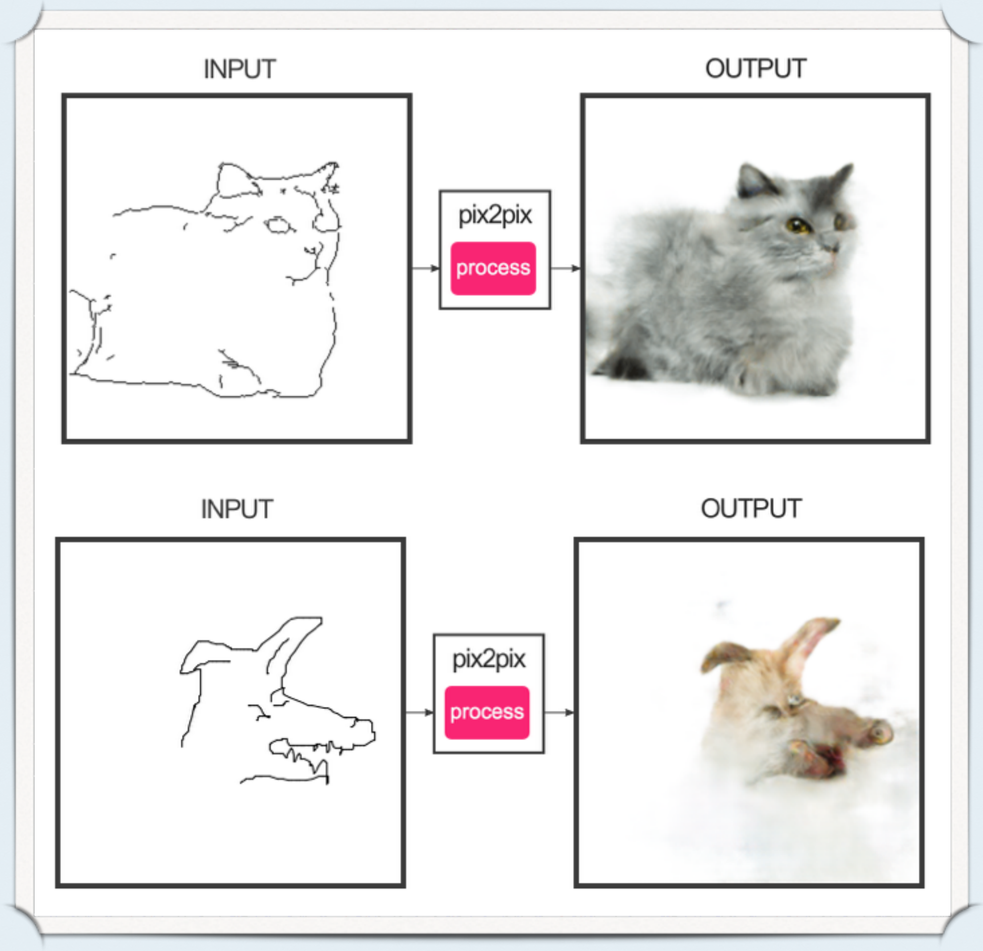
MACHINE LEARNING & DATA MINING



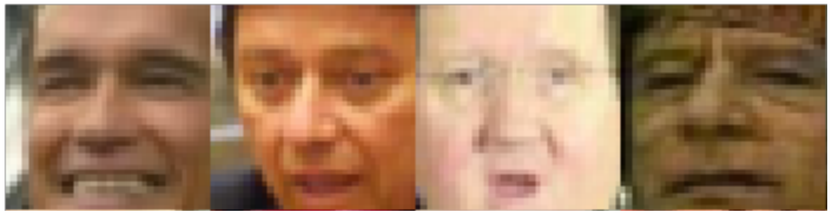
images generated at https://affinelayer.com/pixsrv/

Generative Models

Problem statement:

Given a finite sample of i.i.d. examples from an unknown distribution

$$\{\boldsymbol{x}_i\} \sim p_{\text{data}}$$

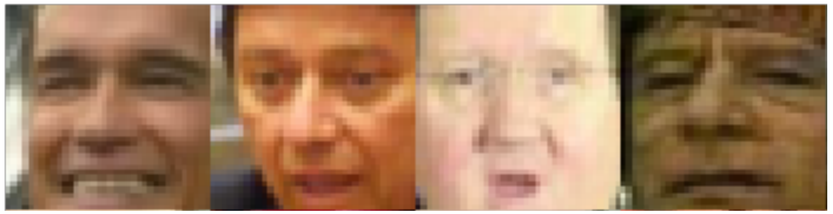


Generative Models

Problem statement:

Given a finite sample of i.i.d. examples from an unknown distribution

$$\{\boldsymbol{x}_i\} \sim p_{\text{data}}$$



We want to learn to draw more samples from that distribution

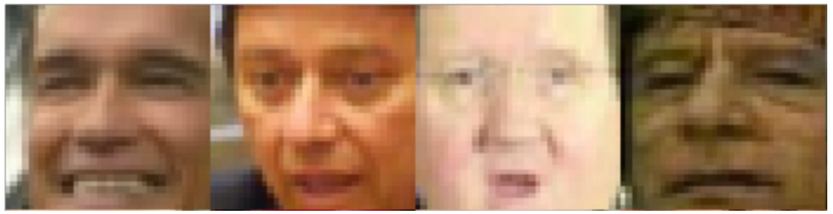
$$\{\boldsymbol{x}_j\} \sim p_{\text{data}}$$

Generative Models

Problem statement:

Given a finite sample of i.i.d. examples from an unknown distribution

$$\{\boldsymbol{x}_i\} \sim p_{\text{data}}$$



We want to learn to draw more samples from ~~that distribution~~

a ‘similar’ distribution

$$\{\boldsymbol{x}_j\} \sim p_{\text{data}}$$

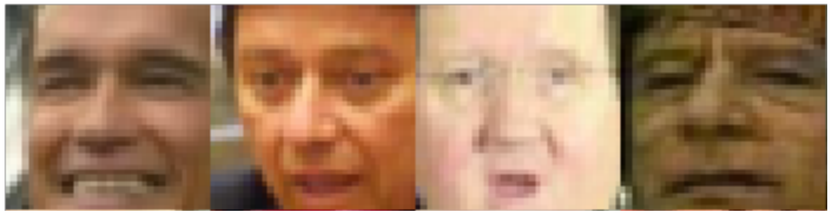
$$p_{gen,\theta} \xrightarrow{\theta \to \theta^*} p_{\text{data}}$$

Generative Models

Problem statement:

Given a finite sample of i.i.d. examples from an unknown distribution

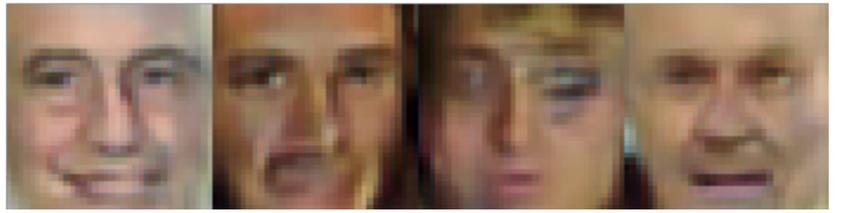
$$\{\boldsymbol{x}_i\} \sim p_{\text{data}}$$



We want to learn to draw more samples from ~~that distribution~~ **a 'similar' distribution**

$$\{\boldsymbol{x}_j\} \sim p_{\text{data}}$$

$$\{\tilde{\boldsymbol{x}}_j\} \sim p_{gen,\theta}$$



$$p_{gen,\theta} \xrightarrow{\theta \to \theta^*} p_{\text{data}}$$

Generative Adversarial Networks

$\{\tilde{\boldsymbol{x}}_j\} \sim p_{gen,\theta}$ — this distribution is built the following way:

$$\tilde{\boldsymbol{x}}_j = G_\theta(\boldsymbol{z}_j)$$

$$\boldsymbol{z}_j \sim N(\mathbf{0}, \mathbb{I})$$

G_θ — generator neural network

I. Goodfellow, et. al. *Generative adversarial nets*. In Advances in neural information processing systems, pages 2672–2680, 2014.

Generative Adversarial Networks

$\{\tilde{\boldsymbol{x}}_j\} \sim p_{gen,\theta}$ — this distribution is built the following way:

$$\tilde{\boldsymbol{x}}_j = G_\theta(\boldsymbol{z}_j)$$

$$\boldsymbol{z}_j \sim N(\mathbf{0}, \mathbb{I})$$

G_θ — generator neural network

We add a classifying network D_ϕ (discriminator) to distinguish between the real and generated samples and train both as follows:

$$V(\phi, \theta) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} \left[log D_\phi(\boldsymbol{x}) \right] + \mathbb{E}_{\tilde{\boldsymbol{x}} \sim p_{gen,\theta}} \left[log \left(1 - D_\phi(\tilde{\boldsymbol{x}}) \right) \right]$$

$$V(\phi, \theta) \rightarrow \min_\theta \max_\phi V(\phi, \theta)$$

I. Goodfellow, et. al. *Generative adversarial nets*. In Advances in neural information processing systems, pages 2672–2680, 2014.

Generative Adversarial Networks

$\{\tilde{\boldsymbol{x}}_j\} \sim p_{gen,\theta}$ — this distribution is built the following way:

$$\tilde{\boldsymbol{x}}_j = G_\theta(\boldsymbol{z}_j)$$

$$\boldsymbol{z}_j \sim N(\mathbf{0}, \mathbb{I})$$

G_θ — generator neural network

Probability the sample was drawn from data

Probability the sample was generated

We add a classifying network D_ϕ (discriminator) to distinguish between the real and generated samples and train both as follows:

$$V(\phi,\theta) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[log D_\phi(\boldsymbol{x})\right] + \mathbb{E}_{\tilde{\boldsymbol{x}}\sim p_{gen,\theta}}\left[log\left(1 - D_\phi(\tilde{\boldsymbol{x}})\right)\right]$$

$$V(\phi,\theta) \to \min_\theta \max_\phi V(\phi,\theta)$$

I. Goodfellow, et. al. *Generative adversarial nets*. In Advances in neural information processing systems, pages 2672–2680, 2014.

Game interpretation

$$V(\phi, \theta) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} \left[log D_\phi(\boldsymbol{x}) \right] + \mathbb{E}_{\tilde{\boldsymbol{x}} \sim p_{gen,\theta}} \left[log \left(1 - D_\phi(\tilde{\boldsymbol{x}}) \right) \right]$$

$$V(\phi, \theta) \rightarrow \min_\theta \max_\phi V(\phi, \theta)$$

- discriminative model (classification): discriminator
- generative model: generator

Min-max game:

- goal of discriminator: distinguish between real and generated samples
- goal of generator: ‘fool’ the discriminator

A view on GANs

- Transition from classical to deep learning: **learning the optimal feature mappings**
- Transition to adversarial loss: **learning the loss function**

$$V(\phi,\theta) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[logD_\phi(\boldsymbol{x})\right] + \mathbb{E}_{\tilde{\boldsymbol{x}}\sim p_{gen,\theta}}\left[log\left(1 - D_\phi(\tilde{\boldsymbol{x}})\right)\right]$$

$$V(\phi,\theta) \rightarrow \min_\theta \max_\phi V(\phi,\theta)$$

Generator loss is learnt by optimizing discriminator

Problems with GANs

(vanishing gradients)

Problems with GANs

(vanishing gradients)

Let's have another look at the objective:

$$V(\phi, \theta) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} \left[log D_\phi(\boldsymbol{x}) \right] + \mathbb{E}_{\tilde{\boldsymbol{x}} \sim p_{gen,\theta}} \left[log \left(1 - D_\phi(\tilde{\boldsymbol{x}}) \right) \right]$$

$$= \int \left[p_{\text{data}}(\boldsymbol{x}) log D_\phi(\boldsymbol{x}) + p_{gen,\theta}(\boldsymbol{x}) log \left(1 - D_\phi(\boldsymbol{x}) \right) \right] d\boldsymbol{x}$$

Problems with GANs

(vanishing gradients)

Let's have another look at the objective:

$$V(\phi,\theta) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[logD_\phi(\boldsymbol{x})\right] + \mathbb{E}_{\tilde{\boldsymbol{x}}\sim p_{gen,\theta}}\left[log\left(1 - D_\phi(\tilde{\boldsymbol{x}})\right)\right]$$

$$= \int \left[p_{\text{data}}(\boldsymbol{x})logD_\phi(\boldsymbol{x}) + p_{gen,\theta}(\boldsymbol{x})log\left(1 - D_\phi(\boldsymbol{x})\right)\right] d\boldsymbol{x}$$

For a given $p_{gen,\theta}$ the optimal discriminator is $D_{\phi^*}(\boldsymbol{x}) = \dfrac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_{gen,\theta}(\boldsymbol{x})}$

Problems with GANs

(vanishing gradients)

Let's have another look at the objective:

$$V(\phi,\theta) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[logD_\phi(\boldsymbol{x})\right] + \mathbb{E}_{\tilde{\boldsymbol{x}}\sim p_{gen,\theta}}\left[log\left(1-D_\phi(\tilde{\boldsymbol{x}})\right)\right]$$

$$= \int \left[p_{\text{data}}(\boldsymbol{x})logD_\phi(\boldsymbol{x}) + p_{gen,\theta}(\boldsymbol{x})log\left(1-D_\phi(\boldsymbol{x})\right)\right]d\boldsymbol{x}$$

For a given $p_{gen,\theta}$ the optimal discriminator is $D_{\phi^*}(\boldsymbol{x}) = \dfrac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_{gen,\theta}(\boldsymbol{x})}$

So the objective is: $V(\phi^*,\theta) = \int \left[p_{\text{data}}(\boldsymbol{x})log\dfrac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_{gen,\theta}(\boldsymbol{x})} + p_{gen,\theta}(\boldsymbol{x})log\dfrac{p_{gen,\theta}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_{gen,\theta}(\boldsymbol{x})}\right]d\boldsymbol{x}$

Problems with GANs

(vanishing gradients)

Let's have another look at the objective:

$$V(\phi,\theta) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[logD_\phi(\boldsymbol{x})\right] + \mathbb{E}_{\tilde{\boldsymbol{x}}\sim p_{gen,\theta}}\left[log\left(1-D_\phi(\tilde{\boldsymbol{x}})\right)\right]$$

$$= \int \left[p_{\text{data}}(\boldsymbol{x})logD_\phi(\boldsymbol{x}) + p_{gen,\theta}(\boldsymbol{x})log\left(1-D_\phi(\boldsymbol{x})\right)\right]d\boldsymbol{x}$$

For a given $p_{gen,\theta}$ the optimal discriminator is $D_{\phi^*}(\boldsymbol{x}) = \dfrac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_{gen,\theta}(\boldsymbol{x})}$

So the objective is: $V(\phi^*,\theta) = \int \left[p_{\text{data}}(\boldsymbol{x})log\dfrac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_{gen,\theta}(\boldsymbol{x})} + p_{gen,\theta}(\boldsymbol{x})log\dfrac{p_{gen,\theta}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_{gen,\theta}(\boldsymbol{x})}\right]d\boldsymbol{x}$

In case p_{data} and $p_{gen,\theta}$ have non-overlapping support

Problems with GANs

(vanishing gradients)

Let's have another look at the objective:

$$V(\phi,\theta) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[logD_\phi(\boldsymbol{x})\right] + \mathbb{E}_{\tilde{\boldsymbol{x}}\sim p_{gen,\theta}}\left[log\left(1 - D_\phi(\tilde{\boldsymbol{x}})\right)\right]$$

$$= \int \left[p_{\text{data}}(\boldsymbol{x})logD_\phi(\boldsymbol{x}) + p_{gen,\theta}(\boldsymbol{x})log\left(1 - D_\phi(\boldsymbol{x})\right)\right] d\boldsymbol{x}$$

For a given $p_{gen,\theta}$ the optimal discriminator is $D_{\phi^*}(\boldsymbol{x}) = \dfrac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_{gen,\theta}(\boldsymbol{x})}$

So the objective is: $V(\phi^*,\theta) = \int \Big[p_{\text{data}}(\boldsymbol{x})log\dfrac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_{gen,\theta}(\boldsymbol{x})} + p_{gen,\theta}(\boldsymbol{x})log\dfrac{p_{gen,\theta}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_{gen,\theta}(\boldsymbol{x})} \Big] d\boldsymbol{x}$

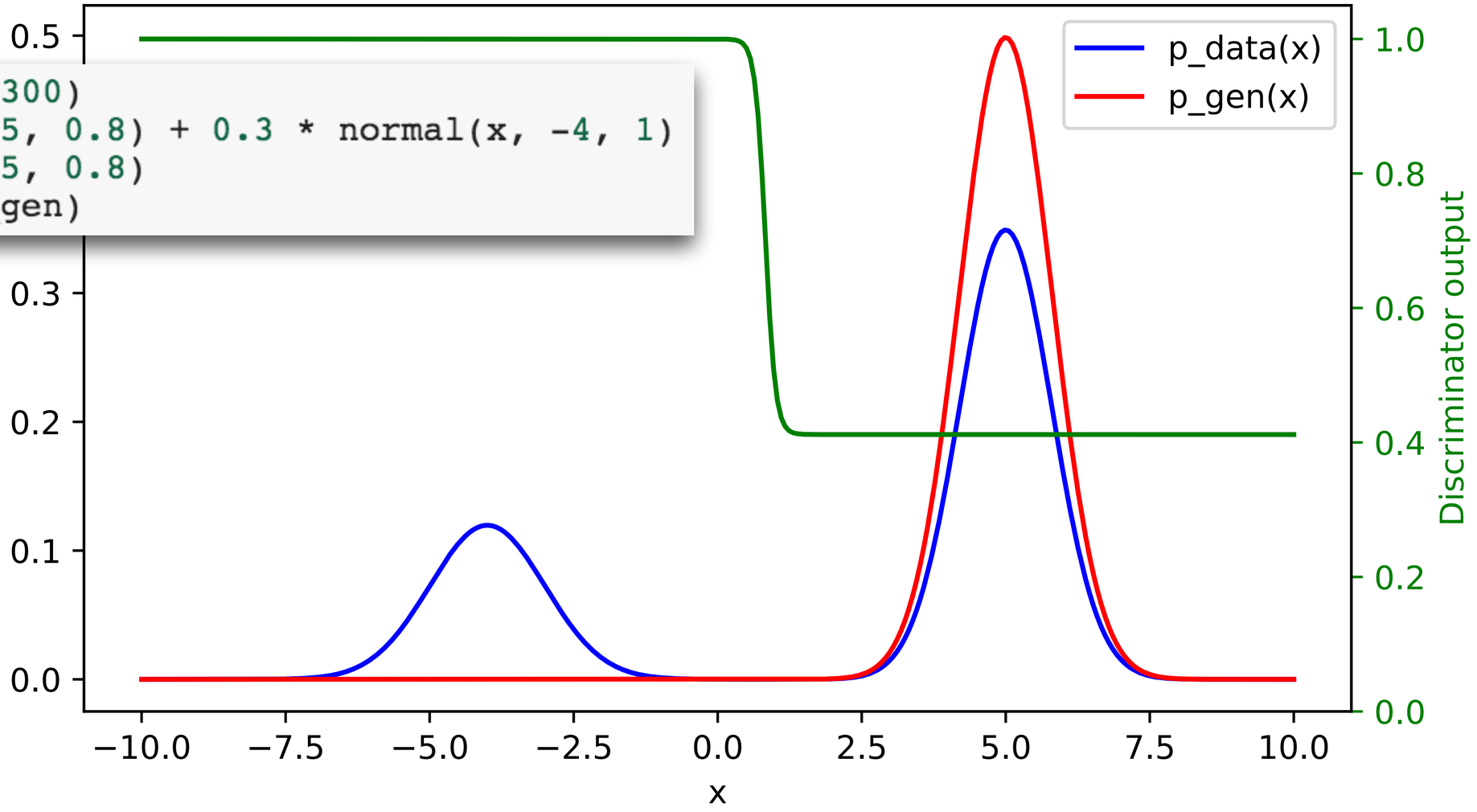
= 0 **(= const)**

In case p_{data} and $p_{gen,\theta}$ have non-overlapping support

Problems with GANs

(mode collapse)

```
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)
```

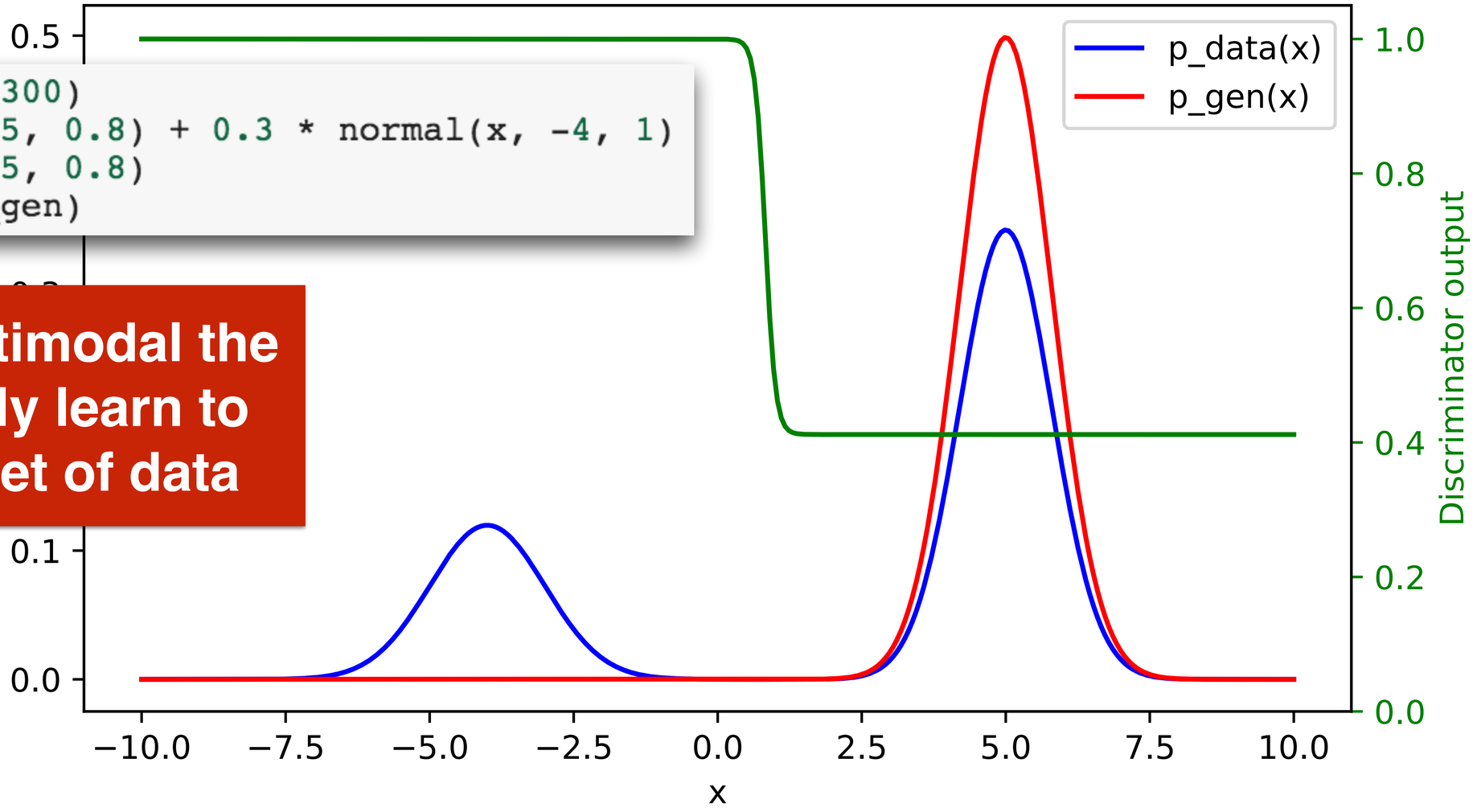


Problems with GANs

(mode collapse)

```
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)
```

In case data is multimodal the generator may only learn to reproduce a subset of data

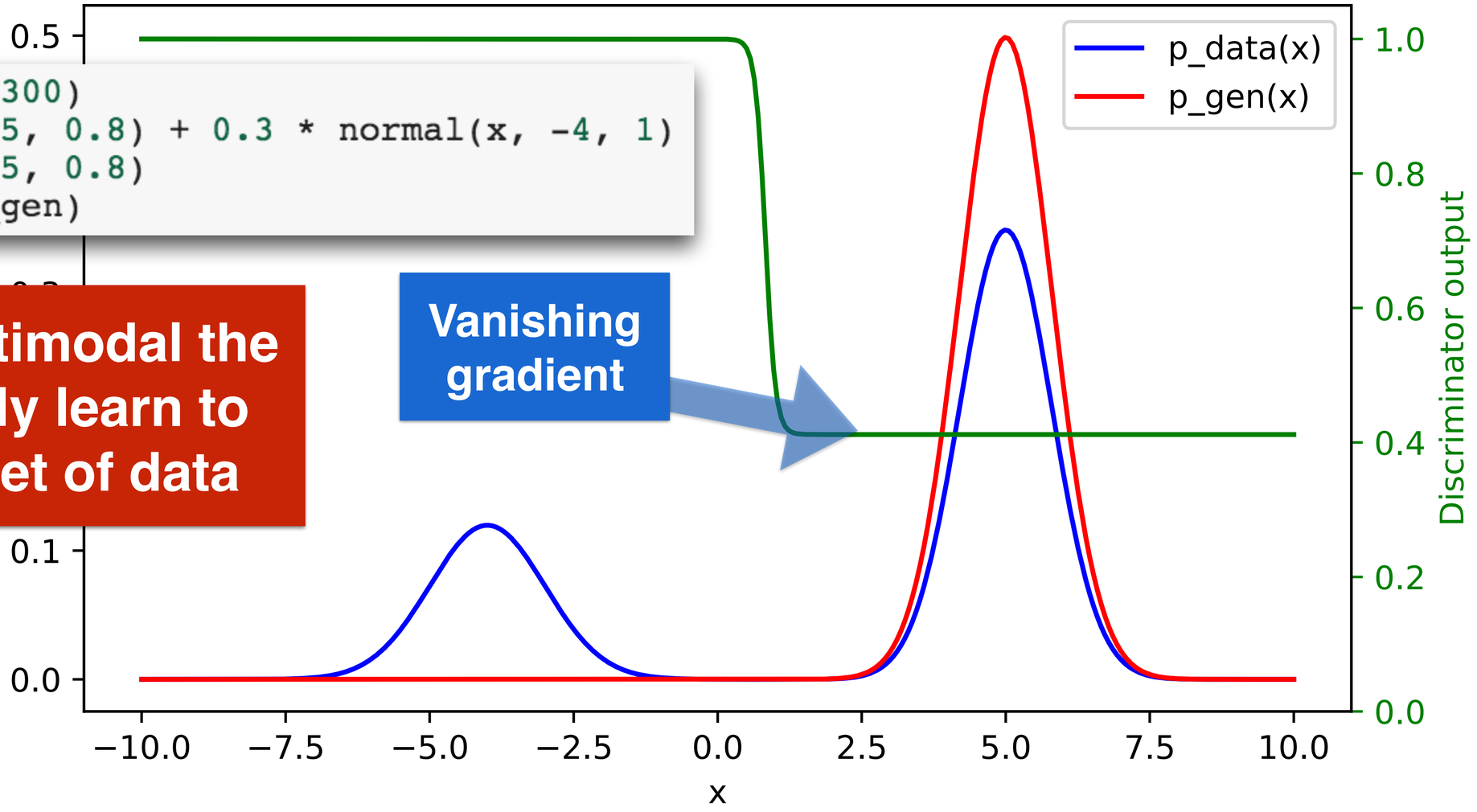


Problems with GANs

(mode collapse)

```
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)
```

In case data is multimodal the generator may only learn to reproduce a subset of data



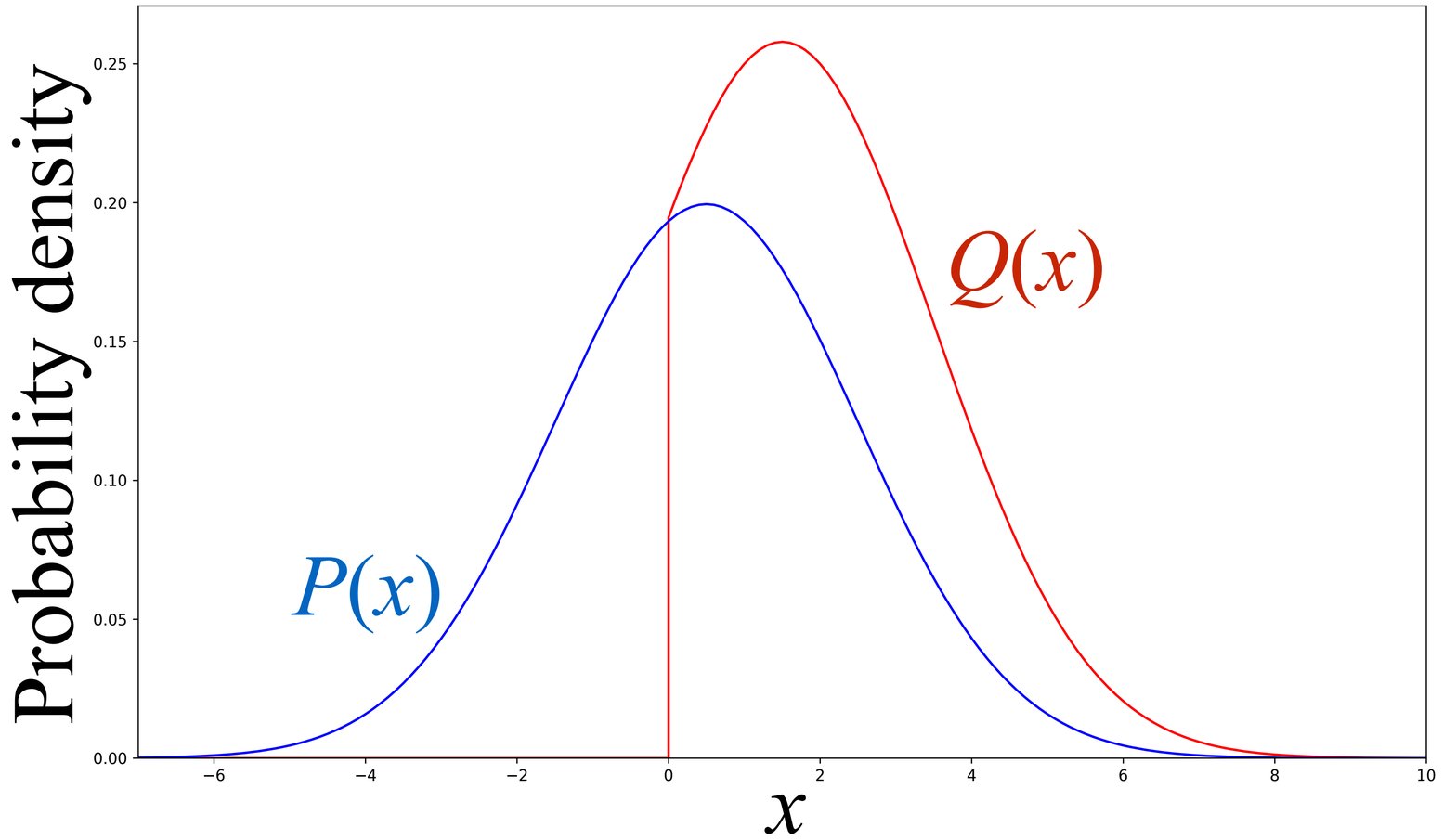
Wasserstein distance

Also called «Earth mover's distance» (EMD)

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

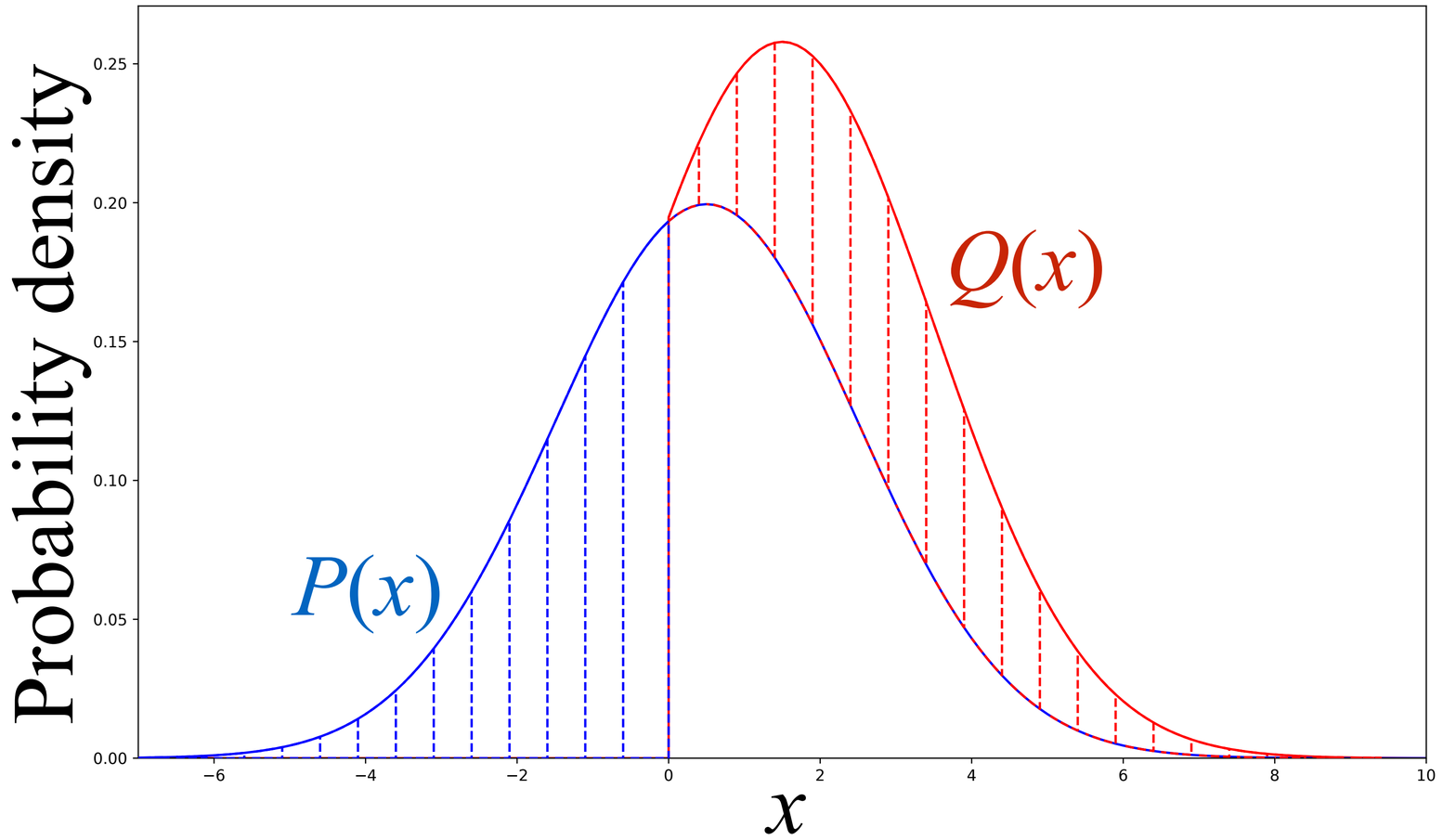


Wasserstein distance

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

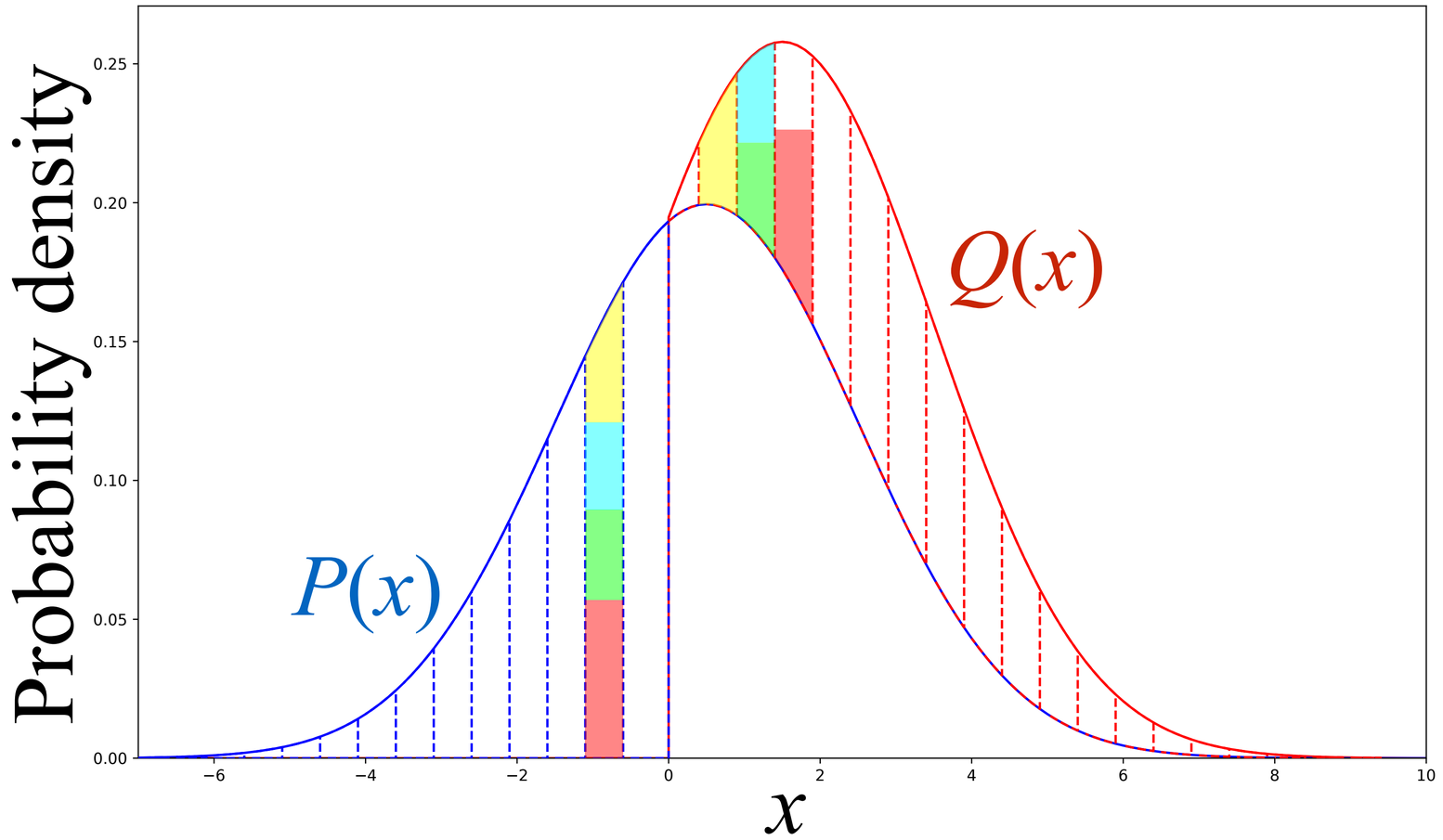


Wasserstein distance

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



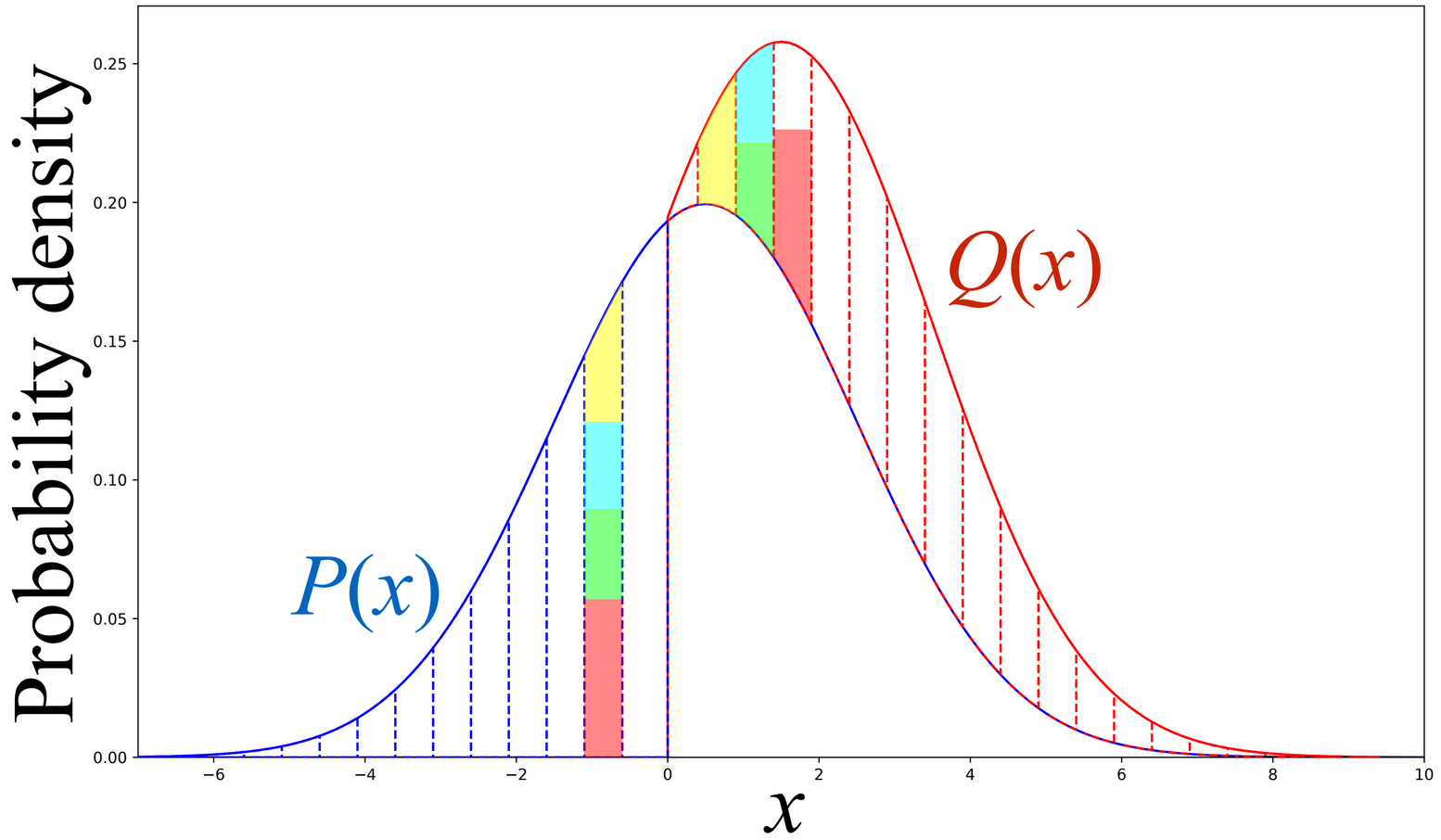
Wasserstein distance

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

The cost of moving an amount m from x_1 to x_2 is $m \times \|x_2 - x_1\|$



Wasserstein distance

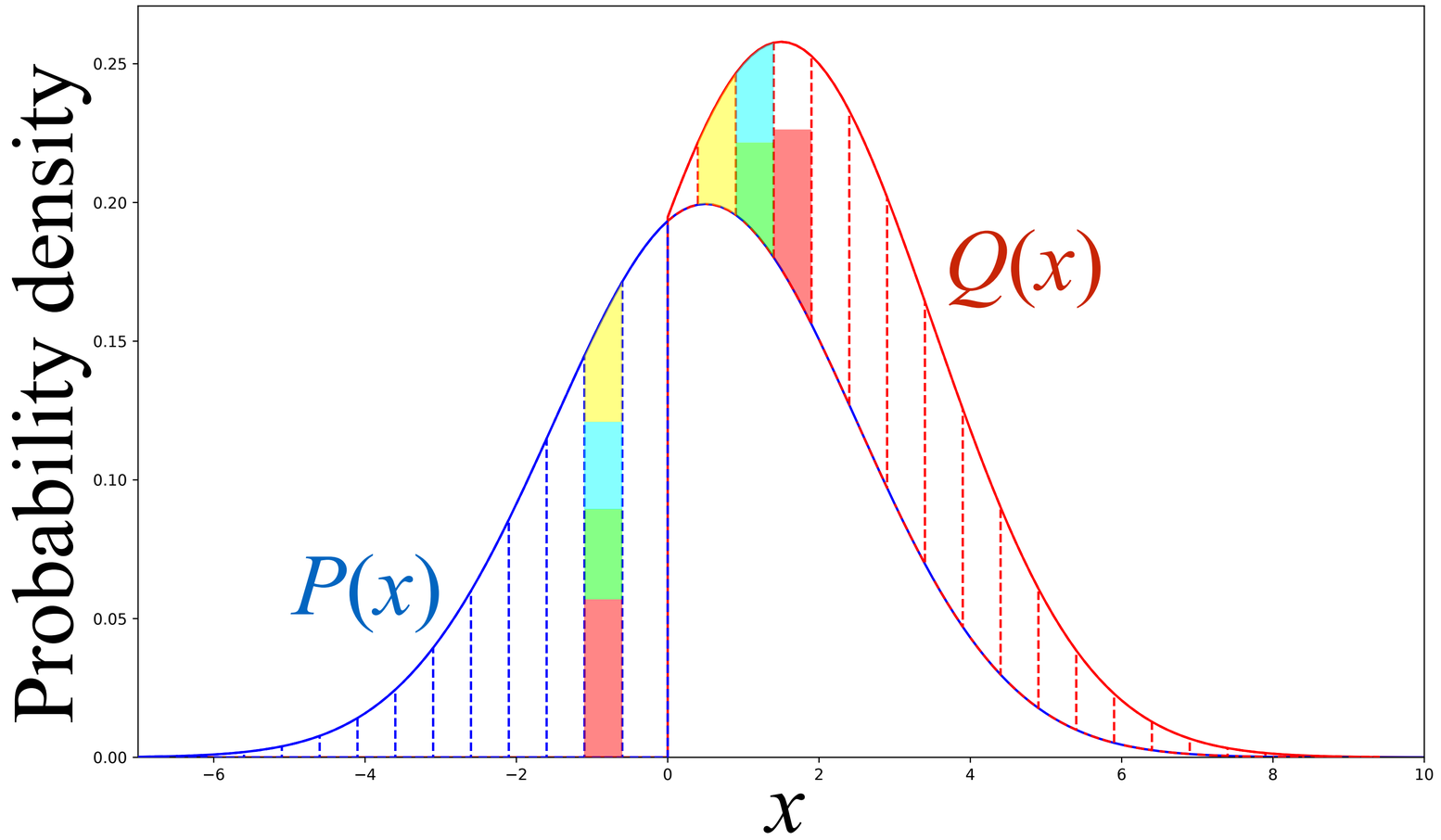
Also called «Earth mover's distance» (EMD)

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 $m \times \|x_2 - x_1\|$

EMD(P, Q) = minimum total cost of converting P into Q



Wasserstein distance

How to define it formally?

Wasserstein distance

How to define it formally?

Say, we have a plan of how much earth to move from $\boldsymbol{x_1}$ to $\boldsymbol{x_2}$: $\gamma(x_1, x_2)$

Wasserstein distance

How to define it formally?

Say, we have a plan of how much earth to move from $\boldsymbol{x_1}$ to $\boldsymbol{x_2}$: $\gamma(x_1, x_2)$

Since we want to get from P to Q, our plan has to satisfy these conditions:

$$\int \gamma(x_1, x_2) dx_1 = Q(x_2), \qquad \int \gamma(x_1, x_2) dx_2 = P(x_1)$$

Wasserstein distance

How to define it formally?

Say, we have a plan of how much earth to move from $\boldsymbol{x_1}$ to $\boldsymbol{x_2}$: $\gamma(x_1, x_2)$

Since we want to get from P to Q, our plan has to satisfy these conditions:

$$\int \gamma(x_1, x_2) dx_1 = Q(x_2), \qquad \int \gamma(x_1, x_2) dx_2 = P(x_1)$$

The cost for some particular values $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ is: $dC = ||x_1 - x_2|| \gamma(x_1, x_2) dx_1 dx_2$

distance **amount**

Wasserstein distance

How to define it formally?

Say, we have a plan of how much earth to move from $\boldsymbol{x_1}$ to $\boldsymbol{x_2}$: $\gamma(x_1, x_2)$

Since we want to get from P to Q, our plan has to satisfy these conditions:

$$\int \gamma(x_1, x_2) dx_1 = Q(x_2), \qquad \int \gamma(x_1, x_2) dx_2 = P(x_1)$$

The cost for some particular values $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ is: $dC = ||x_1 - x_2|| \gamma(x_1, x_2) dx_1 dx_2$

distance **amount**

Then, total cost can simply be written as:

$$C = \iint \gamma(x_1, x_2) ||x_1 - x_2|| dx_1 dx_2$$

Wasserstein distance

How to define it formally?

Say, we have a plan of how much earth to move from $\boldsymbol{x_1}$ to $\boldsymbol{x_2}$: $\gamma(x_1, x_2)$

Since we want to get from P to Q, our plan has to satisfy these conditions:

$$\int \gamma(x_1, x_2) dx_1 = Q(x_2), \qquad \int \gamma(x_1, x_2) dx_2 = P(x_1)$$

The cost for some particular values $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ is: $dC = ||x_1 - x_2|| \gamma(x_1, x_2) dx_1 dx_2$

distance **amount**

Then, total cost can simply be written as:

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

Wasserstein distance

How to define it formally?

Wasserstein distance

How to define it formally?

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

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

Wasserstein distance

How to define it formally?

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

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

Then, the Wasserstein distance is defined as:

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

WGAN
(Wasserstein GAN)

Imagine a GAN trying to minimize EMD between the real and generated distributions. You may be wondering:

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

WGAN
(Wasserstein GAN)

Imagine a GAN trying to minimize EMD between the real and generated distributions. You may be wondering:

How would that be better than a regular GAN?

And more importantly…

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

WGAN
(Wasserstein GAN)

Imagine a GAN trying to minimize EMD between the real and generated distributions. You may be wondering:

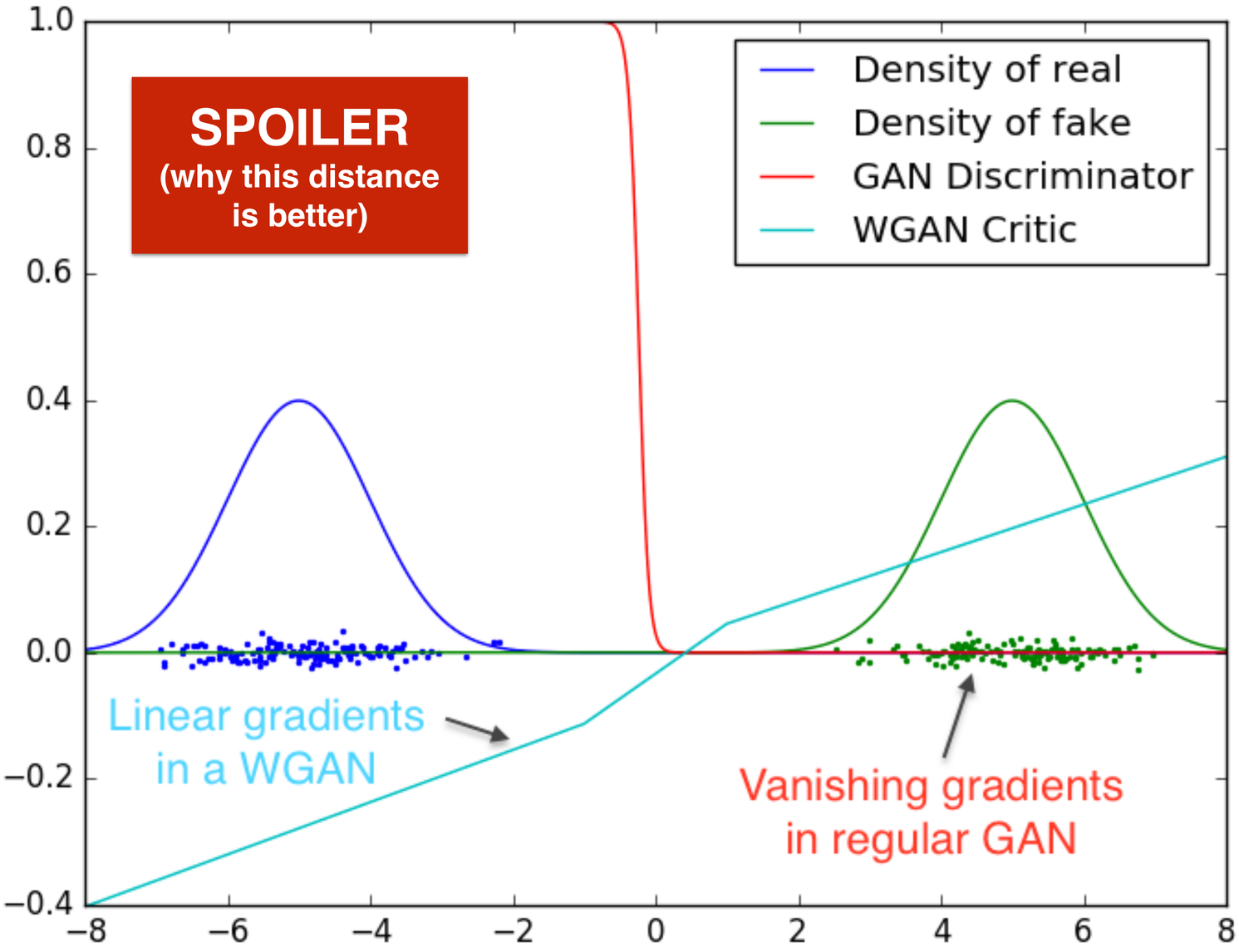
How would that be better than a regular GAN?

And more importantly…

How does one calculate this madness?!

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

WGAN



Looking ahead + answering the first question

https://arxiv.org/abs/1701.07875

Kantorovich-Rubinstein duality

(or how to calculate EMD)

disclaimer: not a strict mathematical derivation

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

Kantorovich-Rubinstein duality

(or how to calculate EMD)

disclaimer: not a strict mathematical derivation

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

Let's add the following term to this expression:

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

$f(x)$ — real-valued function

Kantorovich-Rubinstein duality

(or how to calculate EMD)

disclaimer: not a strict mathematical derivation

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

Let's add the following term to this expression:

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

$f(x)$ — real-valued function

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

Kantorovich-Rubinstein duality

(or how to calculate EMD)

disclaimer: not a strict mathematical derivation

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

Let's add the following term to this expression:

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

$f(x)$ — 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:

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

Kantorovich-Rubinstein duality

(or how to calculate EMD)

disclaimer: not a strict mathematical derivation

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

Let's add the following term to this expression:

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

$f(x)$ — 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:

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

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

Kantorovich-Rubinstein duality

(or how to calculate EMD)

disclaimer: not a strict mathematical derivation

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

Kantorovich-Rubinstein duality

(or how to calculate EMD)

disclaimer: not a strict mathematical derivation

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

Consider the following case: $|f(a) - f(b)| \leq ||a - b||, \quad \forall a,\ b$

We'll denote it as: $||f||_L \leq 1$

Kantorovich-Rubinstein duality

(or how to calculate EMD)

disclaimer: not a strict mathematical derivation

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

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 -∞

Kantorovich-Rubinstein duality

(or how to calculate EMD)

disclaimer: not a strict mathematical derivation

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

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 -∞

Therefore we can finally rewrite the whole thing as:

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

Kantorovich-Rubinstein duality

(or how to calculate EMD)

disclaimer: not a strict mathematical derivation

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

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 -∞

Therefore we can finally rewrite the whole thing as:

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

How is this simpler?

WGAN

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

https://arxiv.org/abs/1701.07875

WGAN

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

Expectations can be estimated as sample mean (per batch)

https://arxiv.org/abs/1701.07875

WGAN

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

The function can be expressed with a neural net – discriminator (‘critic’ in the original paper)

Expectations can be estimated as sample mean (per batch)

https://arxiv.org/abs/1701.07875

WGAN

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

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

Expectations can be estimated as sample mean (per batch)

This property can be replaced by a weaker one: $||f||_L \leq k: \ |f(a) - f(b)| \leq k \cdot ||a - b||, \quad \forall a, \ b$
In such case we'll estimate $k \times \mathrm{EMD}(P, Q)$ instead of $\mathrm{EMD}(P, Q)$

https://arxiv.org/abs/1701.07875

WGAN

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

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

Expectations can be estimated as sample mean (per batch)

This property can be replaced by a weaker one: $||f||_L \leq k: \ |f(a) - f(b)| \leq k \cdot ||a - b||, \quad \forall a, \ b$
In such case we'll estimate $k \times \mathrm{EMD}(P, Q)$ instead of $\mathrm{EMD}(P, Q)$

This can be achieved by clipping critic's weights at some value: $w \rightarrow \mathbf{clip}(w, -c, c)$

https://arxiv.org/abs/1701.07875

WGAN

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

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

Expectations can be estimated as sample mean (per batch)

This property can be replaced by a weaker one: $||f||_L \leq k: \ |f(a) - f(b)| \leq k \cdot ||a - b||, \quad \forall a, \ b$
In such case we'll estimate $k \times \mathrm{EMD}(P, Q)$ instead of $\mathrm{EMD}(P, Q)$

This can be achieved by clipping critic's weights at some value: $w \rightarrow \mathbf{clip}(w, -c, c)$

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

https://arxiv.org/abs/1701.07875

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.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

https://arxiv.org/abs/1701.07875

WGAN-GP

Improved Training of Wasserstein GANs
https://arxiv.org/abs/1704.00028

WGAN-GP

Improved Training of Wasserstein GANs
https://arxiv.org/abs/1704.00028

Weight clipping makes critic less expressive and training process harder to converge

WGAN-GP

Improved Training of Wasserstein GANs
https://arxiv.org/abs/1704.00028

Weight clipping makes critic less expressive and training process harder to converge

Optimal f should satisfy $||\nabla f|| = 1$ almost everywhere under P and Q

WGAN-GP

Improved Training of Wasserstein GANs
https://arxiv.org/abs/1704.00028

Weight clipping makes critic less expressive and training process harder to converge

Optimal f should satisfy $||\nabla f|| = 1$ almost everywhere under P and Q

$$+$$

$$||f||_L \leq 1 \iff ||\nabla f|| \leq 1$$

WGAN-GP

Improved Training of Wasserstein GANs
https://arxiv.org/abs/1704.00028

Weight clipping makes critic less expressive and training process harder to converge

Optimal f should satisfy $\|\nabla f\| = 1$ almost everywhere under P and Q

$$+$$

$$\|f\|_L \leq 1 \iff \|\nabla f\| \leq 1$$

Weight clipping can be replaced by penalizing the gradients to be of unit magnitude

WGAN-GP

«Gradient penalty»

Improved Training of Wasserstein GANs
https://arxiv.org/abs/1704.00028

Weight clipping makes critic less expressive and training process harder to converge

Optimal f should satisfy $\|\nabla f\| = 1$ almost everywhere under P and Q

$$+$$

$$\|f\|_L \leq 1 \iff \|\nabla f\| \leq 1$$

Weight clipping can be replaced by penalizing the gradients to be of unit magnitude

WGAN-GP

«Gradient penalty»

Improved Training of Wasserstein GANs
https://arxiv.org/abs/1704.00028

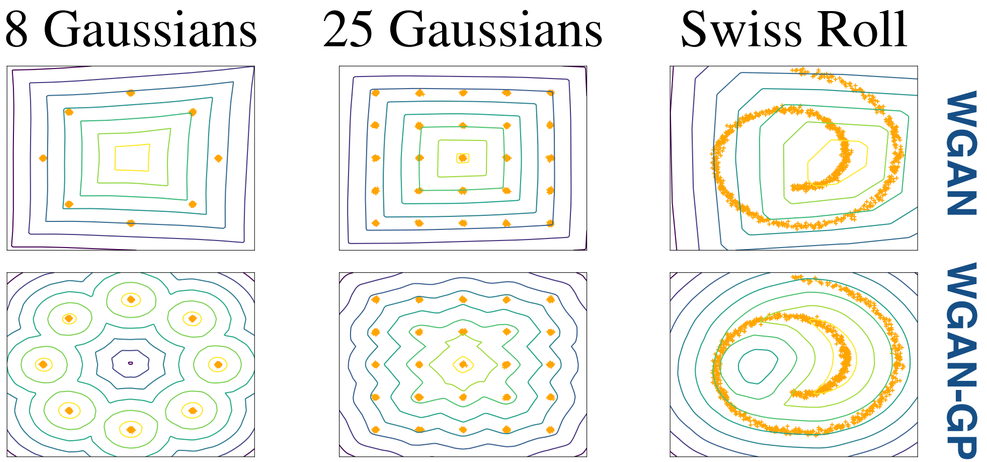
Weight clipping makes critic less expressive and training process harder to converge

Optimal f should satisfy $\|\nabla f\| = 1$ almost everywhere under P and Q

$$+$$

$$\|f\|_L \leq 1 \iff \|\nabla f\| \leq 1$$

Weight clipping can be replaced by penalizing the gradients to be of unit magnitude



Value surfaces of f

WGAN-GP

«Gradient penalty»

Improved Training of Wasserstein GANs
https://arxiv.org/abs/1704.00028

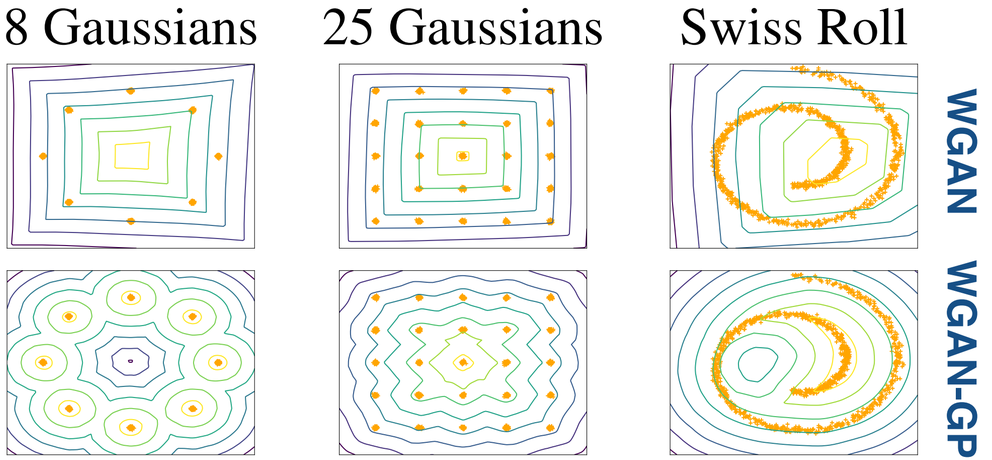
Weight clipping makes critic less expressive and training process harder to converge

Optimal f should satisfy $\|\nabla f\| = 1$ almost everywhere under P and Q

$$+$$

$$\|f\|_L \leq 1 \iff \|\nabla f\| \leq 1$$

Weight clipping can be replaced by penalizing the gradients to be of unit magnitude



Value surfaces of f

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

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

WGAN-GP

«Gradient penalty»

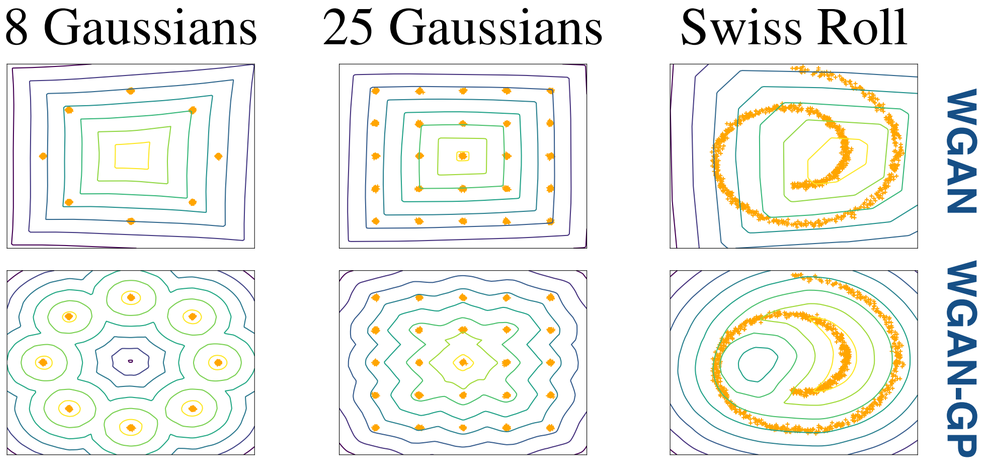
Improved Training of Wasserstein GANs
https://arxiv.org/abs/1704.00028

Weight clipping makes critic less expressive and training process harder to converge

Optimal f should satisfy $||\nabla f|| = 1$ almost everywhere under P and Q

$$+$$

$$||f||_L \leq 1 \iff ||\nabla f|| \leq 1$$



Value surfaces of f

W
pe
m

Alternative 'one-sided' penalty:

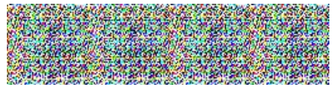
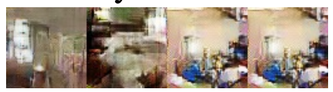
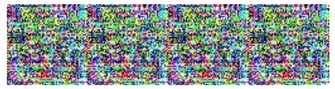
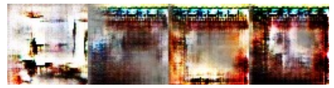
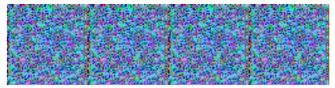
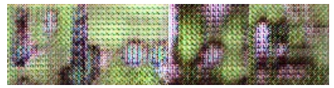
$$\text{GP} = \lambda \mathbb{E}_{\tilde{x} \sim \mathbb{P}_{\tilde{x}}} \left[\max(0, ||\nabla_{\tilde{x}} f(\tilde{x})|| - 1)^2 \right]$$

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

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

WGAN-GP

Improved Training of Wasserstein GANs
https://arxiv.org/abs/1704.00028

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			
			

WGAN-GP

Improved Training of Wasserstein GANs
https://arxiv.org/abs/1704.00028

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

Biased Wasserstein gradients

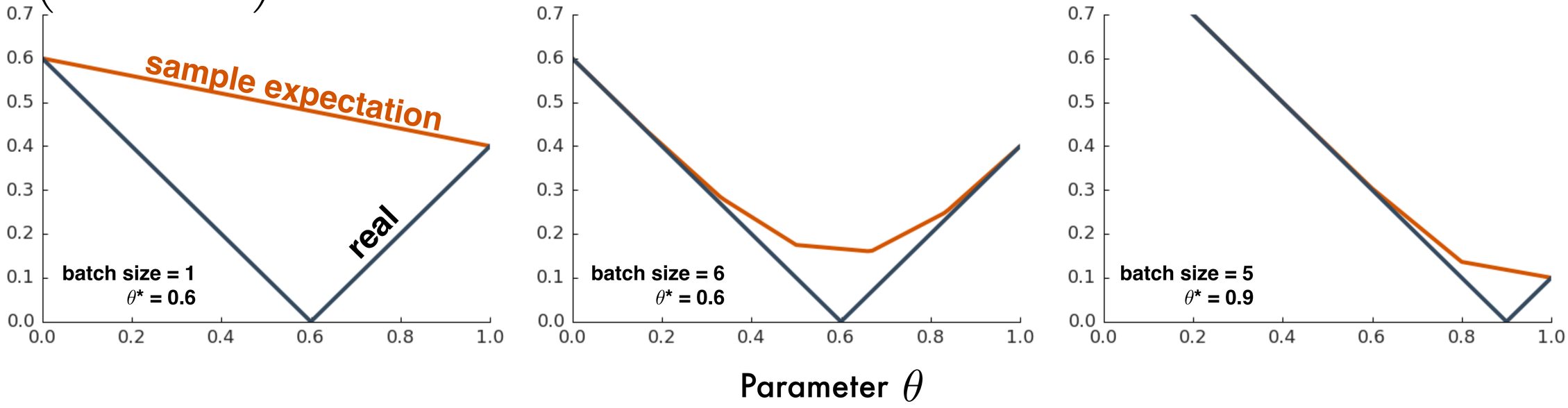
Authors of https://arxiv.org/abs/1705.10743 showed for EMD that expected sample gradients may differ from true gradients

This may lead to a wrong minimum:

$$\mathrm{EMD}\left(\mathcal{B}(\theta^*), \mathcal{B}(\theta)\right)$$

Bernoulli distribution

$$x \sim \mathcal{B}(\theta): \begin{cases} \mathbb{P}(x=0) = 1-\theta \\ \mathbb{P}(x=1) = \theta \end{cases}$$



Cramer and energy distances

Cramer distance:

$$l_2^2(P, Q) := \int_{-\infty}^{\infty} (F_P(x) - F_Q(x))^2 \mathrm{d}x$$

CDFs for P and Q

Cramer and energy distances

Cramer distance:

$$l_2^2(P, Q) := \int_{-\infty}^{\infty} (F_P(x) - F_Q(x))^2 \mathrm{d}x$$

CDFs for P and Q

Energy distance — a natural extension of the Cramer distance to the multivariate case:

$$\mathcal{E}(P, Q) = 2\mathbb{E}_{\substack{x_1 \sim P \\ x_2 \sim Q}} ||x_1 - x_2|| - \mathbb{E}_{x_1, x_1' \sim P} ||x_1 - x_1'|| - \mathbb{E}_{x_2, x_2' \sim Q} ||x_2 - x_2'||$$

Cramer and energy distances

Cramer distance:

$$l_2^2(P, Q) := \int_{-\infty}^{\infty} (F_P(x) - F_Q(x))^2 \mathrm{d}x$$

CDFs for P and Q

Energy distance — a natural extension of the Cramer distance to the multivariate case:

$$\mathcal{E}(P, Q) = 2\mathbb{E}_{\substack{x_1 \sim P \\ x_2 \sim Q}} ||x_1 - x_2|| - \mathbb{E}_{x_1, x_1' \sim P} ||x_1 - x_1'|| - \mathbb{E}_{x_2, x_2' \sim Q} ||x_2 - x_2'||$$

– preserves nice properties of EMD

Cramer and energy distances

Cramer distance:

$$l_2^2(P, Q) := \int_{-\infty}^{\infty} (F_P(x) - F_Q(x))^2 \mathrm{d}x$$

CDFs for P and Q

Energy distance — a natural extension of the Cramer distance to the multivariate case:

$$\mathcal{E}(P, Q) = 2\mathbb{E}_{\substack{x_1 \sim P \\ x_2 \sim Q}} ||x_1 - x_2|| - \mathbb{E}_{x_1, x_1' \sim P} ||x_1 - x_1'|| - \mathbb{E}_{x_2, x_2' \sim Q} ||x_2 - x_2'||$$

– preserves nice properties of EMD

– is proven to have unbiased sample gradients

Cramer and energy distances

Cramer distance: $$l_2^2(P, Q) := \int_{-\infty}^{\infty} (F_P(x) - F_Q(x))^2 \mathrm{d}x$$

CDFs for P and Q

Energy distance — a natural extension of the Cramer distance to the multivariate case:

$$\mathcal{E}(P, Q) = 2\mathbb{E}_{\substack{x_1 \sim P \\ x_2 \sim Q}} ||x_1 - x_2|| - \mathbb{E}_{x_1, x_1' \sim P} ||x_1 - x_1'|| - \mathbb{E}_{x_2, x_2' \sim Q} ||x_2 - x_2'||$$

– preserves nice properties of EMD

– is proven to have unbiased sample gradients

In one-dimensional case: $$l_2^2(P, Q) = \frac{1}{2}\mathcal{E}(P, Q)$$

Cramer GAN

$$\mathcal{E}(P, Q) = 2\mathbb{E}_{\substack{x_1 \sim P \\ x_2 \sim Q}}||x_1 - x_2|| - \mathbb{E}_{x_1, x_1' \sim P}||x_1 - x_1'|| - \mathbb{E}_{x_2, x_2' \sim Q}||x_2 - x_2'||$$

https://arxiv.org/abs/1705.10743

Cramer GAN

$$\mathcal{E}(P, Q) = 2\mathbb{E}_{\substack{x_1 \sim P \\ x_2 \sim Q}}||x_1 - x_2|| - \mathbb{E}_{x_1, x_1' \sim P}||x_1 - x_1'|| - \mathbb{E}_{x_2, x_2' \sim Q}||x_2 - x_2'||$$

Actually, this can already be used to train just the generator (regression task)

https://arxiv.org/abs/1705.10743

Cramer GAN

$$\mathcal{E}(P,Q) = 2\mathbb{E}_{\substack{x_1 \sim P \\ x_2 \sim Q}}||x_1 - x_2|| - \mathbb{E}_{x_1, x_1' \sim P}||x_1 - x_1'|| - \mathbb{E}_{x_2, x_2' \sim Q}||x_2 - x_2'||$$

Actually, this can already be used to train just the generator (regression task)

However, much better quality can be achieved with a critic transforming P and Q into P^* and Q^*:

$$\mathcal{L}_{generator} = \mathcal{E}(P^*, Q^*)$$
$$\mathcal{L}_{critic} = -\mathcal{E}(P^*, Q^*) + \text{regularization}$$
$$x \sim P^* : x = f(x'),\ x' \sim P$$
$$x \sim Q^* : x = f(x'),\ x' \sim Q$$

https://arxiv.org/abs/1705.10743

Cramer GAN

$$\mathcal{E}(P,Q) = 2\mathbb{E}_{\substack{x_1 \sim P \\ x_2 \sim Q}}||x_1 - x_2|| - \mathbb{E}_{x_1, x_1' \sim P}||x_1 - x_1'|| - \mathbb{E}_{x_2, x_2' \sim Q}||x_2 - x_2'||$$

Actually, this can already be used to train just the generator (regression task)

However, much better quality can be achieved with a critic transforming P and Q into P^* and Q^*:

$$\mathcal{L}_{generator} = \mathcal{E}(P^*, Q^*)$$
$$\mathcal{L}_{critic} = -\mathcal{E}(P^*, Q^*) + \text{regularization}$$
$$x \sim P^* : x = f(x'),\ x' \sim P$$
$$x \sim Q^* : x = f(x'),\ x' \sim Q$$

Regularization (e.g. GP) needed to prevent critic from scaling the data to infinity

https://arxiv.org/abs/1705.10743

Cramer GAN

$$\mathcal{E}(P,Q) = 2\mathbb{E}_{\substack{x_1 \sim P \\ x_2 \sim Q}}||x_1 - x_2|| - \mathbb{E}_{x_1, x_1' \sim P}||x_1 - x_1'|| - \mathbb{E}_{x_2, x_2' \sim Q}||x_2 - x_2'||$$

Actually, this can already be used to train just the generator (regression task)

However, much better quality can be achieved with a critic transforming P and Q into P^* and Q^*:

$$\mathcal{L}_{generator} = \mathcal{E}(P^*, Q^*)$$
$$\mathcal{L}_{critic} = -\mathcal{E}(P^*, Q^*) + \text{regularization}$$
$$x \sim P^* : x = f(x'),\ x' \sim P$$
$$x \sim Q^* : x = f(x'),\ x' \sim Q$$

Regularization (e.g. GP) needed to prevent critic from scaling the data to infinity

In the paper 'surrogate' loss ($x_1' \to 0$) is used to avoid sampling from data twice

https://arxiv.org/abs/1705.10743

Which Training Methods for GANs do actually Converge?

https://arxiv.org/abs/1801.04406 — the authors analyzed GAN training behavior on a very simple system:

$$
\begin{aligned}
\text{target distribution:} \quad & p_{\text{data}} = \delta(x) \\
\text{generator distribution:} \quad & p_{gen} = \delta(x - \theta) \\
\text{discriminator:} \quad & D_\psi(x) = \psi \cdot x
\end{aligned}
$$

Ideal solution: $\theta = \psi = 0$

Which Training Methods for GANs do actually Converge?

https://arxiv.org/abs/1801.04406 — the authors analyzed GAN training behavior on a very simple system:

$$
\begin{aligned}
&\text{target distribution:} && p_{\text{data}} &&= \delta(x) \\
&\text{generator distribution:} && p_{gen} &&= \delta(x-\theta) \\
&\text{discriminator:} && D_\psi(x) &&= \psi \cdot x
\end{aligned}
$$

Single-parameter generator

Ideal solution: $\theta = \psi = 0$

Which Training Methods for GANs do actually Converge?

https://arxiv.org/abs/1801.04406 — the authors analyzed GAN training behavior on a very simple system:

$$
\begin{aligned}
&\text{target distribution:} && p_{\text{data}} &&= \delta(x) \\
&\text{generator distribution:} && p_{gen} &&= \delta(x-\theta) \\
&\text{discriminator:} && D_\psi(x) &&= \psi \cdot x
\end{aligned}
$$

Single-parameter generator

Single-parameter discriminator

Ideal solution: $\theta = \psi = 0$

Which Training Methods for GANs do actually Converge?

https://arxiv.org/abs/1801.04406 — the authors analyzed GAN training behavior on a very simple system:

$$\begin{aligned} \text{target distribution:} \quad & p_{\text{data}} = \delta(x) \\ \text{generator distribution:} \quad & p_{gen} = \delta(x-\theta) \\ \text{discriminator:} \quad & D_\psi(x) = \psi \cdot x \end{aligned}$$

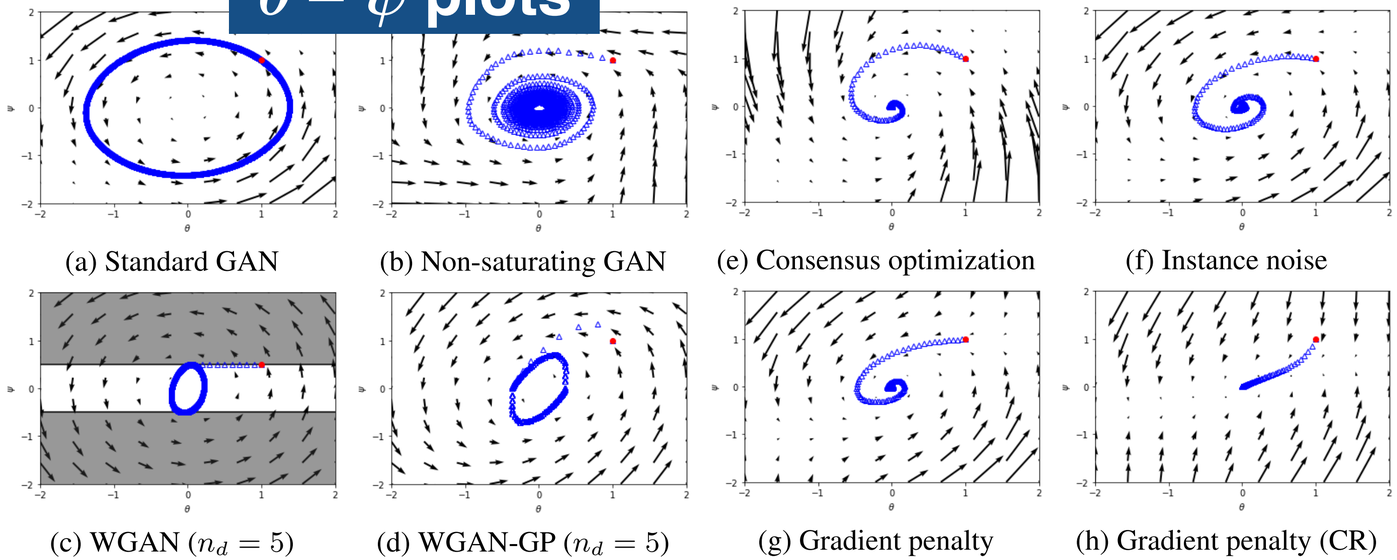
Single-parameter generator

Single-parameter discriminator

Ideal solution: $\theta = \psi = 0$

Examined various training schemes (with fixed learning rate):

θ – ψ plots



(a) Standard GAN (b) Non-saturating GAN (e) Consensus optimization (f) Instance noise

(c) WGAN ($n_d = 5$) (d) WGAN-GP ($n_d = 5$) (g) Gradient penalty (h) Gradient penalty (CR)

Which Training Methods for GANs do actually Converge?

https://arxiv.org/abs/1801.04406 — the authors analyzed GAN training behavior on a very simple system:

$$\begin{aligned} &\text{target distribution:} && p_{\text{data}} &&= && \delta(x) \\ &\text{generator distribution:} && p_{gen} &&= && \delta(x-\theta) \\ &\text{discriminator:} && D_\psi(x) &&= && \psi \cdot x \end{aligned}$$

Single-parameter generator

Single-parameter discriminator

Ideal solution: $\theta = \psi = 0$

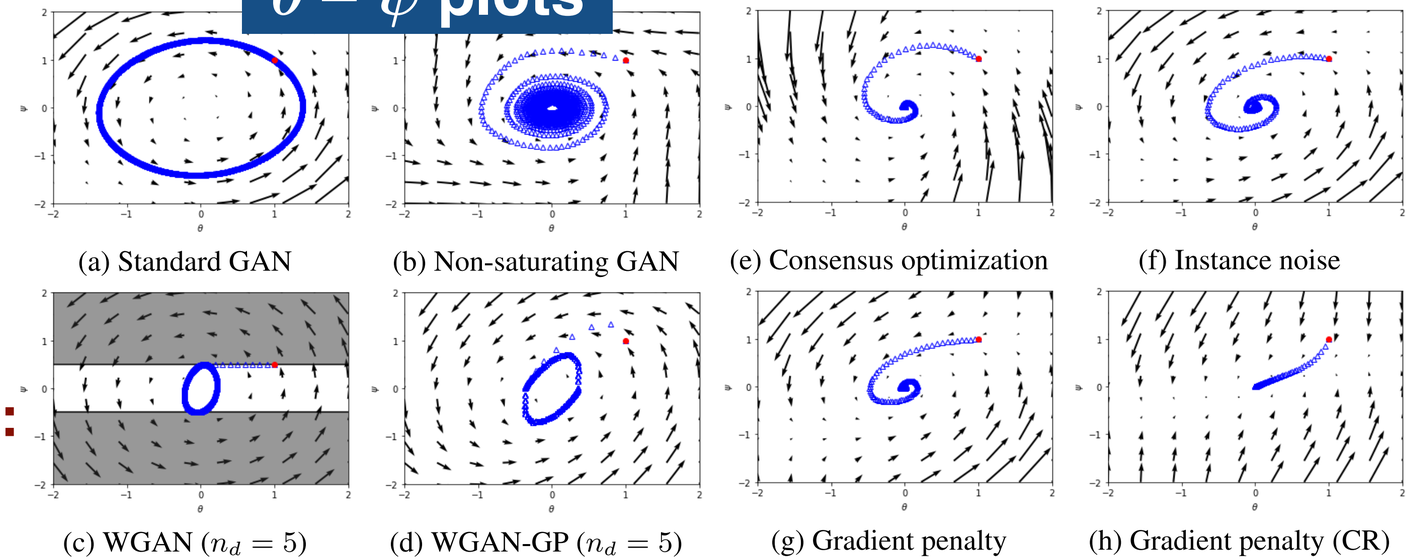
Examined various training schemes (with fixed learning rate):

Proposed theoretically motivated regularization schemes:

- **instance noise (adding noise to any discriminator input)**
- **zero-centered gradient penalty at $x \sim p_{\text{data}}$:**

$$\frac{\gamma}{2}\mathbb{E}_{x\sim p_{\text{data}}}||\nabla D(x)||^2$$

θ – ψ plots



(a) Standard GAN (b) Non-saturating GAN (e) Consensus optimization (f) Instance noise

(c) WGAN ($n_d = 5$) (d) WGAN-GP ($n_d = 5$) (g) Gradient penalty (h) Gradient penalty (CR)

Which Training Methods for GANs do actually Converge?

https://arxiv.org/abs/1801.04406 — the authors analyzed GAN training behavior on a very simple system:

$$\begin{aligned} \text{target distribution:} \quad & p_{\text{data}} = \delta(x) \\ \text{generator distribution:} \quad & p_{gen} = \delta(x-\theta) \\ \text{discriminator:} \quad & D_\psi(x) = \psi \cdot x \end{aligned}$$

Single-parameter generator

Single-parameter discriminator

Ideal solution: $\theta = \psi = 0$

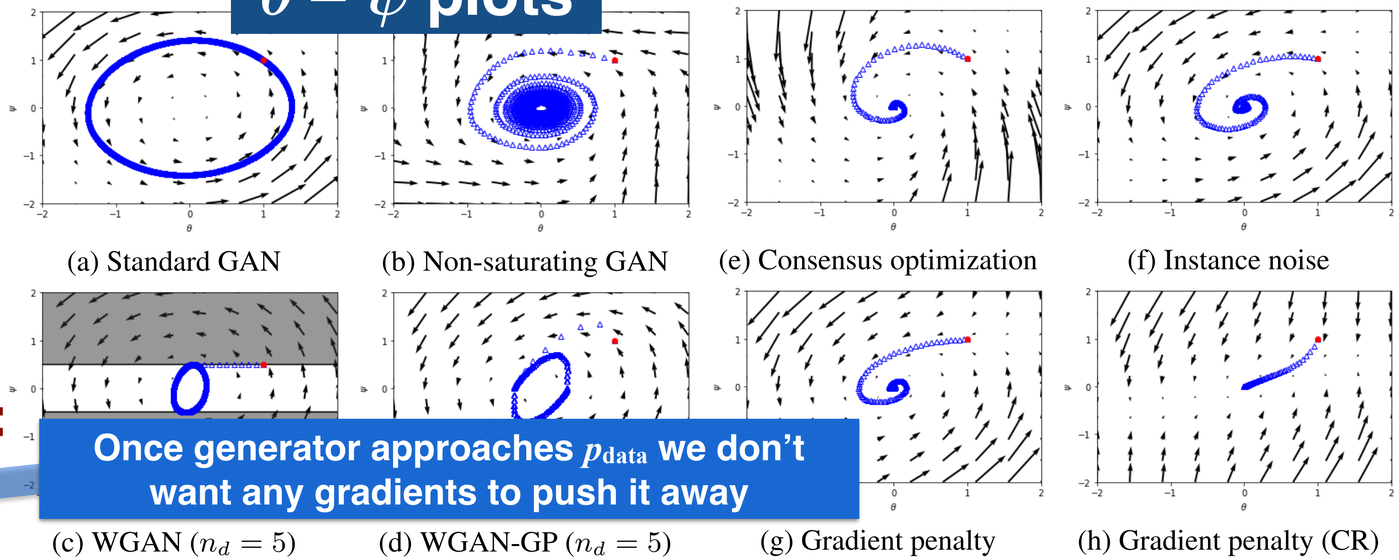
Examined various training schemes (with fixed learning rate):

Proposed theoretically motivated regularization schemes:

- **instance noise (adding noise to any discriminator input)**
- **zero-centered gradient penalty at $x \sim p_{\text{data}}$:**

$$\frac{\gamma}{2}\mathbb{E}_{x\sim p_{\text{data}}}||\nabla D(x)||^2$$

$\theta - \psi$ plots



(a) Standard GAN (b) Non-saturating GAN (e) Consensus optimization (f) Instance noise

(c) WGAN ($n_d = 5$) (d) WGAN-GP ($n_d = 5$) (g) Gradient penalty (h) Gradient penalty (CR)

Once generator approaches p_{data} we don't want any gradients to push it away

Boundary Equilibrium Generative Adversarial Networks

(BEGAN)

https://arxiv.org/abs/1703.10717

Notice for Wasserstein between 1D distributions:

Difference between means

$$\inf \mathbb{E}[|x_1 - x_2|] \geqslant \inf |\mathbb{E}[x_1 - x_2]| = |m_1 - m_2|$$

Boundary Equilibrium Generative Adversarial Networks

(BEGAN)

https://arxiv.org/abs/1703.10717

Notice for Wasserstein between 1D distributions:

Difference between means

$$\inf \mathbb{E}[|x_1 - x_2|] \geqslant \inf |\mathbb{E}[x_1 - x_2]| = |m_1 - m_2|$$

The authors take auto-encoder loss:

$$\mathcal{L}(v) = |v - D(v)|^{\eta} \text{ where } \begin{cases} D : \mathbb{R}^{N_x} \mapsto \mathbb{R}^{N_x} & \text{is the autoencoder function.} \\ \eta \in \{1, 2\} & \text{is the target norm.} \\ v \in \mathbb{R}^{N_x} & \text{is a sample of dimension } N_x. \end{cases}$$

Boundary Equilibrium Generative Adversarial Networks

(BEGAN)

https://arxiv.org/abs/1703.10717

Notice for Wasserstein between 1D distributions:

Difference between means

$$\inf \mathbb{E}[|x_1 - x_2|] \geqslant \inf |\mathbb{E}[x_1 - x_2]| = |m_1 - m_2|$$

The authors take auto-encoder loss:

$$\mathcal{L}(v) = |v - D(v)|^{\eta} \text{ where } \begin{cases} D : \mathbb{R}^{N_x} \mapsto \mathbb{R}^{N_x} & \text{is the autoencoder function.} \\ \eta \in \{1, 2\} & \text{is the target norm.} \\ v \in \mathbb{R}^{N_x} & \text{is a sample of dimension } N_x. \end{cases}$$

Auto-encoder loss for real data

Auto-encoder loss for generated data

$$\begin{cases} \mathcal{L}_D = \mathcal{L}(x; \theta_D) - \mathcal{L}(G(z_D; \theta_G); \theta_D) & \text{for } \theta_D \\ \mathcal{L}_G = -\mathcal{L}_D & \text{for } \theta_G \end{cases}$$

And treat it as 1D distribution for the Wasserstein loss

- D is both auto-encoder and discriminator

Boundary Equilibrium Generative Adversarial Networks

(BEGAN)

https://arxiv.org/abs/1703.10717

$$\begin{cases} \mathcal{L}_D = \mathcal{L}(x;\theta_D) - \mathcal{L}(G(z_D;\theta_G);\theta_D) & \text{for } \theta_D \\ \mathcal{L}_G = -\mathcal{L}_D & \text{for } \theta_G \end{cases} \quad \longleftrightarrow \quad \text{Equilibrium at: } \mathbb{E}\left[\mathcal{L}(x)\right] = \mathbb{E}\left[\mathcal{L}(G(z))\right]$$

Boundary Equilibrium Generative Adversarial Networks

(BEGAN)

https://arxiv.org/abs/1703.10717

$$\begin{cases} \mathcal{L}_D = \mathcal{L}(x;\theta_D) - \mathcal{L}(G(z_D;\theta_G);\theta_D) & \text{for } \theta_D \\ \mathcal{L}_G = -\mathcal{L}_D & \text{for } \theta_G \end{cases} \longleftrightarrow$$

Equilibrium at:

$$\mathbb{E}\left[\mathcal{L}(x)\right] = \mathbb{E}\left[\mathcal{L}(G(z))\right]$$

$$\begin{cases} \mathcal{L}_D = \mathcal{L}(x) - k_t.\mathcal{L}(G(z_D)) & \text{for } \theta_D \\ \mathcal{L}_G = \mathcal{L}(G(z_G)) & \text{for } \theta_G \\ k_{t+1} = k_t + \lambda_k(\gamma\mathcal{L}(x) - \mathcal{L}(G(z_G))) & \text{for each training step } t \end{cases} \longleftrightarrow$$

Equilibrium at:

$$\gamma = \frac{\mathbb{E}\left[\mathcal{L}(G(z))\right]}{\mathbb{E}\left[\mathcal{L}(x)\right]}$$

$\gamma \in [0, 1]$ — parameter to control the trade-off between quality (low γ) and diversity (high γ)

Boundary Equilibrium Generative Adversarial Networks

(BEGAN)

https://arxiv.org/abs/1703.10717

Figure 3: Random 64x64 samples at varying $\gamma \in \{0.3, 0.5, 0.7\}$

Conditional distributions

Sometimes it's necessary to learn not just $P(x)$, but $P(x|a)$

Conditional distributions

~~Sometimes~~ it's necessary to learn not just $P(x)$, but $P(x|a)$

Most of the times

Conditional distributions

~~Sometimes~~ it's necessary to learn not just $P(x)$, but $P(x|a)$

Most of the times

E.g., stochastic detector output y for given particle parameters x

Conditional distributions

~~Sometimes~~ it's necessary to learn not just $P(x)$, but $P(x|a)$

Most of the times

E.g., stochastic detector output y for given particle parameters x

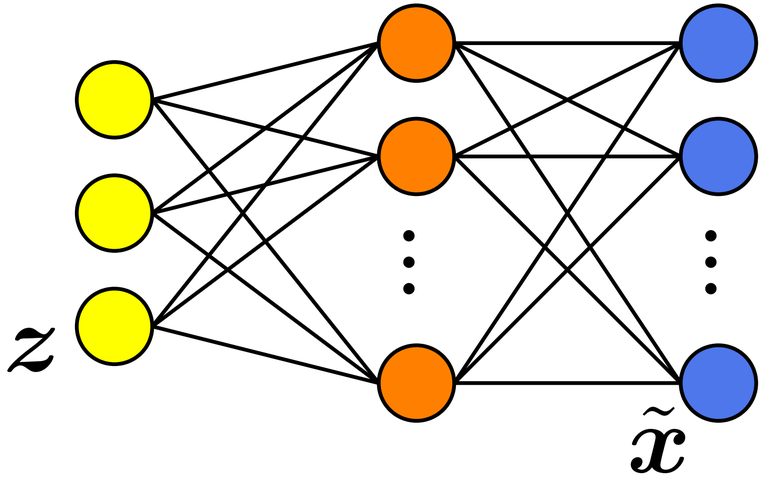
This can be achieved with any GAN architecture:

$$\begin{aligned} \tilde{\boldsymbol{x}}_j = G_\theta(\boldsymbol{z}_j) \quad &\rightarrow \quad \tilde{\boldsymbol{x}}_j = G_\theta(\boldsymbol{z}_j, \boldsymbol{a}_j) \\ D_\phi = D_\phi(\boldsymbol{x}_i) \quad &\rightarrow \quad D_\phi = D_\phi(\boldsymbol{x}_i, \boldsymbol{a}_i) \\ \tilde{D}_\phi = D_\phi(\tilde{\boldsymbol{x}}_j) \quad &\rightarrow \quad \tilde{D}_\phi = D_\phi(\tilde{\boldsymbol{x}}_j, \boldsymbol{a}_j) \end{aligned}$$

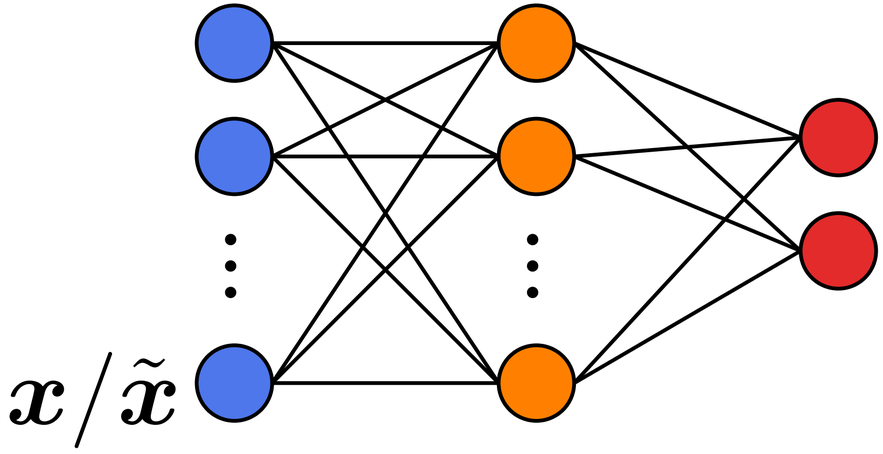
Conditional distributions

(simple case — fully-connected layers)

Generator

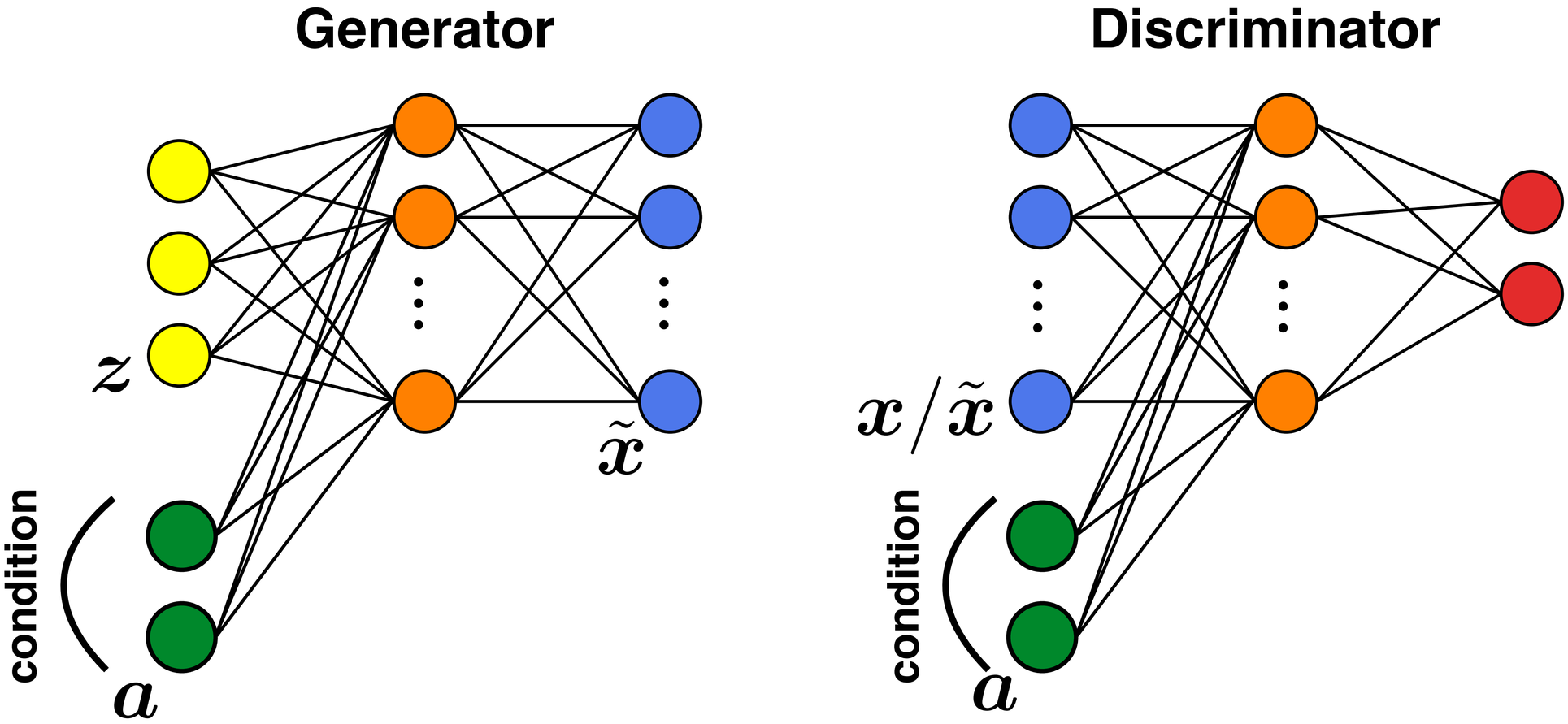


Discriminator



Conditional distributions

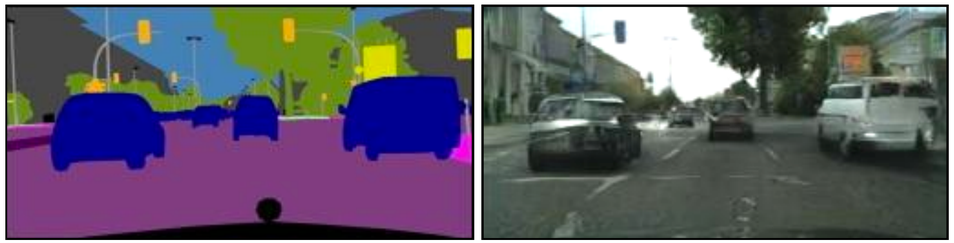
(simple case — fully-connected layers)



pix2pix

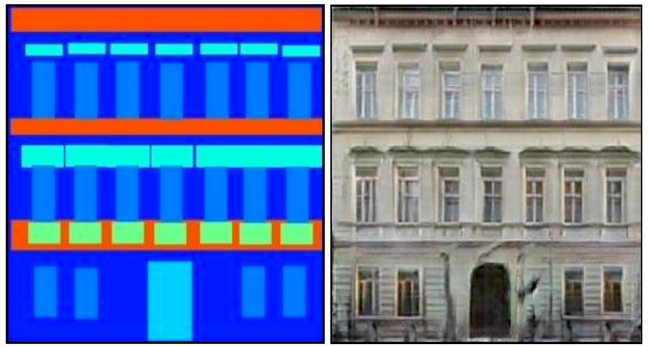
https://arxiv.org/abs/1611.07004

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

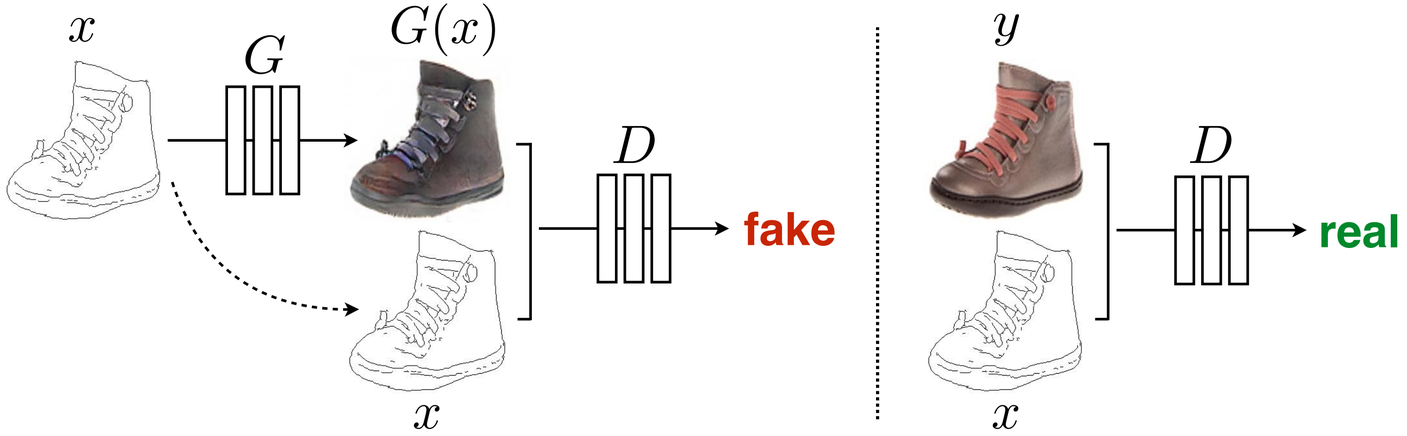
pix2pix

https://arxiv.org/abs/1611.07004

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

Our final objective is

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



pix2pix

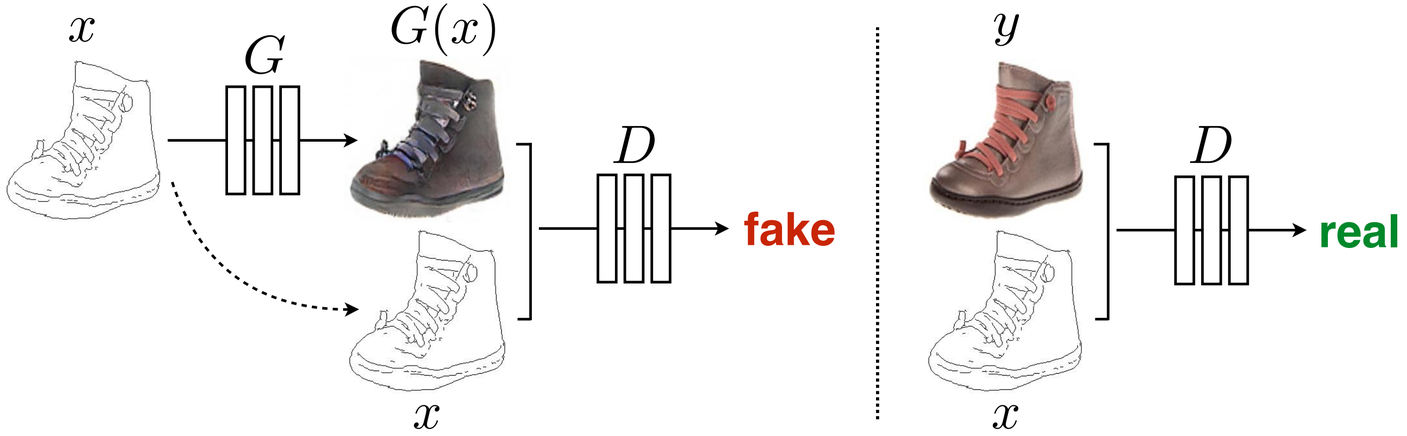
https://arxiv.org/abs/1611.07004

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

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

pix2pix

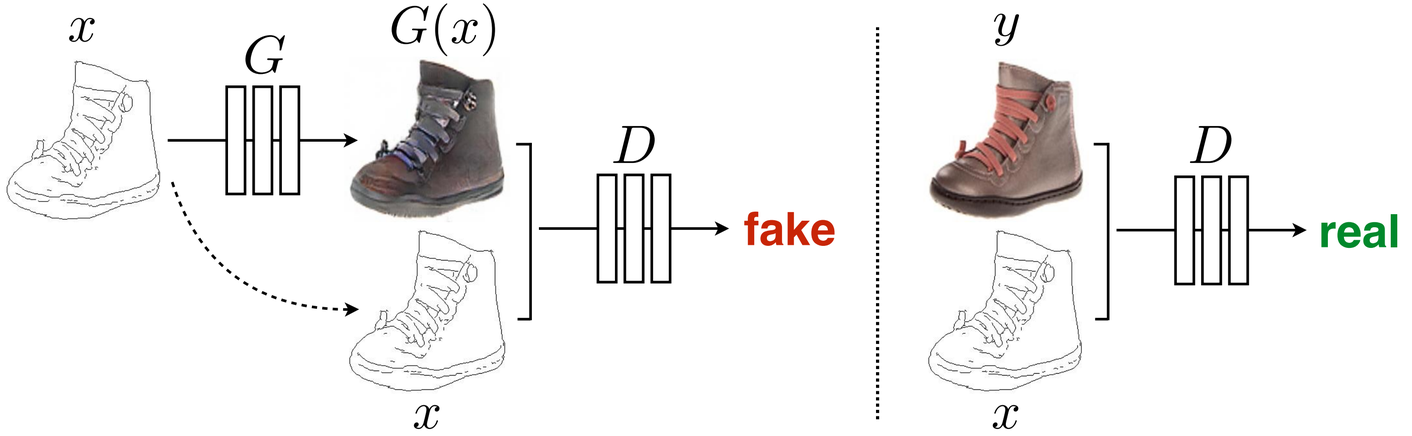
https://arxiv.org/abs/1611.07004

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

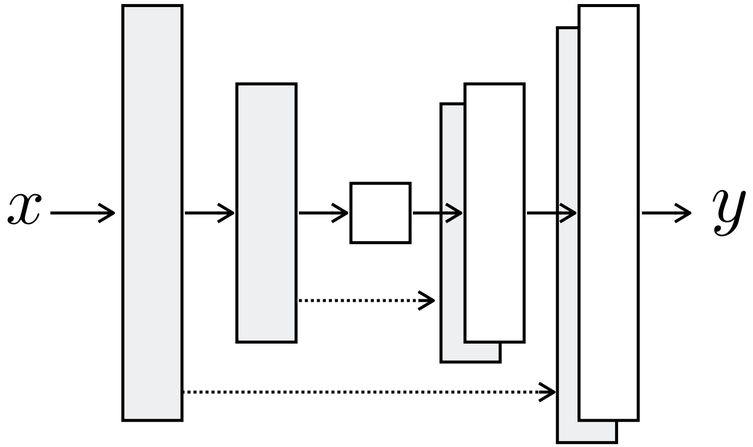
Our final objective is

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

L1 loss term to capture low-frequency information



U-Net



'U-Net' generator architecture

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

pix2pix

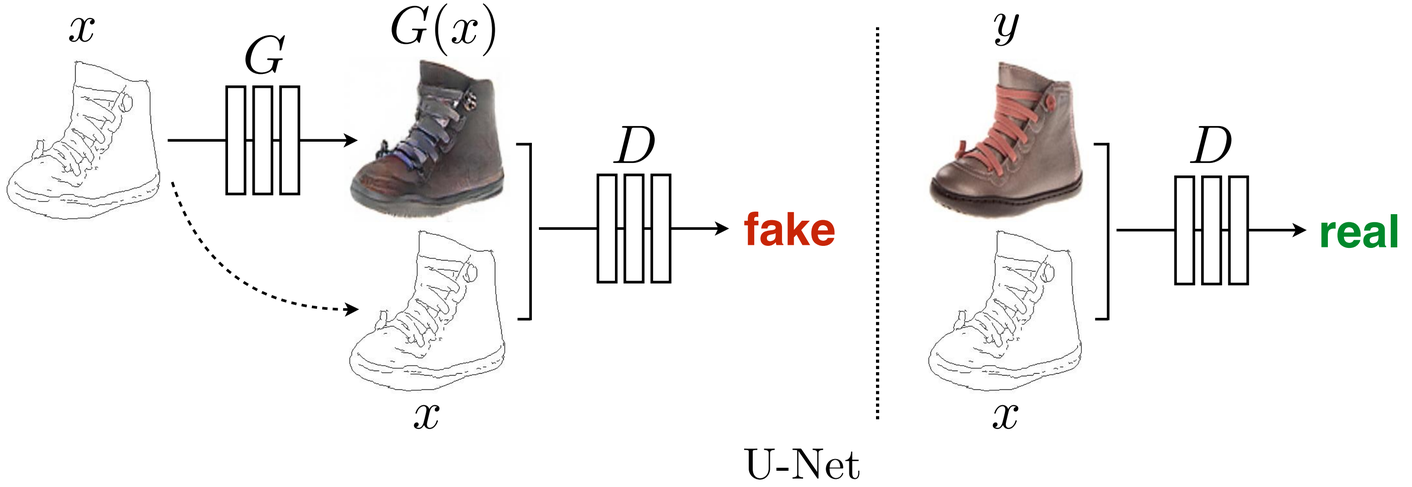
https://arxiv.org/abs/1611.07004

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

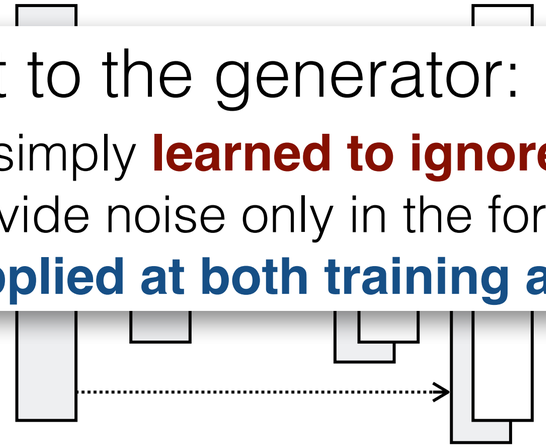
Our final objective is

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

L1 loss term to capture low-frequency information



U-Net



‘U-Net’ generator architecture

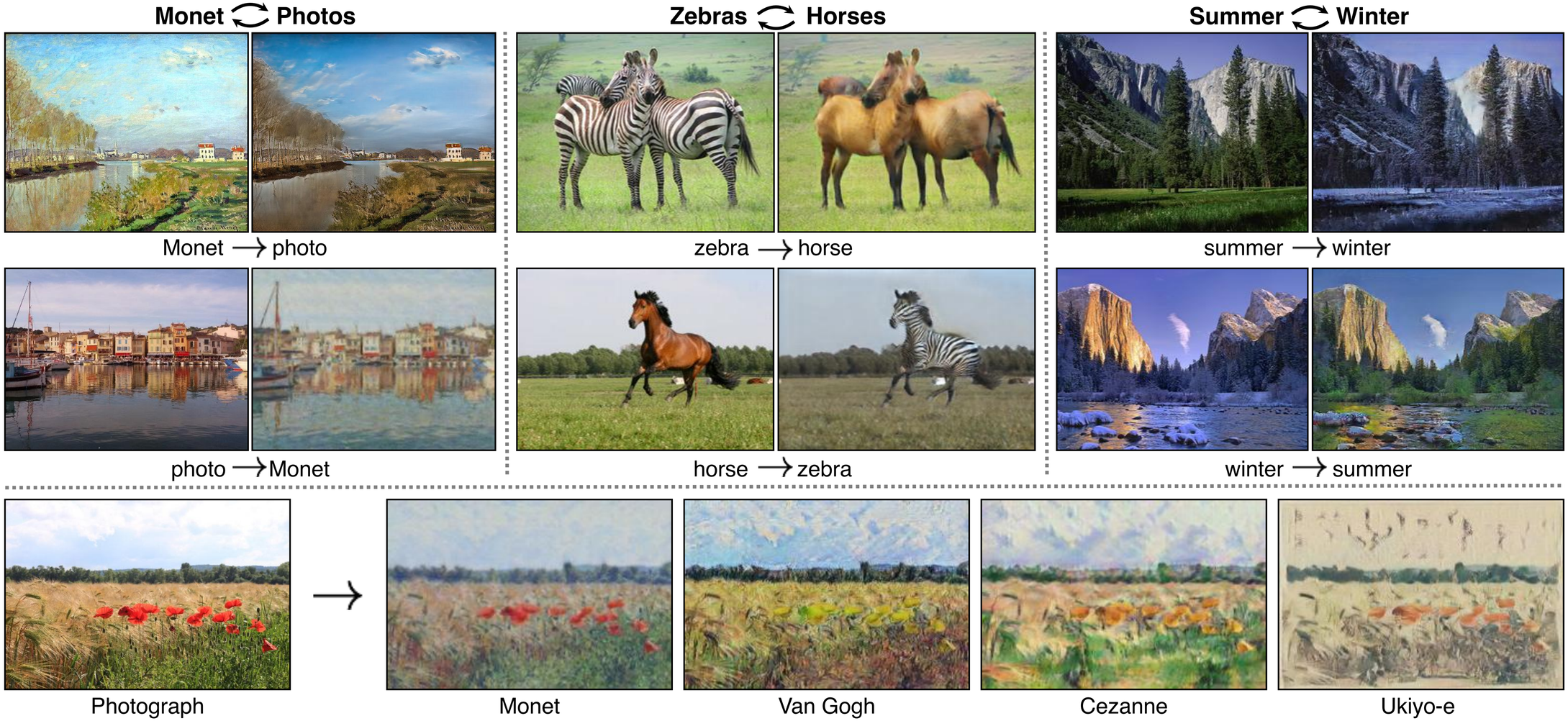
No noise input to the generator:

«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**»

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

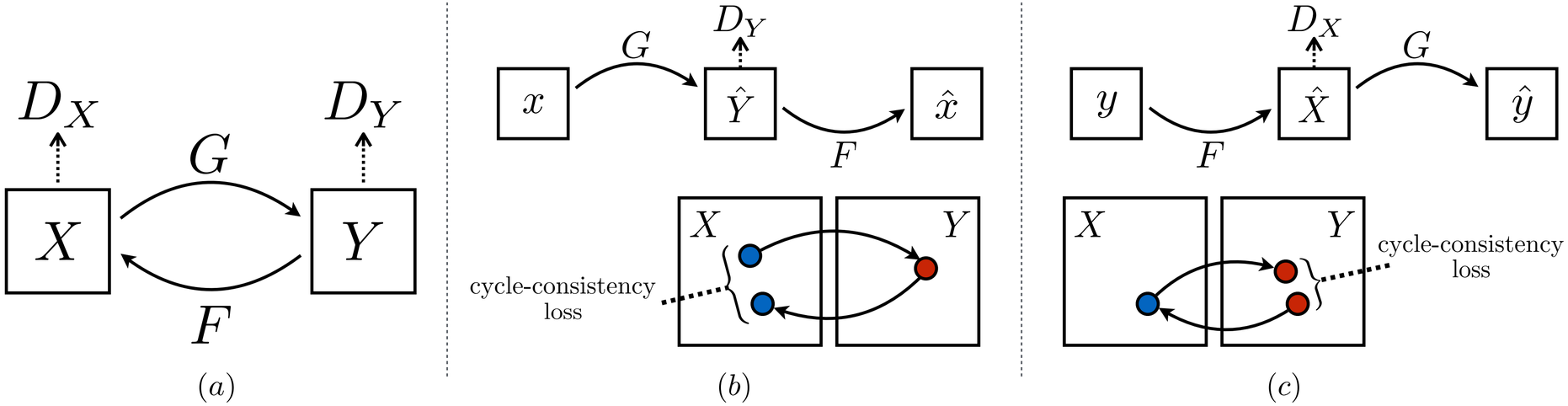
CycleGAN

https://arxiv.org/abs/1703.10593



CycleGAN

https://arxiv.org/abs/1703.10593



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

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

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) + \lambda \mathcal{L}_{\text{cyc}}(G, F), \quad (3)$$

Progressive Growing of GANs

https://arxiv.org/abs/1710.10196

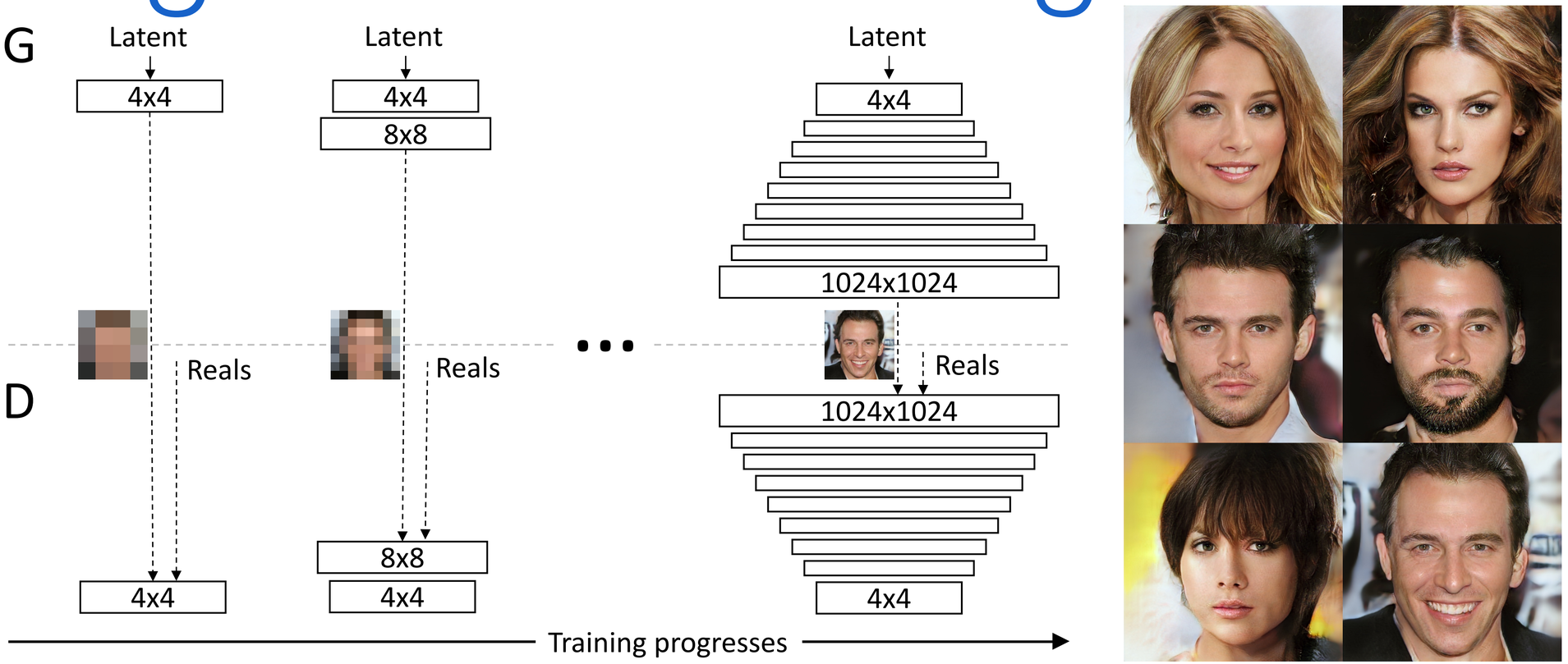


Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout 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. One the right we show six example images generated using progressive growing at 1024×1024.

Progressive Growing of GANs

https://arxiv.org/abs/1710.10196

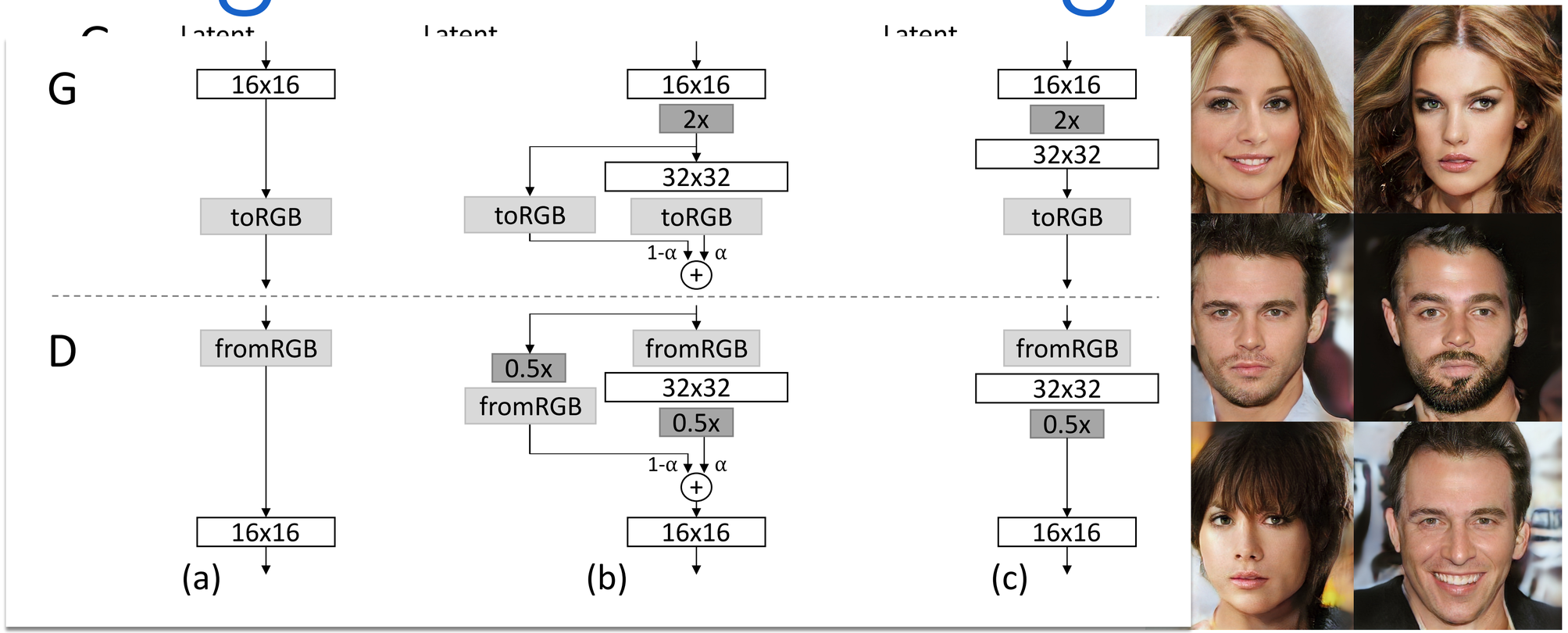


Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout 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. One 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

Evaluating generative models

- No single guide to follow
- Approaches are very problem-specific

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 invariant mass distribution

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 invariant mass distribution
  - Most solutions are adapted to or invented for a given particular task

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 invariant mass distribution
  - Most solutions are adapted to or invented for a given particular task
- We'll mention some notable examples

Evaluating generative models

(most obvious thing to do)

- By-eye comparison
  - Compare individual objects or whole distributions (e.g. in projections) where possible
  - There might be no need to do any complicated evaluation if the model's results simply look bad

Evaluating generative models

(simple things to do)

- Compare meaningful physical characteristics
  - Integral characteristics (e.g. total energy of a calorimeter cluster)
  - Means, medians, standard deviations, etc.

Evaluating generative models

(simple things to do)

- Compare meaningful physical characteristics
  - Integral characteristics (e.g. total energy of a calorimeter cluster)
  - Means, medians, standard deviations, etc.

- Statistical tests (Chi2, Kolmogorov-Smirnov, etc.)
  - Between individual dimensions or projections
  - *p*-values might look insane, probably better to compare the statistics themselves

Evaluating generative models

(the ‘additional classifier’ way)

- Train an independent classifier (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

Evaluating generative models

(Inception score)

Evaluating generative models

(Inception score)

- Introduced in https://arxiv.org/abs/1606.03498

Evaluating generative models

(Inception score)

- Introduced in https://arxiv.org/abs/1606.03498
- Apply the Inception model (pre-trained image classifier) to obtain conditional label distribution $p(y|x)$ for each image x
  - this should be low-entropy (the classifier should be certain)

Evaluating generative models

(Inception score)

- Introduced in https://arxiv.org/abs/1606.03498
- Apply the Inception model (pre-trained image classifier) to obtain 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)

Evaluating generative models

(Inception score)

- Introduced in https://arxiv.org/abs/1606.03498
- Apply the Inception model (pre-trained image classifier) to obtain 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:

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

Evaluating generative models

(Fréchet Inception Distance – FID)

https://arxiv.org/abs/1706.08500

Evaluating generative models

(Fréchet Inception Distance – FID)

https://arxiv.org/abs/1706.08500

- One of drawbacks of IS is that it doesn't care about the true distribution

Evaluating generative models

(Fréchet Inception Distance – FID)

https://arxiv.org/abs/1706.08500

- One of 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)

Evaluating generative models

(Fréchet Inception Distance – FID)

https://arxiv.org/abs/1706.08500

- One of 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

Evaluating generative models

(Fréchet Inception Distance – FID)

https://arxiv.org/abs/1706.08500

- One of 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:

$$\mathrm{FID} = ||\mu_r - \mu_g||^2 + \mathrm{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2})$$

Evaluating generative models

https://arxiv.org/abs/1806.00035 **(Precision and Recall)**

Evaluating generative models

https://arxiv.org/abs/1806.00035 **(Precision and Recall)**

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*

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

Evaluating generative models

https://arxiv.org/abs/1806.00035 **(Precision and Recall)**

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

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

Evaluating generative models

https://arxiv.org/abs/1806.00035 **(Precision and Recall)**

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

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

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

Evaluating generative models

https://arxiv.org/abs/1806.00035

(Precision and Recall)

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

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

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

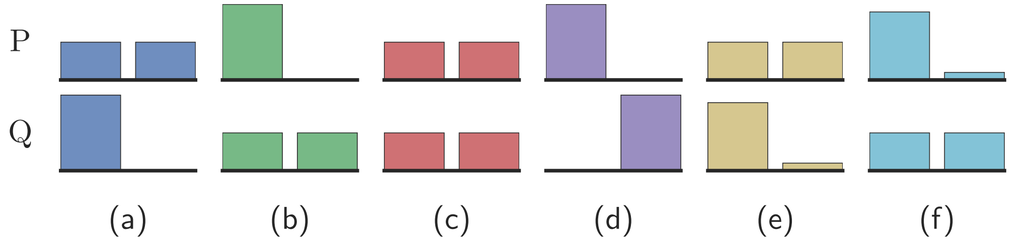


Figure 2: Intuitive examples of P and Q.

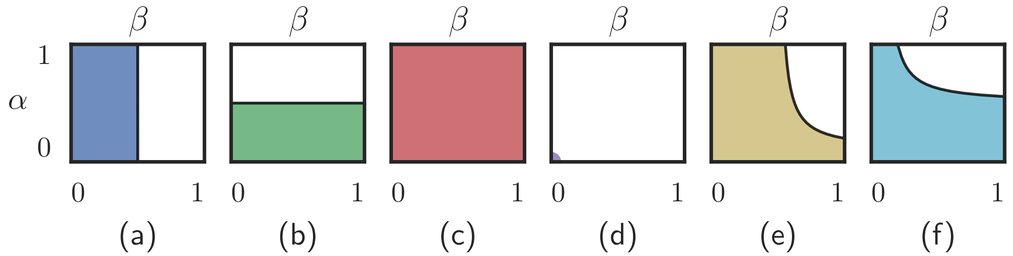


Figure 3: $\mathrm{PRD}(Q, P)$ for the examples above.

Evaluating generative models

https://arxiv.org/abs/1806.00035 **(Precision and Recall)**

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

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

Definition 2. *The set of attainable pairs of precision and recall of a distribution Q w.r.t. a distribution P is denoted by $\mathrm{PRD}(Q, P)$ 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

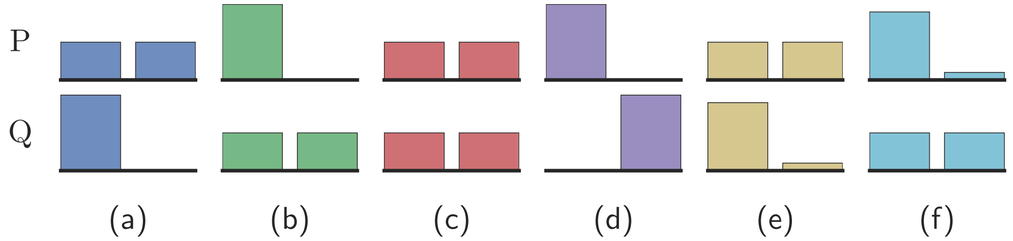


Figure 2: Intuitive examples of P and Q.

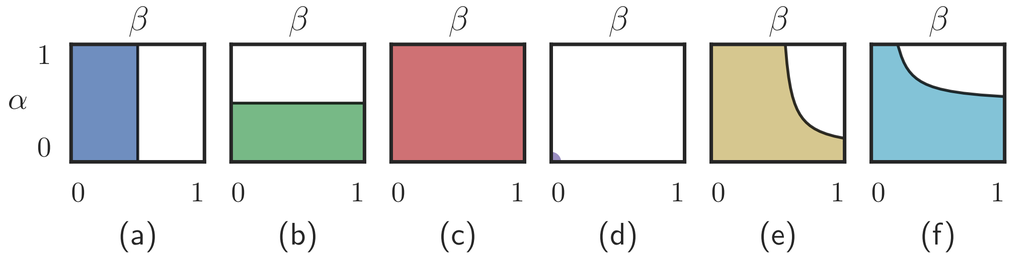


Figure 3: $\mathrm{PRD}(Q, P)$ for the examples above.

Evaluating generative models

https://arxiv.org/abs/1806.00035

(Precision and Recall)

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

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

Definition 2. *The set of attainable pairs of precision and recall of a distribution Q w.r.t. a distribution P is denoted by $\mathrm{PRD}(Q, P)$ 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

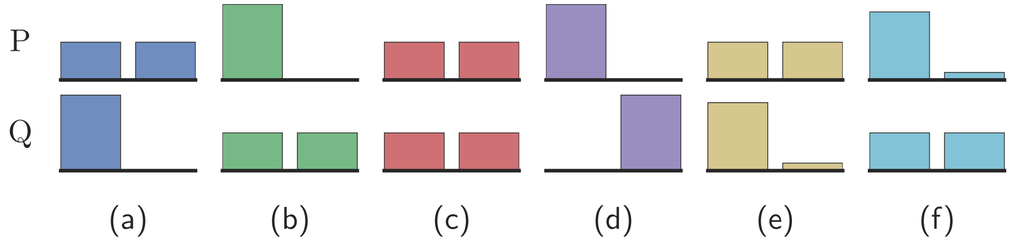


Figure 2: Intuitive examples of P and Q.

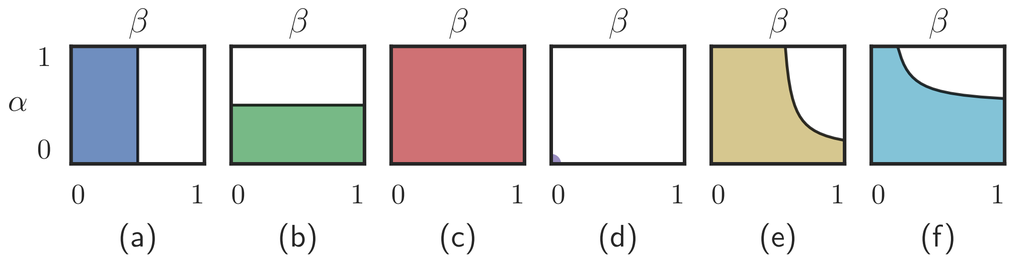


Figure 3: $\mathrm{PRD}(Q, P)$ for the examples above.

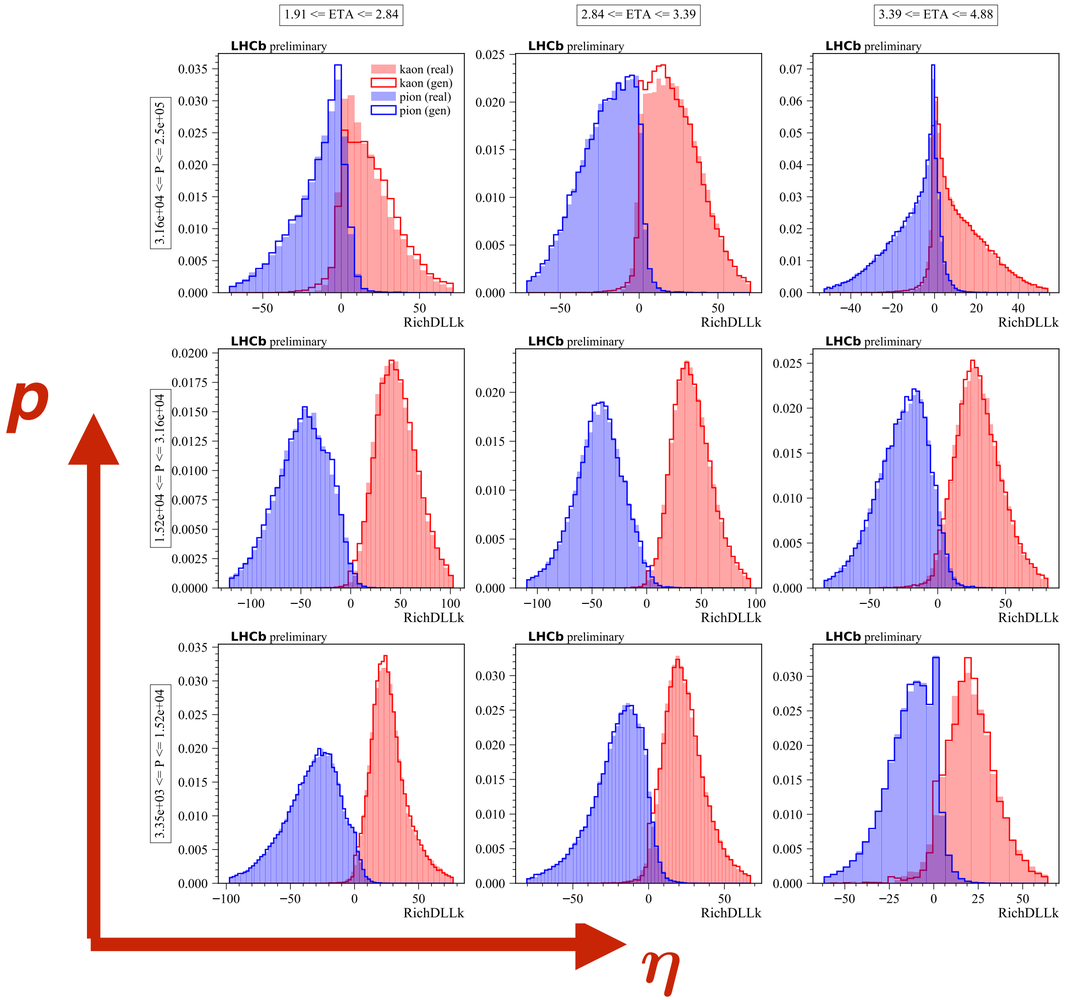
Evaluating generative models

More on the subject (extensive comparison of a large variety of measures):

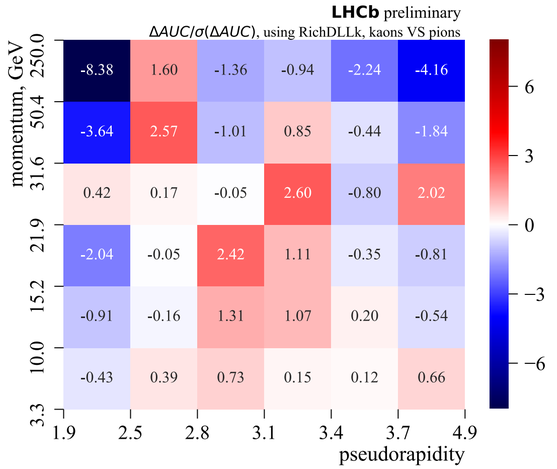
https://arxiv.org/abs/1802.03446

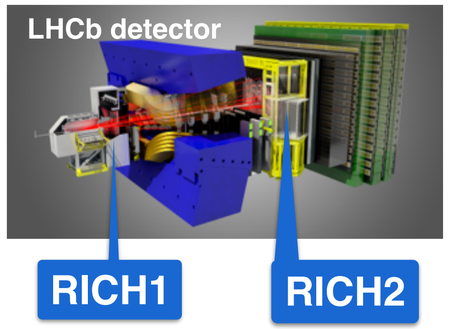
	Measure	Description
Quantitative	1. Average Log-likelihood [18, 22]	• Log likelihood of explaining realworld held out/test data using a density estimated from the generated data (*e.g.* using KDE or Parzen window estimation). $L = \frac{1}{N}\sum_i \log P_{model}(\mathbf{x}_i)$
	2. Coverage Metric [33]	• The probability mass of the true data "covered" by the model distribution $C := P_{data}(dP_{model} > t)$ with t such that $P_{model}(dP_{model} > t) = 0.95$
	3. Inception Score (IS) [3]	• KLD between conditional and marginal label distributions over generated data. $\exp\left(\mathbb{E}_{\mathbf{x}}\left[\mathbb{KL}\left(p\left(y \mid \mathbf{x}\right) \parallel p\left(y\right)\right)\right]\right)$
	4. Modified Inception Score (m-IS) [34]	• Encourages diversity within images sampled from a particular category. $\exp(\mathbb{E}_{\mathbf{x}_i}[\mathbb{E}_{\mathbf{x}_j}[(\mathbb{KL}(P(y\|\mathbf{x}_i)\|\|P(y\|\mathbf{x}_j))]]]$
	5. Mode Score (MS) [35]	• Similar to IS but also takes into account the prior distribution of the labels over real data. $\exp\left(\mathbb{E}_{\mathbf{x}}\left[\mathbb{KL}\left(p\left(y \mid \mathbf{x}\right) \parallel p\left(y^{train}\right)\right)\right] - \mathbb{KL}\left(p\left(y\right) \parallel p\left(y^{train}\right)\right)\right)$
	6. AM Score [36]	• Takes into account the KLD between distributions of training labels vs. predicted labels, as well as the entropy of predictions. $\mathbb{KL}(p(y^{\text{train}}) \parallel p(y)) + \mathbb{E}_{\mathbf{x}}\left[H(y\|\mathbf{x})\right]$
	7. Fréchet Inception Distance (FID) [37]	• Wasserstein-2 distance between multi-variate Gaussians fitted to data embedded into a feature space $FID(r,g) = \|\mu_r - \mu_g\|_2^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{\frac{1}{2}})$
	8. Maximum Mean Discrepancy (MMD) [38]	• Measures the dissimilarity between two probability distributions P_r and P_g using samples drawn independently from each distribution. $M_k(P_r, P_g) = \mathbb{E}_{\mathbf{x},\mathbf{x}'\sim P_r}[k(\mathbf{x},\mathbf{x}')] - 2\mathbb{E}_{\mathbf{x}\sim P_r,\boldsymbol{y}\sim P_g}[k(\mathbf{x},\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{y},\boldsymbol{y}'\sim P_g}[k(\boldsymbol{y},\boldsymbol{y}')]$
	9. The Wasserstein Critic [39]	• The critic (*e.g.* an NN) is trained to produce high values at real samples and low values at generated samples $\hat{W}(\mathbf{x}_{test}, \mathbf{x}_g) = \frac{1}{N}\sum_{i=1}^{N}\hat{f}(\mathbf{x}_{test}[i]) - \frac{1}{N}\sum_{i=1}^{N}\hat{f}(\mathbf{x}_g[i])$
	10. Birthday Paradox Test [27]	• Measures the support size of a discrete (continuous) distribution by counting the duplicates (near duplicates)
	11. Classifier Two Sample Test (C2ST) [40]	• Answers whether two samples are drawn from the same distribution (*e.g.* by training a binary classifier)
	12. Classification Performance [1, 15]	• An indirect technique for evaluating the quality of unsupervised representations (*e.g.* feature extraction; FCN score). See also the GAN Quality Index (GQI) [41].
	13. Boundary Distortion [42]	• Measures diversity of generated samples and covariate shift using classification methods.
	14. Number of Statistically-Different Bins (NDB) [43]	• 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
	15. Image Retrieval Performance [44]	• Measures the distributions of distances to the nearest neighbors of some query images (*i.e.* diversity)
	16. Generative Adversarial Metric (GAM) [31]	• Compares two GANs by having them engaged in a battle against each other by swapping discriminators or generators. $p(\mathbf{x}\|y=1; M_1^{\cdot})/p(\mathbf{x}\|y=1; M_2^{\cdot}) = \big(p(y=1\|\mathbf{x}; D_1)p(\mathbf{x}; G_2)\big)/\big(p(y=1\|\mathbf{x}; D_2)p(\mathbf{x}; G_1)\big)$
	17. Tournament Win Rate and Skill Rating [45]	• 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.
	18. Normalized Relative Discriminative Score (NRDS) [32]	• 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.
	19. Adversarial Accuracy and Divergence [46]	• 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 $\mathbb{KL}(P_g(y\|\mathbf{x}), P_r(y\|\mathbf{x}))$
	20. Geometry Score [47]	• Compares geometrical properties of the underlying data manifold between real and generated data.
	21. Reconstruction Error [48]	• Measures the reconstruction error (*e.g.* L_2 norm) between a test image and its closest generated image by optimizing for z (*i.e.* $min_{\mathbf{z}}\|\|G(\mathbf{z}) - \mathbf{x}^{(test)}\|\|^2$)
	22. Image Quality Measures [49, 50, 51]	• Evaluates the quality of generated images using measures such as SSIM, PSNR, and sharpness difference
	23. Low-level Image Statistics [52, 53]	• 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.
	24. Precision, Recall and F_1 score [23]	• These measures are used to quantify the degree of overfitting in GANs, often over toy datasets.
Qualitative	1. Nearest Neighbors	• To detect overfitting, generated samples are shown next to their nearest neighbors in the training set
	2. Rapid Scene Categorization [18]	• 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
	3. Preference Judgment [54, 55, 56, 57]	• Participants are asked to rank models in terms of the fidelity of their generated images (*e.g.* pairs, triples)
	4. Mode Drop and Collapse [58, 59]	• 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
	5. Network Internals [1, 60, 61, 62, 63, 64]	• Regards exploring and illustrating the internal representation and dynamics of models (*e.g.* space continuity) as well as visualizing learned features

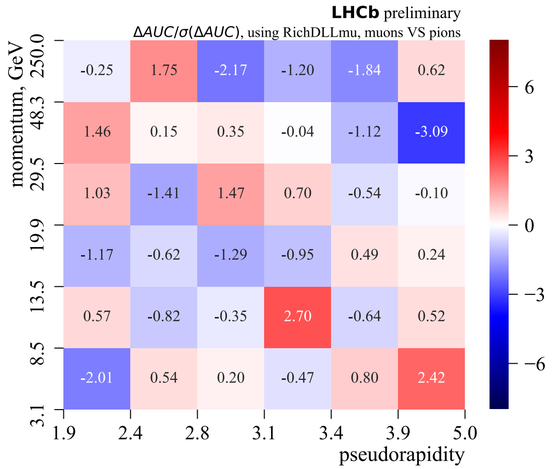
Fast simulation of Cherenkov detectors at LHCb

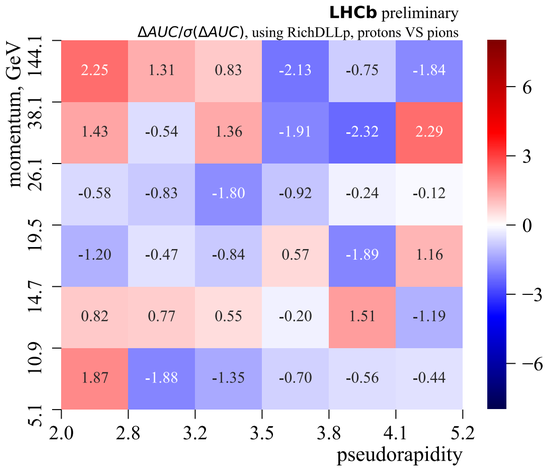


Detector responce to kaons and pions
(real data vs generated with GAN)





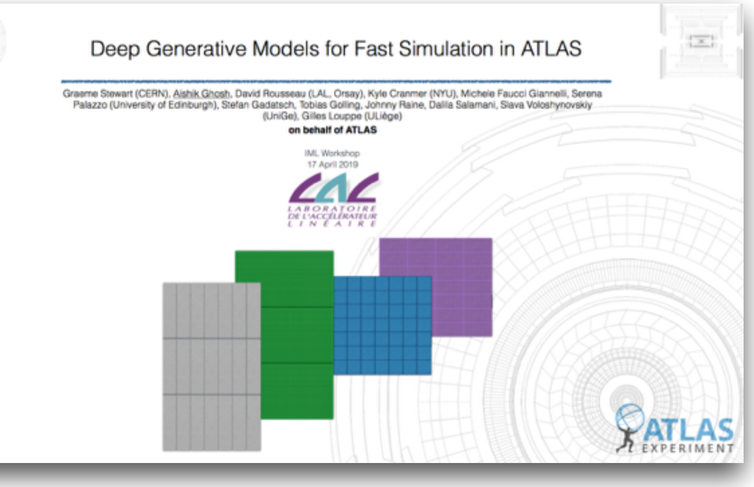




Difference between real and generated
ROC AUCs in units of stat uncertainty

https://arxiv.org/abs/1905.11825

EM calorimeter shower fast simulation in ATLAS



The Calorimeter

2-D Axis: ϕ vs η

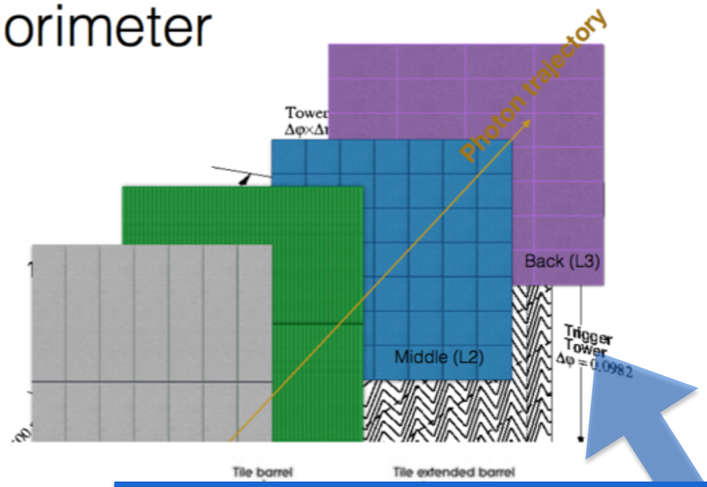
Particle goes through 4 layers in this order:

0. Pre-Sampler : Some energy deposit

1. Strips: Very granular in η; more energy deposit

2. Middle: Thickest layer, maximum energy deposit

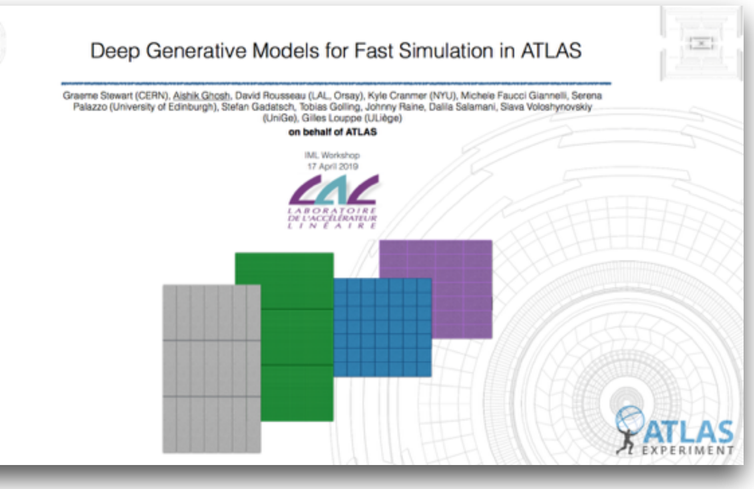
3. Back: Little Energy deposits



Generating output for 3D calorimeter structure (4 calorimeter layers, 266 output channels)

https://indico.cern.ch/event/766872/contributions/3357991/

EM calorimeter shower fast simulation in ATLAS



The Calorimeter

2-D Axis: ϕ vs η

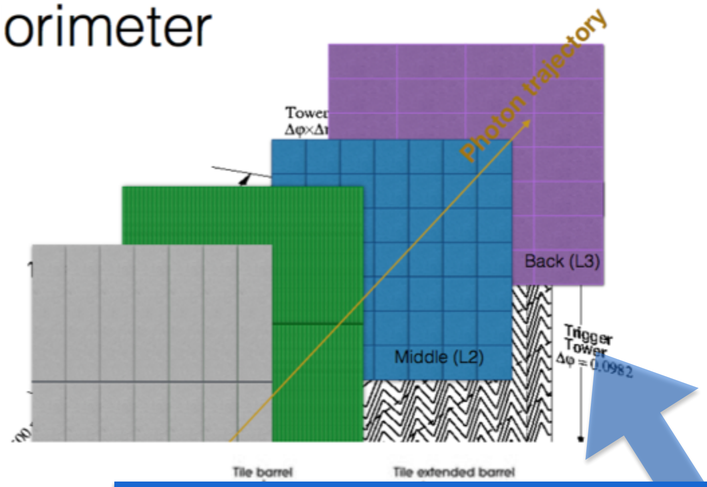
Particle goes through 4 layers in this order:

0. Pre-Sampler : Some energy deposit

1. Strips: Very granular in η; more energy deposit

2. Middle: Thickest layer, maximum energy deposit

3. Back: Little Energy deposits

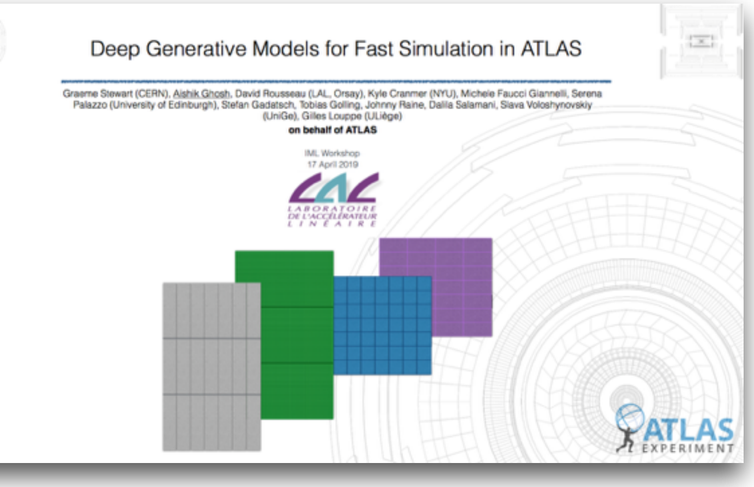


Generating output for 3D calorimeter structure (4 calorimeter layers, 266 output channels)

Evaluating GAN performance through physics observables ($\Delta\eta$, $\Delta\varphi$, $E_{\text{sim}}/E_{\text{truth}}$, etc.)

https://indico.cern.ch/event/766872/contributions/3357991/

EM calorimeter shower fast simulation in ATLAS



The Calorimeter

2-D Axis: ϕ vs η

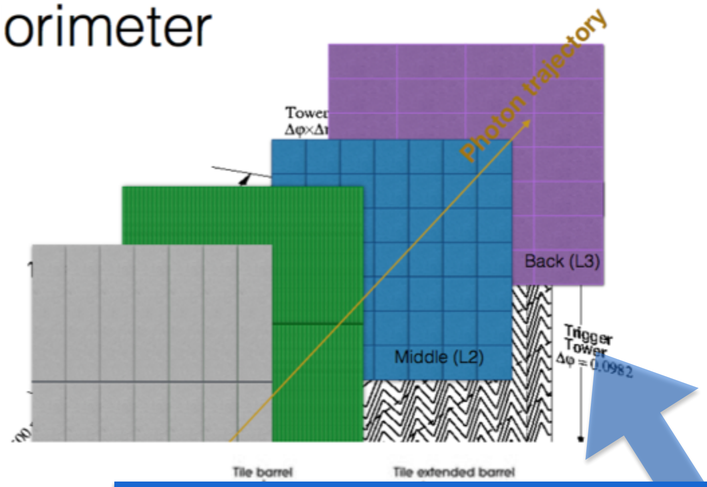
Particle goes through 4 layers in this order:

0. Pre-Sampler : Some energy deposit

1. Strips: Very granular in η; more energy deposit

2. Middle: Thickest layer, maximum energy deposit

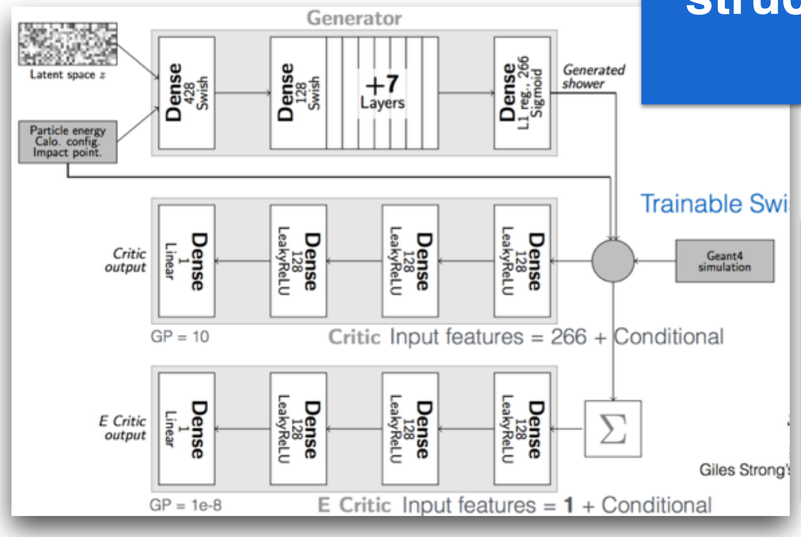
3. Back: Little Energy deposits



Generating output for 3D calorimeter structure (4 calorimeter layers, 266 output channels)

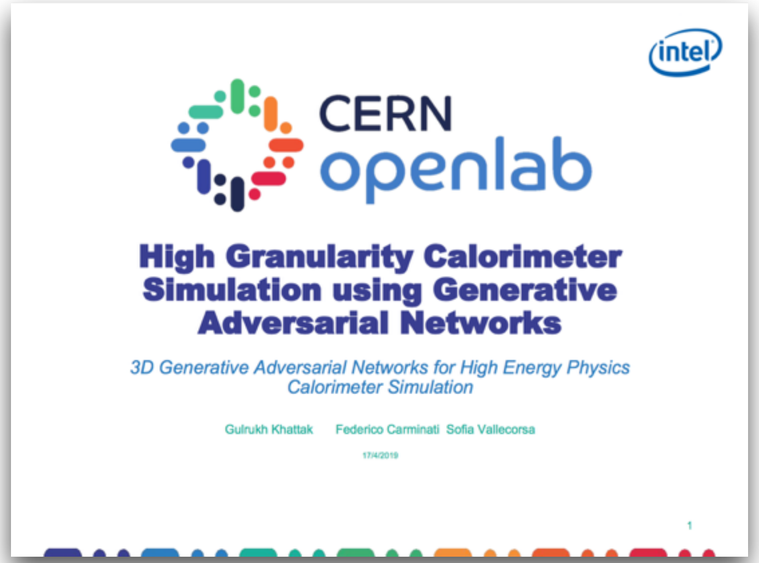
Evaluating GAN performance through physics observables ($\Delta\eta$, $\Delta\varphi$, $E_{\text{sim}}/E_{\text{truth}}$, etc.)

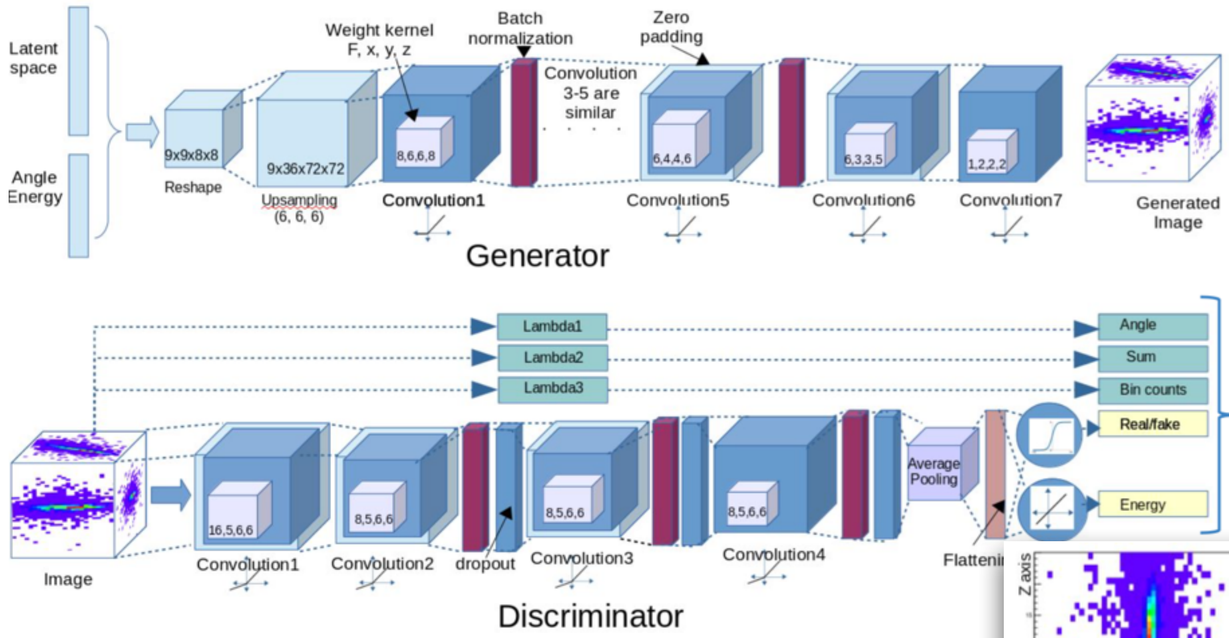
https://indico.cern.ch/event/766872/contributions/3357991/

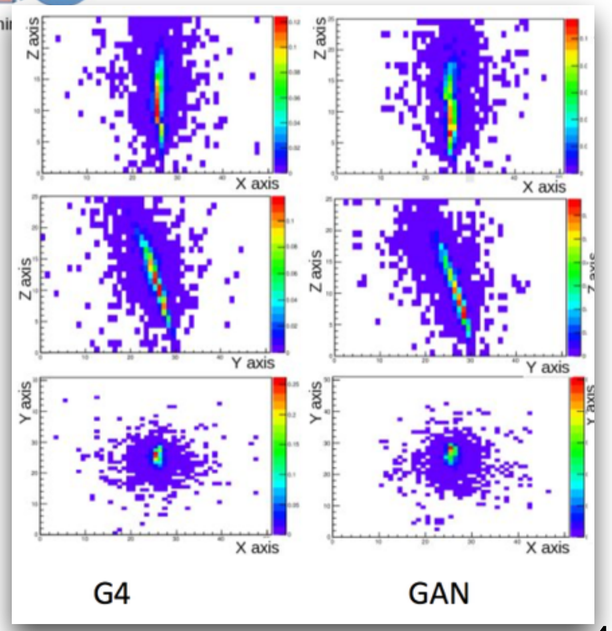


Two critics trained simultaneously (additional critic to look at the total energy distribution)

High granularity calorimeter fast simulation for CLIC

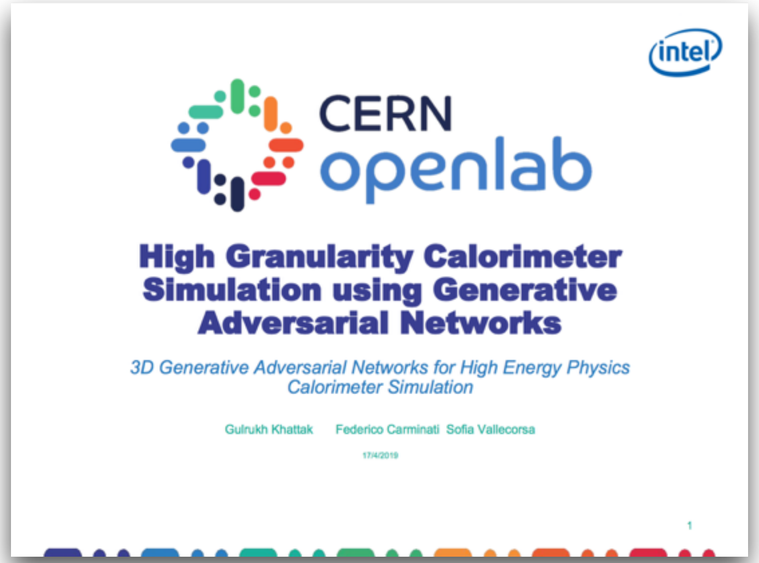


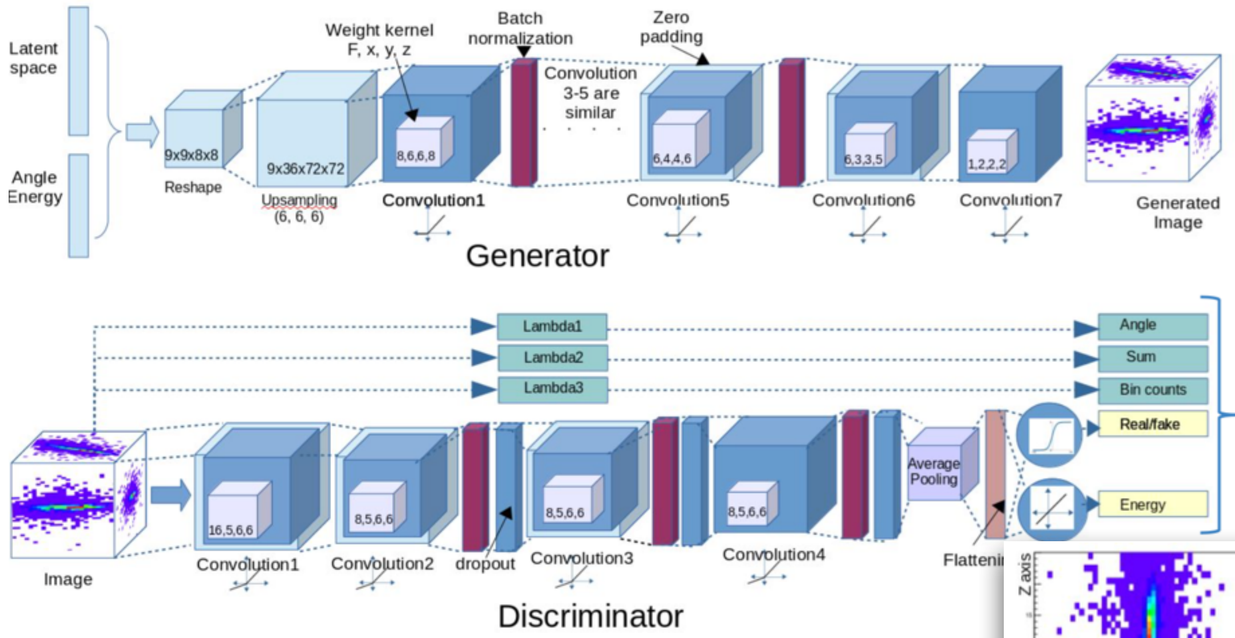




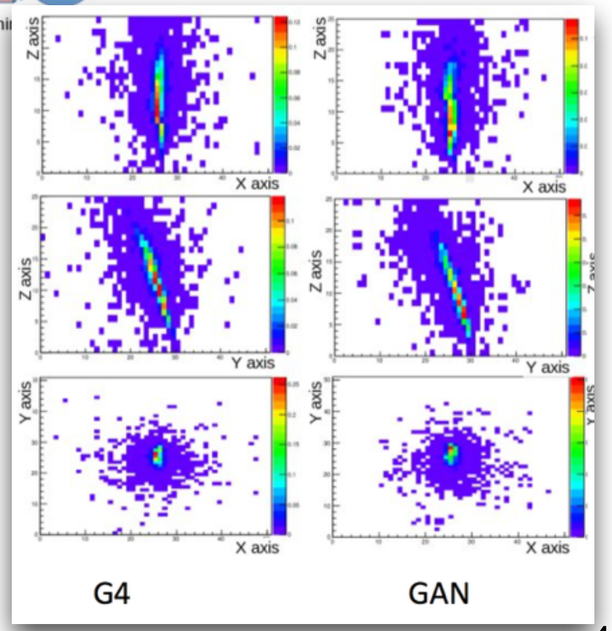
https://indico.cern.ch/event/766872/contributions/3357987/

High granularity calorimeter fast simulation for CLIC



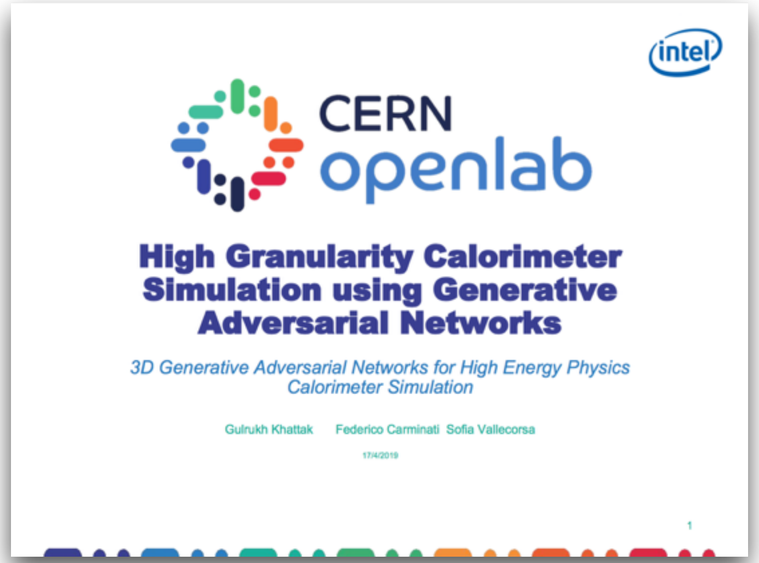


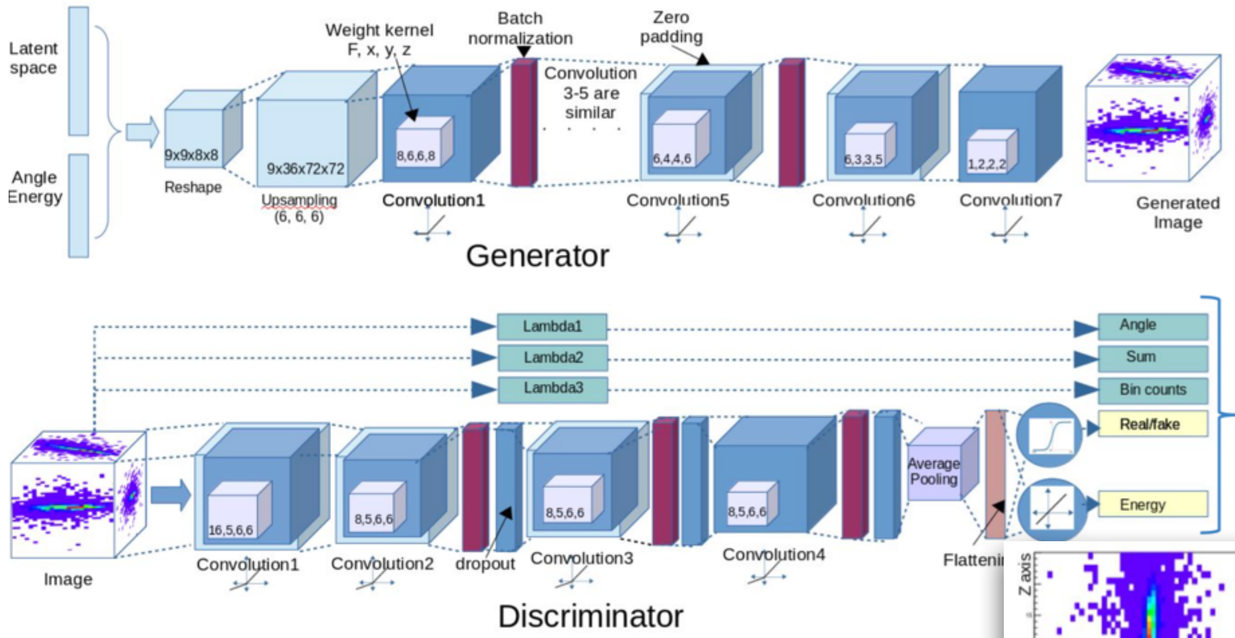
Evaluating with >200 physics observables!!!



https://indico.cern.ch/event/766872/contributions/3357987/

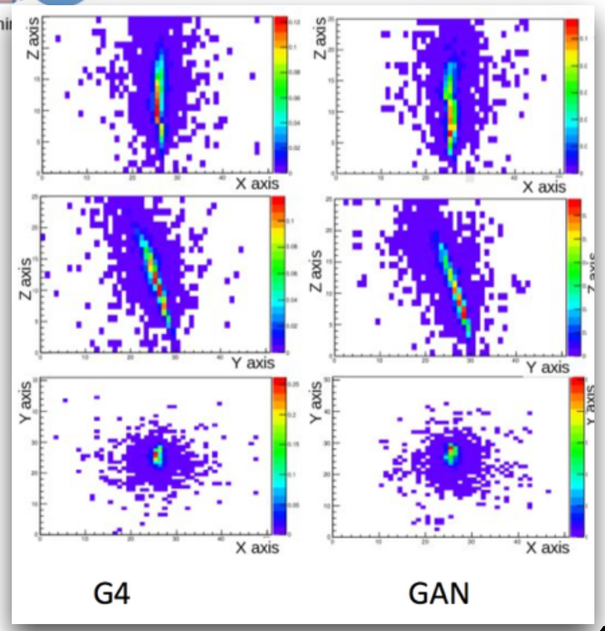
High granularity calorimeter fast simulation for CLIC





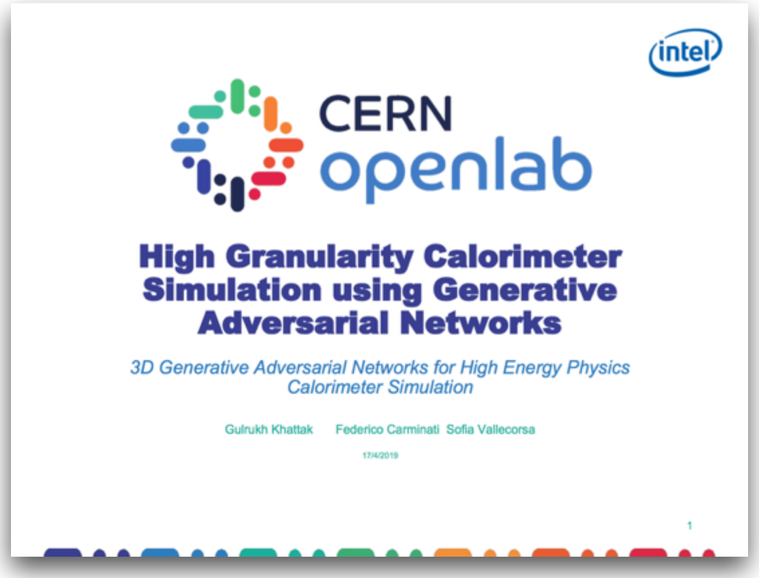
Evaluating with >200 physics observables!!!

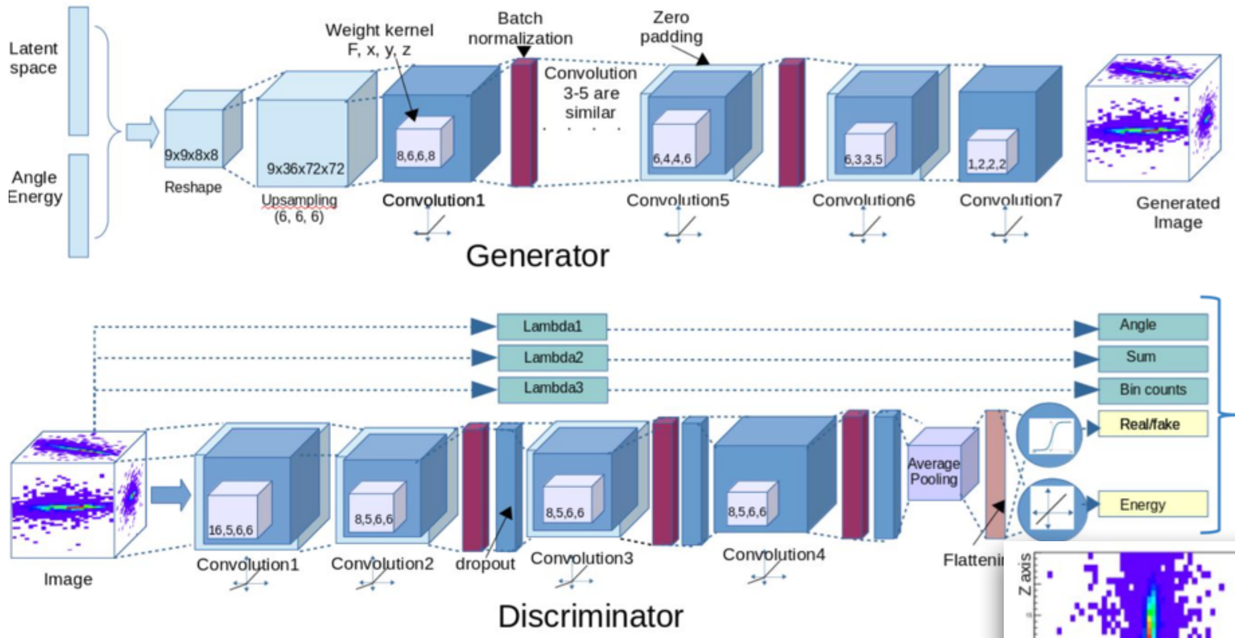
+ using structural similarity index (SSIM) on calorimeter images



https://indico.cern.ch/event/766872/contributions/3357987/

High granularity calorimeter fast simulation for CLIC



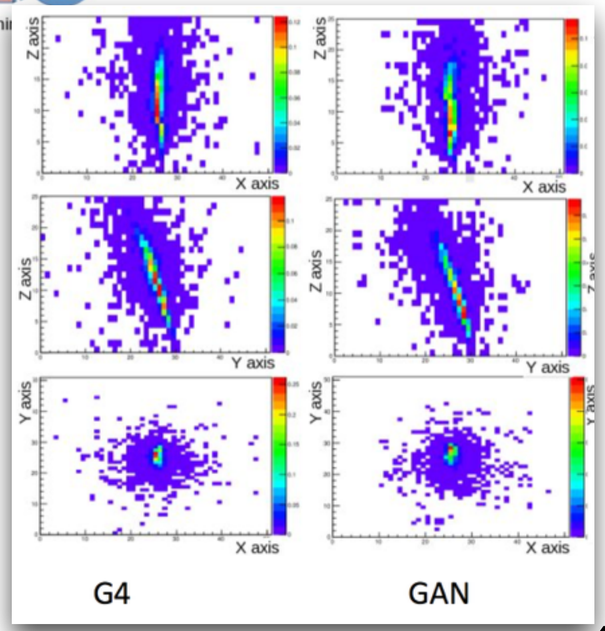


Sparse images with ~65000 pixels!!!

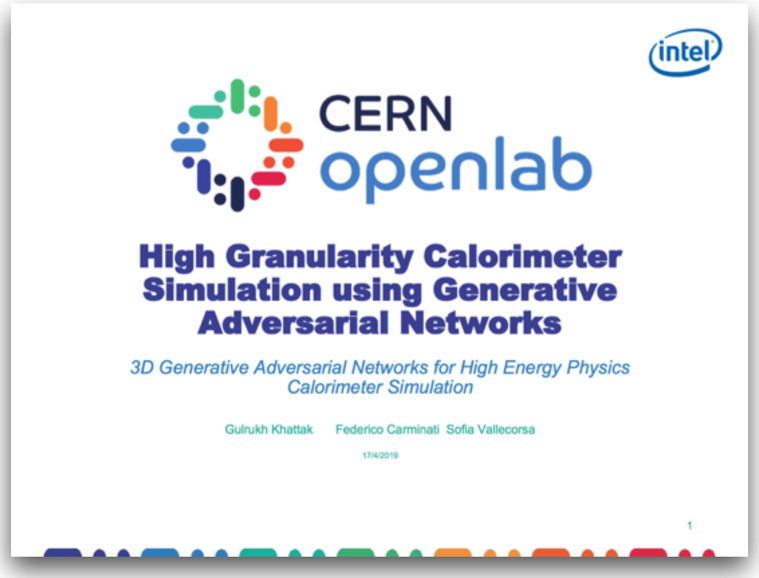
https://indico.cern.ch/event/766872/contributions/3357987/

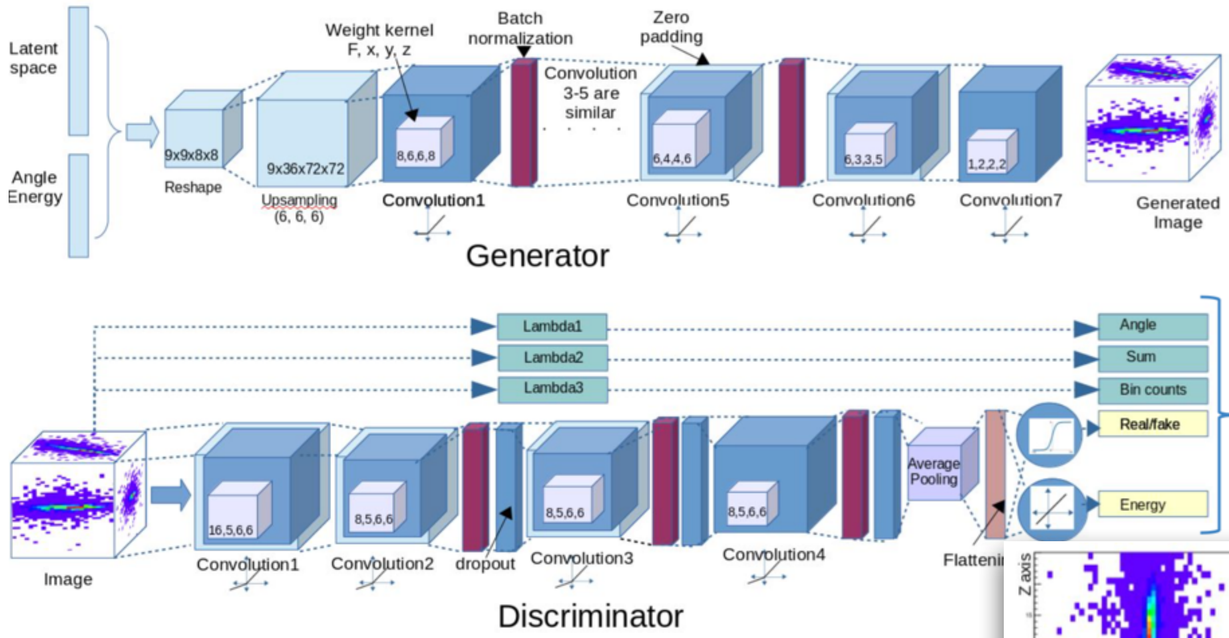
Evaluating with >200 physics observables!!!

+ using structural similarity index (SSIM) on calorimeter images



High granularity calorimeter fast simulation for CLIC





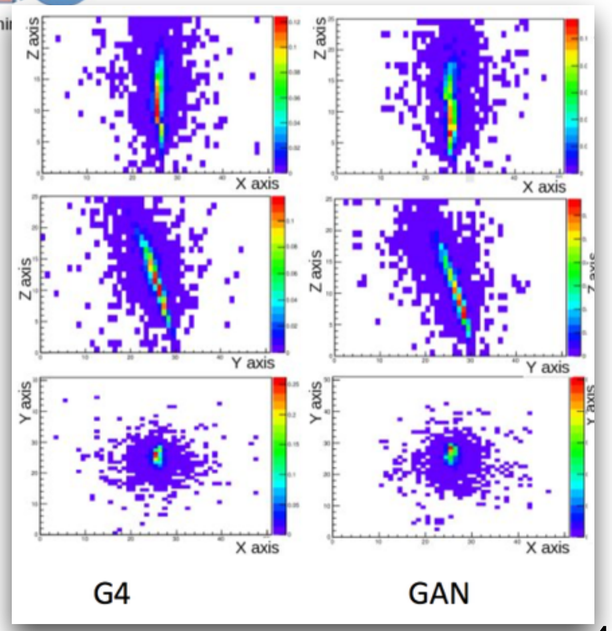
Discriminator also predicts energy

Sparse images with ~65000 pixels!!!

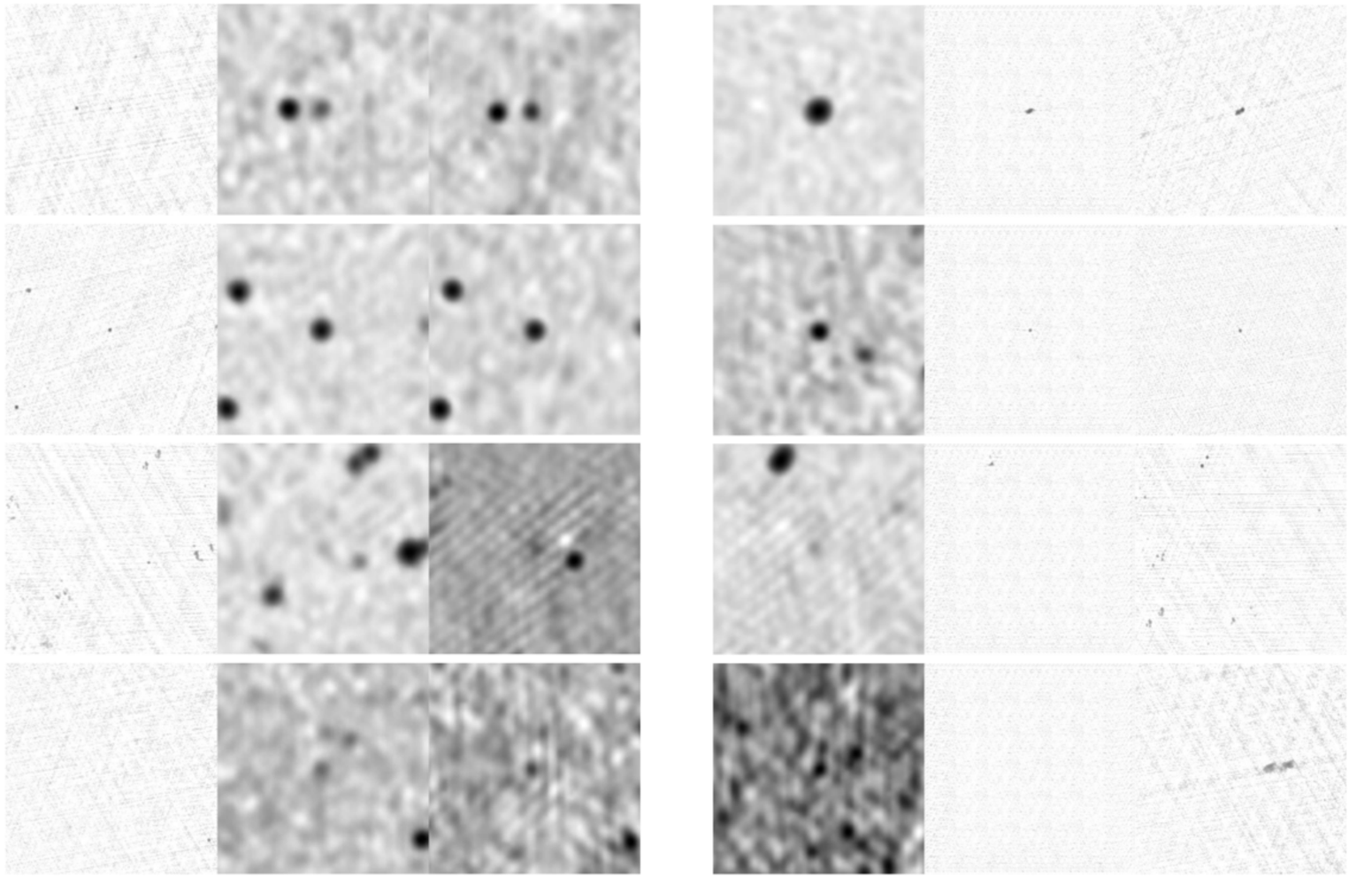
https://indico.cern.ch/event/766872/contributions/3357987/

Evaluating with >200 physics observables!!!

+ using structural similarity index (SSIM) on calorimeter images



RadioGAN (Astrophysics)



GAN to compensate for the trade-off between brightness sensitivity and angular resolution

https://arxiv.org/abs/1906.03874

Summary

- GANs – a rather broad field of ML
  - A lot of different architectures, this lecture doesn't pretend to be comprehensive
  - E.g. here's a single paper introducing 5 new GANs: https://arxiv.org/abs/1606.00709
- 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 a universal quality evaluation method is rather an open question