

# Generative models II

## Generative adversarial networks

Nikita Kazeev

# Where can I use it?

Lots and lots of image manipulation fun

Lots and lots of text manipulation fun

Some (High-Energy) Physics, very preliminary:

Jets images

Simulations of calorimeters and RICH

Cosmological maps

# Problem: comparing images

For naive pixel wise metric “cat on the left” is closer to “dog on the left” than to “cat on the right”

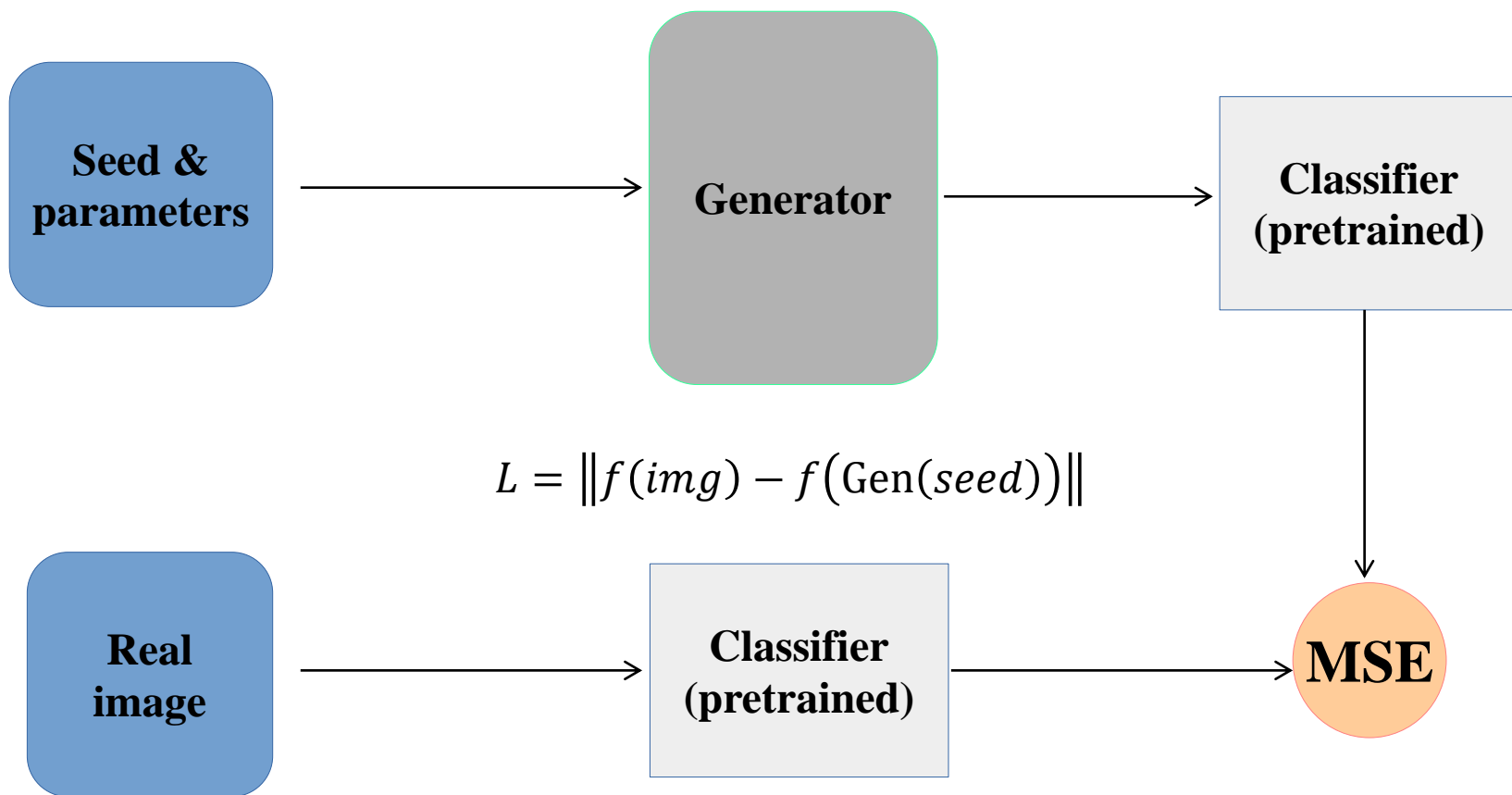
We may want to avoid that effect

Can we obtain image representation that is less sensitive to small shifts?

# | Ideas?

Do we have a representation that focuses on  
“semantics”?

# Sketch: using pre-trained classifiers



**WHAT IF WE TRAIN  
THAT 2-ND NETWORK**



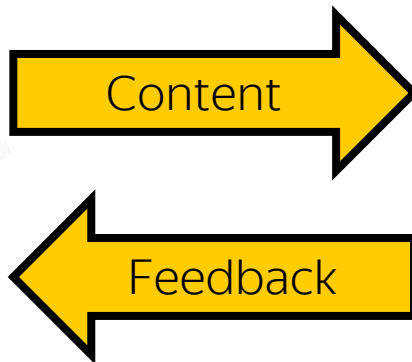
**TO HELP US TRAIN  
THE FIRST NETWORK**

# Generative Adversarial Networks

Generator



Generate image  
(should be plausible)



Discriminator



Tell if image is plausible  
(image)  $\rightarrow$   $P(\text{fake})$

# Generative Adversarial Networks

Generator



Discriminator





# Training a GAN: Algorithm

for  $k$  in  $1 \dots K$

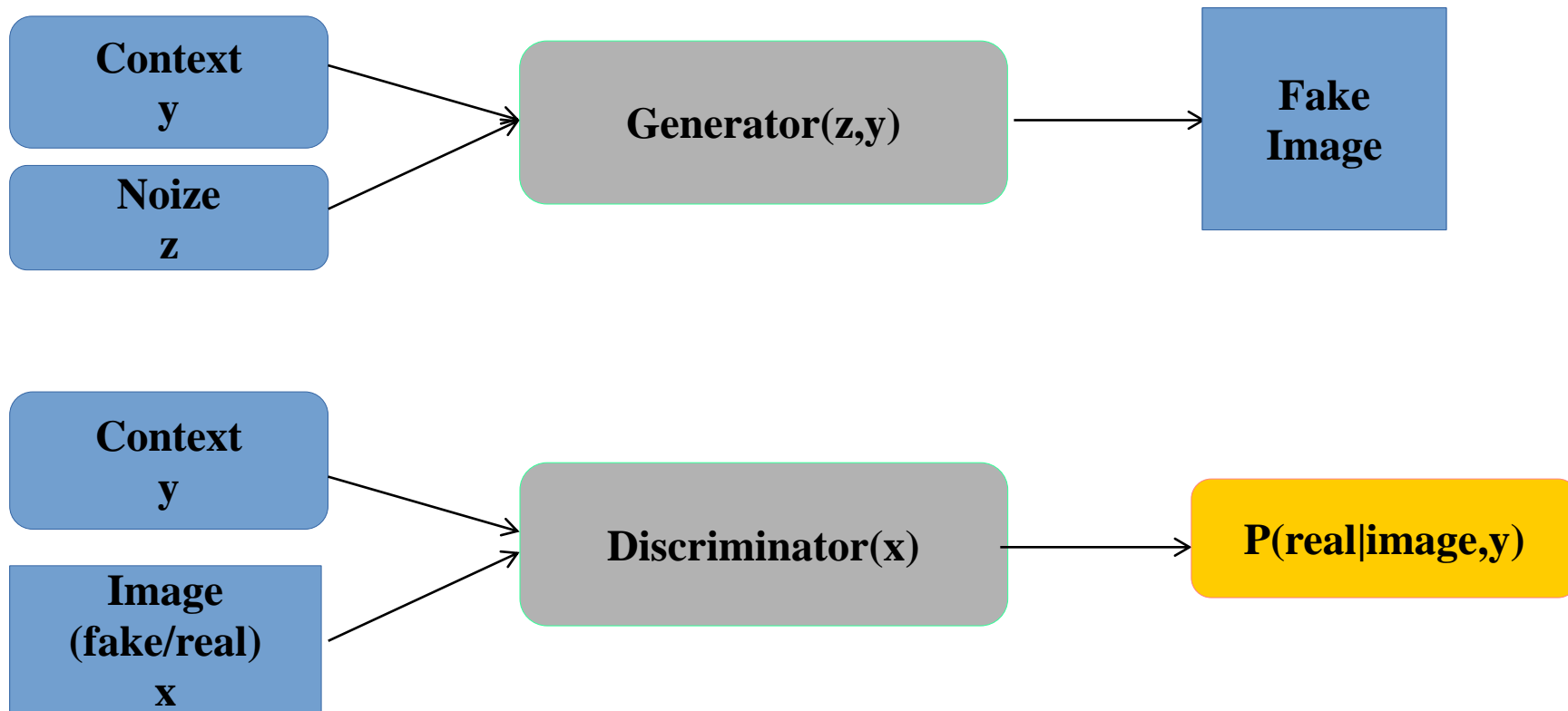
    sample a batch of noise  $\mathbf{z}$  and images  $\mathbf{x}$

    train step for discriminator

sample a batch of noise  $\mathbf{z}$  and images  $\mathbf{x}$

train step for generator

# Conditional Adversarial Networks



# Do we really learn a distribution?

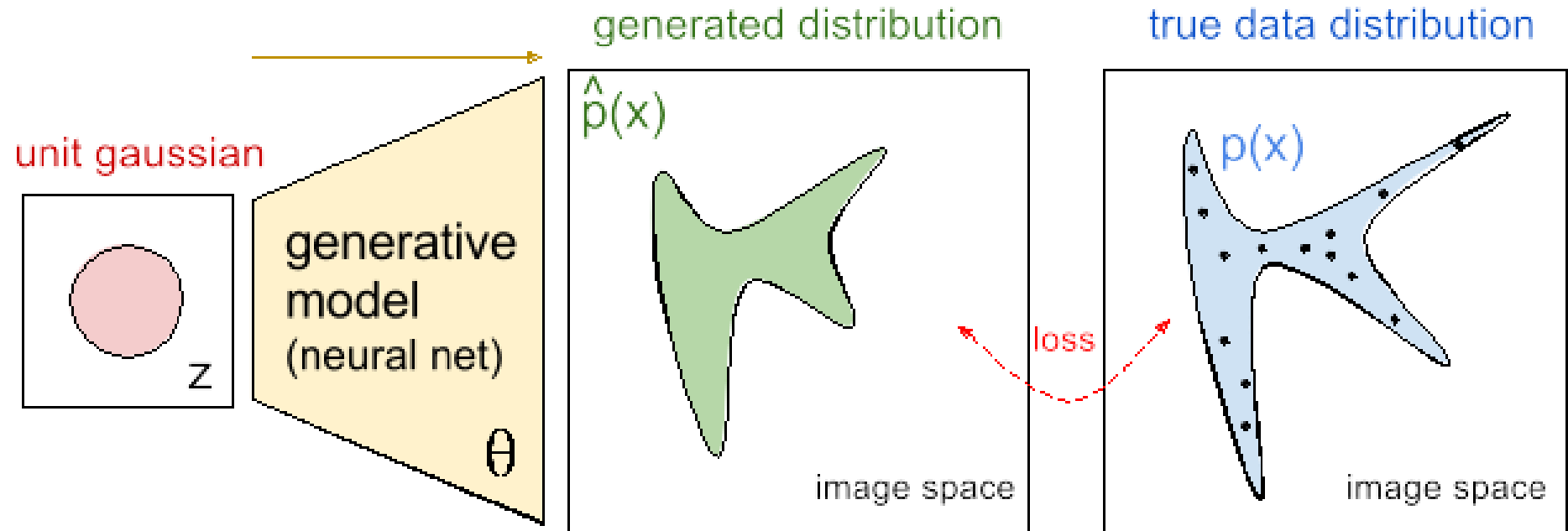


Image: <https://blog.openai.com/generative-models/>



How does one measure  
the distance between  
two distributions?

# Kullback–Leibler divergence

$$D_{KL}(P \parallel Q) = \int \log \left( \frac{P(x)}{Q(x)} \right) P(x) dx$$

Asymmetric

Possibly infinite

Has roots in information theory

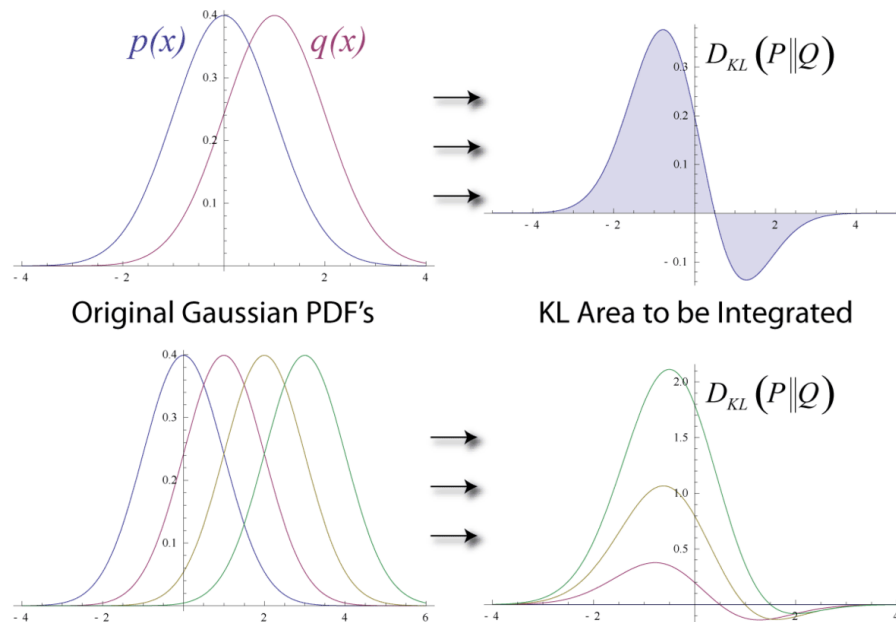


Image: Wikipedia

# Jensen-Shannon (JS) divergence

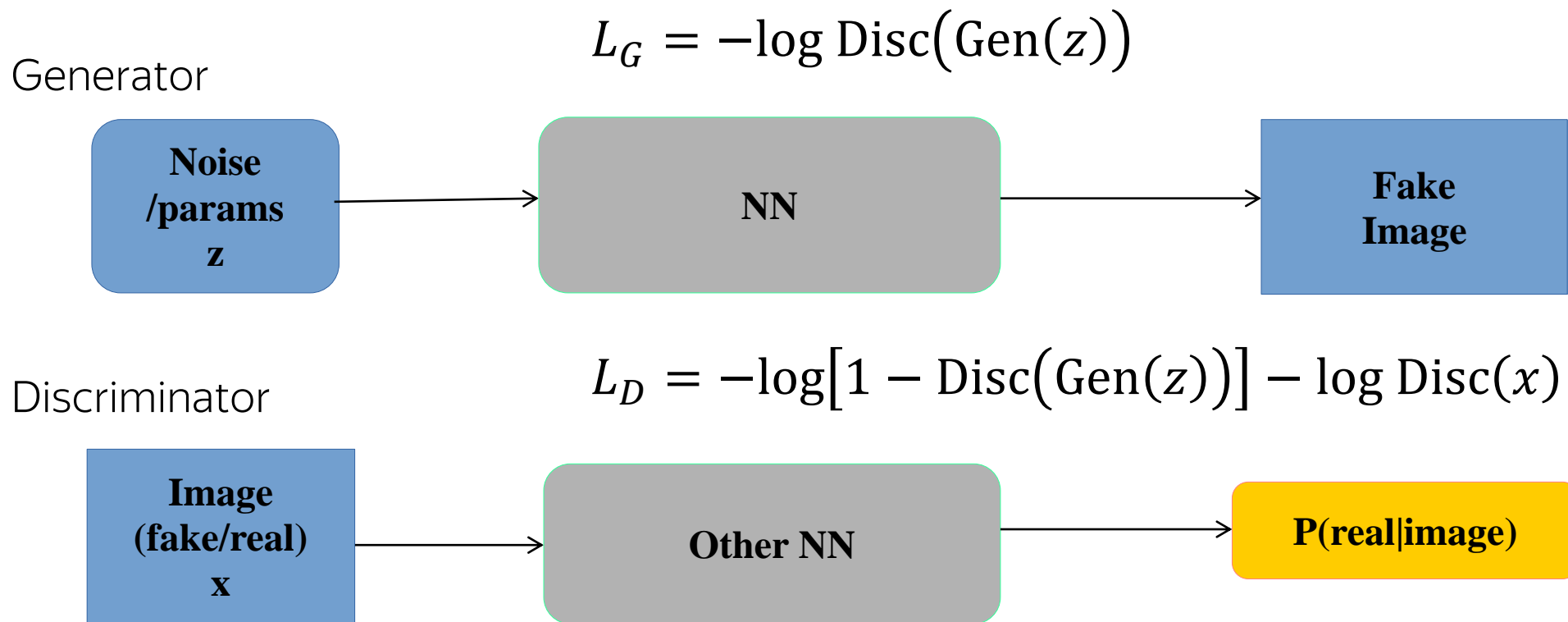
$$D_{\text{JS}}(P \parallel Q) = \frac{1}{2} D_{\text{KL}} \left( P \parallel \frac{P + Q}{2} \right) + \frac{1}{2} D_{\text{KL}} \left( Q \parallel \frac{P + Q}{2} \right)$$

Historically was the first used in GANs (if you hear just “GAN”, it’s likely optimizing JS)

In theory, it’s proven that a GAN converges. In practice however...

Some believe that one reason behind GANs’ success is switching the loss from KL (usually used in maximum-likelihood approach) to JS

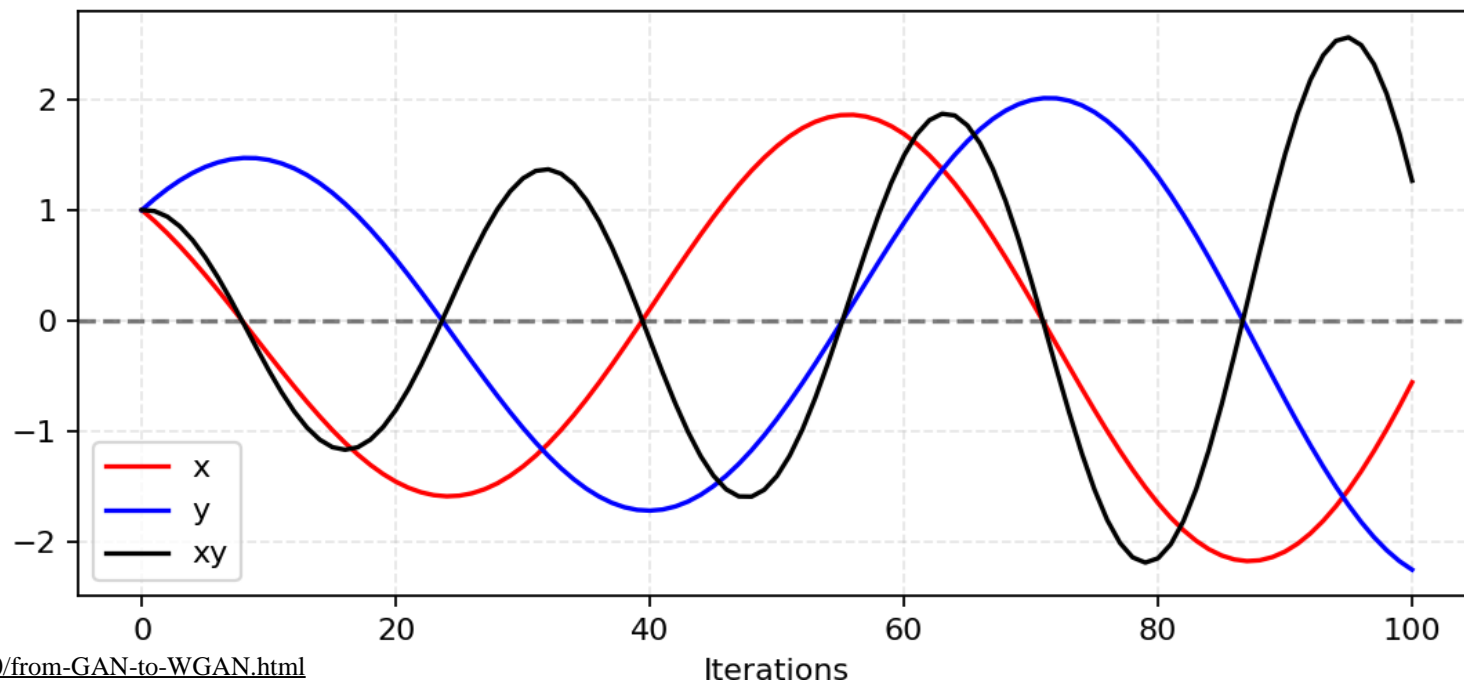
# Jensen-Shannon aka “ordinary” GAN



# Problem #1: Hard to achieve Nash equilibrium

Player I takes control of  $x$  to minimize  $f_1(x) = xy$

Player constantly updates  $y$  to minimize  $f_2(x) = -xy$



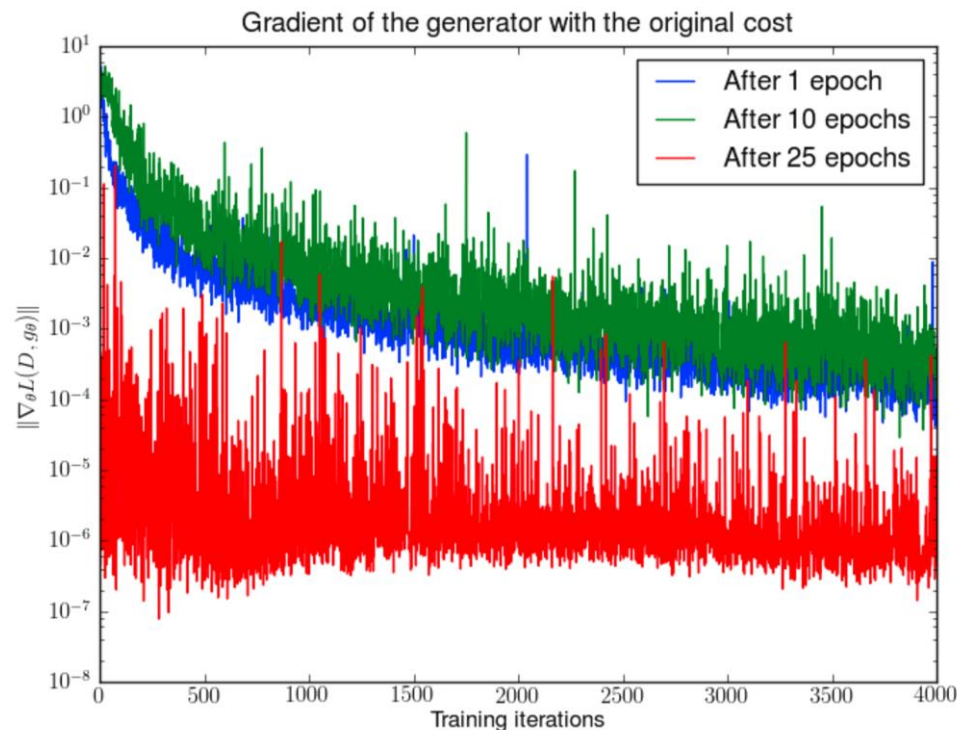


# Problem #2: Vanishing gradient

If discriminator is perfect,  
 $D(x) = 1, \forall x \in p_r$  and  $D(x) = 0, \forall x \in p_g$

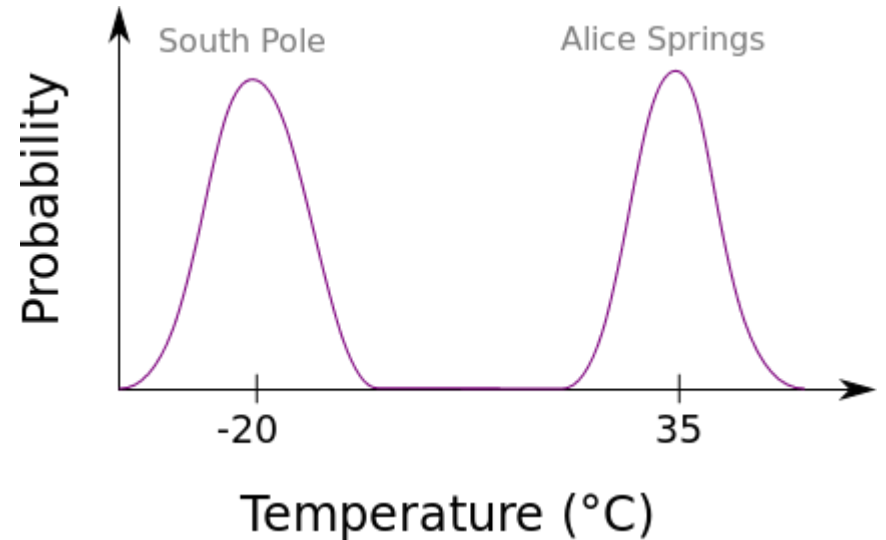
Loss function  $L = 0$

No gradient to update the loss  
during learning iterations



# Problem #3: Mode collapse

Generator can converge to cover just a part of the phase space



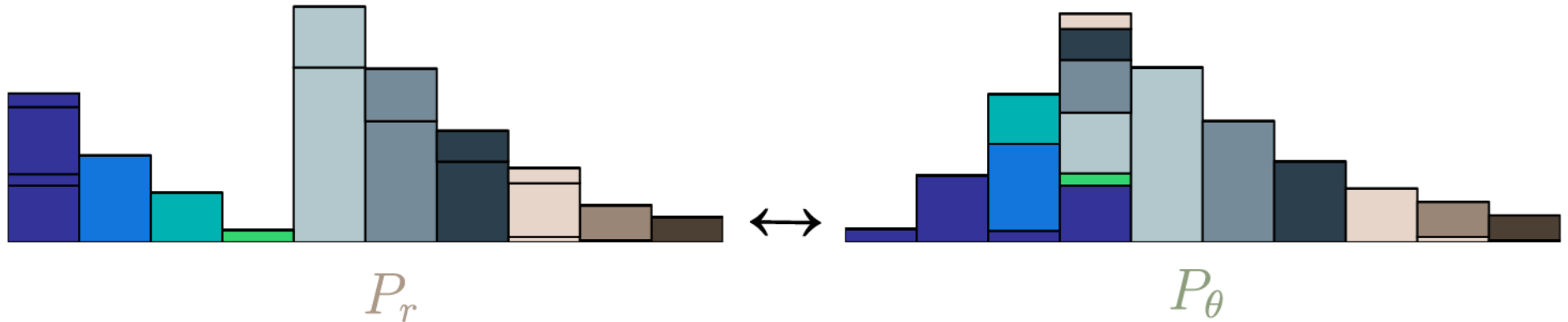
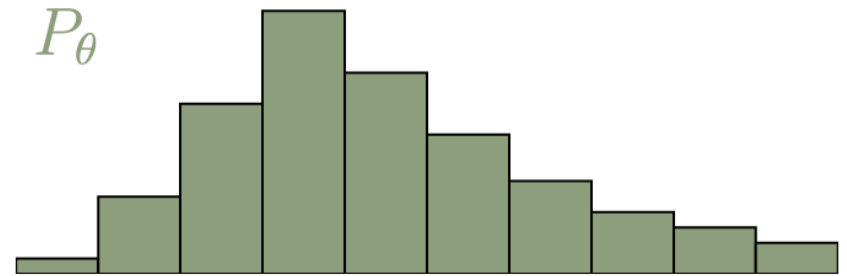
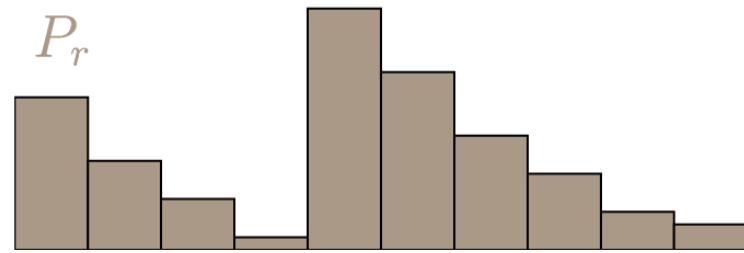
# Problem #4 No convergence metric

Theoretically, there is a Nash equilibrium

In practice it is usually not achieved

Hard to devise a stopping criteria

# Wasserstein distance aka Earth Mover's distance aka EMD

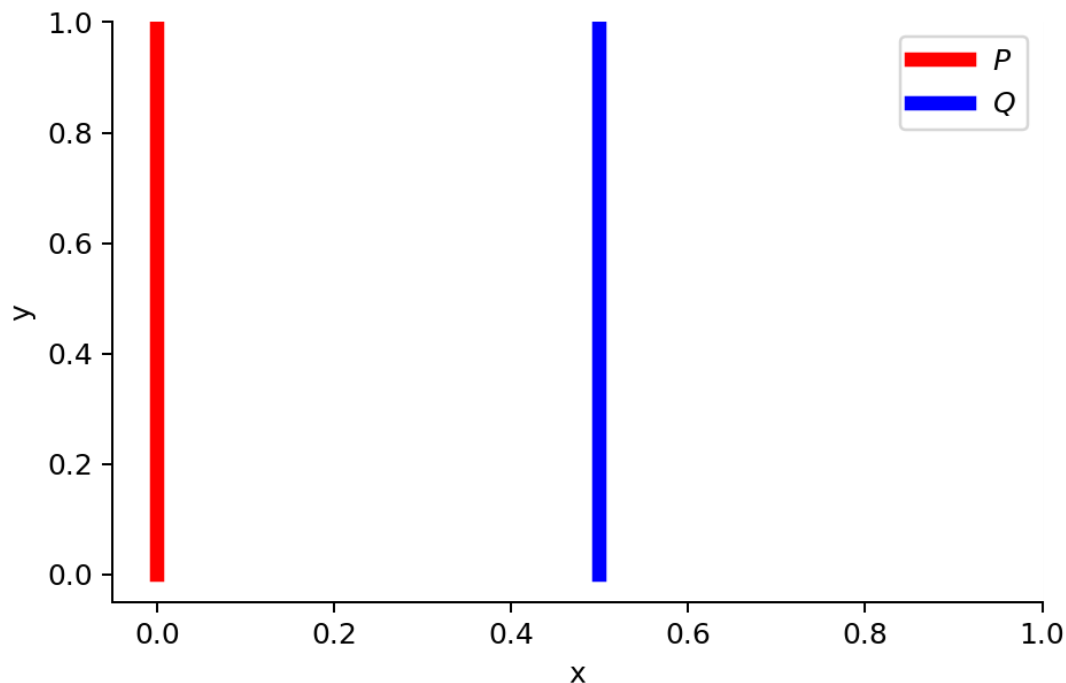


<https://vincentherrmann.github.io/blog/wasserstein/>

# Why Wasserstein is better than JS or KL divergence? #1

Smooth measure, even for disjoint distributions

JS & KL would have given us infinity



# WGAN formulation

Using the Kantorovich-Rubinstein duality,  
EMD can be formulated as following:

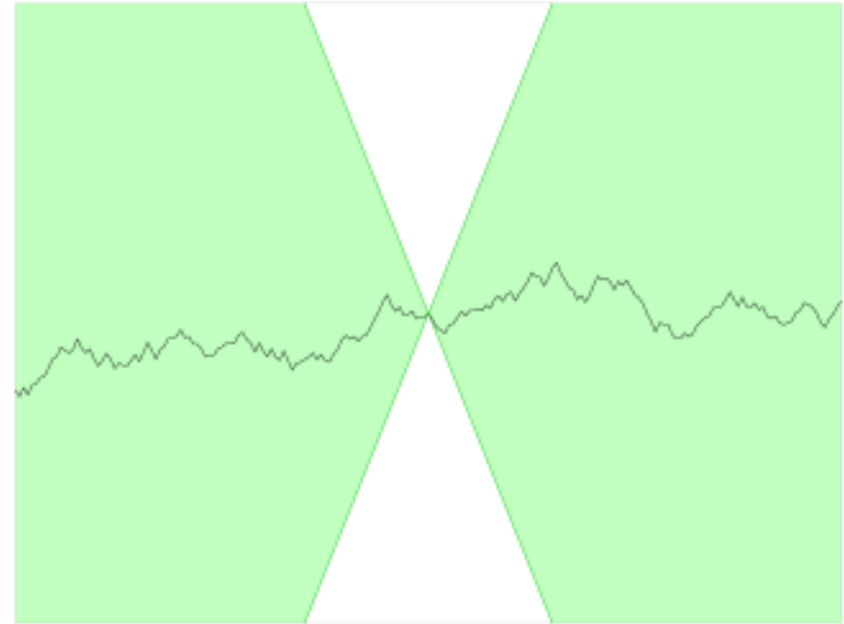
$$W(P_r, P_q) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim P_r}[f(x)] - \mathbb{E}_{x \sim P_q}[f(x)]$$

The catch?  $f$  must be 1-Lipschitz

# K-Lipschitz functions

For a Lipschitz continuous function, there is a double cone (shown in white) whose vertex can be translated along the graph, so that the graph always remains entirely outside the cone

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



# Lipschitz continuity and NN

Naive way: clip weights. Bad for convergence.

Gradient penalty. Optimal critic has gradient 1 almost everywhere, so add a term to the loss

$$-E_x[(\|\nabla_x D(x)\|_2 - 1)^2]$$

This is quite a hot area, so expect more



# WGAN-GP

Generator

$$L_G = -\mathbb{E}_z \text{Disc}(\text{Gen}(z))$$

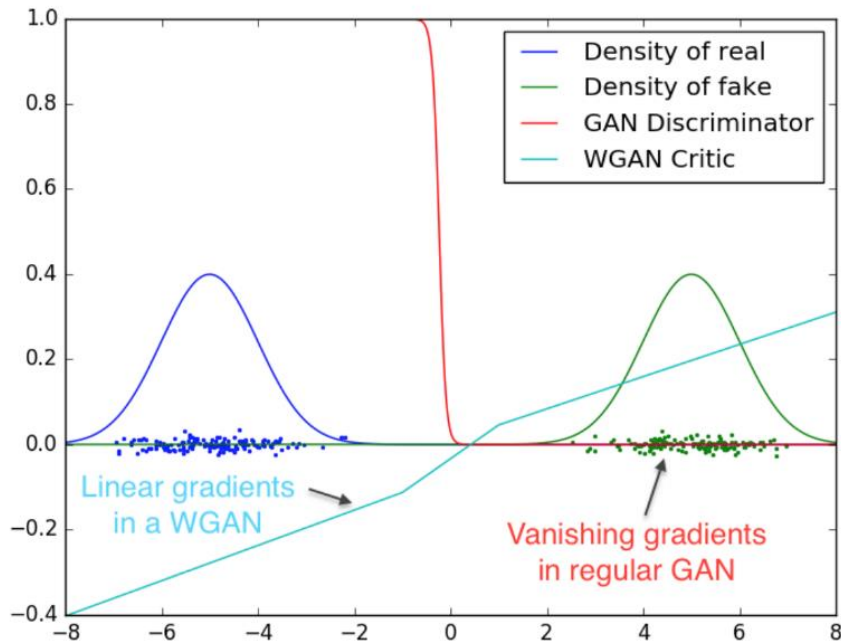


Discr.  $L_D = \mathbb{E}_z \text{Disc}(\text{Gen}(z)) - \mathbb{E}_x \text{Disc}(x) - \lambda \mathbb{E}_x [(\|\nabla_x \text{Disc}(x)\|_2 - 1)^2]$



# Why Wasserstein might be better than JS or KL divergence? #2

No vanishing gradients



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

# Practical way to enforce gradient penalty

Enforcing the unit gradient norm constraint everywhere is intractable

The authors propose choosing random points on lines connecting generated and real examples

# Why Wasserstein might be better than JS or KL divergence? #3

WGAN critic can be trained until convergence

WGAN critic loss is a meaningful estimate of EMD between the real and generated data

# However

## WGAN:



- Doesn't go well batch norm

- Has slower convergence

- Doesn't work with some (e. g.) ELU activations

- Less papers exploring architectures and heuristics

# Note: there are other metrics

| Name  | $D_f(P  Q)$  |
|---|--|
| Total variation                                   | $\frac{1}{2} \int  p(x) - q(x)  \, dx$   |
| Kullback-Leibler                                  | $\int p(x) \log \frac{p(x)}{q(x)} \, dx$   |
| Reverse Kullback-Leibler                          | $\int q(x) \log \frac{q(x)}{p(x)} \, dx$   |
| Pearson $\chi^2$                                  | $\int \frac{(q(x)-p(x))^2}{p(x)} \, dx$  |
| Neyman $\chi^2$                                   | $\int \frac{(p(x)-q(x))^2}{q(x)} \, dx$  |
| Squared Hellinger                                 | $\int \left( \sqrt{p(x)} - \sqrt{q(x)} \right)^2 \, dx$  |
| Jeffrey   | $\int (p(x) - q(x)) \log \left( \frac{p(x)}{q(x)} \right) \, dx$   |
| Jensen-Shannon                                    | $\frac{1}{2} \int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} \, dx$   |
| Jensen-Shannon-weighted                           | $\pi \int p(x) \log \frac{p(x)}{\pi p(x)+(1-\pi)q(x)} + (1-\pi) \int q(x) \log \frac{q(x)}{\pi p(x)+(1-\pi)q(x)} \, dx$                      |
| GAN   | $\int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} \, dx - \log(4)$   |
| $\alpha$ -divergence ( $\alpha \notin \{0, 1\}$ ) | $\frac{1}{\alpha(\alpha-1)} \int \left( p(x) \left[ \left( \frac{q(x)}{p(x)} \right)^\alpha - 1 \right] - \alpha(q(x) - p(x)) \right) \, dx$ |

f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization

# Training a GAN

How many iterations of generator and discriminator training?

For JS usually around 1:1

For WGAN, the original paper used 5 discriminator per 1 generator – but you can make it as high as you want

# Heuristics for Convolutional GANs (DCGAN)

Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator)

Use batchnorm in both the generator and the discriminator

Remove fully connected hidden layers for deeper architectures

Use ReLU activation in generator for all layers except for the output, which uses Tanh

Use LeakyReLU activation in the discriminator for all layers



Intermission: Measuring  
datasets similarity



# Measuring datasets similarity

Sometimes one just has several complex datasets and needs a to measure a meaningful distance between them

- FastFoo generator vs. data
- MC vs data

# Measuring datasets similarity: classifier AUC

Pick your favorite classifier (NN, gradient boosting, whatever)

Use cross-validation to check whether it is able to distinguish the datasets

Pro: easy to understand AUC

Con: no real theoretical guarantees

Con: answer depends on the classifier

# Measuring datasets similarity: EMD

Train a Wasserstein discriminator

Pro: EMD has a well defined meaning

Con: the number is not exactly EMD, there is uncertainty due to the optimization procedure and the way Lipschitz continuity is enforced

GANs @ HEP



# Classic cWGAN for Cherenkov detector high-level observables

**Track kinematics**  
**Event occupancy proxies**

**Noise**

concatenation

**Generator**  
**Fully-connected NN**

**Track DLLs**

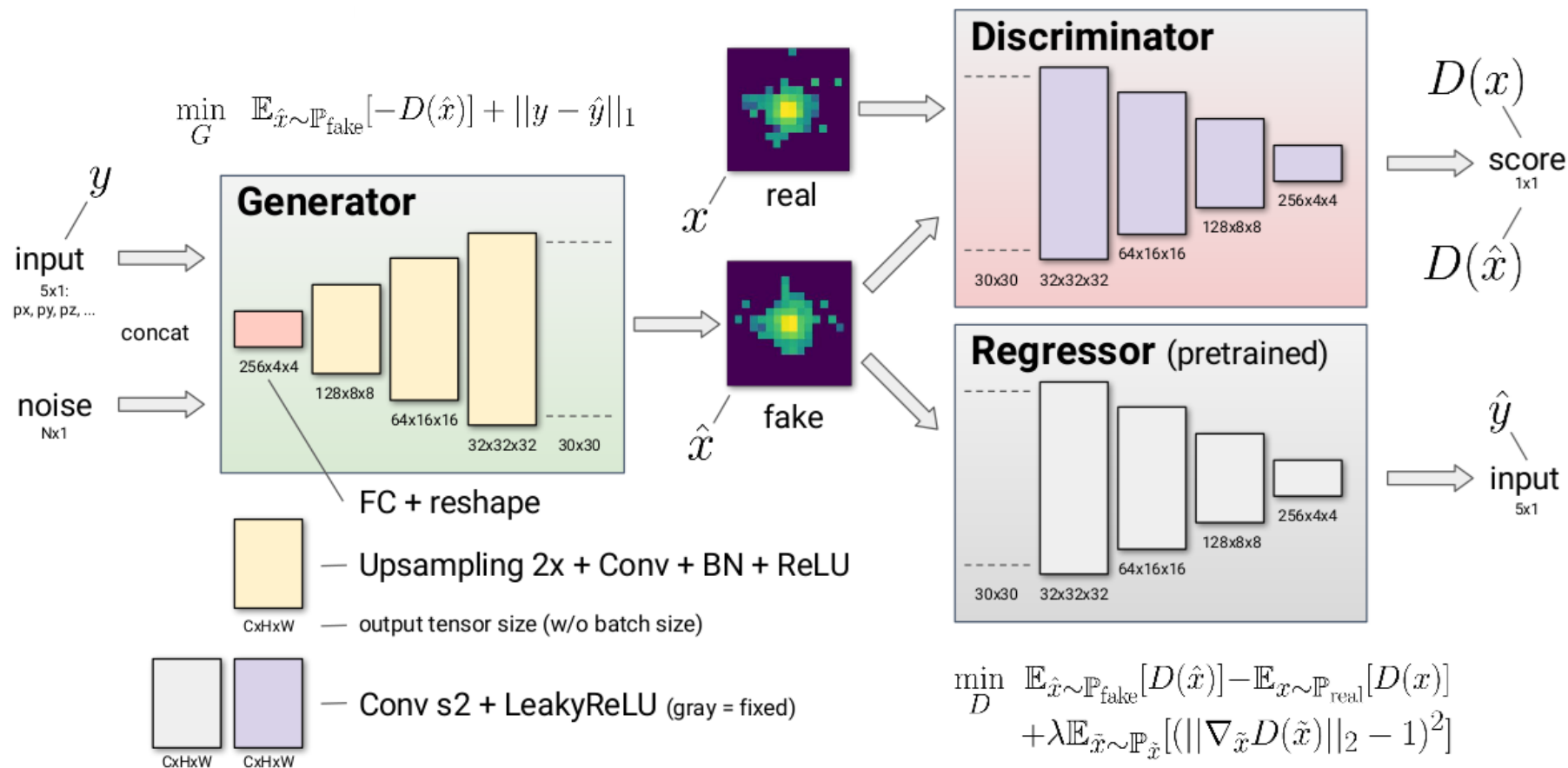
**Track kinematics**  
**Event occupancy proxies**

**Track DLLs**  
**real/generated**

concatenation

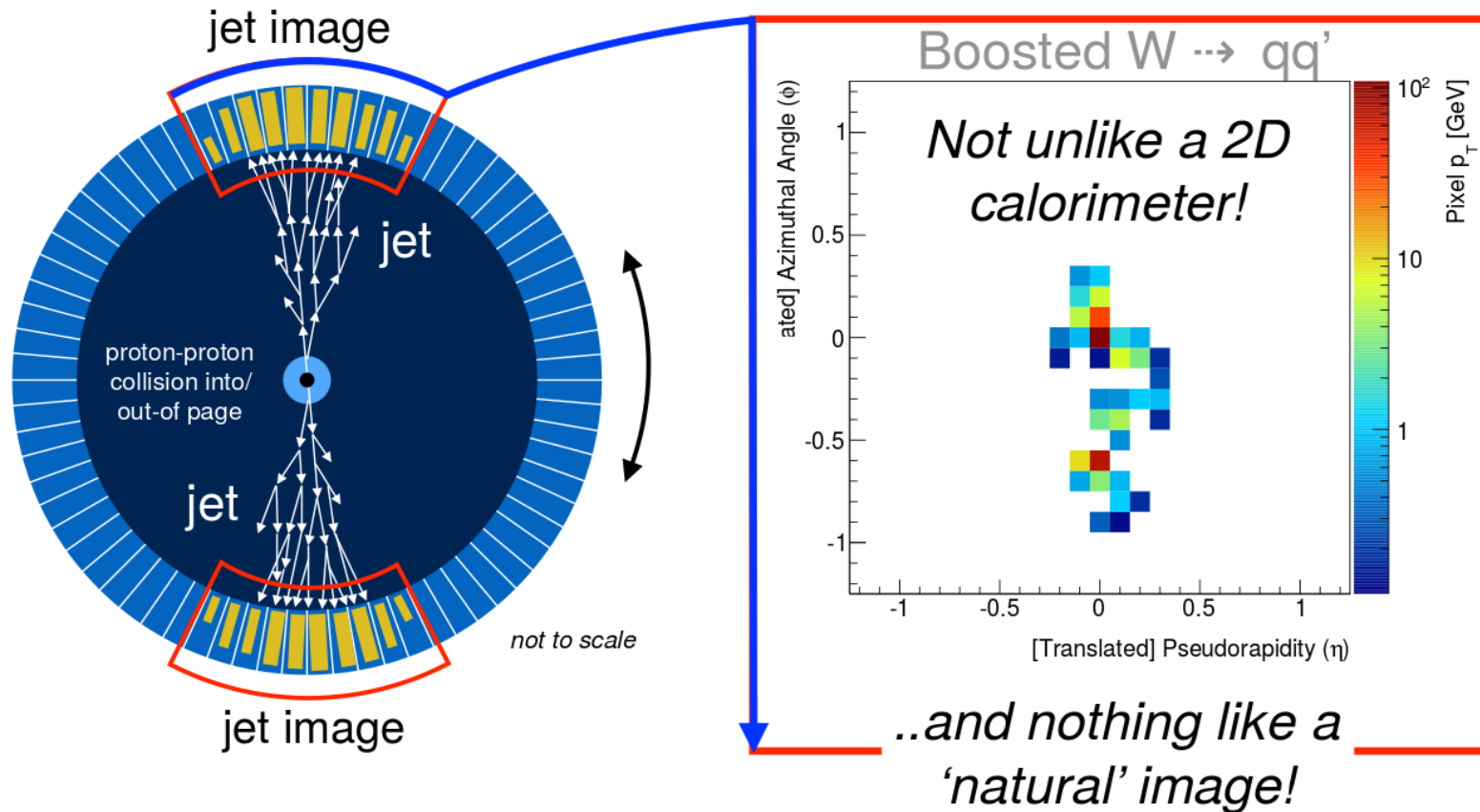
**Critic aka discriminator**  
**Fully-connected NN**

# Calorimeter response generation



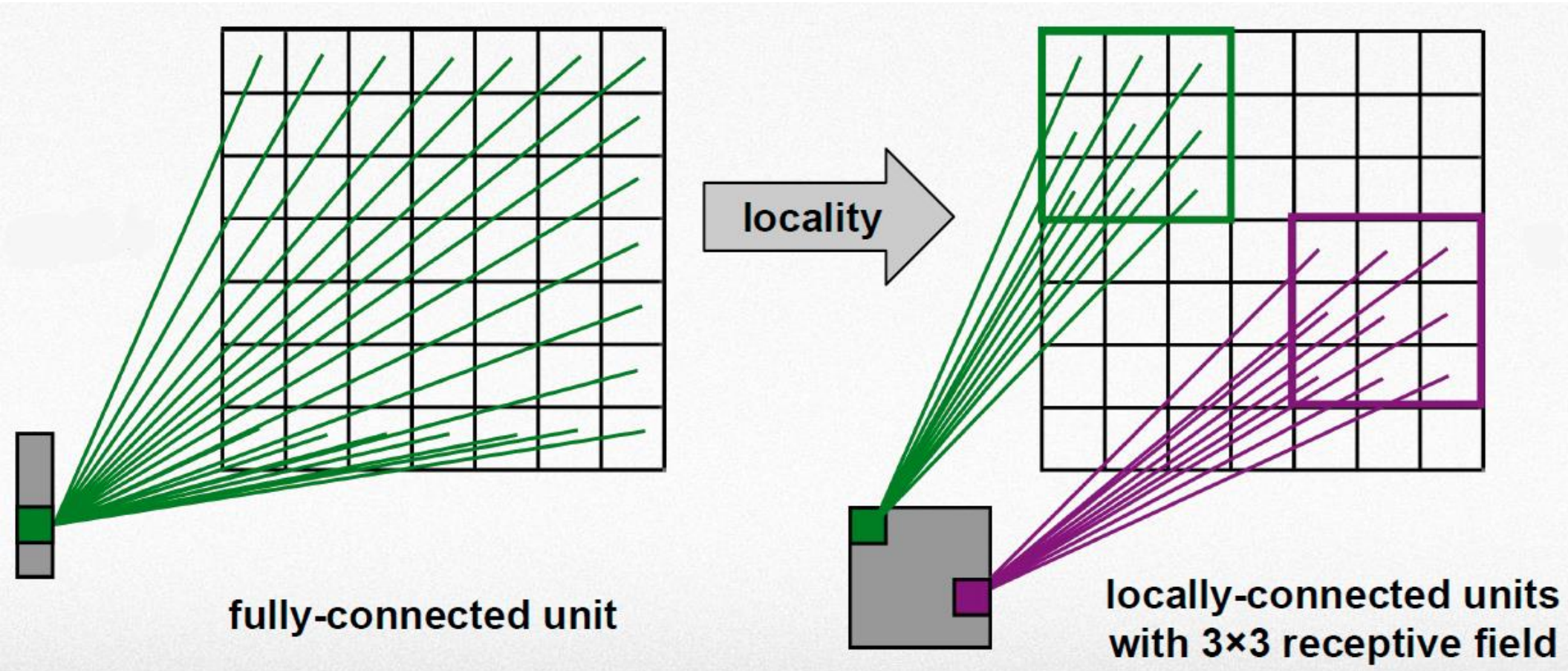
# The Jet Image

Jet Image: A two-dimensional fixed representation of the radiation pattern inside a jet





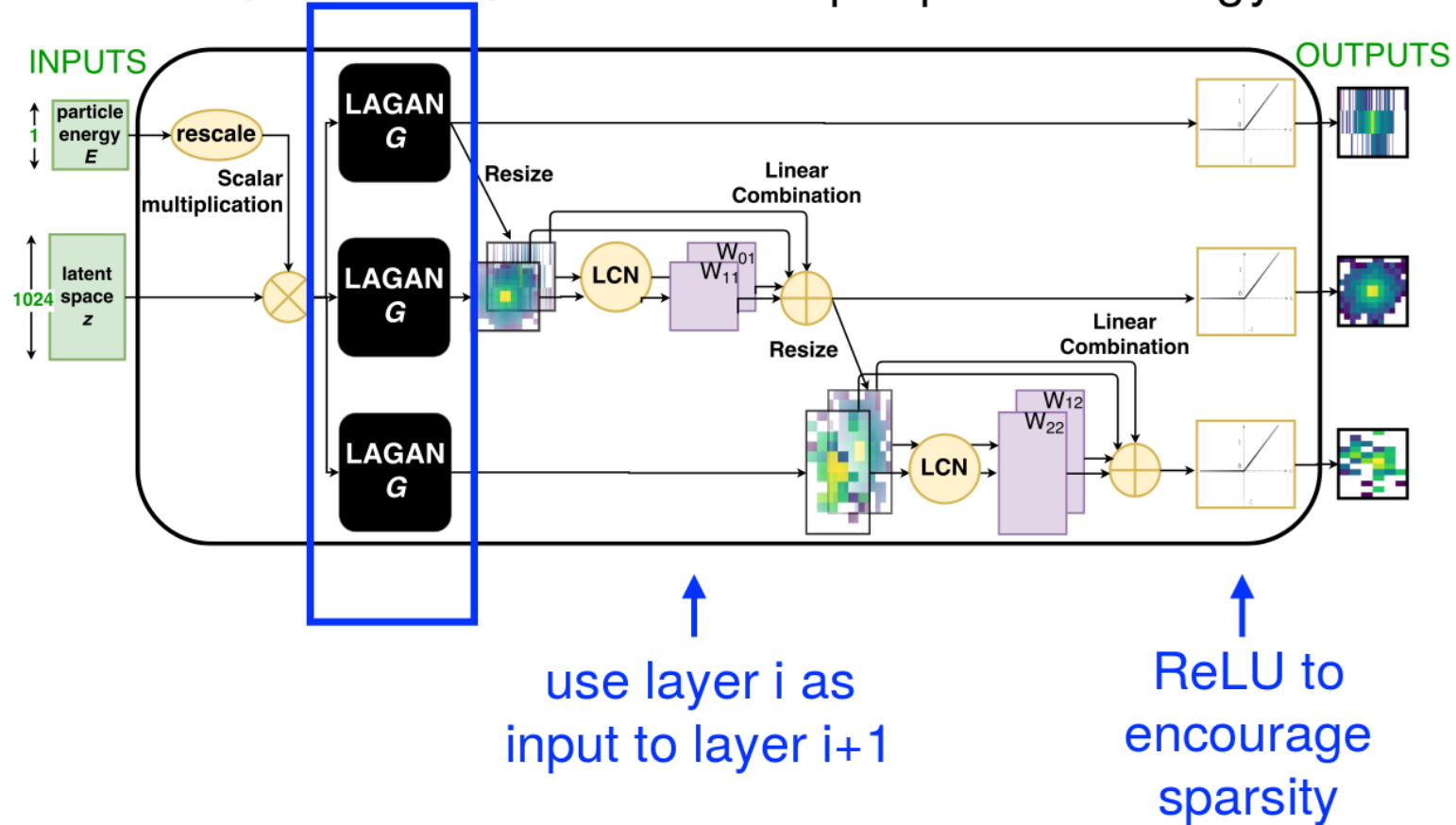
# Recap: Locally Connected Layer



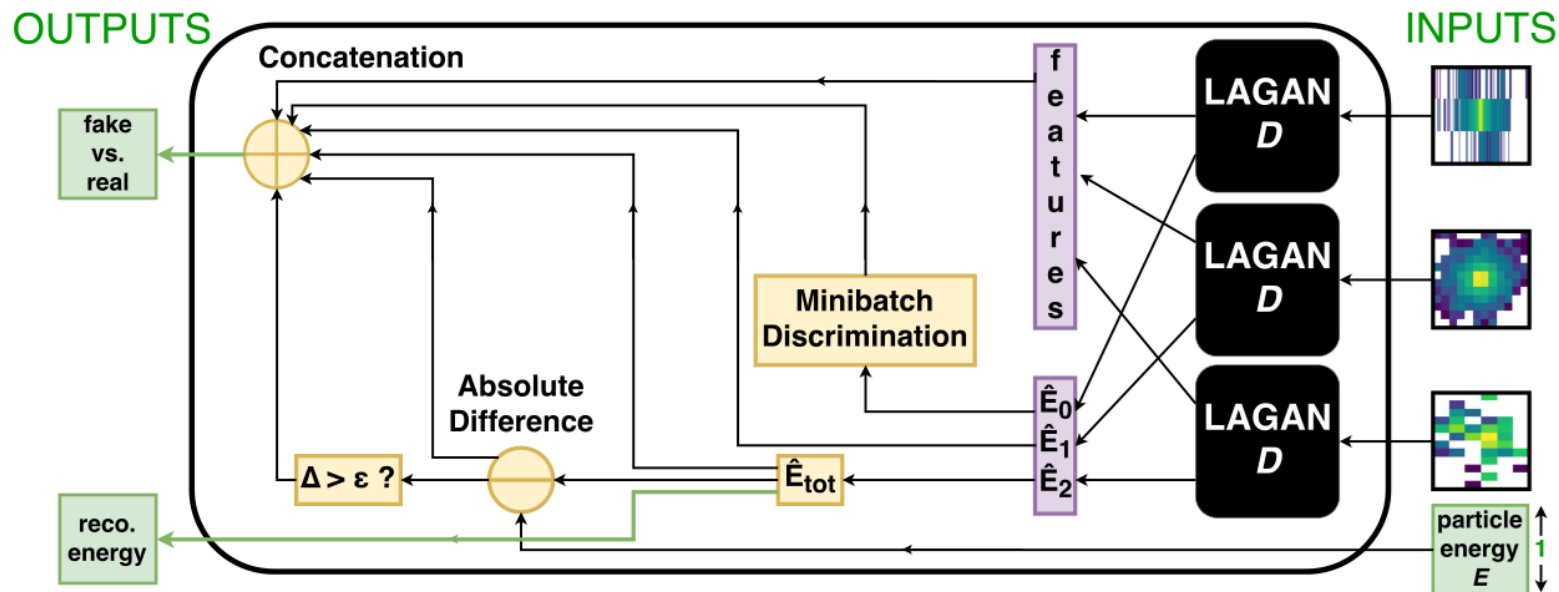
# Generator Network for CaloGAN

One 'jet image'  
per calo layer

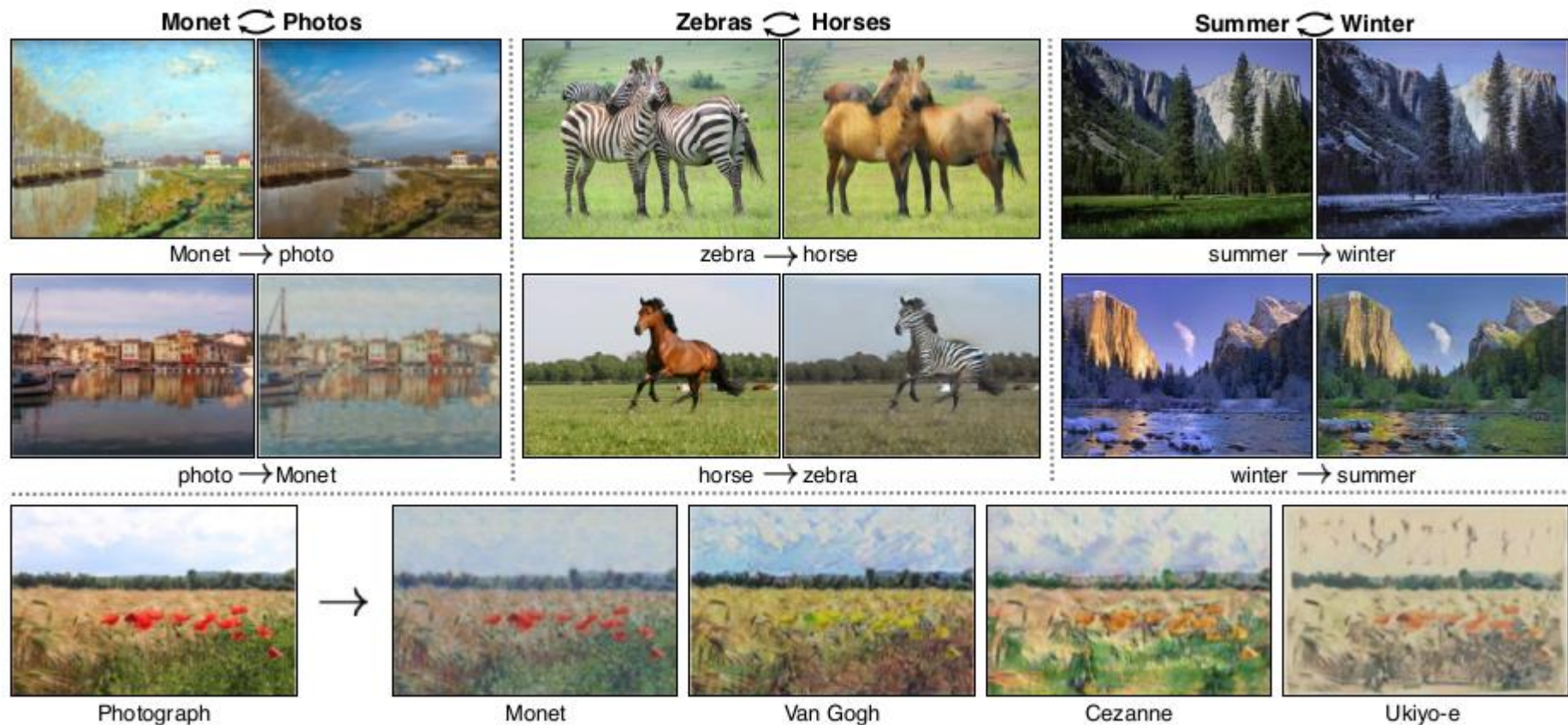
One network per particle type;  
input particle energy



# Discriminator Network for CaloGAN



# Cycle GAN for unpaired images



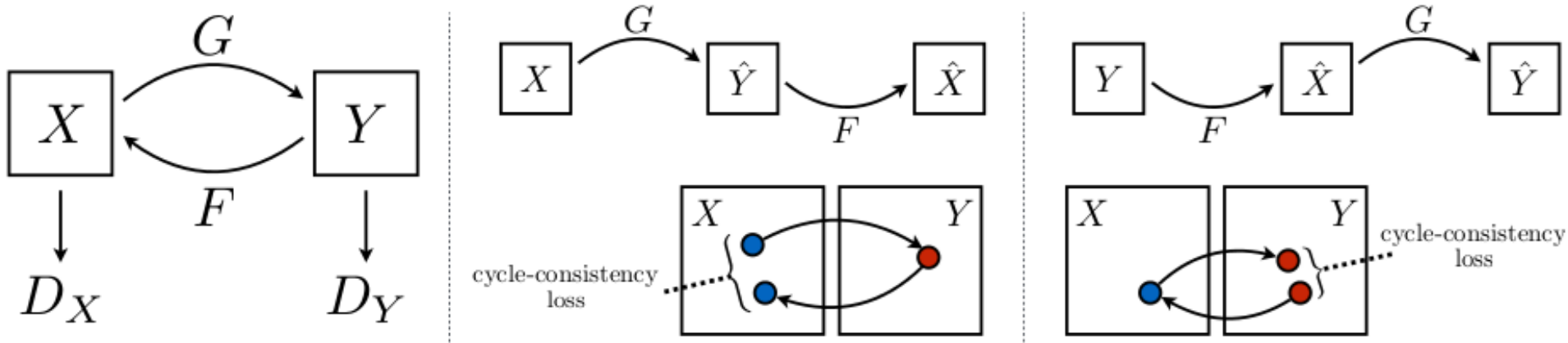
<https://arxiv.org/abs/1703.10593>



# Cycle GAN for unpaired images

Idea: if we don't have image pairs, train two conditional generators  $G(z, y) \rightarrow x, F(g, x) \rightarrow y$

- use non-conditional  $D(x), D(y)$
- make sure  $|F(G(x)) - x| \rightarrow \min$



Data  $\leftrightarrow$  MC ?



Image: Indiana Jones and the Last Crusade

PartyGAN or Maxim's  
quest for the Holy Grail



# Feel free to drop a line

Nikita Kazeev

HSE, Rome Sapienza, YSDA, trace amounts of  
Yandex proper



kazeevn@yandex-team.ru



telegram.me/kazeevn



# Bonus



# Art style transfer



+



=



| Ideas?

# Art style transfer

Formulate and optimize texture loss

$$L = \|\text{Texture}(x_{ref}) - \text{Texture}(x_{cand})\| + \|\text{Content}(x_{orig}) - \text{Content}(x_{cand})\|$$

