

Generative Modelling for HEP

Deep-faking a high-energy physics detector

Quarks online workshop “Advanced Computing in Particle Physics”

June 8-9, 2021

Artem Maevskiy

National Research University Higher School of Economics



LAMBDA • HSE

June 09, 2021

This X does not exist



This Person Does Not Exist

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

Created by Phillip Wang.



This Cat Does Not Exist

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

Created by Ryan Hoover.



This Rental Does Not Exist

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

Created by Christopher Schmidt.

<https://thisxdoesnotexist.com/>

Style transfer



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

Generative models progress



2014



2015



2016



2017

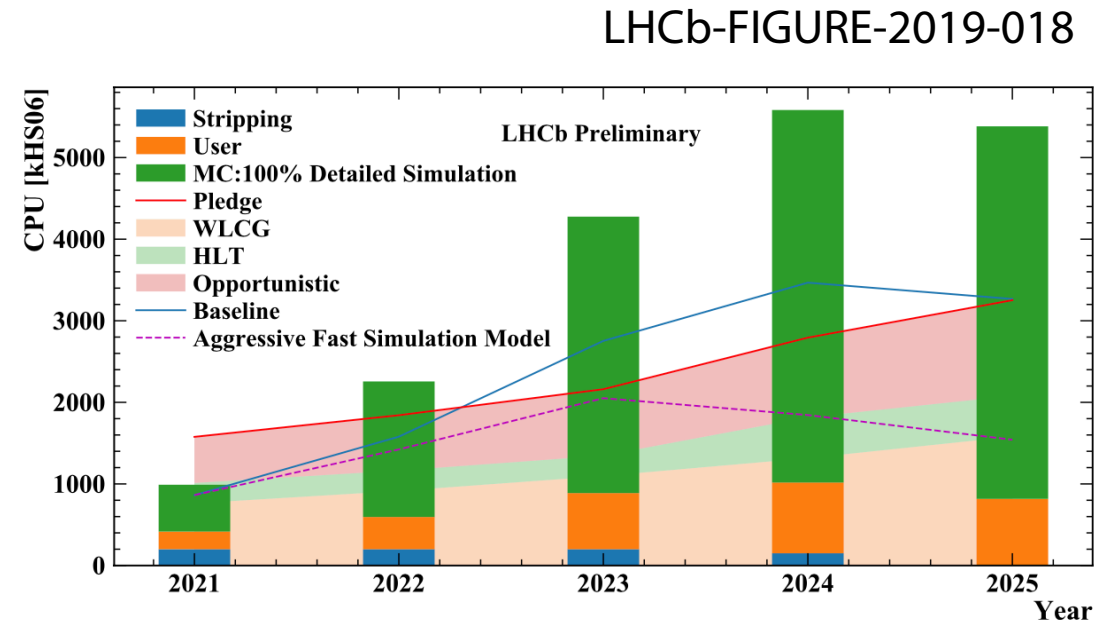


2018

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

Deep generative models for fast detector simulation

- ▶ What if, instead of generating images, we train these models to generate detector responses?
- ▶ Generating a response as fast as a single forward pass through the network
 - Should be orders of magnitude faster compared to e.g. detailed Geant4 simulation



Outline

- ▶ Generative modelling
- ▶ Deep learning models for generative modelling
 - GANs
 - VAEs
- ▶ HEP applications

Generative modelling



Problem setup

- ▶ Training data – a set of objects, e.g.:
 - Photos of animals / people faces / rooms / whatever
 - Text
 - Audio of speech / music / whatever
 - Signals from a high energy physics experiment detector
- ▶ Goal: build a model to sample similar data

Set of objects:

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

Problem setup

- ▶ Training data – a set of objects, e.g.:
 - Photos of animals / people faces / rooms / whatever
 - Text
 - Audio of speech / music / whatever
 - Signals from a high energy physics experiment detector
- ▶ Goal: build a model to sample similar data

Set of objects:

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

Population PDF:

$$p(x)$$

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

Problem setup

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

Set of objects:

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

Population PDF:

$$p(x)$$

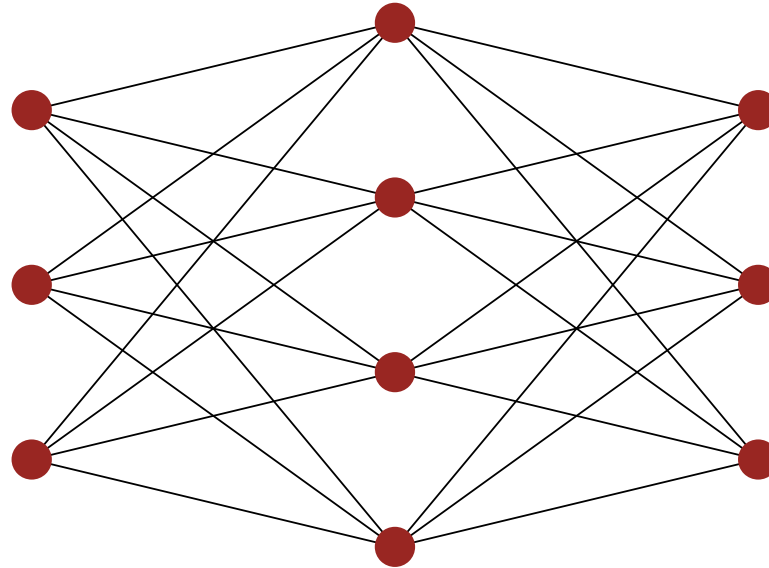
i.e. $\{x_i\}$ are i.i.d.
sampled from $p(x)$

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

Generative Adversarial Networks (GANs)

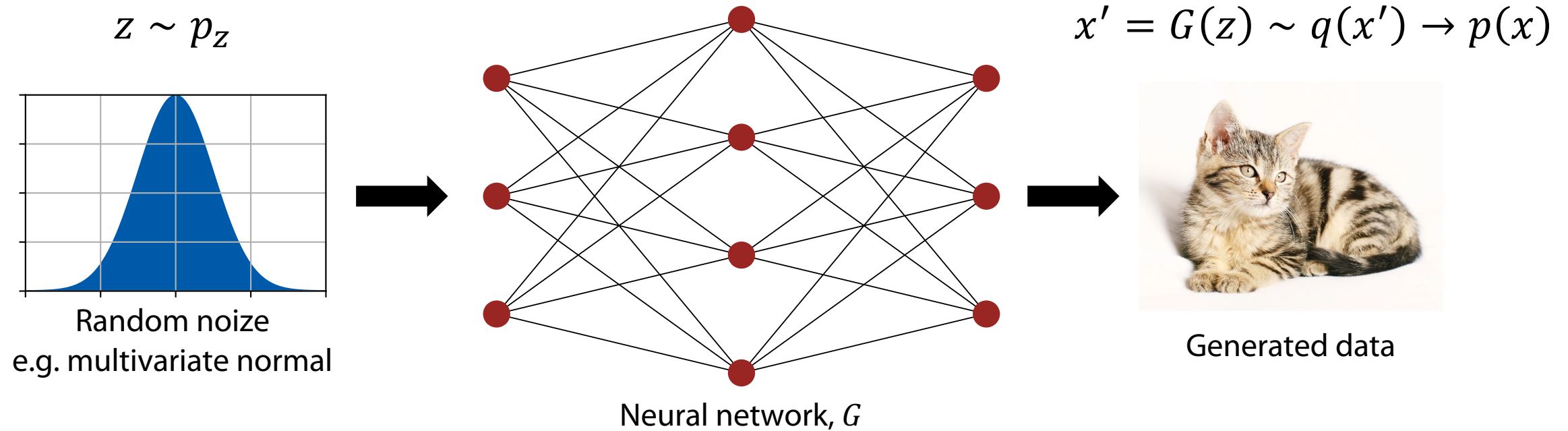


How can a neural network generate data?



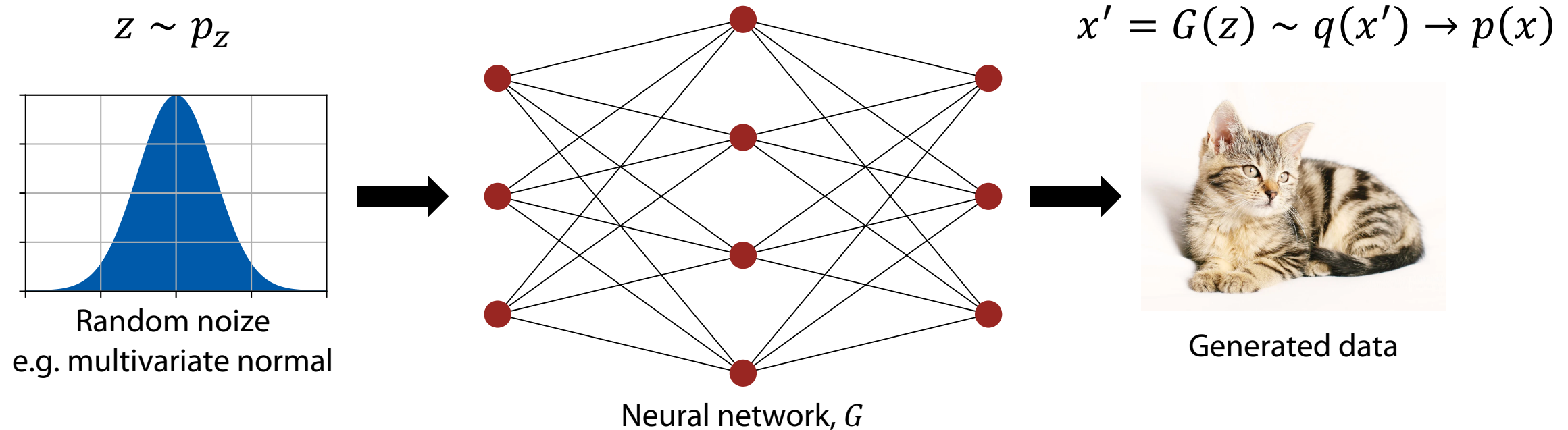
Neural network, G

How can a neural network generate data?



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

How can a neural network generate data?

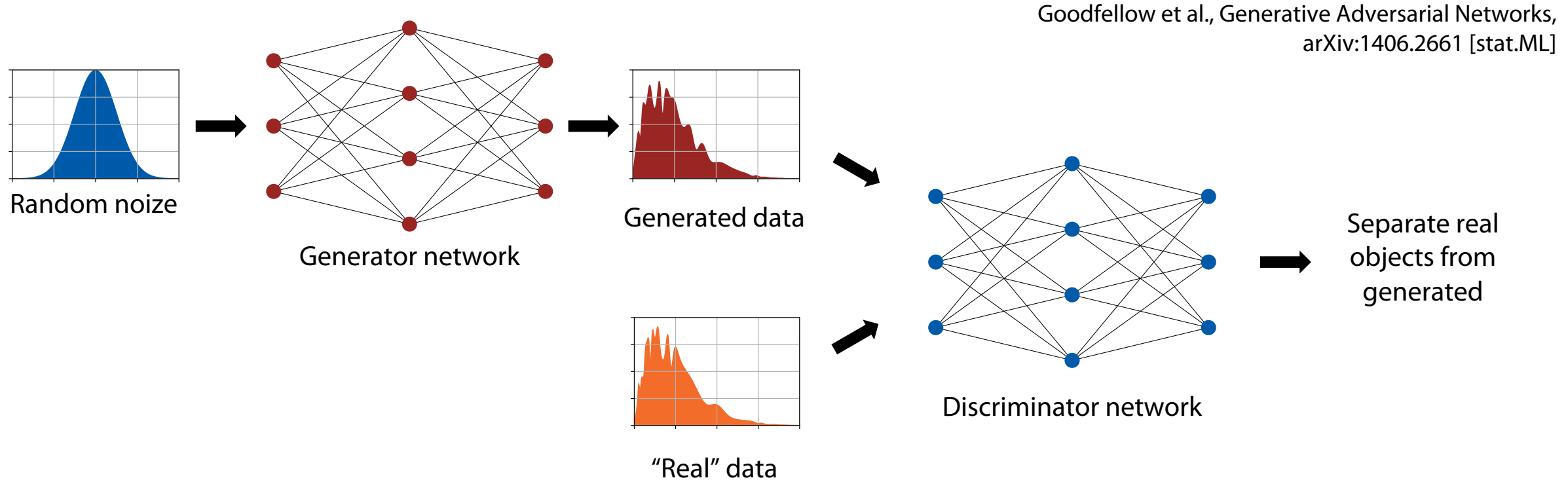


- This makes the generated object being a **differentiable function** of the network parameters

How to train such a generator?

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

Adversarial approach



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

Training the networks

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

**discriminator
steps**

for number of training iterations **do**

for k steps **do**

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

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

end for

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

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

**generator
steps**

end for

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

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

Variational Autoencoders (VAE)

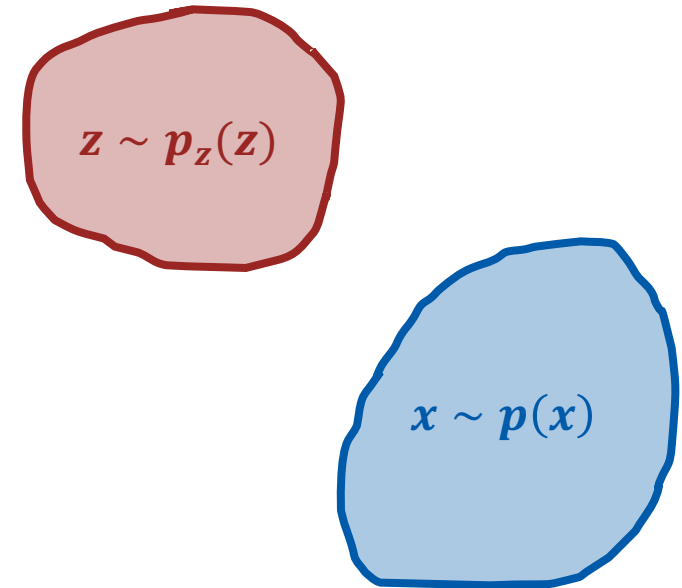


VAEs in a nutshell (1/3)

[arXiv:1401.4082](https://arxiv.org/abs/1401.4082)

[arXiv:1312.6114](https://arxiv.org/abs/1312.6114)

- ▶ Similarly to GANs, we want to find a transformation from a known distribution $p_z(z)$ to the data distribution $p(x)$, using only samples from $p(x)$

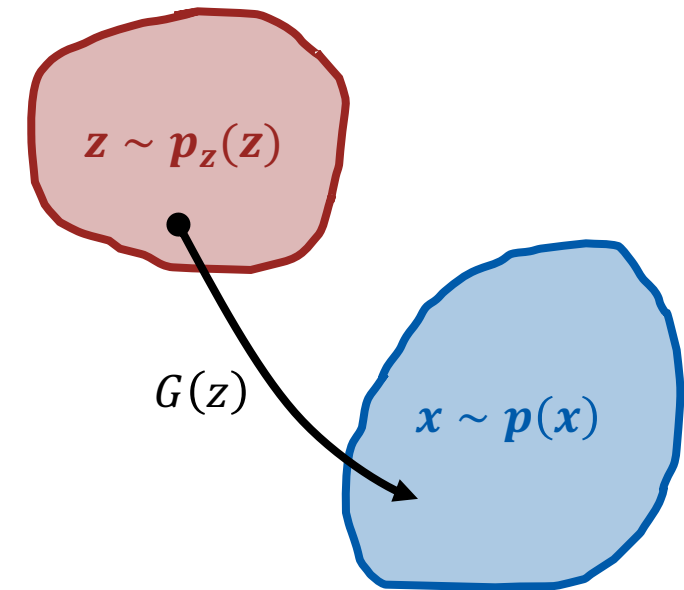


VAEs in a nutshell (1/3)

[arXiv:1401.4082](https://arxiv.org/abs/1401.4082)

[arXiv:1312.6114](https://arxiv.org/abs/1312.6114)

- ▶ Similarly to GANs, we want to find a transformation from a known distribution $p_z(z)$ to the data distribution $p(x)$, using only samples from $p(x)$
- ▶ In GANs, this transformation is deterministic ($x' = G(z)$)

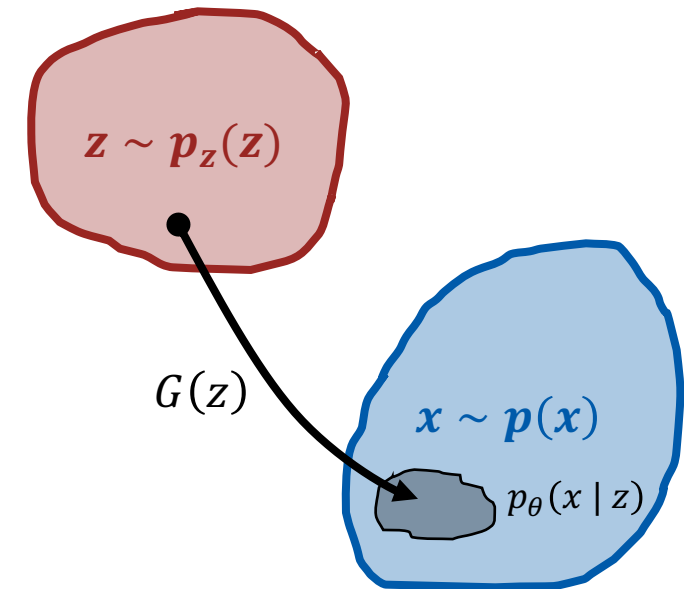


VAEs in a nutshell (1/3)

[arXiv:1401.4082](https://arxiv.org/abs/1401.4082)

[arXiv:1312.6114](https://arxiv.org/abs/1312.6114)

- ▶ Similarly to GANs, we want to find a transformation from a known distribution $p_z(z)$ to the data distribution $p(x)$, using only samples from $p(x)$
- ▶ In GANs, this transformation is deterministic ($x' = G(z)$)
- ▶ In VAEs, it is stochastic, modelled with parametric distribution $p_\theta(x | z)$
 - E.g.: $p_\theta(x | z) \equiv p(x; \lambda = G_\theta(z))$,
 - i.e. a neural network $G_\theta(z)$ maps (**decodes**) latent codes z to parameters λ of some distribution $p(x; \lambda)$



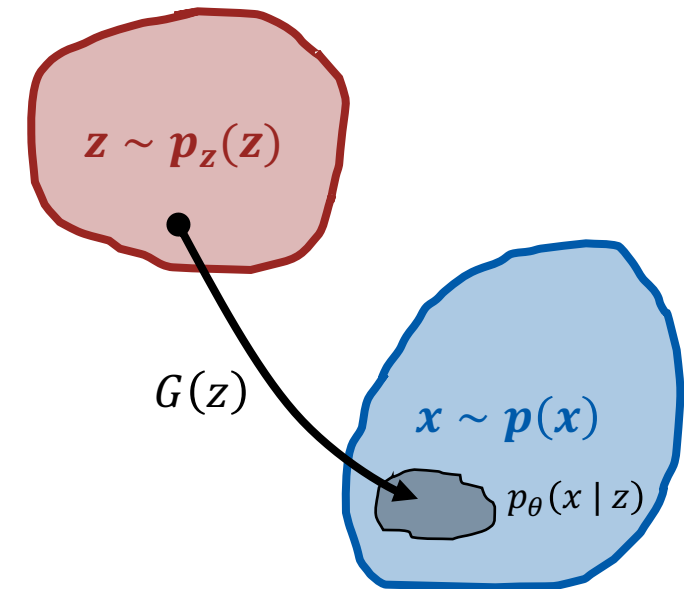
VAEs in a nutshell (1/3)

[arXiv:1401.4082](#)

[arXiv:1312.6114](#)

- ▶ Similarly to GANs, we want to find a transformation from a known distribution $p_z(z)$ to the data distribution $p(x)$, using only samples from $p(x)$
- ▶ In GANs, this transformation is deterministic ($x' = G(z)$)
- ▶ In VAEs, it is stochastic, modelled with parametric distribution $p_\theta(x | z)$
 - E.g.: $p_\theta(x | z) \equiv p(x; \lambda = G_\theta(z))$,
 - i.e. a neural network $G_\theta(z)$ maps (**decodes**) latent codes z to parameters λ of some distribution $p(x; \lambda)$
- ▶ So, our approximation to the target distribution is:

$$p_\theta(x) = \int p_\theta(x, z) dz = \int p_\theta(x | z) p_z(z) dz = \mathbb{E}_{z \sim p_z} p_\theta(x | z)$$

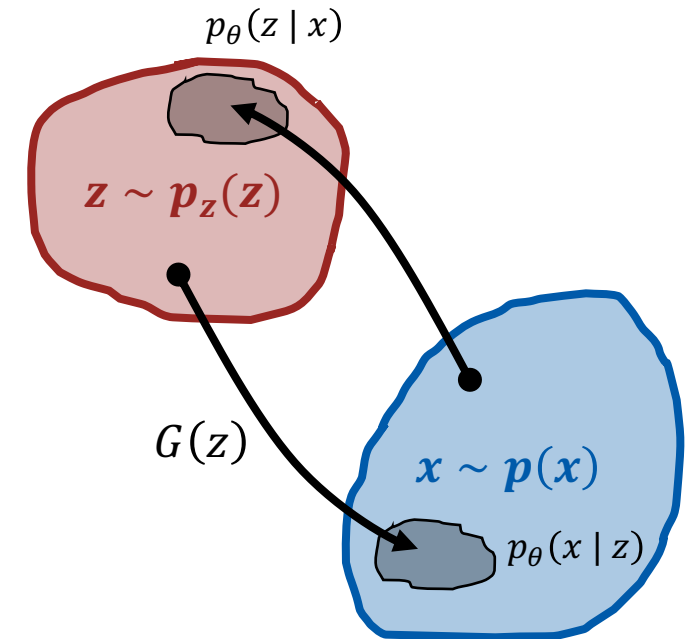


VAEs in a nutshell (2/3)

[arXiv:1401.4082](https://arxiv.org/abs/1401.4082)

[arXiv:1312.6114](https://arxiv.org/abs/1312.6114)

- ▶ Assume we know the inverse transformation $p_{\theta}(z|x)$
 - (though, this is typically intractable)



VAEs in a nutshell (2/3)

[arXiv:1401.4082](#)

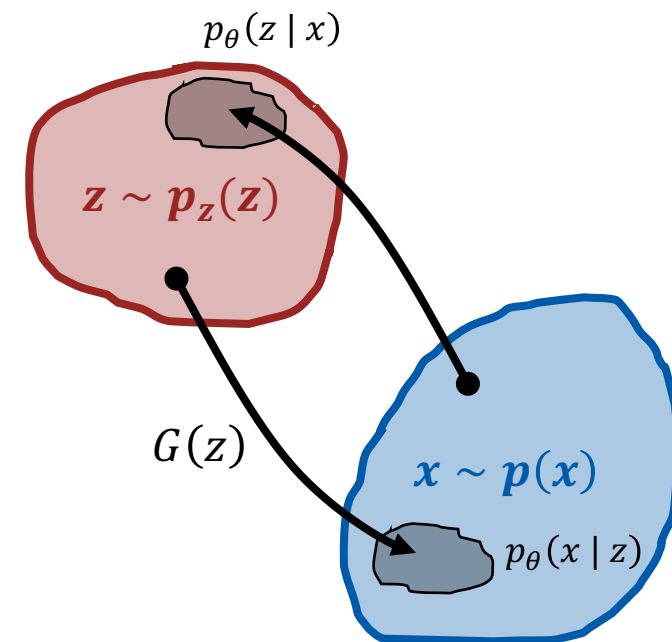
[arXiv:1312.6114](#)

- ▶ Assume we know the inverse transformation $p_{\theta}(z|x)$
 - (though, this is typically intractable)
- ▶ Then, we could efficiently train our model by maximizing the log-likelihood:

$$\begin{aligned}\log p_{\theta}(x) &= \mathbb{E}_{z \sim p_{\theta}(z|x)} \log \left[p_{\theta}(x) \frac{p_{\theta}(z|x)}{p_z(z)} \right] \\ &= \mathbb{E}_{z \sim p_{\theta}(z|x)} [\log p_{\theta}(x, z) - \log p_{\theta}(z|x)] \\ &= \underbrace{\mathbb{E}_{z \sim p_{\theta}(z|x)} \log p_{\theta}(x, z)}_{\text{encourages placing high probability mass on many } z \text{ values that could've generated } x} + \underbrace{\mathcal{H}(p_{\theta}(z|x))}_{\text{encourages placing high probability mass on many } z \text{ values that could've generated } x}\end{aligned}$$


So, for the log-likelihood we're sampling not all z values, but only those corresponding to this particular x

Maximizing this encourages placing high probability mass on many z values that could've generated x



VAEs in a nutshell (3/3)

e.g., $\mathcal{N}(z; (\mu, \sigma) = D_\phi(x))$



- ▶ In practice, $p_\theta(z|x)$ is not known, so we approximate it with some $q_\phi(z|x)$:

$$[\log p_\theta(x)]_{\text{approx.}, \phi} = \mathbb{E}_{z \sim q_\phi(z|x)} \log p_\theta(x, z) + \mathcal{H}(q_\phi(z|x))$$

- ▶ One can prove, that this approximate log-likelihood is a **lower bound** to the true log-likelihood $\log p_\theta(x)$
- ▶ Maximizing it wrt θ and ϕ will also maximize the true log-likelihood
 - Will lead to the true optimum if the family $q_\phi(z|x)$ is rich enough to include $p_\theta(z|x)$ for any θ
- ▶ It's easy to derive this alternative form, which is simpler to optimize:

$$[\log p_\theta(x)]_{\text{approx.}, \phi} = \mathbb{E}_{z \sim q_\phi(z|x)} \log p_\theta(x|z) - D_{KL}(q_\phi(z|x) \| p(z)) \rightarrow \max_{\theta, \phi}$$

Applications in HEP



Deep learning for fast simulation in HEP

- ▶ Quite a developing field!
- ▶ (not pretending to be able to cover all applications)

- [8] A. Maevskiy *et al.* [LHCb Collaboration], “Fast Data-Driven Simulation of Cherenkov Detectors Using Generative Adversarial Networks,” [arXiv:1905.11825 \[physics.ins-det\]](#).
- [9] D. Belayneh *et al.*, “Calorimetry with Deep Learning: Particle Simulation and Reconstruction for Collider Physics,” [arXiv:1912.06794 \[physics.ins-det\]](#).
- [10] J. R. Vlimant, F. Pantaleo, M. Pierini, V. Loncar, S. Vallecorsa, D. Anderson, T. Nguyen and A. Zlokapa, “Large-Scale Distributed Training Applied to Generative Adversarial Networks for Calorimeter Simulation,” *EPJ Web Conf.* **214**, 06025 (2019) doi:10.1051/epjconf/201921406025.
- [11] D. Lancierini, P. Owen and N. Serra, “Simulating the LHCb hadron calorimeter with generative adversarial networks,” *Nuovo Cim. C* **42**, no. 4, 197 (2019) doi:10.1393/ncc/i2019-19197-3.
- [12] L. de Oliveira, M. Paganini and B. Nachman, “Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis,” *Comput. Softw. Big Sci.* **1**, no. 1, 4 (2017) doi:10.1007/s41781-017-0004-6 [[arXiv:1701.05927 \[stat.ML\]](#)].
- [13] S. Carrazza and F. A. Dreyer, “Lund jet images from generative and cycle-consistent adversarial networks,” *Eur. Phys. J. C* **79**, no. 11, 979 (2019) doi:10.1140/epjc/s10052-019-7501-1 [[arXiv:1909.01359 \[hep-ph\]](#)].
- [14] M. Paganini, L. de Oliveira and B. Nachman, “Accelerating Science with Generative Adversarial Networks: An Application to 3D Particle Showers in Multilayer Calorimeters,” *Phys. Rev. Lett.* **120**, no. 4, 042003 (2018) doi:10.1103/PhysRevLett.120.042003 [[arXiv:1705.02355 \[hep-ex\]](#)].
- [15] L. de Oliveira, M. Paganini and B. Nachman, “Controlling Physical Attributes in GAN-Accelerated Simulation of Electromagnetic Calorimeters,” *J. Phys. Conf. Ser.* **1085**, no. 4, 042017 (2018) doi:10.1088/1742-6596/1085/4/042017 [[arXiv:1711.08813 \[hep-ex\]](#)].
- [16] M. Paganini, L. de Oliveira and B. Nachman, “CaloGAN : Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks,” *Phys. Rev. D* **97**, no. 1, 014021 (2018) doi:10.1103/PhysRevD.97.014021 [[arXiv:1712.10321 \[hep-ex\]](#)].
- [17] F. Carminati, A. Gheata, G. Khattak, P. Mendez Lorenzo, S. Sharan and S. Vallecorsa, “Three dimensional Generative Adversarial Networks for fast simulation,” *J. Phys. Conf. Ser.* **1085**, no. 3, 032016 (2018) doi:10.1088/1742-6596/1085/3/032016.
- [18] M. Erdmann, L. Geiger, J. Glombitza and D. Schmidt, “Generating and refining particle detector simulations using the Wasserstein distance in adversarial networks,” *Comput. Softw. Big Sci.* **2**, no. 1, 4 (2018) doi:10.1007/s41781-018-0008-x [[arXiv:1802.03325 \[astro-ph.IM\]](#)].
- [19] P. Musella and F. Pandolfi, “Fast and Accurate Simulation of Particle Detectors Using Generative Adversarial Networks,” *Comput. Softw. Big Sci.* **2**, no. 1, 8 (2018) doi:10.1007/s41781-018-0015-y [[arXiv:1805.00850 \[hep-ex\]](#)].
- [20] M. Erdmann, J. Glombitza and T. Quast, “Precise simulation of electromagnetic calorimeter showers using a Wasserstein Generative Adversarial Network,” *Comput. Softw. Big Sci.* **3**, no. 1, 4 (2019) doi:10.1007/s41781-018-0019-7 [[arXiv:1807.01954 \[physics.ins-det\]](#)].
- [21] S. Vallecorsa, F. Carminati and G. Khattak, “3D convolutional GAN for fast simulation,” *EPJ Web Conf.* **214**, 02010 (2019) doi:10.1051/epjconf/201921402010.
- [22] S. Otten, S. Caron, W. de Swart, M. van Beekveld, L. Hendriks, C. van Leeuwen, D. Podareanu, R. R. de Austri and R. Verheyen, “Event Generation and Statistical Sampling for Physics with Deep Generative Models and a Density Information Buffer,” [arXiv:1901.00875 \[hep-ph\]](#).
- [23] A. Butter, T. Plehn and R. Winterhalder, “How to GAN LHC Events,” *SciPost Phys.* **7**, no. 6, 075 (2019) doi:10.21468/SciPostPhys.7.6.075 [[arXiv:1907.03764 \[hep-ph\]](#)].
- [24] C. Ahdida *et al.* [SHIP Collaboration], “Fast simulation of muons produced at the SHiP experiment using Generative Adversarial Networks,” *JINST* **14**, P11028 (2019) doi:10.1088/1748-0221/14/11/P11028 [[arXiv:1909.04451 \[physics.ins-det\]](#)].
- [25] S. Farrell, W. Bhimji, T. Kurth, M. Mustafa, D. Bard, Z. Lukic, B. Nachman and H. Patton, “Next Generation Generative Neural Networks for HEP,” *EPJ Web Conf.* **214**, 09005 (2019) doi:10.1051/epjconf/201921409005.
- [26] J. Arjona Martínez, T. Q. Nguyen, M. Pierini, M. Spiropulu and J. R. Vlimant, “Particle Generative Adversarial Networks for full-event simulation at the LHC and their application to pileup description,” [arXiv:1912.02748 \[hep-ex\]](#).
- [27] B. Hashemi, N. Amin, K. Datta, D. Olivito and M. Pierini, “LHC analysis-specific datasets with Generative Adversarial Networks,” [arXiv:1901.05282 \[hep-ex\]](#).
- [28] R. Di Sipio, M. Fucci Giannelli, S. Ketabchi Haghighat and S. Palazzo, “A Generative-Adversarial Network Approach for the Simulation of QCD Dijet Events at the LHC,” *PoS LeptonPhoton* **2019**, 050 (2019) doi:10.22323/1.367.0050.
- [29] R. Di Sipio, M. Fucci Giannelli, S. Ketabchi Haghighat and S. Palazzo, “DijetGAN: A Generative-Adversarial Network Approach for the Simulation of QCD Dijet Events at the LHC,” *JHEP* **1908**, 110 (2020) doi:10.1007/JHEP08(2019)110 [[arXiv:1903.02433 \[hep-ex\]](#)].
- [30] Y. Alanazi, N. Sato, T. Liu, W. Melnitchouk, M. P. Kuchera, E. Pritchard, M. Robertson, R. Strauss, L. Velasco and Y. Li, “Simulation of electron-proton scattering events by a Feature-Augmented and Transformed Generative Adversarial Network (FAT-GAN),” [arXiv:2001.11103 \[hep-ph\]](#).

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

Where it all started: LAGAN

- de Oliveira, L., Paganini, M. & Nachman, B., arXiv:1701.05927

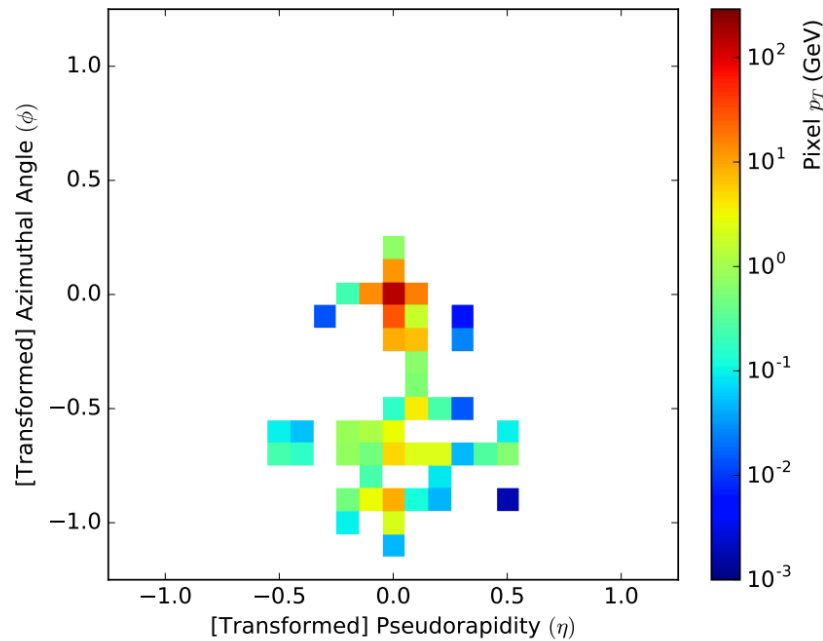


Figure 1: A typical jet image.

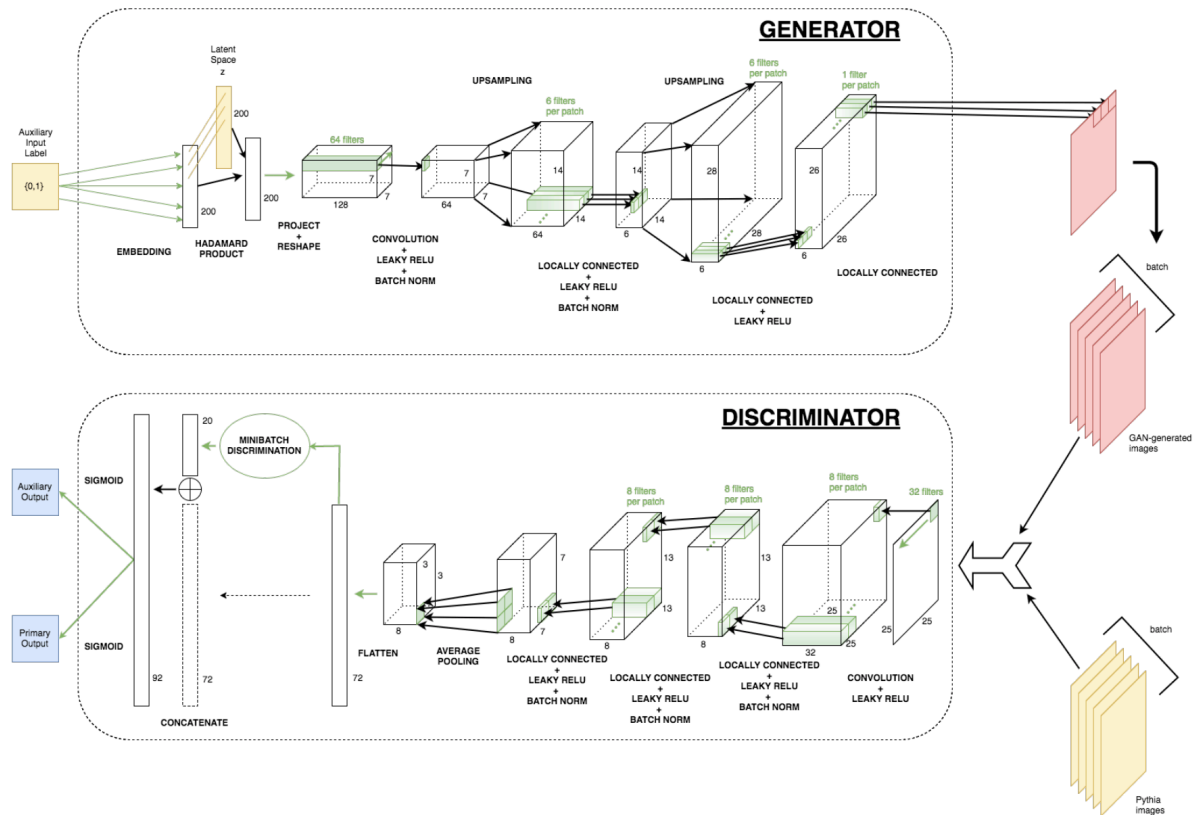
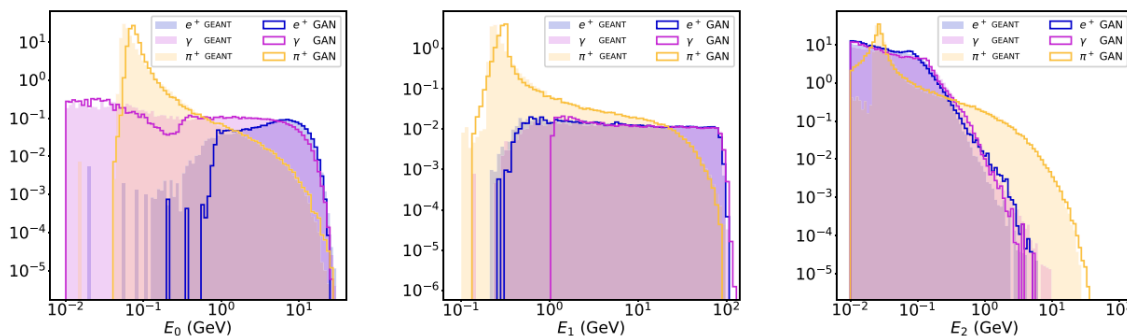


Figure 4: LAGAN architecture

- Demonstrated the ability to generate realistic jet images

CaloGAN (3D calorimeter)

- ▶ L., Paganini, de Oliveira, M. & Nachman, B., arXiv:1705.02355
- ▶ Up to $\mathcal{O}(10^3)$ time improvement on CPU
- ▶ Up to $\mathcal{O}(10^5)$ on GPU



Some physically-motivated variables for validation
(not seen at training time)

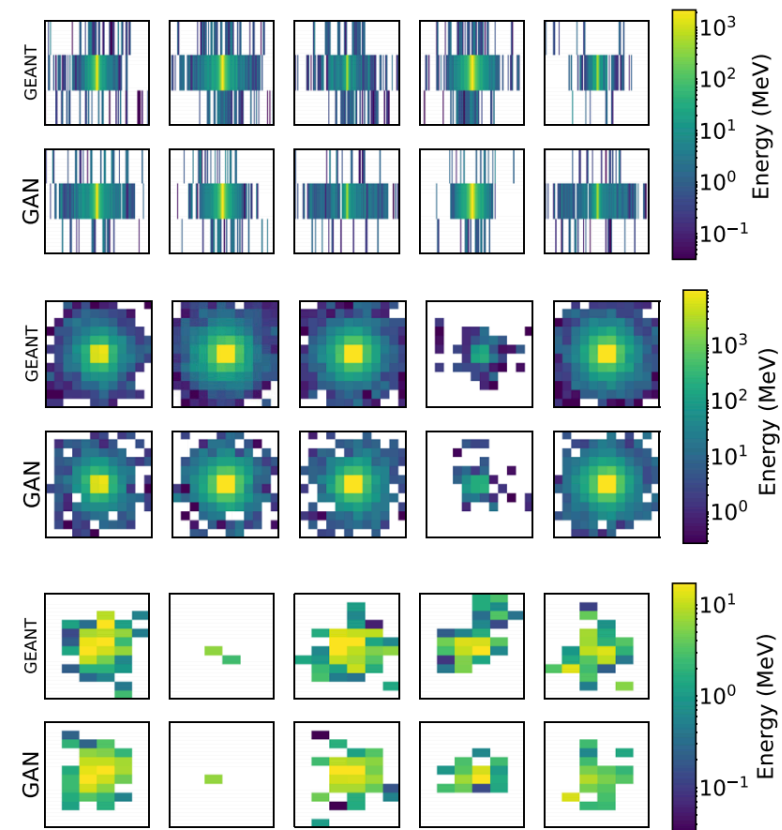


FIG. 2. Five randomly selected γ showers per calorimeter layer from GEANT4 (top rows) and their five nearest neighbors (by Euclidean distance) from a set of CaloGAN candidates.

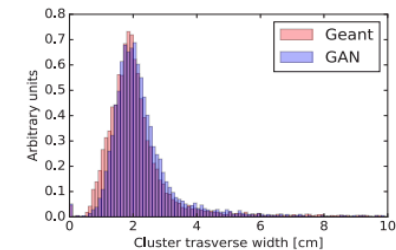
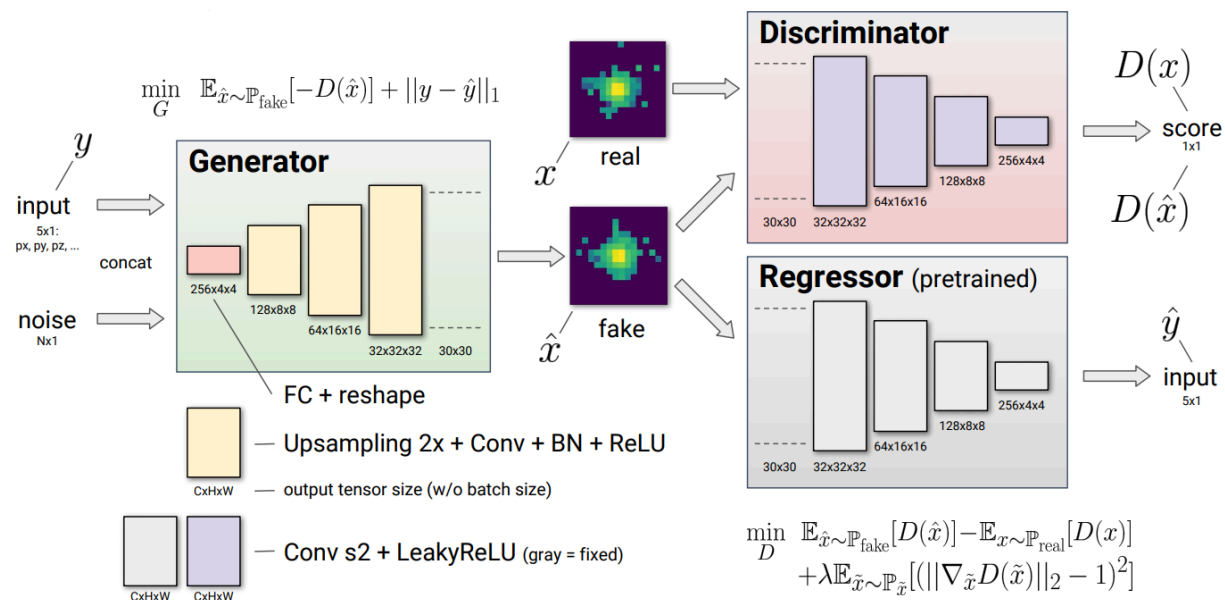
Fast Calorimeter Simulation: the LHCb case

► [arXiv:1812.01319](https://arxiv.org/abs/1812.01319)

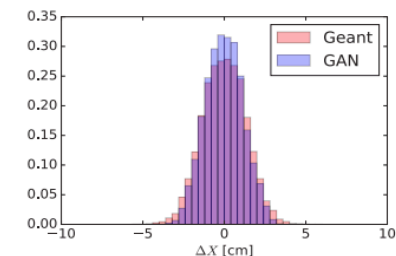
- Uses WGAN-GP modification of GAN (arXiv:1704.00028)

► 0.07 ms per sample (GPU)

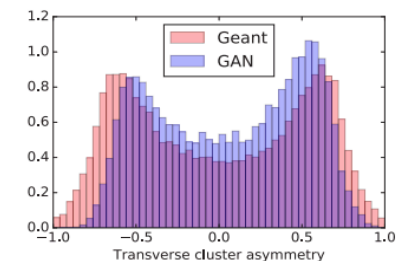
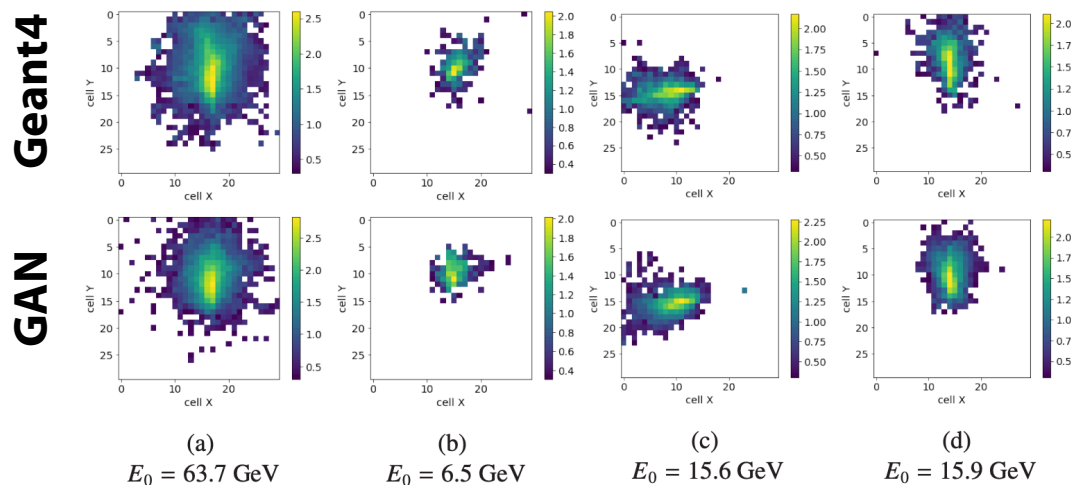
► 4.9 ms per sample (CPU)



(a) The transverse width of real and generated clusters



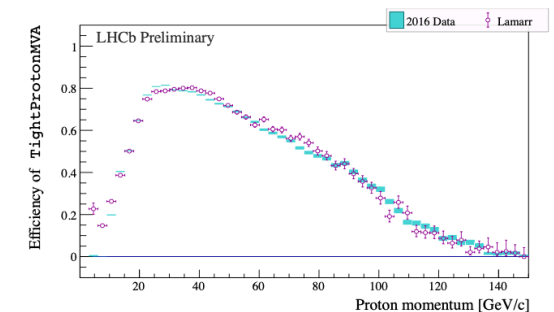
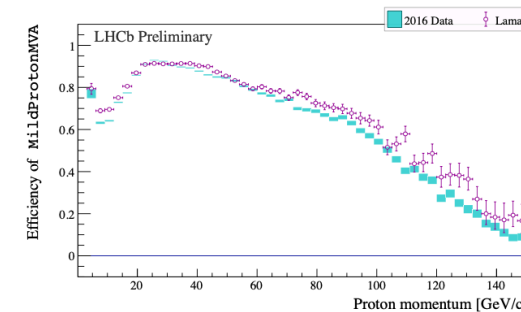
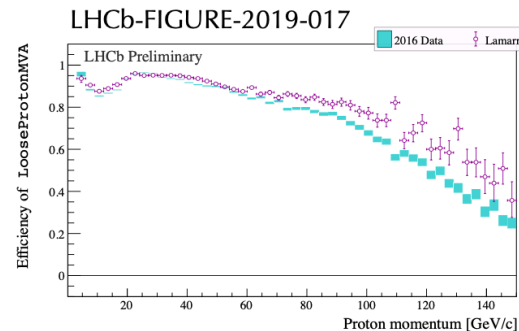
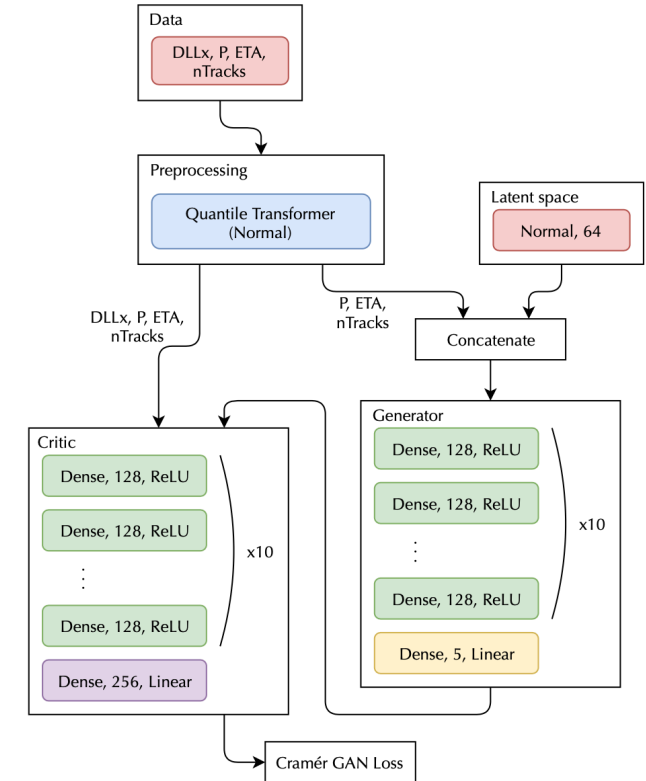
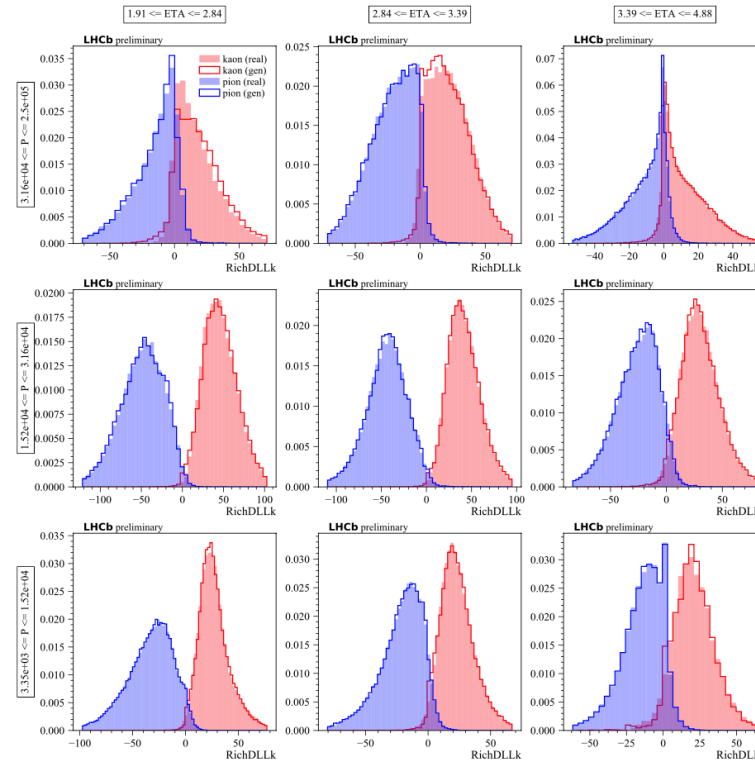
(c) ΔX between cluster center of mass and the true particle coordinate



(e) The transverse asymmetry of real and generated clusters

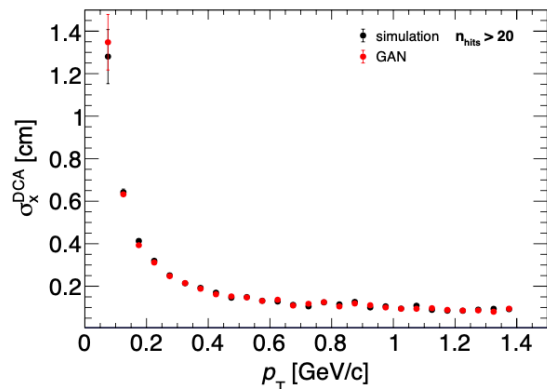
Data-driven simulation of LHCb Cherenkov detectors

- ▶ [arXiv:1905.11825](https://arxiv.org/abs/1905.11825)
- ▶ Cramer-GAN
(arXiv:1705.10743)
- ▶ Trained on real data
 - Utilized sPlot for background subtraction

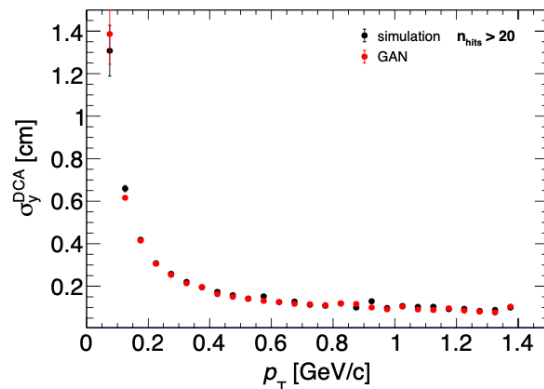


Time projection chamber fastsim at MPD (NICA)

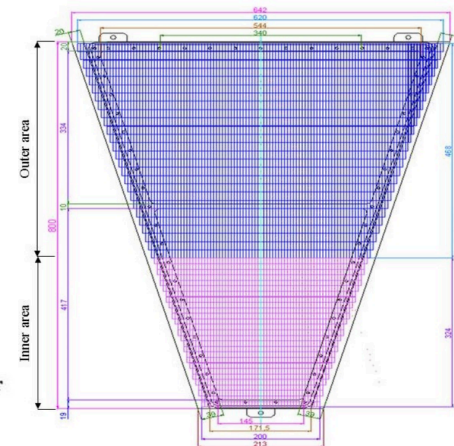
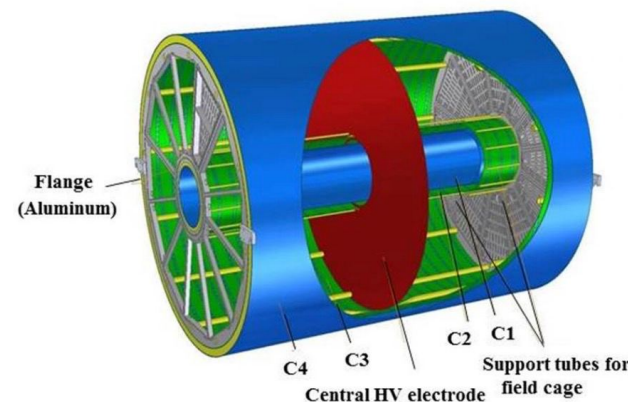
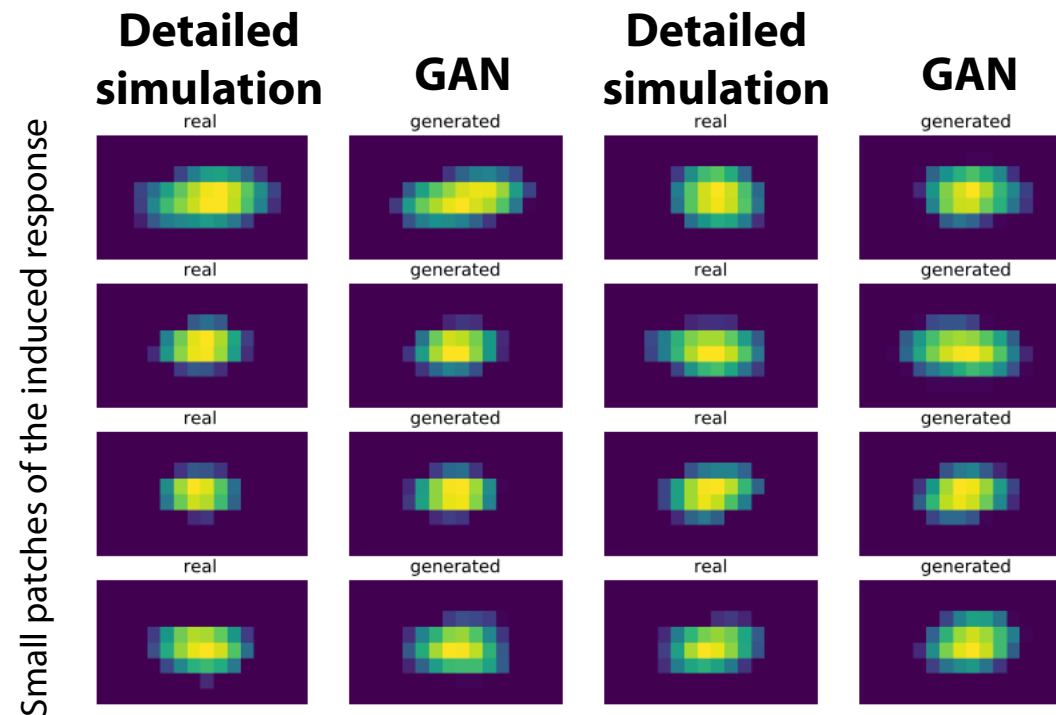
- ▶ [arXiv:2012.04595](https://arxiv.org/abs/2012.04595)
- ▶ Tracking characteristics are spot-on
- ▶ $\mathcal{O}(10)$ speed-up factor



(a) Distance of closest approach resolution along x

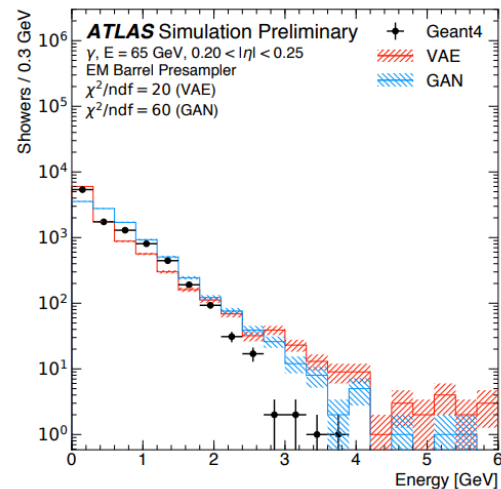
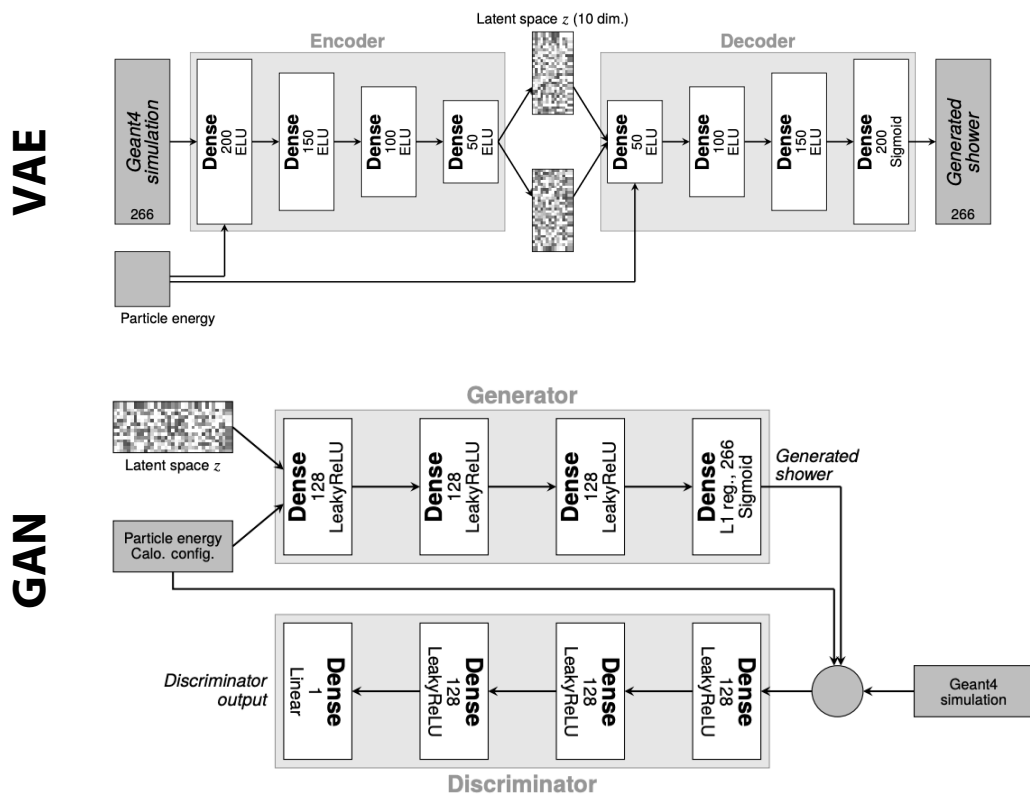


(b) Distance of closest approach resolution along y

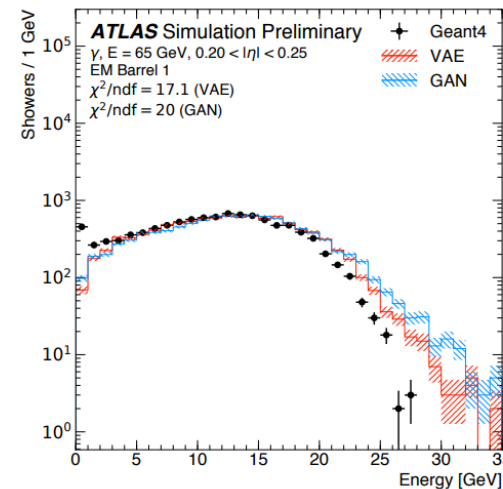


Fast shower simulation in ATLAS

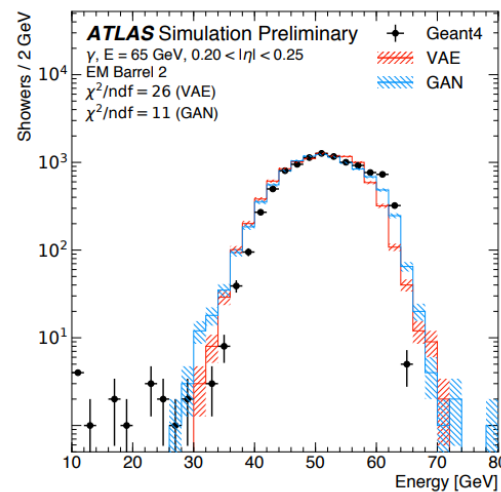
- ▶ ATL-SOFT-PUB-2018-001
- ▶ 3d calorimeter simulation
- ▶ Tested both GAN and VAE



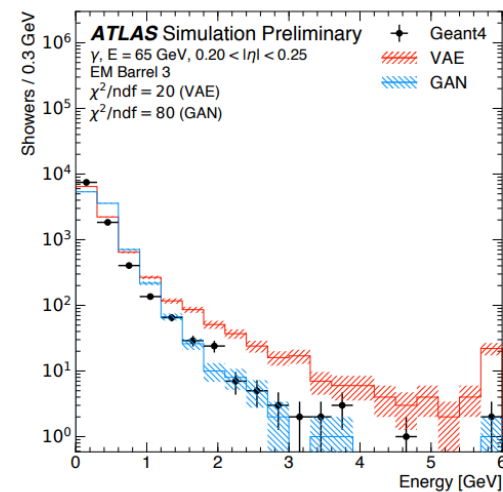
(a) Presampler



(b) Front layer



(c) Middle layer



(d) Back layer

There's many more...

► Check out this list if interested:

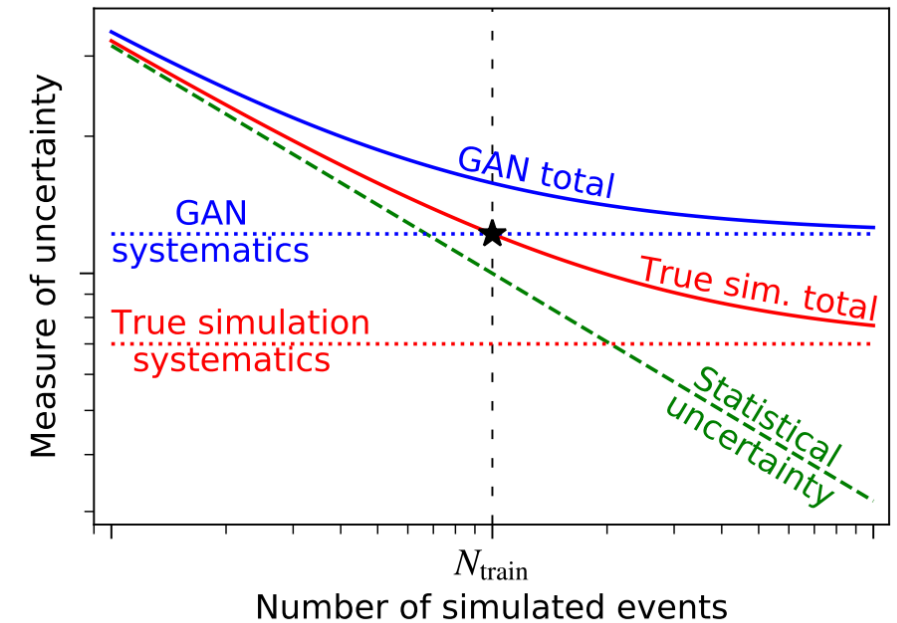
<https://github.com/iml-wg/HEPML-LivingReview>

- Generative models / density estimation
 - GANs:
 - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis [DOI]
 - Accelerating Science with Generative Adversarial Networks: An Application to 3D Particle Showers in Multilayer Calorimeters [DOI]
 - CaloGAN : Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks [DOI]
 - Image-based model parameter optimization using Model-Assisted Generative Adversarial Networks [DOI]
 - How to GAN Event Subtraction [DOI]
 - Particle Generative Adversarial Networks for full-event simulation at the LHC and their application to pileup description [DOI]
 - How to GAN away Detector Effects [DOI]
 - 3D convolutional GAN for fast simulation
 - Fast simulation of muons produced at the SHIP experiment using Generative Adversarial Networks [DOI]
 - Lund jet images from generative and cycle-consistent adversarial networks [DOI]
 - How to GAN LHC Events [DOI]
 - Machine Learning Templates for QCD Factorization in the Search for Physics Beyond the Standard Model [DOI]
 - DijetGAN: A Generative-Adversarial Network Approach for the Simulation of QCD Dijet Events at the LHC [DOI]
 - LHC analysis-specific datasets with Generative Adversarial Networks
 - Generative Models for Fast Calorimeter Simulation.LHCb case [DOI]
 - Deep generative models for fast shower simulation in ATLAS
 - Regressive and generative neural networks for scalar field theory [DOI]
 - Three dimensional Generative Adversarial Networks for fast simulation
 - Generative models for fast simulation
 - Unfolding with Generative Adversarial Networks
 - Fast and Accurate Simulation of Particle Detectors Using Generative Adversarial Networks [DOI]
 - Generating and refining particle detector simulations using the Wasserstein distance in adversarial networks [DOI]
 - Generative models for fast cluster simulations in the TPC for the ALICE experiment
 - RICH 2018 [DOI]
 - GANs for generating EFT models [DOI]
 - Precise simulation of electromagnetic calorimeter showers using a Wasserstein Generative Adversarial Network [DOI]
 - Reducing Autocorrelation Times in Lattice Simulations with Generative Adversarial Networks [DOI]
 - Tips and Tricks for Training GANs with Physics Constraints
 - Controlling Physical Attributes in GAN-Accelerated Simulation of Electromagnetic Calorimeters [DOI]
 - Next Generation Generative Neural Networks for HEP
 - Calorimetry with Deep Learning: Particle Classification, Energy Regression, and Simulation for High-Energy Physics
 - Calorimetry with Deep Learning: Particle Simulation and Reconstruction for Collider Physics [DOI]
 - Getting High: High Fidelity Simulation of High Granularity Calorimeters with High Speed
 - AI-based Monte Carlo event generator for electron-proton scattering
 - DCTRGAN: Improving the Precision of Generative Models with Reweighting [DOI]
 - GANplifying Event Samples
 - Graph Generative Adversarial Networks for Sparse Data Generation in High Energy Physics
 - Simulating the Time Projection Chamber responses at the MPD detector using Generative Adversarial Networks
 - Explainable machine learning of the underlying physics of high-energy particle collisions
 - A Data-driven Event Generator for Hadron Colliders using Wasserstein Generative Adversarial Network [DOI]
 - Reduced Precision Strategies for Deep Learning: A High Energy Physics Generative Adversarial Network Use Case [DOI]
 - Validation of Deep Convolutional Generative Adversarial Networks for High Energy Physics Calorimeter Simulations
 - Compressing PDF sets using generative adversarial networks
 - Physics Validation of Novel Convolutional 2D Architectures for Speeding Up High Energy Physics Simulations
 - Autoencoders
 - Deep Learning as a Parton Shower
 - Deep generative models for fast shower simulation in ATLAS
 - Variational Autoencoders for Anomalous Jet Tagging
 - Variational Autoencoders for Jet Simulation
 - Foundations of a Fast, Data-Driven, Machine-Learned Simulator
 - Decoding Photons: Physics in the Latent Space of a BIB-AE Generative Network
 - Bump Hunting in Latent Space
 - (End-to-end Sinkhorn Autoencoder with Noise Generator
 - Graph Generative Models for Fast Detector Simulations in High Energy Physics
 - DeepRICH: Learning Deeply Cherenkov Detectors [DOI]
 - Normalizing flows
 - Flow-based generative models for Markov chain Monte Carlo in lattice field theory [DOI]
 - Equivariant flow-based sampling for lattice gauge theory [DOI]
 - Flows for simultaneous manifold learning and density estimation
 - Exploring phase space with Neural Importance Sampling [DOI]
 - Event Generation with Normalizing Flows [DOI]
 - i-flow: High-Dimensional Integration and Sampling with Normalizing Flows [DOI]
 - Anomaly Detection with Density Estimation [DOI]
 - Data-driven Estimation of Background Distribution through Neural Autoregressive Flows
 - SARM: Sparse Autoregressive Model for Scalable Generation of Sparse Images in Particle Physics [DOI]
 - Measuring QCD Splittings with Invertible Networks
 - Efficient sampling of constrained high-dimensional theoretical spaces with machine learning
 - Physics-inspired
 - JUNIPR: a Framework for Unsupervised Machine Learning in Particle Physics
 - Binary JUNIPR: an interpretable probabilistic model for discrimination [DOI]
 - Exploring the Possibility of a Recovery of Physics Process Properties from a Neural Network Model [DOI]
 - Explainable machine learning of the underlying physics of high-energy particle collisions
 - Symmetry meets AI
 - Mixture Models
 - Data Augmentation at the LHC through Analysis-specific Fast Simulation with Deep Learning
 - Mixture Density Network Estimation of Continuous Variable Maximum Likelihood Using Discrete Training Samples
 - Phase space generation
 - Efficient Monte Carlo Integration Using Boosted Decision
 - Exploring phase space with Neural Importance Sampling [DOI]
 - Event Generation with Normalizing Flows [DOI]
 - i-flow: High-Dimensional Integration and Sampling with Normalizing Flows [DOI]
 - Neural Network-Based Approach to Phase Space Integration [DOI]
 - VegasFlow: accelerating Monte Carlo simulation across multiple hardware platforms [DOI]
 - A Neural Resampler for Monte Carlo Reweighting with Preserved Uncertainties [DOI]
 - Improved Neural Network Monte Carlo Simulation [DOI]
 - Phase Space Sampling and Inference from Weighted Events with Autoregressive Flows [DOI]
 - How to GAN Event Unweighting
 - Gaussian processes
 - Modeling Smooth Backgrounds and Generic Localized Signals with Gaussian Processes
 - Accelerating the BSM interpretation of LHC data with machine learning [DOI]
 - \mathcal{X}_{sec} : the cross-section evaluation code [DOI]
 - AI-optimized detector design for the future Electron-Ion Collider: the dual-radiator RICH case [DOI]

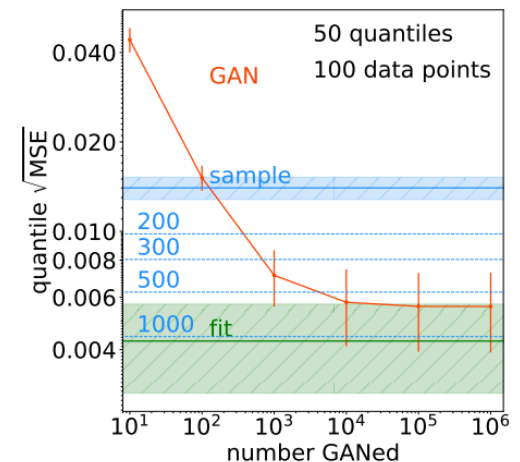
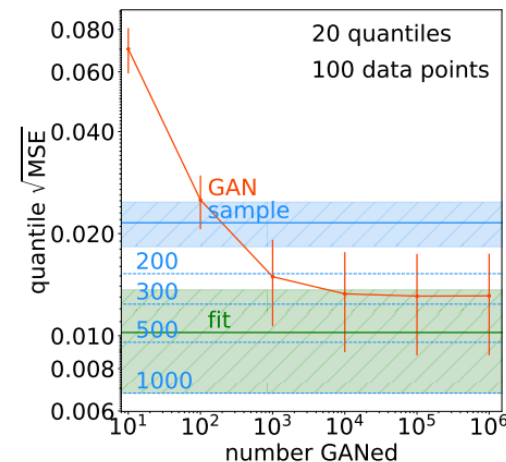
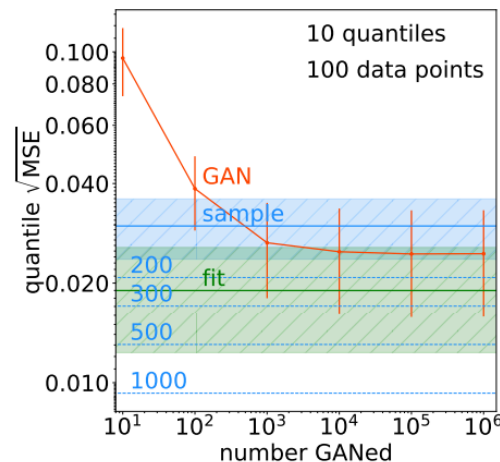
Note on systematic uncertainties

- ▶ Trained on a finite sample, generative models introduce **additional systematics**
- ▶ However, a generative model may contain **more statistical power** than the original training dataset — due to interpolation

arXiv:2002.06307 [hep-ph]



arXiv:2008.06545 [hep-ph]



Summary

- ▶ Deep generative models are an exciting and very quickly developing field of machine learning
- ▶ They promise to be good candidates for fast simulation models in HEP
- ▶ GANs and VAEs briefly covered in this talk, did not get to other models, e.g. based on Normalizing Flows or Mixture models
- ▶ Performance evaluation of a trained model is always tricky

Thank you!



amaevskij@hse.ru



SiLiKhon



hse_lambda

Artem Maevskiy

Backup



GAN



Let's put it in formulas

- ▶ Noise samples:

$$z_i \sim p_z(z)$$

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

- ▶ Generated samples:

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

where G_θ is the generator network with parameters θ .

- ▶ Discriminator network (with parameters ϕ):

$$D_\phi(x)$$

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

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

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

Probability the sample
was generated

Log-likelihood for discriminator prediction

Training the networks

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

for number of training iterations **do**

for k steps **do**

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

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

end for

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

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

end for

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

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

Problems with GANs



The optimal discriminator solution

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

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

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

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

- ▶ So the objective becomes:

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

$$= -\log 4 + JSD(p \parallel p_{\text{gen},\theta})$$

↑
Jensen-Shannon divergence

Vanishing gradients

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

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

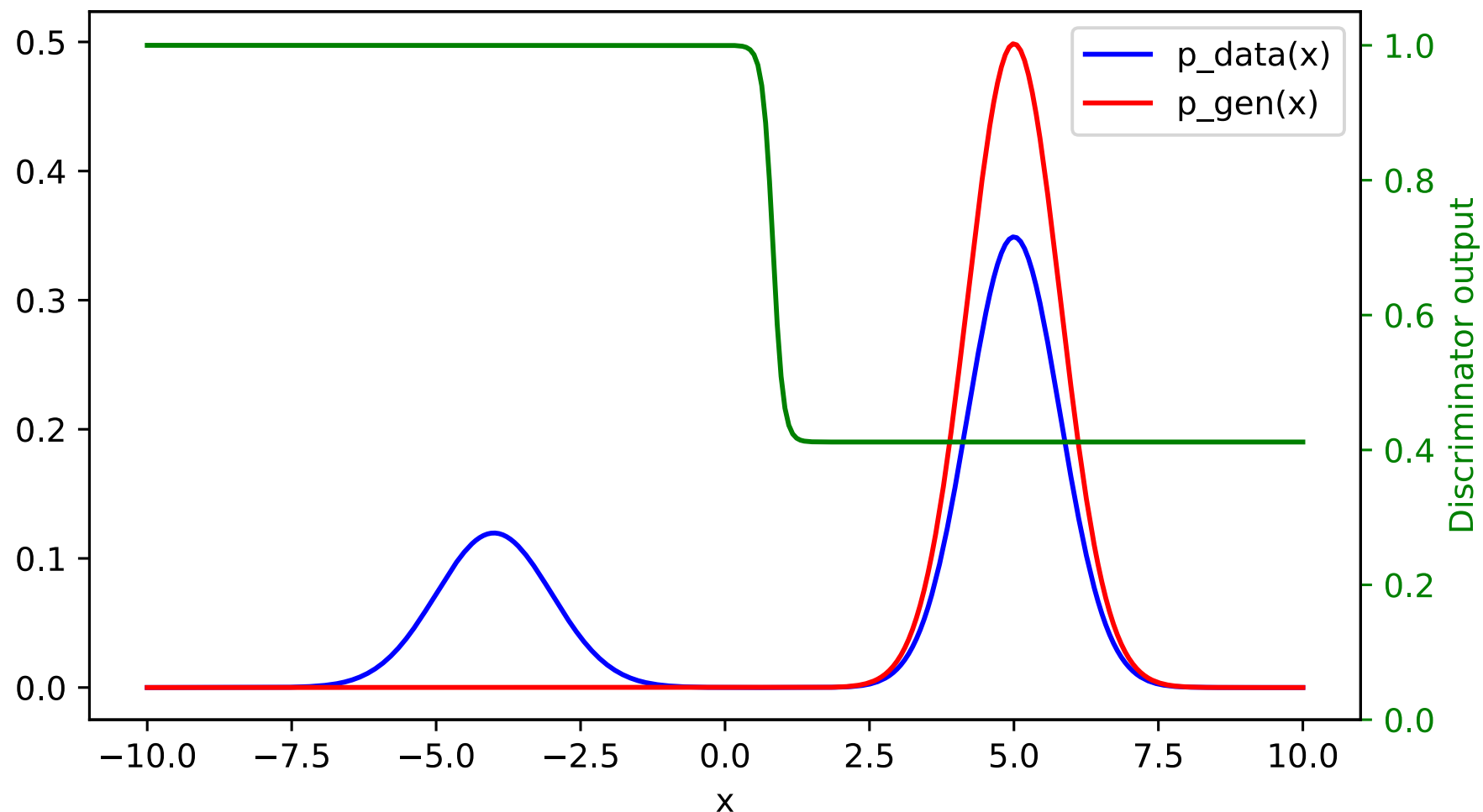
$$= \mathbb{E}_{x \sim p(x)} \left[\log \frac{p(x)}{p(x)} \right] + \mathbb{E}_{x \sim p_{\text{gen},\theta}(x)} \left[\log \frac{p_{\text{gen},\theta}(x)}{p_{\text{gen},\theta}(x)} \right] = 0 = \text{const}$$

- ▶ No meaningful gradient, can't learn

Mode collapse

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

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



Wasserstein GAN



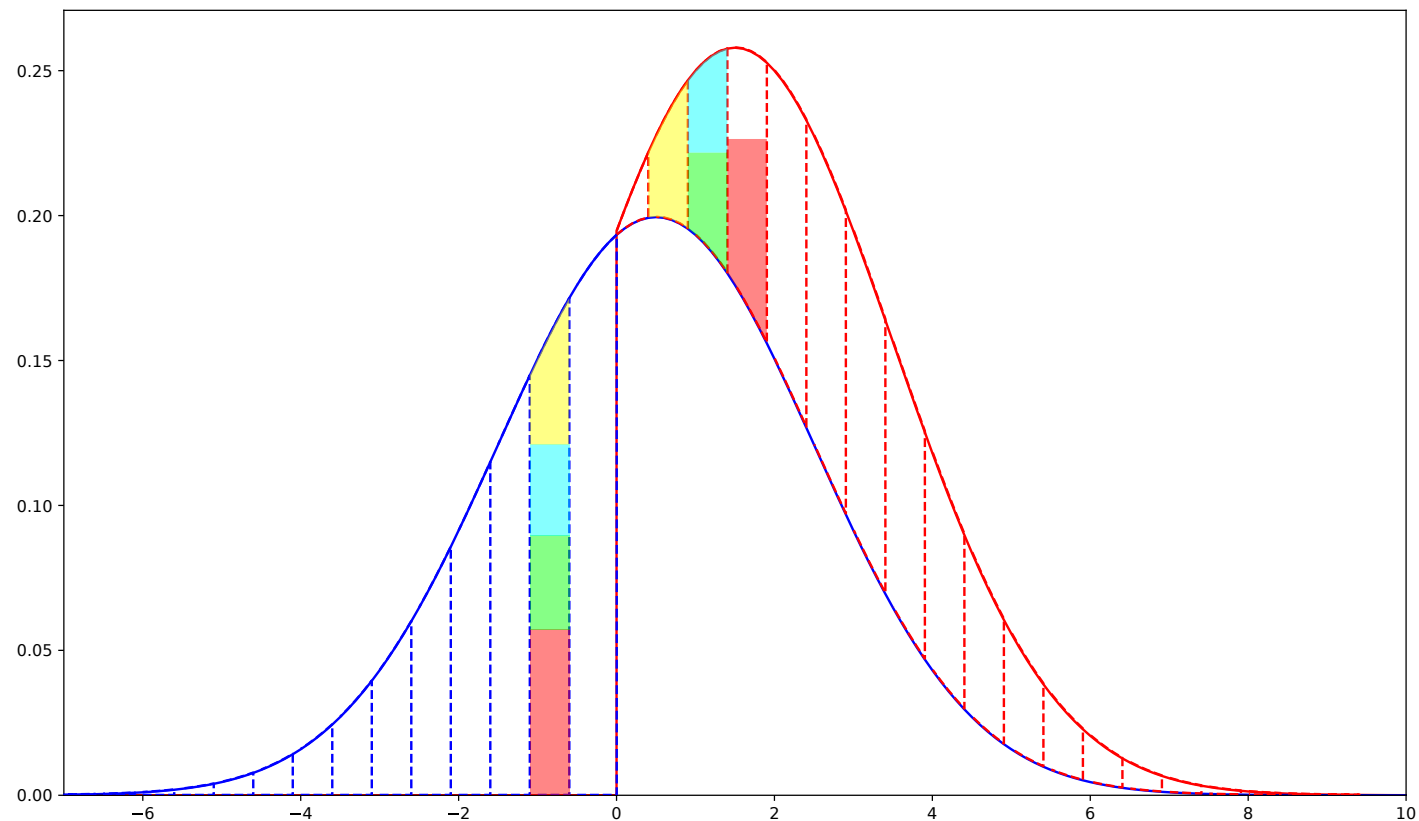
Alternative distance measure

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

Wasserstein distance

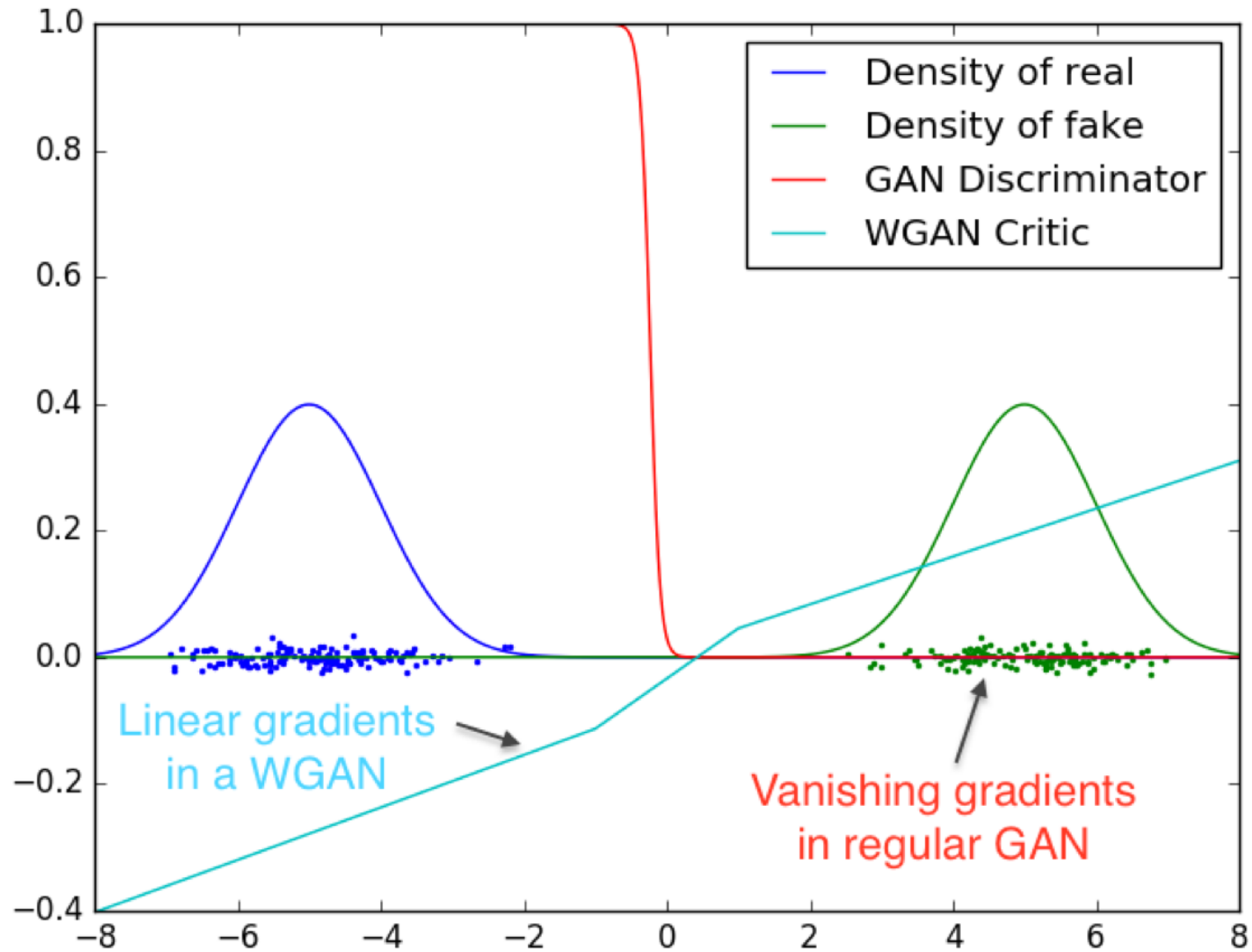
Also called “Earth mover’s distance” (EMD)

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



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

Why is it better?



Formal definition

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

$\gamma(x_1, x_2)dx_1dx_2$ – how much dirt we're moving from
 $[x_1, x_1 + dx_1]$ to $[x_2, x_2 + dx_2]$

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

$$\|x_2 - x_1\| \cdot \gamma(x_1, x_2)dx_1dx_2$$

- ▶ and the total cost is:

$$C = \int_{x_1, x_2} \|x_2 - x_1\| \cdot \gamma(x_1, x_2)dx_1dx_2 = \mathbb{E}_{x_1, x_2 \sim \gamma(x_1, x_2)} \|x_2 - x_1\|$$

Interpreting γ as a PDF



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

$$\int_{x_1} \gamma(x_1, x_2)dx_1 = Q(x_2), \quad \int_{x_2} \gamma(x_1, x_2)dx_2 = P(x_1)$$

Formal definition

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

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

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

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

Optimization over all transport plans – not too friendly

- ▶ Dual form (Kantorovich-Rubinstein duality):

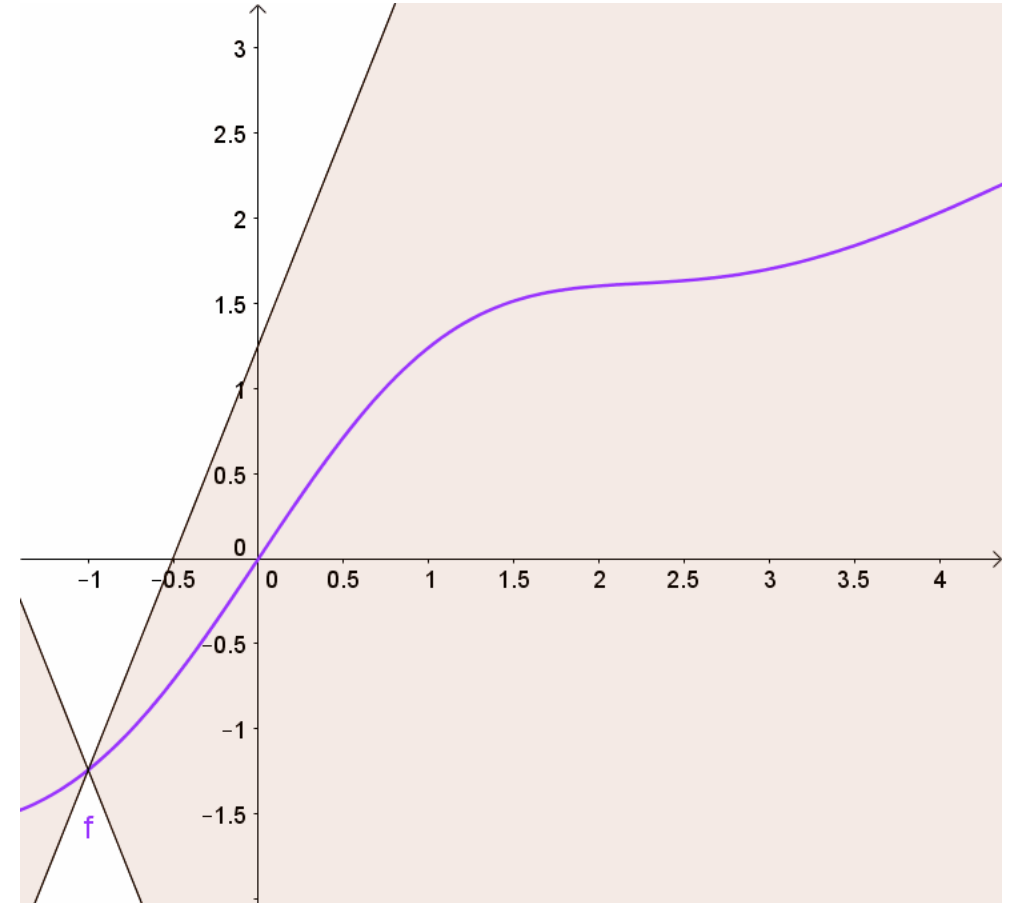
$$\text{EMD}(P, Q) = \sup_{\|f\|_L \leq 1} \left[\mathbb{E}_{x \sim P} f(x) - \mathbb{E}_{x \sim Q} f(x) \right]$$

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

Lipschitz continuity

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

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



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

[intuition behind the dual form]

disclaimer: not a strict
mathematical derivation

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

Let's add the following term to this expression:

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

$f(x)$ — real-valued function

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

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

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

Infimum and supremum operations can be swapped under certain conditions

(satisfied here — see <https://vincenthermann.github.io/blog/wasserstein/> for more detailed info)

[intuition behind the dual form]

**disclaimer: not a strict
mathematical derivation**

$$= \sup_f \inf_{\gamma} \left[\mathbb{E}_{s \sim P} f(s) - \mathbb{E}_{t \sim Q} f(t) + \underbrace{\mathbb{E}_{x_1, x_2 \sim \gamma} [||x_1 - x_2|| - (f(x_1) - f(x_2))]}_{\text{Consider the following case: } |f(a) - f(b)| \leq ||a - b||, \forall a, b}$$

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

For such case this term is 0

Otherwise the whole expression is $-\infty$

Therefore we can finally rewrite the whole thing as:

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

WGAN

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

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

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

WGAN

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

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

Require: : w_0 , initial critic parameters. θ_0 , initial generator's parameters.

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

WGAN-GP

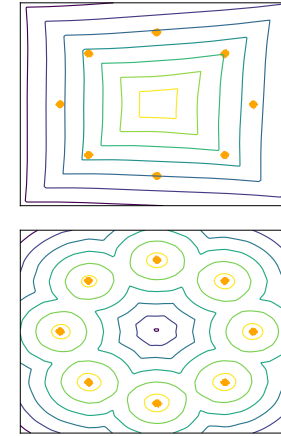
- ▶ Weight clipping makes the critic less expressive and the training harder to converge
- ▶ Optimal f should satisfy $\|\nabla f\| = 1$ almost everywhere under P and Q
- ▶ Also: $\|f\|_L \leq 1 \iff \|\nabla f\| \leq 1$
- ▶ Can replace weight clipping with a gradient penalty term:

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

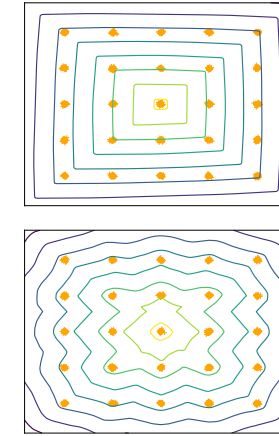
or alternatively ('one-sided' penalty):

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

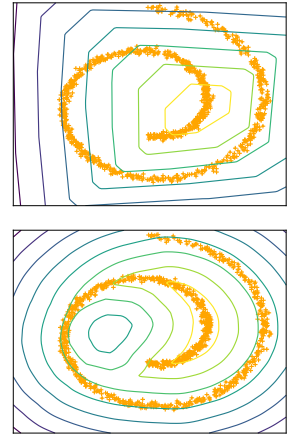
8 Gaussians



25 Gaussians



Swiss Roll



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

Sidenote

- ▶ There's an argument that the (true) Wasserstein distance might not be ideal for generative modelling
 - being a function of L2 norm of the difference vector (e.g. per-pixel difference between images)

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

- ▶ A curious reading:
 - J. Stanczuk et. al. Wasserstein GANs Work Because They Fail (to Approximate the Wasserstein Distance), <https://arxiv.org/abs/2103.01678>