SCHOOL OF DATA ANALYSIS

LAMBDA

# GANs, GANs everywhere

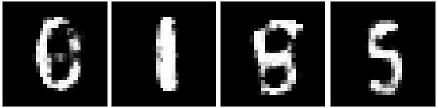
## particularly, in High Energy Physics

Maxim Borisyak

Yandex, NRU Higher School of Economics

# Generative

# Generative models

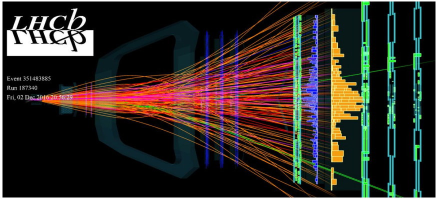


Given samples of a random variable $X$ find $X'$ such as:

$$P_X \approx P_{X'}$$

Approaches:

- probability density function;
- sampling procedure.



# Generator vs probability function

- Learning directly a probability density function $P_{X'}$ is hard:
  - normalization is the main issue;
  - sampling might be computationally costly (usually, long MC);
  - $P_{X'}$ is usually heavily restricted (e.g. RBM).

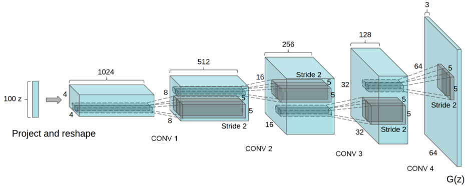
- Learning a **generator** is easier:

$$X' = G(Z)$$

where:

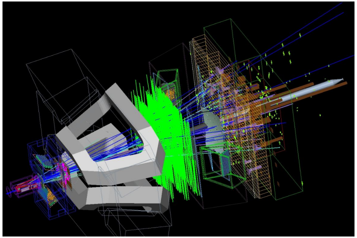
- $G$ - a **parametrized** deterministic function;
- $Z$ - predetermined and easy to sample.

# Generator




Generator might be any program:

- a neural network (traditionally);
- a Monte-Carlo simulation.



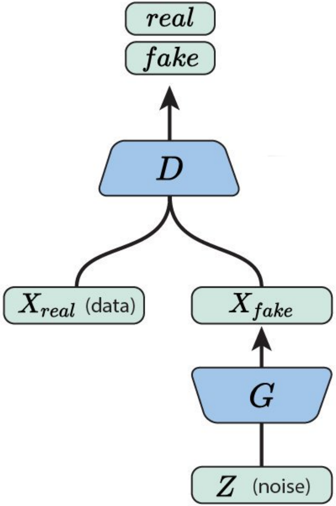
# Generative Adversarial

# Adversarial training

Generator is trained to maximize goodness of produced samples.

GAN defines goodness of a generator via a classifier $D$:

- learns to discriminate $X$ against $X'$;
- if quality is close to a random guess: $X'$ is similar to $X$;
- if quality is high: $G$ should be improved.



# Discriminator

- usually called **adversary** or **critic**;
- traditionally, also a neural network;
- discriminator defines goodness of generated samples:
  - rich set of methods for classification;
  - easy to identify important properties of good generator and use inside discriminator;
  - produces interpretable quality metric.

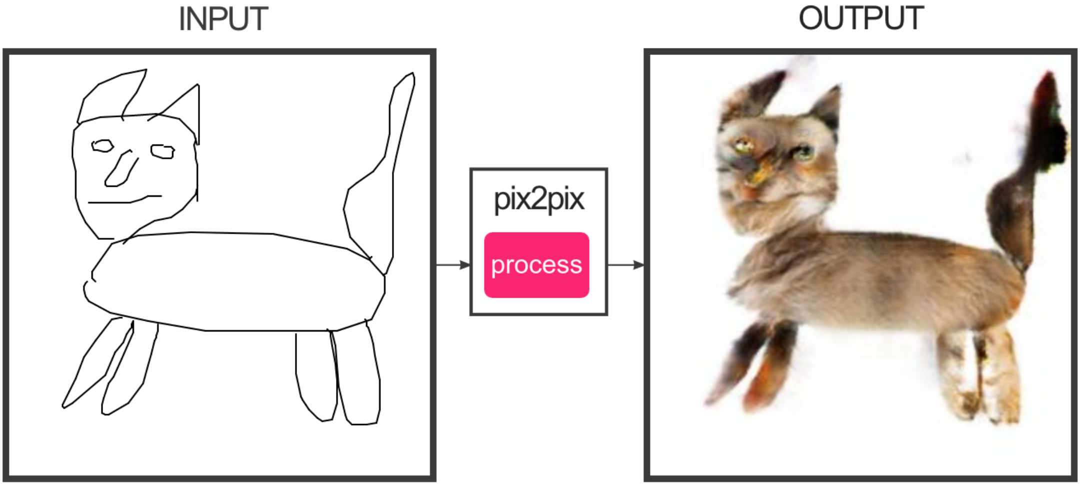
# Generative Adversarial Networks

# GANs

Generative Adversarial Networks:

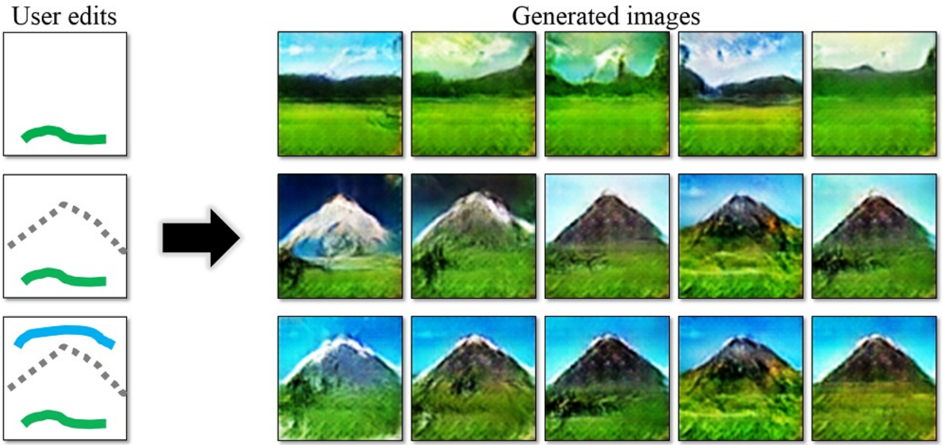
- introduced recently;
- a lot of promising results and development;
- adoptable:
  - conditional GANs;
  - GANs as auxiliary loss;

# GANs in the wild




Source: https://affinelayer.com/pixsrv/

# GANs in the wild




Source: https://github.com/junyanz/iGAN

# Summary

Generative Adversarial Networks:

- generator training;
- performance of **trained** classifier as quality measure;
- easily adoptable for problems beyond pure generation.

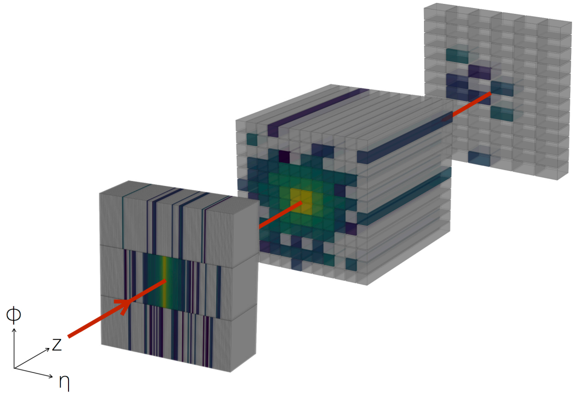
# GANs in HEP

# GANs IN High Energy Physics

- CaloGAN:
  - fast simulation;
- PartyGAN:
  - learning (almost) black-box physical process;
- MC tuning:
  - selecting best parameters for simulation to match data.

# CaloGAN

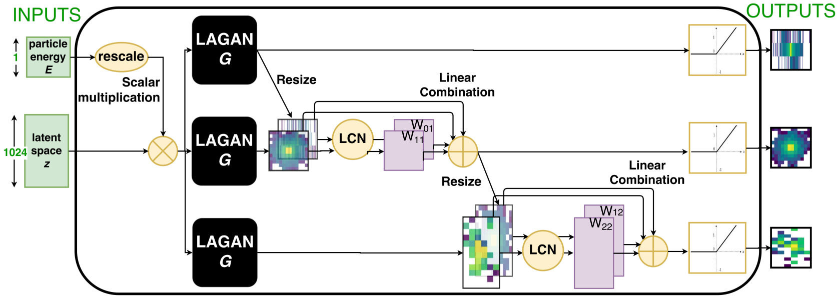
Samples EM calorimeter showers.




Source: https://github.com/hep-lbdl/CaloGAN

# CaloGAN

- direct sampling from the response space;
- no intermediate steps involved;
- significant speedup.

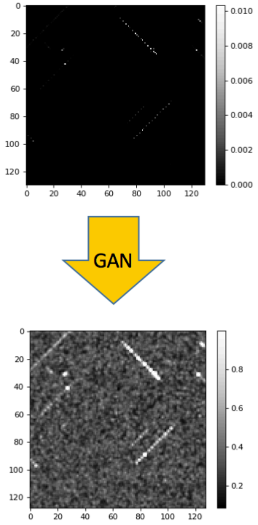



Source: https://github.com/hep-lbdl/CaloGAN

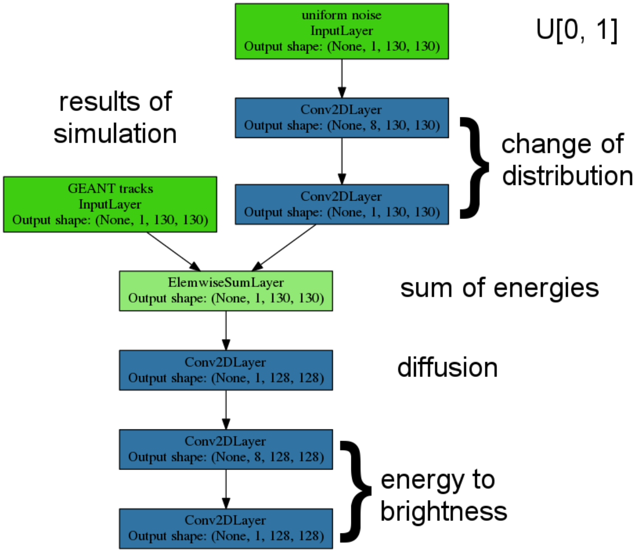
# PartyGAN

Simulation of cosmic rays interaction with a CMOS sensor:

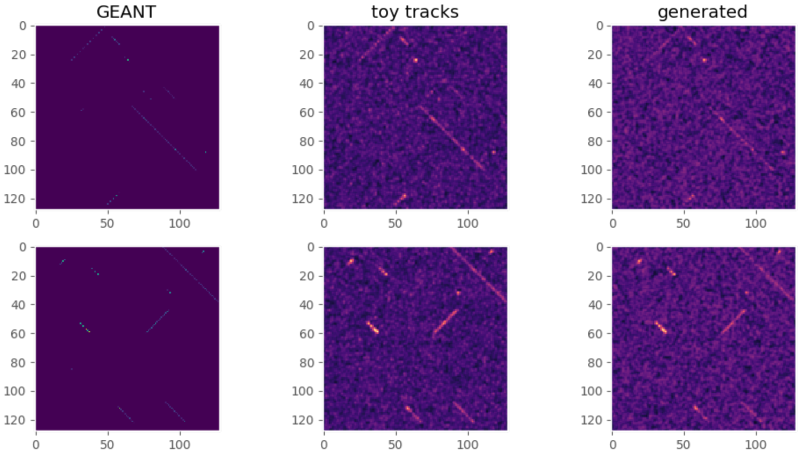
- partially simulated by GEANT, **without readout;**
- generator transforms GEANT output into realistic images:
  - generator is to learn readout-process;
- matching to the real data.



# PartyGAN

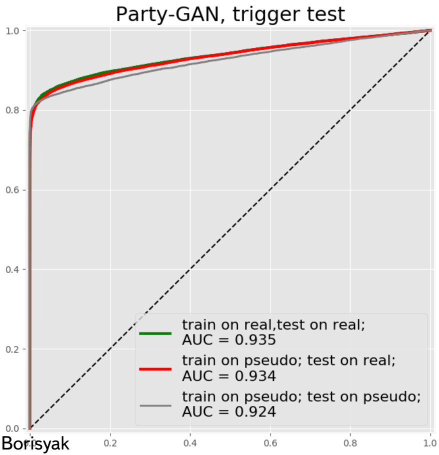

uniform noise
InputLayer
Output shape: (None, 1, 130, 130)
U[0, 1]
results of simulation
Conv2DLayer
Output shape: (None, 8, 130, 130)
change of distribution
GEANT tracks
InputLayer
Output shape: (None, 1, 130, 130)
Conv2DLayer
Output shape: (None, 1, 130, 130)
ElemwiseSumLayer
Output shape: (None, 1, 130, 130)
sum of energies
Conv2DLayer
Output shape: (None, 1, 128, 128)
diffusion
Conv2DLayer
Output shape: (None, 8, 128, 128)
Conv2DLayer
Output shape: (None, 1, 128, 128)
energy to brightness

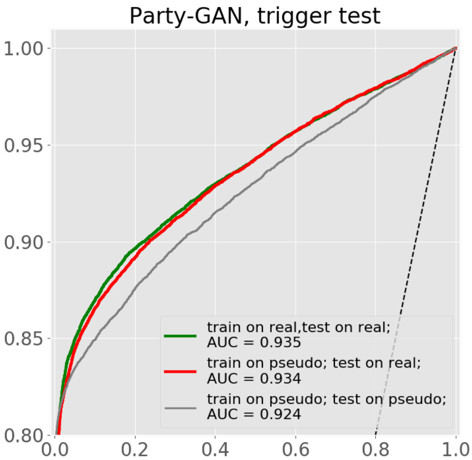
# PartyGAN: toy readout model



# PartyGAN: quality test

Trigger is training on $D_1$, evaluated on $D_2$.



# MC tuning

A MC simulation as generator:

- parameter tuning to match real data;
- non-differentiable generator;
- gradient-free optimization methods.

# Summary

# Summary

- Generative Adversarial Networks is powerful tool for modeling distributions;
- easily adoptable for various settings, even beyond pure generative tasks;
- GANs in HEP:
  - fast simulation: e.g. CaloGAN:
    - possible applications: RICH-GAN, VeLo-GAN;
  - learning behavior of an almost black-box system: e.g. PartyGAN;
  - tuning Monte-Carlo parameters: Adversarial Optimization.

# References, GAN


- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2014. Generative adversarial nets. In Advances in neural information processing systems (pp. 2672-2680).
- Isola, P., Zhu, J.Y., Zhou, T. and Efros, A.A., 2016. Image-to-image translation with conditional adversarial networks. arXiv preprint arXiv:1611.07004.

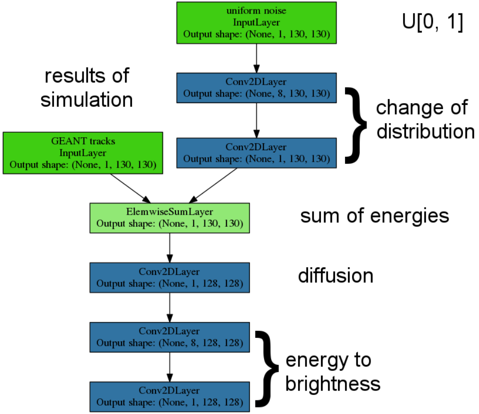
# References, GAN in HEP


- Paganini, M., de Oliveira, L. and Nachman, B., 2017. CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks. arXiv preprint arXiv:1705.02355.
- PartyGAN at ACAT-2017 `https://indico.cern.ch/event/567550/contributions/2629720/`
- Louppe, G. and Cranmer, K., 2017. Adversarial Variational Optimization of Non-Differentiable Simulators. arXiv preprint arXiv:1707.07113.

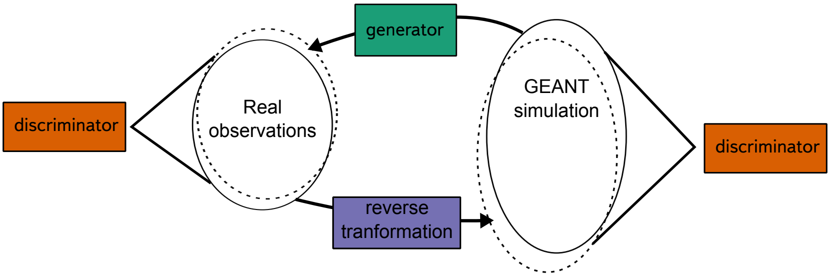
# Backup

# PartyGAN details

- GEANT output as generator's input;
- real data as reference data;
- event rate adjustment;
- Cycle-GAN to enforce reversibility of the generator;
- restricted $(3 \times 3)$ receptive field of the generator.



# Party-GAN details



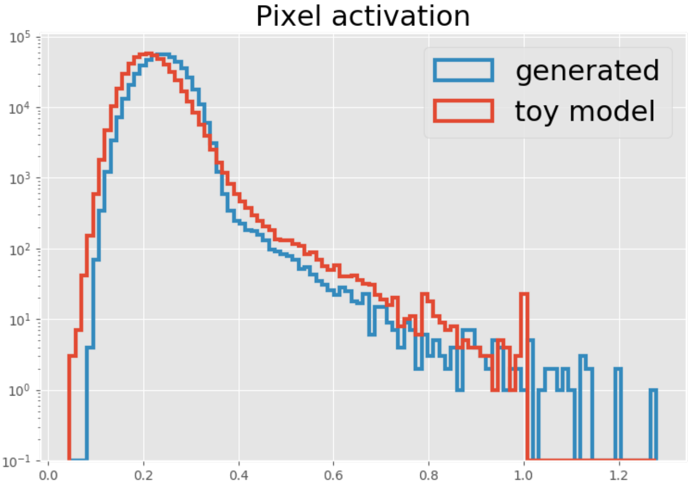
# Different flavors of GANs, part I

- GAN as auxiliary loss:
  - restricts e.g. regression to realistic results;
- Conditional GAN:
  - replaces $Z \sim \mathcal{N}^m(0, 1)$ by another dataset;
- Cycle-GAN:
  - a conditional GAN that learns reversible generator;
- staked GANs, ensembles of GANs, ...

# Different flavors of GANs, part II

- Classical GAN:
  - cross-entropy discriminator: proxy to KL distance;
  - vanishing/exploding gradients.
- EB-GAN:
  - energy-based discriminator: proxy to total variation distance;
  - rarely vanishing, smooth gradients;
- Wasserstein-GAN:
  - critic, a $L_1$ function: proxy for earth-moving distance;
  - never vanishing, smooth gradients;
  - difficulties training: keeping critic in $L_1$.

# PartyGAN



Pixel activation
generated
toy model
$10^5$
$10^4$
$10^3$
$10^2$
$10^1$
$10^0$
$10^{-1}$
0.0
0.2
0.4
0.6
0.8
1.0
1.2