimer: not a strict mathematical derivation

$$\text{EMD}(P, Q) = \inf_{\gamma \in \pi} \mathbb{E}_{x_1, x_2 \sim \gamma} \|x_1 - x_2\|$$

Let's add the following term to this expression:

$$+ \inf_{\gamma} \sup_f \mathbb{E}_{x_1, x_2 \sim \gamma} [\mathbb{E}_{s \sim P} f(s) - \mathbb{E}_{t \sim Q} f(t) - (f(x_1) - f(x_2))]$$

$f(x)$ — real-valued function

These cancel out when $\gamma \in \pi$
otherwise supremum over $f(x)$ goes to $+\infty$

Therefore, we can remove the $\gamma \in \pi$ condition from the whole expression:

$$= \inf_{\gamma} \sup_f \mathbb{E}_{x_1, x_2 \sim \gamma} [\|x_1 - x_2\| + \mathbb{E}_{s \sim P} f(s) - \mathbb{E}_{t \sim Q} f(t) - (f(x_1) - f(x_2))]$$

Infimum and supremum operations can be swapped under certain conditions
(satisfied here — see <https://vinctherrmann.github.io/blog/wasserstein/> for more detailed info)

[intuition behind the dual form]

disclaimer: not a strict mathematical derivation

$$= \sup_f \inf_{\gamma} \left[\mathbb{E}_{s \sim P} f(s) - \mathbb{E}_{t \sim Q} f(t) + \mathbb{E}_{x_1, x_2 \sim \gamma} [| |x_1 - x_2| | - (f(x_1) - f(x_2))] \right]$$

Consider the following case: $|f(a) - f(b)| \leq | |a - b| |, \quad \forall a, b$
We'll denote it as: $\|f\|_L \leq 1$

For such case this term is 0

Otherwise the whole expression is $-\infty$

Therefore we can finally rewrite the whole thing as:

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

WGAN

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

- ▶ The function can be expressed as a neural net – discriminator ('critic' in the original paper)

<https://arxiv.org/abs/1701.07875>

WGAN

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

- ▶ The function can be expressed as a neural net – discriminator ('critic' in the original paper)
- ▶ The expectations can be estimated as sample mean

WGAN

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

- ▶ The function can be expressed as a neural net – discriminator ('critic' in the original paper)
- ▶ Lipschitz-1 continuity can be replaced with Lipschitz-k continuity
 - In such case we'll estimate $k \times \text{EMD}(P, Q)$
- ▶ The expectations can be estimated as sample mean

<https://arxiv.org/abs/1701.07875>

WGAN

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

- ▶ The function can be expressed as a neural net – discriminator ('critic' in the original paper)
- ▶ Lipschitz-1 continuity can be replaced with Lipschitz-k continuity
 - In such case we'll estimate $k \times \text{EMD}(P, Q)$
- ▶ The expectations can be estimated as sample mean

We wouldn't know what k is, but it doesn't matter: all we want is to minimize the EMD!

WGAN

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

- ▶ The function can be expressed as a neural net – discriminator ('critic' in the original paper)
- ▶ Lipschitz-1 continuity can be replaced with Lipschitz-k continuity
 - In such case we'll estimate $k \times \text{EMD}(P, Q)$
 - Can be achieved by clipping the weights of the critic: $w \rightarrow \text{clip}(w, -c, c)$ with some constant c
- ▶ The expectations can be estimated as sample mean

We wouldn't know what k is, but it doesn't matter: all we want is to minimize the EMD!

WGAN

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: : α , the learning rate. c , the clipping parameter. m , the batch size. n_{critic} , the number of iterations of the critic per generator iteration.

Require: : w_0 , initial critic parameters. θ_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 prior samples.
5:      $g_w \leftarrow \nabla_w \left[ \frac{1}{m} \sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)})) \right]$ 
6:      $w \leftarrow w + \alpha \cdot \text{RMSPProp}(w, g_w)$ 
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:   $\theta \leftarrow \theta - \alpha \cdot \text{RMSPProp}(\theta, g_\theta)$ 
12: end while
```

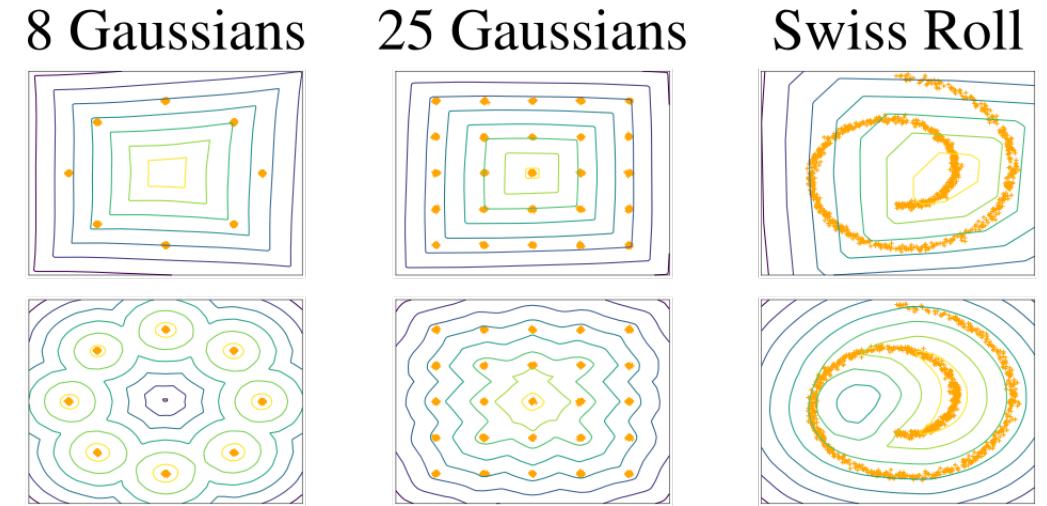
WGAN-GP

- ▶ Weight clipping makes the critic less expressive and the training harder to converge
- ▶ Optimal f should satisfy $\|\nabla f\| = 1$ almost everywhere under P and Q
- ▶ Also: $\|f\|_L \leq 1 \iff \|\nabla f\| \leq 1$
- ▶ Can replace weight clipping with a gradient penalty term:

$$\text{GP} = \lambda \mathbb{E}_{\tilde{x} \sim P_{\tilde{x}}} \left[\left(\|\nabla_{\tilde{x}} f(\tilde{x})\| - 1 \right)^2 \right]$$

or alternatively ('one-sided' penalty):

$$\text{GP} = \lambda \mathbb{E}_{\tilde{x} \sim P_{\tilde{x}}} \left[\max(0, \|\nabla_{\tilde{x}} f(\tilde{x})\| - 1)^2 \right]$$

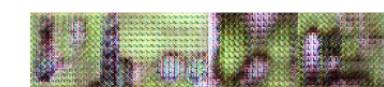


$$P_{\tilde{x}} : \begin{bmatrix} \tilde{x} = \alpha x_1 + (1 - \alpha) x_2 \\ \alpha \sim \text{Uniform}(0, 1) \\ x_1 \sim P \\ x_2 \sim Q \end{bmatrix}$$

<https://arxiv.org/abs/1704.00028>

WGAN-GP

This technique allowed for very deep networks to be used for GANs

DCGAN	LSGAN	WGAN (clipping)	WGAN-GP (ours)	
Baseline (G : DCGAN, D : DCGAN)				
G : No BN and a constant number of filters, D : DCGAN				
G : 4-layer 512-dim ReLU MLP, D : DCGAN				
No normalization in either G or D				
Gated multiplicative nonlinearities everywhere in G and D				
tanh nonlinearities everywhere in G and D				
101-layer ResNet G and D				

<https://arxiv.org/abs/1704.00028>

Some notable architectures



Conditional distributions

- ▶ It's often necessary to learn not just $P(x)$, but $P(x | a)$
 - E.g. generate the face of a person with a given hair color
- ▶ This can be achieved by:

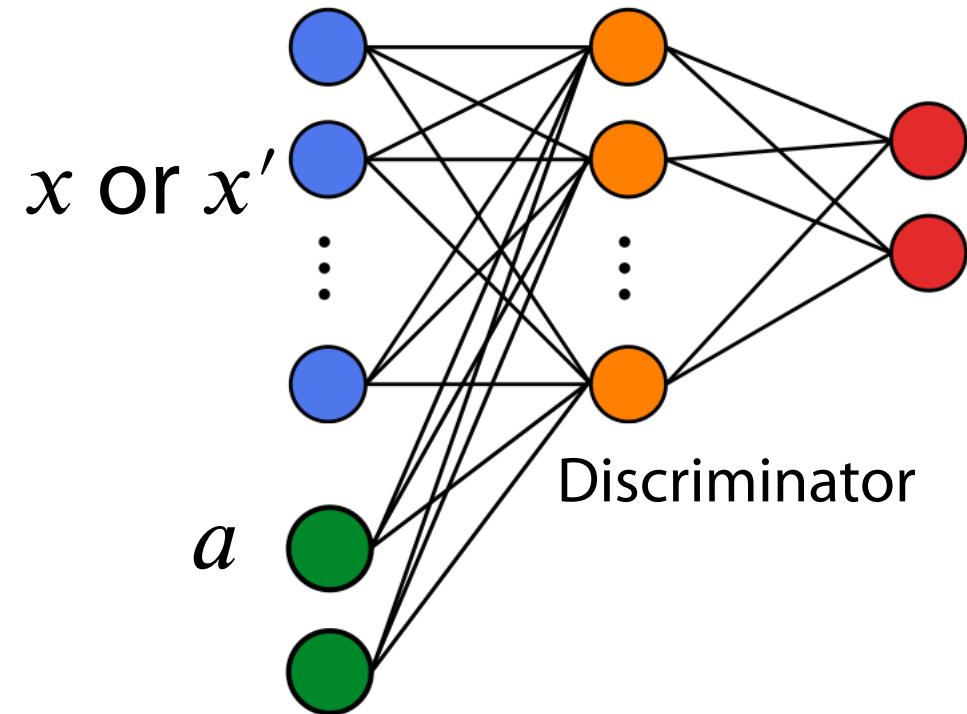
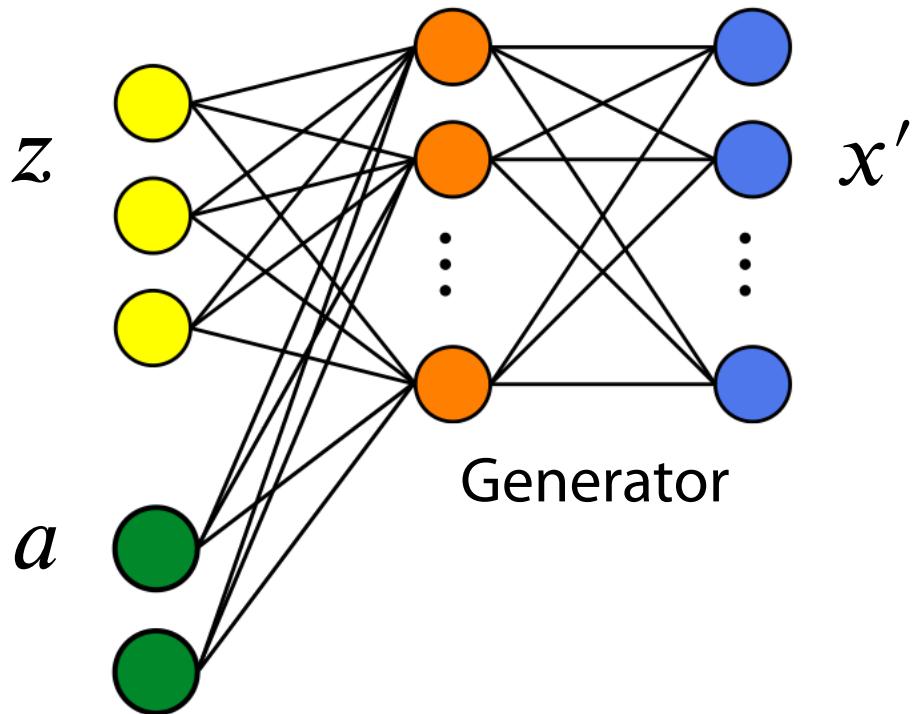
$$x'_j = G_\theta(z_j) \rightarrow x'_j = G_\theta(z_j, a_j)$$

$$D_\phi = D_\phi(x_i) \rightarrow D_\phi = D_\phi(x_i, a_i)$$

- ▶ I.e. we need to provide this information to the generator and discriminator

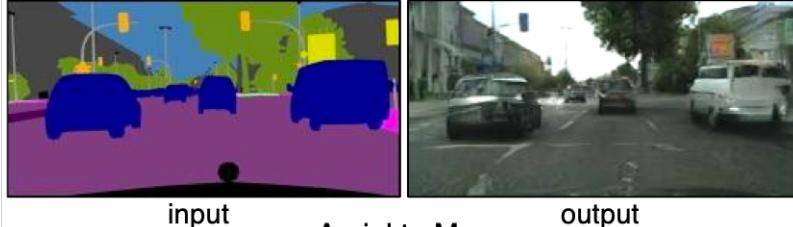
Conditional distributions

- ▶ Simple for fully-connected architectures



pix2pix

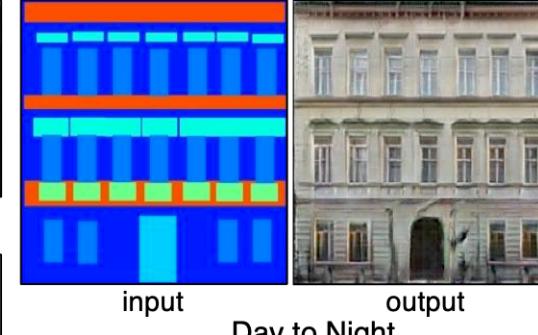
Labels to Street Scene



input

output

Labels to Facade



input

output

BW to Color



input

output

Aerial to Map



input

output

Day to Night



input

output

Edges to Photo



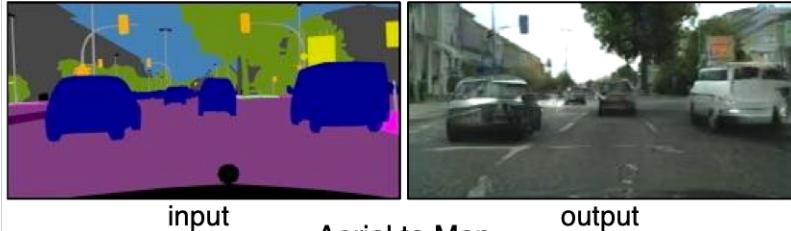
input

output

<https://arxiv.org/abs/1611.07004>

pix2pix

Labels to Street Scene



input

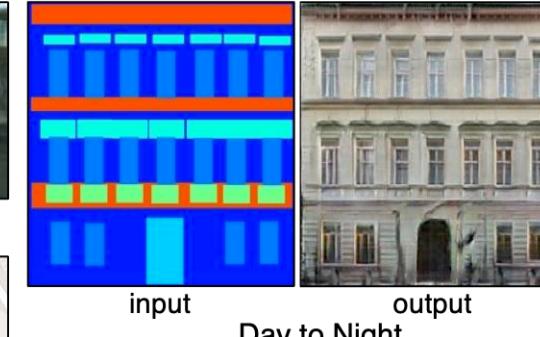
Aerial to Map



input

output

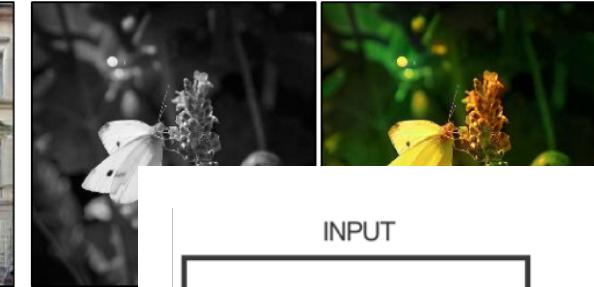
Labels to Facade



input

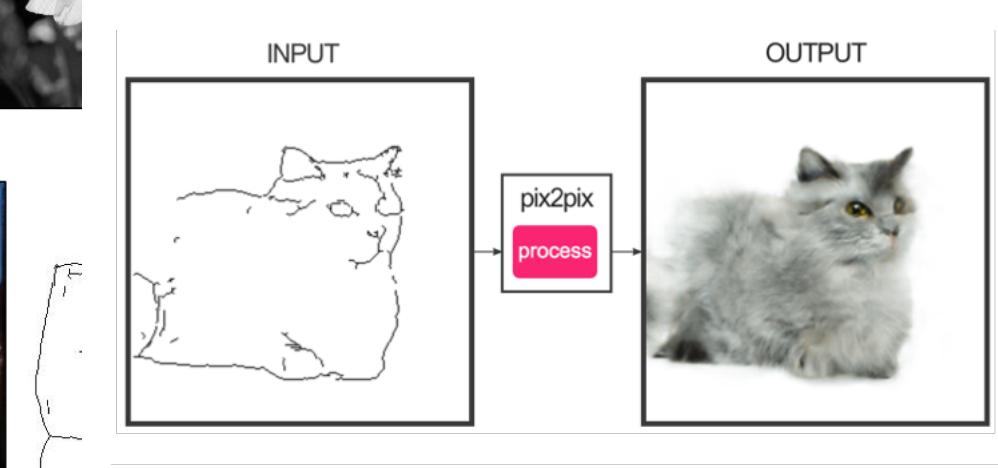
output

BW to Color



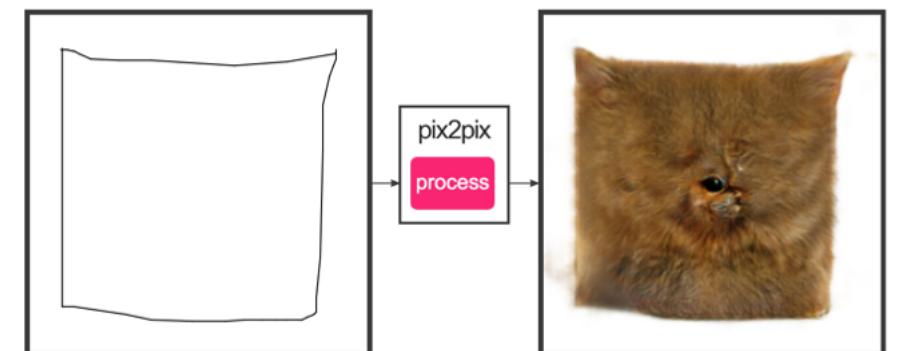
INPUT

OUTPUT



INPUT

OUTPUT



images generated at <https://affinelayer.com/pixsrv/>

pix2pix

<https://arxiv.org/abs/1611.07004>

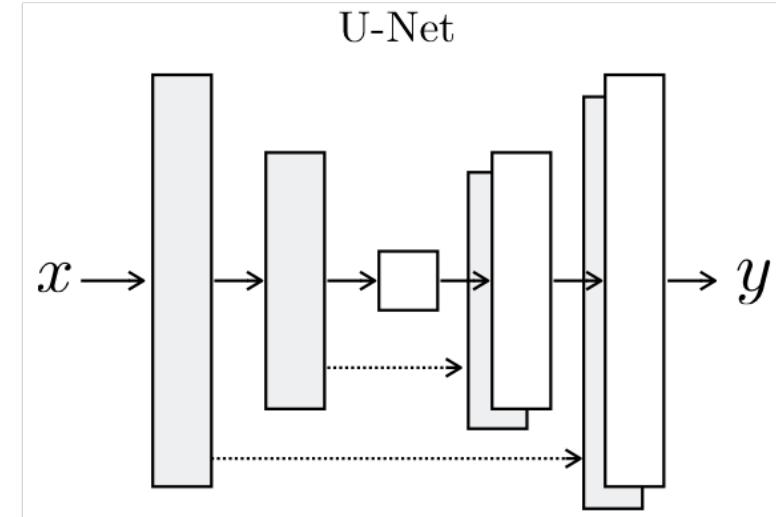
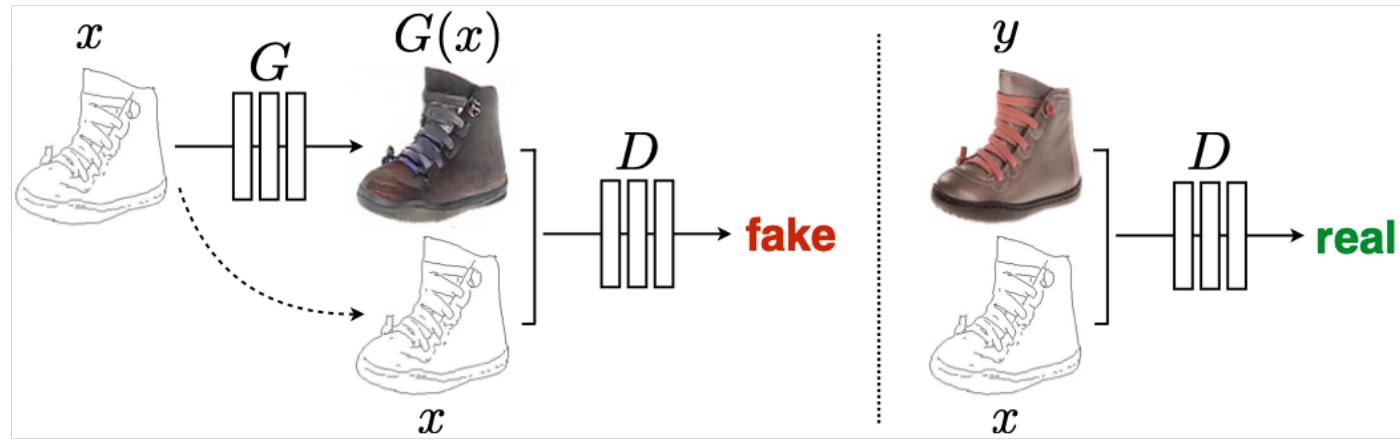
$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1].$$

Our final objective is

$$G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G).$$

L1 loss term to
capture low-frequency
information

- Discriminator doesn't need to classify the entire image
- Instead it 'convolutionally' scans smaller patches of the image and averages the result



'U-Net' generator
architecture

pix2pix

<https://arxiv.org/abs/1611.07004>

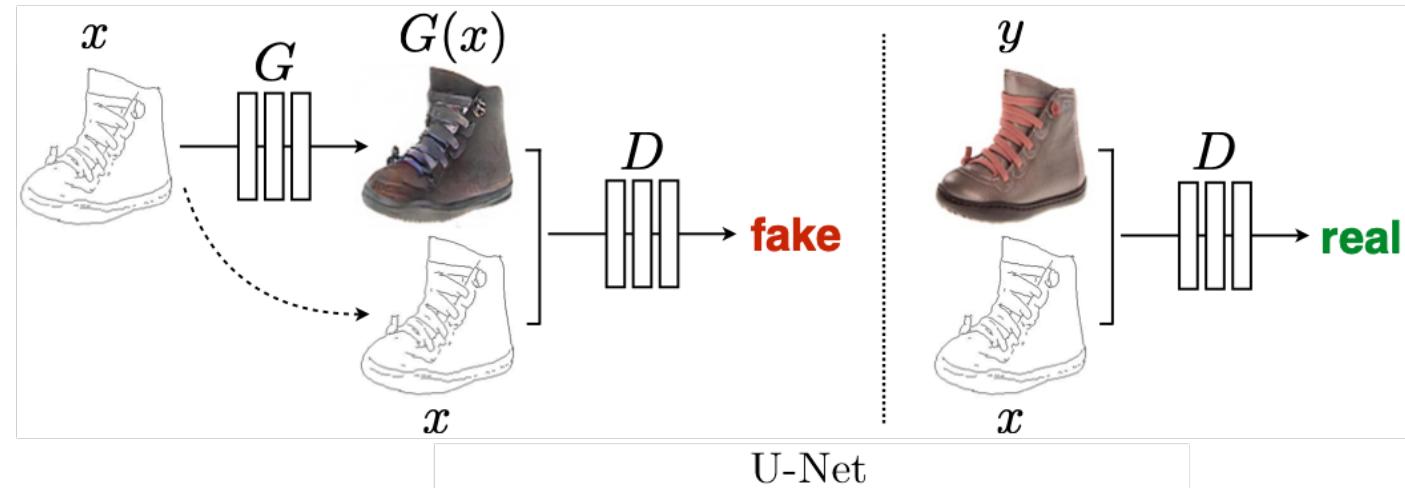
$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1].$$

Our final objective is

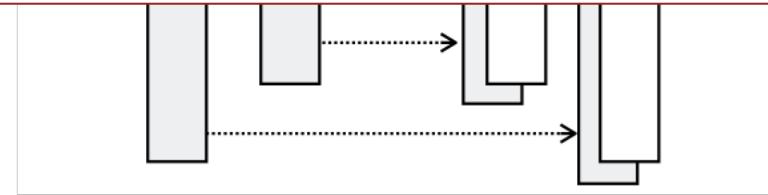
$$G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G).$$

L1 loss term to
capture low-frequency
information

- Discriminator doesn't need to classify the entire image
- Instead it 'convolutionally' scans smaller patches of the image and averages the result



No noise input to the generator:
«The generator simply **learned to ignore the noise** ...
Instead we provide noise only in the form of **dropout**,
applied at both training and test time»



'U-Net' generator
architecture

CycleGAN

Monet \leftrightarrow Photos



Monet \rightarrow photo

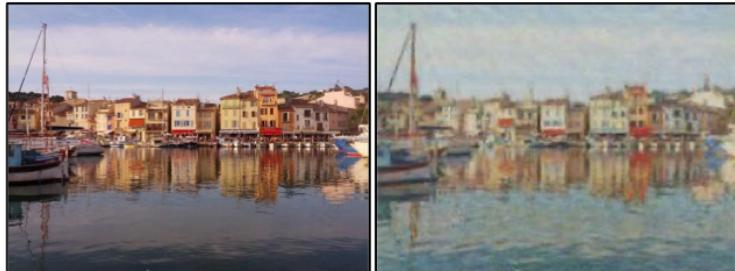


photo \rightarrow Monet

Zebras \leftrightarrow Horses



zebra \rightarrow horse

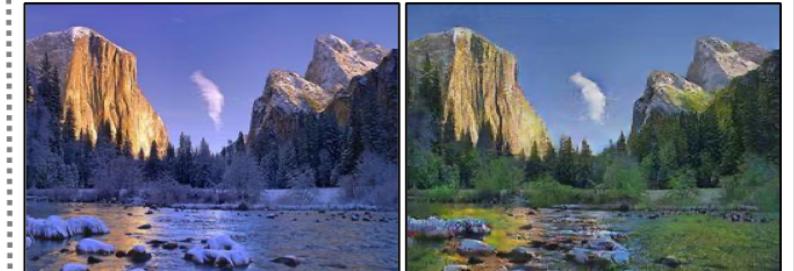


horse \rightarrow zebra

Summer \leftrightarrow Winter



summer \rightarrow winter



winter \rightarrow summer



Monet



Van Gogh



Cezanne

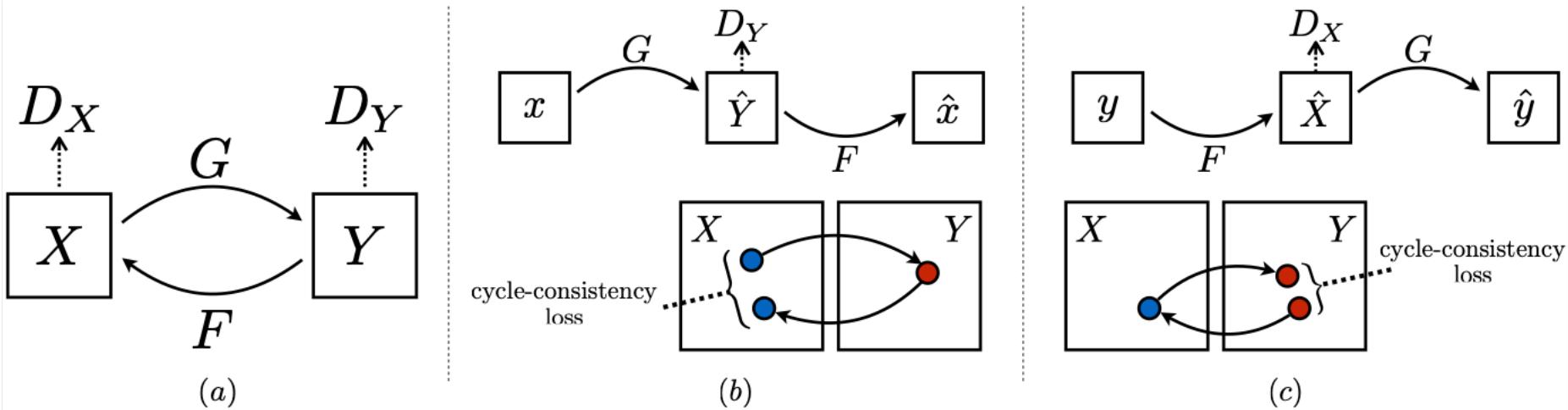


Ukiyo-e

<https://arxiv.org/abs/1703.10593>

CycleGAN

<https://arxiv.org/abs/1703.10593>



$$\begin{aligned} \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = & \mathbb{E}_{y \sim p_{\text{data}}(y)} [\log D_Y(y)] \\ & + \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log(1 - D_Y(G(x)))] \end{aligned} \quad (1)$$

$$\begin{aligned} \mathcal{L}(G, F, D_X, D_Y) = & \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ & + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \end{aligned} \quad (3)$$

$$\begin{aligned} \mathcal{L}_{\text{cyc}}(G, F) = & \mathbb{E}_{x \sim p_{\text{data}}(x)} [\|F(G(x)) - x\|_1] \\ & + \mathbb{E}_{y \sim p_{\text{data}}(y)} [\|G(F(y)) - y\|_1]. \end{aligned} \quad (2)$$

Progressive Growing of GANs

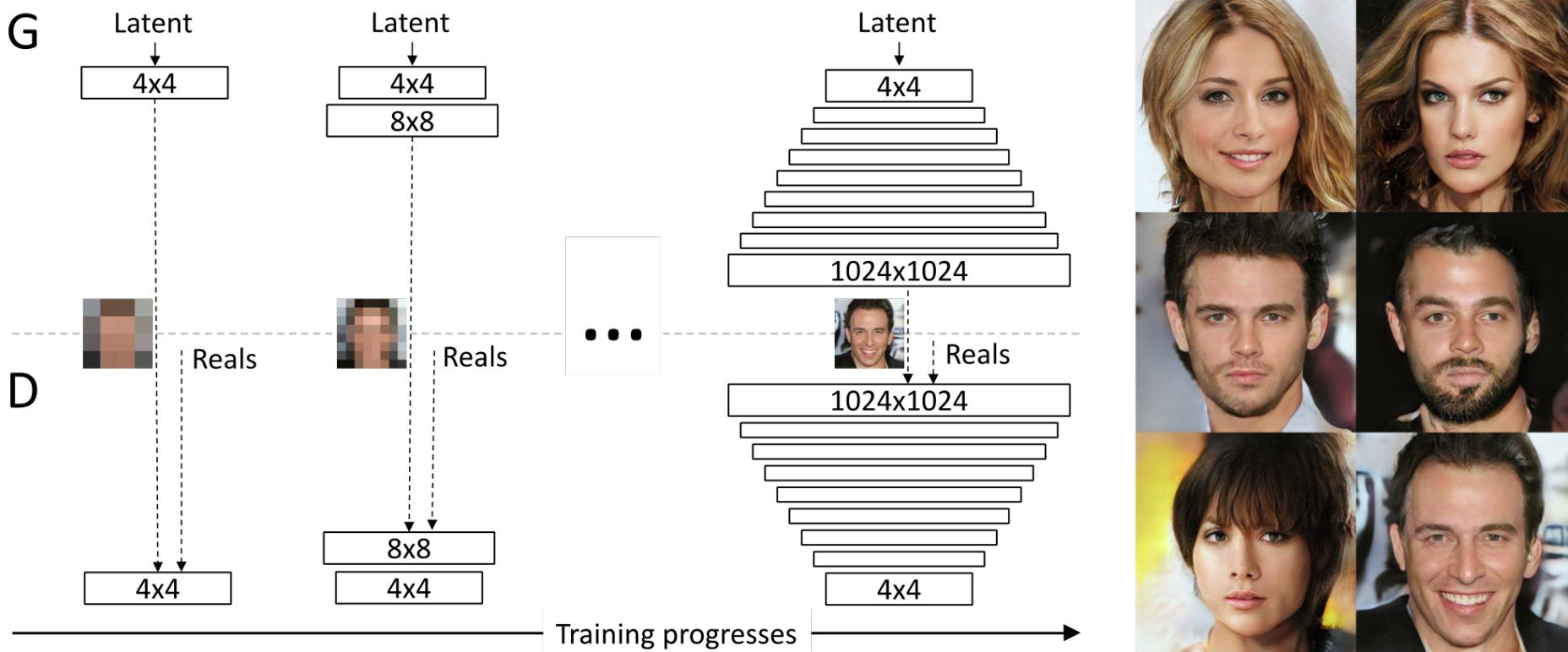


Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout the process. Here $N \times N$ refers to convolutional layers operating on $N \times N$ spatial resolution. This allows stable synthesis in high resolutions and also speeds up training considerably. On the right we show six example images generated using progressive growing at 1024×1024 .

Progressive Growing of GANs

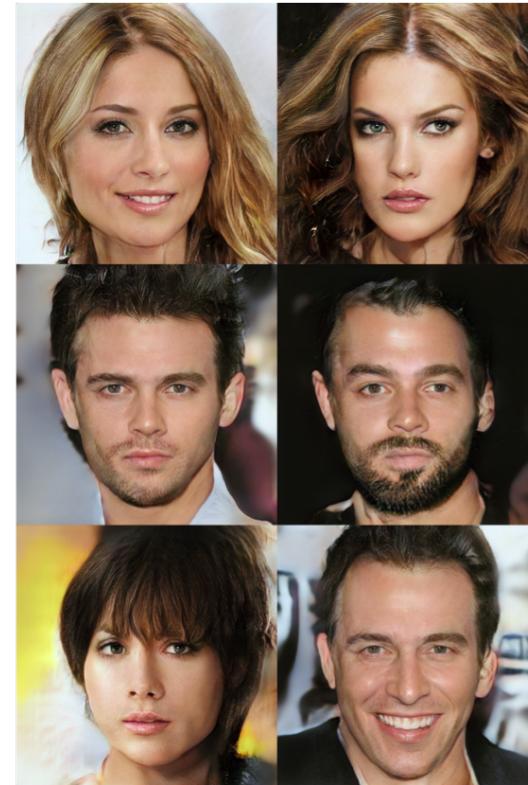
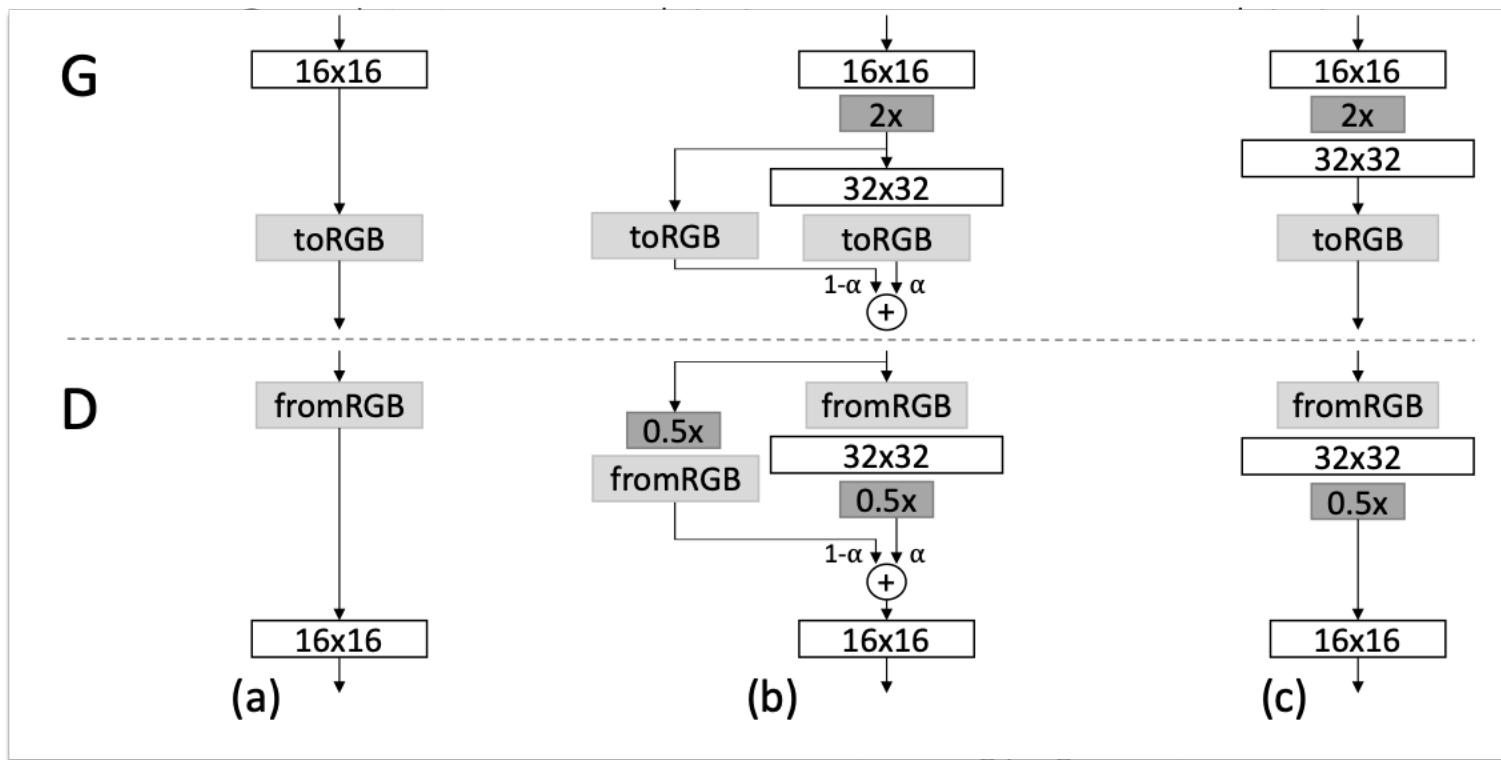


Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout the process. Here $N \times N$ refers to convolutional layers operating on $N \times N$ spatial resolution. This allows stable synthesis in high resolutions and also speeds up training considerably. On the right we show six example images generated using progressive growing at 1024×1024 .

Evaluating generative models



Evaluating generative models

- ▶ No single guide to follow
- ▶ Approaches are very problem-specific
 - E.g. perceived visual quality of generated images vs. quality of a generated dental crown model (<https://arxiv.org/abs/1804.00064>)
 - Most solutions are adapted to or invented for a given particular task
- ▶ We'll mention some approaches

Evaluating generative models

(most obvious thing to do)

- ▶ By-eye comparison (if your data allows that)
 - Compare individual objects or whole distributions (e.g. in projections)
 - There might be no need to do any complicated evaluation if the model results simply look bad

Evaluating generative models

(simple things to do)

- ▶ Compare meaningful physical characteristics (if applicable)
 - Means, medians, standard deviations, etc.
 - Correlations
- ▶ Statistical tests (χ^2 , Kolmogorov-Smirnov, etc.)
 - between individual dimensions or projections

Additional classifier

- ▶ Train an independent model (e.g. xgboost) to distinguish real and fake samples
- ▶ Evaluate your GAN by checking the classifier's score (e.g. ROC AUC)
- ▶ Pros:
 - An objective quality measure
- ▶ Cons:
 - Resource consuming
 - Requires hyper-parameter tuning
 - May get picky to things that are not important

Inception score

- ▶ Introduced in <https://arxiv.org/abs/1606.03498>
- ▶ Apply the Inception model (pre-trained image classifier) to obtain the conditional label distribution $p(y|x)$ for each image x
 - this should be low-entropy (the classifier should be certain)
- ▶ Calculate marginal $p(y) = \int p(y|x = G(z))p(z)dz$
 - this should be high-entropy (diversity of samples)
- ▶ Combining these two requirements:

$$\text{IS} = \exp \left[\mathbb{E}_x \left[\text{KL} \left(p(y|x) \parallel p(y) \right) \right] \right]$$

Fréchet inception distance (FID score)

- ▶ Introduced in <https://arxiv.org/abs/1706.08500>
- ▶ One of the drawbacks of IS is that it doesn't care about the true distribution
- ▶ Instead one can compare distributions of activations at some Inception layer (originally – last pooling layer)
- ▶ The authors proposed calculating the Fréchet (aka Wasserstein-2) distance
- ▶ Distance between multivariate Gaussian approximations:

$$\text{FID} = \|\mu_r - \mu_g\|^2 + \text{Tr}\left[\Sigma_r + \Sigma_g - 2\left(\Sigma_r \Sigma_g\right)^{1/2}\right]$$

Precision and Recall distance (PRD)

Definition 1. For $\alpha, \beta \in (0, 1]$, the probability distribution Q has precision α at recall β w.r.t. P if there exist distributions μ, ν_P and ν_Q such that

Decomposition with a common part

$$P = \beta\mu + (1 - \beta)\nu_P \quad \text{and} \quad Q = \alpha\mu + (1 - \alpha)\nu_Q. \quad (3)$$

Definition 2. The set of attainable pairs of precision and recall of a distribution Q w.r.t. a distribution P is denoted by $\text{PRD}(Q, P)$ and it consists of all (α, β) satisfying Definition 1 and the pair $(0, 0)$.

- The authors provide an algorithm to calculate it for discrete distributions
- They convert Inception activations to discrete distribution using k -means clustering

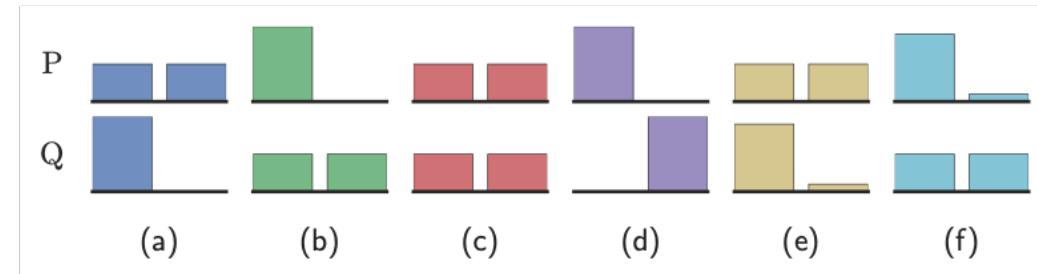


Figure 2: Intuitive examples of P and Q .

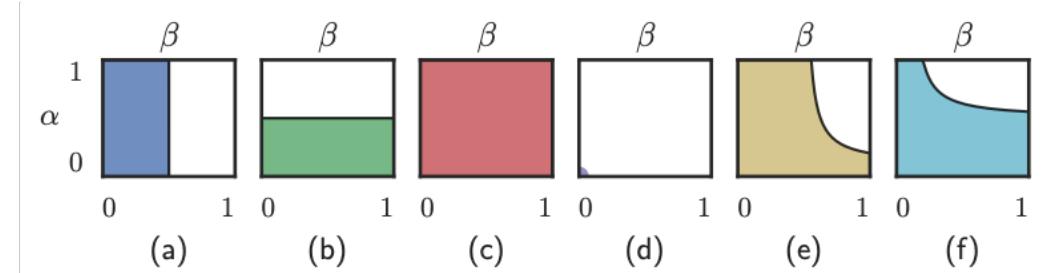


Figure 3: $\text{PRD}(Q, P)$ for the examples above.

<https://arxiv.org/abs/1806.00035>

More metrics...

► An extensive comparison of a large variety of measures:

— <https://arxiv.org/abs/1802.03446>

Measure	Description
Quantitative	<ul style="list-style-type: none"> • Log likelihood of explaining realworld held out/test data using a density estimated from the generated data (e.g. using KDE or Parzen window estimation). $L = \frac{1}{N} \sum_i \log P_{model}(\mathbf{x}_i)$ • The probability mass of the true data "covered" by the model distribution $C := P_{data}(dP_{model} > t)$ with t such that $P_{model}(dP_{model} > t) = 0.95$ • KLD between conditional and marginal label distributions over generated data. $\exp(\mathbb{E}_{\mathbf{x}} [\text{KL}(p(y \mathbf{x}) \ p(y))])$ • Encourages diversity within images sampled from a particular category. $\exp(\mathbb{E}_{\mathbf{x}_i} [\mathbb{E}_{\mathbf{x}_j} ([\text{KL}(P(y \mathbf{x}_i) \ P(y \mathbf{x}_j))])])$ • Similar to IS but also takes into account the prior distribution of the labels over real data. $\exp(\mathbb{E}_{\mathbf{x}} [\text{KL}(p(y \mathbf{x}) \ p(y^{train}))]) - \text{KL}(p(y) \ p(y^{train}))$ • Takes into account the KLD between distributions of training labels vs. predicted labels, as well as the entropy of predictions. $\text{KLL}(p(y^{train}) \ p(y)) + \mathbb{E}_{\mathbf{x}} [H(y \mathbf{x})]$ • Wasserstein-2 distance between multi-variate Gaussians fitted to data embedded into a feature space $FID(r, g) = \ \mu_r - \mu_g\ _2^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{\frac{1}{2}})$ • Measures the dissimilarity between two probability distributions P_r and P_g using samples drawn independently from each distribution. $M_k(P_r, P_g) = \mathbb{E}_{\mathbf{x}, \mathbf{x}' \sim P_r} [k(\mathbf{x}, \mathbf{x}')] - 2\mathbb{E}_{\mathbf{x} \sim P_r, \mathbf{y} \sim P_g} [k(\mathbf{x}, \mathbf{y})] + \mathbb{E}_{\mathbf{y}, \mathbf{y}' \sim P_g} [k(\mathbf{y}, \mathbf{y}')]$ • The critic (e.g. an NN) is trained to produce high values at real samples and low values at generated samples $\hat{W}(\mathbf{x}_{test}, \mathbf{x}_g) = \frac{1}{N} \sum_{i=1}^N \hat{f}(\mathbf{x}_{test}[i]) - \frac{1}{N} \sum_{i=1}^N \hat{f}(\mathbf{x}_g[i])$ • Measures the support size of a discrete (continuous) distribution by counting the duplicates (near duplicates) • Answers whether two samples are drawn from the same distribution (e.g. by training a binary classifier) • An indirect technique for evaluating the quality of unsupervised representations (e.g. feature extraction; FCN score). See also the GAN Quality Index (GQI) [41]. • Measures diversity of generated samples and covariate shift using classification methods. • Given two sets of samples from the same distribution, the number of samples that fall into a given bin should be the same up to sampling noise • Measures the distributions of distances to the nearest neighbors of some query images (i.e. diversity) • Compares two GANs by having them engaged in a battle against each other by swapping discriminators or generators. $p(\mathbf{x} y=1; M'_1)/p(\mathbf{x} y=1; M'_2) = (p(y=1 \mathbf{x}; D_1)p(\mathbf{x}; G_2))/(p(y=1 \mathbf{x}; D_2)p(\mathbf{x}; G_1))$ • Implements a tournament in which a player is either a discriminator that attempts to distinguish between real and fake data or a generator that attempts to fool the discriminators into accepting fake data as real. • Compares n GANs based on the idea that if the generated samples are closer to real ones, more epochs would be needed to distinguish them from real samples. • Adversarial Accuracy. Computes the classification accuracies achieved by the two classifiers, one trained on real data and another on generated data, on a labeled validation set to approximate $P_g(y \mathbf{x})$ and $P_r(y \mathbf{x})$. Adversarial Divergence: Computes $\text{KLL}(P_g(y \mathbf{x}), P_r(y \mathbf{x}))$ • Compares geometrical properties of the underlying data manifold between real and generated data. • Measures the reconstruction error (e.g. L_2 norm) between a test image and its closest generated image by optimizing for z (i.e. $\min_z \ G(z) - \mathbf{x}^{(test)}\ ^2$) • Evaluates the quality of generated images using measures such as SSIM, PSNR, and sharpness difference • Evaluates how similar low-level statistics of generated images are to those of natural scenes in terms of mean power spectrum, distribution of random filter responses, contrast distribution, etc. • These measures are used to quantify the degree of overfitting in GANs, often over toy datasets.
	<ul style="list-style-type: none"> • To detect overfitting, generated samples are shown next to their nearest neighbors in the training set • In these experiments, participants are asked to distinguish generated samples from real images in a short presentation time (e.g. 100 ms); i.e. real v.s fake • Participants are asked to rank models in terms of the fidelity of their generated images (e.g. pairs, triples) • Over datasets with known modes (e.g. a GMM or a labeled dataset), modes are computed as by measuring the distances of generated data to mode centers • Regards exploring and illustrating the internal representation and dynamics of models (e.g. space continuity) as well as visualizing learned features
Qualitative	<ol style="list-style-type: none"> 1. Nearest Neighbors
	<ol style="list-style-type: none"> 2. Rapid Scene Categorization [18]
	<ol style="list-style-type: none"> 3. Preference Judgment [54, 55, 56, 57]
	<ol style="list-style-type: none"> 4. Mode Drop and Collapse [58, 59]
	<ol style="list-style-type: none"> 5. Network Internals [1, 60, 61, 62, 63, 64]

Summary

- ▶ GANs – a rather broad field of ML
 - Lots of different architectures, this lecture doesn't pretend to be comprehensive
- ▶ May say, that in GANs we 'learn' the loss for the generator
- ▶ Wasserstein GAN is a useful technique that allows really deep networks to be used for data generation
- ▶ Finding universal quality evaluation method is rather an open question

GANs for fast simulation

- Quite a developing field!
- Important note: one cannot increase the statistics with GANs
- GANs rather memorize and interpolate the available data

K. Matchev, P. Shyamsundar, Uncertainties associated with GAN-generated datasets in high energy physics, arXiv:2002.06307 [hep-ph]

- [8] A. Maevskiy *et al.* [LHCb Collaboration], “Fast Data-Driven Simulation of Cherenkov Detectors Using Generative Adversarial Networks,” arXiv:1905.11825 [physics.ins-det].
- [9] D. Belayneh *et al.*, “Calorimetry with Deep Learning: Particle Simulation and Reconstruction for Collider Physics,” arXiv:1912.06794 [physics.ins-det].
- [10] J. R. Vlimant, F. Pantaleo, M. Pierini, V. Loncar, S. Vallecorsa, D. Anderson, T. Nguyen and A. Zlokapa, “Large-Scale Distributed Training Applied to Generative Adversarial Networks for Calorimeter Simulation,” EPJ Web Conf. **214**, 06025 (2019) doi:10.1051/epjconf/201921406025.
- [11] D. Lancierini, P. Owen and N. Serra, “Simulating the LHCb hadron calorimeter with generative adversarial networks,” Nuovo Cim. C **42**, no. 4, 197 (2019) doi:10.1393/ncc/2019-19197-3.
- [12] L. de Oliveira, M. Paganini and B. Nachman, “Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis,” Comput. Softw. Big Sci. **1**, no. 1, 4 (2017) doi:10.1007/s41781-017-0004-6 [arXiv:1701.05927 [stat.ML]].
- [13] S. Carrazza and F. A. Dreyer, “Lund jet images from generative and cycle-consistent adversarial networks,” Eur. Phys. J. C **79**, no. 11, 979 (2019) doi:10.1140/epjc/s10052-019-7501-1 [arXiv:1909.01359 [hep-ph]].
- [14] M. Paganini, L. de Oliveira and B. Nachman, “Accelerating Science with Generative Adversarial Networks: An Application to 3D Particle Showers in Multilayer Calorimeters,” Phys. Rev. Lett. **120**, no. 4, 042003 (2018) doi:10.1103/PhysRevLett.120.042003 [arXiv:1705.02355 [hep-ex]].
- [15] L. de Oliveira, M. Paganini and B. Nachman, “Controlling Physical Attributes in GAN-Accelerated Simulation of Electromagnetic Calorimeters,” J. Phys. Conf. Ser. **1085**, no. 4, 042017 (2018) doi:10.1088/1742-6596/1085/4/042017 [arXiv:1711.08813 [hep-ex]].
- [16] M. Paganini, L. de Oliveira and B. Nachman, “CaloGAN : Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks,” Phys. Rev. D **97**, no. 1, 014021 (2018) doi:10.1103/PhysRevD.97.014021 [arXiv:1712.10321 [hep-ex]].
- [17] F. Carminati, A. Gheata, G. Khattak, P. Mendez Lorenzo, S. Sharan and S. Vallecorsa, “Three dimensional Generative Adversarial Networks for fast simulation,” J. Phys. Conf. Ser. **1085**, no. 3, 032016 (2018) doi:10.1088/1742-6596/1085/3/032016.
- [18] M. Erdmann, L. Geiger, J. Glombitza and D. Schmidt, “Generating and refining particle detector simulations using the Wasserstein distance in adversarial networks,” Comput. Softw. Big Sci. **2**, no. 1, 4 (2018) doi:10.1007/s41781-018-0008-x [arXiv:1802.03325 [astro-ph.IM]].
- [19] P. Musella and F. Pandolfi, “Fast and Accurate Simulation of Particle Detectors Using Generative Adversarial Networks,” Comput. Softw. Big Sci. **2**, no. 1, 8 (2018) doi:10.1007/s41781-018-0015-y [arXiv:1805.00850 [hep-ex]].
- [20] M. Erdmann, J. Glombitza and T. Quast, “Precise simulation of electromagnetic calorimeter showers using a Wasserstein Generative Adversarial Network,” Comput. Softw. Big Sci. **3**, no. 1, 4 (2019) doi:10.1007/s41781-018-0019-7 [arXiv:1807.01954 [physics.ins-det]].
- [21] S. Vallecorsa, F. Carminati and G. Khattak, “3D convolutional GAN for fast simulation,” EPJ Web Conf. **214**, 02010 (2019) doi:10.1051/epjconf/201921402010.
- [22] S. Otten, S. Caron, W. de Swart, M. van Beekveld, L. Hendriks, C. van Leeuwen, D. Podareanu, R. R. de Austri and R. Verheyen, “Event Generation and Statistical Sampling for Physics with Deep Generative Models and a Density Information Buffer,” arXiv:1901.00875 [hep-ph].
- [23] A. Butter, T. Plehn and R. Winterhalder, “How to GAN LHC Events,” SciPost Phys. **7**, no. 6, 075 (2019) doi:10.21468/SciPostPhys.7.6.075 [arXiv:1907.03764 [hep-ph]].
- [24] C. Ahdida *et al.* [SHiP Collaboration], “Fast simulation of muons produced at the SHiP experiment using Generative Adversarial Networks,” JINST **14**, P11028 (2019) doi:10.1088/1748-0221/14/11/P11028 [arXiv:1909.04451 [physics.ins-det]].
- [25] S. Farrell, W. Bhimji, T. Kurth, M. Mustafa, D. Bard, Z. Lukic, B. Nachman and H. Patton, “Next Generation Generative Neural Networks for HEP,” EPJ Web Conf. **214**, 09005 (2019) doi:10.1051/epjconf/201921409005.
- [26] J. Arjona Martínez, T. Q. Nguyen, M. Pierini, M. Spiropulu and J. R. Vlimant, “Particle Generative Adversarial Networks for full-event simulation at the LHC and their application to pileup description,” arXiv:1912.02748 [hep-ex].
- [27] B. Hashemi, N. Amin, K. Datta, D. Olivito and M. Pierini, “LHC analysis-specific datasets with Generative Adversarial Networks,” arXiv:1901.05282 [hep-ex].
- [28] R. Di Sipio, M. Faucci Giannelli, S. Ketabchi Haghight and S. Palazzo, “A Generative-Adversarial Network Approach for the Simulation of QCD Dijet Events at the LHC,” PoS LeptonPhoton **2019**, 050 (2019) doi:10.22323/1.367.0050.
- [29] R. Di Sipio, M. Faucci Giannelli, S. Ketabchi Haghight and S. Palazzo, “DijetGAN: A Generative-Adversarial Network Approach for the Simulation of QCD Dijet Events at the LHC,” JHEP **1908**, 110 (2020) doi:10.1007/JHEP08(2019)110 [arXiv:1903.02433 [hep-ex]].
- [30] Y. Alanazi, N. Sato, T. Liu, W. Melnitchouk, M. P. Kuchera, E. Pritchard, M. Robertson, R. Strauss, L. Velasco and Y. Li, “Simulation of electron-proton scattering events by a Feature-Augmented and Transformed Generative Adversarial Network (FAT-GAN),” arXiv:2001.11103 [hep-ph].

GANs for fast simulation

► Quite a developing field!

(Shameless plug)

► Important note: one cannot increase the statistics with GANs

► GANs rather memorize and interpolate the available data

K. Matchev, P. Shyamsundar, Uncertainties associated with GAN-generated datasets in high energy physics, arXiv:2002.06307 [hep-ph]

- [8] A. Maevskiy *et al.* [LHCb Collaboration], “Fast Data-Driven Simulation of Cherenkov Detectors Using Generative Adversarial Networks,” arXiv:1905.11825 [physics.ins-det].
- [9] D. Belaynen *et al.*, “Calorimetry with Deep Learning: Particle Simulation and Reconstruction for Collider Physics,” arXiv:1912.06794 [physics.ins-det].
- [10] J. R. Vlimant, F. Pantaleo, M. Pierini, V. Loncar, S. Vallecorsa, D. Anderson, T. Nguyen and A. Zlokapa, “Large-Scale Distributed Training Applied to Generative Adversarial Networks for Calorimeter Simulation,” EPJ Web Conf. **214**, 06025 (2019) doi:10.1051/epjconf/201921406025.
- [11] D. Lancierini, P. Owen and N. Serra, “Simulating the LHCb hadron calorimeter with generative adversarial networks,” Nuovo Cim. C **42**, no. 4, 197 (2019) doi:10.1393/ncc/2019-19197-3.
- [12] L. de Oliveira, M. Paganini and B. Nachman, “Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis,” Comput. Softw. Big Sci. **1**, no. 1, 4 (2017) doi:10.1007/s41781-017-0004-6 [arXiv:1701.05927 [stat.ML]].
- [13] S. Carrazza and F. A. Dreyer, “Lund jet images from generative and cycle-consistent adversarial networks,” Eur. Phys. J. C **79**, no. 11, 979 (2019) doi:10.1140/epjc/s10052-019-7501-1 [arXiv:1909.01359 [hep-ph]].
- [14] M. Paganini, L. de Oliveira and B. Nachman, “Accelerating Science with Generative Adversarial Networks: An Application to 3D Particle Showers in Multilayer Calorimeters,” Phys. Rev. Lett. **120**, no. 4, 042003 (2018) doi:10.1103/PhysRevLett.120.042003 [arXiv:1705.02355 [hep-ex]].
- [15] L. de Oliveira, M. Paganini and B. Nachman, “Controlling Physical Attributes in GAN-Accelerated Simulation of Electromagnetic Calorimeters,” J. Phys. Conf. Ser. **1085**, no. 4, 042017 (2018) doi:10.1088/1742-6596/1085/4/042017 [arXiv:1711.08813 [hep-ex]].
- [16] M. Paganini, L. de Oliveira and B. Nachman, “CaloGAN : Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks,” Phys. Rev. D **97**, no. 1, 014021 (2018) doi:10.1103/PhysRevD.97.014021 [arXiv:1712.10321 [hep-ex]].
- [17] F. Carminati, A. Gheata, G. Khattak, P. Mendez Lorenzo, S. Sharan and S. Vallecorsa, “Three dimensional Generative Adversarial Networks for fast simulation,” J. Phys. Conf. Ser. **1085**, no. 3, 032016 (2018) doi:10.1088/1742-6596/1085/3/032016.
- [18] M. Erdmann, L. Geiger, J. Glombitza and D. Schmidt, “Generating and refining particle detector simulations using the Wasserstein distance in adversarial networks,” Comput. Softw. Big Sci. **2**, no. 1, 4 (2018) doi:10.1007/s41781-018-0008-x [arXiv:1802.03325 [astro-ph.IM]].
- [19] P. Musella and F. Pandolfi, “Fast and Accurate Simulation of Particle Detectors Using Generative Adversarial Networks,” Comput. Softw. Big Sci. **2**, no. 1, 8 (2018) doi:10.1007/s41781-018-0015-y [arXiv:1805.00850 [hep-ex]].
- [20] M. Erdmann, J. Glombitza and T. Quast, “Precise simulation of electromagnetic calorimeter showers using a Wasserstein Generative Adversarial Network,” Comput. Softw. Big Sci. **3**, no. 1, 4 (2019) doi:10.1007/s41781-018-0019-7 [arXiv:1807.01954 [physics.ins-det]].
- [21] S. Vallecorsa, F. Carminati and G. Khattak, “3D convolutional GAN for fast simulation,” EPJ Web Conf. **214**, 02010 (2019) doi:10.1051/epjconf/201921402010.
- [22] S. Otten, S. Caron, W. de Swart, M. van Beekveld, L. Hendriks, C. van Leeuwen, D. Podareanu, R. R. de Austri and R. Verheyen, “Event Generation and Statistical Sampling for Physics with Deep Generative Models and a Density Information Buffer,” arXiv:1901.00875 [hep-ph].
- [23] A. Butter, T. Plehn and R. Winterhalder, “How to GAN LHC Events,” SciPost Phys. **7**, no. 6, 075 (2019) doi:10.21468/SciPostPhys.7.6.075 [arXiv:1907.03764 [hep-ph]].
- [24] C. Ahdida *et al.* [SHiP Collaboration], “Fast simulation of muons produced at the SHiP experiment using Generative Adversarial Networks,” JINST **14**, P11028 (2019) doi:10.1088/1748-0221/14/11/P11028 [arXiv:1909.04451 [physics.ins-det]].
- [25] S. Farrell, W. Bhimji, T. Kurth, M. Mustafa, D. Bard, Z. Lukic, B. Nachman and H. Patton, “Next Generation Generative Neural Networks for HEP,” EPJ Web Conf. **214**, 09005 (2019) doi:10.1051/epjconf/201921409005.
- [26] J. Arjona Martínez, T. Q. Nguyen, M. Pierini, M. Spiropulu and J. R. Vlimant, “Particle Generative Adversarial Networks for full-event simulation at the LHC and their application to pileup description,” arXiv:1912.02748 [hep-ex].
- [27] B. Hashemi, N. Amin, K. Datta, D. Olivito and M. Pierini, “LHC analysis-specific datasets with Generative Adversarial Networks,” arXiv:1901.05282 [hep-ex].
- [28] R. Di Sipio, M. Faucci Giannelli, S. Ketabchi Haghight and S. Palazzo, “A Generative-Adversarial Network Approach for the Simulation of QCD Dijet Events at the LHC,” PoS LeptonPhoton **2019**, 050 (2019) doi:10.22323/1.367.0050.
- [29] R. Di Sipio, M. Faucci Giannelli, S. Ketabchi Haghight and S. Palazzo, “DijetGAN: A Generative-Adversarial Network Approach for the Simulation of QCD Dijet Events at the LHC,” JHEP **1908**, 110 (2020) doi:10.1007/JHEP08(2019)110 [arXiv:1903.02433 [hep-ex]].
- [30] Y. Alanazi, N. Sato, T. Liu, W. Melnitchouk, M. P. Kuchera, E. Pritchard, M. Robertson, R. Strauss, L. Velasco and Y. Li, “Simulation of electron-proton scattering events by a Feature-Augmented and Transformed Generative Adversarial Network (FAT-GAN),” arXiv:2001.11103 [hep-ph].

Thank you!



al-maeeni@hse.ru



@afdee1c



@AFDEE1C