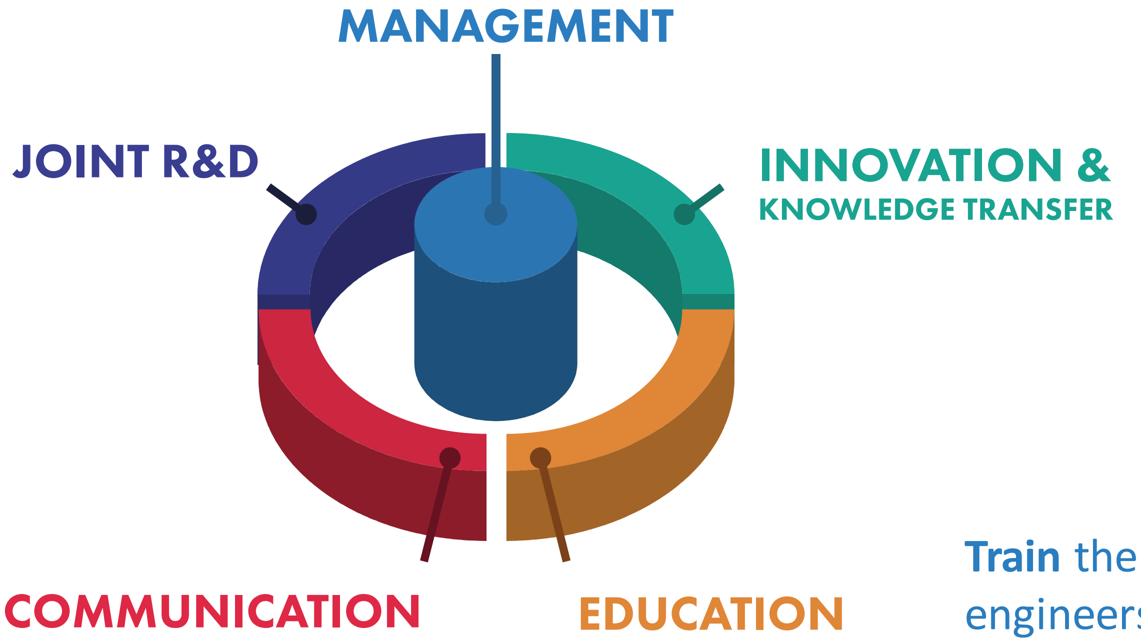
Generative Models for fast simulation

Sofia Vallecorsa

Laboratoire Leprince-Ringuet - Palaiseau - 15/10/2018

CERN OPENLAB

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



Collaborate and exchange ideas with other communities to create knowledge and innovation.

Communicate results, demostrate impact, and reach new audiences.

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

JOINT R&D PROJECTS

High-bandwidth fabrics, accelerated platforms

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

Simulation, HPC on the Cloud, benchmarking

Data Acquisition (LHCb, IT-CF)

Data Analytics, Machine Learning (many)

Code modernization (EP-SFT, IT-CF)

Control Systems (BE-ICS)

Cloud infra (IT-CM)

Networks (IT-CS)

Data Storage (IT-ST, IT-DB)

Predictive/proactive maintenance and operations

Cloud federations, containers, scalability

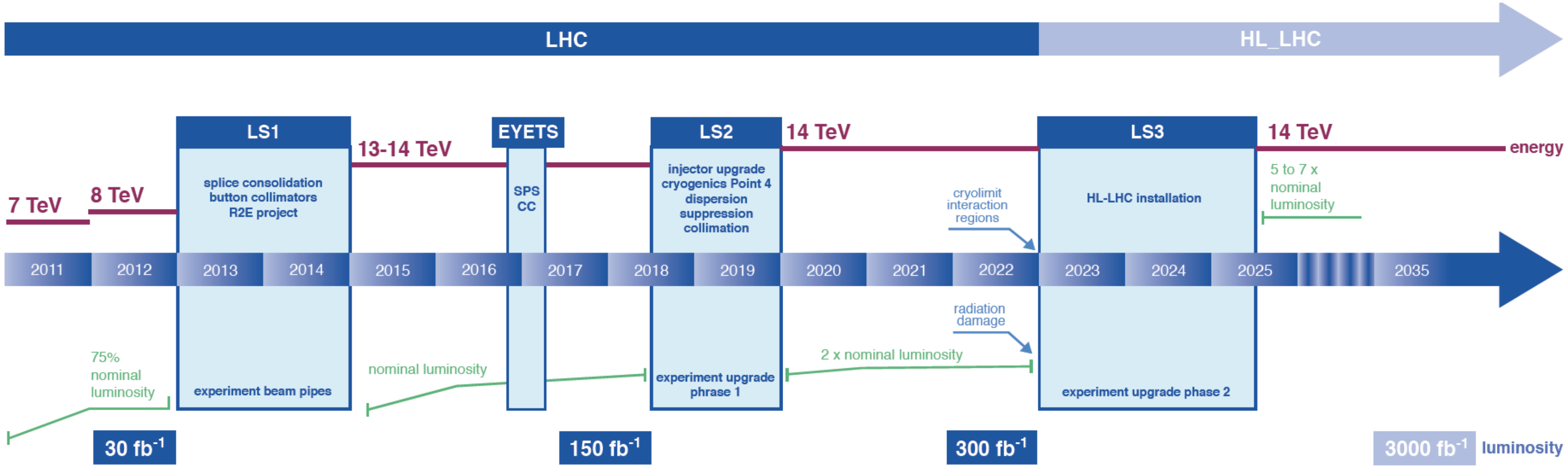
Software Defined Networks, Security

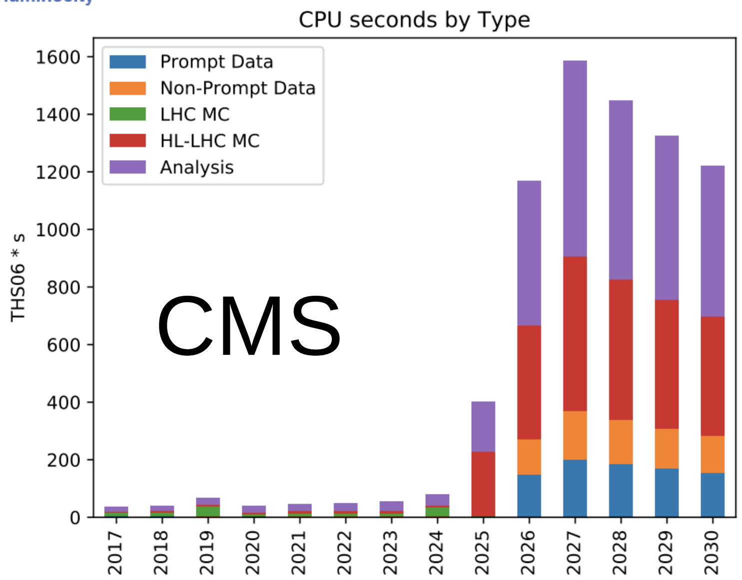
Storage architectures, scalability, monitoring

Outline

- Introduction
- Deep Learning
  - Historic perspective and basic NN concepts
  - Applications
- Generative Models
  - Basics
  - Challenges - Performance
  - Generative Adversarial Networks
- Our work
  - Status
  - Generalisation
  - Computing performance
- Other Applications
- Conclusion - Discussion

The problem





HL-LHC raw data volume increases exponentially

Technology at ~20%/year can bring x6-10

Estimates of resource needs x10 above what is realistic to expect

Today: 50% of WLCG resources are devoted to simulation

Speeding up simulation

Intense R&D activity on code modernisation

- Improve existing code (**Geant4** – scalar processing)
  - Reduce memory consumption
  - Implement event level parallelism
- Prototype fine grained parallelism through the **GeantV** “project”
  - Improved, vectorised physics models
  - Improved, vectorised geometry (**VecGeom**)
  - Smart track level parallel transport
  - Back-propagate improvements to Geant4

http://geant.cern.ch

Fast Simulation

Already used for searches, upgrade studies,…

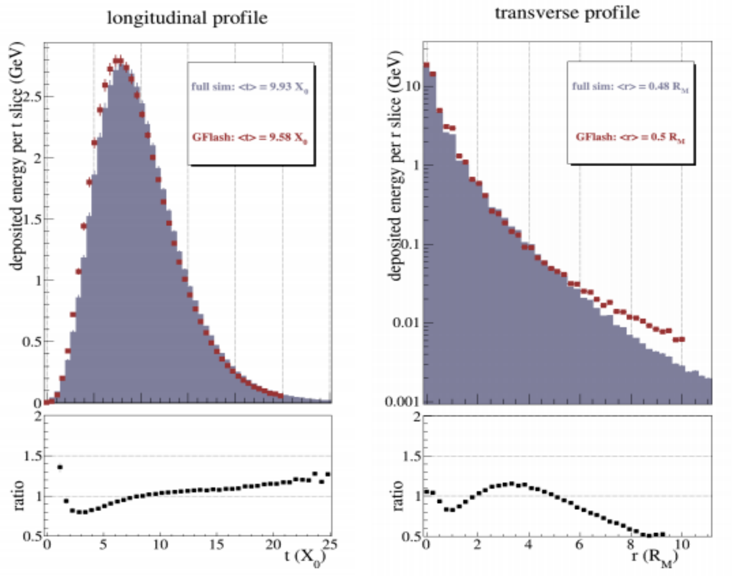
Different techniques

- Shower libraries (pre-simulated EM showers, fwd calorimeters in ATLAS/CMS)
- Shower shapes parametrizations (GFlash,..)
- Fast trackers simulation (ATLAS FATRAS, ..)
- Look-up tables
    - Hit library (LHCb)
- Fully parametrized simulation (DELPHES)

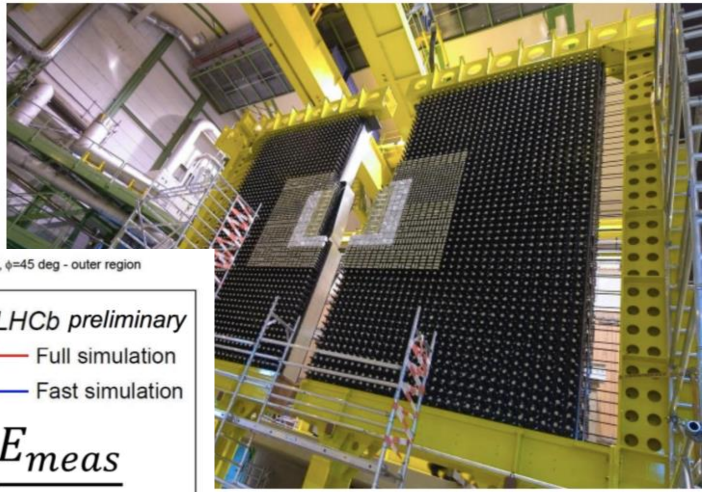
Different performance

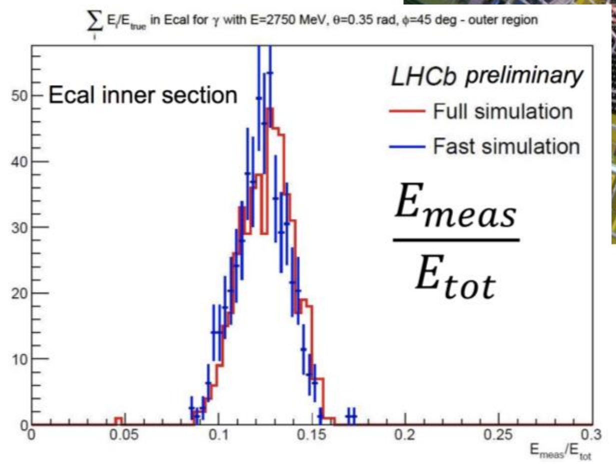
- Different speed improvements (x10 - x1000)
- Different levels of accuracy (~10% wrt full sim)

FCChh



Zaborowska, CHEP2016





M. Rama, LHCb, CHEP2018

A generic framework?

MC need to integrate fast simulation

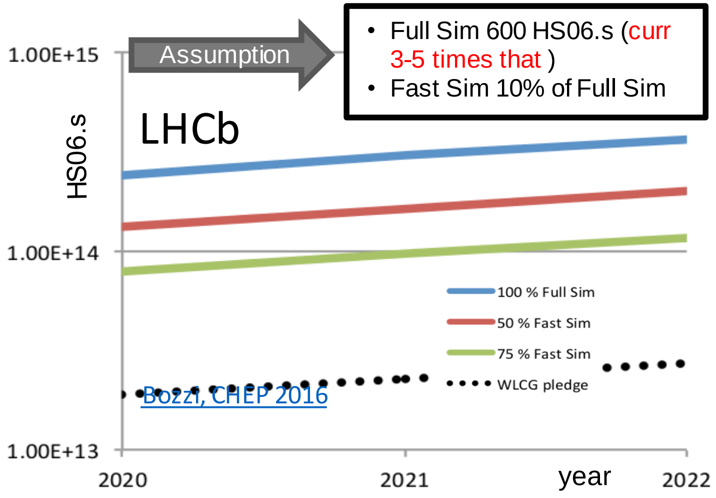
- Geant4 has mechanism to mix fast and full simulation: user-defined models within "envelopes" → few use it

Towards a common framework providing

- Algorithms and tools
- Mechanism to mix fast and full simulation according to particle type and detector

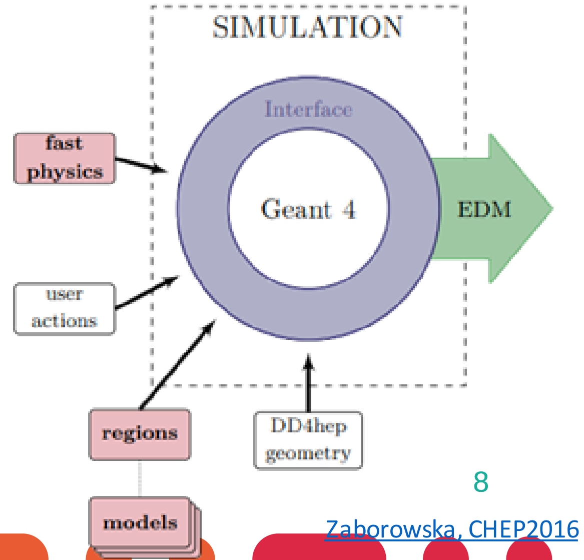
R&D within CERN openlab to develop a generic fully customizable fast sim framework

- Deep Learning based



Bozzi, CHEP 2016

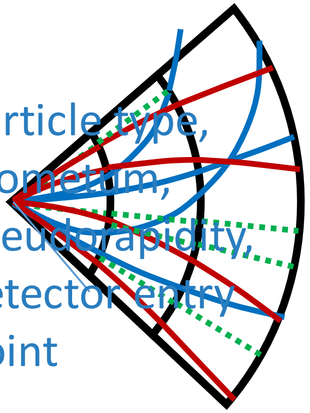
FCC Gaudi framework



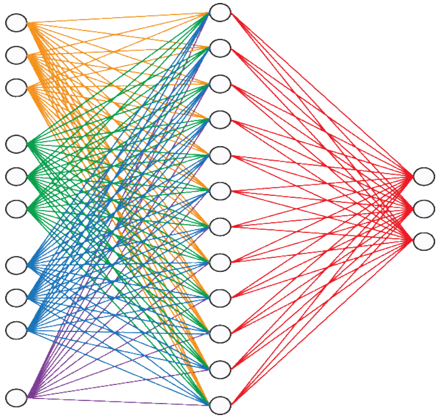
Zaborowska, CHEP2016

Deep Learning for fast sim

EX. SIMULATION OF A CALORIMETER

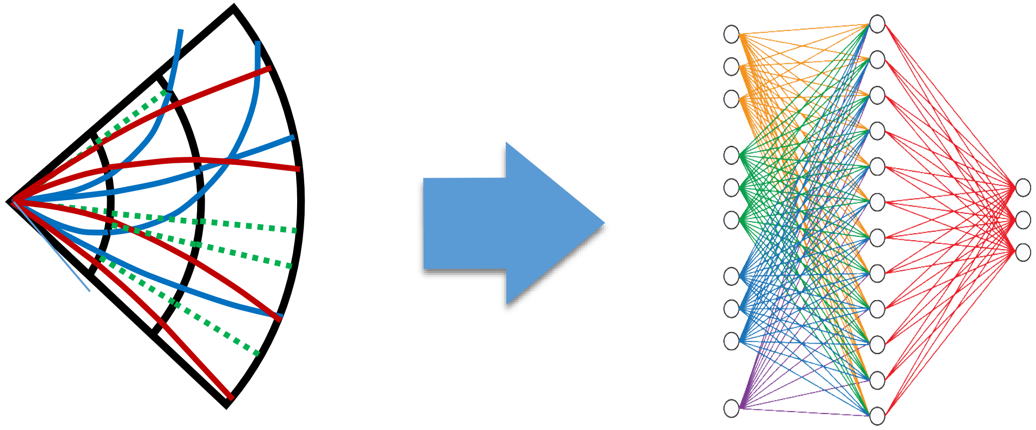


Particle type, momentum, pseudorapidity, detector entry point



Energy depositions in cells

Deep Learning for fast sim



Generic approach

Can encapsulate expensive computations

DNN inference step is generally faster than algorithmic approach

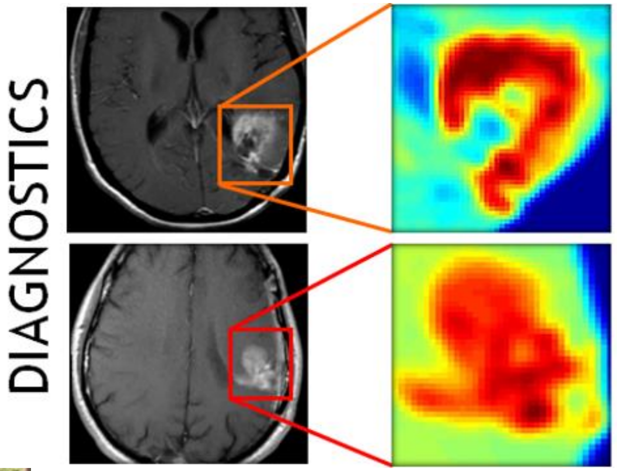
Already parallelized and optimized for GPUs/HPCs.

Industry building highly optimized software, hardware, and cloud services.

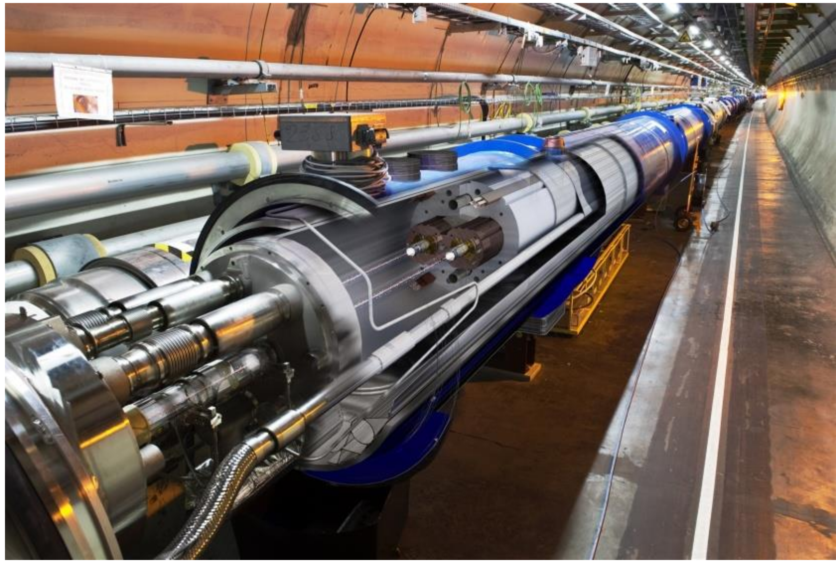
Numerous R&D activities (LHC and beyond)

Deep Learning

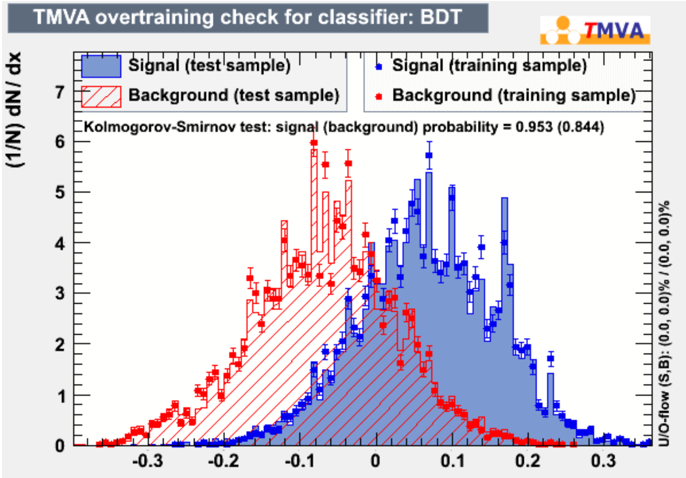
A quick intro

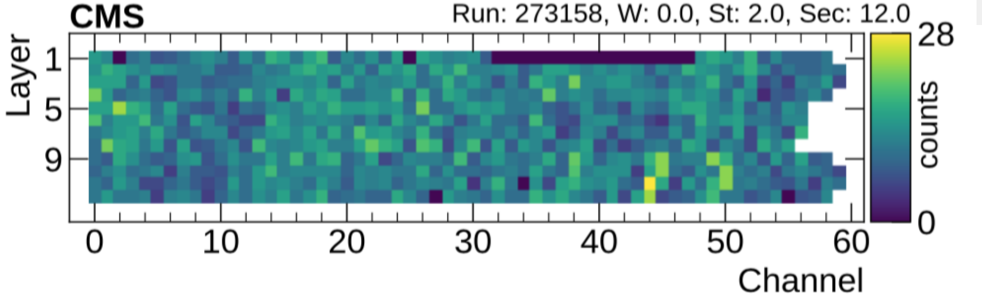


ML in HEP



- Analysis:
  - Classifying signal from background
  - B-tagging and improving energy / mass resolution
- Reconstruction:
  - Improving detector level inputs to reconstruction
  - Particle identification tasks
  - Calibration
- Trigger
- Data Quality Monitoring and Anomaly Detection in control systems
- Computing
  - Estimating dataset popularity, and determining how number and location of dataset replicas
  - Resource optimisation …

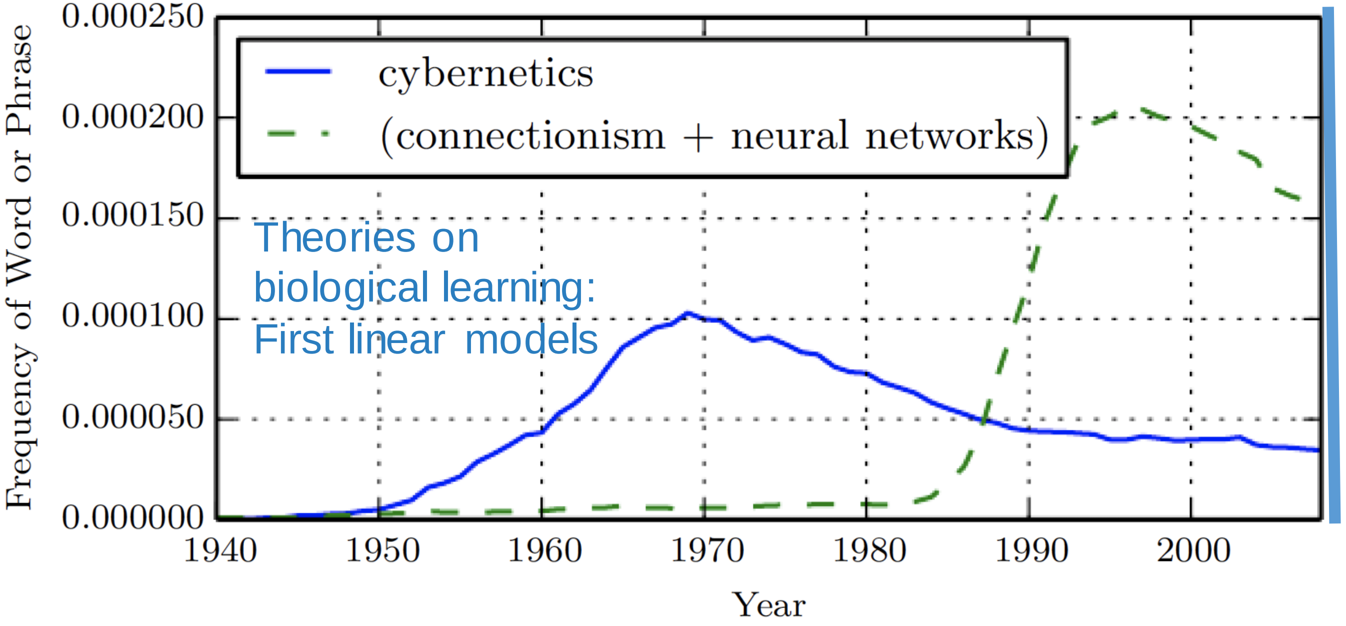




Goodfellow, 2017

Historic perspective

First network inspired by biological systems



Theories on biological learning: First linear models

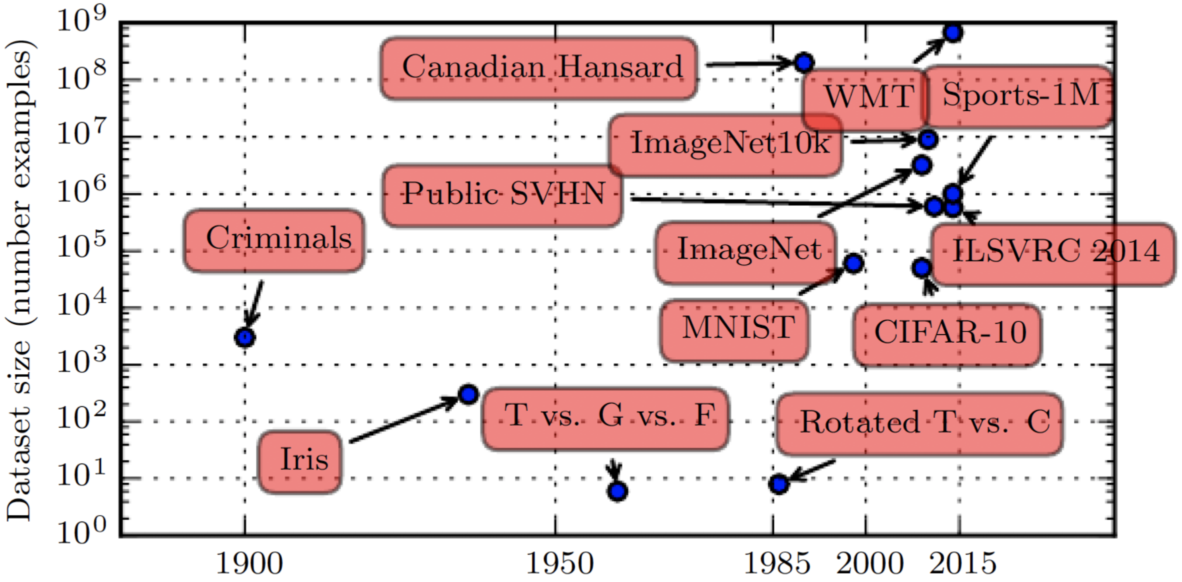
2006:
Modern Deep Learning

Back-propagation to train shallow NN: apply the derivatives "chain rule" to speed up NN training.

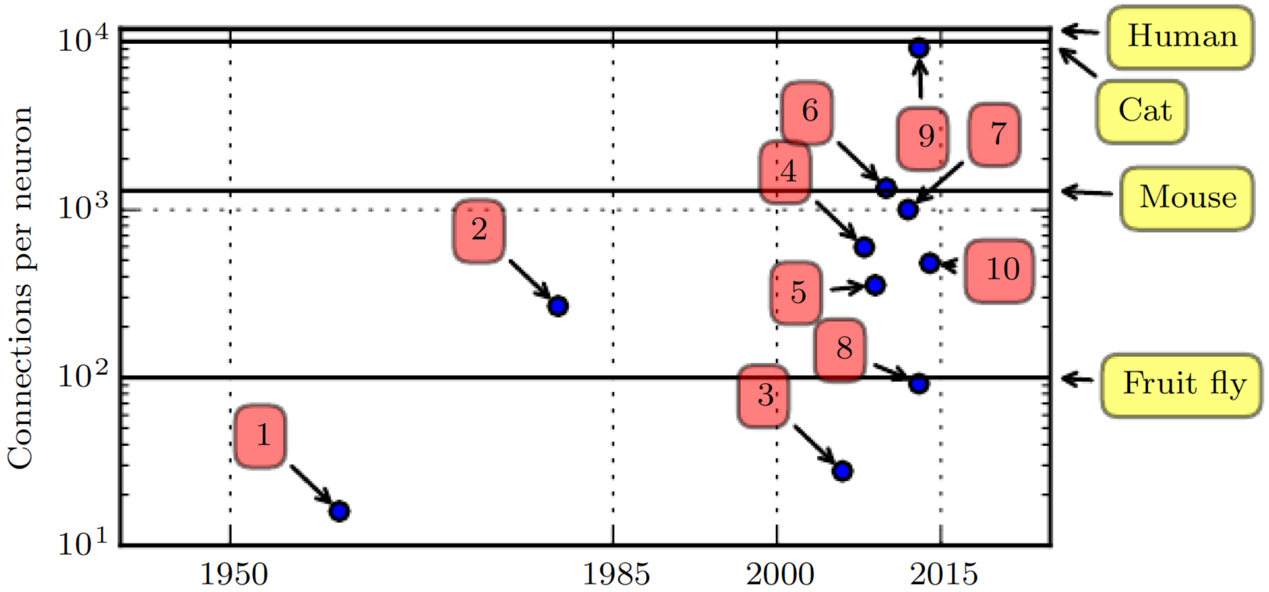
Goodfellow, 2017

Increasing sizes

Datasets:



Model connections:



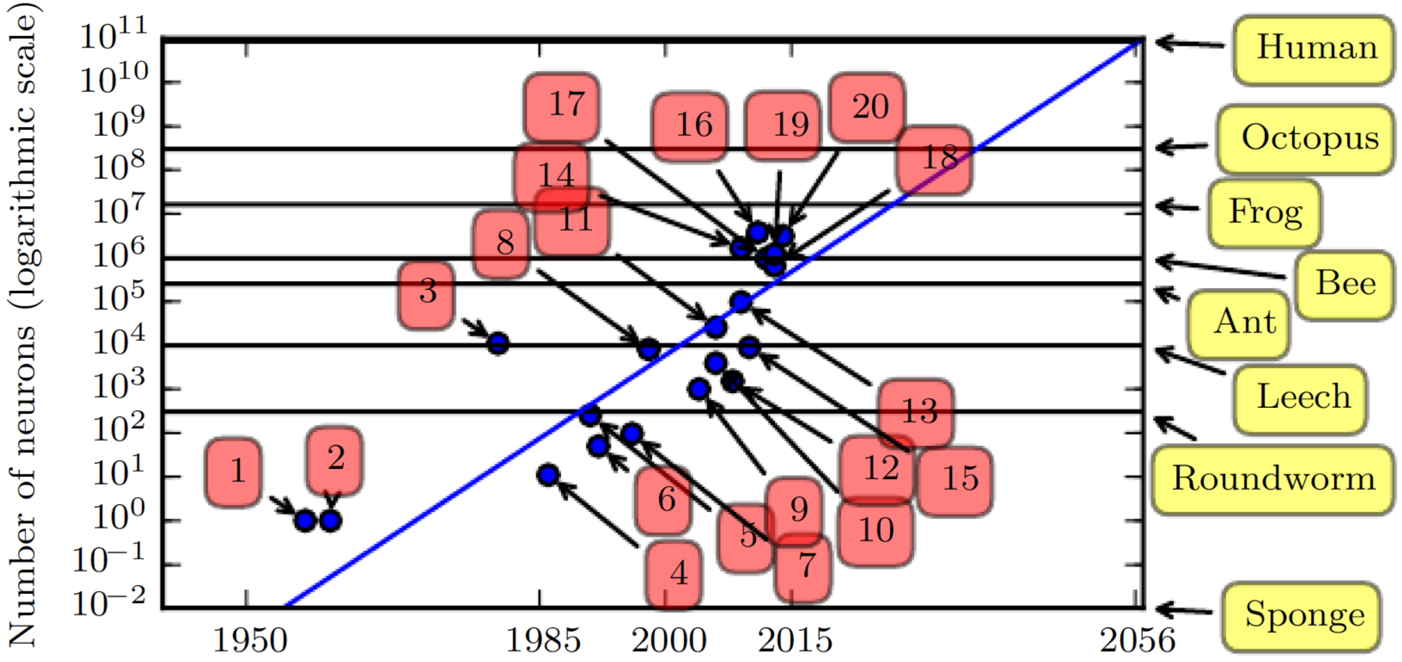
1. Adaptive linear element (Widrow and Hoff, 1960)
2. Neocognitron (Fukushima, 1980)
3. GPU-accelerated convolutional network (Chellapilla *et al.*, 2006)
4. Deep Boltzmann machine (Salakhutdinov and Hinton, 2009a)
5. Unsupervised convolutional network (Jarrett *et al.*, 2009)
6. GPU-accelerated multilayer perceptron (Ciresan *et al.*, 2010)
7. Distributed autoencoder (Le *et al.*, 2012)
8. Multi-GPU convolutional network (Krizhevsky *et al.*, 2012)
9. COTS HPC unsupervised convolutional network (Coates *et al.*, 2013)
10. GoogLeNet (Szegedy *et al.*, 2014a)

CERN openlab

Increasing sizes (II)

Goodfellow, 2017

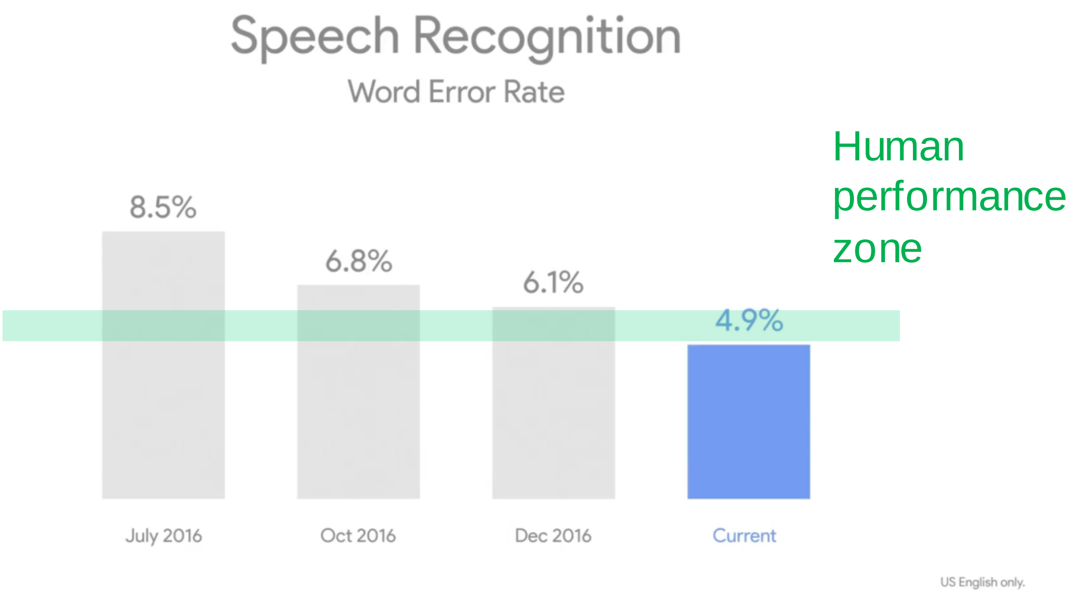
Model neurons:



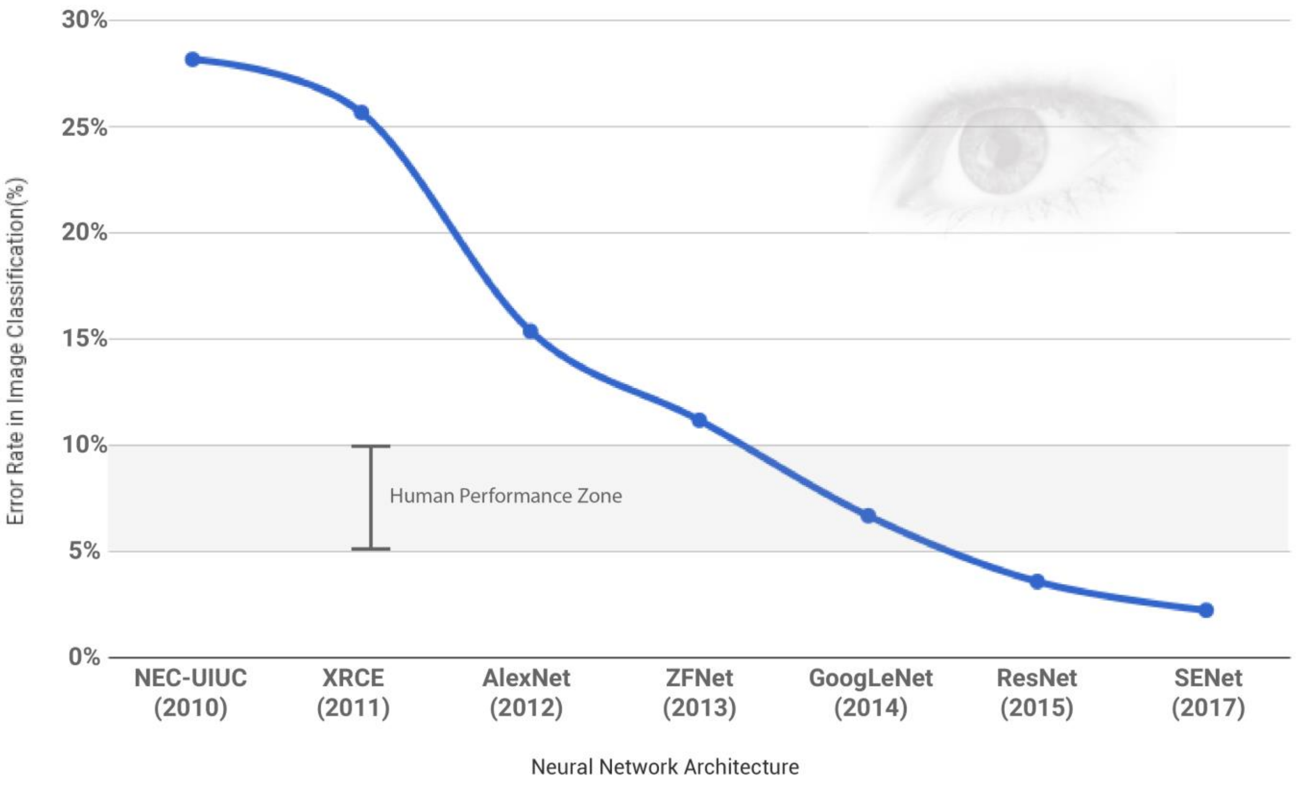
1. Perceptron (Rosenblatt, 1958, 1962)
2. Adaptive linear element (Widrow and Hoff, 1960)
3. Neocognitron (Fukushima, 1980)
4. Early back-propagation network (Rumelhart *et al.*, 1986b)
5. Recurrent neural network for speech recognition (Robinson and Fallside, 1991)
6. Multilayer perceptron for speech recognition (Bengio *et al.*, 1991)
7. Mean field sigmoid belief network (Saul *et al.*, 1996)
8. LeNet-5 (LeCun *et al.*, 1998b)
9. Echo state network (Jaeger and Haas, 2004)
10. Deep belief network (Hinton *et al.*, 2006)
11. GPU-accelerated convolutional network (Chellapilla *et al.*, 2006)
12. Deep Boltzmann machine (Salakhutdinov and Hinton, 2009a)
13. GPU-accelerated deep belief network (Raina *et al.*, 2009)
14. Unsupervised convolutional network (Jarrett *et al.*, 2009)
15. GPU-accelerated multilayer perceptron (Ciresan *et al.*, 2010)
16. OMP-1 network (Coates and Ng, 2011)
17. Distributed autoencoder (Le *et al.*, 2012)
18. Multi-GPU convolutional network (Krizhevsky *et al.*, 2012)
19. COTS HPC unsupervised convolutional network (Coates *et al.*, 2013)
20. GoogLeNet (Szegedy *et al.*, 2014a)

Performance growth

Closing in on narrow AI!



2017 Google results

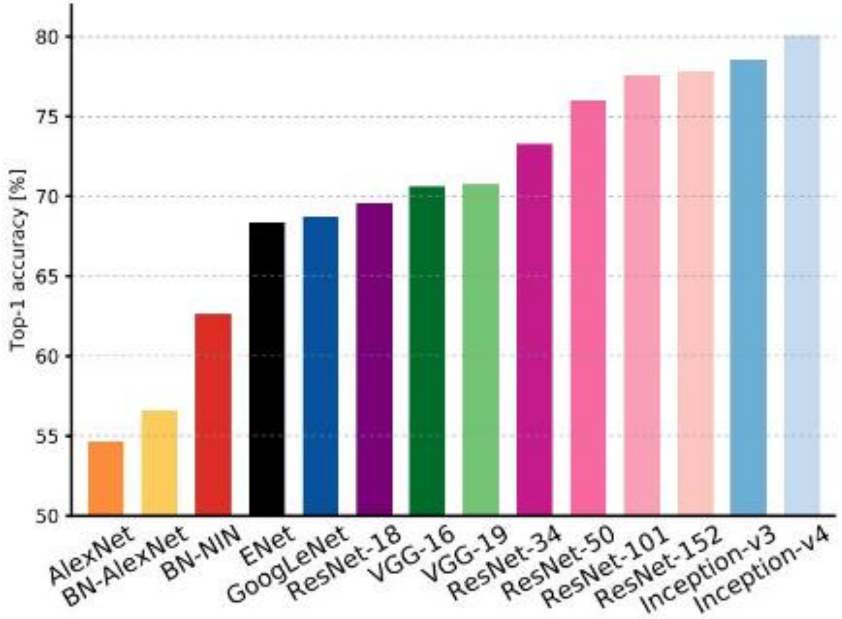


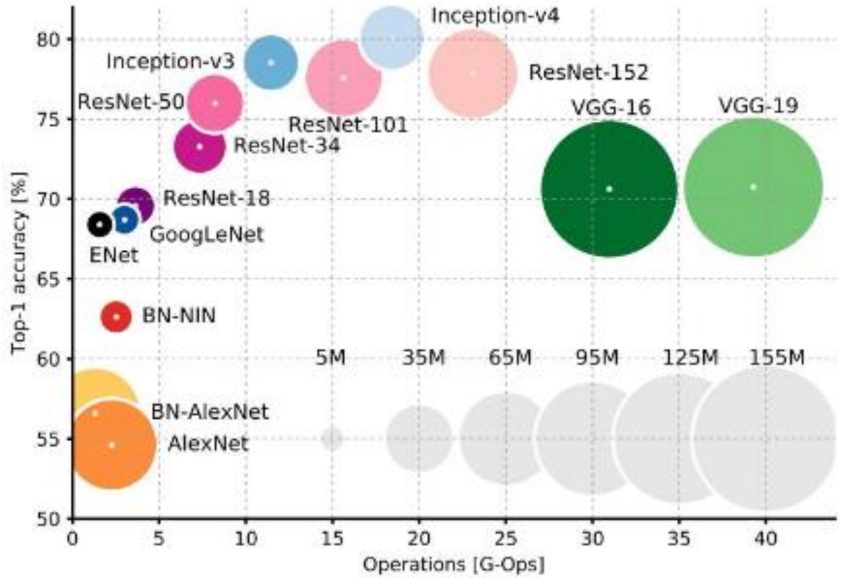
https://arxiv.org/pdf/1409.0575.pdf

Imagenet Large Scale Visual Recognition Challenge

Imagenet dataset: >14 M labelled images across 20K hierarchical categories

ILVRC Challenge started in 2010 with 100 classes: 1000 classes now





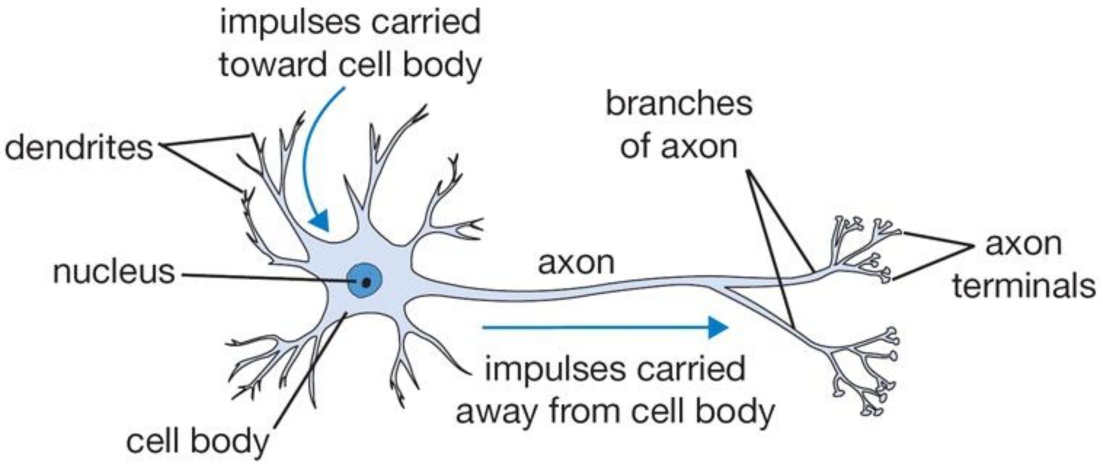
2017: 28/30 participants reached better than human error rate

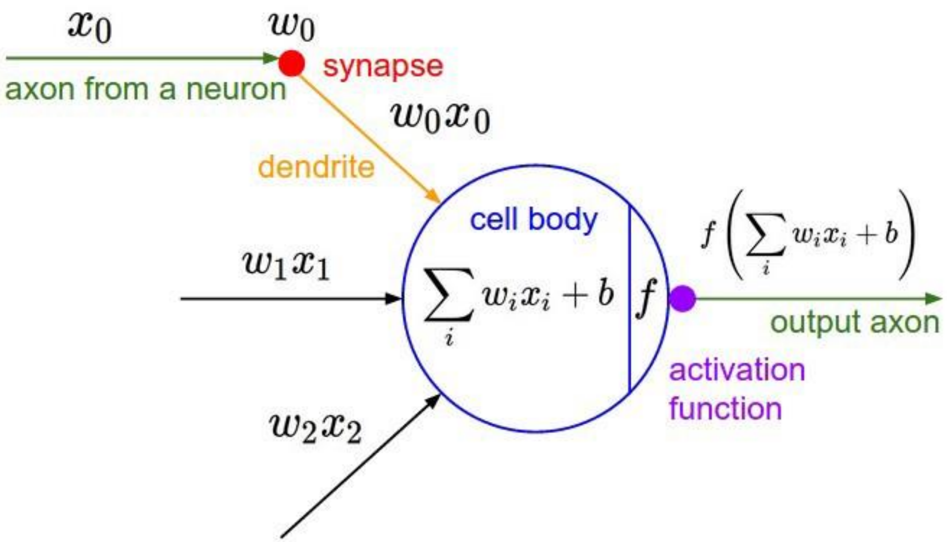
2018 challenge introduces video reconstruction

Basics

Artificial Neural Networks

ANN are computational models inspired by biological neural networks.





Feed-forward networks

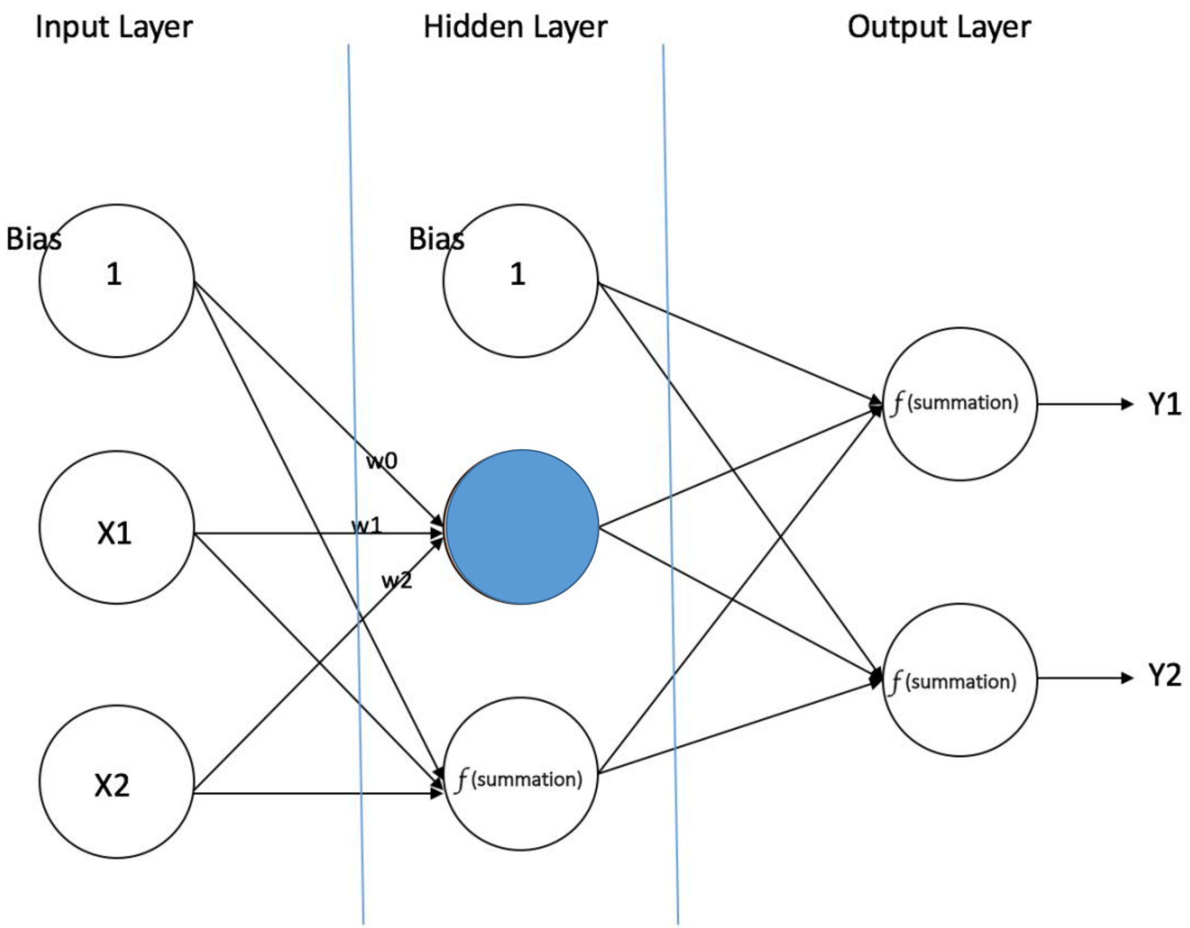
Multiple nodes arranged in **layers**.

Nodes from adjacent layers have **connections** (with weights).

Ex. fully-connected layer

Multi Layer Perceptron (MLP) contains one or more hidden layers

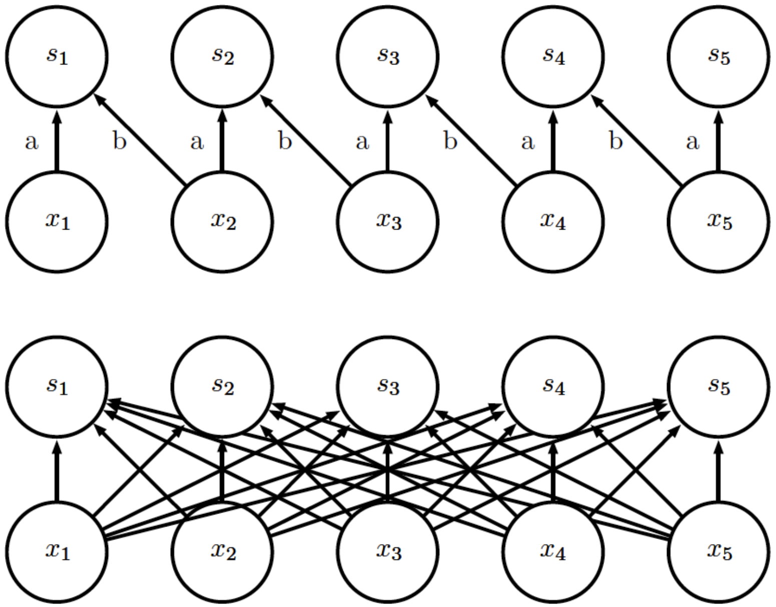
Solving a MLP can be thought of as matrix multiplication calculation



NN with at least one hidden layer are universal approximators

Convolutional Neural Networks

- Applicable to any input that is laid out on a grid (1-D, 2-D, 3-D, …)
- Sparse connections
- Parameter sharing
- Automatically generalize across spatial translations of inputs
- Easily scalable to process large images and video sequences



Convolutions

x ∈ $\mathbb{R}^M$ and kernel **u** ∈ $\mathbb{R}^k$

discrete convolution **x ∗ u** is vector of size M-k+1

$$(x * u)_i = \sum_{b=0}^{k-1} x_{i+b} u_b$$

2D convolutions extract features from input image using "small squares of input data"

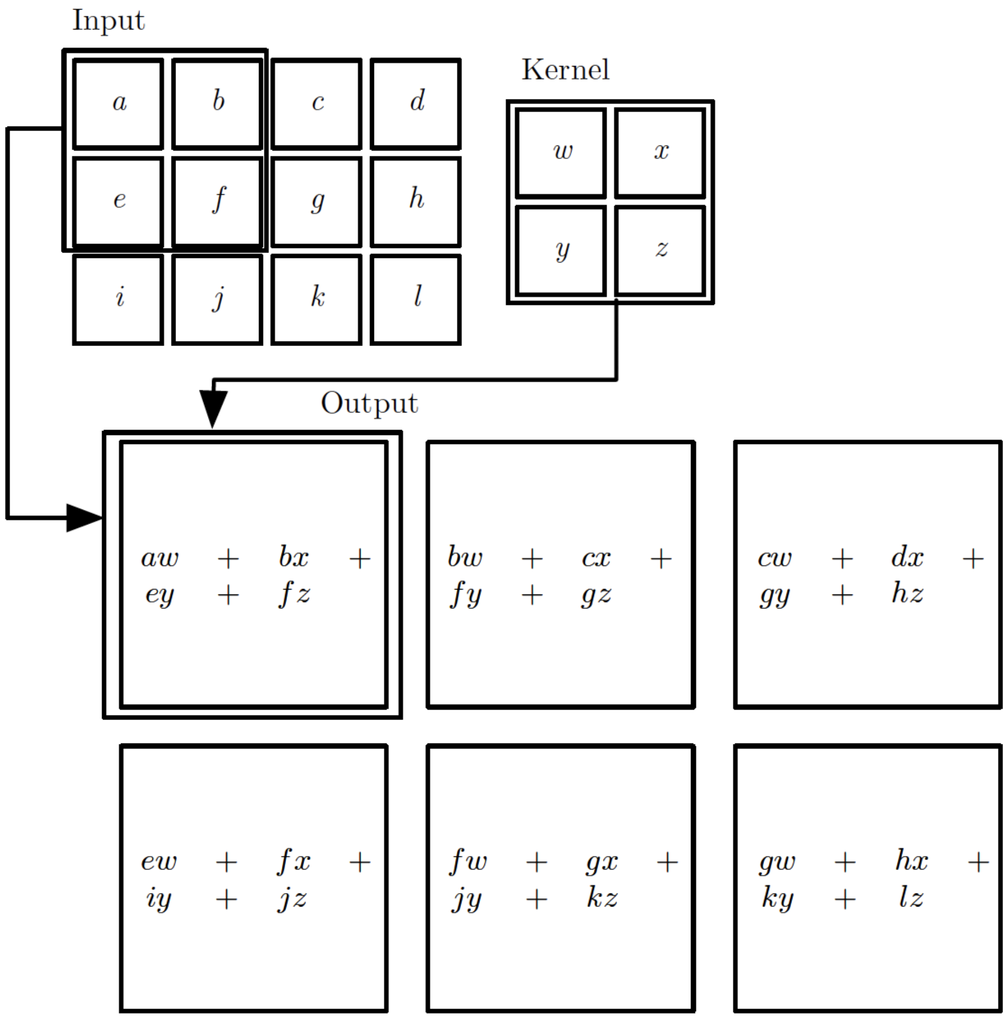
preserve spatial relationship between pixels.

Ex: 5 x 5 input image, 3 x 3 kernel

Slide the filter matrix

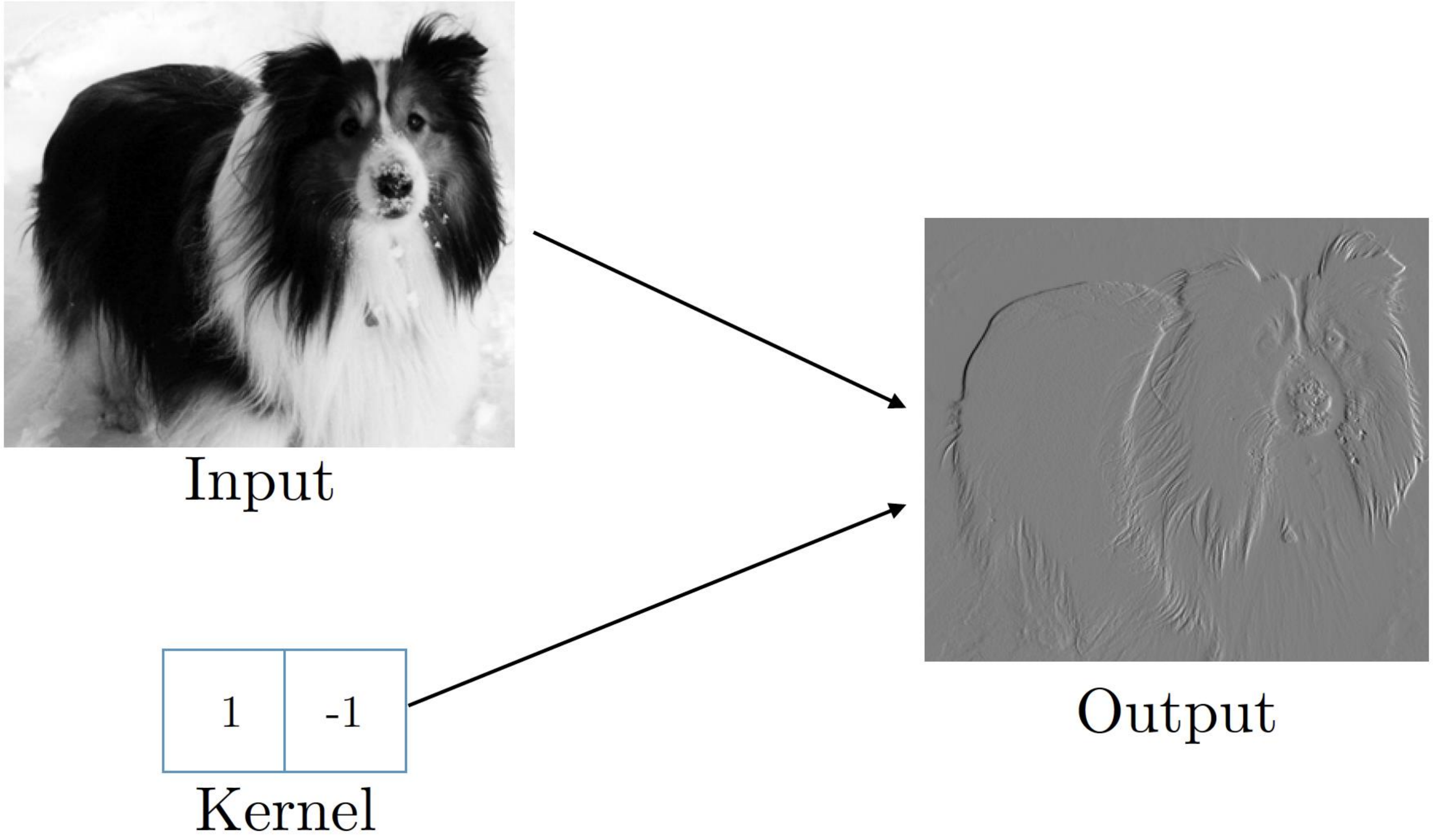
Element wise multiplication

Sum of the multiplication outputs



CERN openlab

Ex. Edge detection



Example

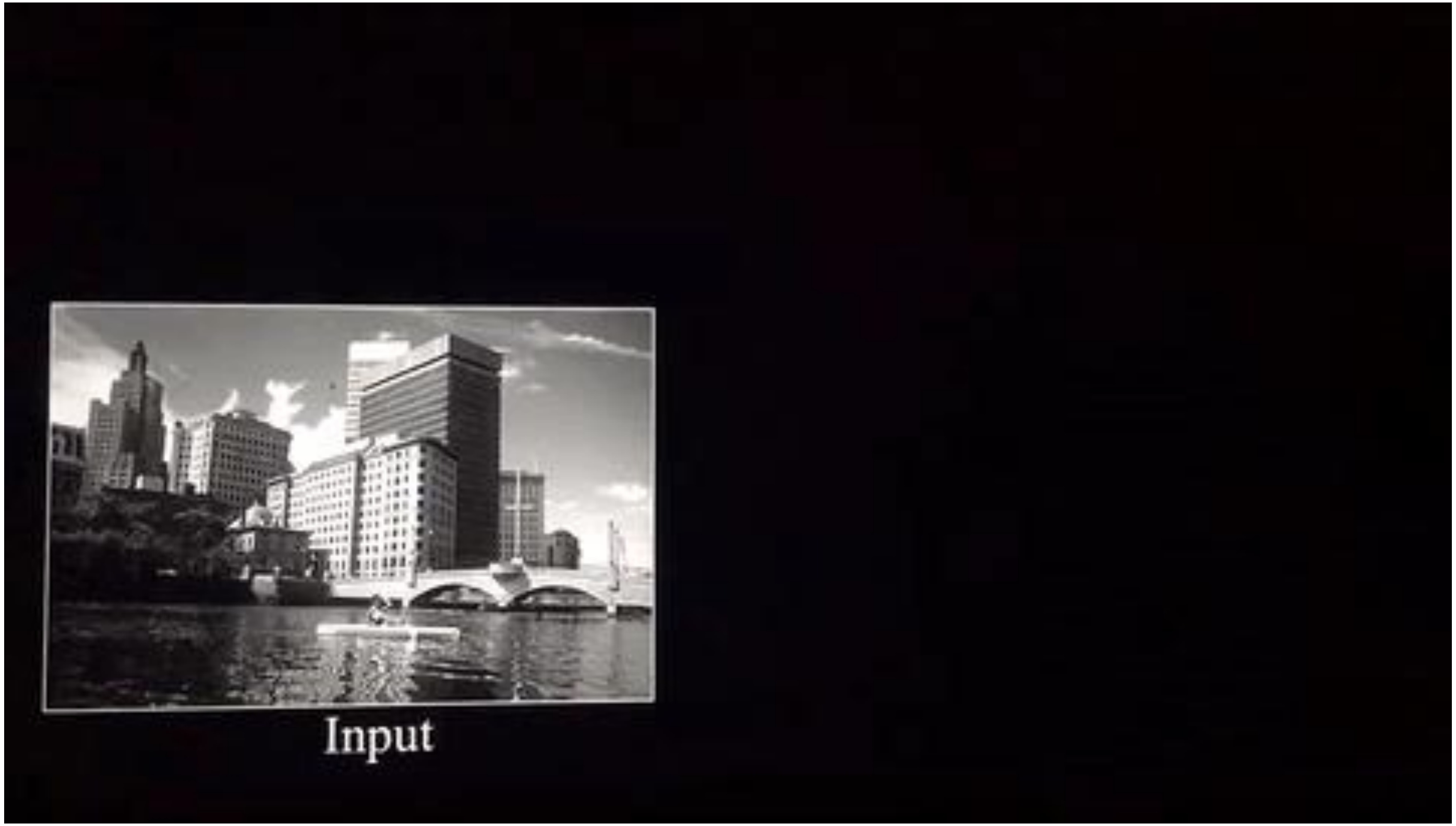
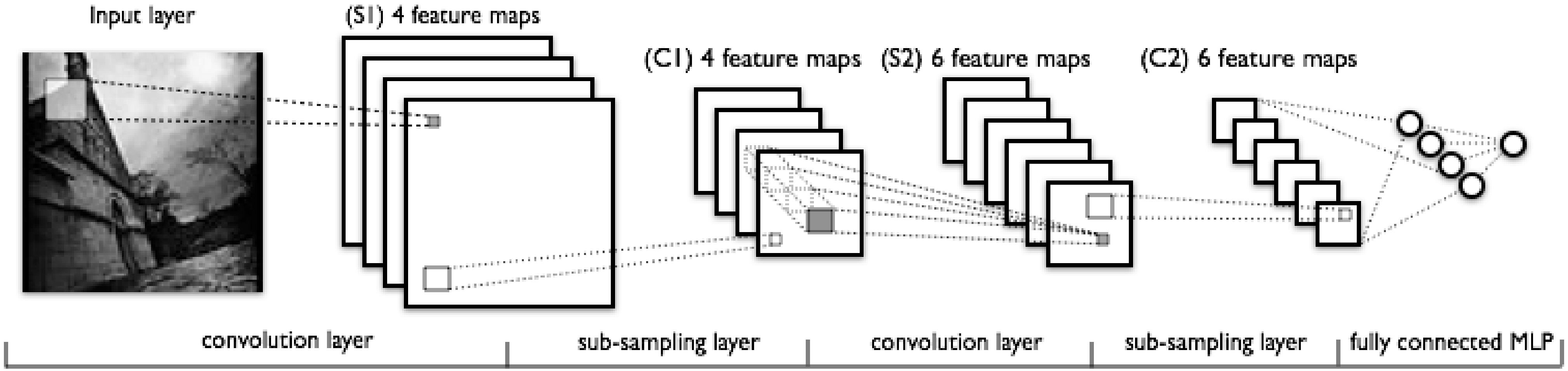


Image from CVPR 2012 tutorial

LeNet

Pioneering 7 layers CNN to recognize hand-written numbers on checks

Digitized in 32x32 pixel greyscale input images.



Need larger CNN to process higher resolution images

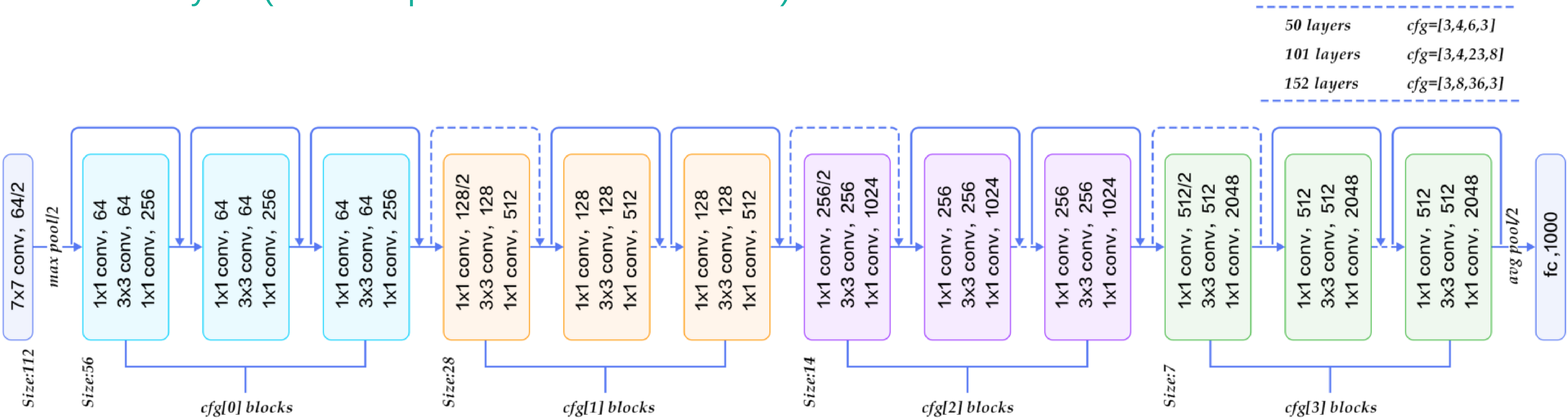
availability of computing resources!

LeCun et al, 1998

ILSVRC 2015: ResNet

Residual Neural Network introduced gate recurrent units and heavy batch normalization.

- 152 layers (with less parameters than VGGNet): 3.57% error rate

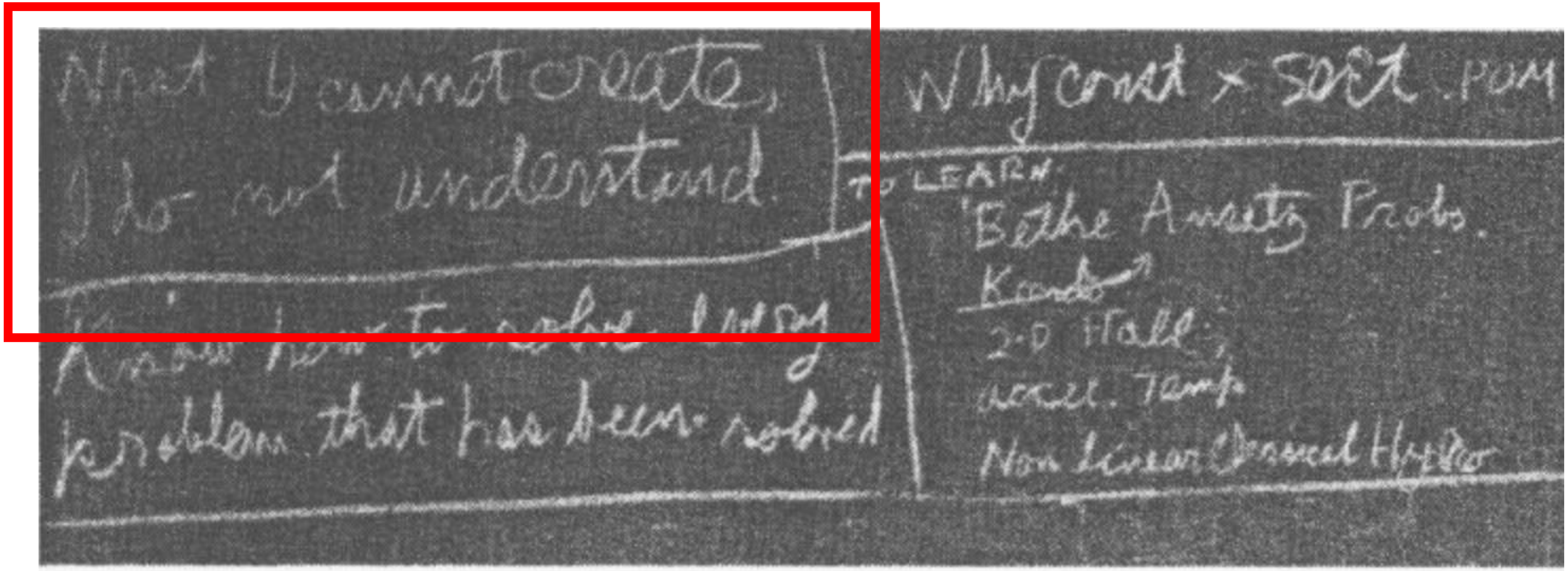


Kaiming He et al, 2015

Generative Models

What I cannot create I don't understand

R. Feynman



Generative models

The problem:

Assume data sample follows p_{data} distribution

Can we draw samples from distribution p_{model} such that $p_{model} \approx p_{data}$?

A well known solution:

Assume some form for p_{model} (using prior knowledge, parameterized by θ)

Find the maximum likelihood estimator

$$\theta^* = \arg\max_\theta \sum_{\mathbf{x} \in \mathcal{D}} \log(p_{\text{model}}(\mathbf{x}; \theta))$$

draw samples from $p_{\theta *}$

Generative models don't assume any prior form for p_{models}

Use Neural Networks instead

Generative models for simulation

Many models: Generative Stochastic Networks, Auto-Econders, Generative Adversarial Networks ..

Realistic generation of samples

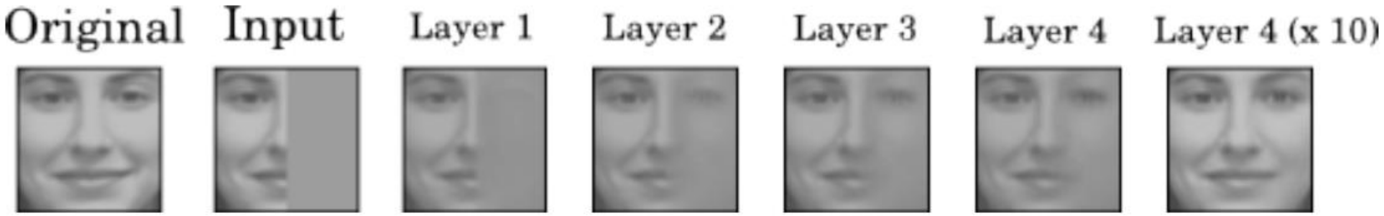
Use complicated probability distributions

Optimise multiple output for a single input

Can do interpolation

Work well with missing data

'Small blue bird with black wings' →
'Small yellow bird with black wings'



https://arxiv.org/pdf/1605.05396.pdf

Ranzato, Susskind, Mnih, Hinton, IEEE CVPR 2011

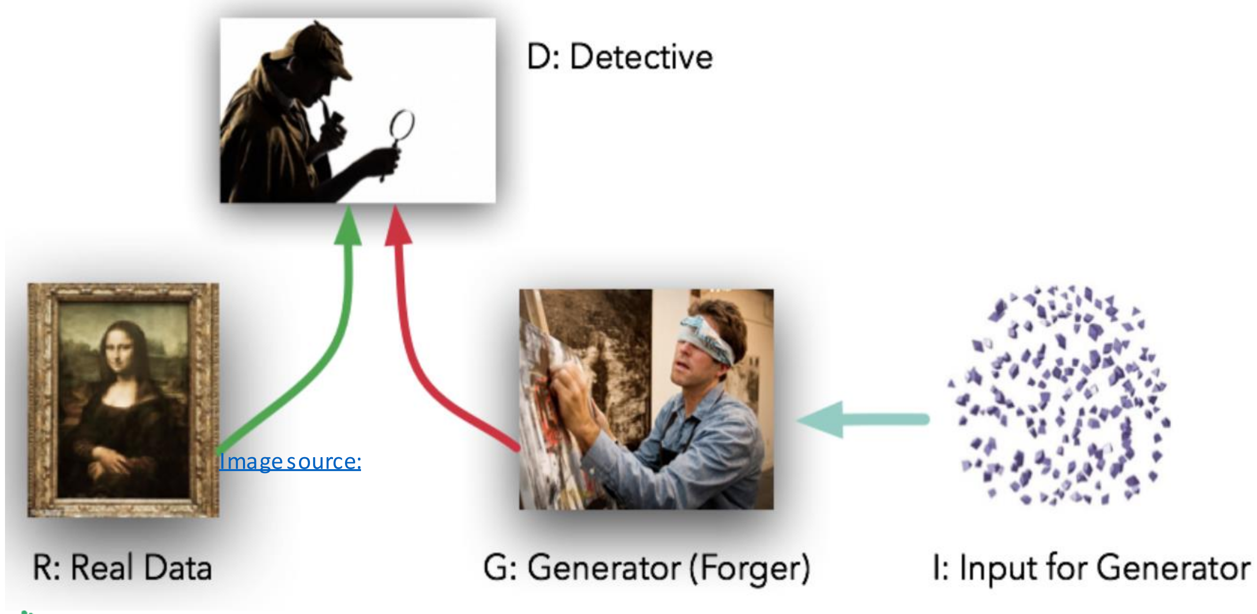
Generative adversarial networks

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



The counterfeiter/detective case

Counterfeiter shows the Monalisa

Detective says it is fake and gives feedback

Counterfeiter makes new Monalisa based on feedback

Iterate until detective is fooled

Generative adversarial training

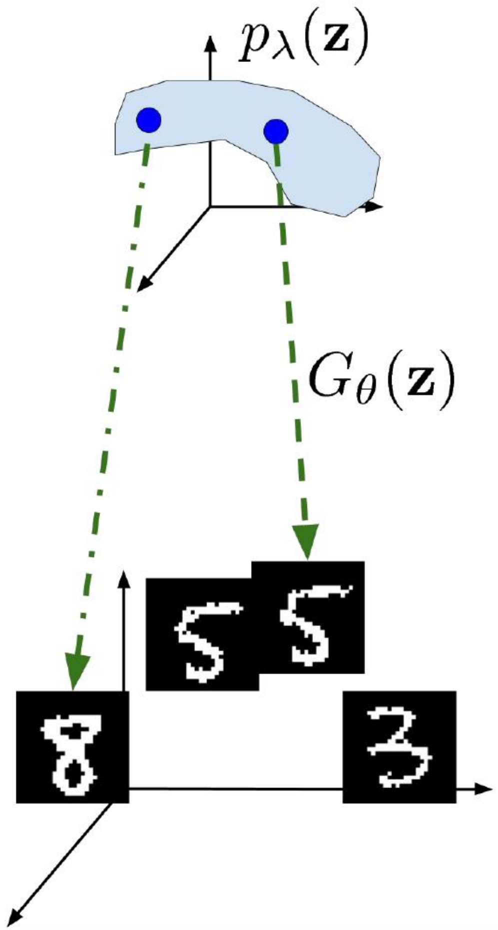
Assume a deterministic generator: $\mathbf{x} = G_\theta(\mathbf{z})$

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 GAN
Arjowski et al ‘17

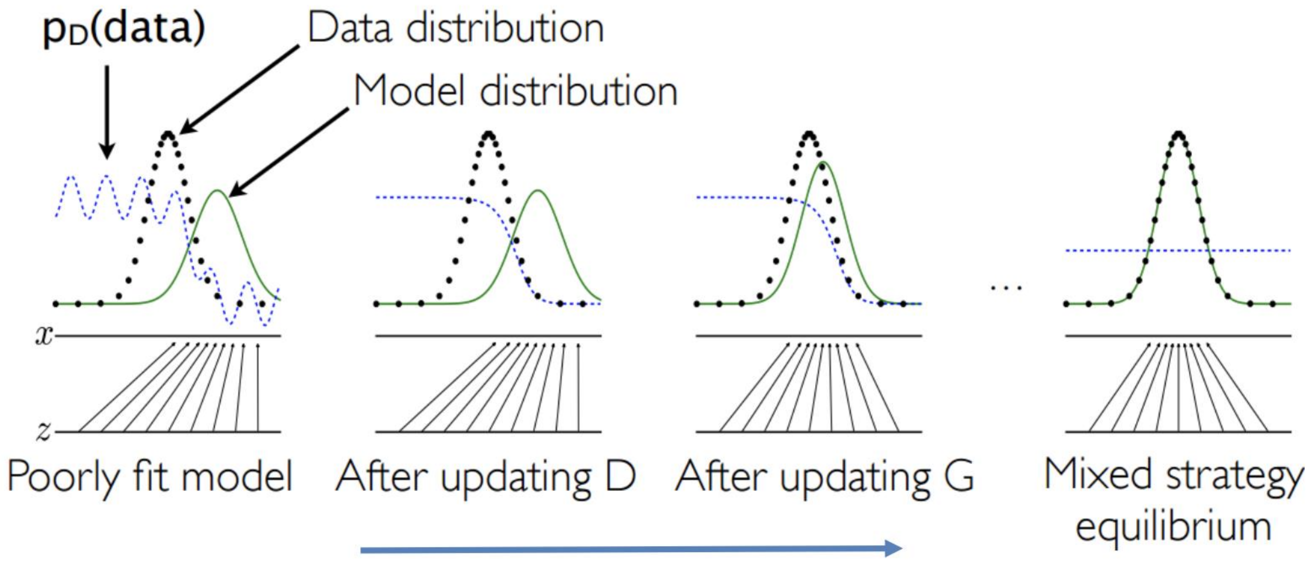
arXiv:1406.2661v1

Generative adversarial training (II)

Generator is trained to maximize the probability of Discriminator making a mistake

D gradient guide G to regions more likely to be classified as data

Initially D is not an accurate classifier



G and D don't improve anymore.
D is unable to differentiate

D is trained to discriminate samples from data

How well does it work?

2014:

How well does it work?

2018:

https://research.nvidia.com/sites/default/files/pubs/2017-10_Progressive-Growing-of/karras2018iclr-paper.pdf

How well does it work?

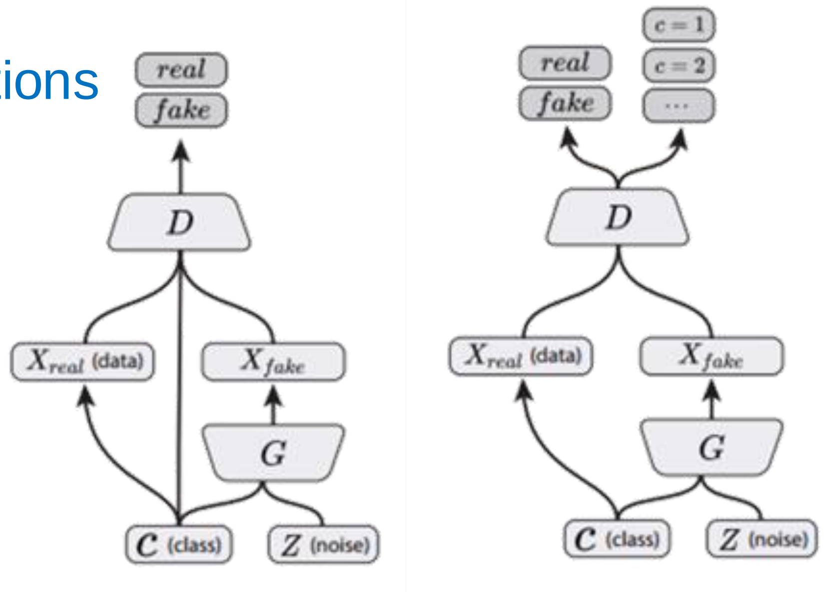
2018:

https://research.nvidia.com/sites/default/files/pubs/2017-10_Progressive-Growing-of/karras2018iclr-paper.pdf

GAN flavors

monarch butterfly goldfinch daisy

- Original GAN was based on MLP in 2014
- Deep Convolutional GAN in 2015
- 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 GAN
- Stack GAN
- BIGAN ...



Conditional GAN (Mirza & Osindero, 2014)

AC-GAN (Present Work)

arXiv: 1411.1784

arXiv:1610.0958

CERN openlab

Generalisation

Does the Generator fully learn the target distribution from small training set?

GANs produce distributions with limited support

Support size grows ~linearly with discriminator size (Zhang A., ICML'17)

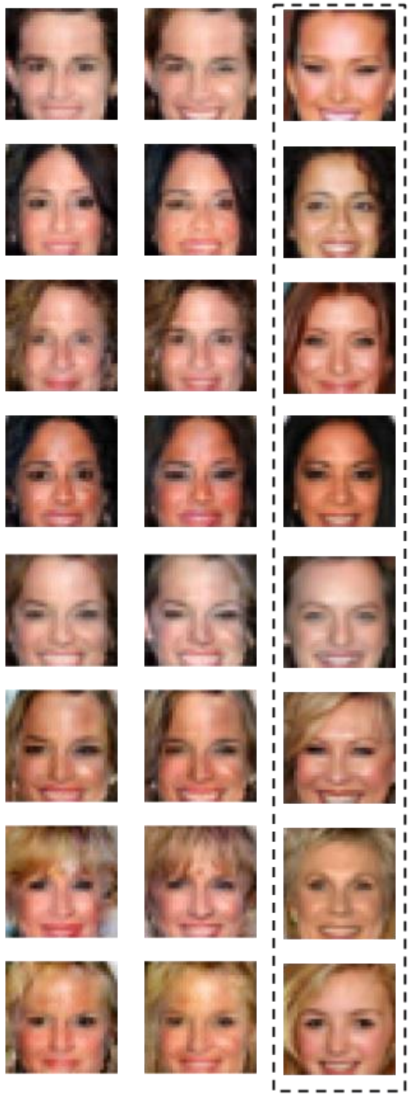
- Training dataset size does not help much for a given discriminator

BIGAN (on faces dataset)

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

Depending on the application, in practice, this might not be an issue

Search for nearest neighbor

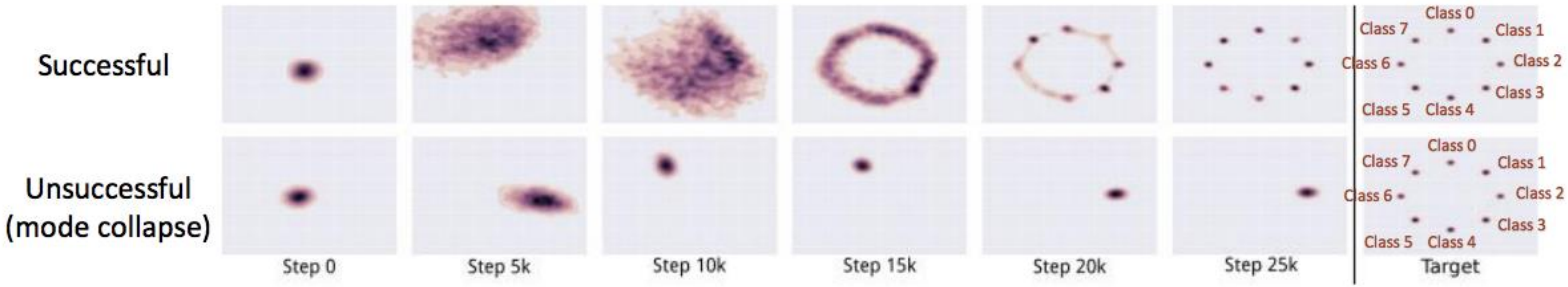


DCGAN

One extreme case: Mode collapse

Goal is to generate fake examples imitating real samples

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



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

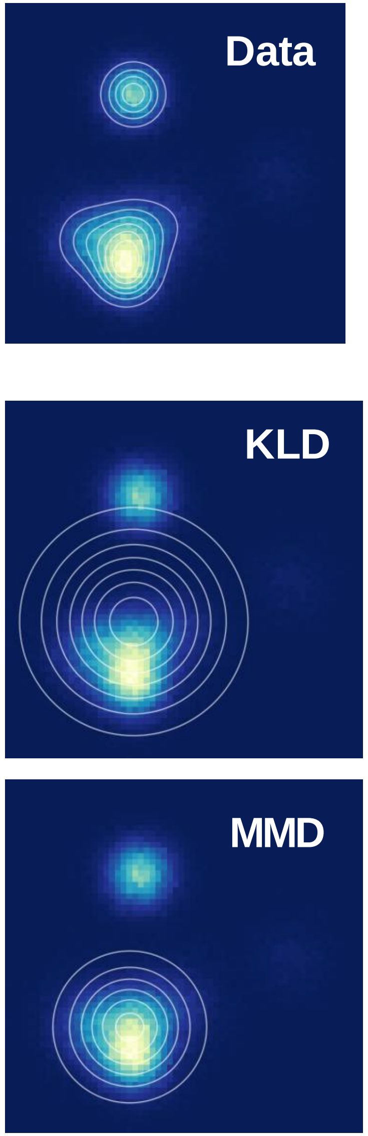
Performance evaluation

Check similarity between image distributions:

- Mixing and coverage (diversity)
- Saliency
- Mode collapse or mode dropping
- Overfitting (has the network memorized samples?)

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
  - Kullback-Leibler Divergence
  - Inception score
  - Maximum Mean Discrepancy
  - Fréchet Inception Distance



https://arxiv.org/pdf/1511.01844v2.pdf

Applications

Some HEP applications

LAGAN for Jet Images. (arxiv:1701.05927)

CaloGAN (arxiv:1705.02355)

GAN based LHCb Calorimeter simulation (CHEP2018)

Generative models for ALICE TPC simulation (CHEP2018)

Conditional Wasserstein GANs for fast simulation of electromagnetic showers in a CMS HGCAL prototype (IML WG 04/18)

Variational AutoEncoders to simulate ATLAS LAr calorimeter (PASC18)

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

Refining Detector Simulation using Adversarial Networks (IML WG 04/18)

arxiv:1701.05927

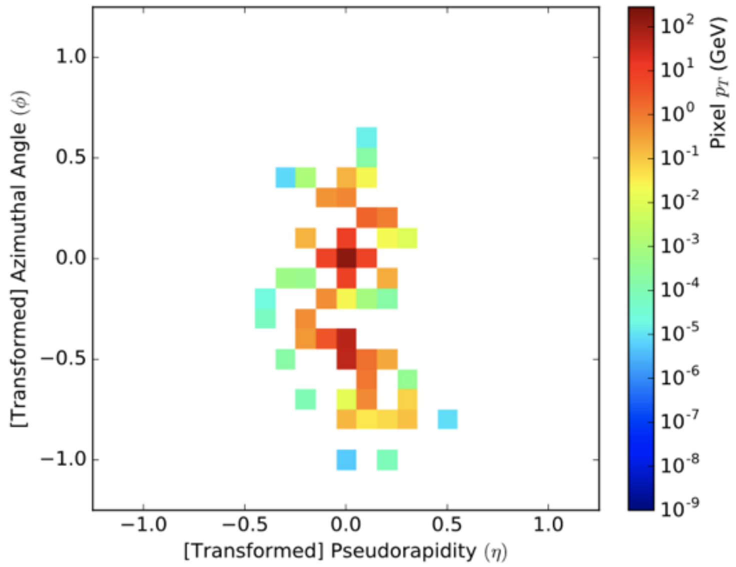
Location Aware GAN

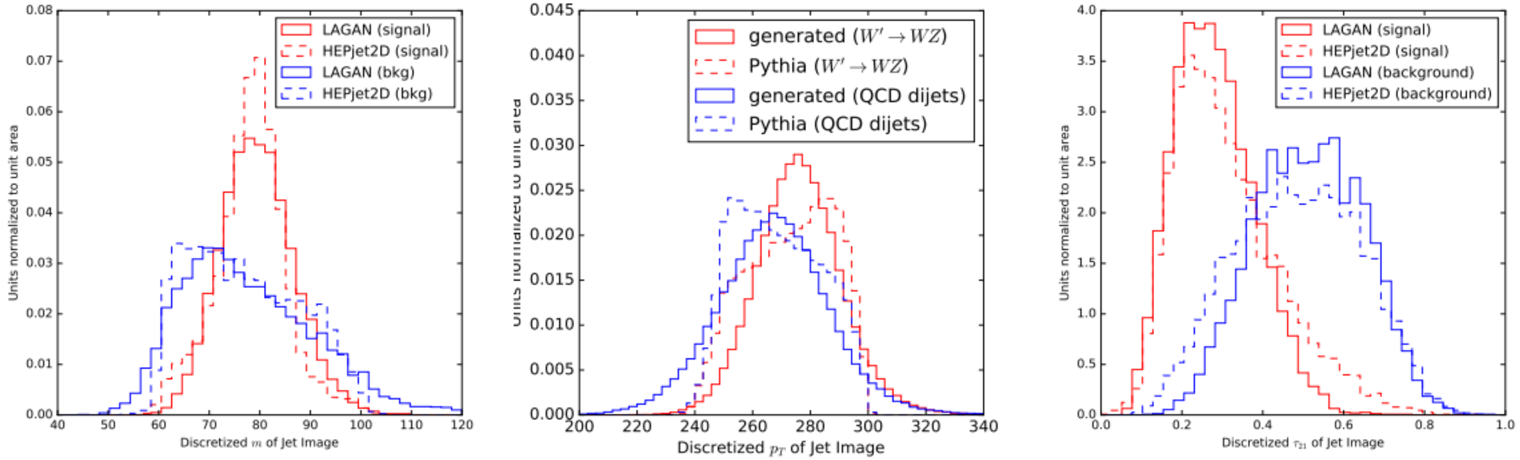
Reproduce 2D generator level anti-kT jet images

- Inspired by DCGAN (convolutions) and ACGAN (uses particle type information)

Image features:

- Sparse
- Location dependent features
- Large dynamic range

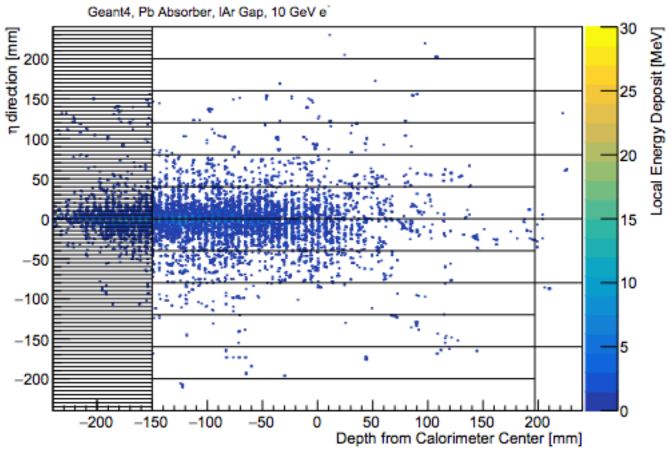


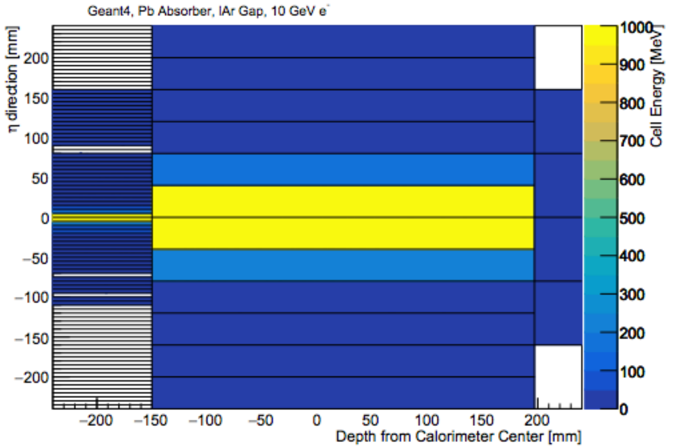


CERN openlab

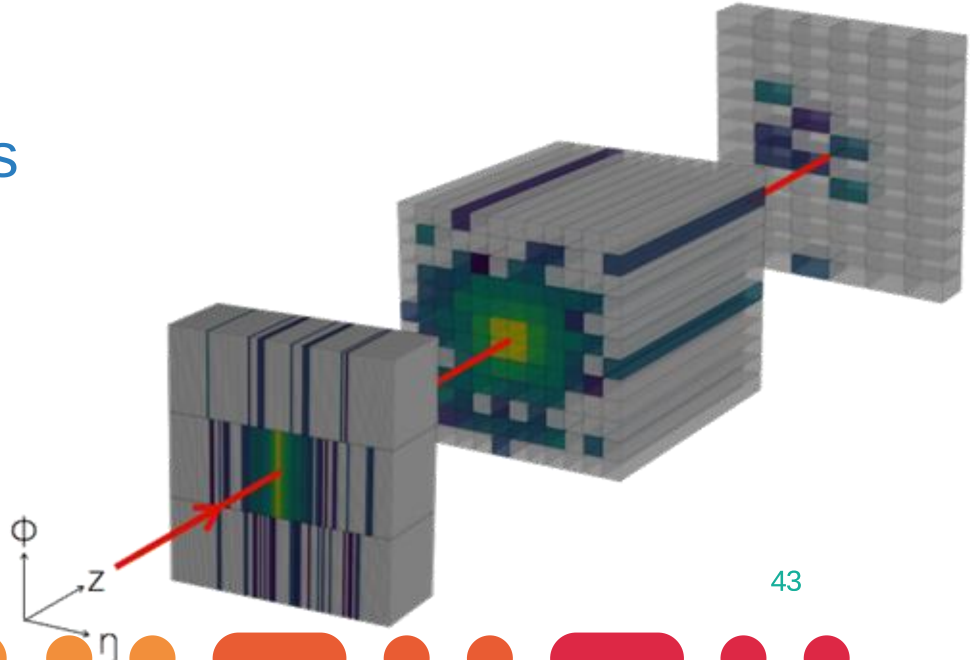
arxiv:1705.02355

CaloGAN



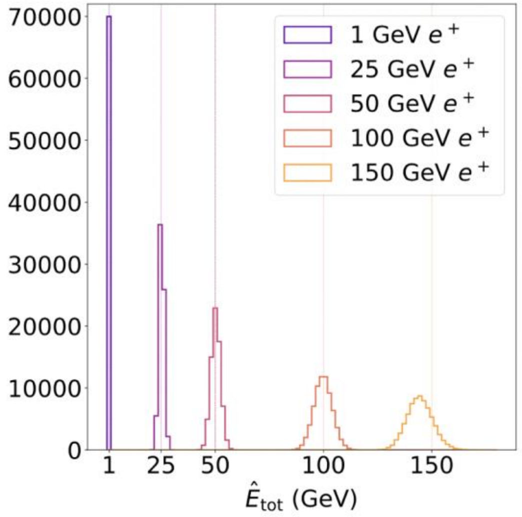


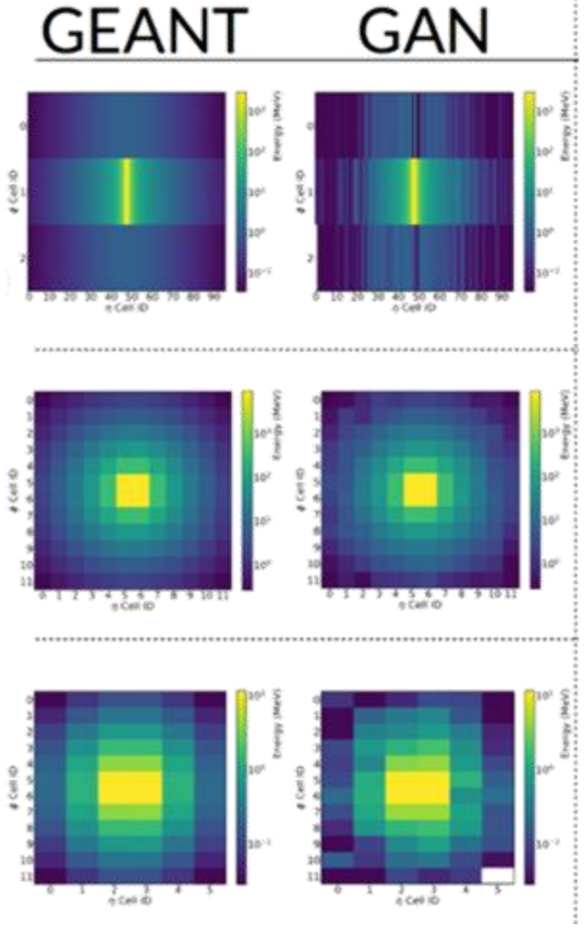
- ATLAS LAr calorimeter
  - Heterogeneous longitudinal segmentation into 3 layers
  - Irregular granularity in eta and phi
- Energy deposition in each layer as a 2D image
- Build one LAGAN per layer
- Trainable transfer unit to preserve layer correlations
- Result is a concatenation of 2D images that reproduce full 3D picture

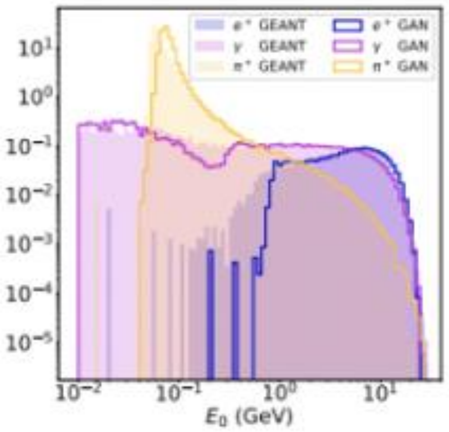


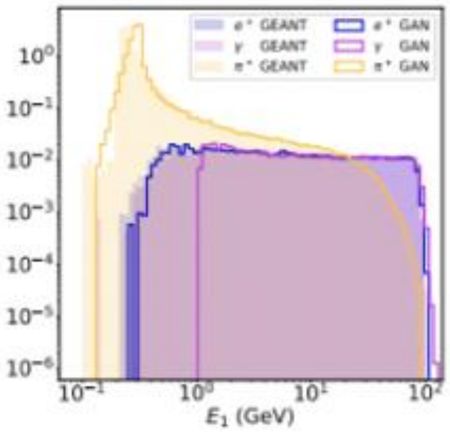
CaloGAN performance

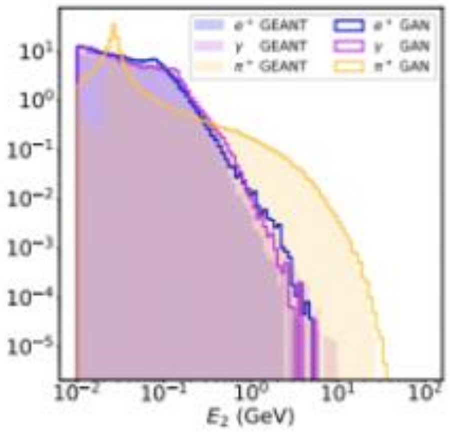
- Comparison to full simulation:
  - Average showers
  - Shape variables (depth, width, layer energy..) and event variables (sparsity level per layer)
- Energy reconstruction
  - First hints at “extrapolation” capabilities

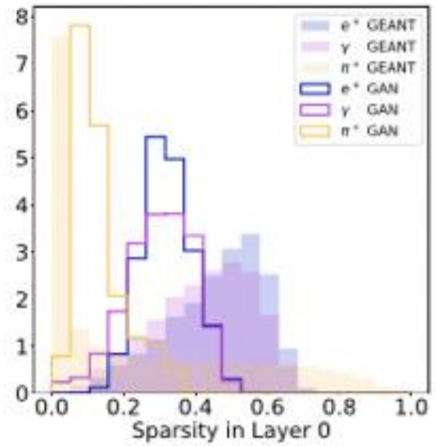






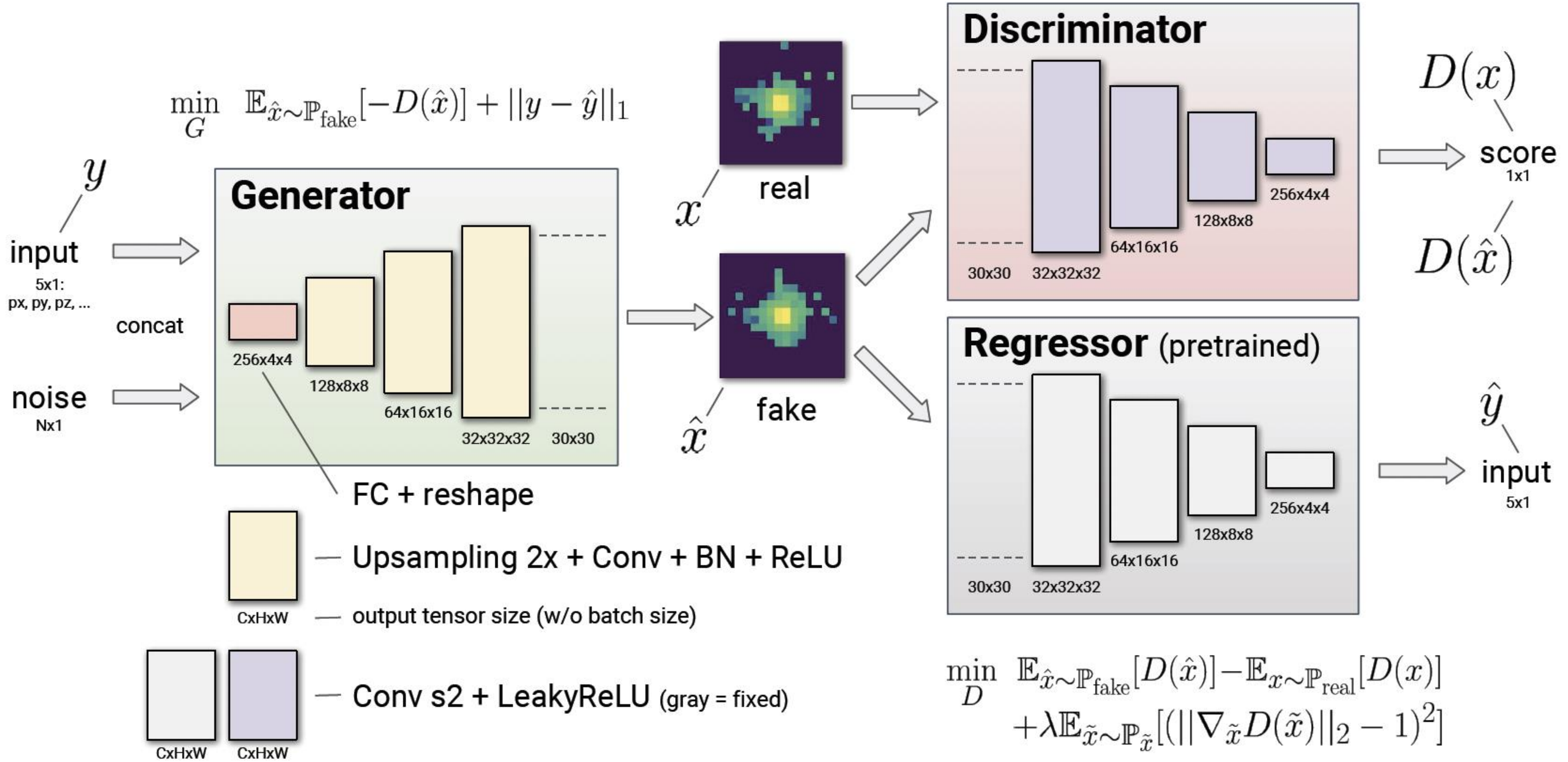






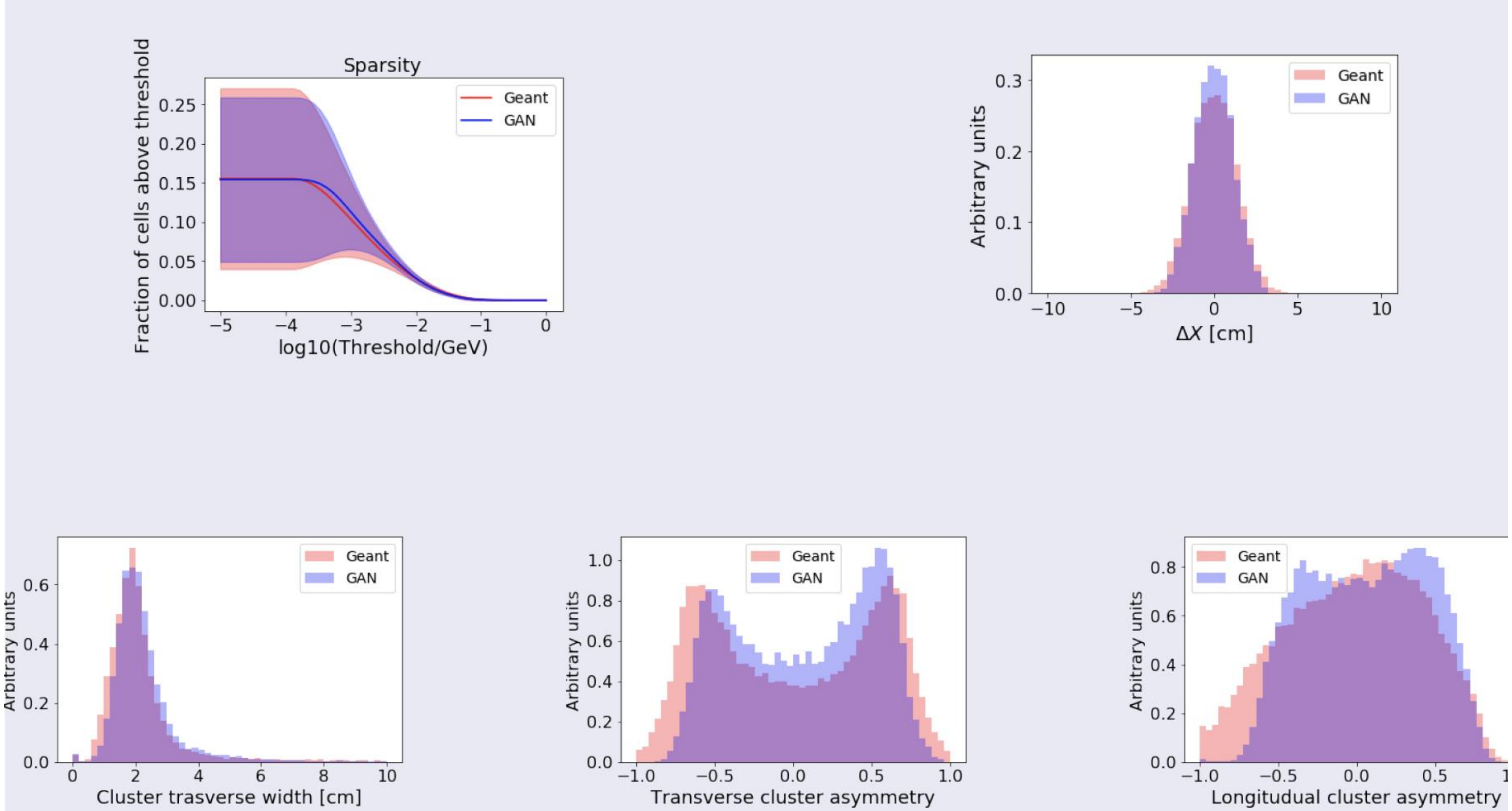
LHCb Calorimeter fast simulation

Wasserstein Convolutional GAN



Performance

Chekalina, CHEP2018



Refining Simulation using GANs

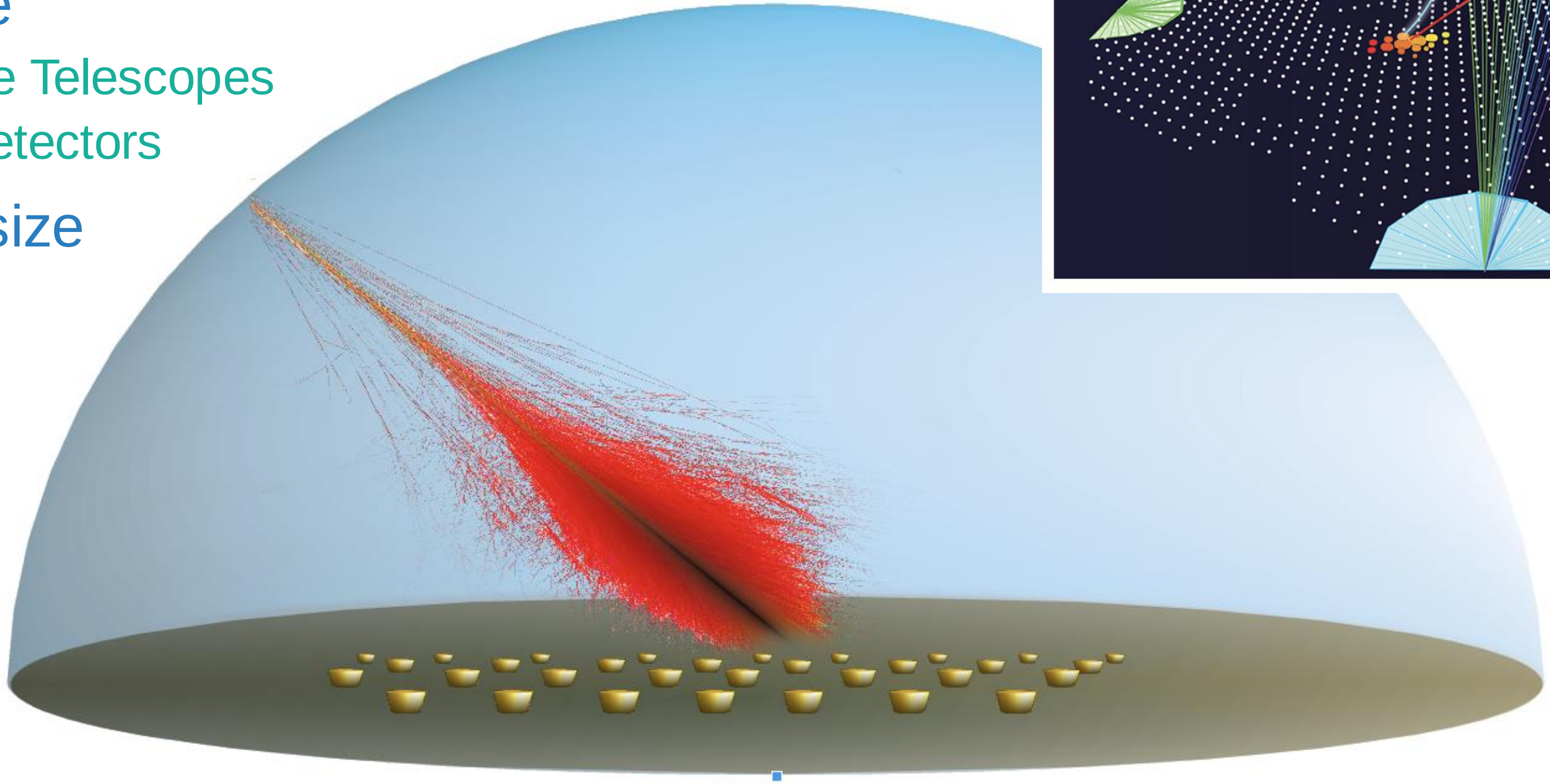
Pierre Auger Observatory

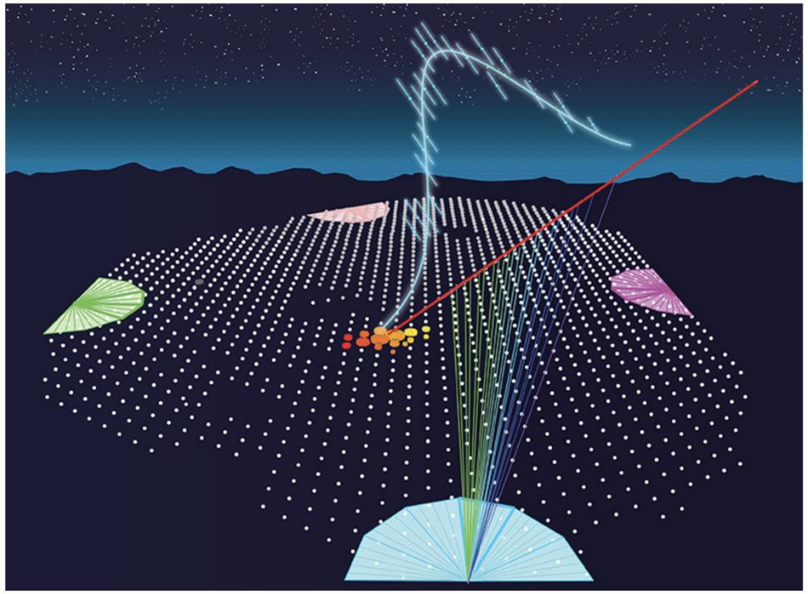
Detection of UHECR E>$10^{17.5}$ eV

Hybrid Technique

- 27 Fluorescence Telescopes
- 1660 Surface detectors

3000 km^2 array size

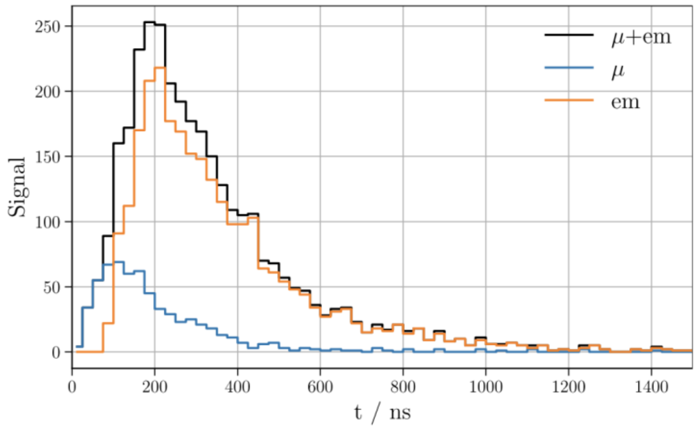


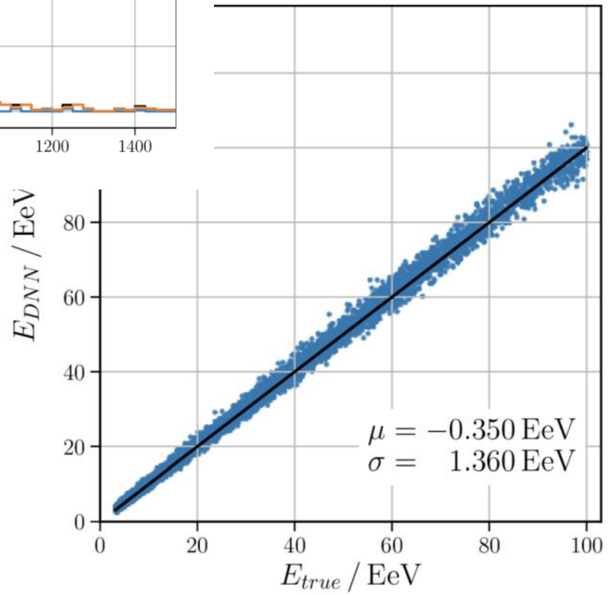


Refining Simulation using GANs

Energy reconstruction: Simulation

Showers: 70% electromagnetic 30% muonic

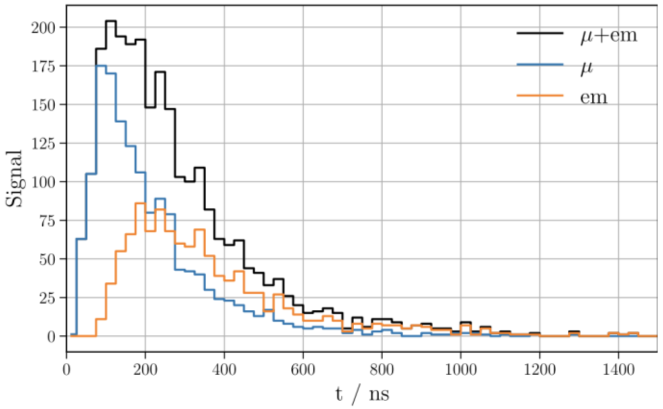


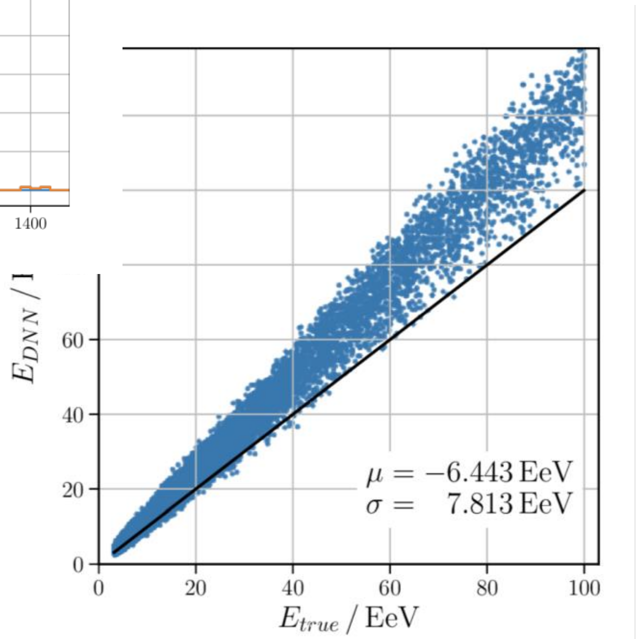


Energy reconstruction: Data

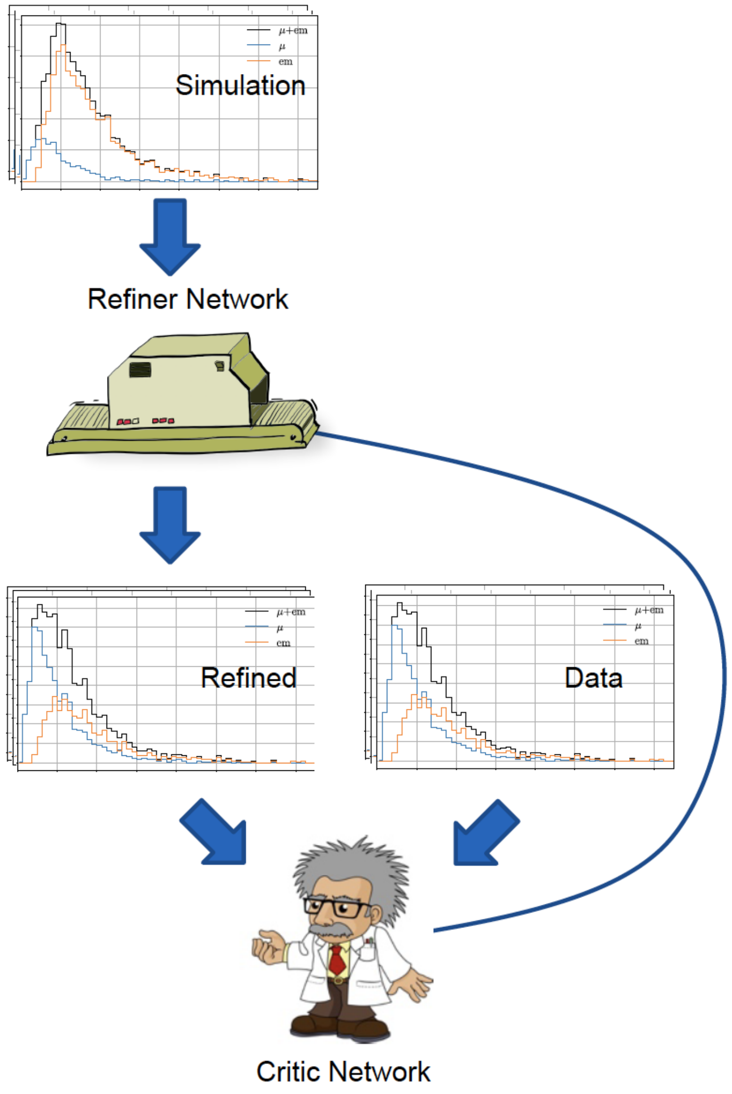
Showers: 30% electromagnetic 70% muonic

+ Increased noise





Refining Simulation using GANs



Refiner: tries to refine the simulation to look like data

Critic: measure similarity between data / simulation

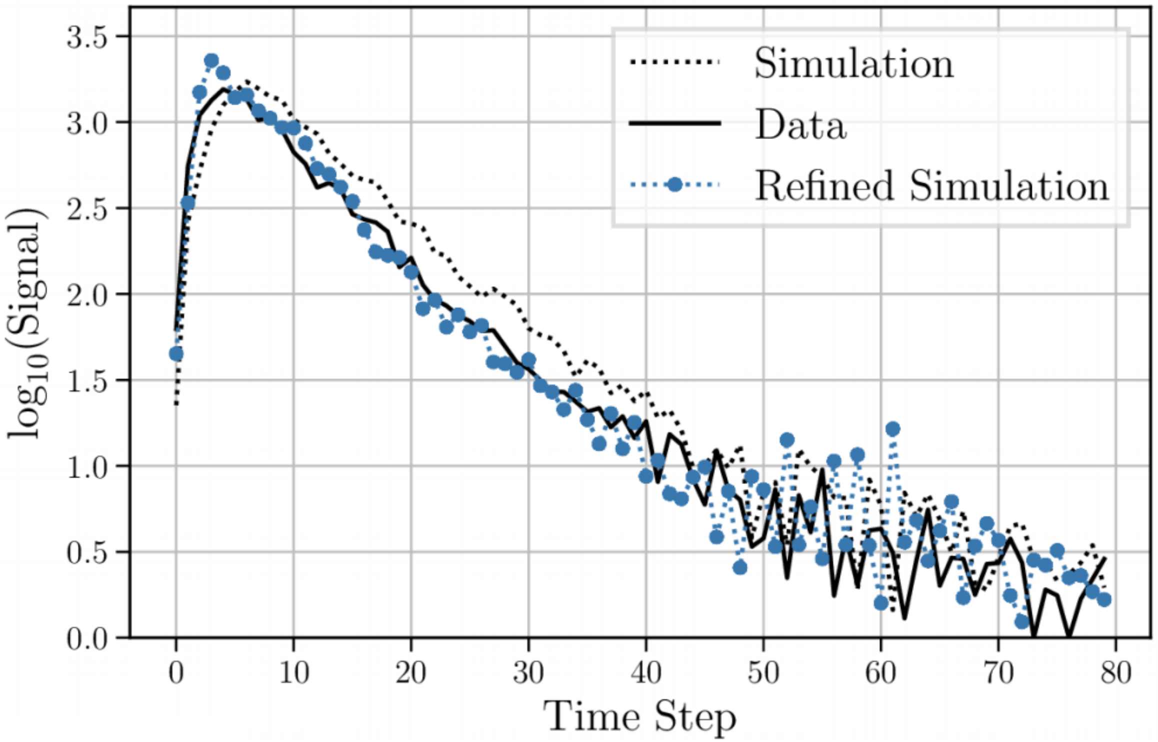
- Feedback of critic improves refiner performance

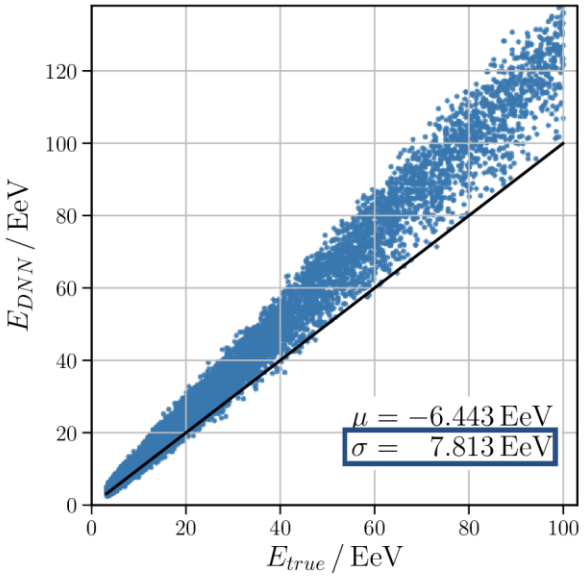
Promising results to make DNN robust to data applications

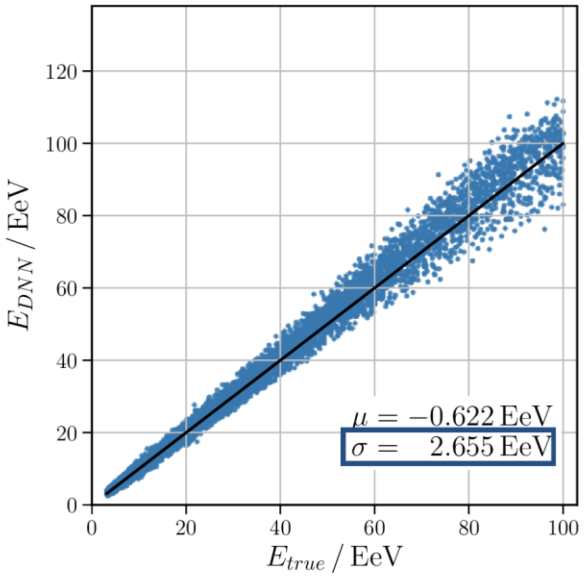
- Alternative application for continuous simulation scale factors

arXiv:1802.03325

Refining Simulation using GANs



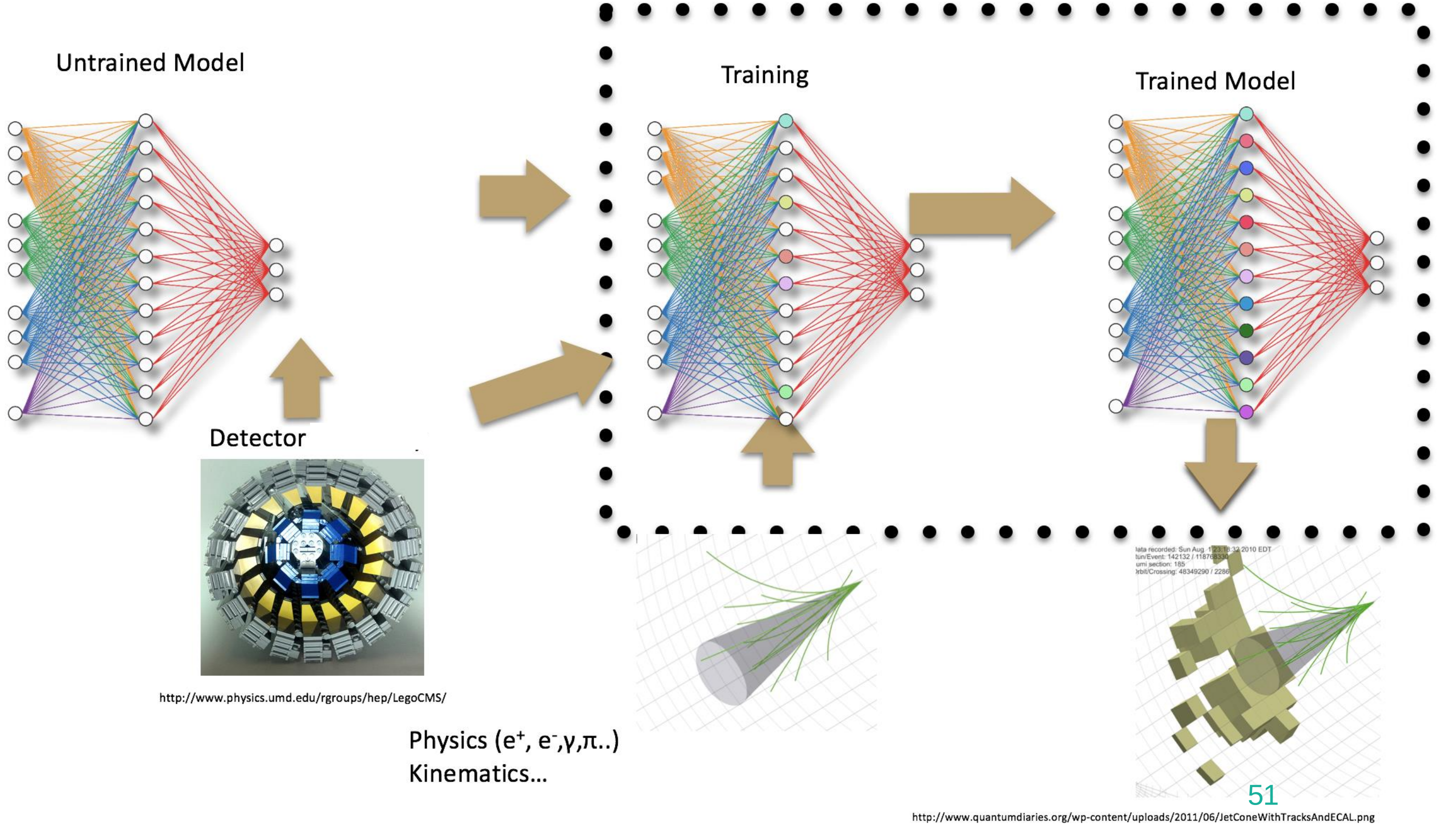




A DL engine for fast simulation

Design a tool that can be configured and trained for different detectors

- Start with time consuming detectors
  - Next generation highly granular calorimeters
- Train on detailed simulation
  - Test training on real data
- Test different models
  - CNN, RNN, …



CERN openlab

A plan in two steps

Is generative model output accurate enough?

Can we sustain the increase in detector complexity?

- A first proof of concept
- Understand performance and validate accuracy

How generic is this approach?

What portion of the original distribution do networks learn?

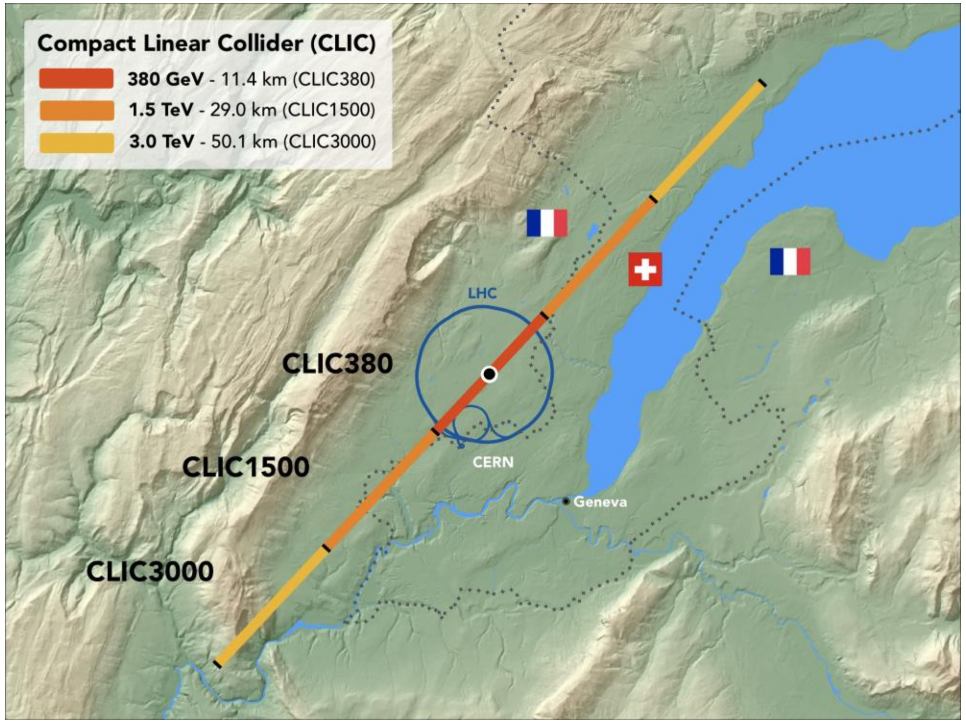
Can we “adjust” architecture to fit a larger class of detectors?

What resources are needed?

- Measure “coverage”
- Prove generalisation is possible
- Understand and optimise computing resources

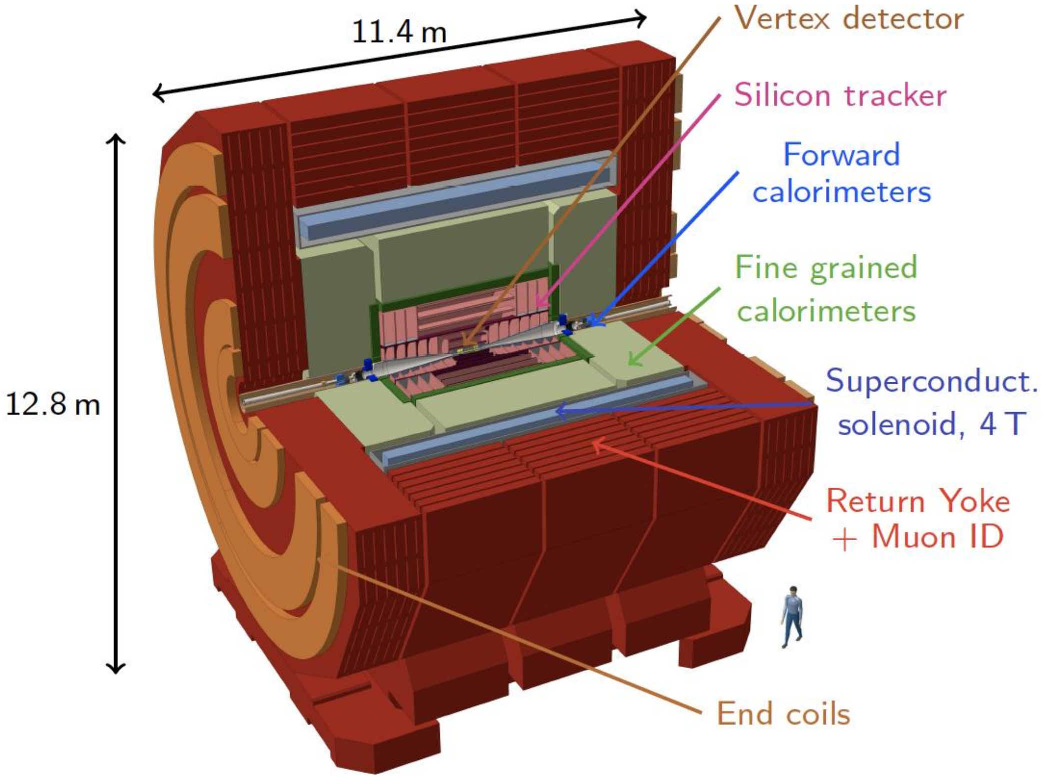
Proof of concept, benchmarking and validation

Compact LInear Collider



High-luminosity linear e+e- collider

Three energy stages up to 3 TeV



Electromagnetic calorimeter detector design

1.5 m inner radius

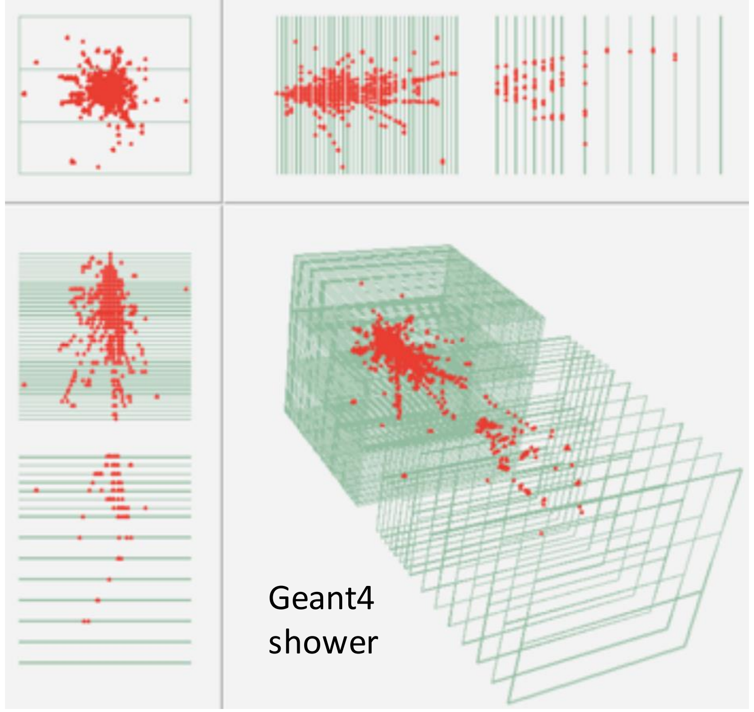
5 mm×5 mm segmentation

25 tungsten absorber layers + silicon sensors

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

CLIC calorimeter simulation

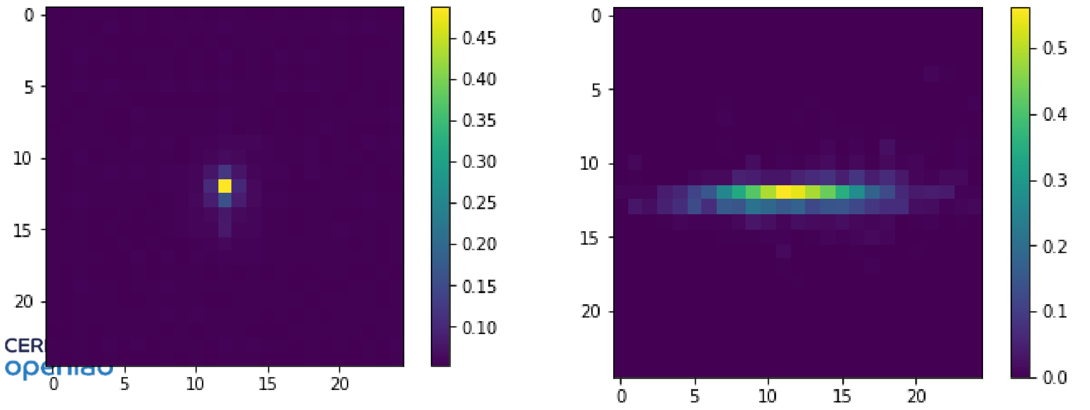
Data is essentially a 3D image

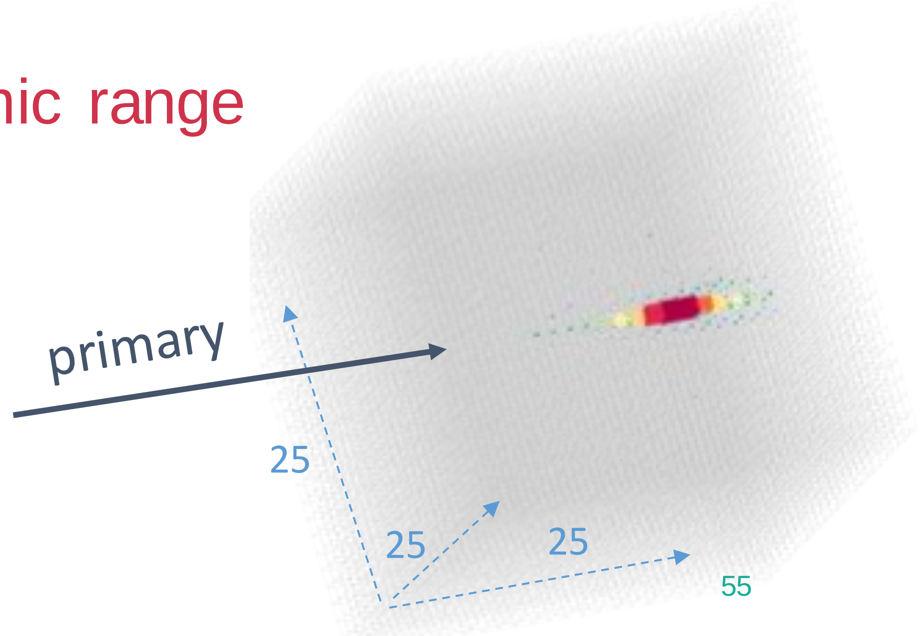


1M single particle samples (e,γ,π)

- Flat energy spectrum (10-500) GeV
- Orthogonal to detector surface
- +/- 30° random incident angle

Images are highly segmented and sparse, large dynamic range





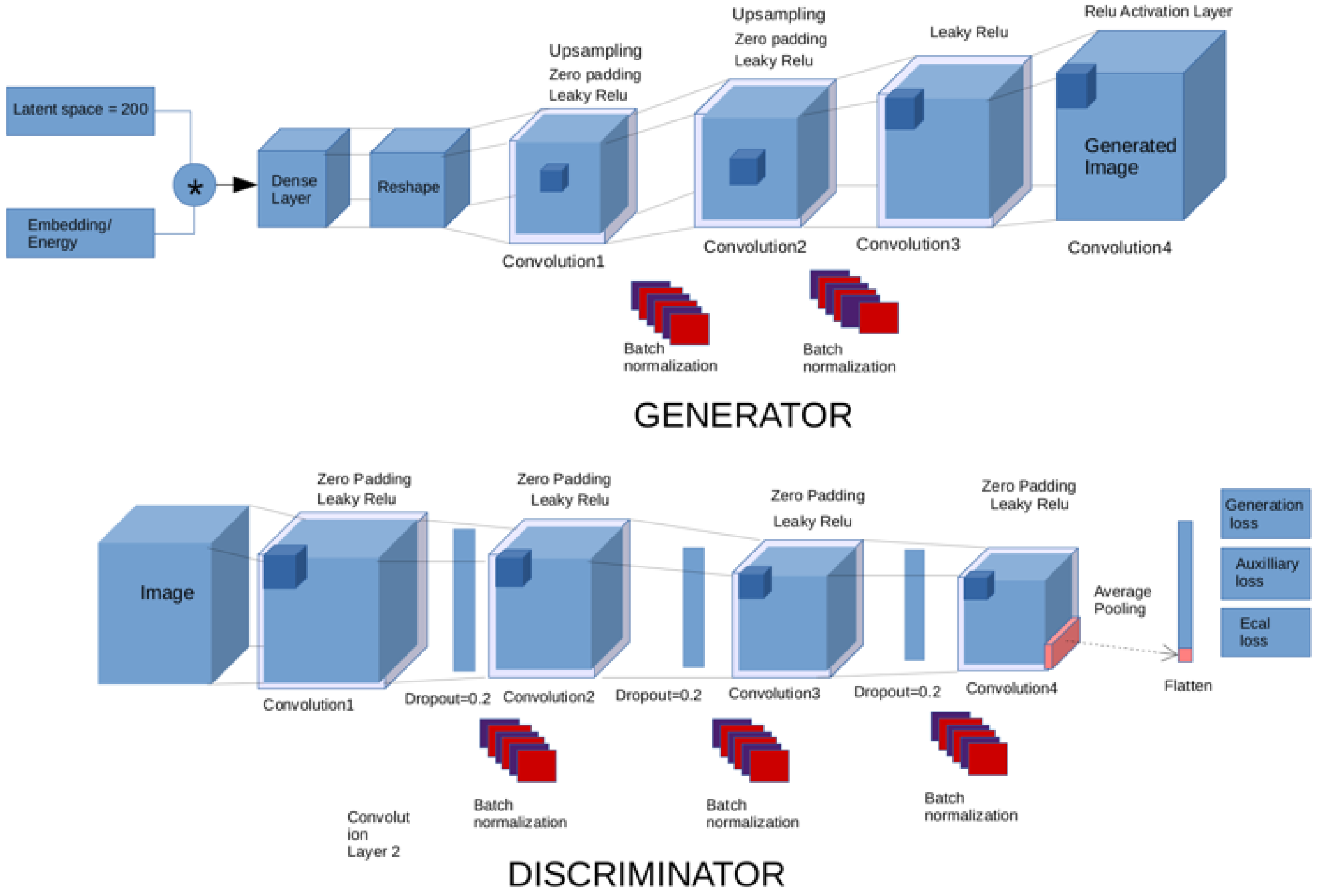
The model: 3D convolutional GAN

Similar discriminator and generator models

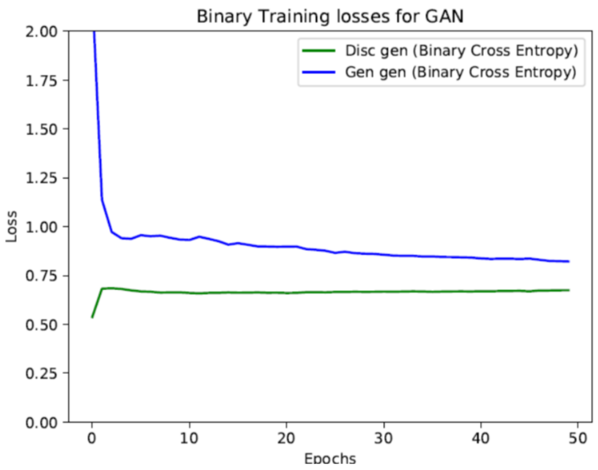
- 3d convolutions (keep X,Y symmetry)
- Upsampling layers
- Batch normalisation

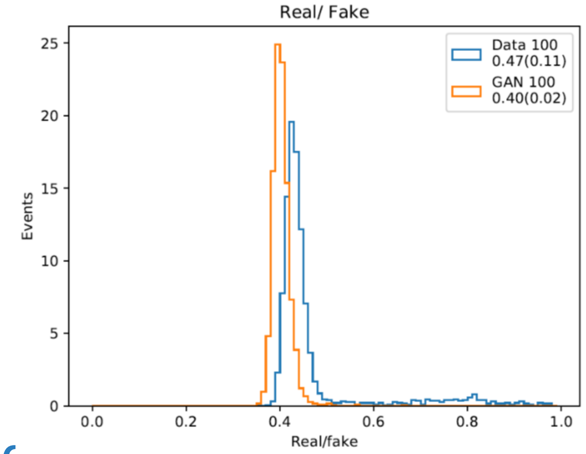
Condition training on input particle

Auxiliary regression tasks assigned to the discriminator improve convergence



Validation and optimisation





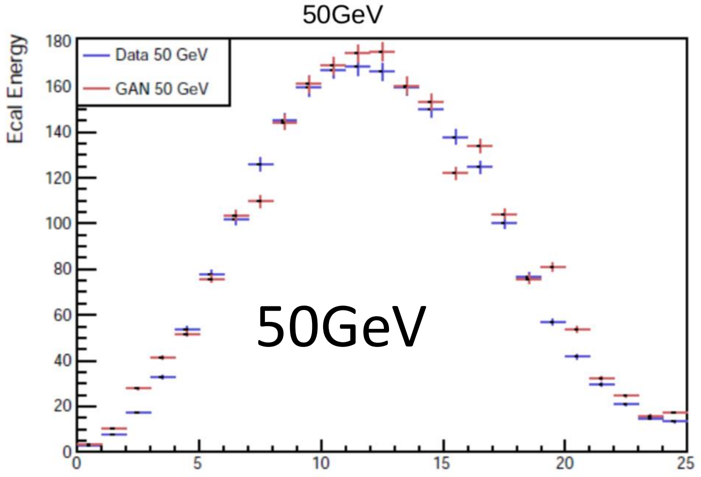
- Detailed GAN vs GEANT4 comparison (More than 200 Plots!)
  - High level quantities (shower shapes)
  - Calorimeter response (single cell response)
  - Particle properties (primary particle energy)
- Optimisation on
  - Network Architecture (Layers, filters, kernels, initialisation)
  - Losses definition
  - Data pre-processing
- Rely on GAN losses only !! No physics variable explicitly constrained.
- Results agree within a few % to Geant4 (labelled “DATA” in next slides ☺)
- We are running reconstruction code on G4 and GAN samples

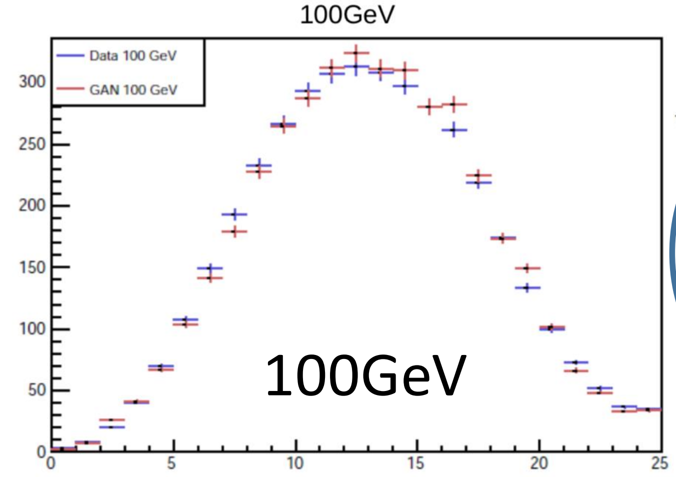
CERN openlab

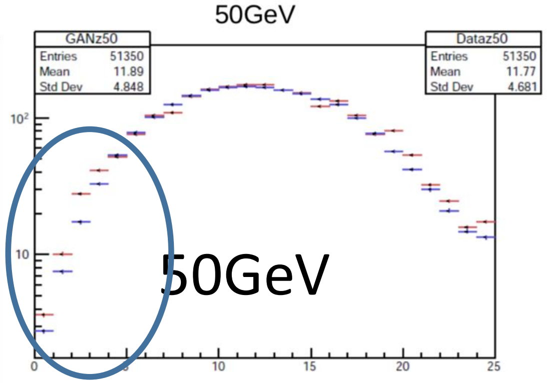
We run on Caltech ibanks GPU cluster thanks to Prof M. Spiropulu

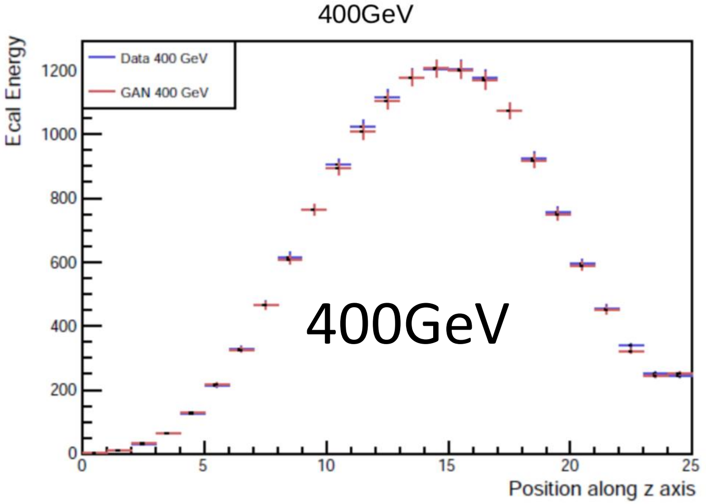
Electrons shower shapes

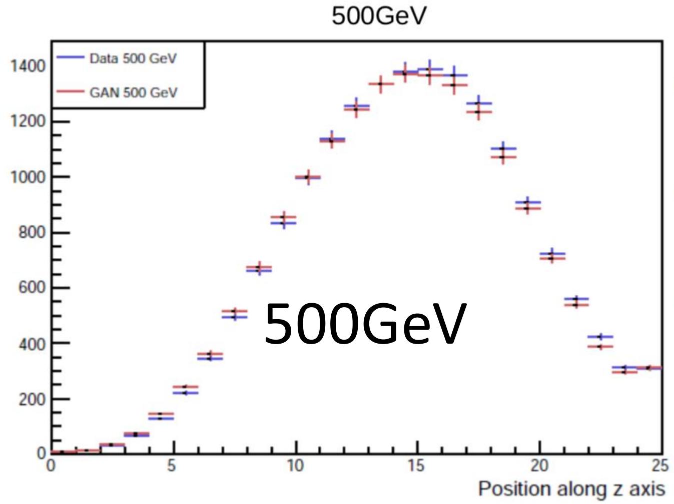
Orthogonal trajectory

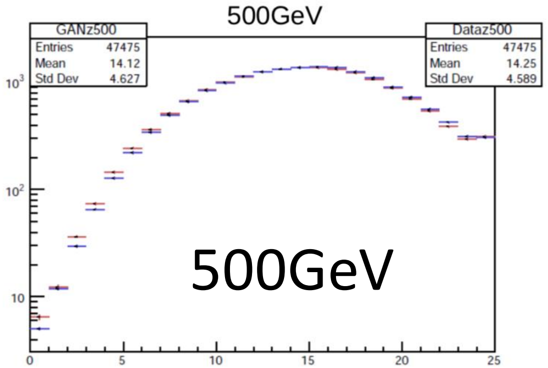












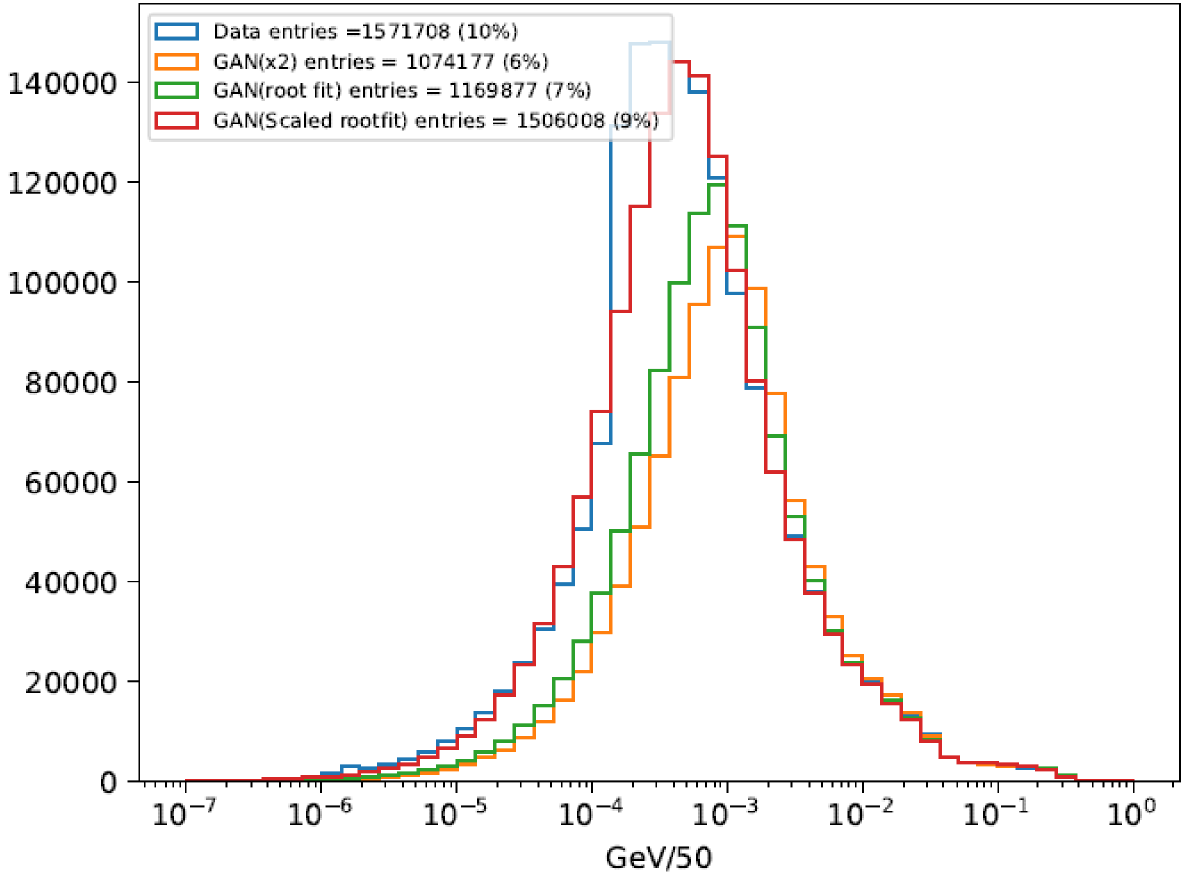
Single cell energy

Pixel dynamic range

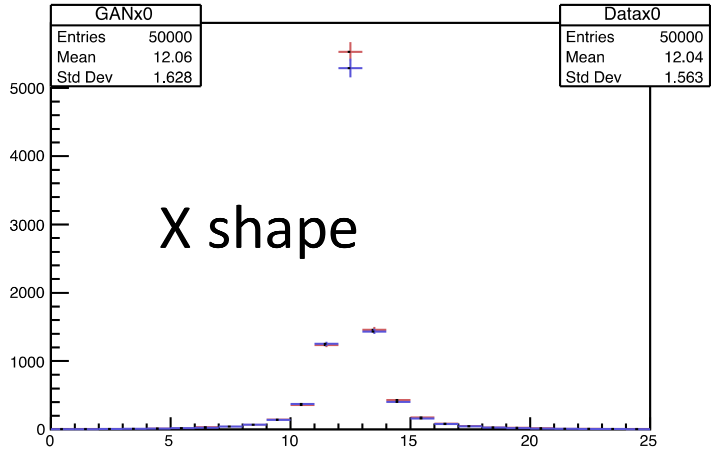
Single cell energy represents greyscale pixel intensity in the "image interpretation"

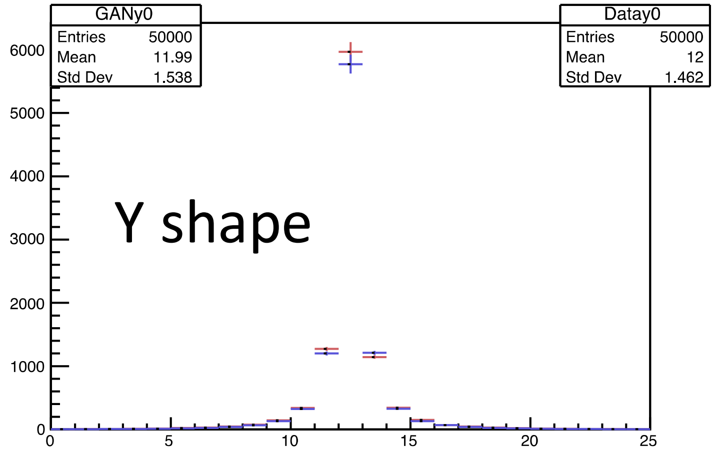
Very large range

Pre-processing changes performance



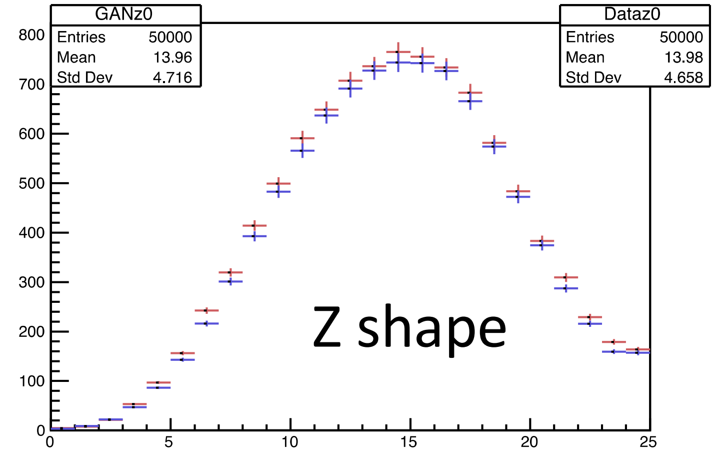
Neutral Pions



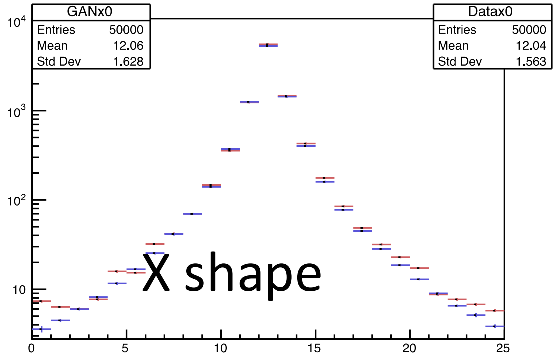


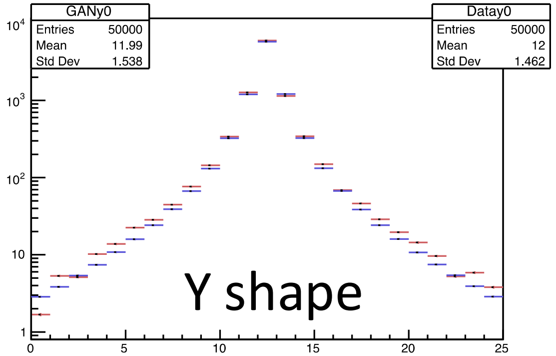
10-500 GeV

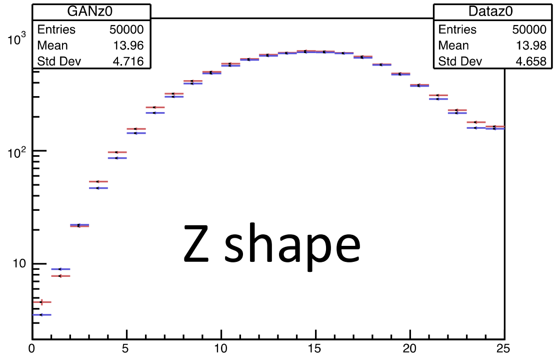
Orthogonal trajectory



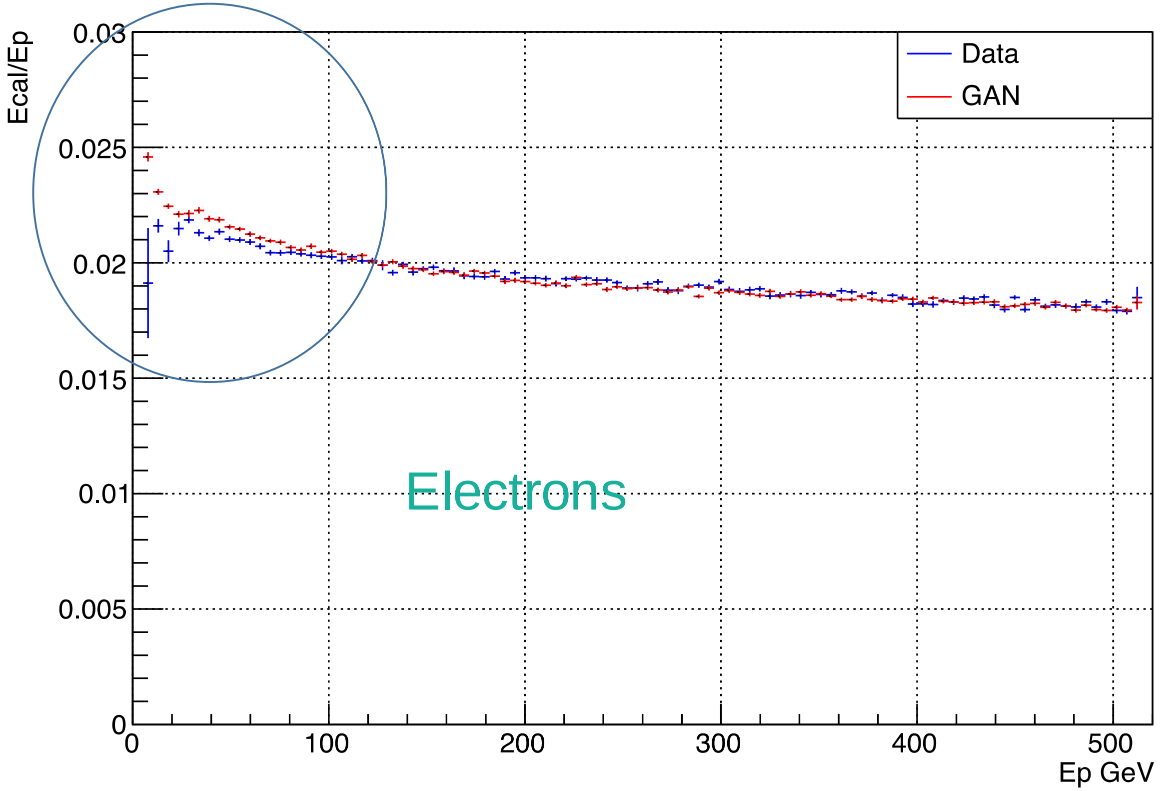
Log scale

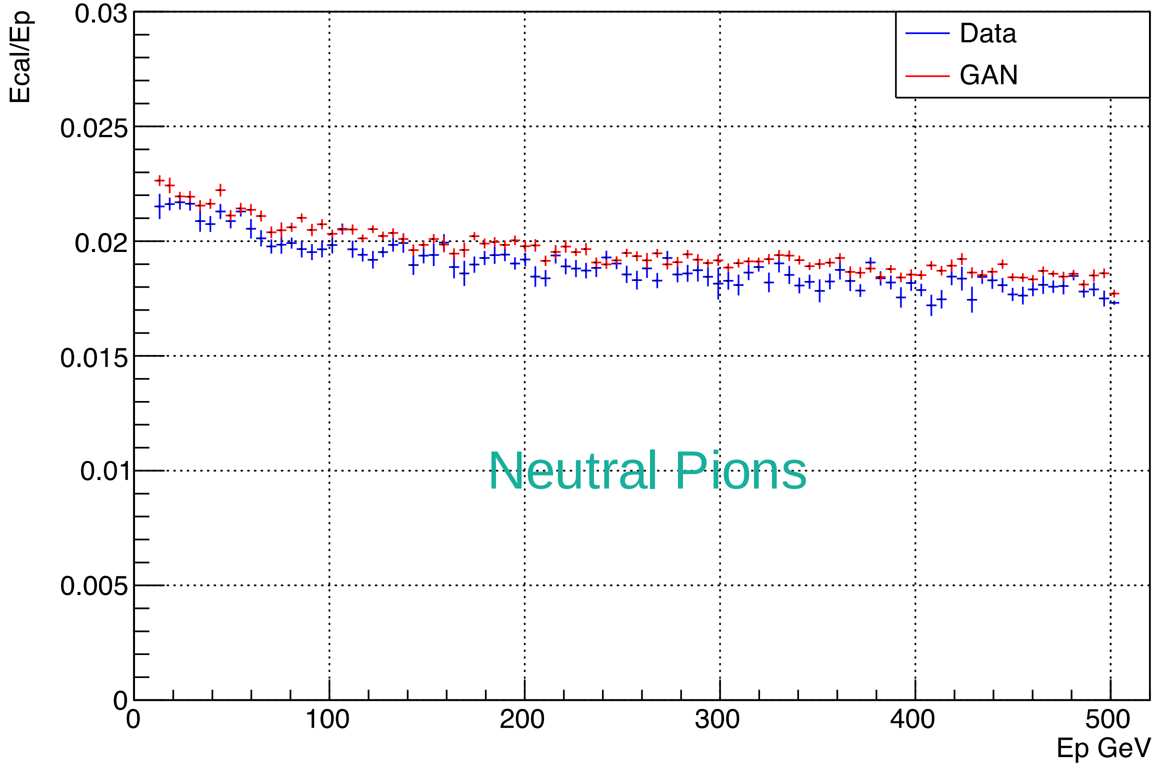






Calorimeter sampling fraction



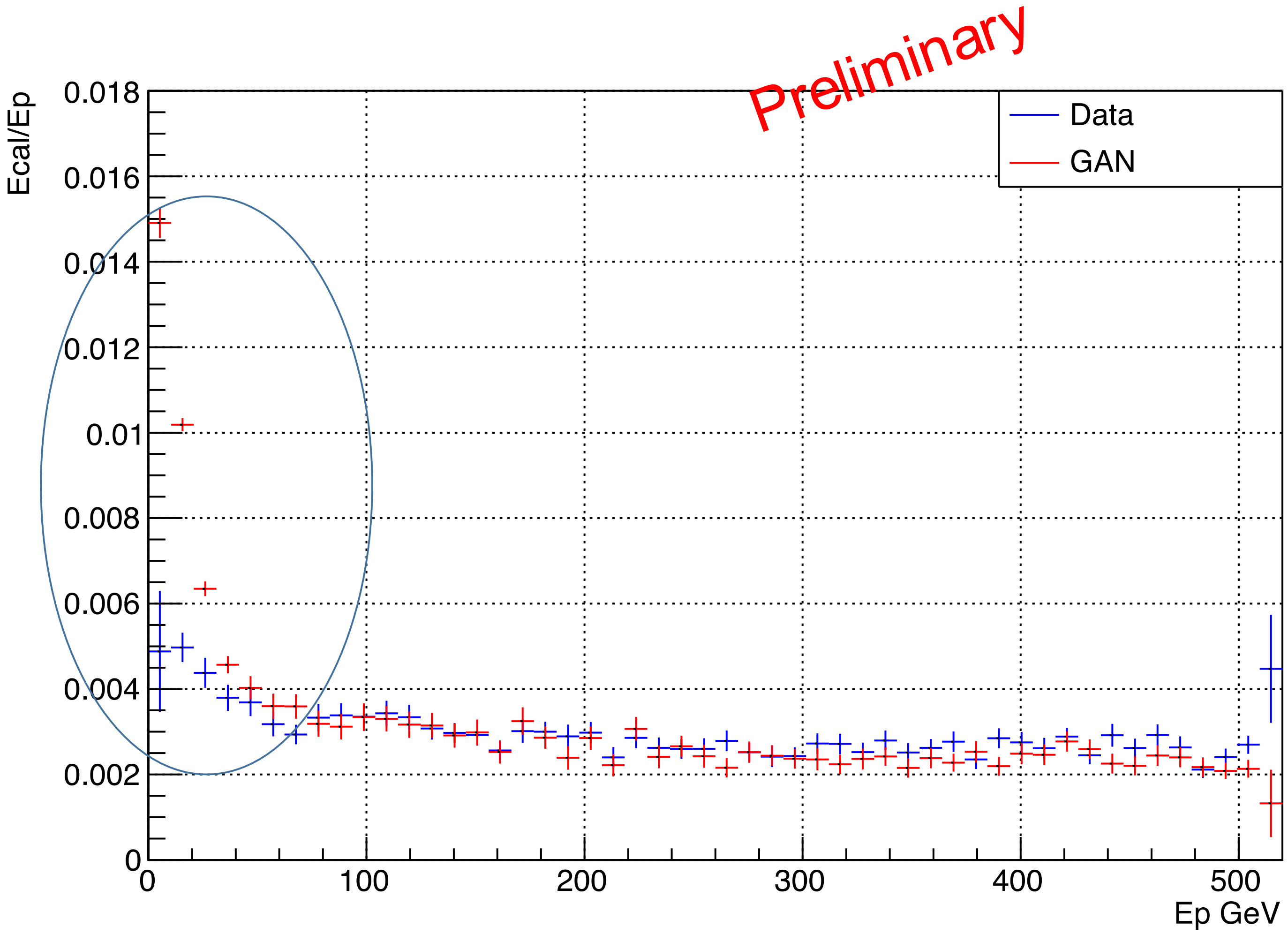


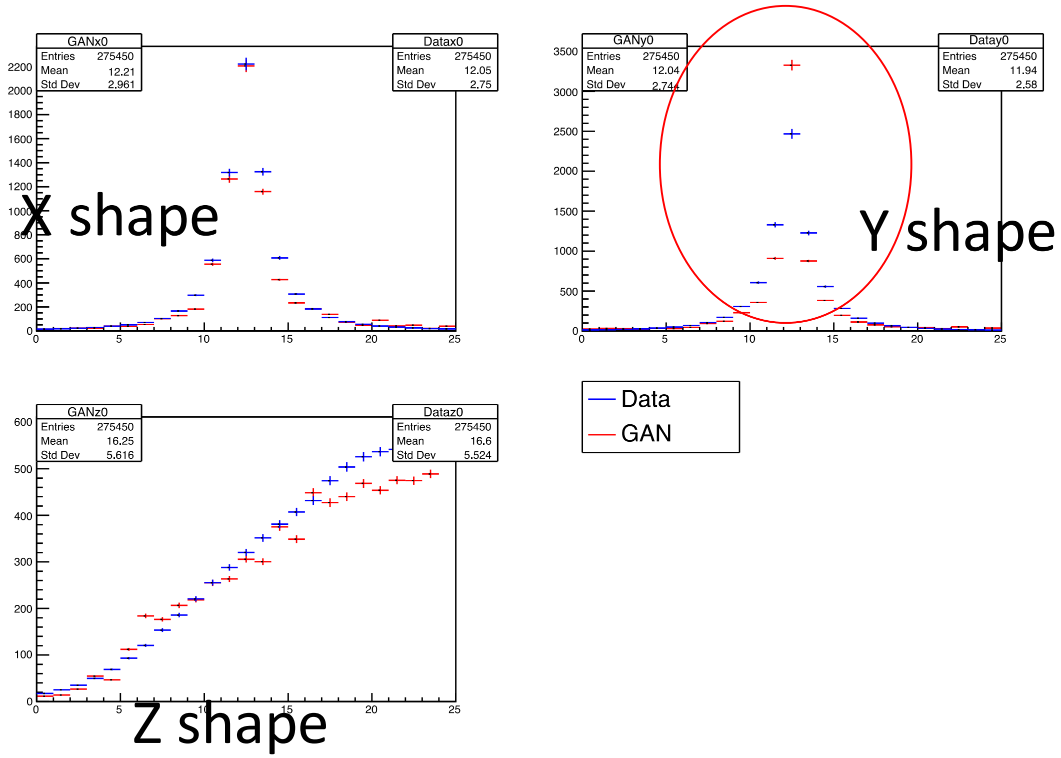
GAN seems to slightly overestimate slightly neutral pions energy deposits

Charged Pions

Charged pions have small energy deposits

Energy showers are delayed along Z

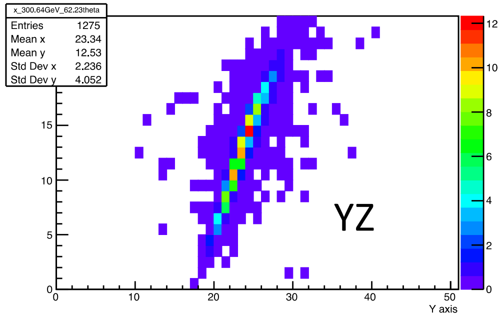


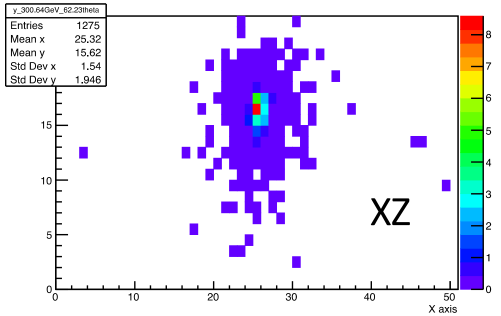


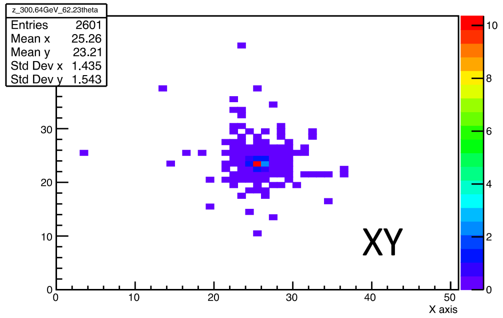
CERN openlab

Variable incident angle

Electrons enter the calorimeter with a 60°-120° angle range





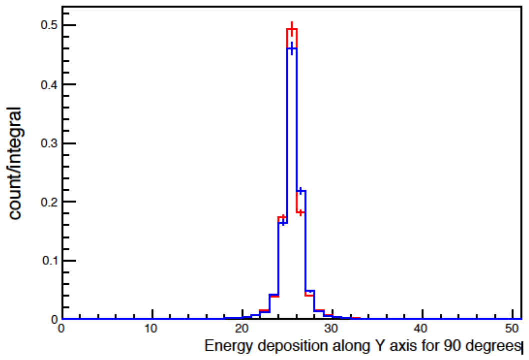


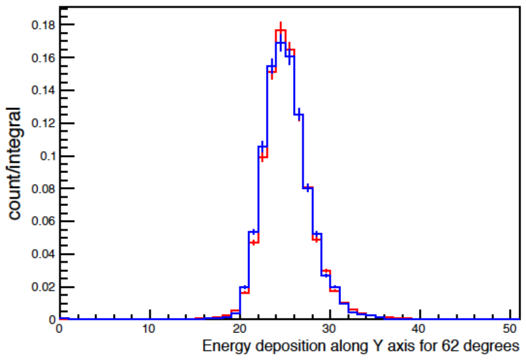
Preliminary

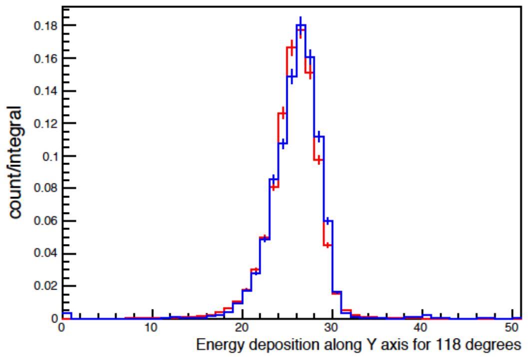
Adjust convolution parameters to improve energy description vs angle

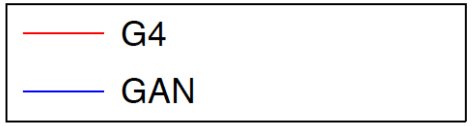
Minimal architecture changes

Wider/asymmetric image size (51x51x25):







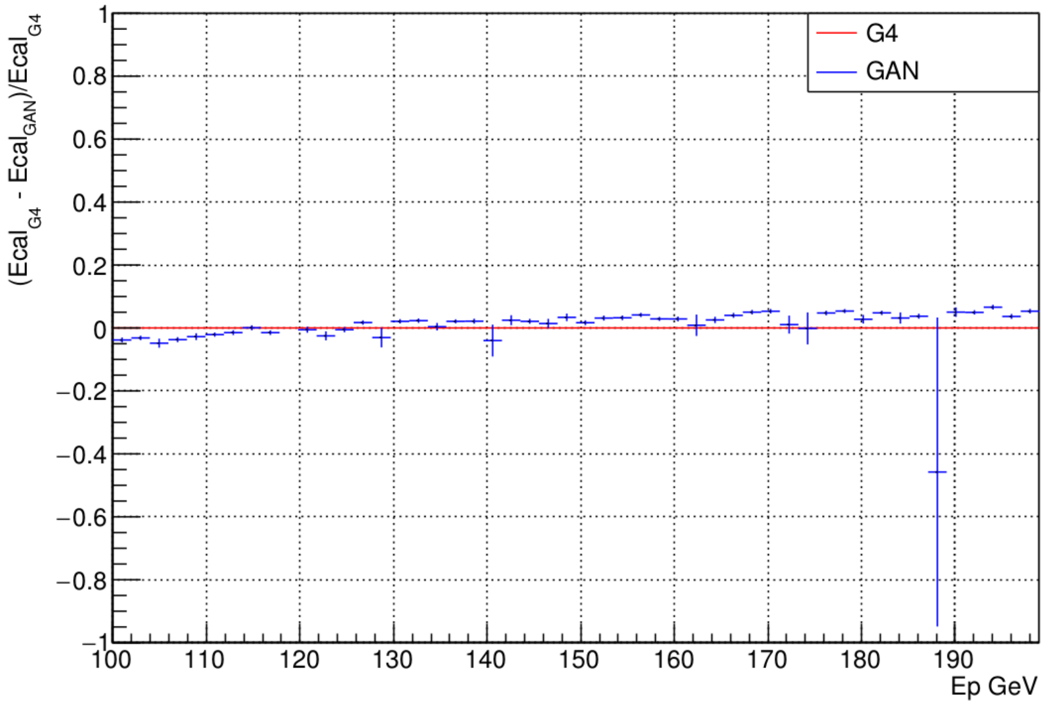


Y shower shapes for different angles

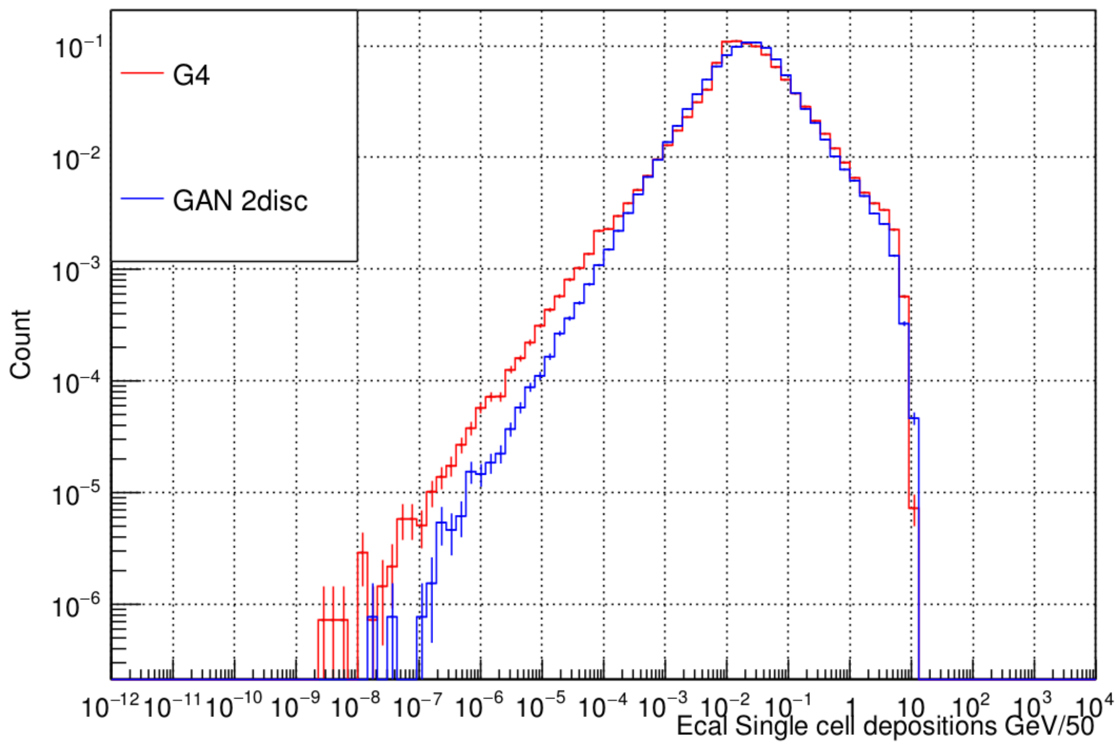
CERN openlab

Variable incident angle (II)

Total deposited energy (relative error)



Single cell energy



Computing resources

Distributed training

Computing resources: Fast!

Using a trained model is very fast

Single node performance.

Inference:

On Intel Xeon speedup factor is >2000

On NVIDIA P100 → speedup > $4 \cdot 10^5$

Training:

45 min/epoch on NVIDIA P100

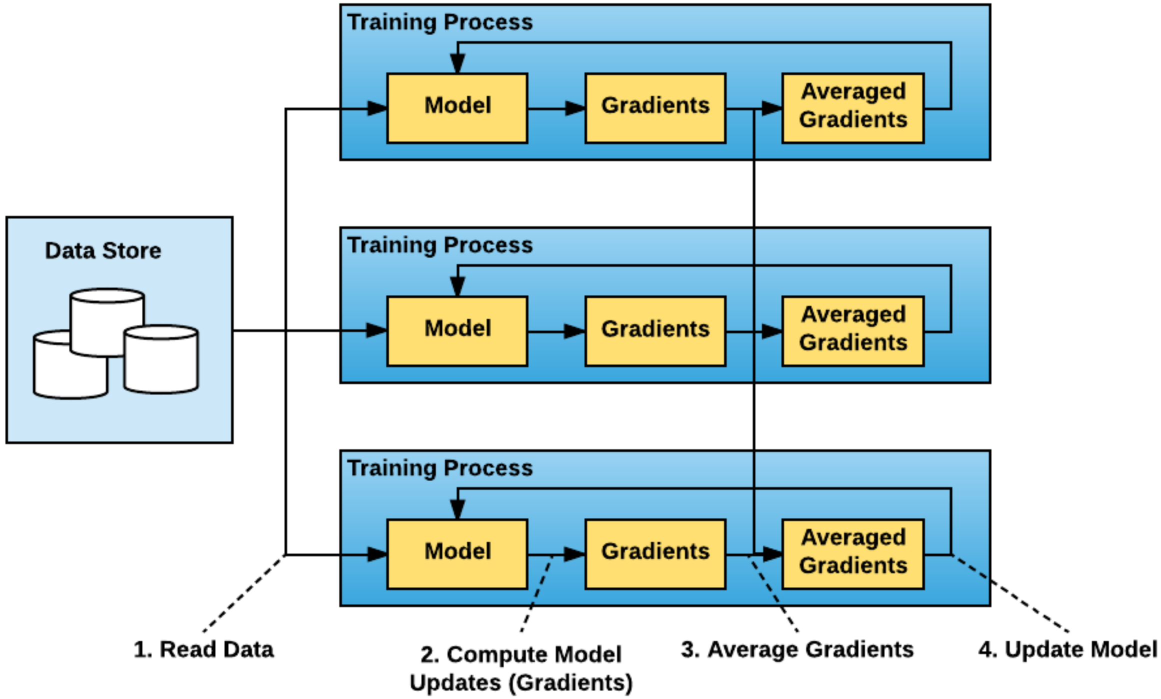
200K Geant4 events are needed for training

Time to create an electron shower		
Method	Machine	Time/Shower (msec)
MC Simulation (geant4)	Intel Xeon Platinum 8180	17000
3D GAN (batch size 128)	**Intel Xeon Platinum 8180**	**7**
3D GAN (batch size 128)	**NVIDIA P100**	**0.04**

CERN openlab

Distributed Training

Data Model:



- Data distribution
  - Compute gradients on several batches independently
  - Update the model synchronously or async
  - Applicable to large dataset
- Gradient distribution
  - Compute the gradient of one batch in parallel
  - Update the model with the aggregated gradient.
  - Applicable to large sample (large events)
- Model distribution
  - Compute the gradient and updates of part of the model separately
  - Applicable to large model

Data distribution

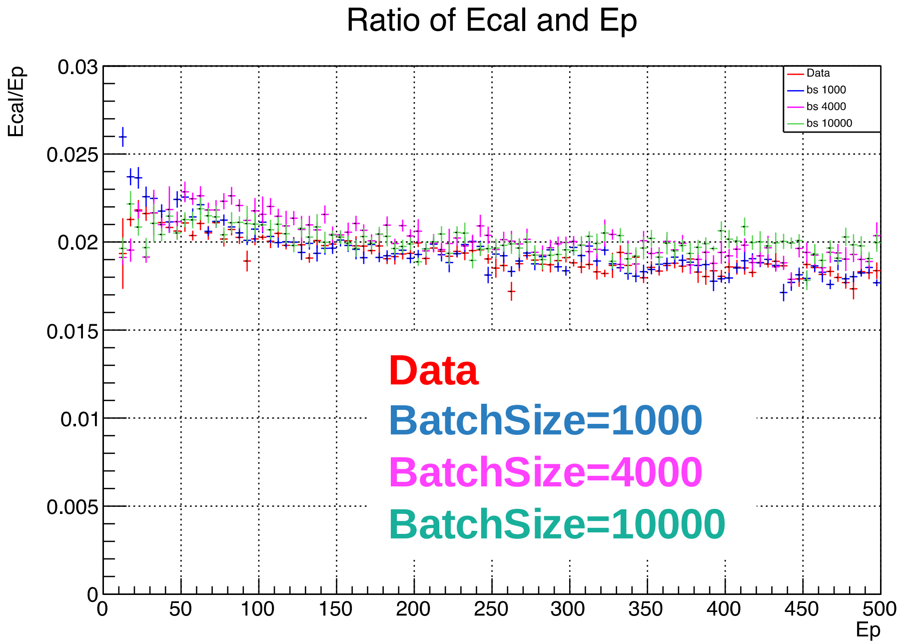
HPC friendly!

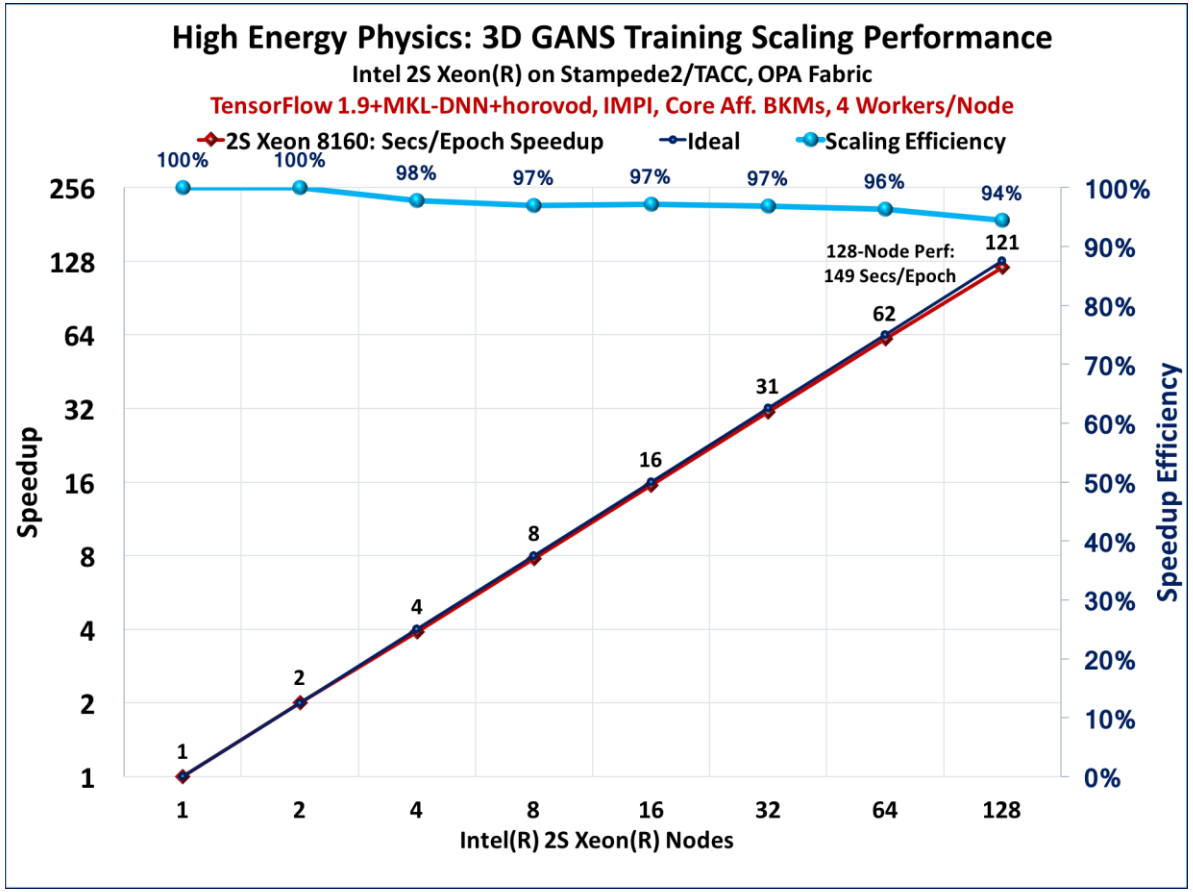
Run on TACC Stampede2 cluster:

- Dual socket Intel Xeon 8160
- 2x 24 cores per node, 192 GB RAM
- Intel® Omni-Path Architecture

Keras + Tensorflow 1.9

Study performance degradation





Generalisation & Development

Understanding performance and coverage

Test different performance figures:

- Features-based
- Pixel-based
- “Inception-like”

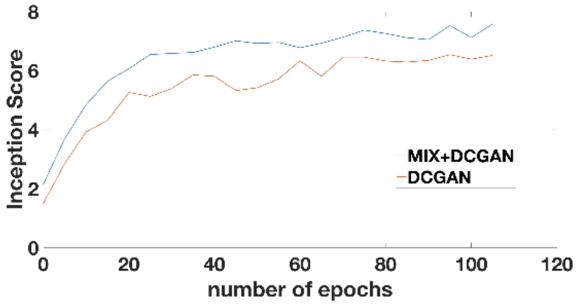
Understand coverage

An empirical contribution: MIX + GAN protocol

(suggested by our analaysis of equilibrium)

- Take any existing GAN (any architecture)
- Black box change: replace generator by weighted mixture of k generators (k = largest number that allows training to fit in your GPU)
- Train mixing weights via backpropagation; use entropy regularizer to discourage mixture from collapsing.

Often stabilizes and improve training.
Effective way to add capacity to generator.
(hyper-parameter search is not easy for GANs)

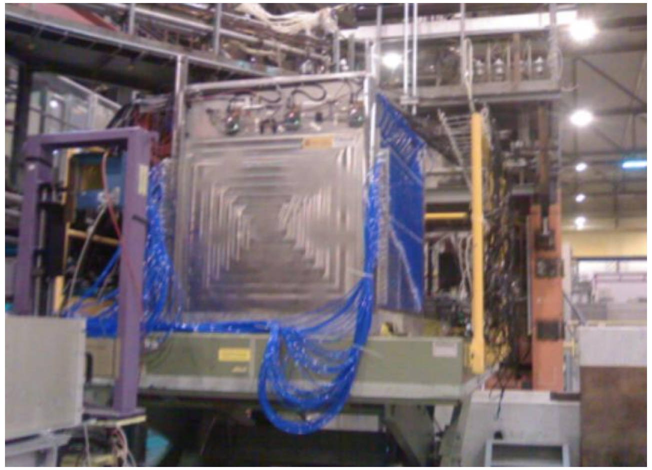


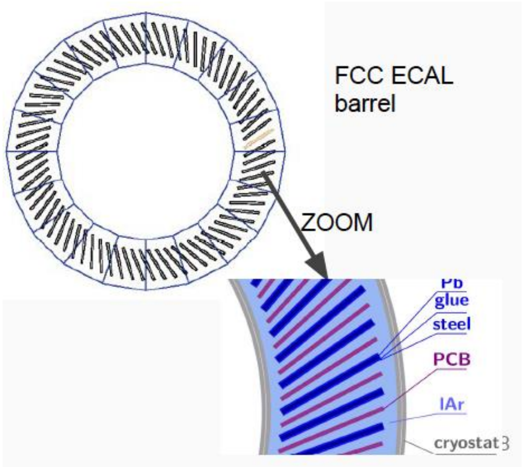
Sanjeeva Arora, ICCV 2017

Generalisation to different calorimeters

- Our baseline is an example of next generation highly granular calorimeter
- Extend to other cases
  - FCC LAr calorimeter
  - CALICE SDHCAL
  - HGCAL
- Explore optimal network topology according to the problem to solve
  - Hyper-parameters tuning and meta-optimization
    - mpi_learn/mpi_opt

SDHCAL prototype during SPS test beam





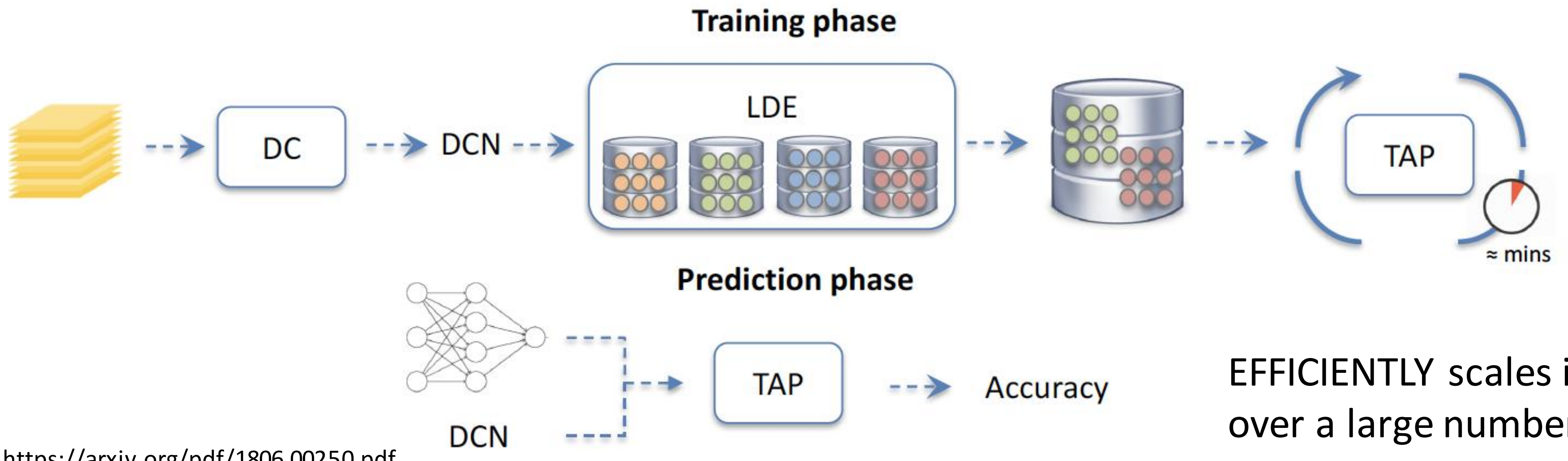
TAPAS: Train-less Accuracy Predictor for Architecture Search

An automated approach to determine optimal network architectures, with a low amount of training

Determine and train the best CNN configuration for the simulation of a specific detector.

input : detailed simulation of the detector and the detector parameters

Output: a trained CNN with optimal parameters for the problem at hand.



EFFICIENTLY scales in a few seconds over a large number of networks.

https://arxiv.org/pdf/1806.00250.pdf

Summary and Plan

Deep Learning is a powerful approach to solve many problems in society, industry and science

- Thanks to the availability of data and computing resources

R&D on Generative Models is extremely active

- Promising approach to solve the "fast simulation problem"
- More work is needed to fully understand performance and limits of the approach
  - And choose the applicabilty range!

Many promising applications in our field

Our fastsim R&D project has reached very good results

- Work on-going to detail performance

Collaboration within and beyond the HEP community is essential!

GANs for earth observation

- Train Progressively Growing GANs on UNOSAT Rukban Camp Dataset.
  - Preliminary test shows encouraging results
  - GAN generated tiles of 256x256 pixels
- Further steps:
  - Assess accuracy and image fidelity
  - Measure sample variance
  - Scale up to ~1M pixels
  - Generate multi-spectral images

S.P. Mohanty

Thanks!

Questions?

References

http://cs231n.github.io/

- Pattern Recognition and Machine Learning, Bishop (2006)
- Elements of Statistical Learning (2nd Ed.) Hastie, Tibshirani & Friedman 2009
- Introduction to machine learning, Murray: http://videolectures.net/bootcamp2010_murray_iml/
- What is Machine Learning, Ravikumar and Stone: http://www.cs.utexas.edu/sites/default/files/legacy_files/research/documents/MLSSIntro.pdf
- CS181, Parkes and Rush, Harvard University: http://cs181.fas.harvard.edu
- CS229, Ng, Stanford University: http://cs229.stanford.edu/
- Machine learning in high energy physics, Alex Rogozhnikov: https://indico.cern.ch/event/497368/

http://scs.ryerson.ca/~aharley/vis

http://cs. stanford.edu/people/karpathy/convnetjs/

Keras.io

http://www.asimovinstitute.org/neural-network-zoo/

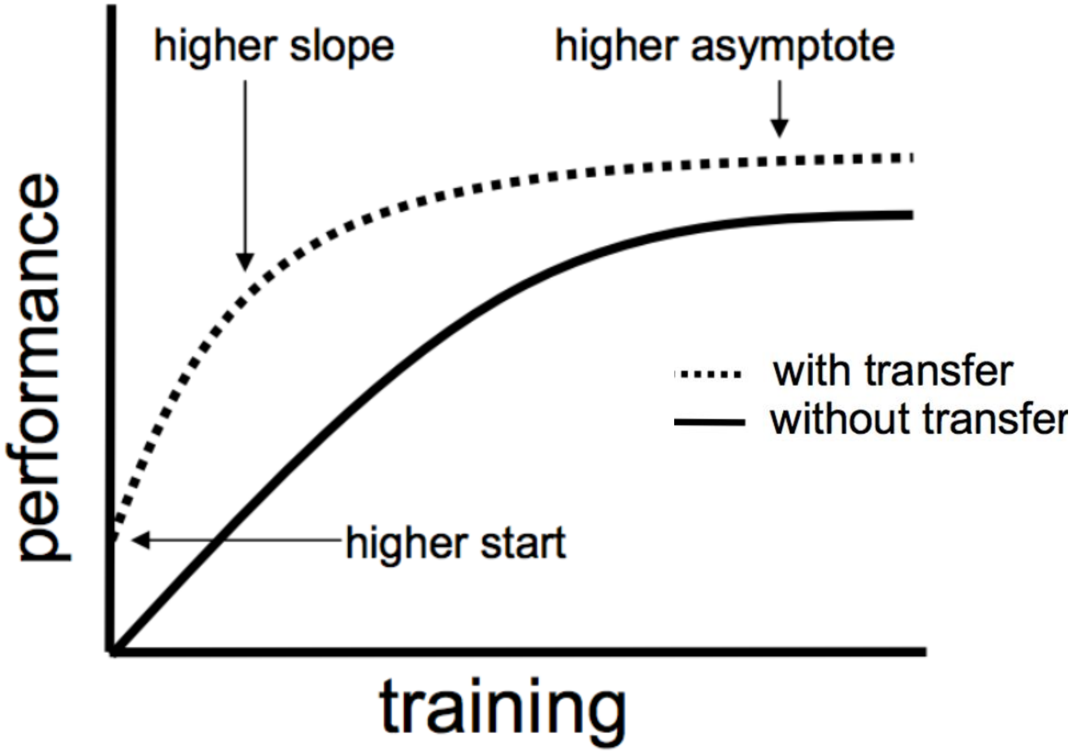
http://scikit-learn.org/

Transfer learning

“Transfer learning and domain adaptation refer to the situation where what has been learned in one setting … is exploited to improve generalization in another setting”

Page 526, Deep Learning, 2016.

- Improvement of learning on a new task, transferring the knowledge already learned on a similar task
  - Might be the only option given the amount of resources needed from training
- Carefully choose how much of the pre-trained model to use in new one
  - CNN features are more generic in early layers and more dataset-specific in later layers

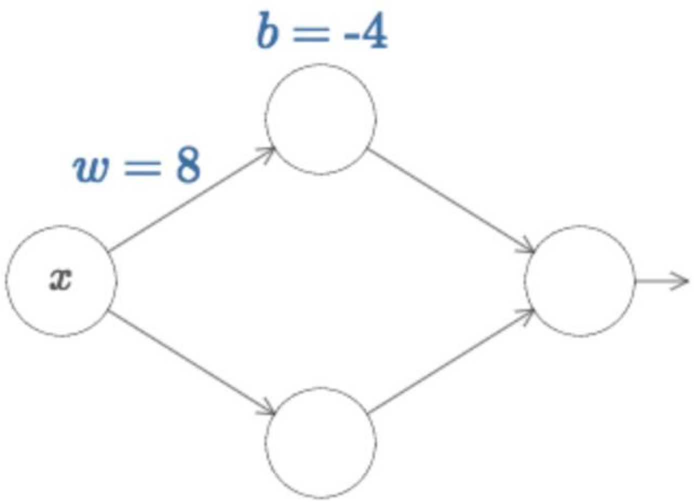


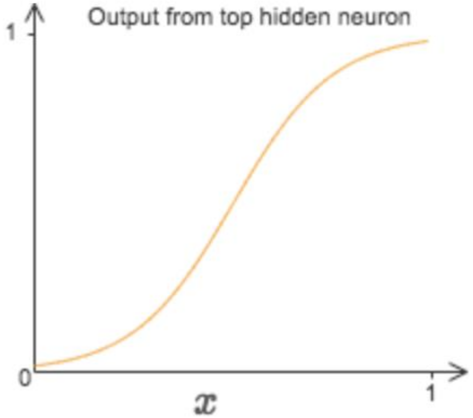
Representational power

NN with at least one hidden layer are *universal approximators*

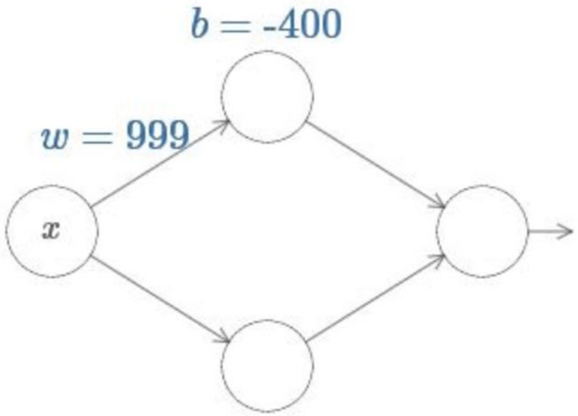
$\sigma(wx + b)$

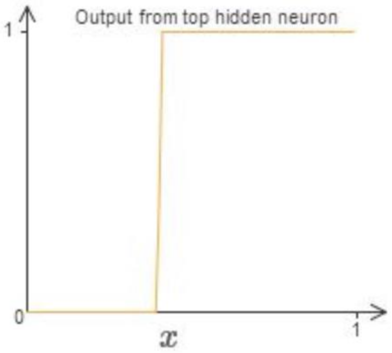
$\sigma(z) \equiv 1/(1 + e^{-z})$





Playing with the w, b parameters we can modify the shape of the sigmoid



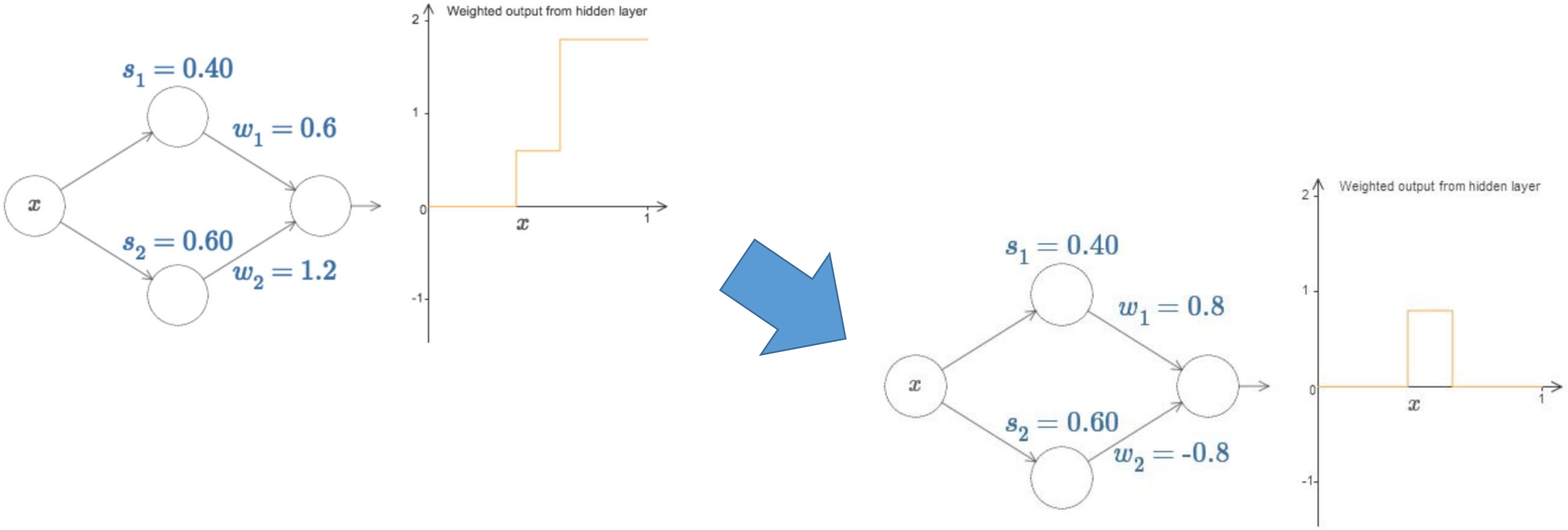


http://neuralnetworksanddeeplearning.com/chap4.html

Or Approximation by Superpositions of Sigmoidal Function

Representational power

We can add nodes and introduce "steps"



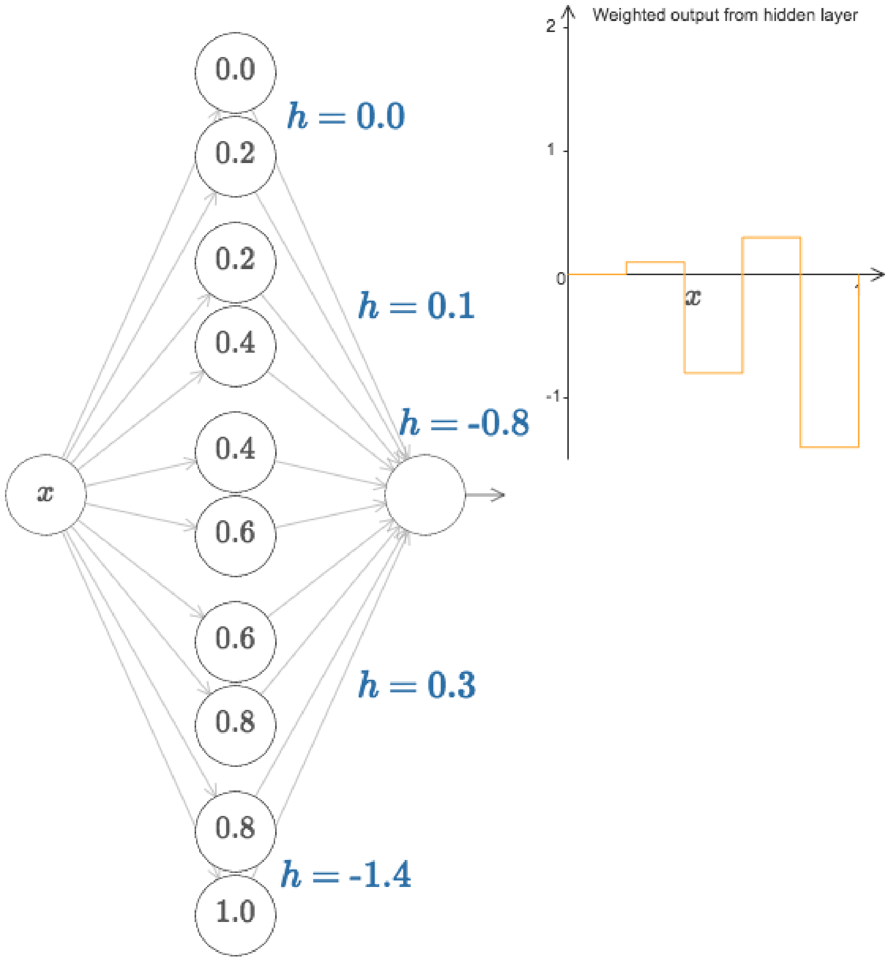
http://neuralnetworksanddeeplearning.com/chap4.html

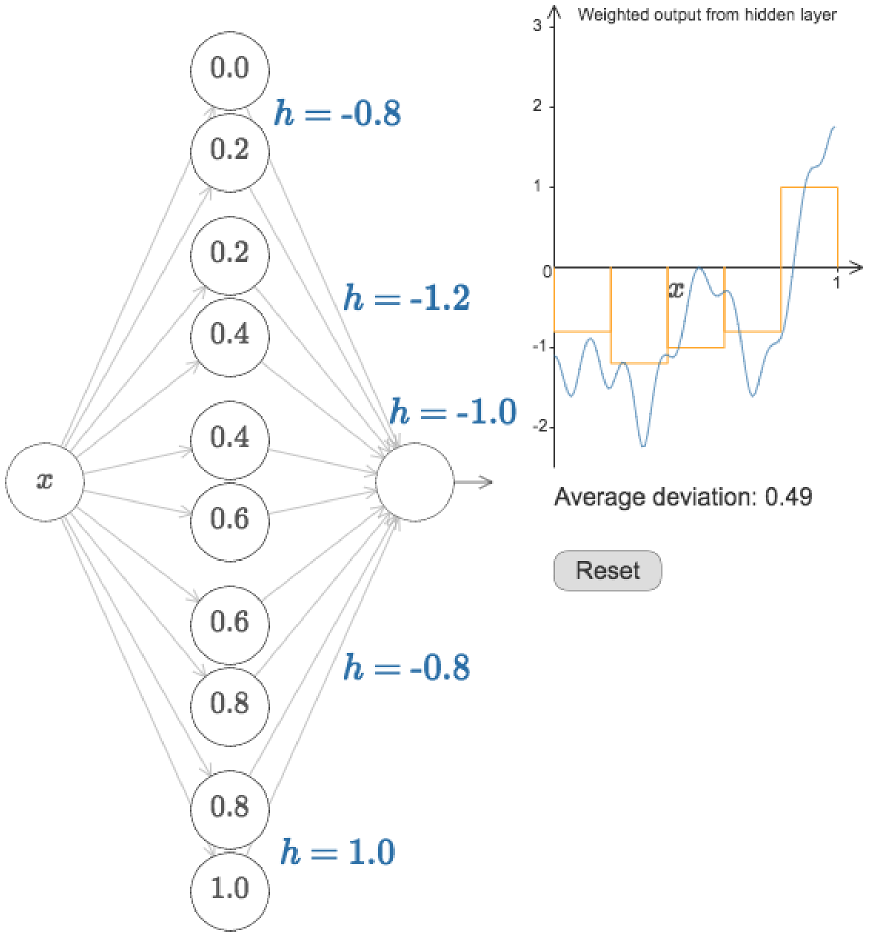
Or Approximation by Superpositions of Sigmoidal Function

Representational power

Increasing complexity

NN with a single hidden layer can be used to approximate any continuous function to any desired precision





http://neuralnetworksanddeeplearning.com/chap4.html

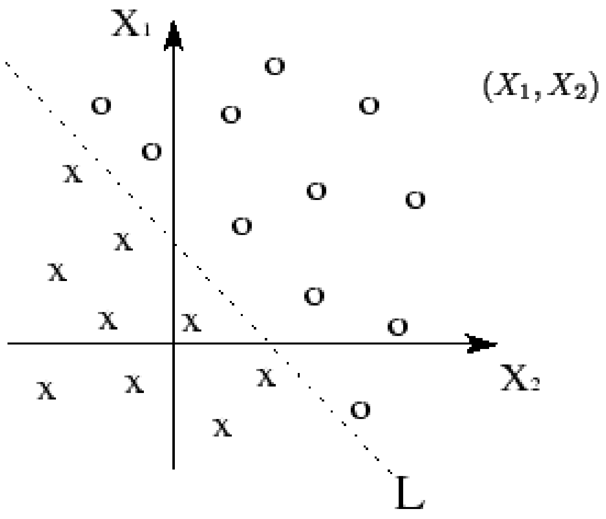
Or Approximation by Superpositions of Sigmoidal Function

The need for depth

A single layer perceptron can categorize "linearly separable" patterns

Two classes classification:

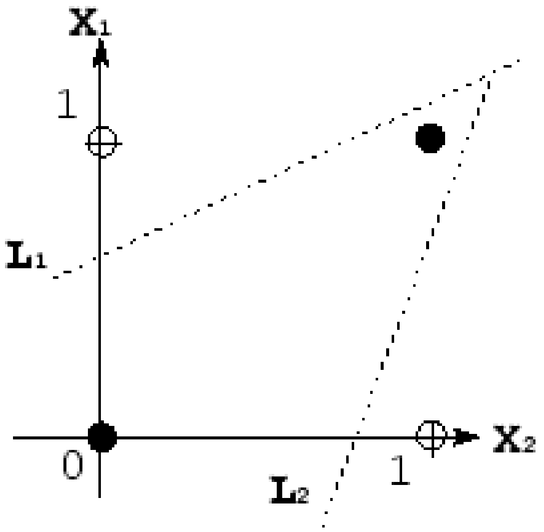
(OR function) (linearly separable)



Exclusive OR is an example of a non linearly separable patter:

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

$$Y = X_1 \oplus X_2$$

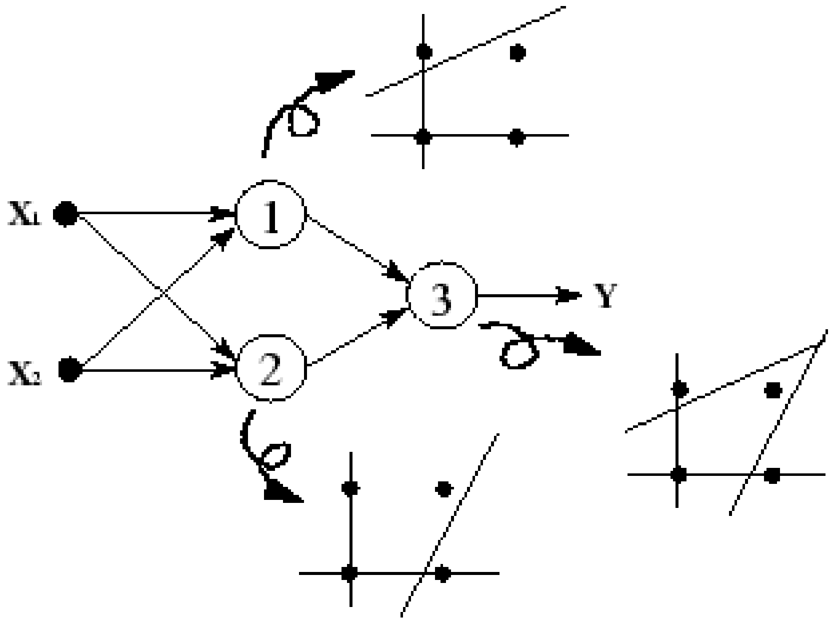


(tutorial) http://www.theprojectspot.com/tutorial-post/introduction-to-artificial-neural-networks-part-1/7
(images)http://www.ece.utep.edu/research/webfuzzy/docs/kk-thesis/kk-thesis-html/node19.html

The need for depth

Need a Multi-Layer architecture to solve the ex OR problem:

Two-stages approach



(tutorial) http://www.theprojectspot.com/tutorial-post/introduction-to-artificial-neural-networks-part-1/7
(images)http://www.ece.utep.edu/research/webfuzzy/docs/kk-thesis/kk-thesis-html/node19.html

Back-propagation

A simple visual example

$$f(x, y, z) = (x + y)z.$$

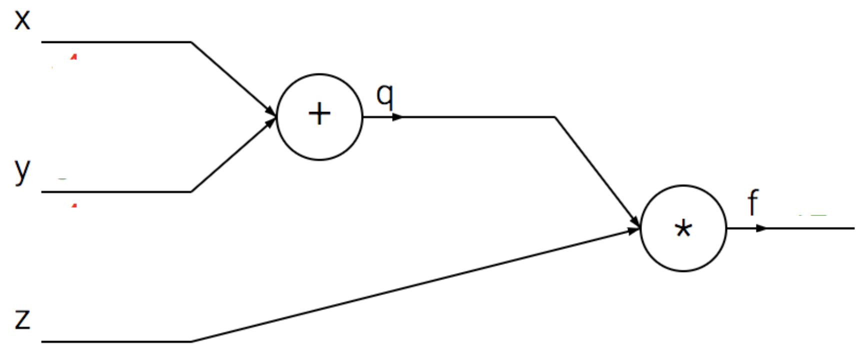
$$q = x + y$$

$$f = qz$$

$$\frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$\frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q \quad \Rightarrow \quad \frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

Forward pass computes values from inputs to output
X = -2, Y = 5, Z = -4



How does a change in Y affect f?
Calculate (forward) derivatives

OR
use **backward derivatives**: starts at the end and recursively applies the chain rule

Back-propagation

A simple visual example

$$f(x, y, z) = (x + y)z.$$

$$q = x + y$$

$$f = qz$$

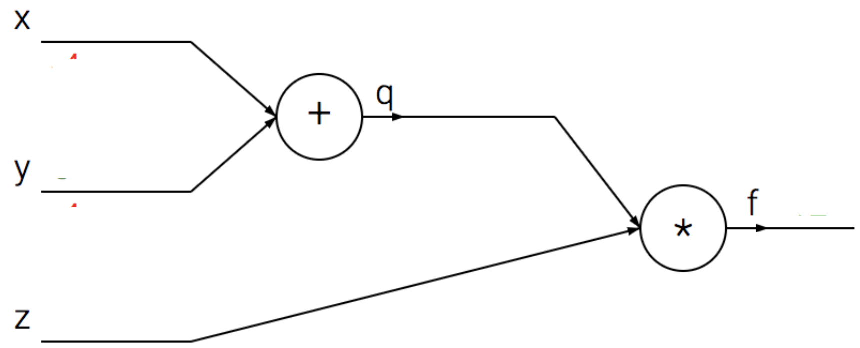
$$\frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$\frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

Forward pass computes values from inputs to output

X = -2, Y = 5, Z = -4



How does a change in Y affect f?

Calculate (forward) derivatives

OR

use **backward derivatives**: starts at the end and recursively applies the chain rule

Back-propagation

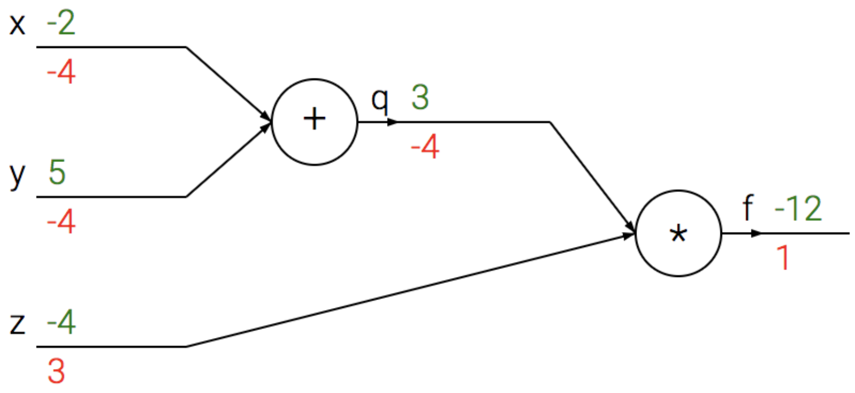
A visual example

Backward derivatives approach is much more efficient in the case of large graphs

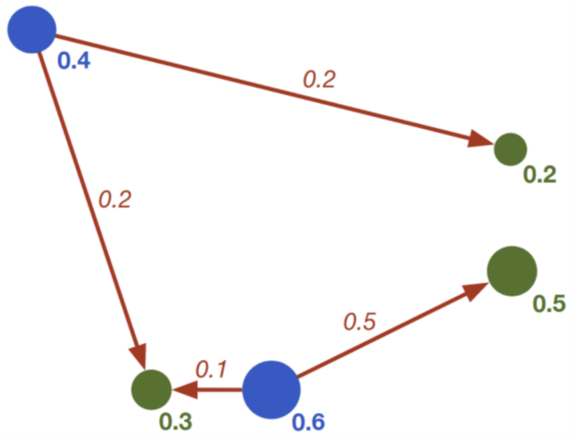
Because of the chain rules, at each step the derivative depends only

- On the derivatives already calculated for the parent nodes
- On the node values calculated during the forward pass

Gradients flow "backward" through the graph



Wasserstein GAN

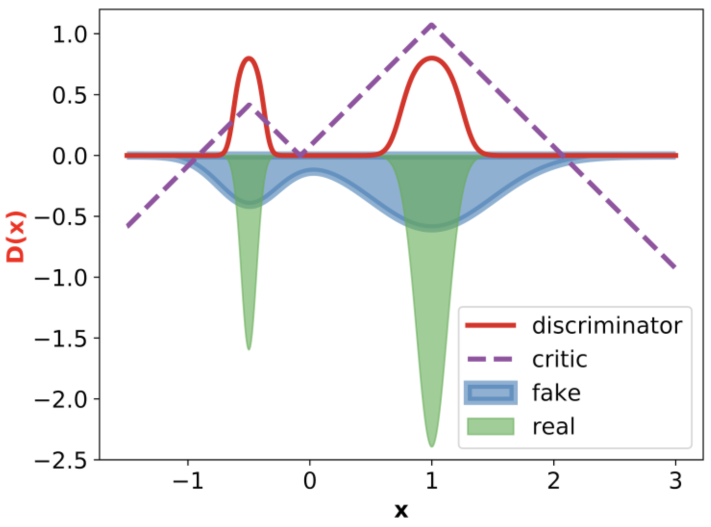


Also called Earth-Mover-Distance:

Interpret one distribution as target, one as earth heap

Distance of distributions = effort to move earth heap to target (mass x distance)

$$D_W = \underbrace{\min_{\gamma \in \prod(P_x, P_{\hat{x}})}}_{\text{optimal transport plan}} \underbrace{\mathbb{E}_{(x,\hat{x})\sim\gamma}}_{\text{mass}} \underbrace{\|x - \hat{x}\|_2}_{\text{distance}}$$



Performance metrics

Kullback-Leibler divergence:

$$D_{\mathrm{KL}}(P \| Q) = \sum_i P(i) \log\left(\frac{P(i)}{Q(i)}\right)$$

Inception score: use Google Inception network (Szegedy et al., 2016), pre-trained on the ImageNet (Deng et al., 2009) dataset

$$\mathrm{IS}(\mathbb{P}_g) = e^{\mathbb{E}_{\mathbf{x} \sim \mathbb{P}_g}[KL(p_{\mathcal{M}}(y|\mathbf{x}) || p_{\mathcal{M}}(y))]}$$

Maximum Min Discrepancy: measures dissimilarity between two distributions for some fixed kernel function

$$\mathrm{MMD}(\mathbb{P}_r, \mathbb{P}_g) = \left(\mathbb{E}_{\substack{\mathbf{x}_r, \mathbf{x}_r' \sim \mathbb{P}_r, \\ \mathbf{x}_g, \mathbf{x}_g' \sim \mathbb{P}_g}} \left[k(\mathbf{x}_r, \mathbf{x}_r') - 2k(\mathbf{x}_r, \mathbf{x}_g) + k(\mathbf{x}_g, \mathbf{x}_g') \right] \right)^{\frac{1}{2}}$$

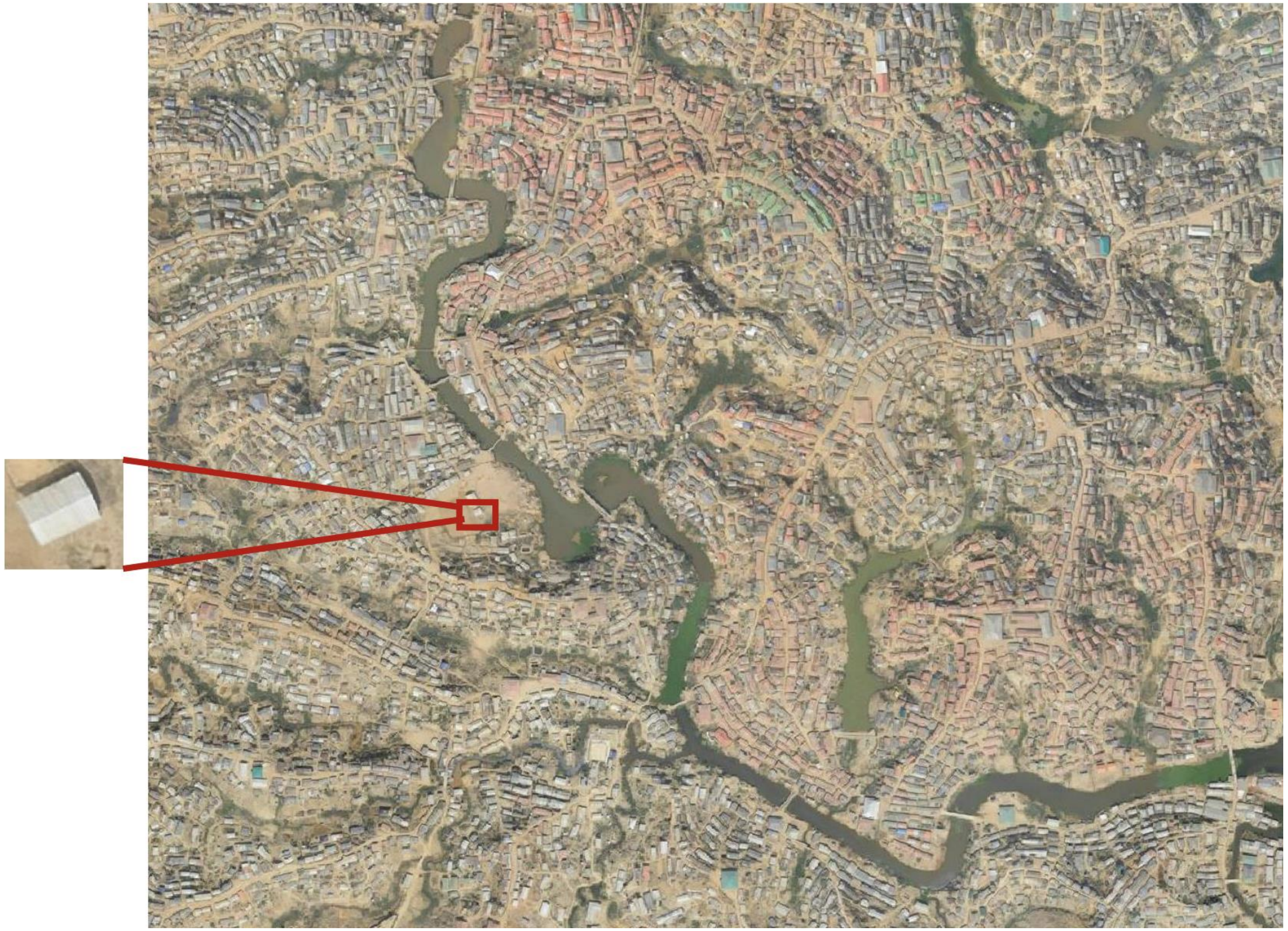
Fréchet Inception Distance: compares mean and covariance of real and GAN probability distribution

$$\mathrm{FID}(\mathbb{P}_r, \mathbb{P}_g) = ||\mu_r - \mu_g|| + \mathrm{Tr}(\mathbf{C}_r + \mathbf{C}_g - 2(\mathbf{C}_r \mathbf{C}_g)^{1/2}).$$

Counting shelters in refugee camps

CERN openlab and UNOSAT collaboration

(UN Operational Satellite Applications Centre)



Scan million pixels satellite photos for disaster relief:

- Evolution of refugee camps
- Natural disasters
- Building damage

Because of the high level of precision required

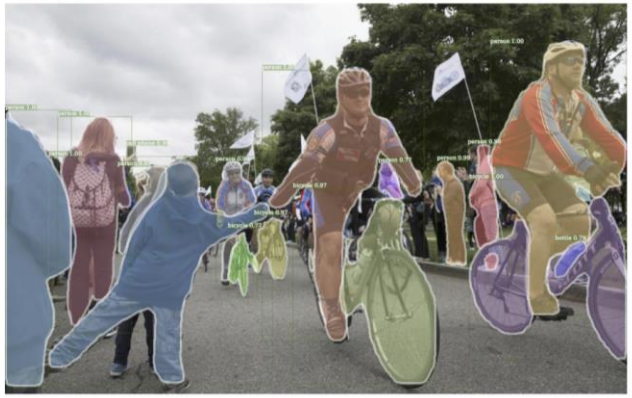
- it's done MANUALLY!!!!

Counting shelters in refugee camps

CERN openlab and UNOSAT collaboration

(UN Operational Satellite Applications Centre)

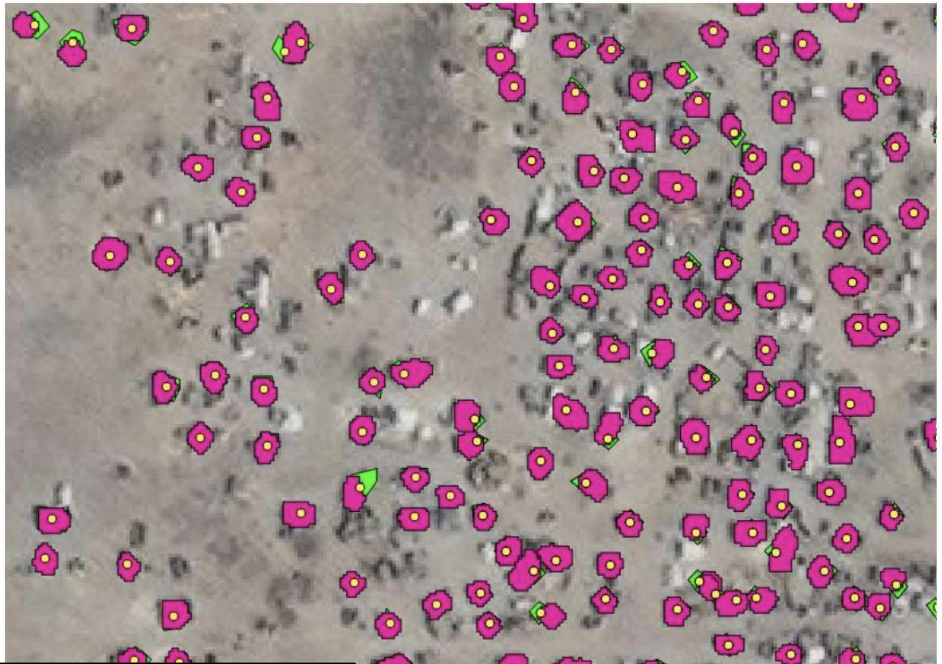
Why not use CNN instead??

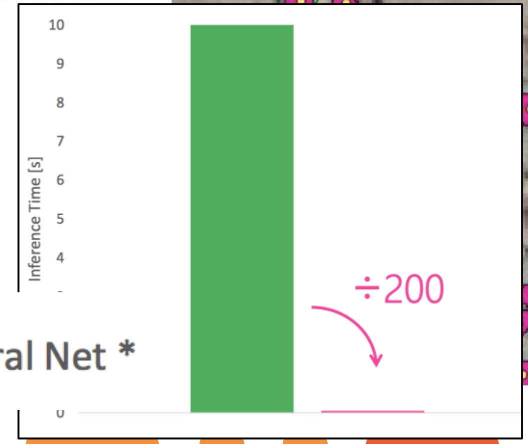


Detectron Framework (FacebookAI)

Retrain & encode point data cleverly

Unosat Adapted model





Human Neural Net *

Transfer learning from RCNN model
Average precision is 82%
Speedup is x200

CERN openlab

https://indico.cern.ch/event/727274/contributions/3100369/