CERN openlab

Generative Models in High Energy Physics

Sofia Vallecorsa

February 3rd, 2020

Conclusions

Generative Models are powerful techniques

- Object of an intense R&D , require careful **validation**
- **Many applications** in High Energy Physics

Simulation is one of the main applications

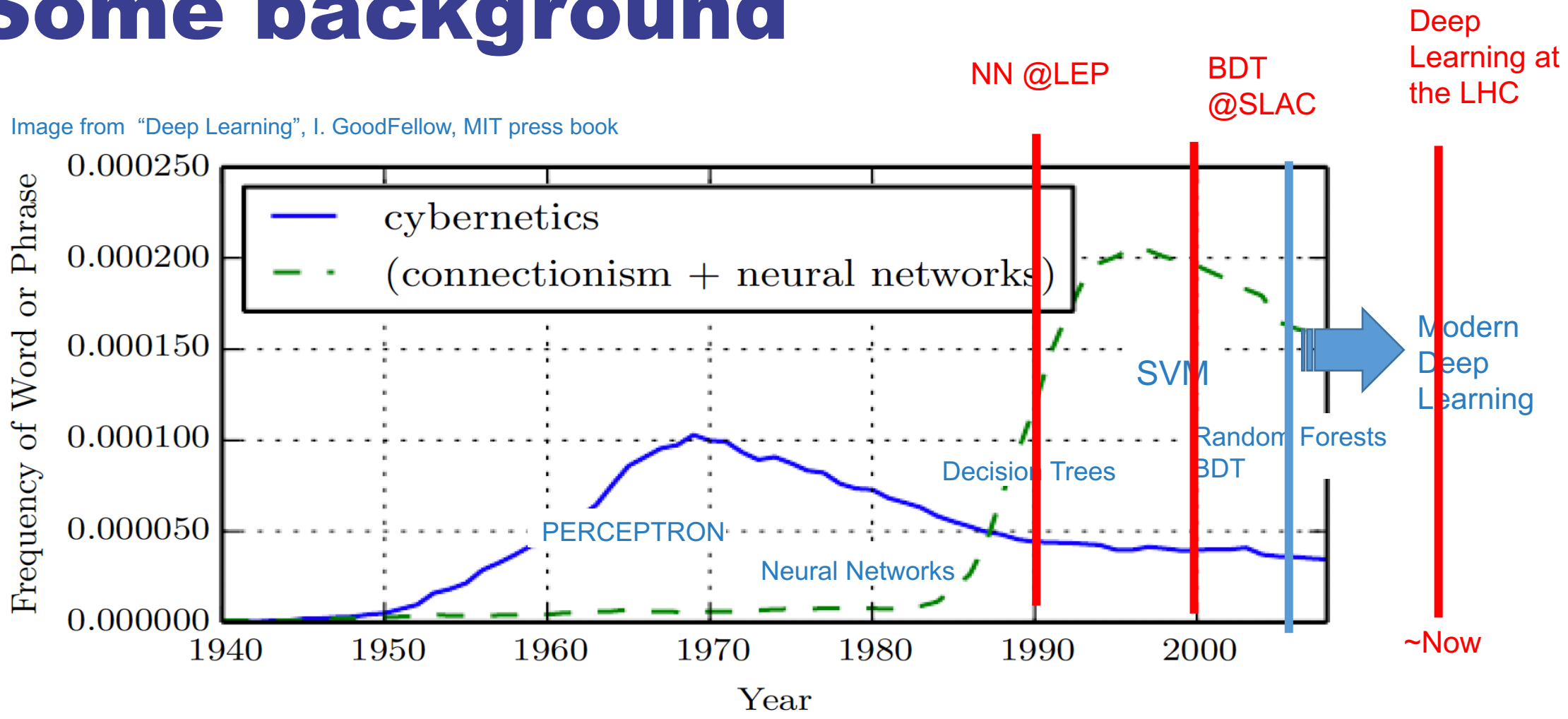
- High level of **accuracy** & **speed**
- **Multiple features/capabilities**

Deep Learning is a driving force toward changing our computing model

- DL workloads are HPC friendly
- Benefit from dedicated hardware and accelerators

Some background

Image from "Deep Learning", I. GoodFellow, MIT press book



Theories on biological learning
First linear models
Layer-wise pre-training through greedy algorithms

Back-propagation

Deep Learning in HEP

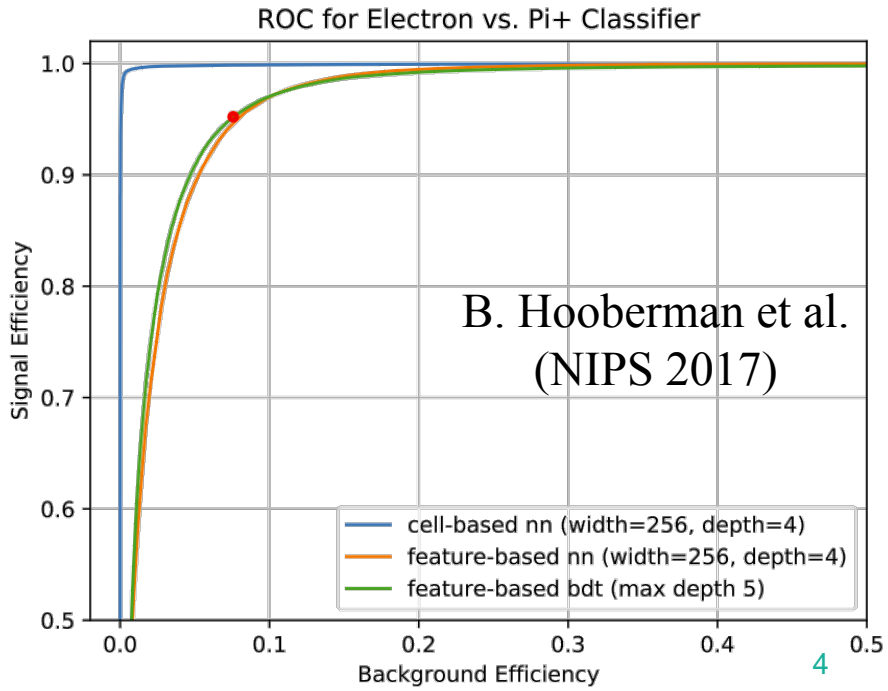
DL **recognize patterns** in large complicated data sets
- Better performances if applied directly to **raw data**

Re-cast physics problems as “DL problems”
- Detector output as **images** and apply **computer vision** techniques
- Physics events as **sentences** and apply NLP techniques

Intense R&D activity

Performance requirements
- Model **interpretability**
- Results **Validation** against classical methods
- Detailed **Systematics studies**

“New” **computing models**
- **Accelerators** and dedicated hardware
- **HPC** integration
- **Cloud** environment & **Big Data** platforms

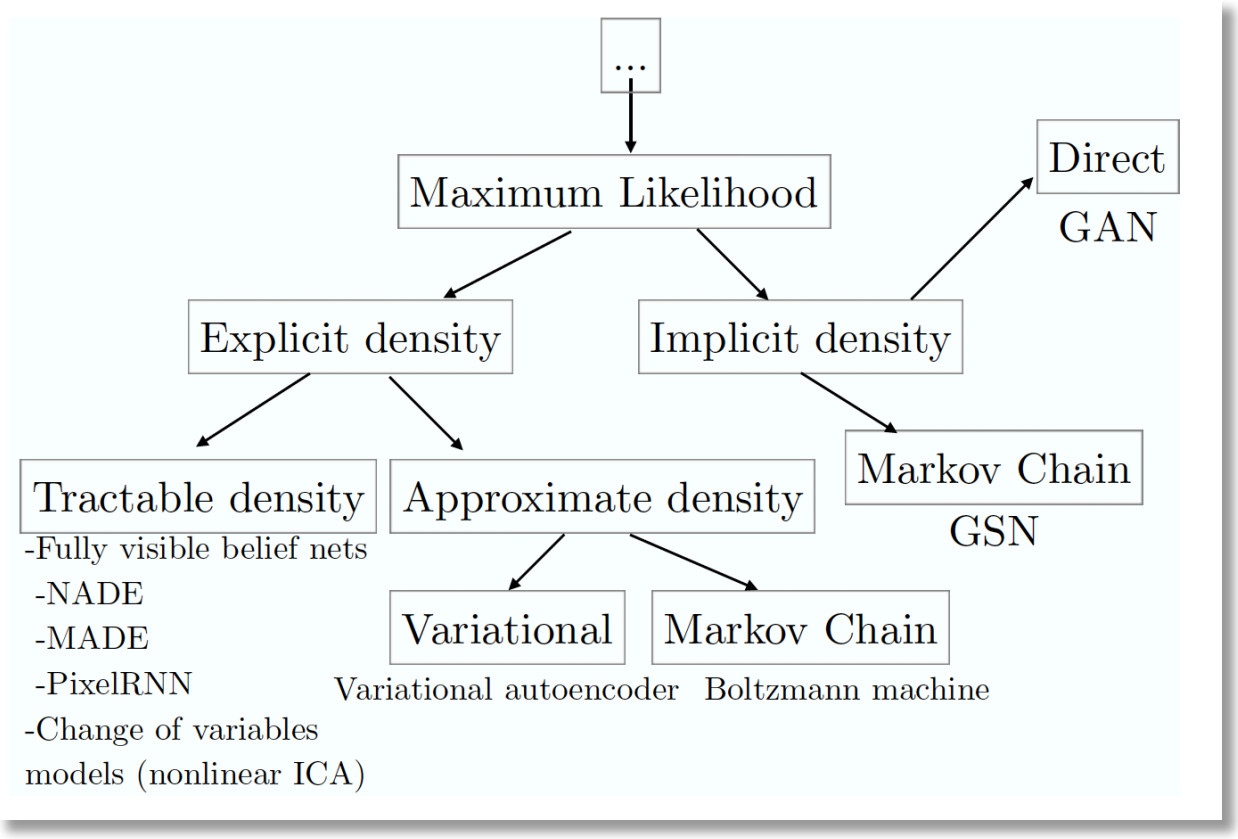


Generative Models

Basic concepts

Some references:

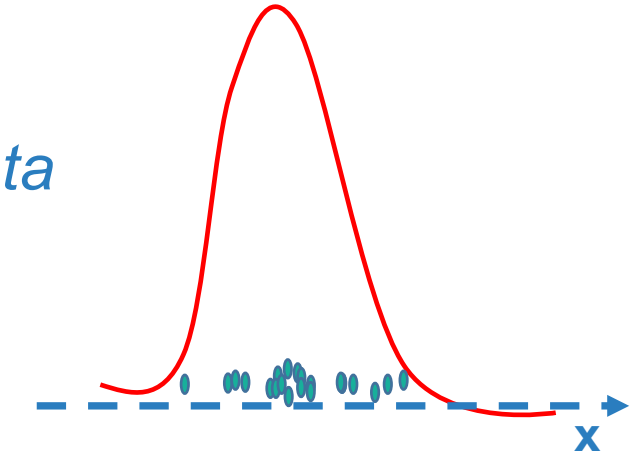
- NIPS 2016 Tutorial on Generative Adversarial Networks, I. Goodfellow
- 2018 Generative Models tutorial, D. Rezende
- 2019 Stanford course on Deep Generative Models



"GM Taxonomy" Image from NIPS 2016 Tutorial, I. Goodfellow

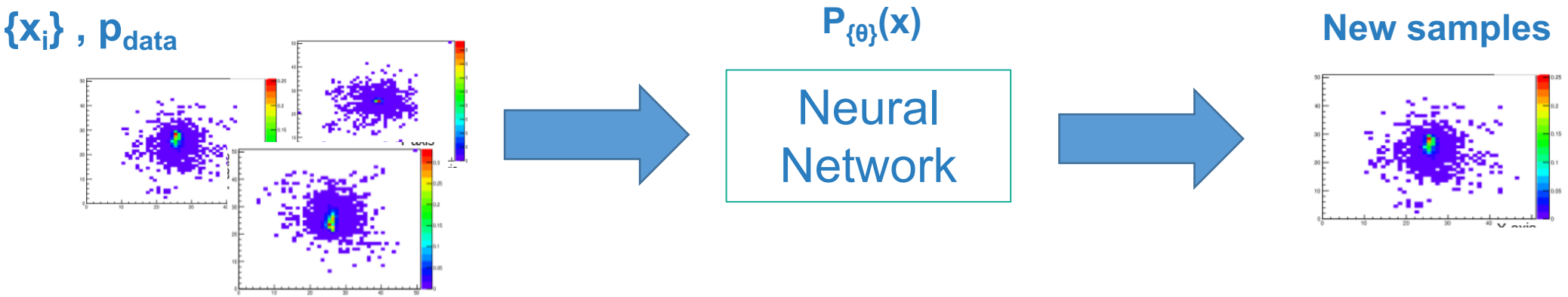
GM as Density Estimators

Generative models learn probability distributions from data



Given a collection of **samples x_i** and the underlying **p_{data} distribution**

Choose parameters {θ} so that **$p_{\{\theta\}}(x) \approx p_{data}$**



Prior knowledge?

*Generative models use *some* prior knowledge*

Extract meaningful **representations directly from data** but need:

- Loss functions
- Learning principle
- Optimisation algorithms
- Domain knowledge

Probability divergence D(p,q)

Name	Formula
f-divergences	$D(p;q) = \mathbb{E}_q[f(\frac{q}{p})]$
Relative entropy (KL)	$D(p;q) = \mathbb{E}_q[\ln \frac{p}{q}]$
Jensen-Shannon (JS)	$D(p;q) = \frac{1}{2}\mathrm{KL}(p;m) + \frac{1}{2}\mathrm{KL}(q;m)$
Stein divergence	$D(p;q) = \sup_f \mathbb{E}_q[\nabla \ln pf + \nabla f]^2$
Energy distance	$D(p;q) = \mathbb{E}[2\|x-y\| - \|x-x'\| - \|y-y'\|]$
Wasserstein distance	$D_\alpha(p;q) = [\inf_\rho \mathbb{E}_\rho[\|x-x'\|^\alpha]]^{\frac{1}{\alpha}}$
Max-min dis. (MMD)	$D(p;q) = \sup_f(\mathbb{E}[f]_p - \mathbb{E}[f]_q)$

D. Rezende: https://danilorezende.com/2018/07/12/short-notes-on-divergence-measures/

Multiple Applications

Event sampling (**Simulation**)

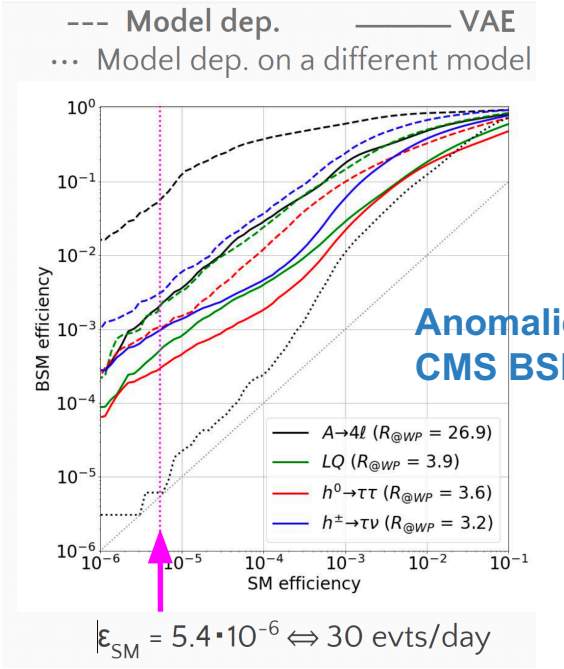
Find Underlying Factors (**Discovery**)

Detect Rare events (**Anomaly Detection**)

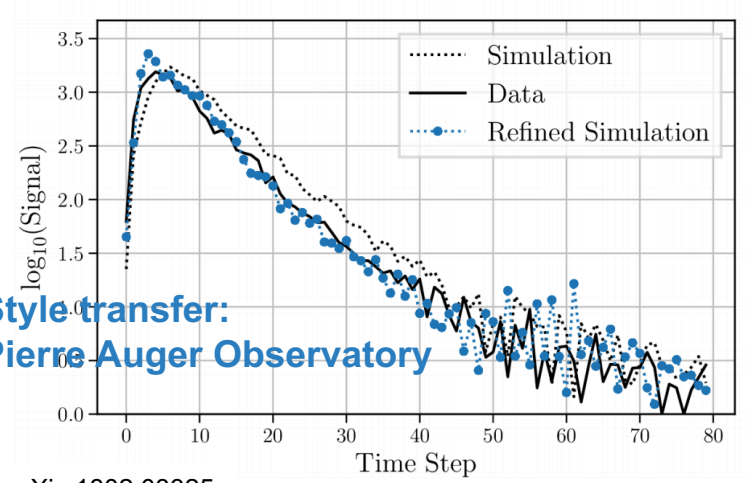
Predict future events (**Planning**)

Find Analogies (**Transfer Learning & Style Transfer**)

O. Cerri, ACAT2019, arXiv:1811.10276

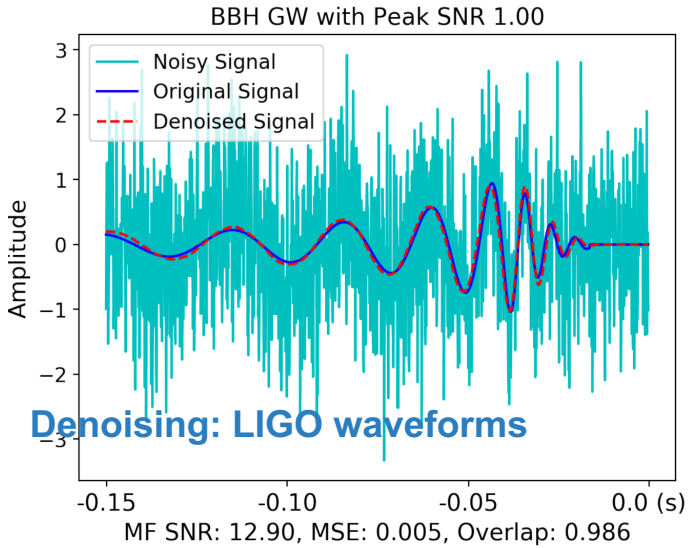


Anomalies:
CMS BSM trigger



Style transfer:
Pierre Auger Observatory

arXiv:1802.03325



Denoising: LIGO waveforms

Shen H, 2017 (Preprint 1711.09919)

(Deep) Generative Models

Internal representations learned by shallow systems are simple (Bengio & LeCun 2007, Bengio 2009)

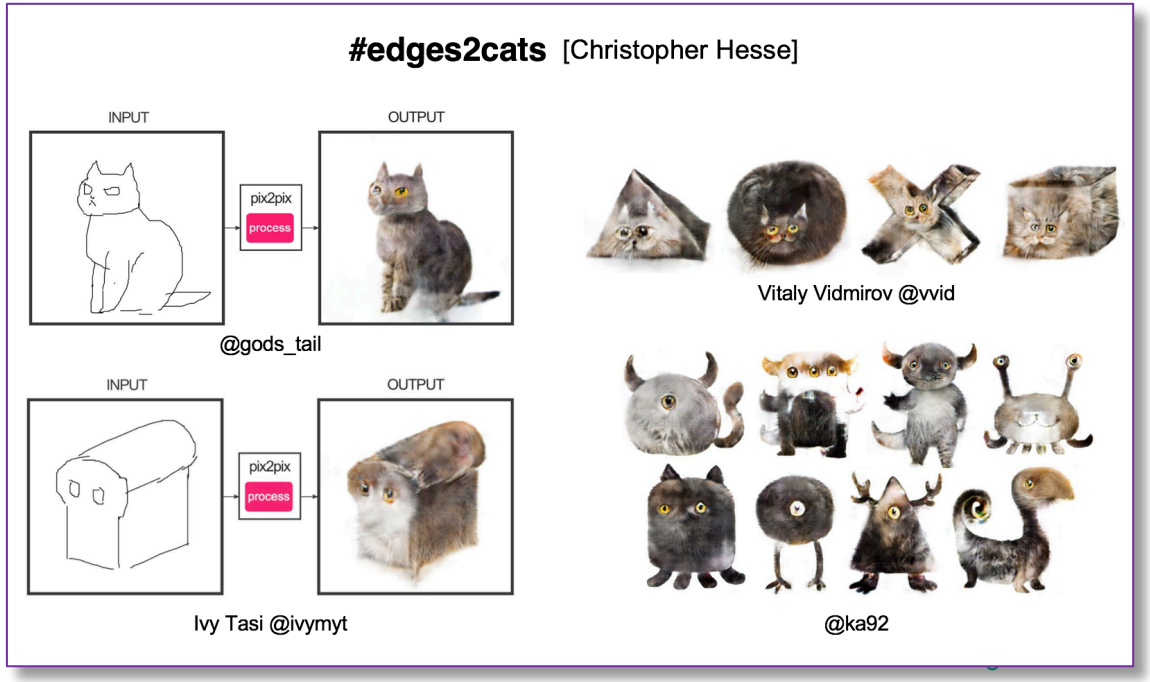
- Incapable of learning complex hidden structures
- Large amounts of labeled data

→ Deep Generative Models

- → **Higher levels of abstraction**
- → Improved **generalisation** and **transfer**

→ Categorized according to

- → **Feature** representation
  - → **Fully observed**, or **latent variables** based
- → **Learning principles**
  - → **MLE**, **variational**, or **likelihood-free**

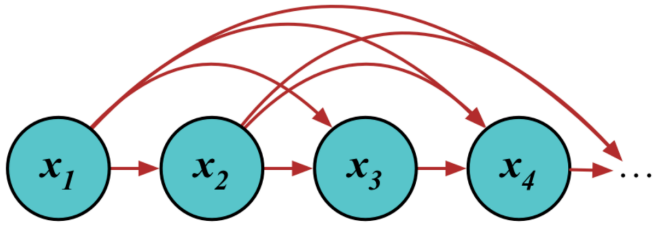


CERN openlab

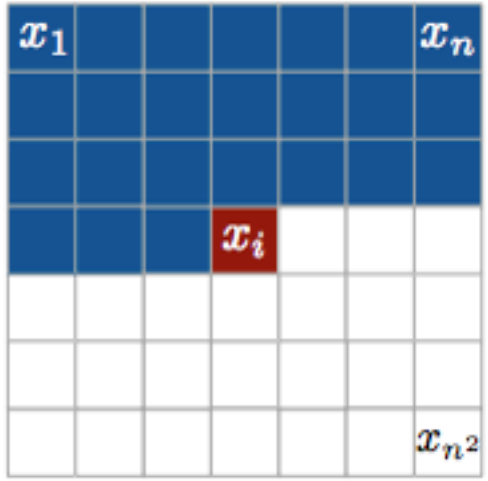
Fully Observed Models

Directly observe data without introducing new local (latent) variables

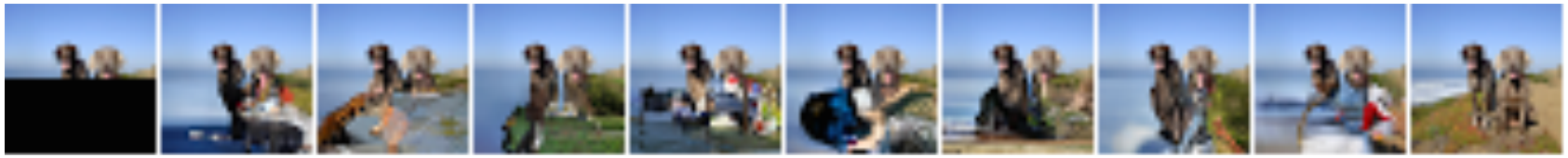
$$p(x_{1,\ldots,N}) = \prod_{i=1}^{N} p(x_i | x_{1,\ldots,(i-1)})$$



Ex. Pixel Recurrent Neural Networks:



$$p(x_t \mid x_{1:t-1}) = p(x_t^{red} \mid x_{1:t-1}) p(x_t^{green} \mid x_{1:t-1}, x_t^{red}) p(x_t^{blue} \mid x_{1:t-1}, x_t^{red}, x_t^{green})$$



https://arxiv.org/pdf/1601.06759.pdf

Latent Variables models

Unsupervised representation learning

Introduce an **unobserved random variable** for every observed data point to **explain features**.

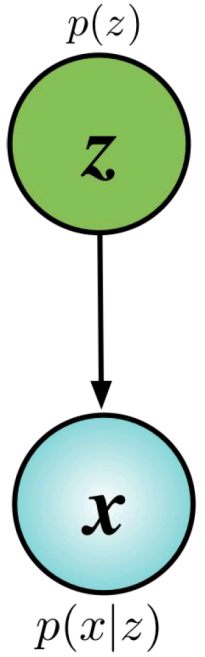
Prescribed models: Use observer likelihoods and assume observation noise.

Implicit models: Likelihood-free models.

$$x \in \mathbb{R}^{d_x} \quad z \in \mathbb{R}^{d_z} \quad \theta \in \mathbb{R}^{d_\theta}$$
$$\mathcal{D} = \{x_i\} \quad i \in \{1, ..., N\}$$

$$\log p_\theta(x) = \log \int p_\theta(x|z)p(z)dz = \log \mathbb{E}_{p(z)}[p_\theta(x|z)]$$

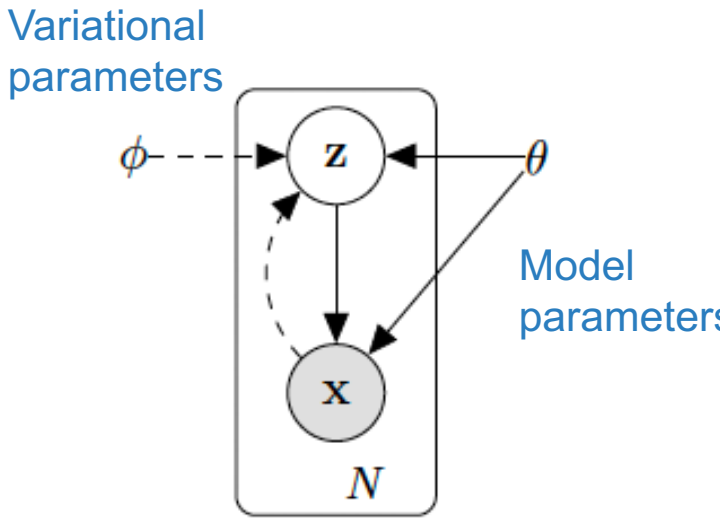
$$\log p_\theta(\mathcal{D}) = \sum_{i=1}^{N} \log \mathbb{E}_{p(z)}[p_\theta(x_i|z)]$$



CERN openlab

Variational AutoEncoders

Describe training dataset in latent space

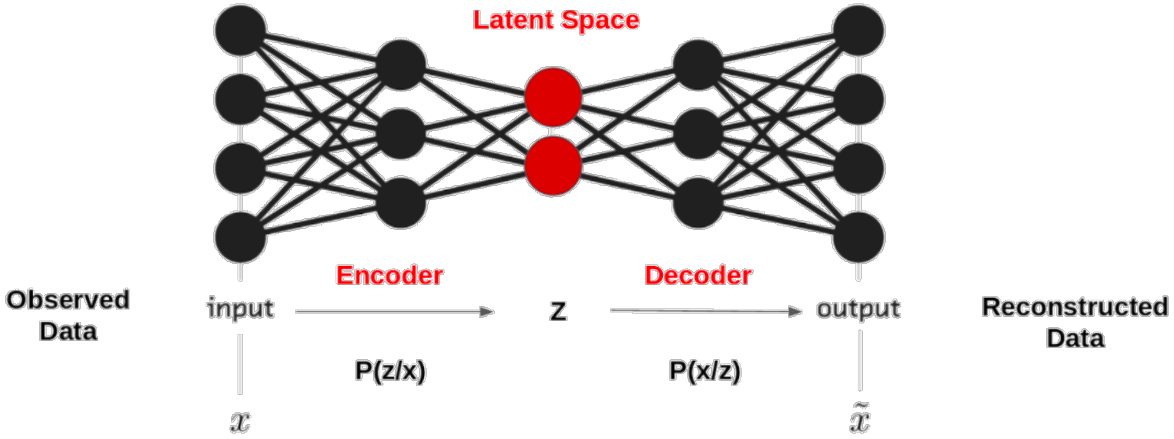


Explicit constraints on the encoded representations

- Learn the **latent variable distribution**

Two components in the loss function

- **reconstruction loss**
- **KL divergence** between the learned latent distribution and the prior regularization



https://arxiv.org/abs/1312.6114

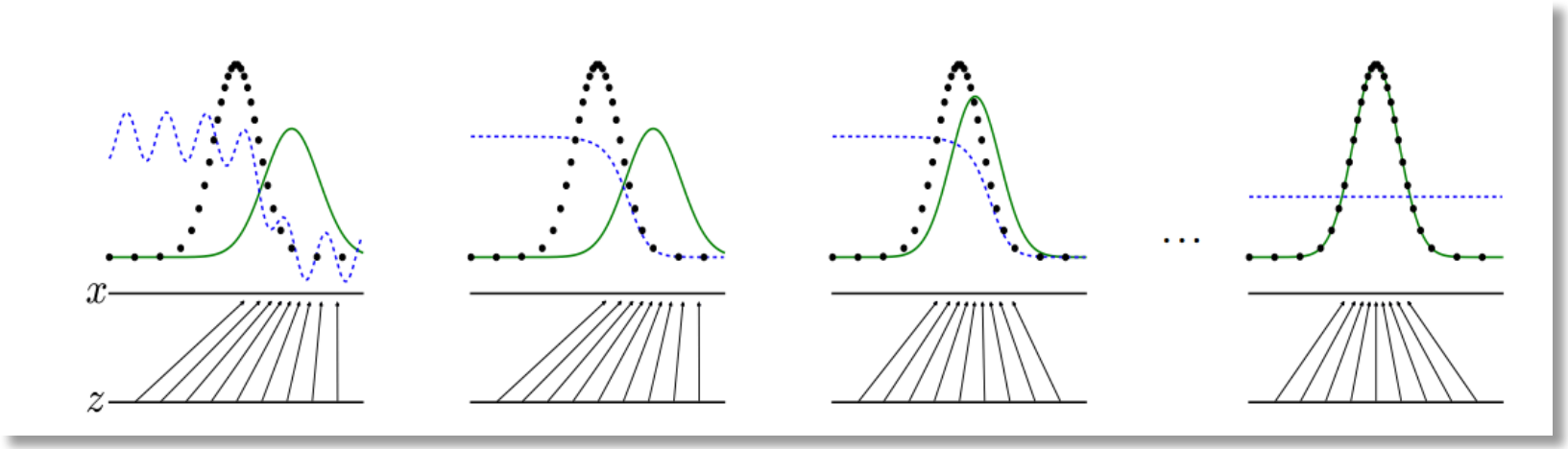
Likelihood-free learning

Density estimation by comparison

Compare the **estimated distribution** q(x) to the **true distribution** p*(x) using samples.

Build an **auxiliary model** to indicate how data simulated from the generative model differs from observed data.

Adjust model parameters to better match the data distribution



https://arxiv.org/pdf/1406.2661.pdf

Adversarial training

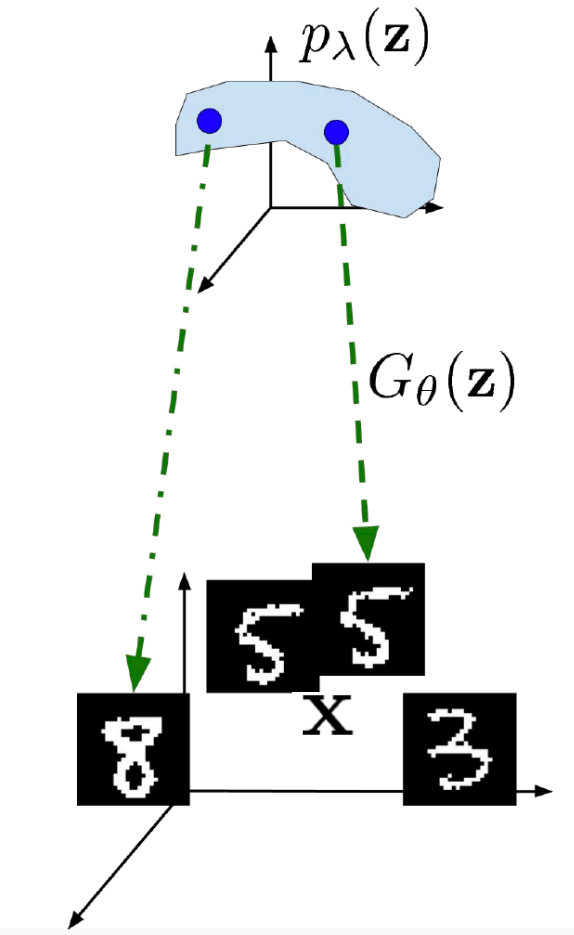
Assume a deterministic generator: $\mathbf{x} = G_\theta(\mathbf{z})$

A prior over latent space: $\mathbf{z} \sim p_\lambda(\mathbf{z})$

Define a discriminator: $D_\psi(\mathbf{x}) \in [0, 1]$

A learnable loss function from the min-max game

$$\min_\theta \max_\psi \mathbb{E}_{\mathbf{x}\sim p_{data}}\Big[\ln D_\psi(\mathbf{x})\Big] - \mathbb{E}_{\mathbf{z}\sim p_\lambda(\mathbf{z})}\Big[\ln\big(1 - D_\psi(G(\mathbf{z}))\big)\Big]$$



$$\min \max \quad \mathbf{E}_{x\sim\mathcal{D}_{real}}[D_\psi(x)] - \mathbf{E}_h[D_\psi(G_\theta(h))]$$

Wasserstein loss

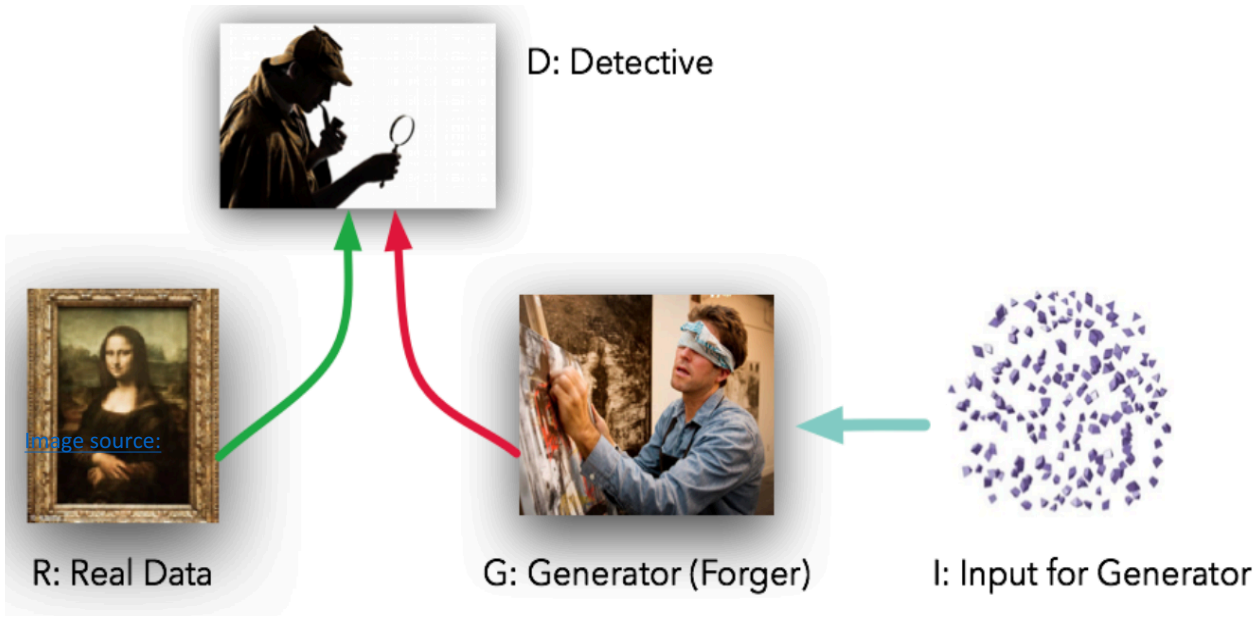
arXiv:1406.2661v1

Generative adversarial networks

Simultaneously train two networks that compete and cooperate with each other

Generator G generates data from random noise

Discriminator D learns how to distinguish real data from generated data



https://arxiv.org/pdf/1701.00160v1.pdf

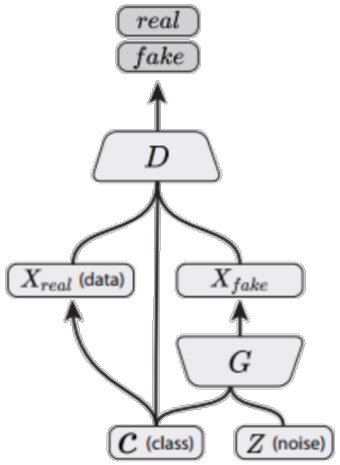
Multiple GAN flavors

Different base layers technologies:

- Original GAN was based on MLP in 2014
- Deep Convolutional GAN in 2015
- Graph GAN
- Recurrent -GAN

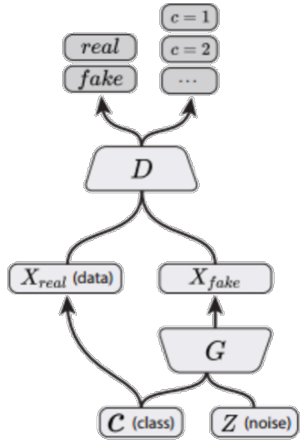
Different "information paths"

- Conditional GAN
  - Extended to learn a parameterized generator $p_{model}(x|\theta)$
  - Useful to obtain a single generator object for all θ configurations
  - Interpolate between distribution
- Auxiliary Classifier GAN
  - D can assign a class to the image
- Progressive growing GAN
- VAE-GAN
- Stack GAN
- BiGAN



Conditional GAN
(Mirza & Osindero, 2014)

arXiv: 1411.1784



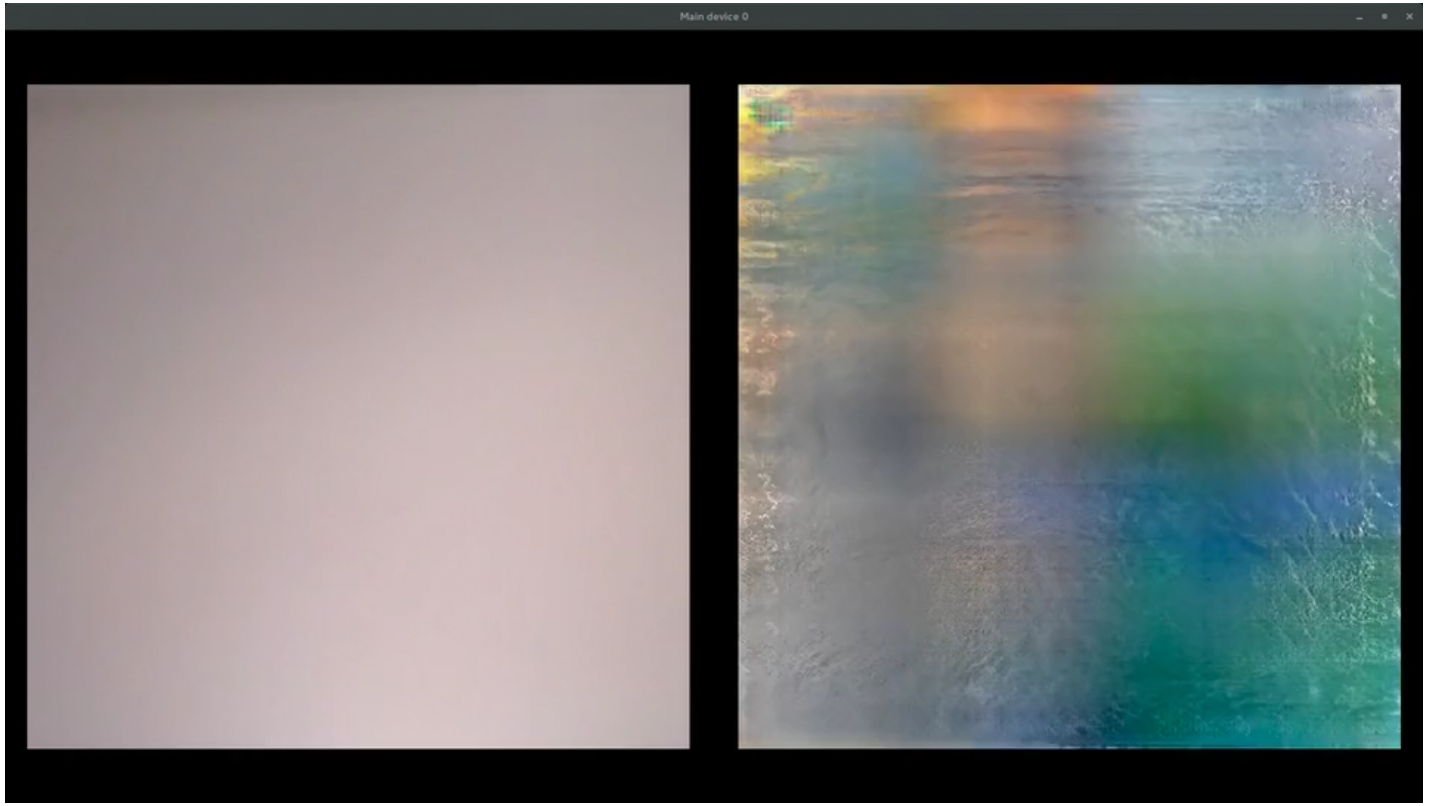
AC-GAN
(Present Work)

arXiv:1610.0958

https://thispersondoesnotexist.com/

GAN for Art?

Image-to-Image Translation with Conditional Adversarial Networks

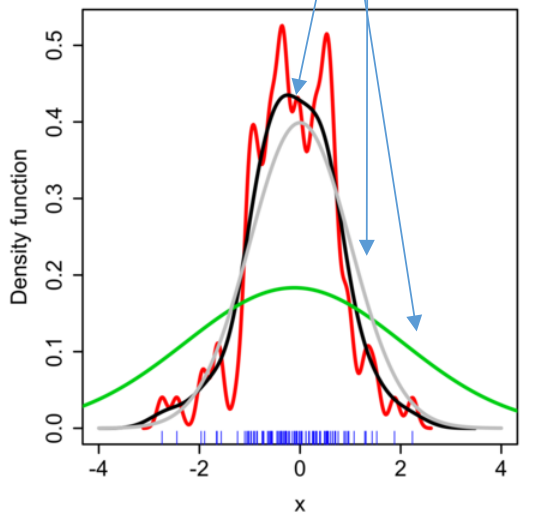


http://www.memo.tv/portfolio/learning-to-see/ https://github.com/phillipi/pix2pix

Evaluating performance is tricky

- Different applications
  - Density estimation
  - **Sampling/generation**
  - Latent representation learning
  - Custom task (i.e. image translation, compressions..)
- Tractable likelihood models
  - Split dataset into train, validation, test sets
  - Evaluate gradients on **train** set
  - Tune hyperparameters on **validation** set
  - Evaluate generalization by reporting likelihoods on **test** set
- Non-tractable likelihood or likelihood-free models
  - Use **lower bounds** or **approximations** (I,e ELBO, KDE)

$$\hat{p}(\mathbf{x}) = \frac{1}{n} \sum_{\mathbf{x}^{(i)} \in \mathcal{S}} K\left(\frac{\mathbf{x} - \mathbf{x}^{(i)}}{\sigma} \right)$$



https://deepgenerativemodels.github.io/syllabus.html

GAN for simulation

- Measure **convergence**
  - Evaluate difference between model PDF and real PDF
- At convergence evaluate **sample quality**
  - Mixing and coverage (diversity)
  - Saliency
  - Mode collapse or mode dropping
  - Overfitting
- Need quantities that are **invariant** to small translation, rotation, intensity changes
  - Simple pixel space Euclidean distances don’t work
  - Define a way to **map input into a feature space**
    - Inception score
    - Maximum Mean Discrepancy
    - Fréchet Inception Distance
    - Structural Similarity Index

+ Physics Quantities Validation

Generated support space

Size of the dataset

GANs produce distributions with limited support

Support size grows ~linearly with discriminator size

- Training dataset size does not help much for a given discriminator

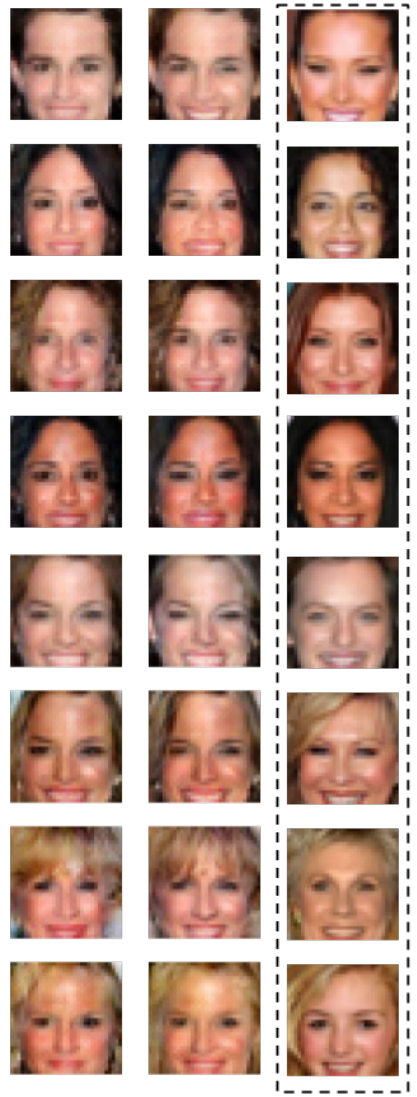
Example: BiGAN

- support size is around 1M (training set ~200k)

Birthday paradox test*

- If a sample of size **s** has near-duplicate images with prob > 1/2, then distribution has only $\mathbf{s^2}$ distinct images.

Search for nearest neighbor



DCGAN

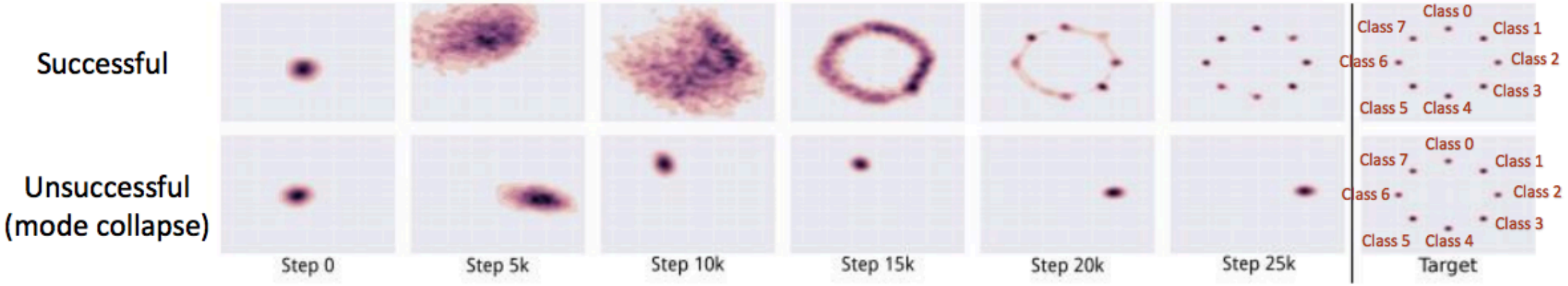
*Zhang A., ICML'17

Birthday Paradox: Suppose a distribution is supported on N images. Then P[sample of size √N has a duplicate image] > ½.

One extreme case: Mode collapse

Goal is to generate fake examples imitating real sample

Simple solution is to just generate easy modes (classes).



Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks (2016).

P. Grnarova, NeurIPS 2019

Duality Gap

A natural metric in minimax games

For a given set of Dicriminator and Generator: $(u^*, v^*) \in K_1 \times K_2$

$$\mathrm{DualGap} := \max_{\mathrm{v} \in \mathrm{K}_2} \mathrm{M}(\mathrm{u}^*, \mathrm{v}) - \min_{\mathrm{u} \in \mathrm{K}_1} \mathrm{M}(\mathrm{u}, \mathrm{v}^*)$$

Fix generator, find most adversarial discriminator

Fix discriminator, find most adversarial generator

- Obtaining an unbiased estimate

1. Train a GAN and obtain (G^*, D^*)
(use a training set)

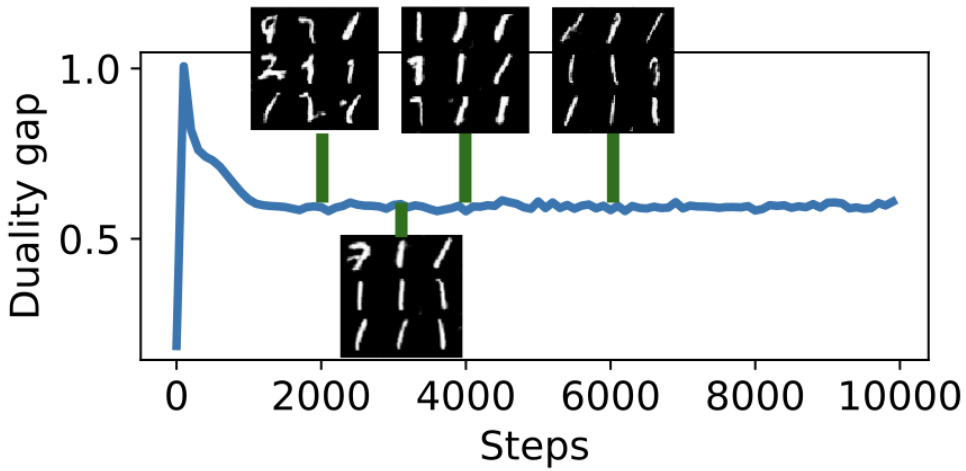
2. Find the worst D and G i.e.

$$G_{\text{worst}} \leftarrow \arg\min_G M(G, D^*) \quad D_{\text{worst}} \leftarrow \arg\max_D M(G^*, D)$$

(use an adversary finding set)

3. Compute: $\mathrm{DG} := \mathrm{M}(\mathrm{G}^*, D_{\text{worst}}) - M(G_{\text{worst}}, D^*)$
(use a test set)

Mode Collapse detection



Conclusions

Generative Models are powerful techniques

- Object of an intense R&D , require careful **validation**
- **Many applications** in High Energy Physics

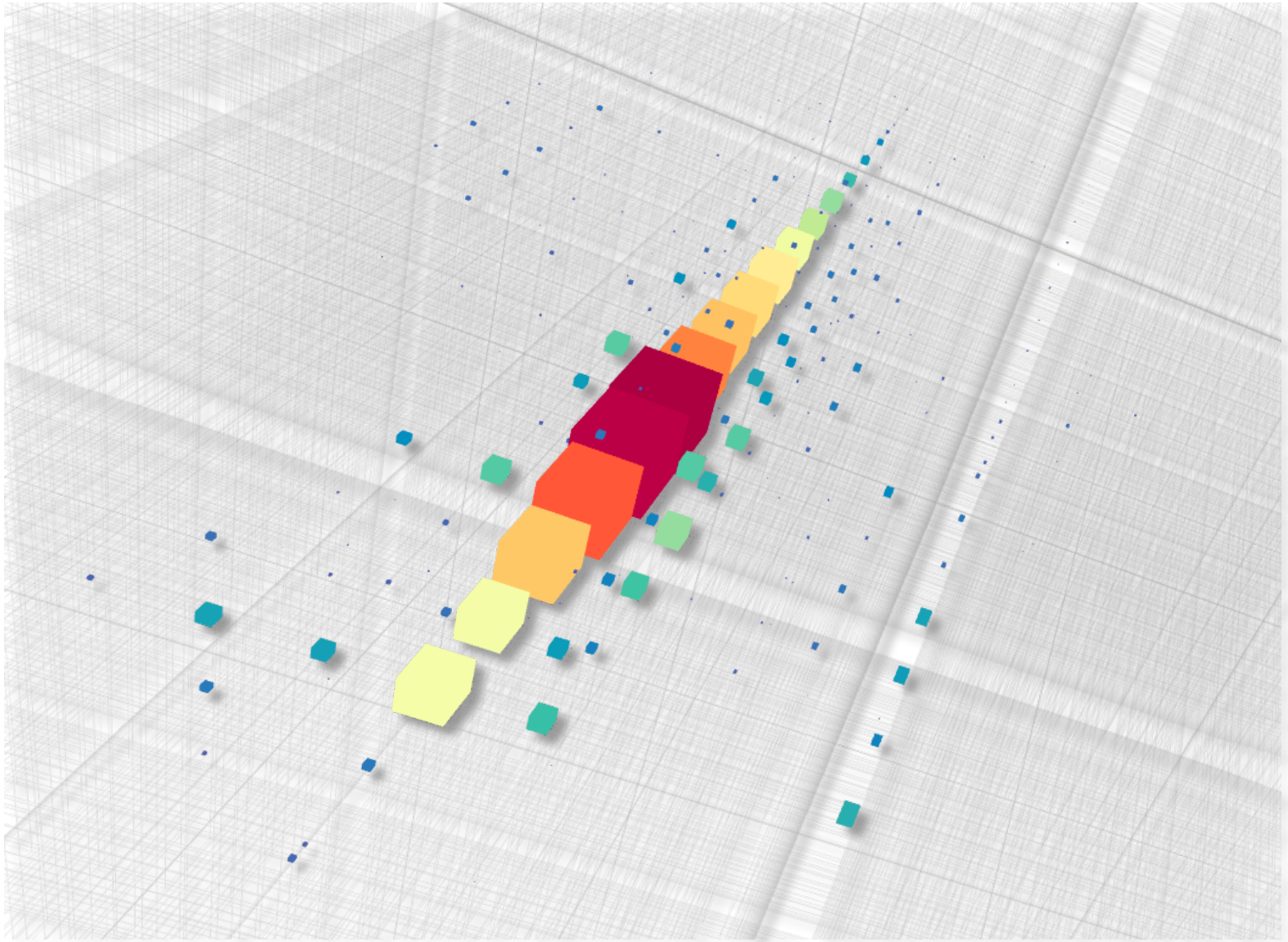
Simulation is one of the main applications

- High level of **accuracy** & **speed**
- **Multiple features/capabilities**

Deep Learning is a driving force toward changing our computing model

- DL workloads are HPC friendly
- Benefit from dedicated hardware and accelerators

3DGAN



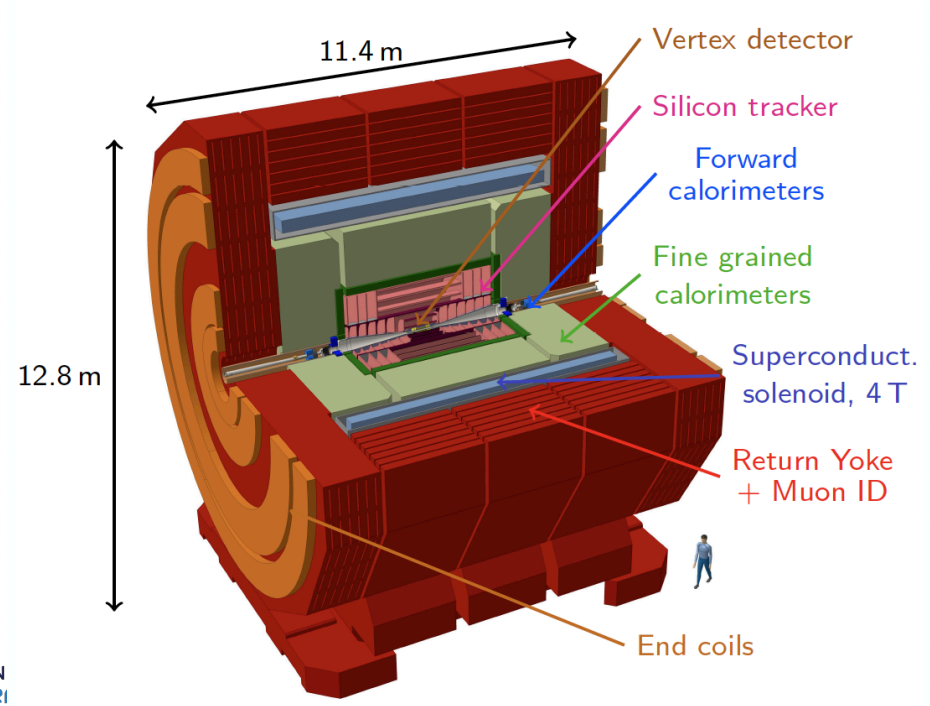
https://openlab.cern/project/fast-simulation

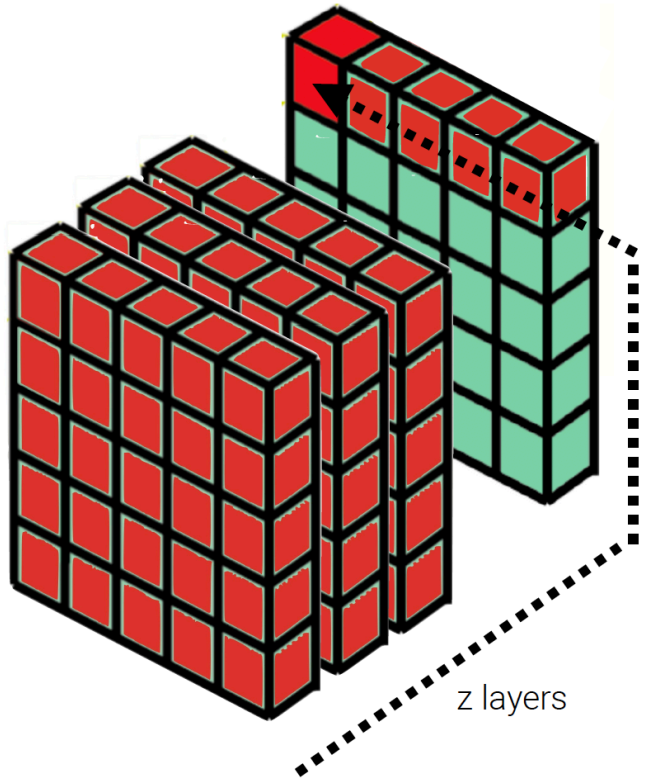
Compact LInear Collider

High-luminosity linear e+e- collider

Three energy stages up to 3 TeV

Associated detector studies





Electromagnetic calorimeter detector design

1.5 m inner radius

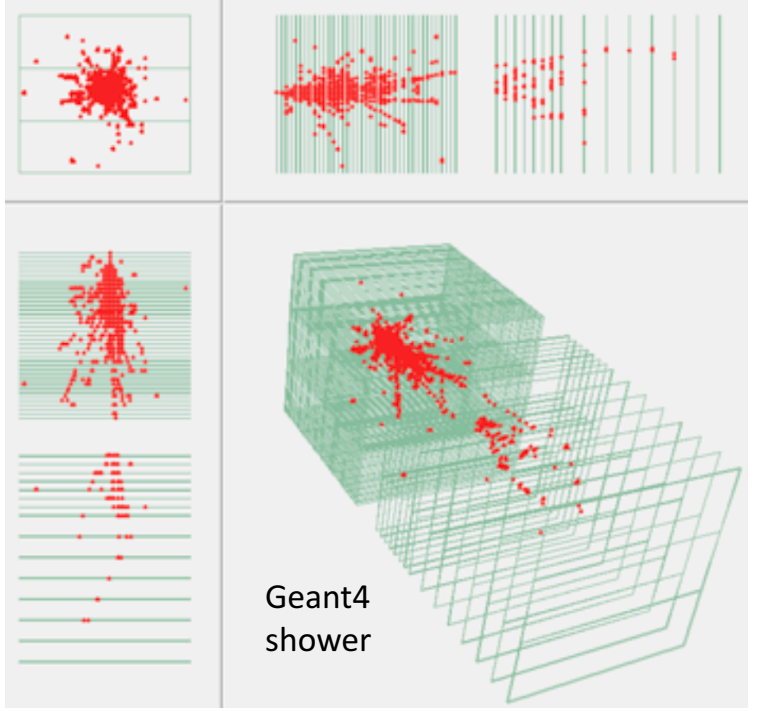
5 mm×5 mm segmentation

25 tungsten absorber layers + silicon sensors

http://cds.cern.ch/record/2254048#

High Granularity Calorimeter

Open data set developed for ML applications [1]



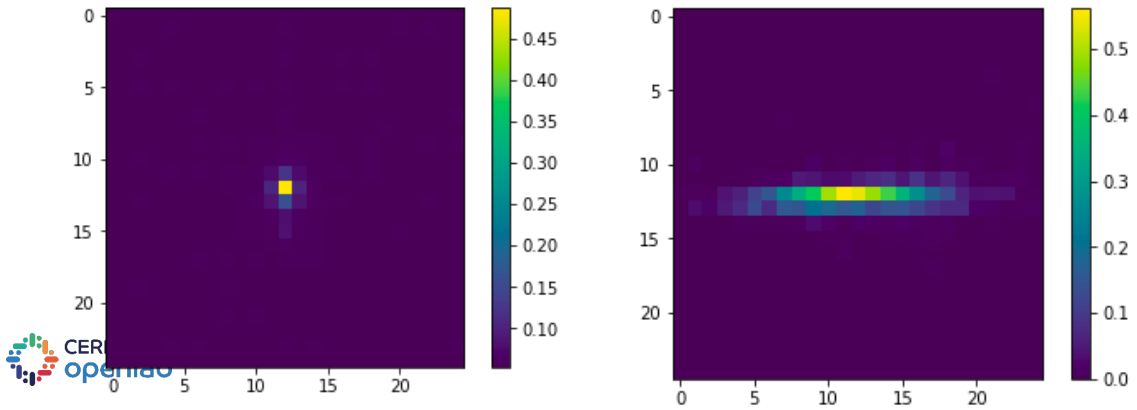
Pixelized datector volume to fully contain shower

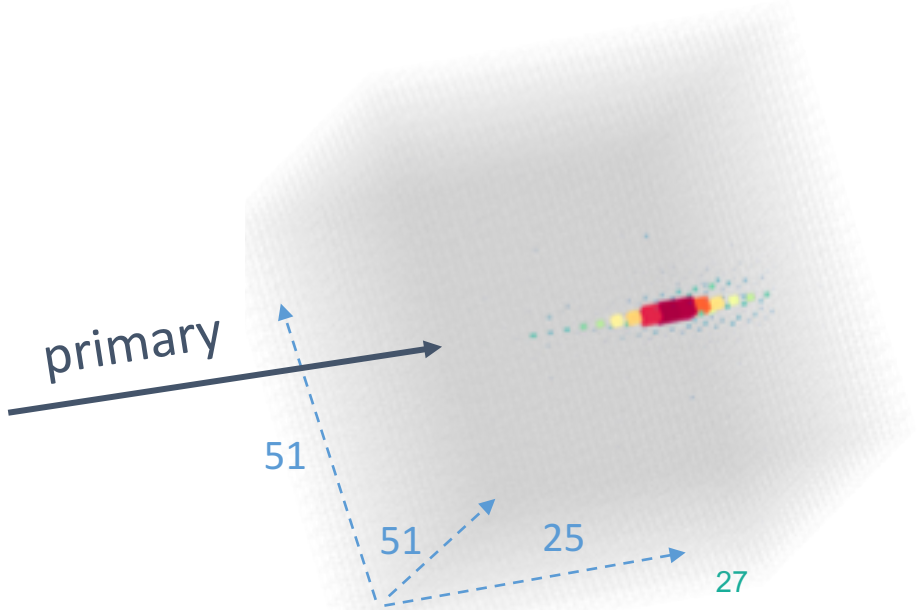
1M single particle samples (e,γ,π) with flat energy spectrum (10-500) GeV

α=+/- 30° random incident angle

Detector response as 3D images (51x51x25 pixels)

Large dynamic range, Sparsity





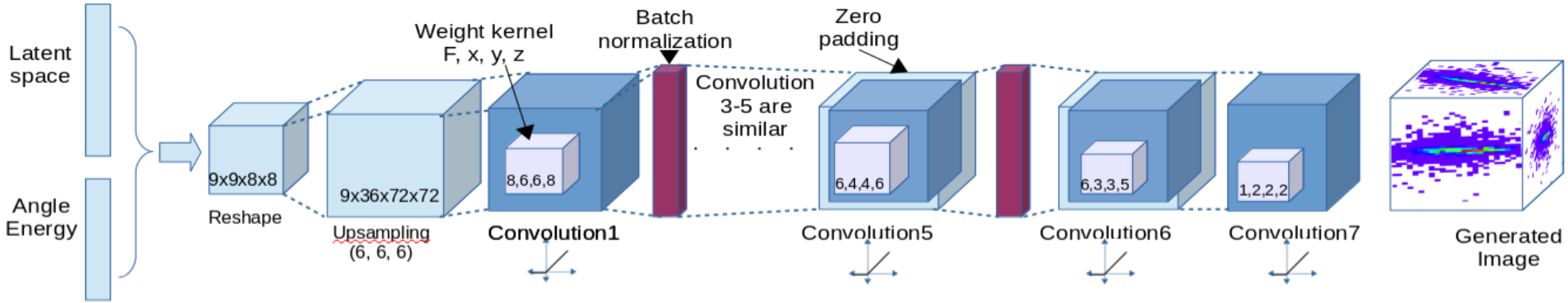
[1] M. Pierini et al. NIPS 2017

3D Generative Adversarial Networks

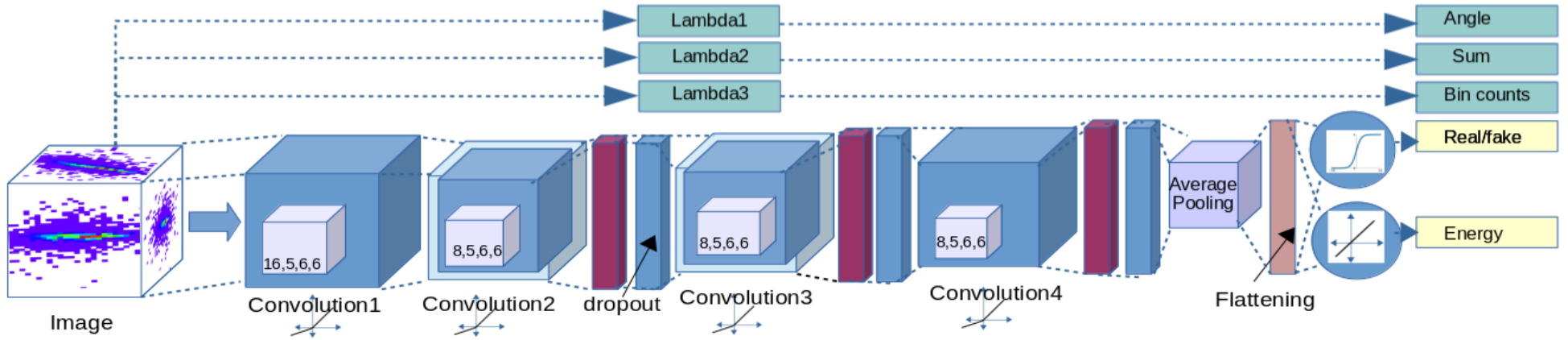
Condition training on input variables, **Custom losses**

Auxiliary regression tasks assigned to the discriminator

Generator:



Discriminator:

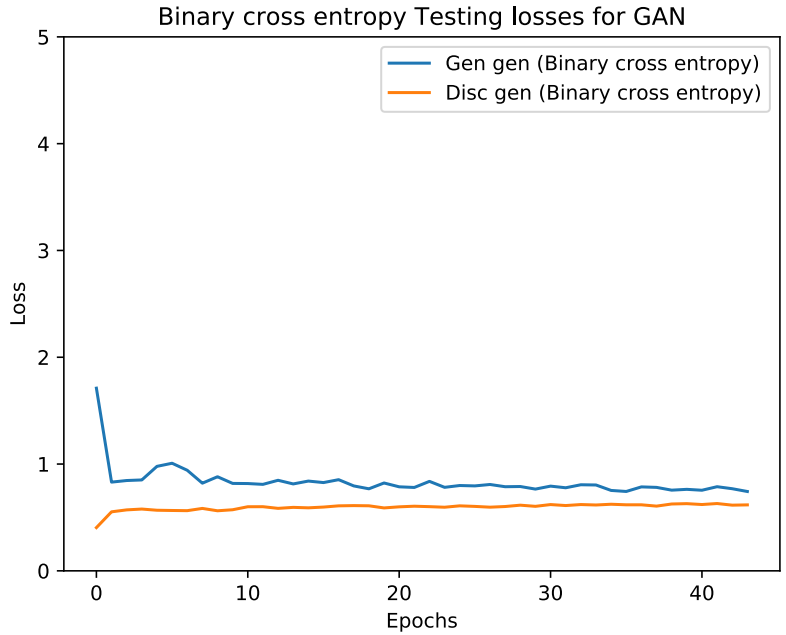


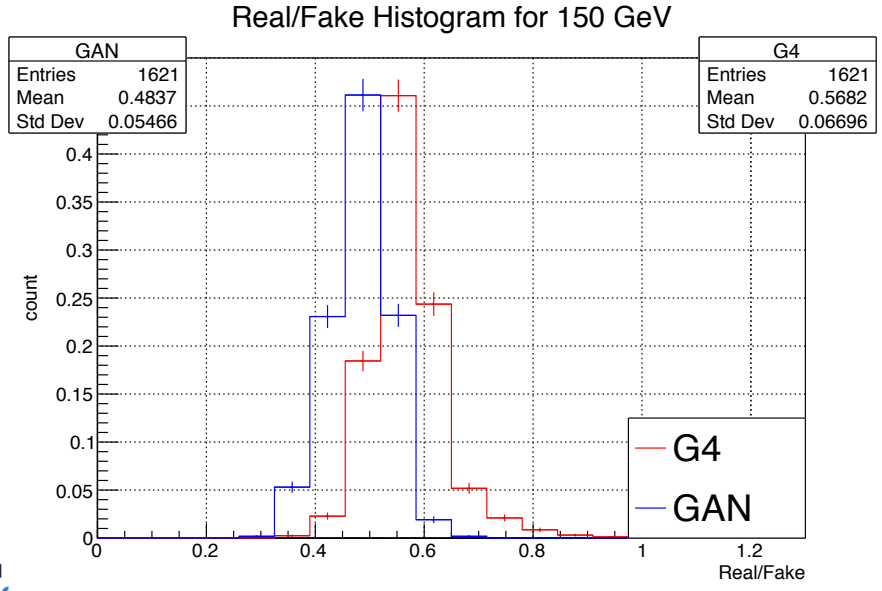
CERN openlab

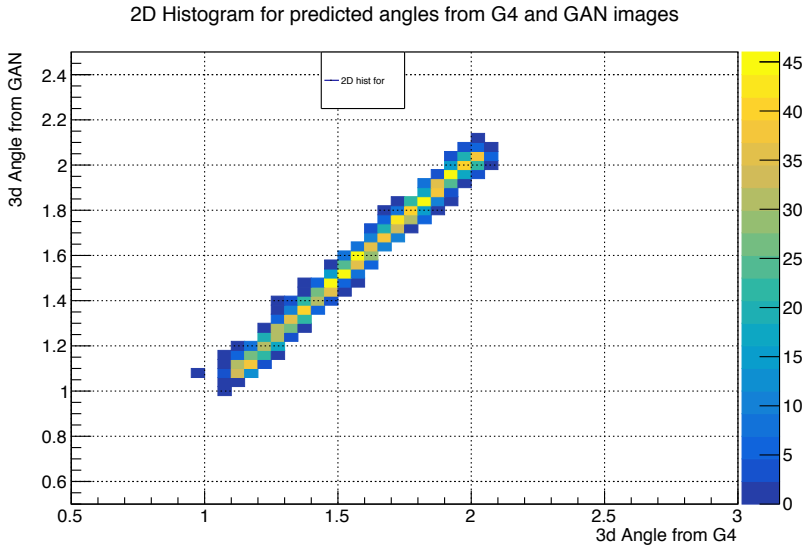
Convergence

and discriminator performance

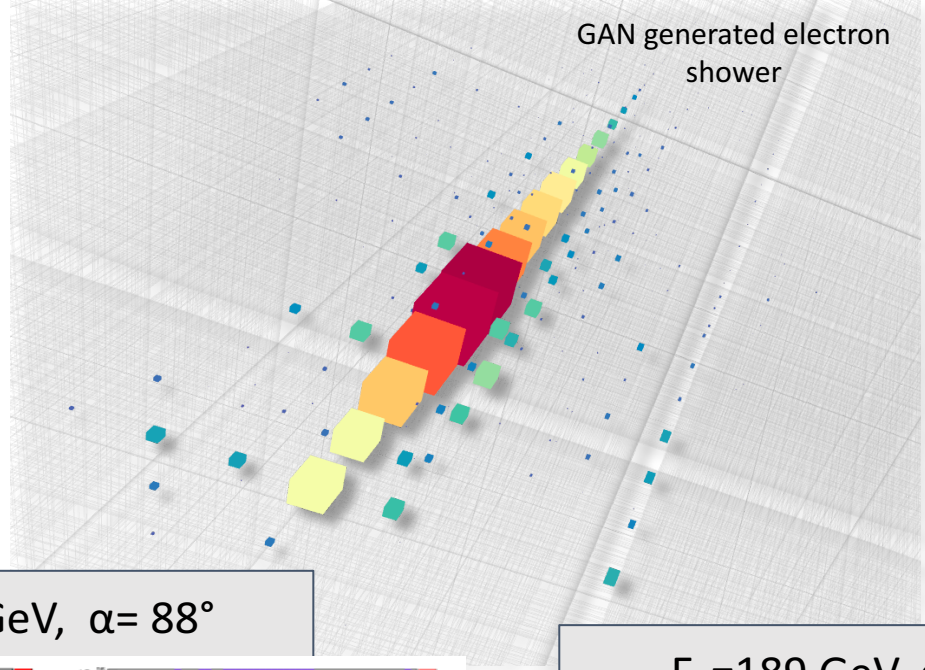
- Stable test loss
- Discriminator Real/Fake probability peaks at ~50%
- Correct incident angle



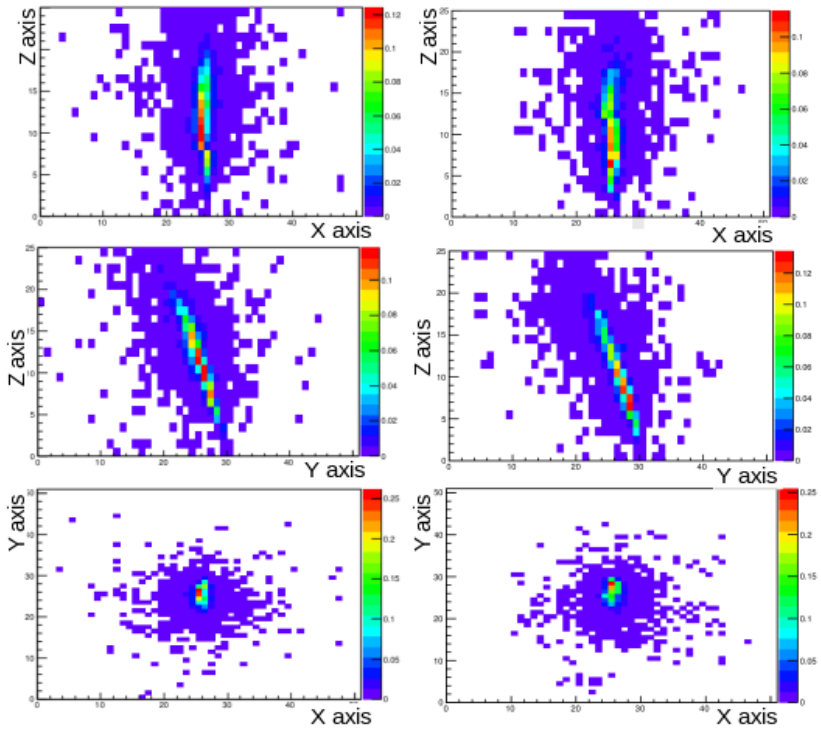




Generated events

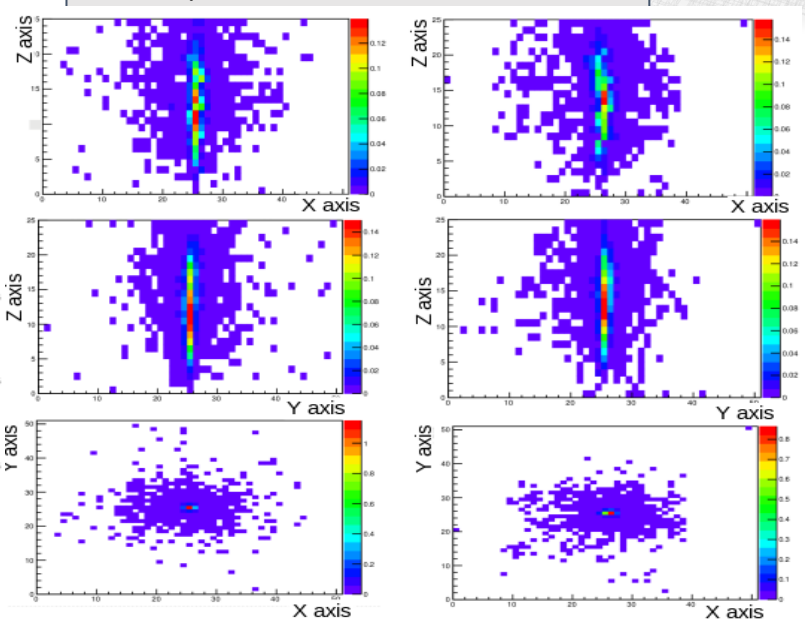


E_p=111 GeV, α=115°



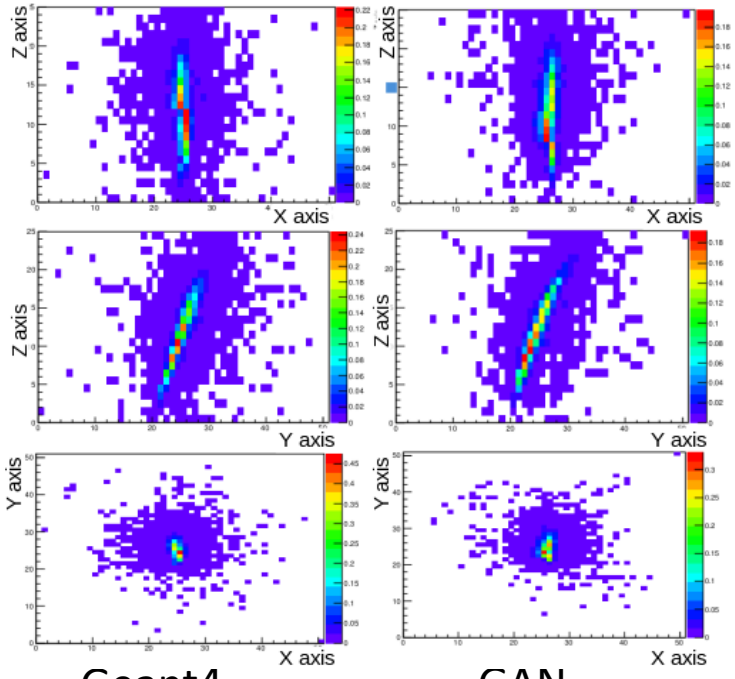
Geant4 GAN

E_p=147 GeV, α= 88°



Geant4 GAN

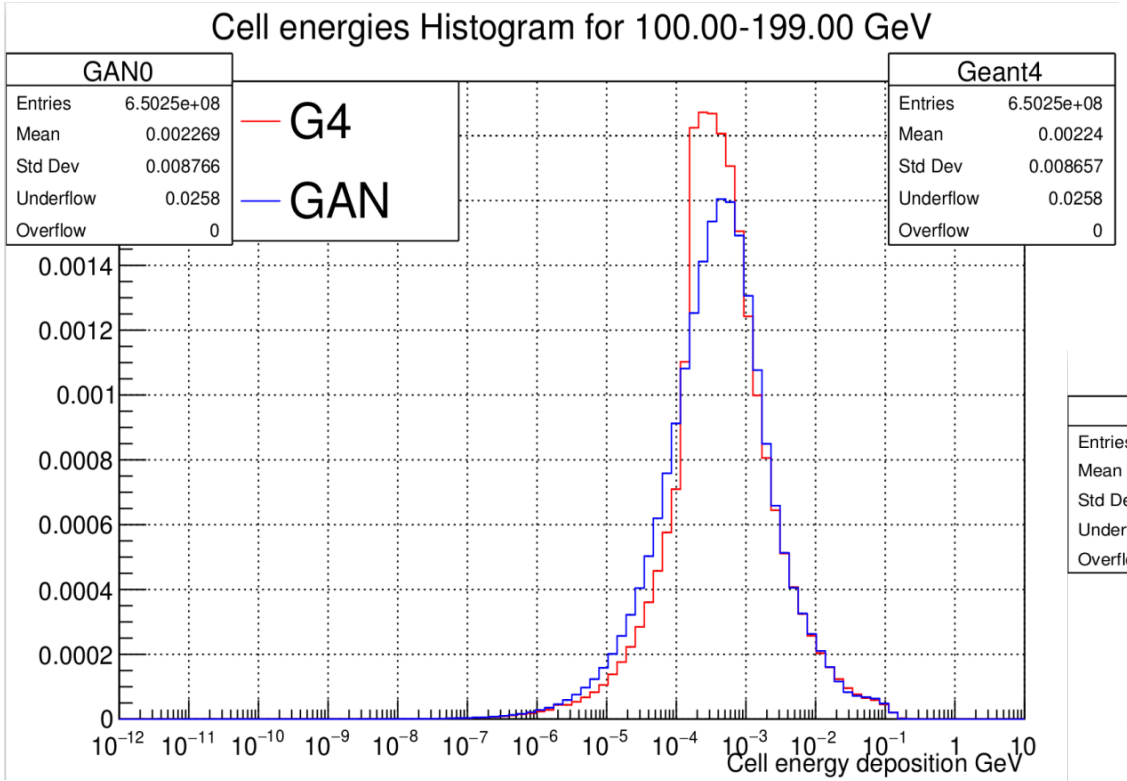
E_p=189 GeV, α= 63°

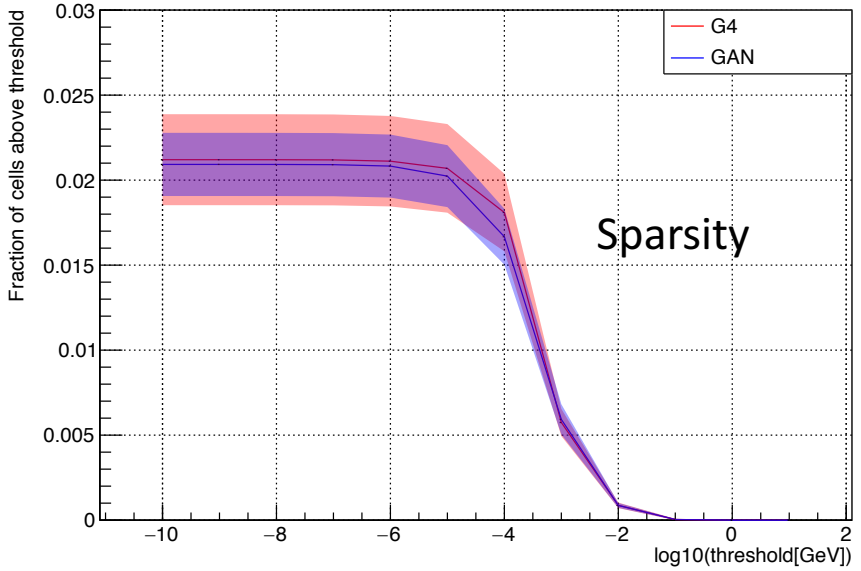


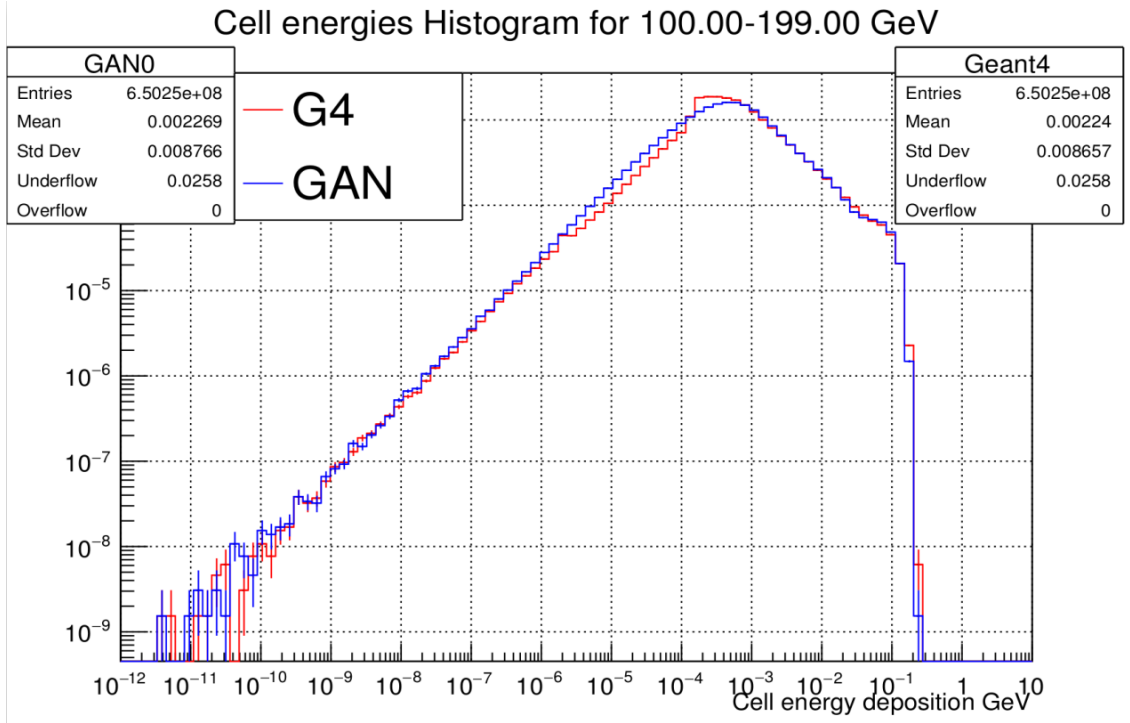
Geant4 GAN

CERN openlab

Single cell energy



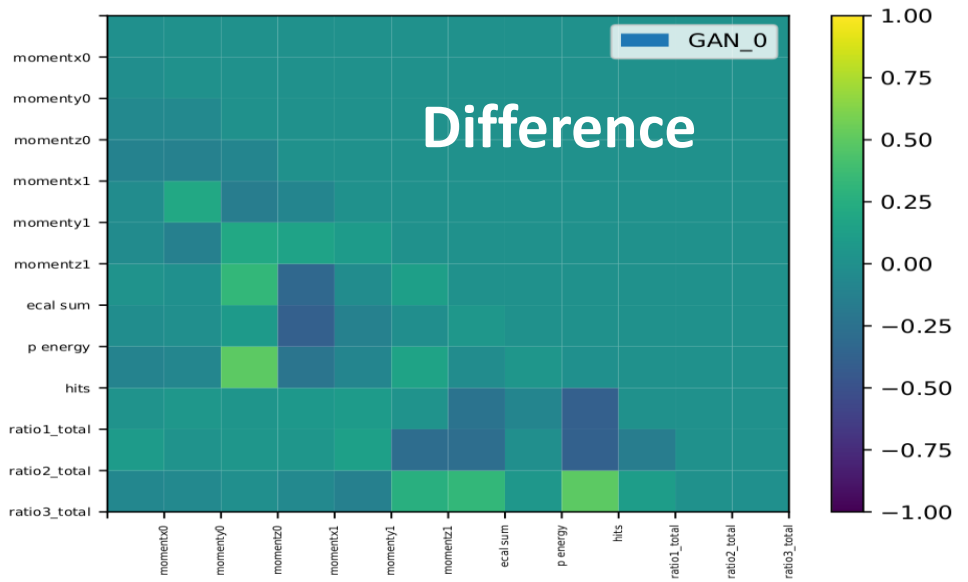


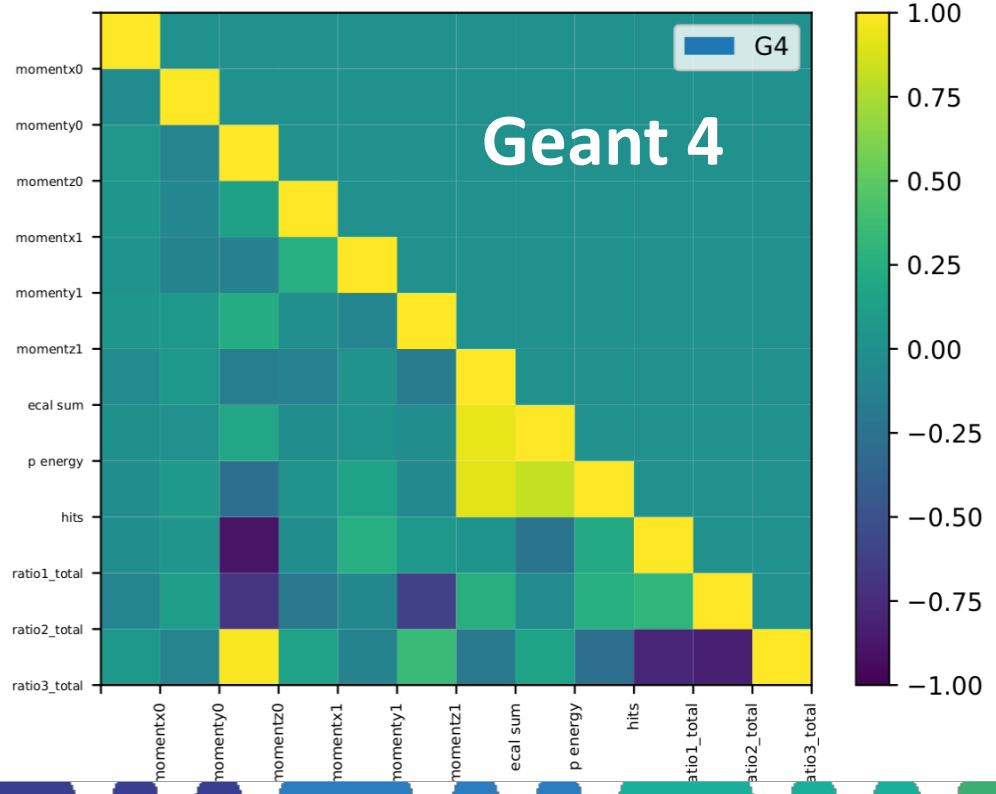


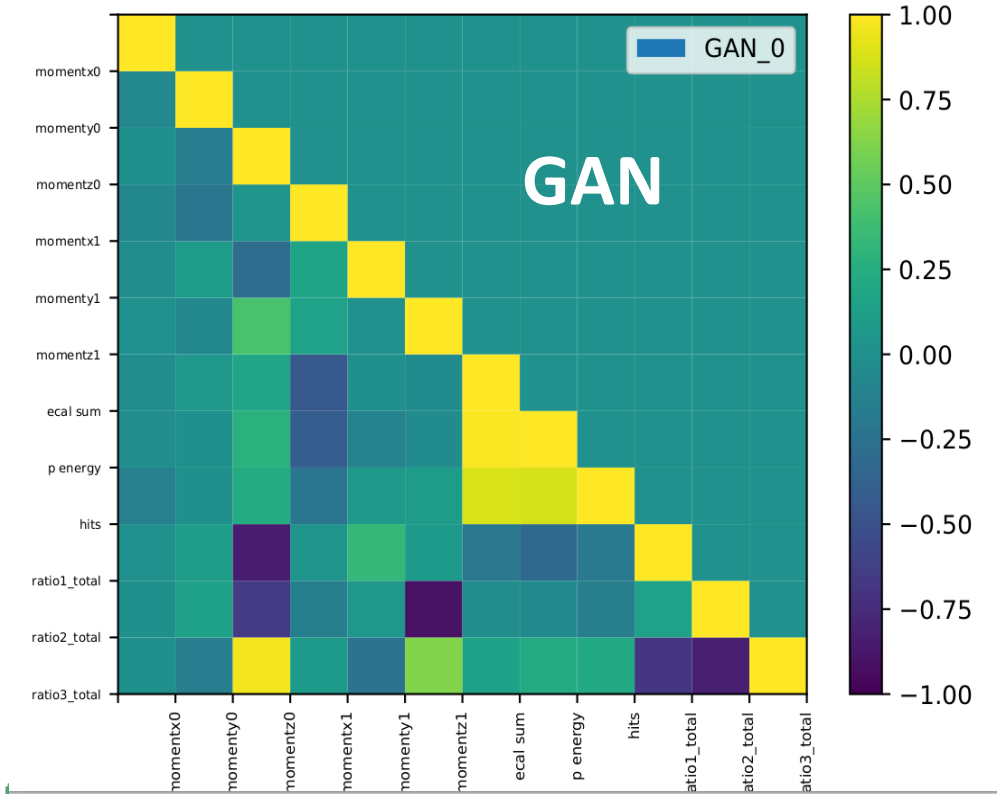
Agreement to Geant4 down to 500 keV

Low energy features missing

Internal Correlations

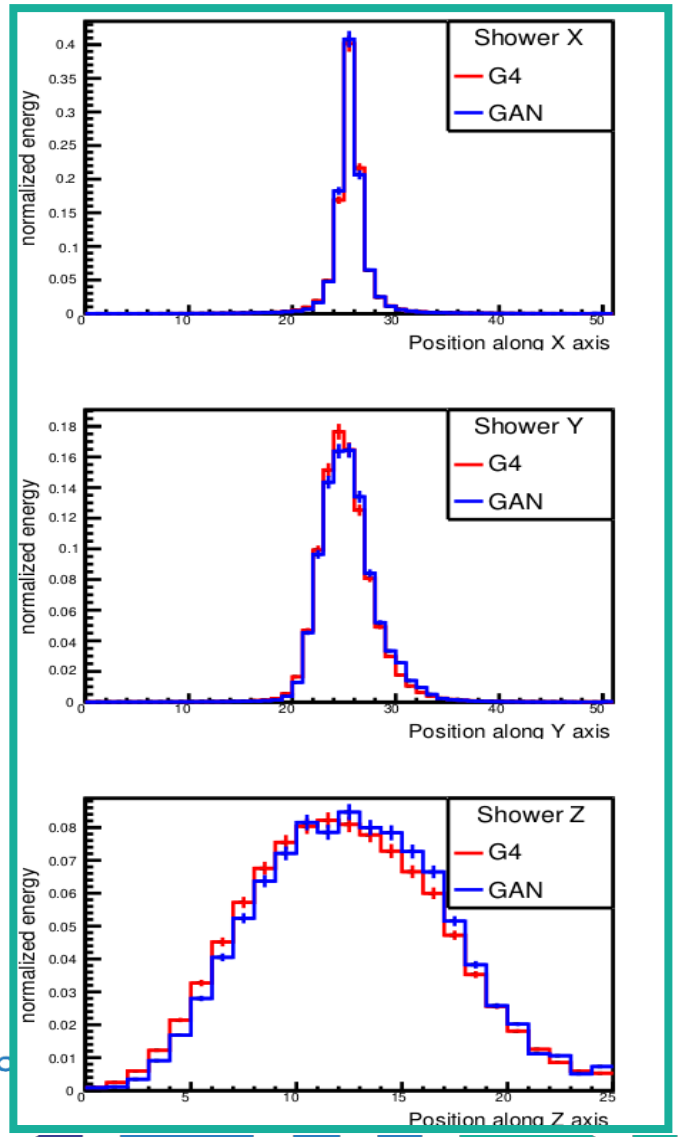




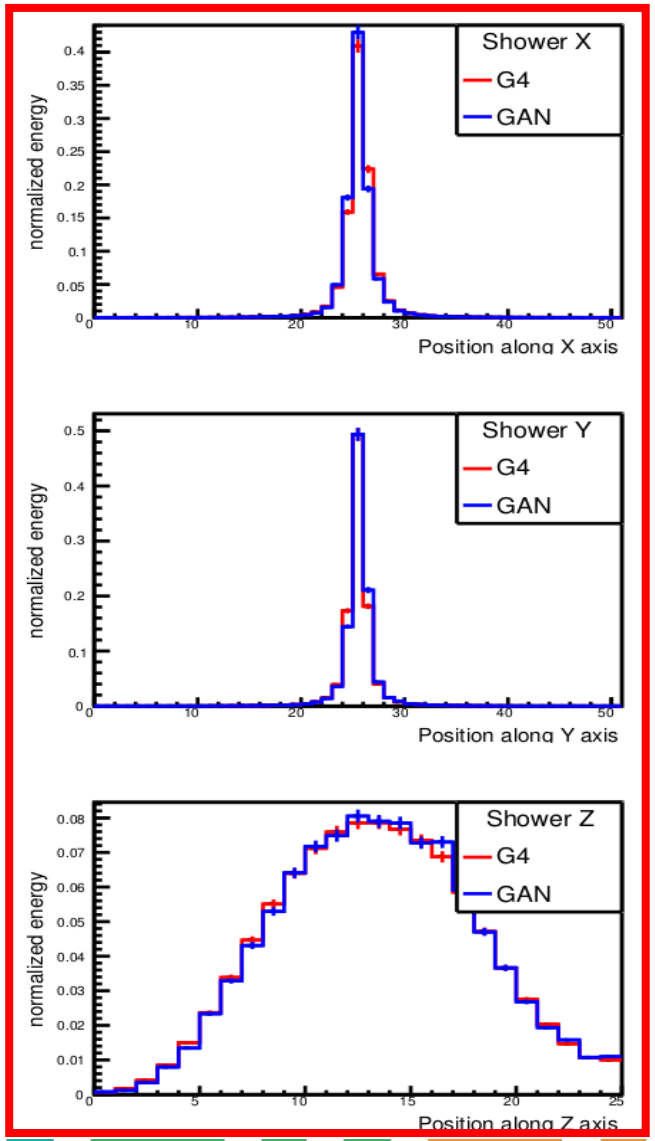


Energy shower shapes

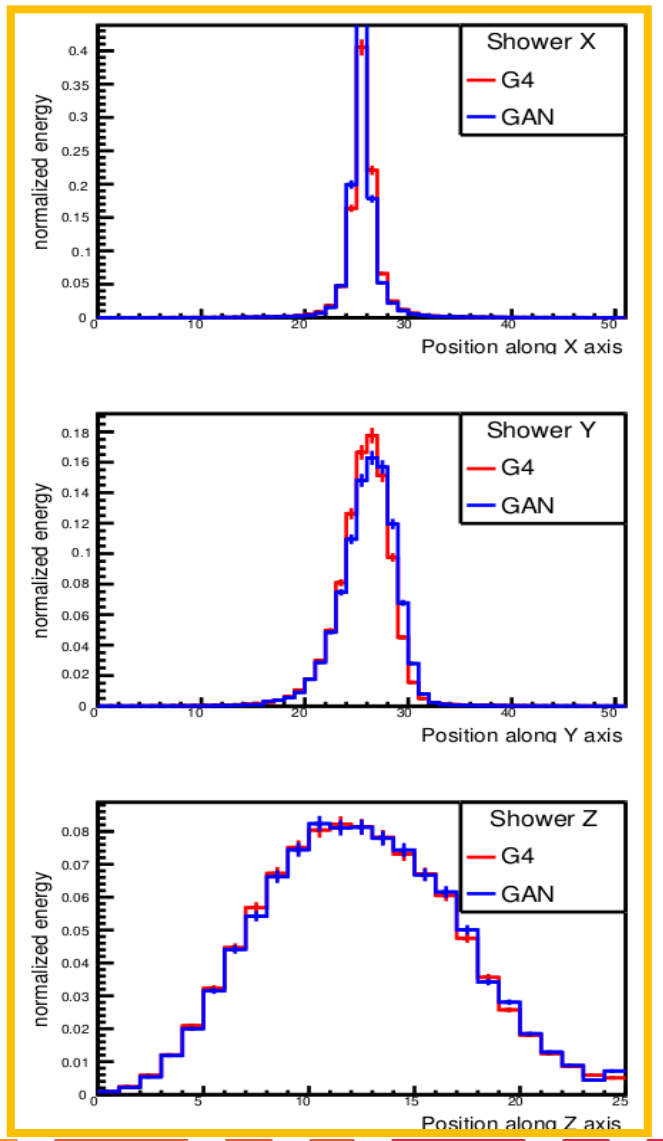
α= 60°



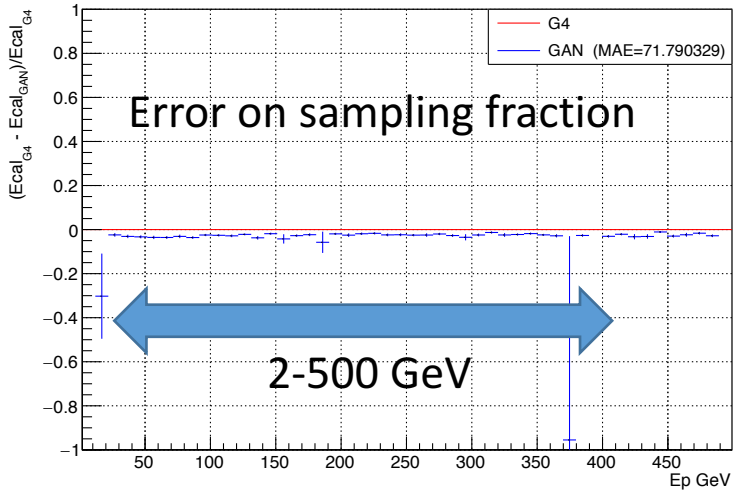
α=90°

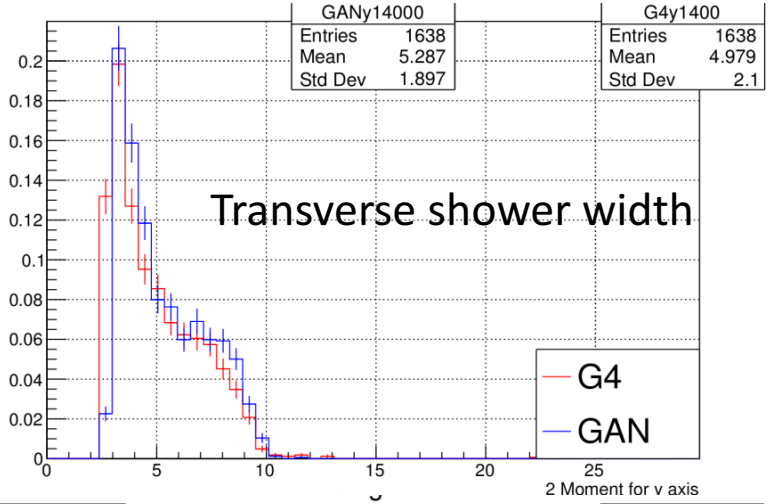


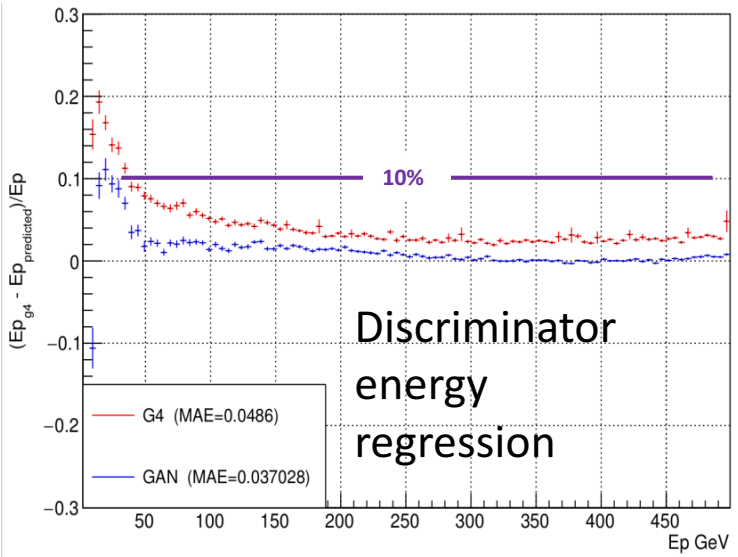
α= 120°

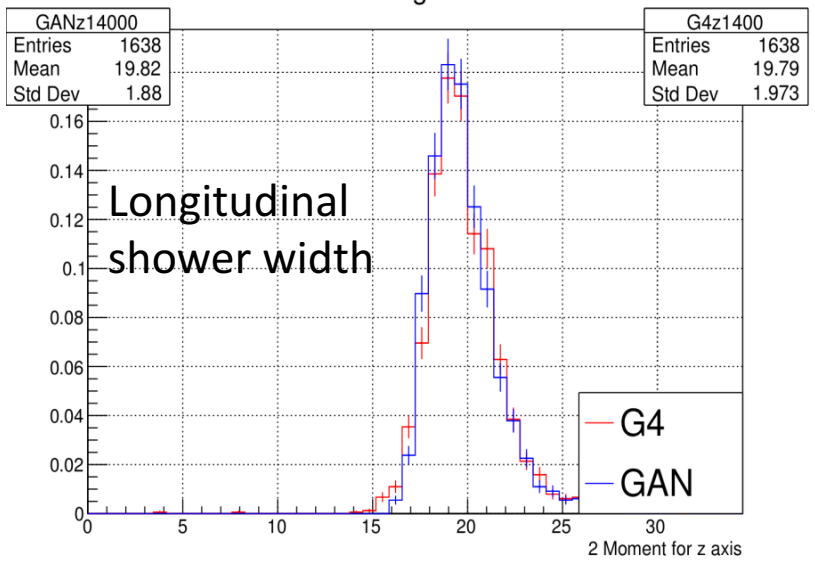


Transfer learning: extending the energy range



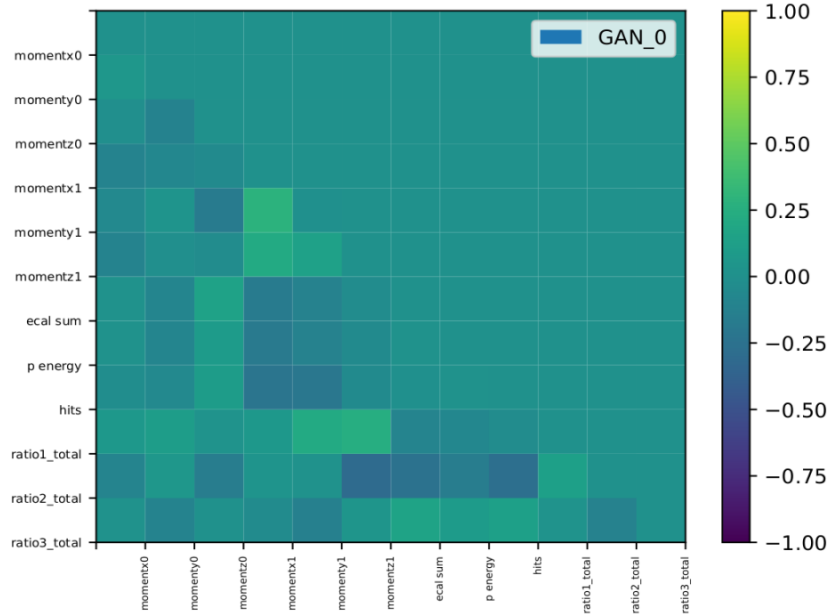






- Transfer learning from 100-200 GeV pre-trained network
- Double training dataset statistics, 3 epochs training

Improved correlation description!



Further Validation (I)

Structural Similarity Index (**SSIM**) is used to assess image similarity

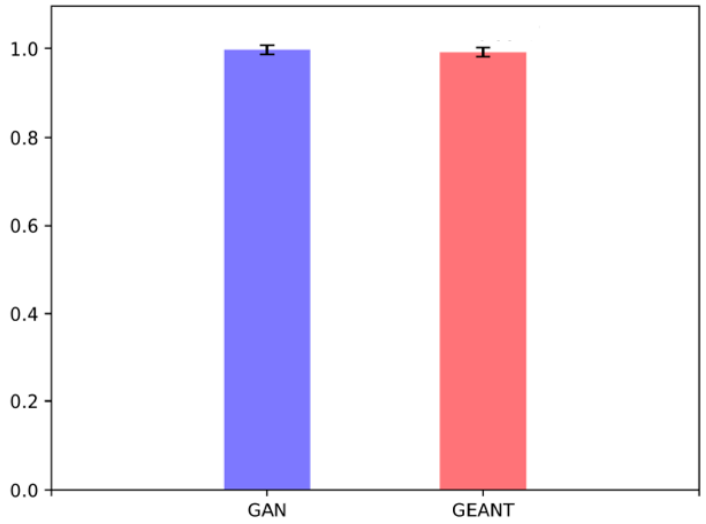
Measure **diversity** in GAN generated images

Run Triforce on GAN/GEANT4 events

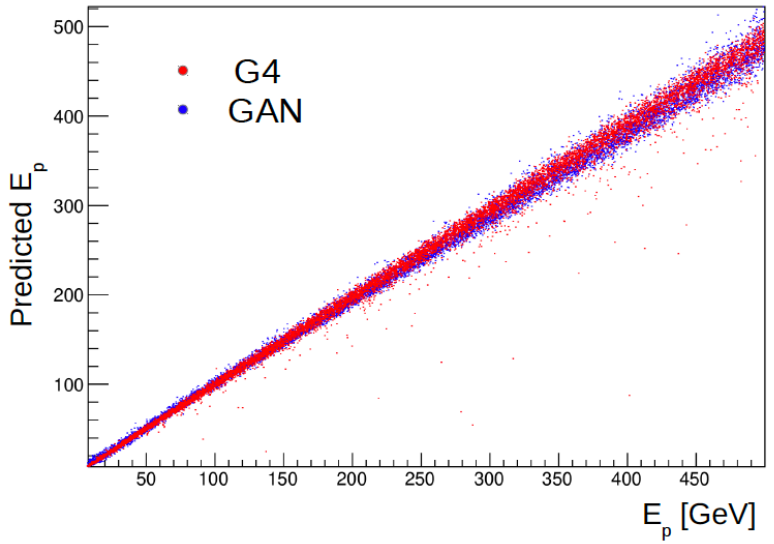
Trained on Geant4 data

SSIM (scale)	SSIM G4 vs G4	SSIM GAN vs GAN
1	0.94	0.95
1e-2	0.21	**0.25**
1e-4	0.045	**0.061**
1e-6	0.045	**0.051**

electron identification



energy regression



Triforce, Matt Zhang,
https://github.com/BucketOfFish/Triforce_CaloML

[1]S. Ravuri, O. Vynials, NeurIPS 2019

Further validation (II)

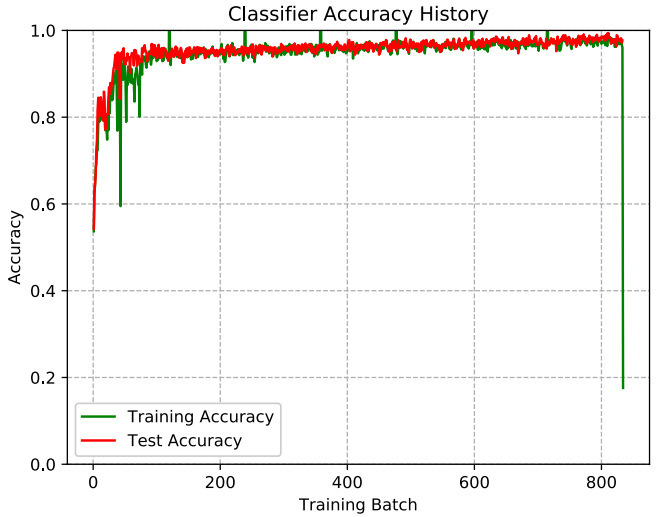
Train TriForce classifier on GAN data

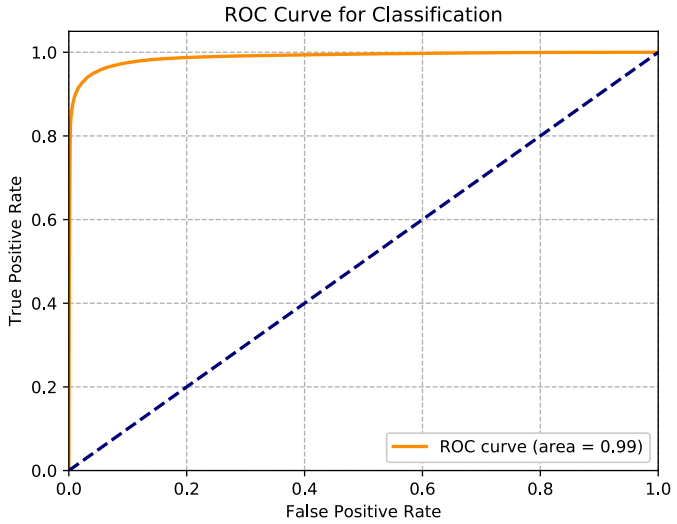
Similar to **Classification Accuracy Score** idea[1]

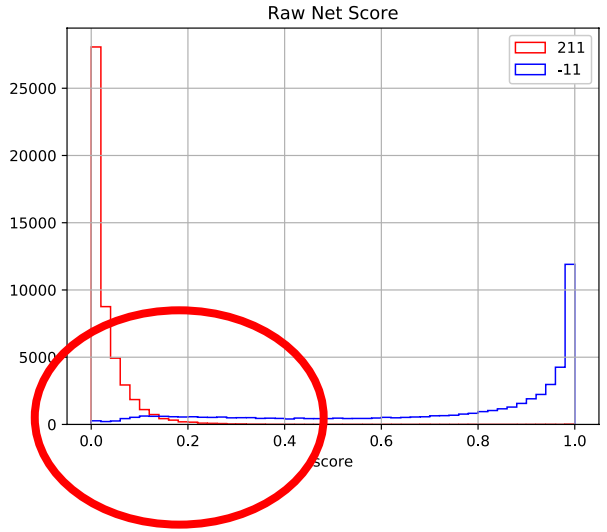
Train Triforce with a mix of GAN (electrons) and **Geant4** (pions) data

Validate performance on Geant4:

Obtain very good results (99% AUC)







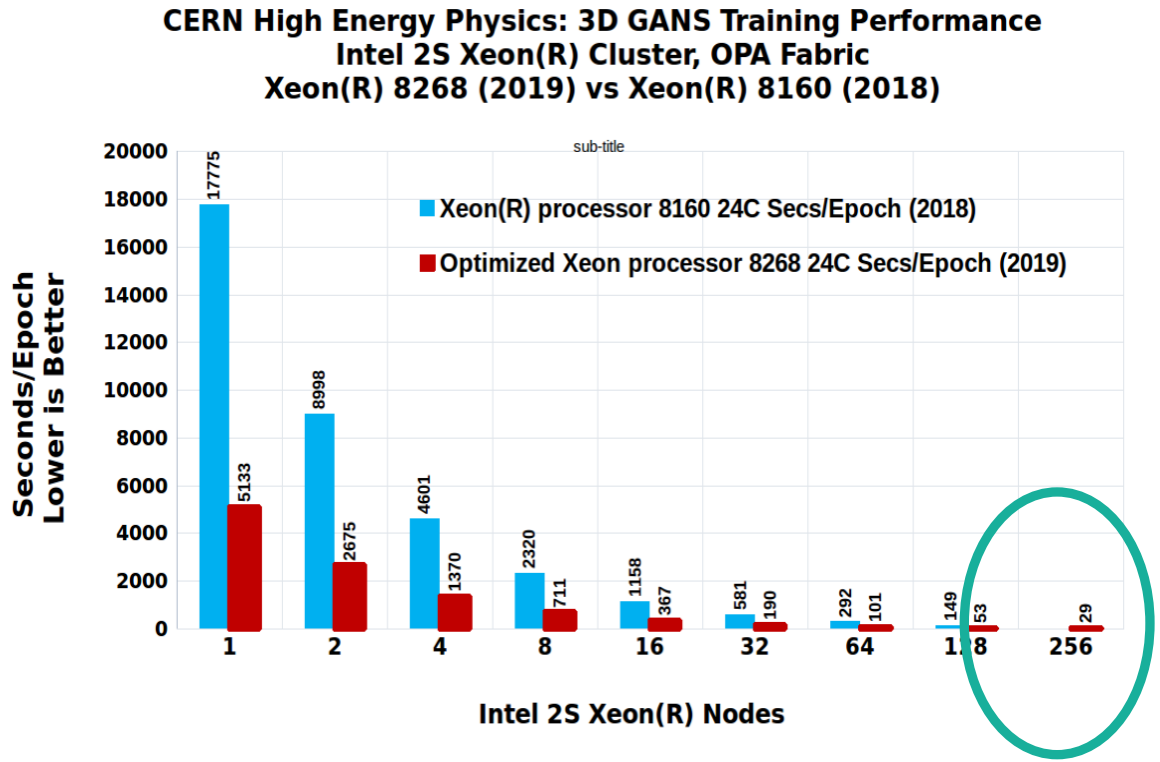
Some electrons are not classified correctly

Computing performance

20000x faster than Geant4

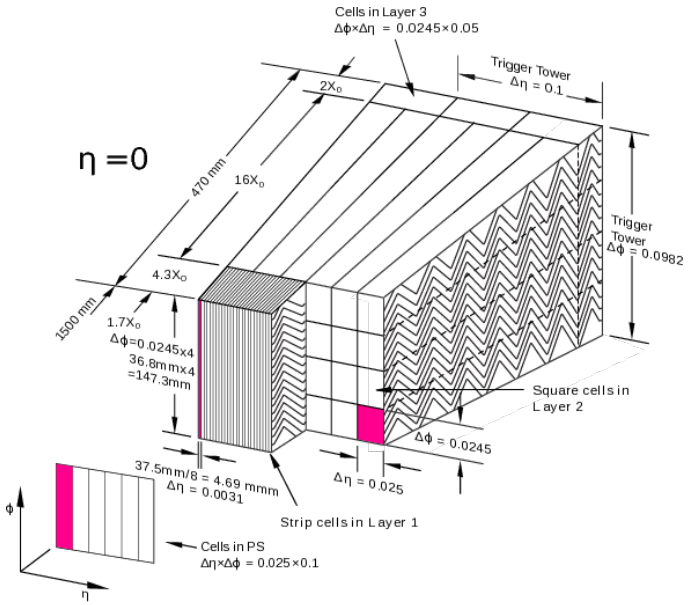
Time to create an electron shower		
Method	Machine	Time/Shower (msec)
Full Simulation (geant4)	Intel Xeon Platinum 8180	17000
3d GAN (batch size 128)	**Intel Xeon Platinum 8160 (TF 1.12)**	**0.8**

Convergence in 15min



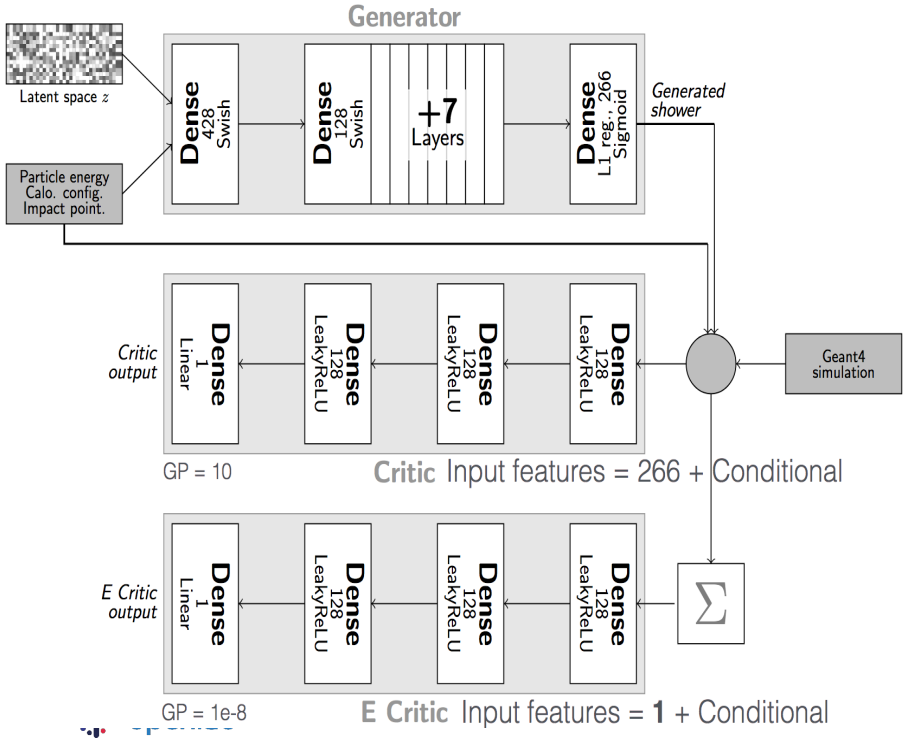
Other simulation examples at the LHC

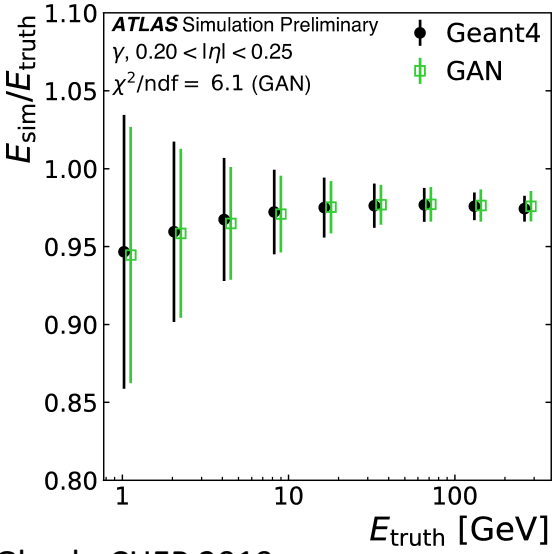
ATLAS LAr calorimeter



Complex geometrical structure

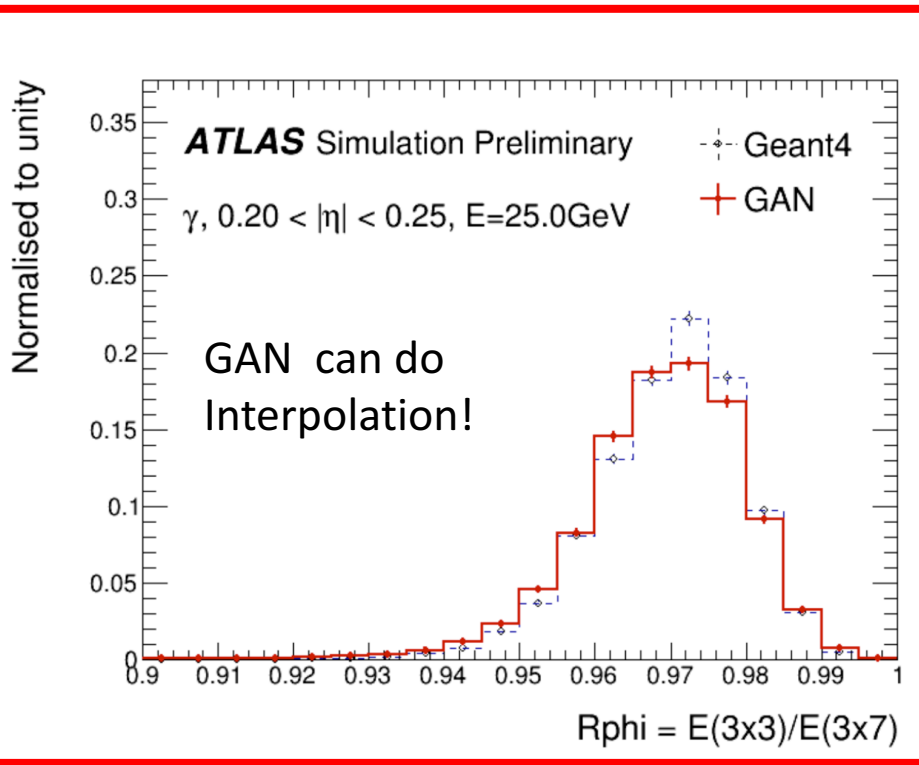
Wasserstein GAN: One generator against two critics





Ghosh, CHEP 2019

ATL-SOFT-PUB-2018-001



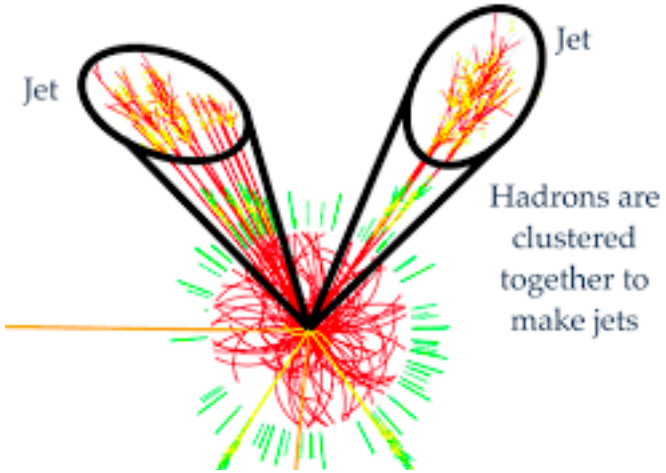
DijetGAN

QCD dijet events at the LHC

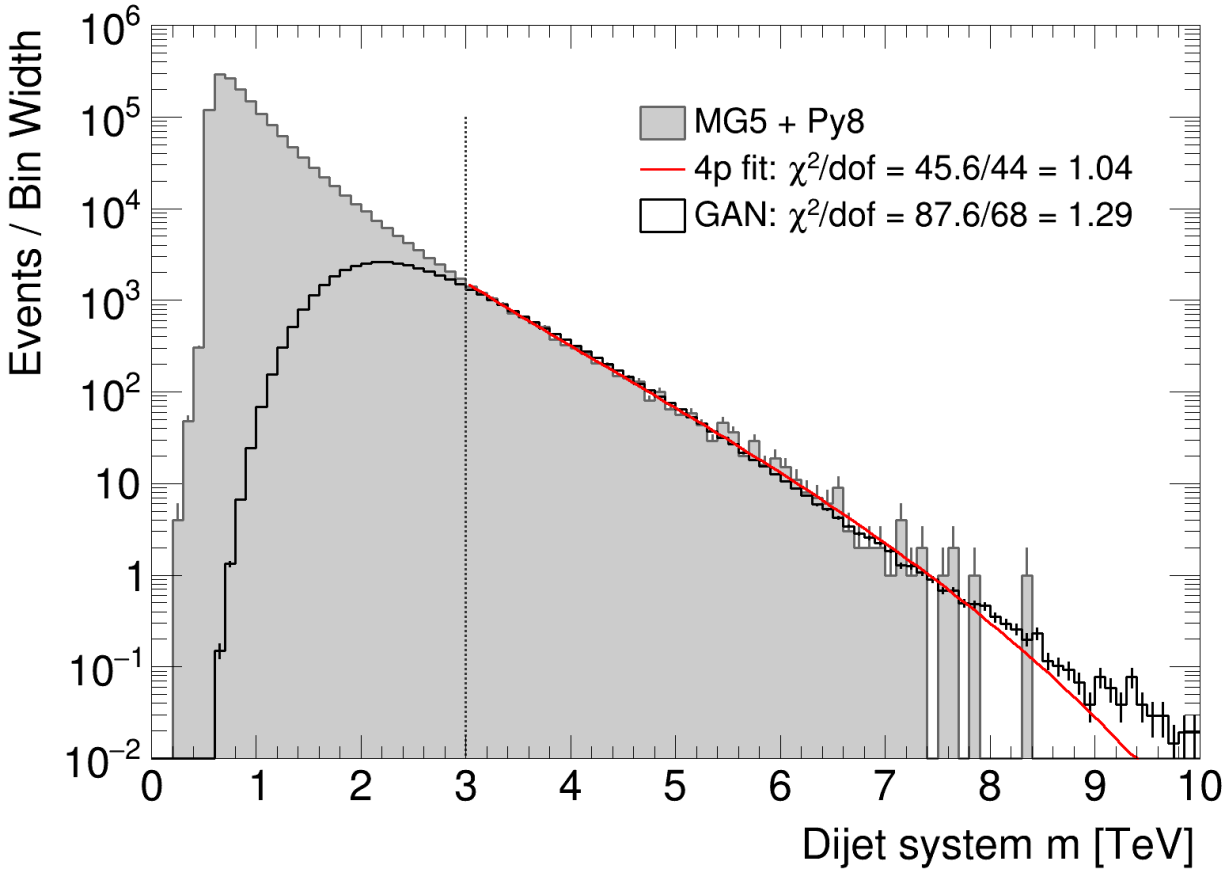
Investigate regions of the phase spaces with **low**

cross-section → interesting kinematic region for BSM searches

GAN can produce "**reasonable**" output beyond the training data phase space!



S. Palazzo, IML WG 2019

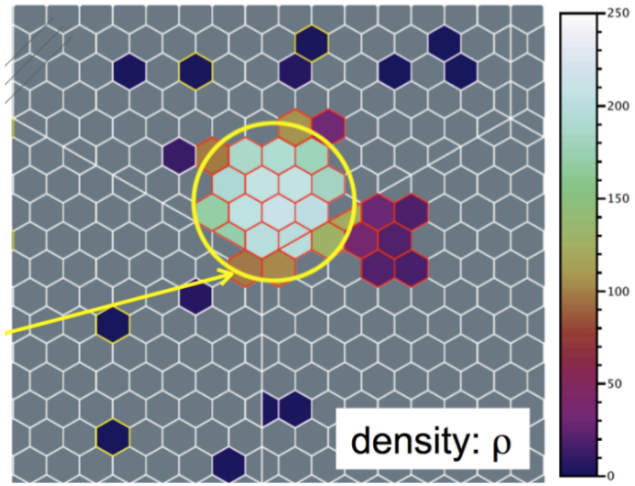


CERN openlab

arxiv.org: 1807.01954

CMS HGCAL prototype

High Granularity and Hexagonal cells *for CMS upgrade*

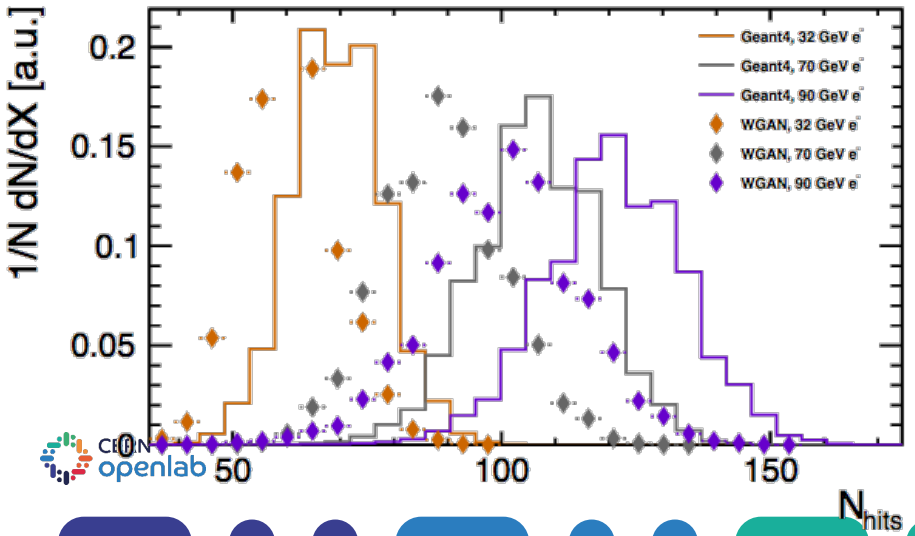


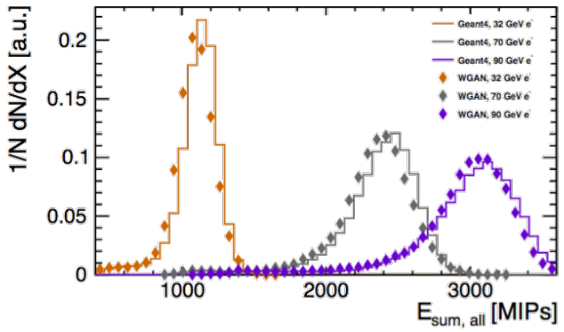
Wasserstein conditional GAN (**convolutions**)

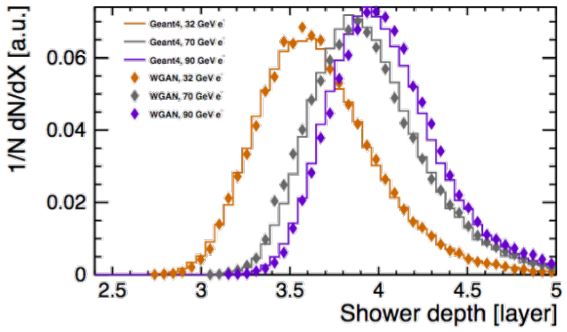
Train a generator against a **critic and 2 constrainer networks** reconstructing energy and impact point coordinates

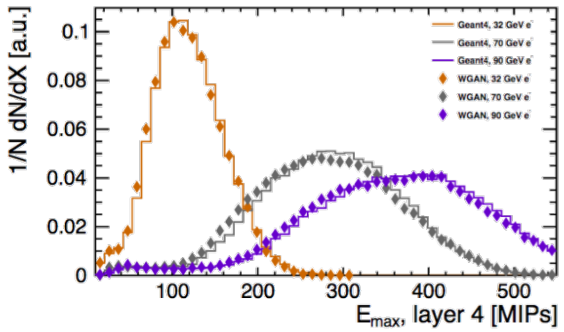
Good agreement to Geant4

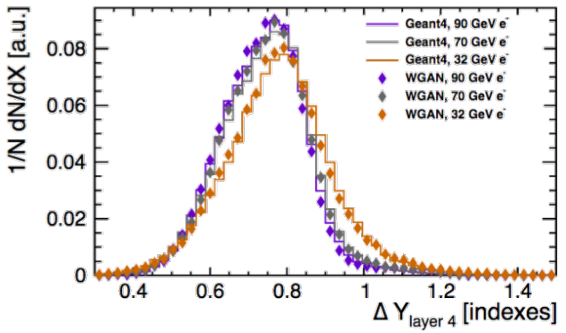
Some problems at low energy











Other applications in HEP (simulation)

Generative models for ALICE TPC simulation (ACAT2019)

GAN in LHCb: calorimeter and RICH simulation (CHEP2019)

Graph-GAN for CMS HGCAL (ACAT2019)

Variational AutoEncoders to simulate ATLAS LAr calorimeter (CHEP19)

Wasserstein GANs to generate high-level physics variables based on Monte Carlo ttH (superfast-simulation) (IML WG 04/18)

Particle-GAN for Full Event Simulation at the LHC (ACAT2019)

Refining Detector Simulation using Adversarial Networks (CHEP2019)

Model-Assisted GANs for the optimisation of simulation parameters (AISIS2019)

Conclusions

Generative Models are powerful techniques

- Object of an intense R&D , require careful **validation**
- **Many applications** in High Energy Physics

Simulation is one of the main applications

- High level of **accuracy** & **speed**
- **Multiple features/capabilities**

Deep Learning is a driving force toward changing our computing model

- DL workloads are HPC friendly
- Benefit from dedicated hardware and accelerators

Addressing Computing challenges

Millions of operations

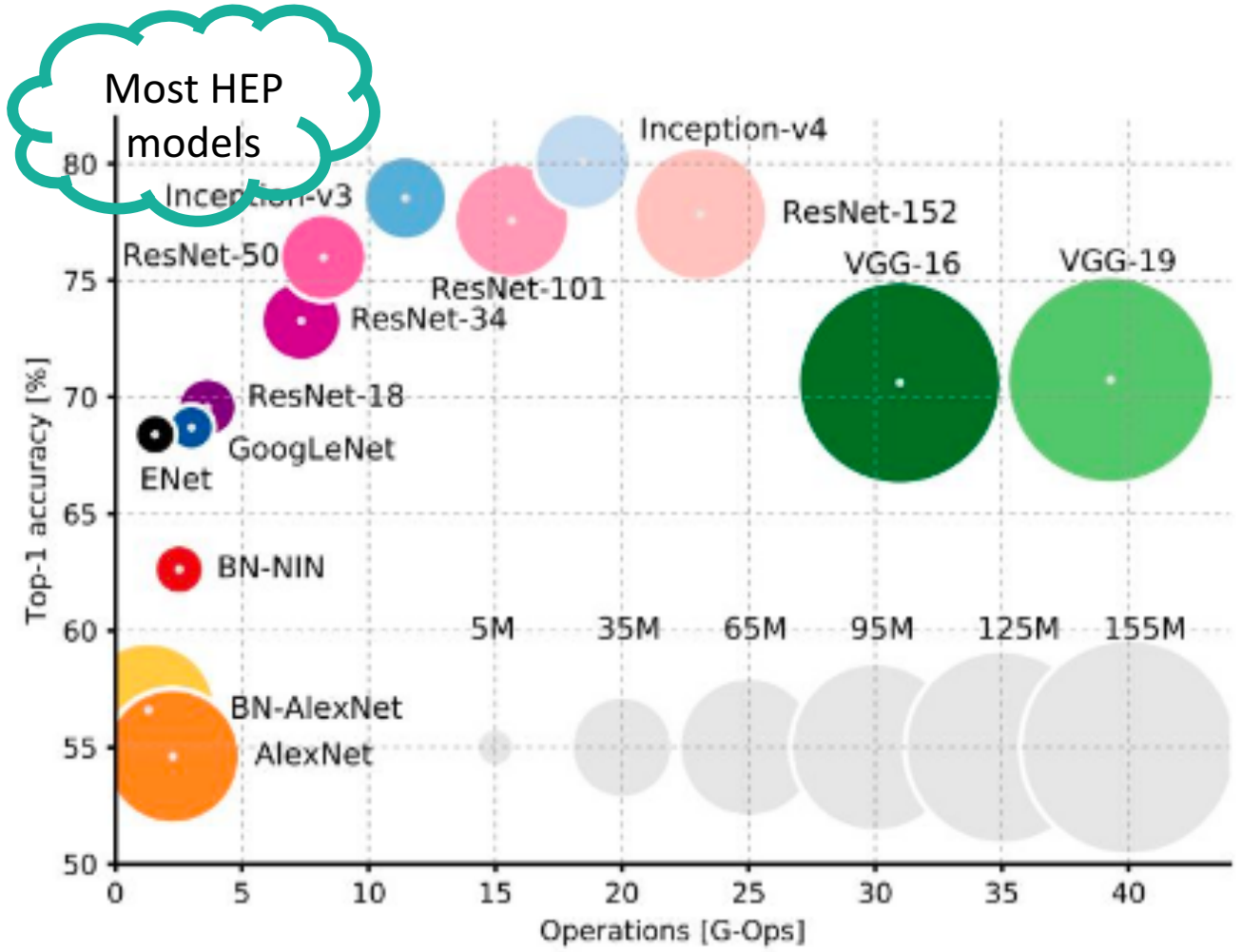
- Mostly matrix-multiplications

HEP models are designed and **optimised for specific tasks**

- Generally **custom** models
- **~Fewer weights** and operations than out-of-the-box tools
- **Higher accuracy**

Depending on the task, we might need:

- **Fast inference**
- **Online training** capability
- **Fast training** for large optimisations



CERN openlab

Accelerators

Deep Learning workloads are naturally **accelerator friendly**

- Large number of frameworks and ecosystems to simplify deployments
- Can work with half precision arithmetic (16FP, …)

GPUs are de-facto **standard** to run DL

- R&D on reducing bottlenecks (memory size, I/O. …)

FPGAs can provide **low latency** inference

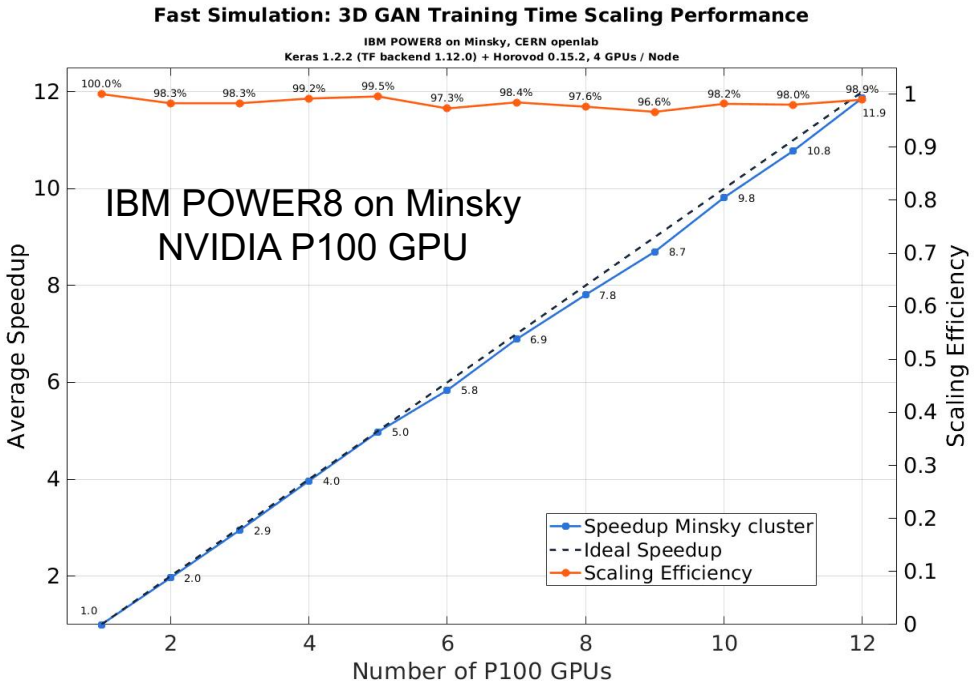
- Network compression/quantization/parallelisation
- Different programming approaches Hardware Description Language vs High Level Synthesis

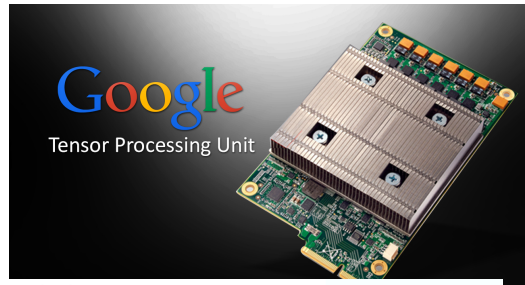
Frameworks exist that “compile” ML code for different hardware

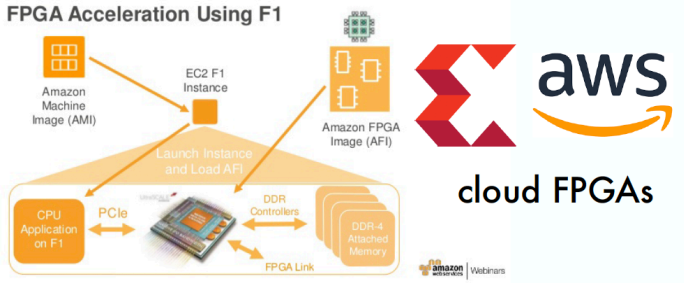
- Customisation

Available in cloud environments for **on-demand access**

- Initial tests to time inference on cloud vs local







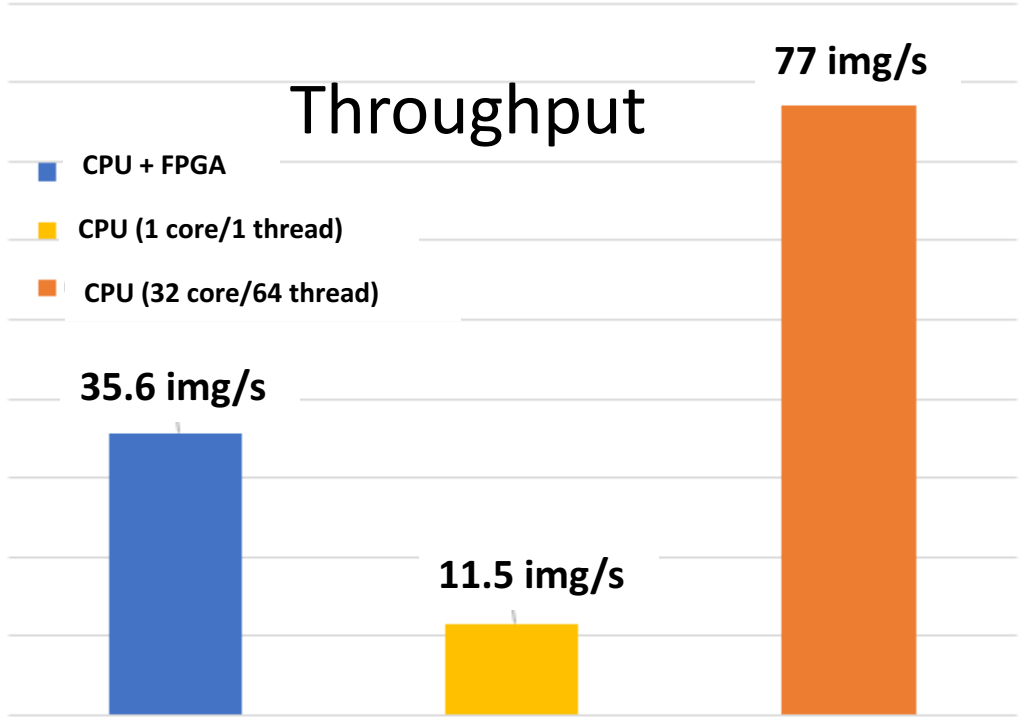
Heterogeneous computing

*FPGA-acceleration of **3DGAN** inference*

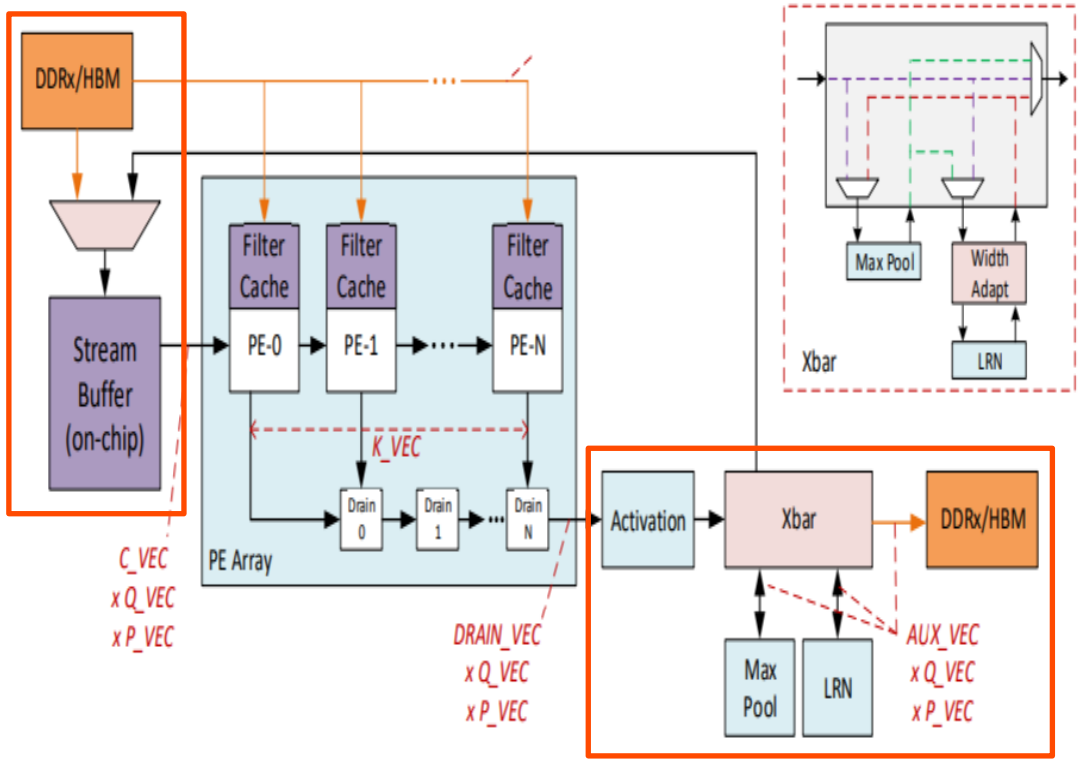
Intel PAC Arria 10 card; **OpenVINO**; **DLA** design suite

Implement missing components:

- 3D convolutions (done)
- 3D upsampling (inprogress)



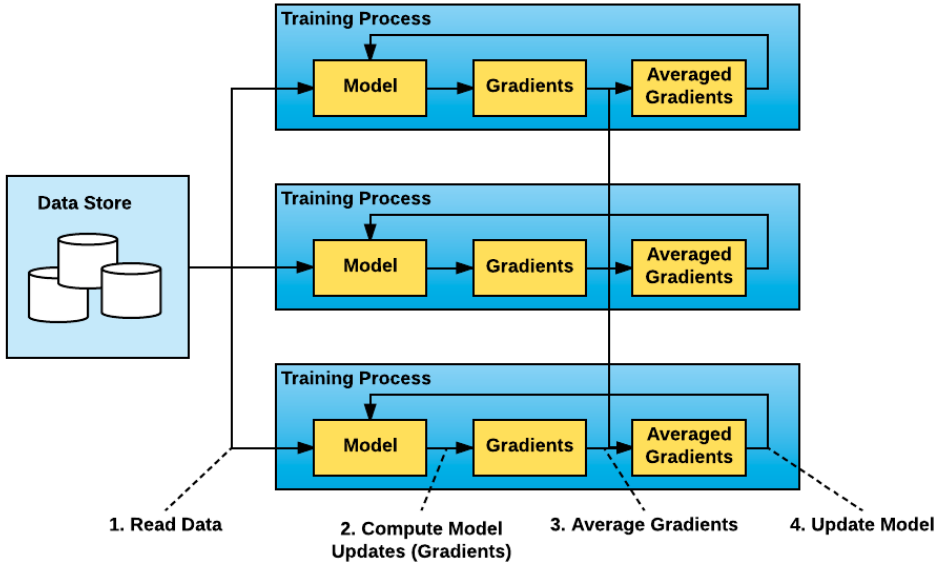
DLA customization



presented at CHEP2019

Distributing the training process

- **Data** distribution
  - Compute gradients on several batches **independently**
  - Update the model **synchronously or asynchronously**
  - Applicable to **large dataset**

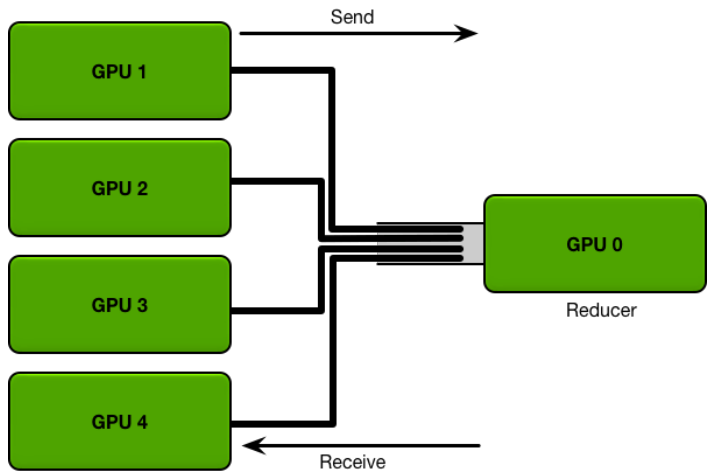


Data is sent and received from a single node

Can have high communication costs

Smart update strategies

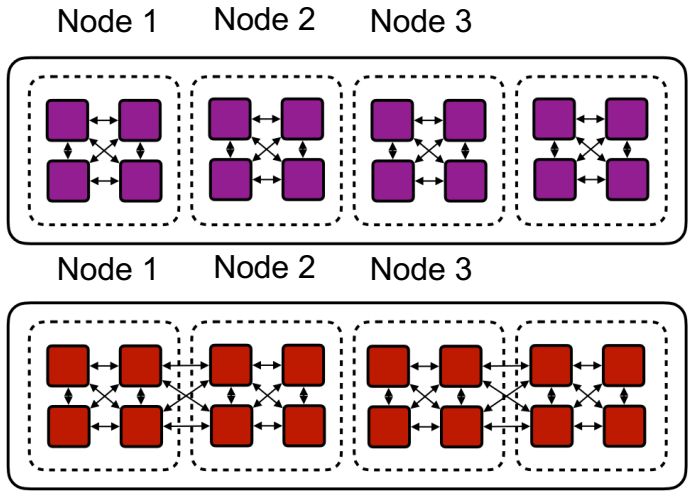
Computation or bandwidth bottleneck



HPC resources

HPC "creates" new DL models

HTC



Efficient Intra-node communication is key

Most powerful systems are **hybrid**
- Ease access to the resources
- Integration in HEP infrastructure

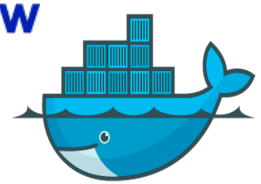
Software stack **deployment** ?
- Containers
- Most DL framework provide containerised versions

Workflow management ?
- Native **DL platforms** are natural choice (i.e. Kubeflow)
- Adapt HTC job schedulers ?

Data access/management ?
- S3 works very well
- Supported widely in **commercial clouds**
- Not all HPC centers allow access to it

Kubeflow

kubernetes



docker

presented at IXPUG

Distributed training on Cascade Lake

Training 3DGAN can take days

Implements data parallel approach using Horovod.

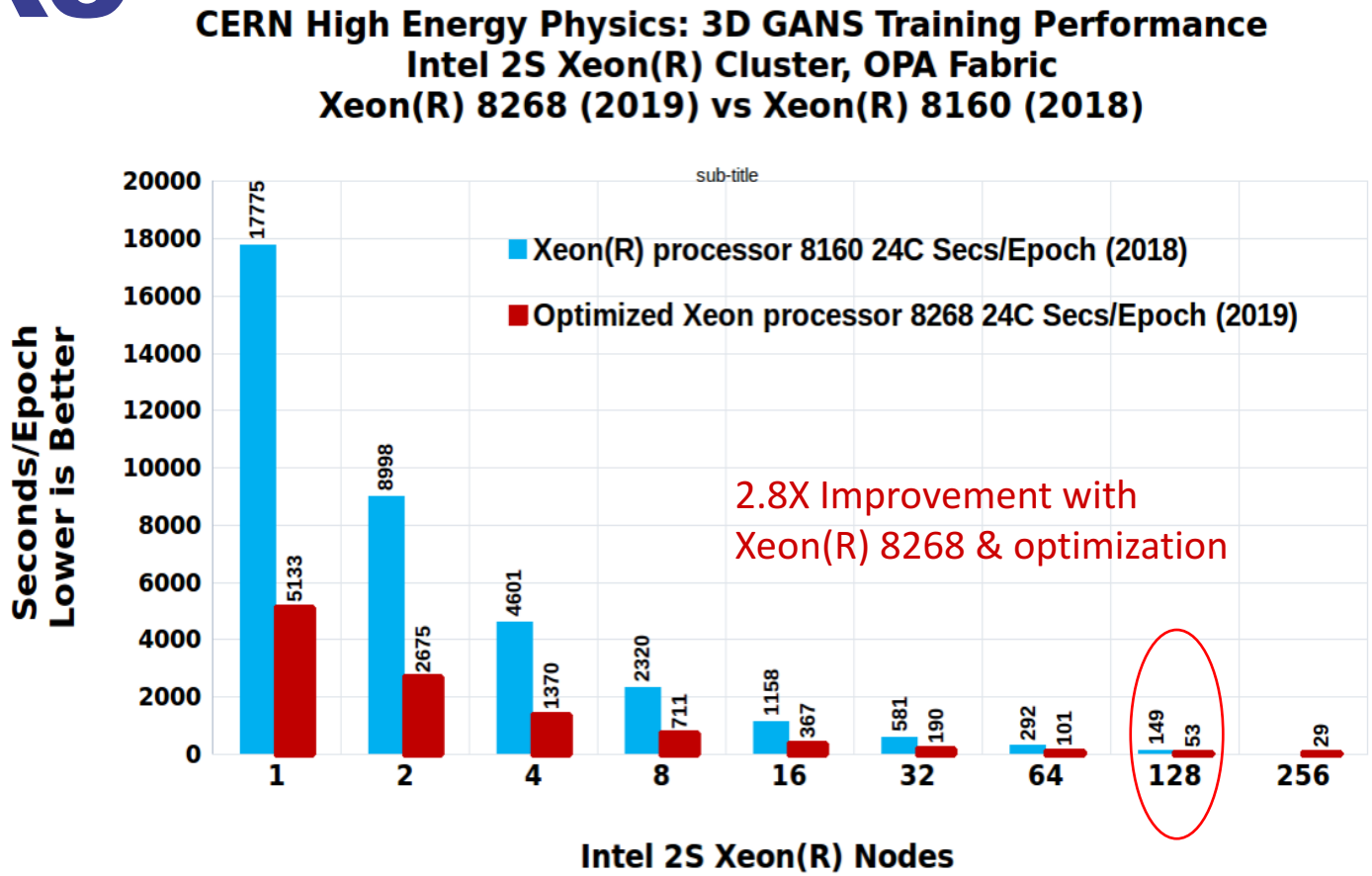
128 Nodes:

Xeon 8160 (Sky Lake): ~2 Mins/Epoch

Xeon 8268 (Cascade Lake): < **1 Min**/Epoch

256 Nodes:

Time to Convergence**: ~15 Mins**



Intel Endeavour cluster: ***Xeon® 8268 Cascade Lake,*** *2 Sockets /node, 24 cores per socket, Intel® Omni-Path Architecture*

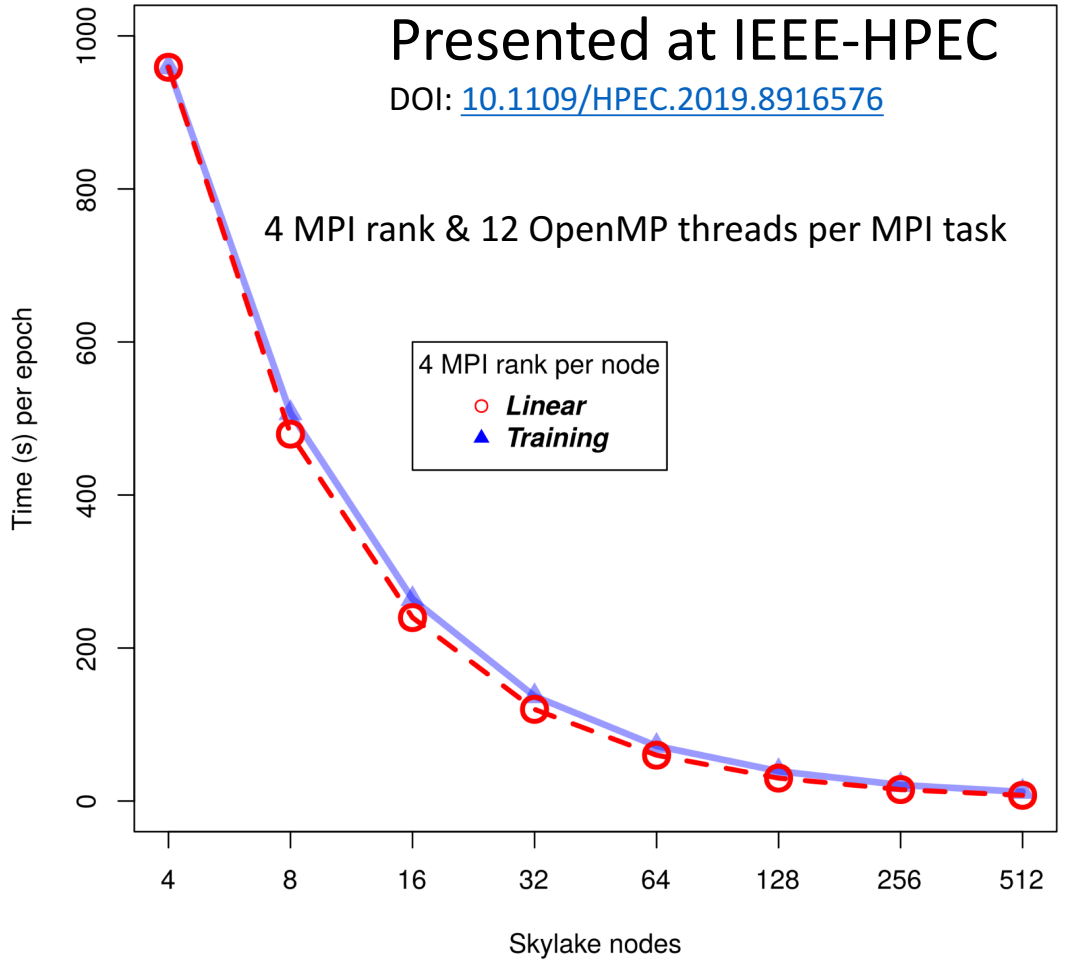
Software: ***Tensorflow 1.14 (Intel optimized),*** *MKL-DNN 0.18, Horovod 0.16.4, Keras 2.2.4*

Containerized deployment on HPC

Transition AI algorithms from laptops to supercomputers

Deploy 3DGAN training on SuperMuc-NG using CharlieCloud

- Intel Xeon Platinum 8174 (Skylake)
  - 48 cores
  - Intel® Omni-Path Architecture



https://doku.lrz.de/display/PUBLIC/SuperMUC-NG

Charliecloud

50

Cloud resources

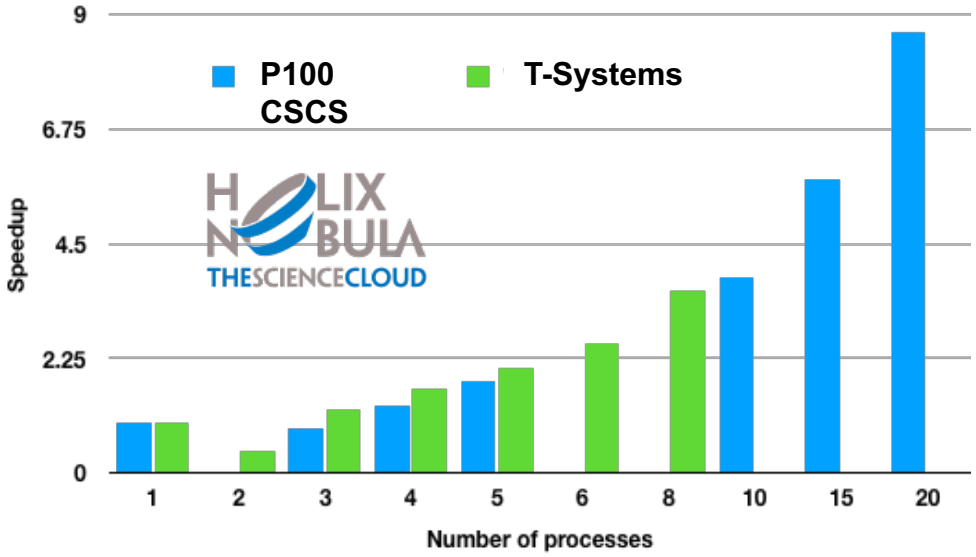
A solution to consider for offline applications

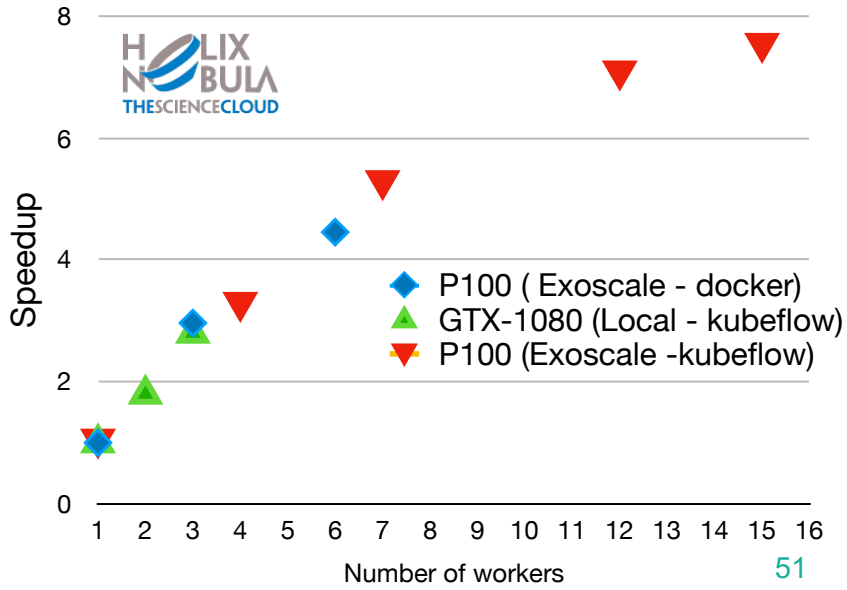
- Not always feasible/effective to **buy specialized hardware**
- **Most MLaaS solutions are not customizable** enough for scientific use cases
- **Great opportunities for R&D with industry**

New initiatives to increase access to **commercial clouds**

deploy **hybrid models** (OCRE in the context of the EOSC)

3DGAN training on public cloud:
Docker + Kubernetes/Kubeflow





Bonus Material

Refining Simulation using GANs : Ultra High Energy Cosmic Rays

Pushing the boundaries: Quantum GAN

Refining Simulation using GANs

Pierre Auger Observatory

Detection of UHECR E>$10^{17.5}$ eV

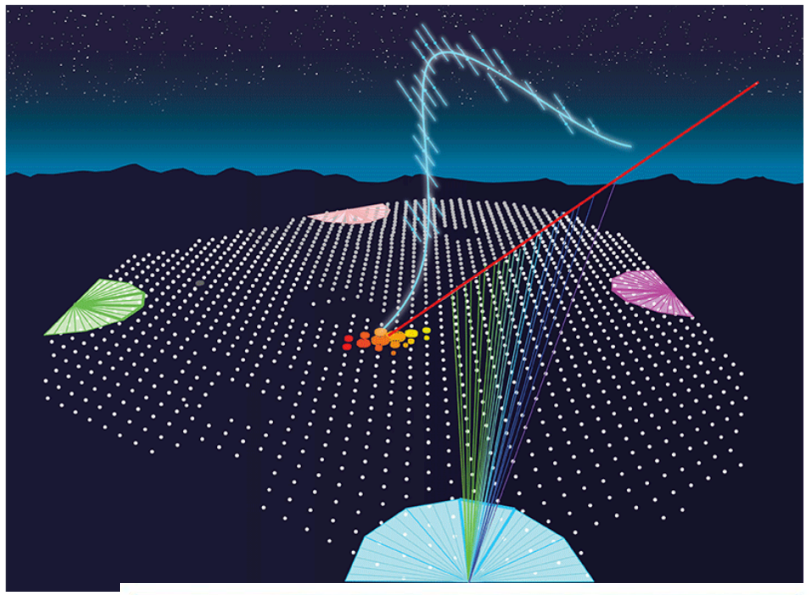
Hybrid Technique

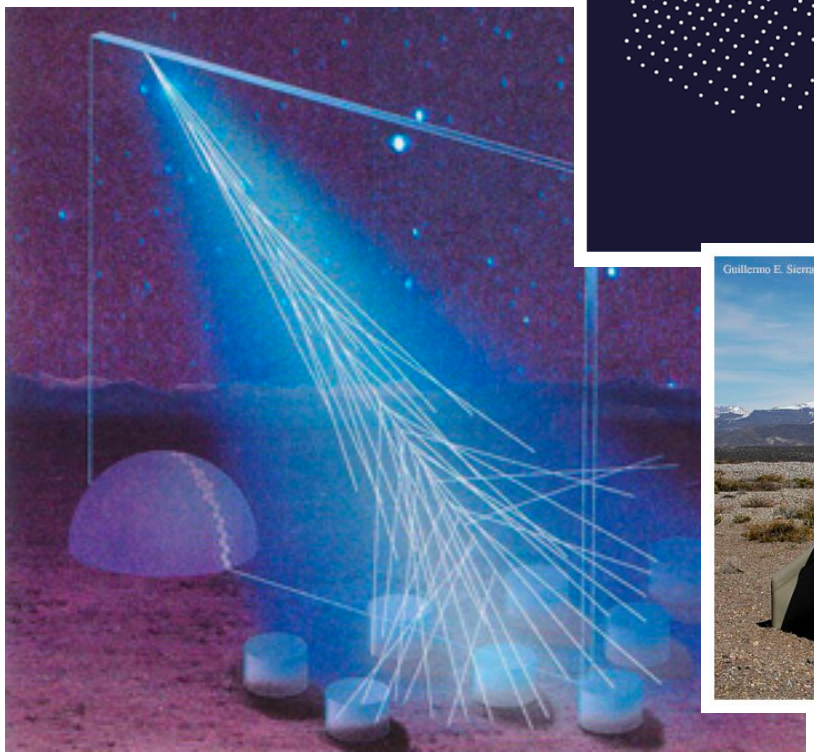
- 27 Nitrogen Fluorescence (ultra-violet) telescopes
- 1660 Surface detectors (water tanks)

3000 km^2 array size

Simulation/Data mistmatch

- **Refine simulation using WGAN**
- Train DNN on refined simulation





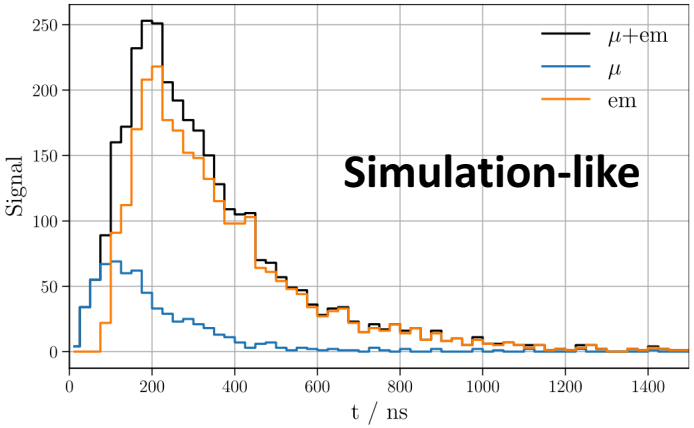
www.auger.org

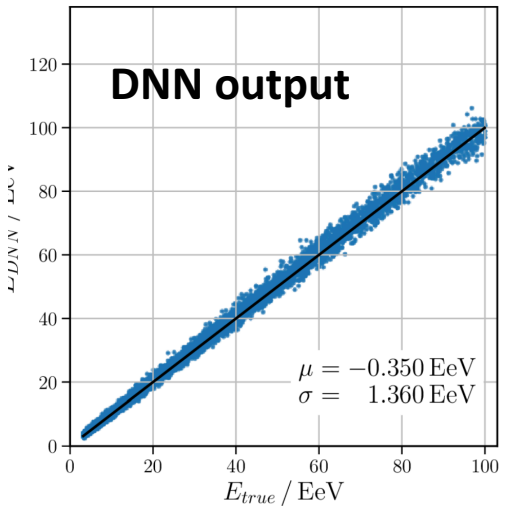
arXiv:1802.03325

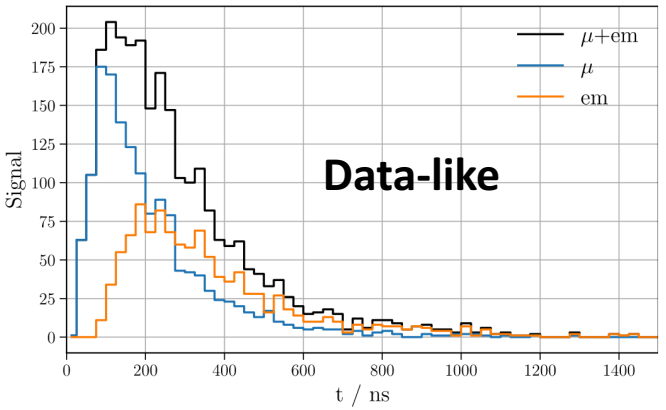
Refining Simulation using GANs

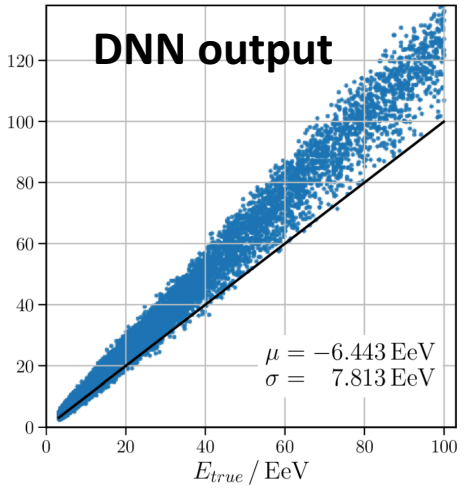
DNNs very sensitive to simulation / data mismatches
Simulation underestimates the muon component
Generate 2 datasets “data-like” and “simulation-like
Run DNN regression on them

→ DNN does not fit the “data-like” events because it is trained on simulation

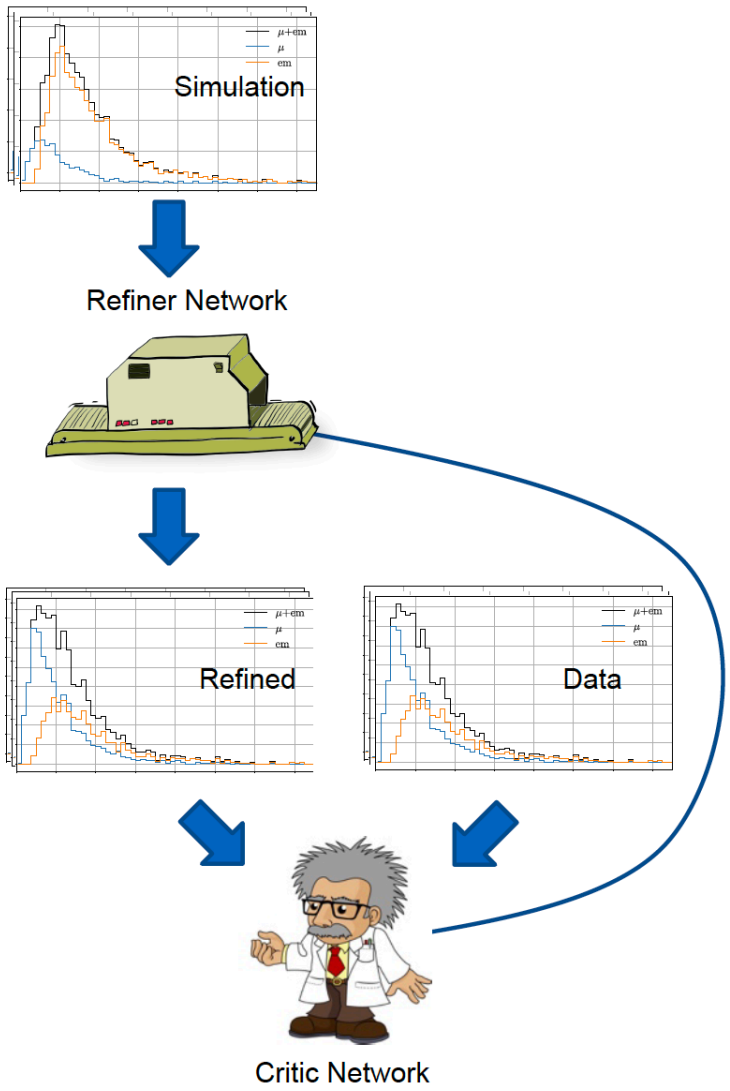


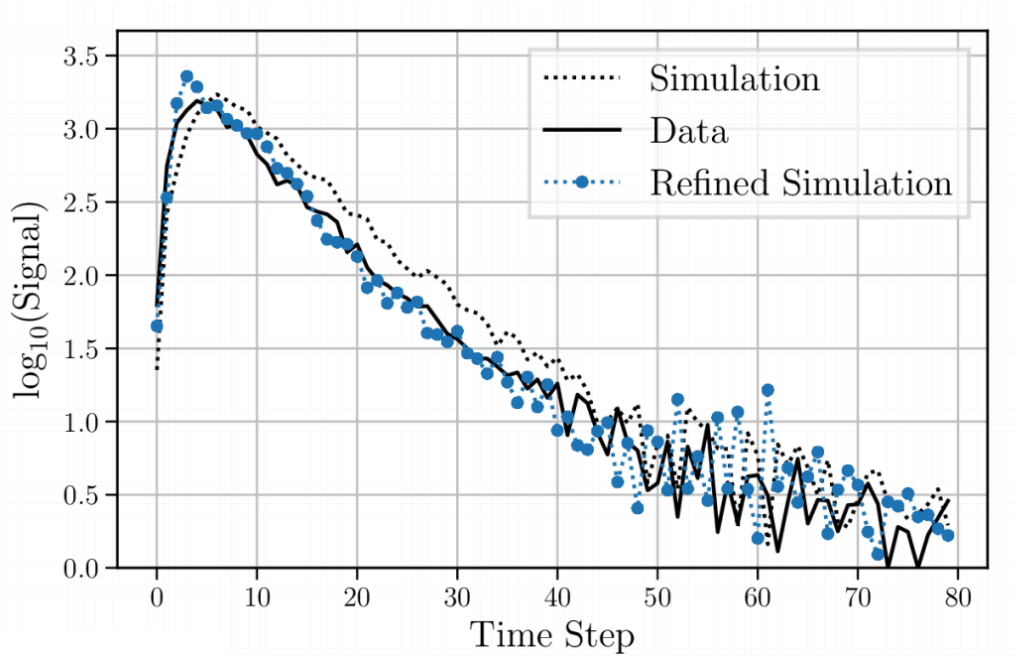


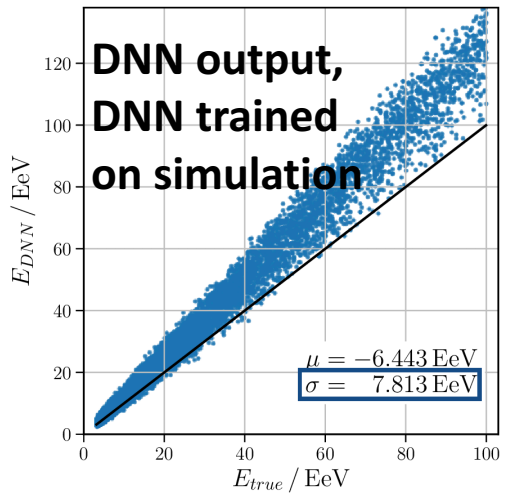


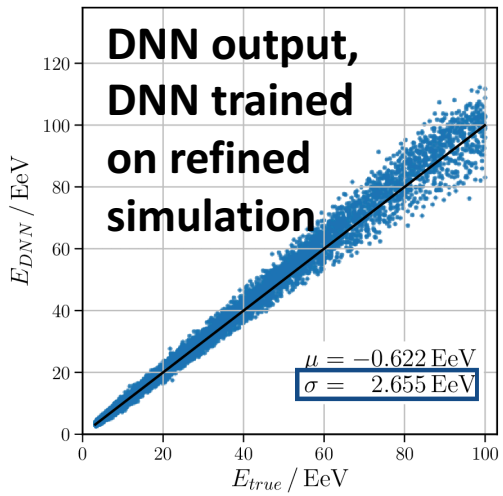


Refining Simulation using GANs









Why Quantum Machine Learning?

Quantum approach to ML could solve more complicated problems… faster

- ML based tool can recognize complicated (hidden) patterns in data
- Quantum processors can produce statistical patterns that are computationally difficult to produce with classical approaches

→ Could quantum processors recognize more complicated patterns in data?

Defining what quantum speed-up means is a complicated task

Need to compare to the “best available” classical algorithm

Quantum ML

... and ML for Quantum Computing

- QML introduces quantum algorithms as part of a larger implementation
  - Fully quantum or hybrid classical/quantum approaches
  - Input data could be quantistic → ML for QC
- How do we construct Quantum Neural Networks (QNN) ?
  - Direct association between neurons and qubits
  - Encode information into amplitudes of a quantum state
- How do we represent learning rules?
  - Need association rule between NN activation patterns and pure quantum states
- How do we address data loading?
  - Quantum state preparation
  - Direct access through qRAM ?
    - Possible to train on large datasets by only loading a small number of samples!

Principle of Quantum Computing

Circuit model

$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix},\ |1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$$

Based on quantum mechanics → Encoding in Quantum Bits (= Qubit)

Superposition of states ; 2^n possible states $|0\rangle, |1\rangle, \ldots, |2^n-1\rangle$ for n qubit quantum system

$$|\psi\rangle = \sum_{k=0}^{2^n-1} \alpha_k |k\rangle, \sum_{k=0}^{2^n-1} |\alpha_k|^2 = 1$$

Able to manipulate computations in **parallel**

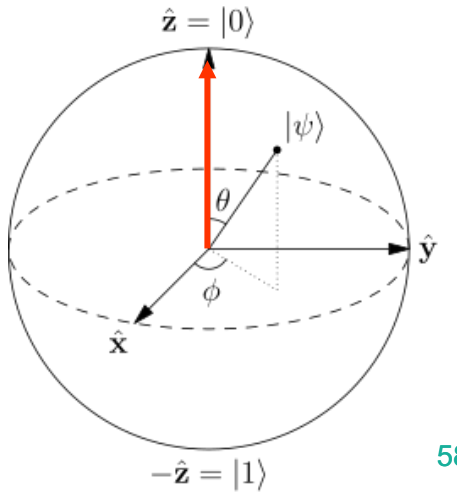
Computation described in unitary quantum logic gates → **Reversible**

Exponentially reduced in space and time

Example :
Pauli-Y-Rotation

$$R_y(\theta) = \begin{pmatrix}\cos\frac{\theta}{2} & -\sin\frac{\theta}{2}\\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2}\end{pmatrix}$$

$$R_y(\theta)|0\rangle = \cos(\theta/2)|0\rangle + \sin(\theta/2)\,|1\rangle$$



Quantum GAN

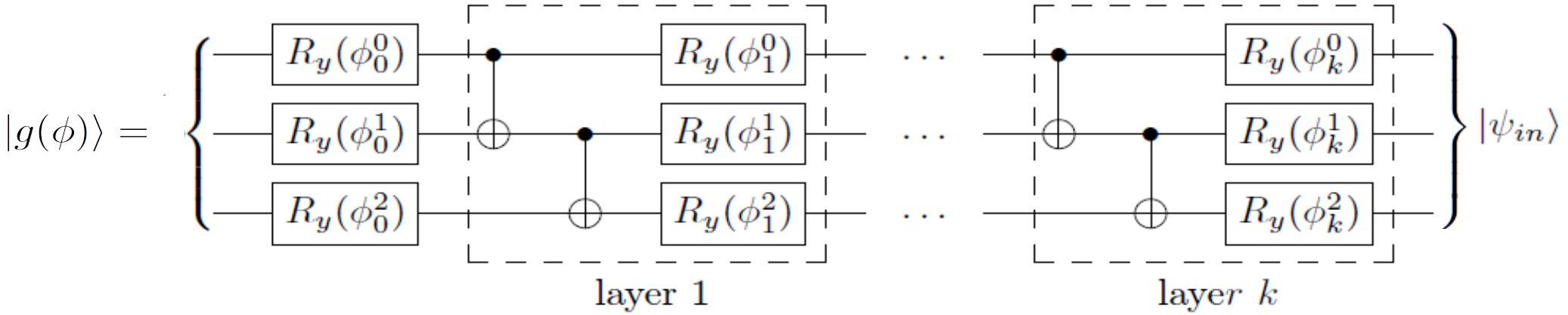
$$G(\phi)\,|\psi_{in}\rangle = |g(\phi)\rangle = \sum_{i=0}^{N-1} \sqrt{p_g^i(\phi)}\,|i\rangle$$

Based on **IBM QGAN** implementation

Hybrid model : Quantum Generator + Classical Discriminator

Efficient in **loading and learning a probability** over discrete values

Increase resolution by adding qubit

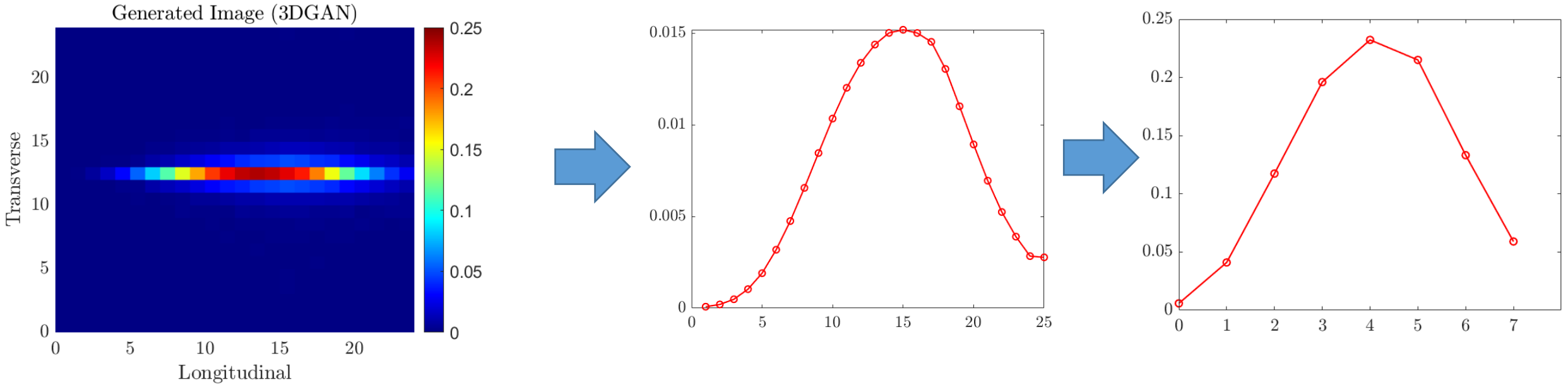


https://doi.org/10.1038/s41534-019-0223-2

A simplified 3DGAN

25x25x25 too big for initial test: reduce to 1D distribution

Focus on longitudinal energy profile



Binned into $2^n = N$ pixels $\rightarrow$ Map to 8 values expressed by 3-qubit generator

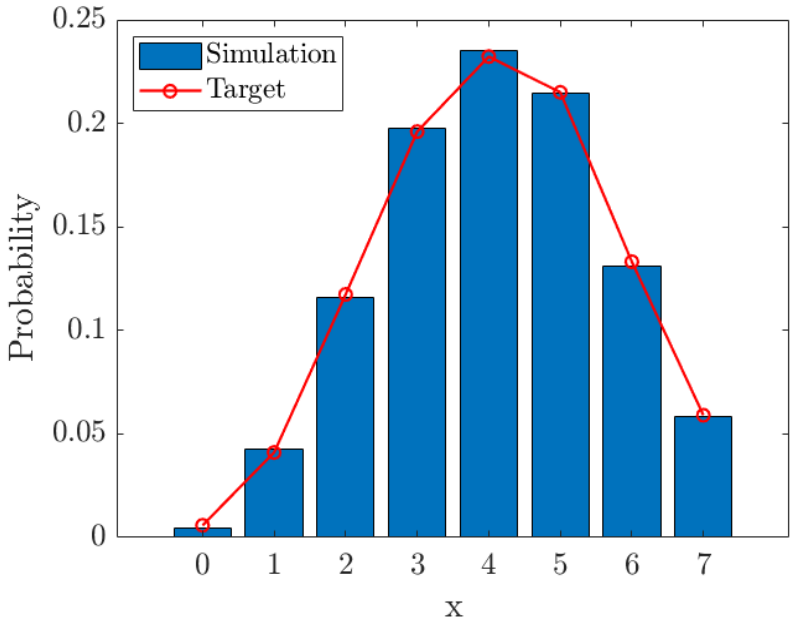
Probability of getting state $|k\rangle$ = (Relative) Energy at pixel k

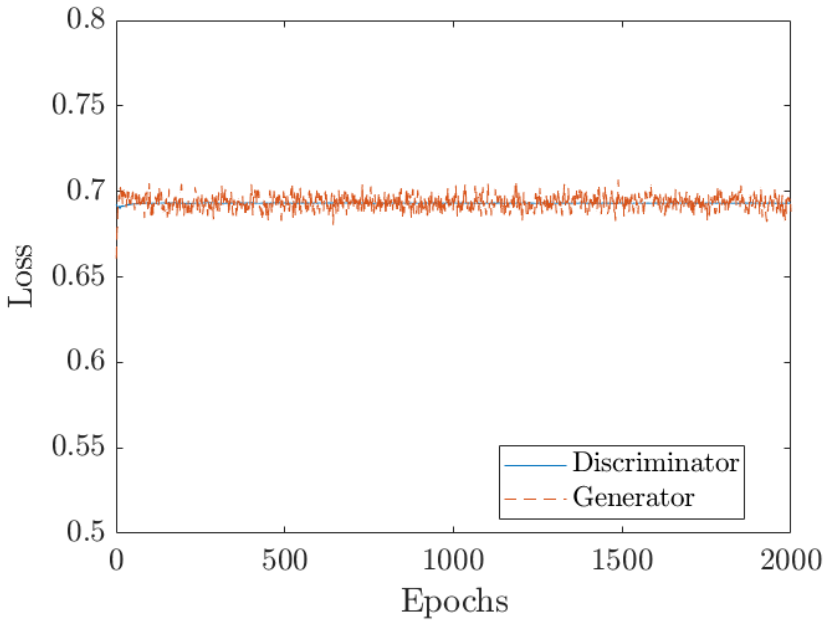
Reproducing longitudinal energy shape

Generator: 3 qubits, 3 layers

Classical Discriminator: 512 nodes + Leaky ReLU → 216 nodes + Leaky ReLU → single-node + sigmoid

AMSGRAD optimizer for both generator and discriminator

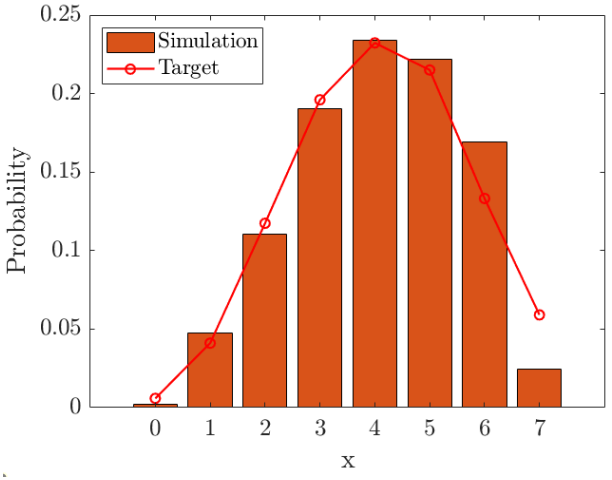




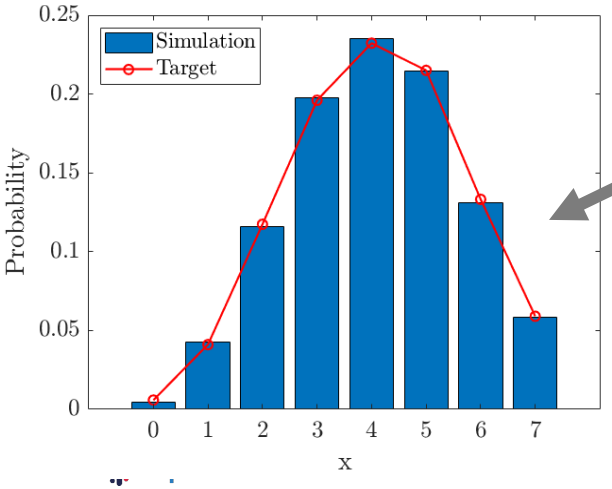
Preparation of the initial state

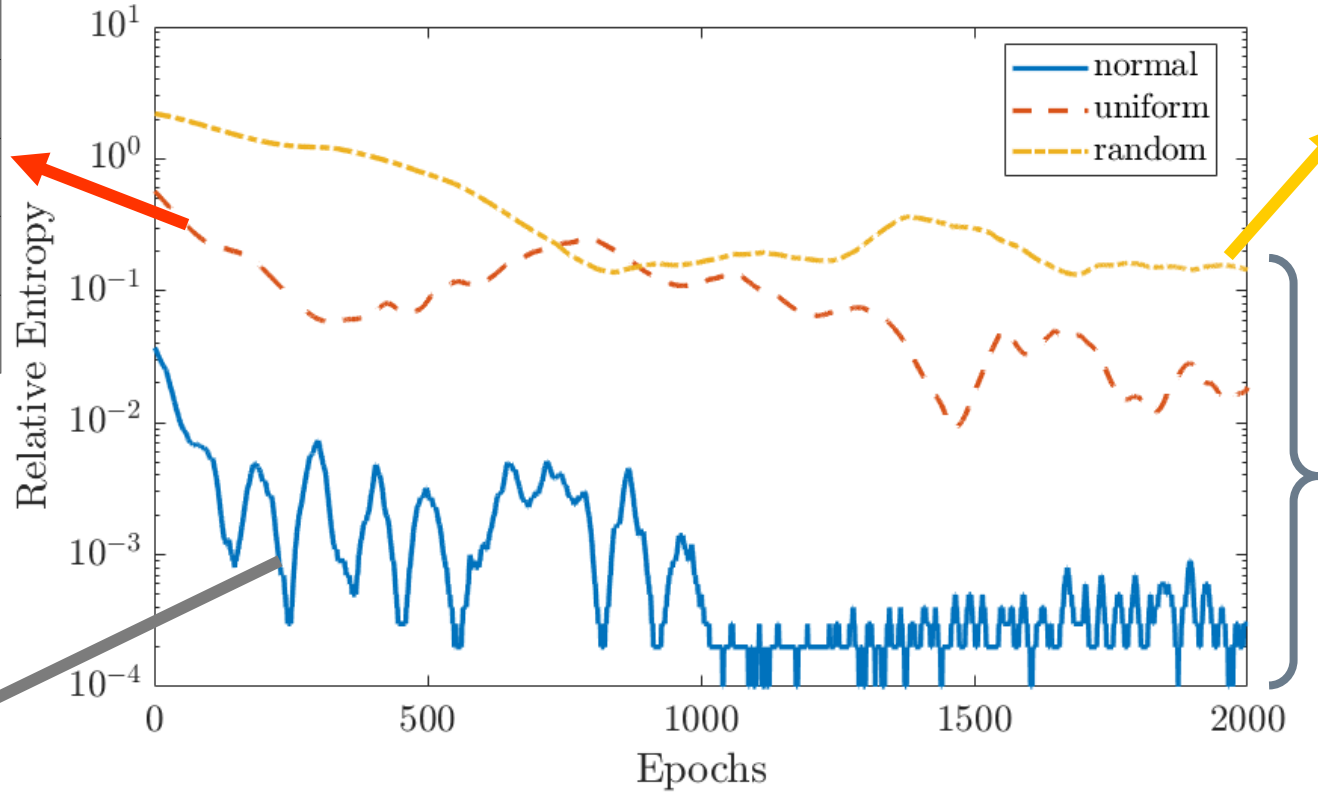
Progress in relative entropy for 3 qubits & depth 3

Uniform

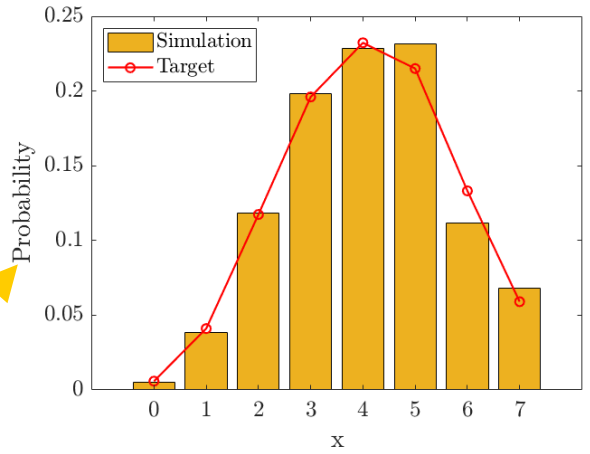


Normal





Random



Quality of result depends on initial states

Understanding performance

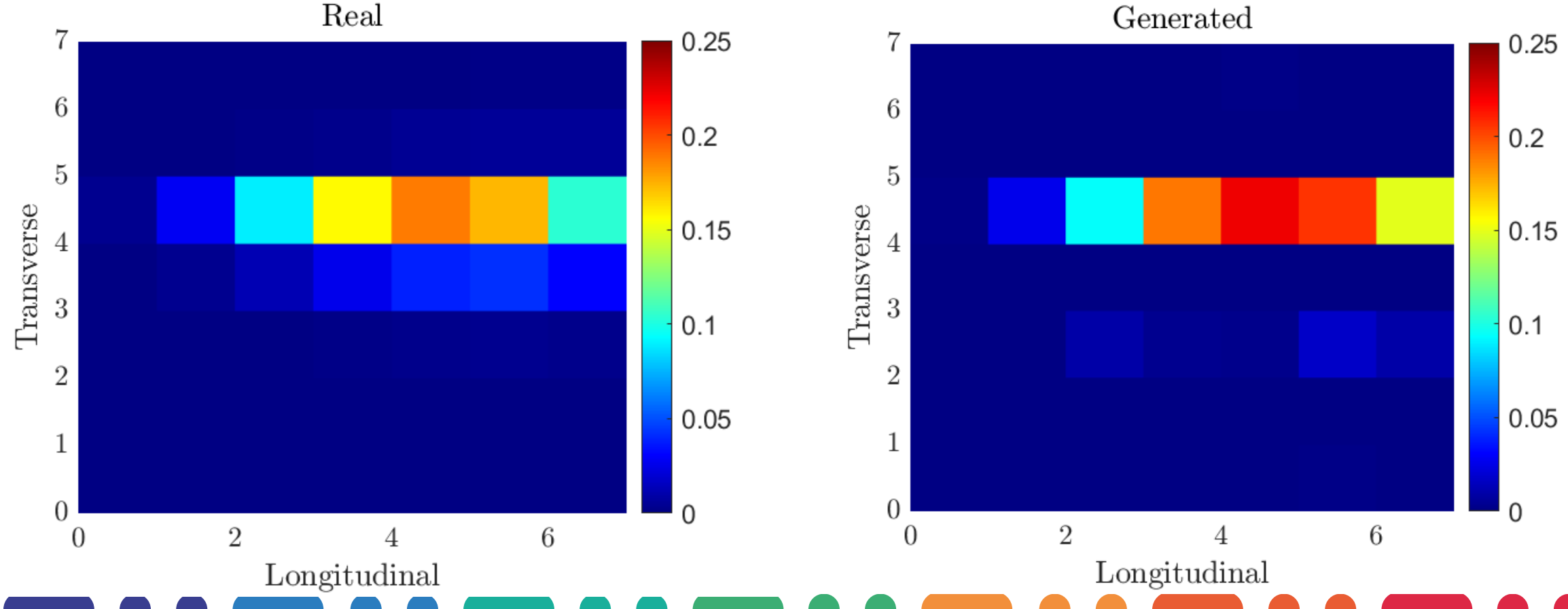
Kolmogrov-Smirnov Statistics : Measures equality between P(x) & Q(x)

→ With 95% confidence level & 1,000 samplings, null hypothesis is accepted if $D_{KS} \leqslant 0.0547$

Initialization	depth	D_{KL}	D_{KS} $(\times 10^{-2})$	Accept/Reject
normal	1	0.031 ± 0.042	0.033 ± 0.024	Accept
	2	$(1.9 \pm 1.7) \cdot 10^{-3}$	0.024 ± 0.002	Accept
	3	$(5.0 \pm 3.0) \cdot 10^{-4}$	0.038 ± 0.008	Accept
uniform	1	0.044 ± 0.003	0.048 ± 0.007	Accept
	2	0.12 ± 0.04	0.072 ± 0.018	Reject
	3	0.070 ± 0.039	0.054 ± 0.017	Accept
random	1	0.051 ± 0.096	0.042 ± 0.015	Accept
	2	0.096 ± 0.162	0.042 ± 0.026	Accept
	3	0.072 ± 0.078	0.059 ± 0.017	Reject

First attempts at 2D (6 qubits)

Using more powerful discriminator (4 hidden layers)
Uniform initialization + depth 1
After 3,000 epochs, able to reproduce similar shape



Conclusions... again

Generative Models are powerful techniques

Fast simulation applications are reaching a high level of accuracy

Performance validation remains a key issue

Need to start designing **common procedures**

Results validations

Integration

Computing resources availability is a driving factor of the problem size we can solve

Benefit from dedicated hardware and accelerators (GPUs, FPGAs, TPUs)

DL development is accelerated by a diversified community

Continuous R&D and **collaboration** with external partner is essential

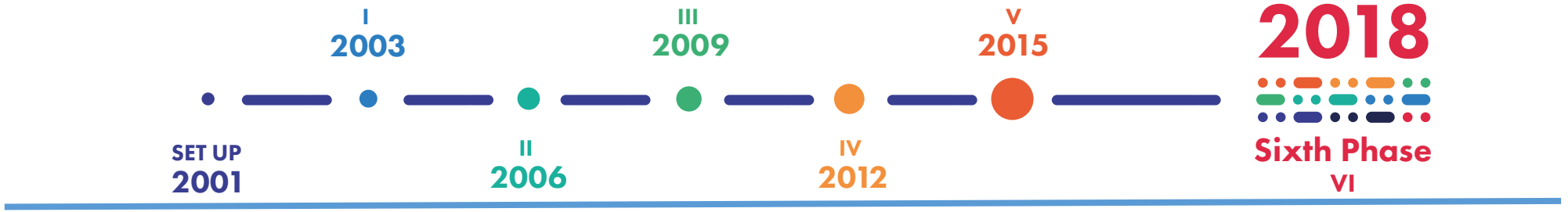
Thanks!

Questions?

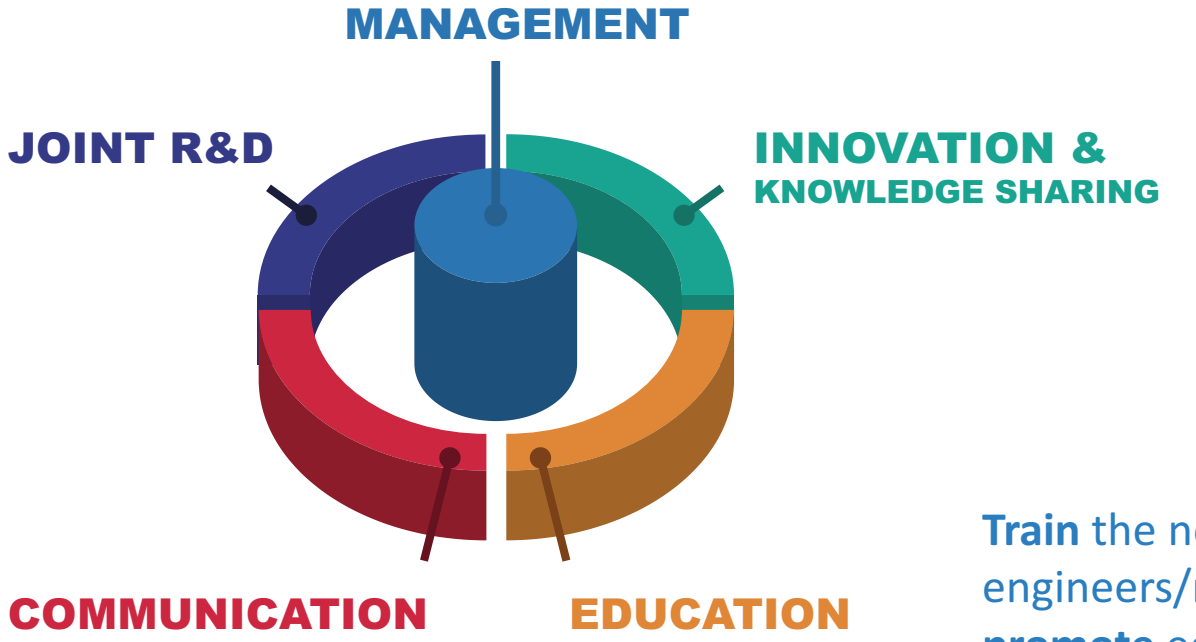
https://openlab.cern

CERN OPENLAB

Driving Innovation since 2001



Evaluate and test state-of-the-art technologies in a challenging environment. **Improve** them in collaboration with industry

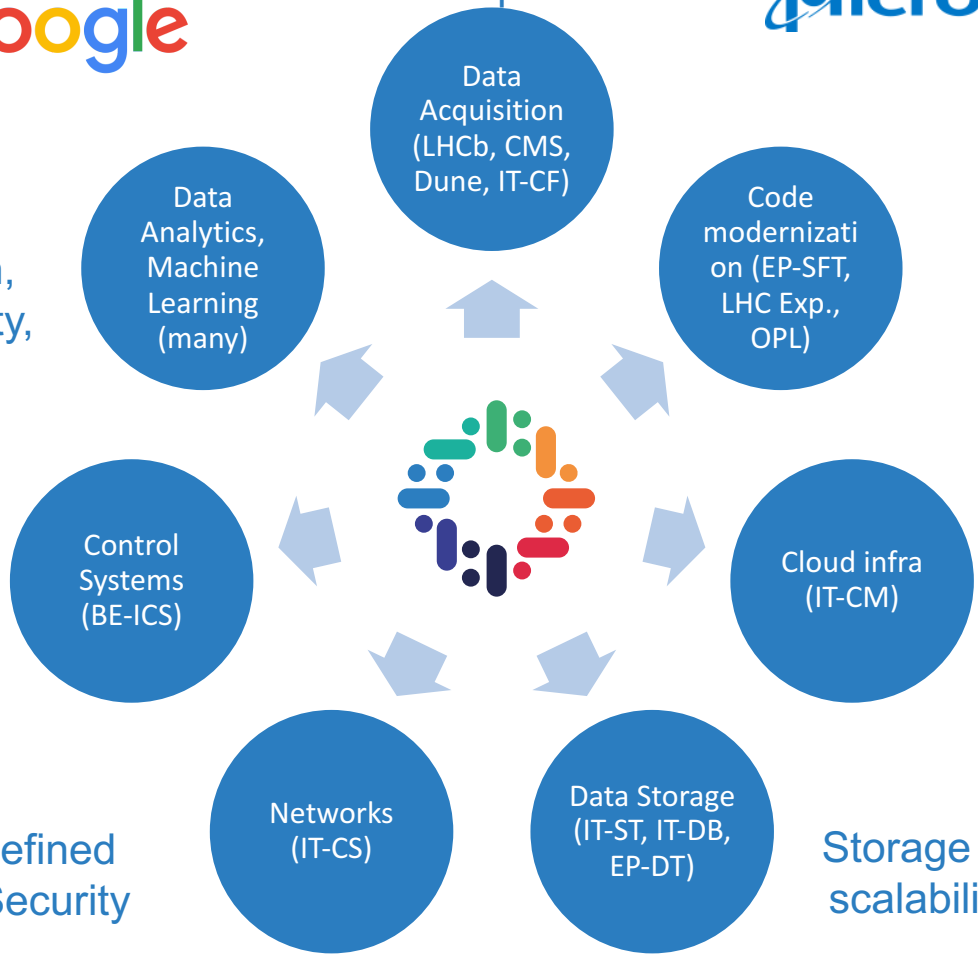


Collaborate with other communities. Create knowledge and innovation.

Communicate results, demostrate **impact**, and **outreach**.

Train the next generation of engineers/researchers, **promote** education and cultural exchanges.

JOINT R&D PROJECTS



High-bandwidth fabrics, accelerated platforms for data acquisition

Fast simulation, Data quality monitoring, anomaly detection, physics data reduction, benchmarking/scalability, systems biology and large-scale multi-disciplinary platforms

HPC, Cloud, Quantum

Cloud technology, containers, scalability

Predictive/proactive maintenance and operations

Software Defined Networks, Security

Storage architectures, scalability, monitoring