Machine Learning at CERN

Lorenzo Moneta
(CERN EP-SFT)

ECMWF / EUMETSAT Visit, September, 26, 2019

Where Machine Learning is Used ?

- Natural Language Processing
- Speech and handwriting recognition
- Object and image recognition
- Fraud detection
- Financial marker analysis
- Search engines
- Spam and virus detection
- Medical diagnosis
- Robotics control
- Automation: e.g. self-driving cars
- Advertising (recommender systems)
- ……..

Growing very fast and thanks to deep learning !

Facial recognition

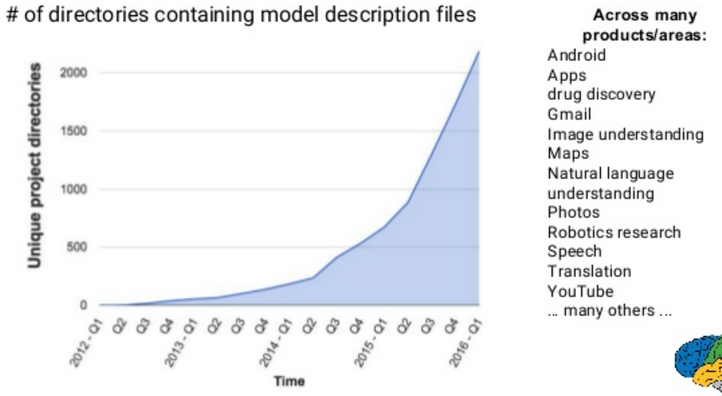
Autonomous ("self-driving") vehicles

amazon

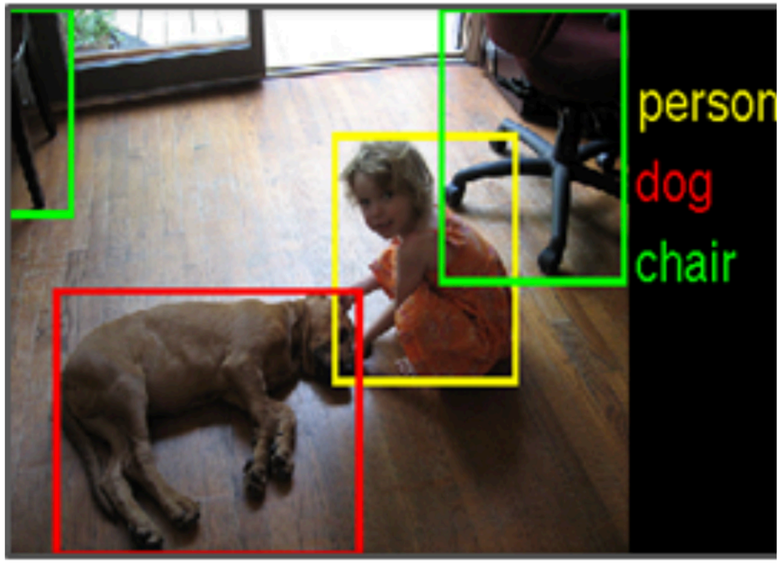
NETFLIX SOUNDCLOUD

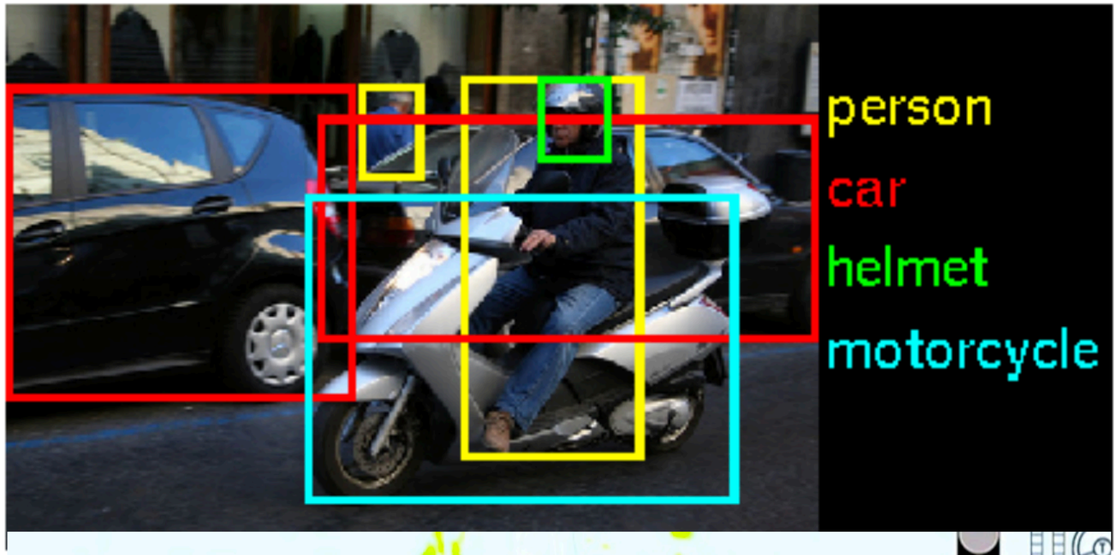
Recommendation engines

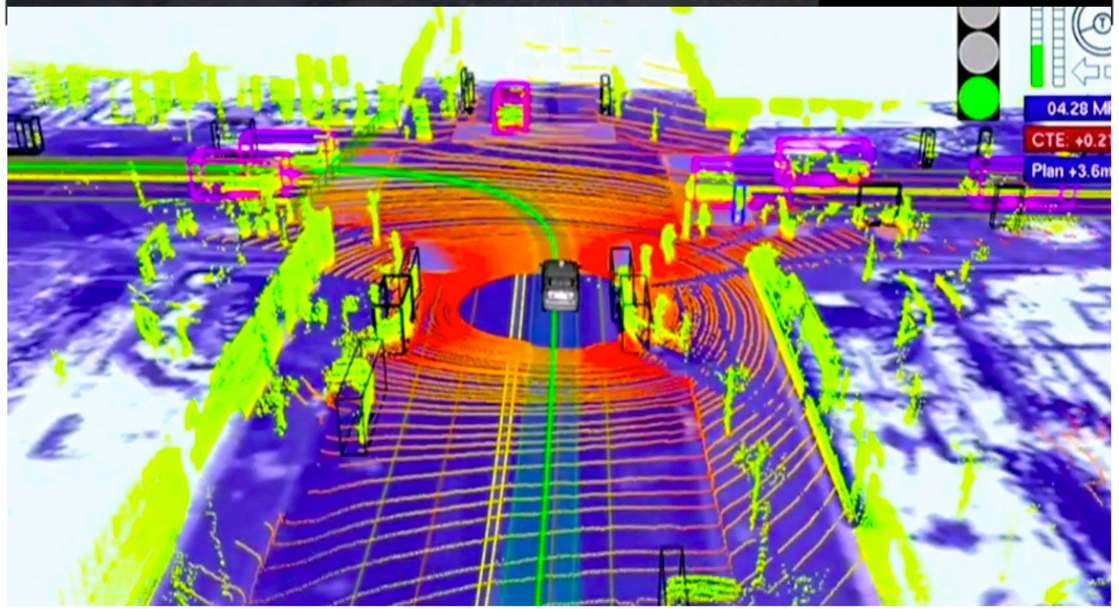
Growing Use of Deep Learning at Google



Examples of ML Applications

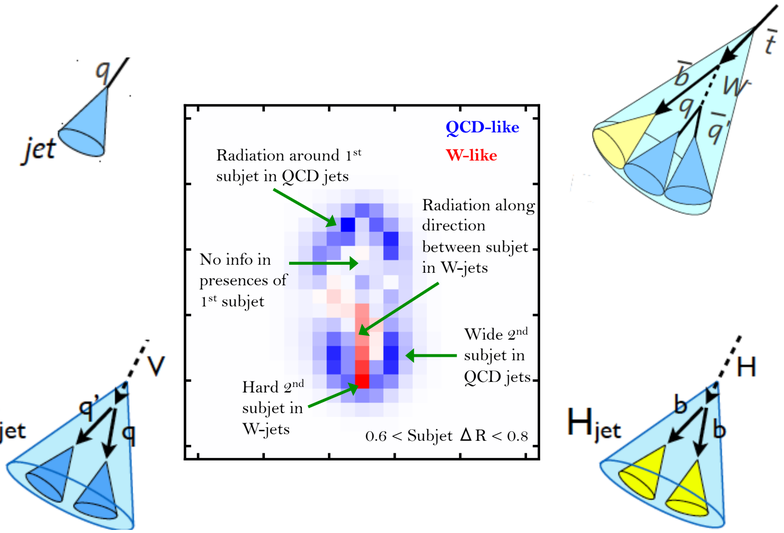


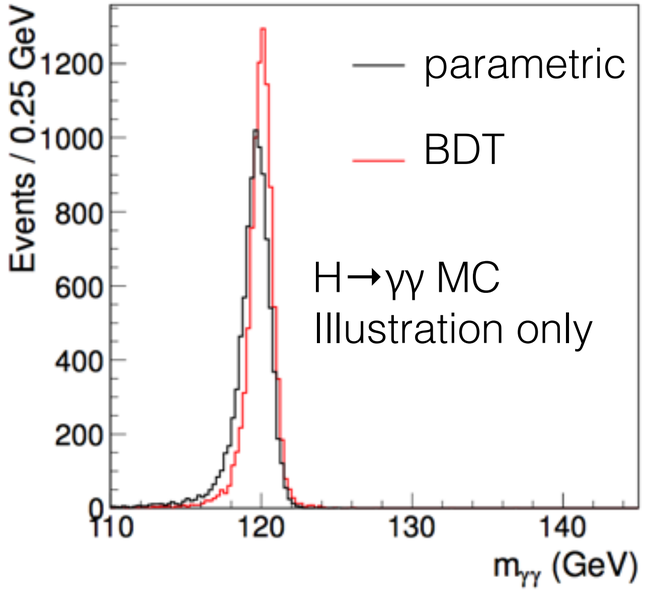




Machine Learning in High Energy Physics

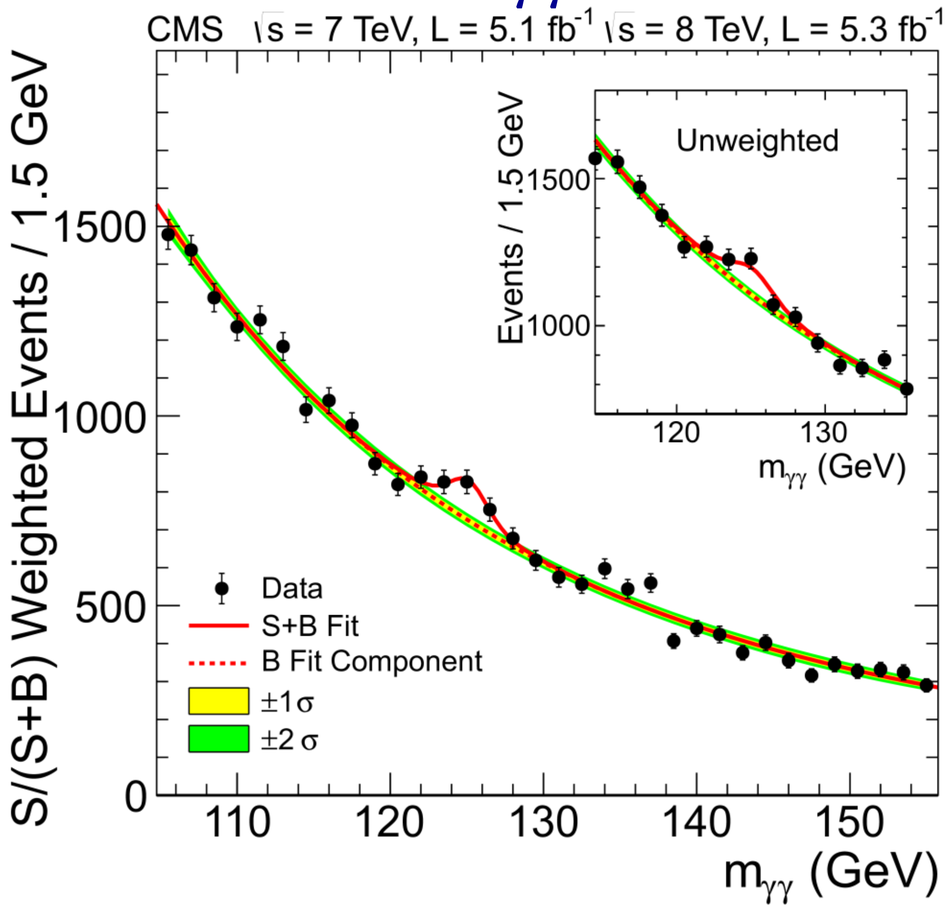
- **In analysis and reconstruction**
  - Classifying signal from background events
  - Reconstructing and identifying particles
  - Improving energy/mass resolution
  - Energy calibration
  - Fast simulation
- **In trigger (online selection) and data acquisition**
  - Quickly identifying complex final states
  - Data quality monitoring
- **In computing**
  - Estimate dataset popularity
  - Optimisation of resources
  - Computer security



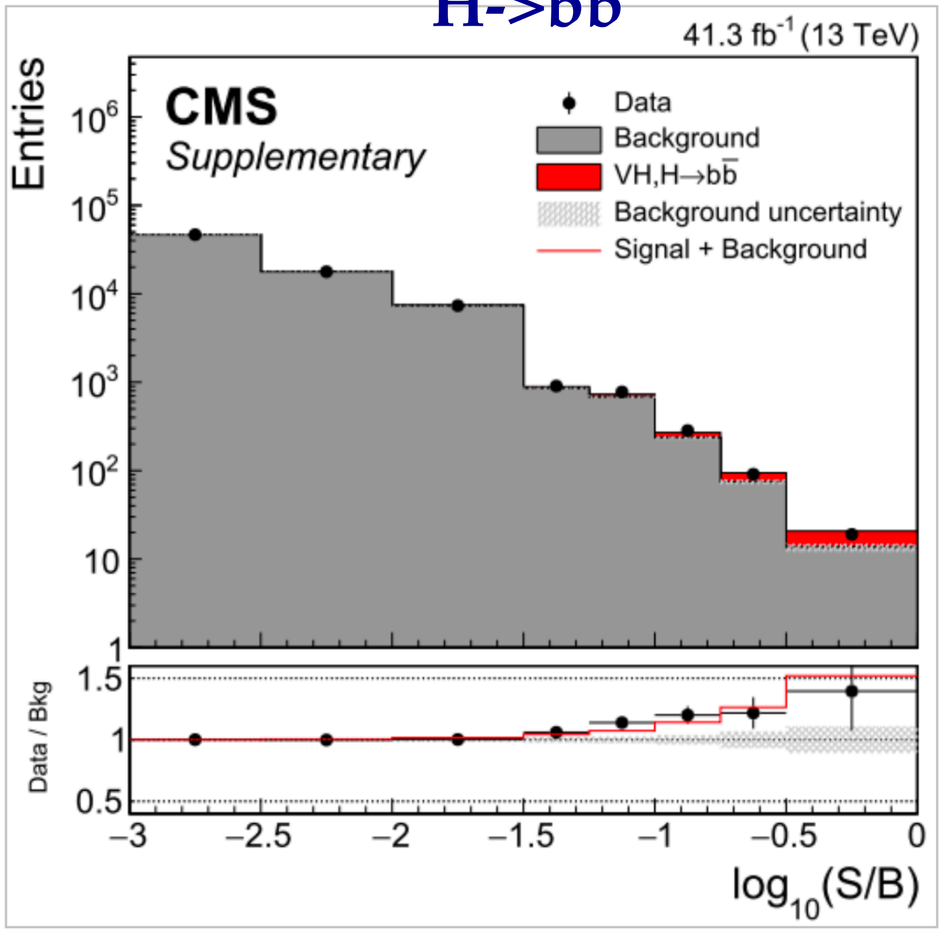


Example: Higgs Discovery

H->γγ



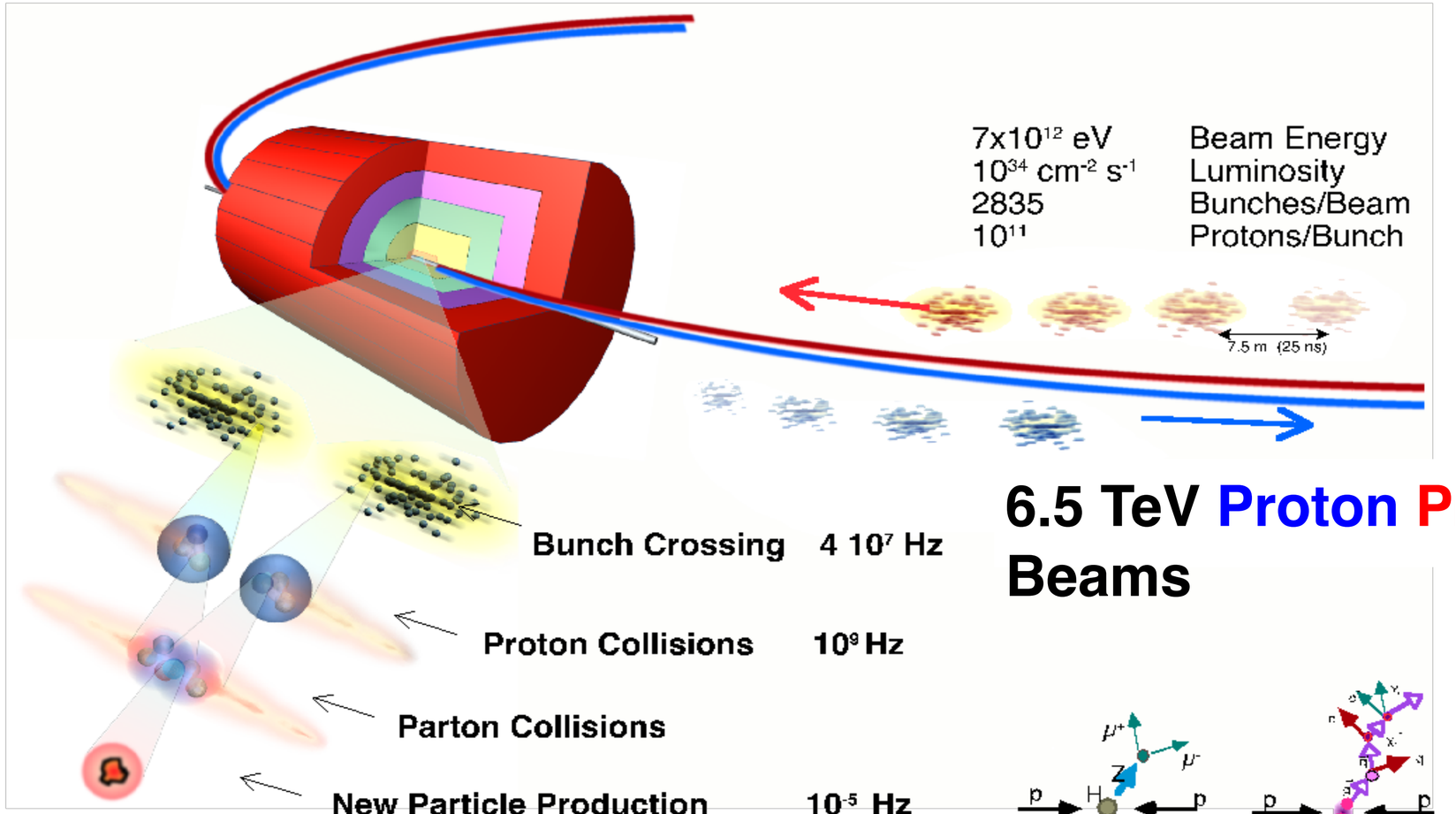
H->bb



Improvement in analysis from all areas using machine learning

[S. Gleyzer]

Collisions in the LHC



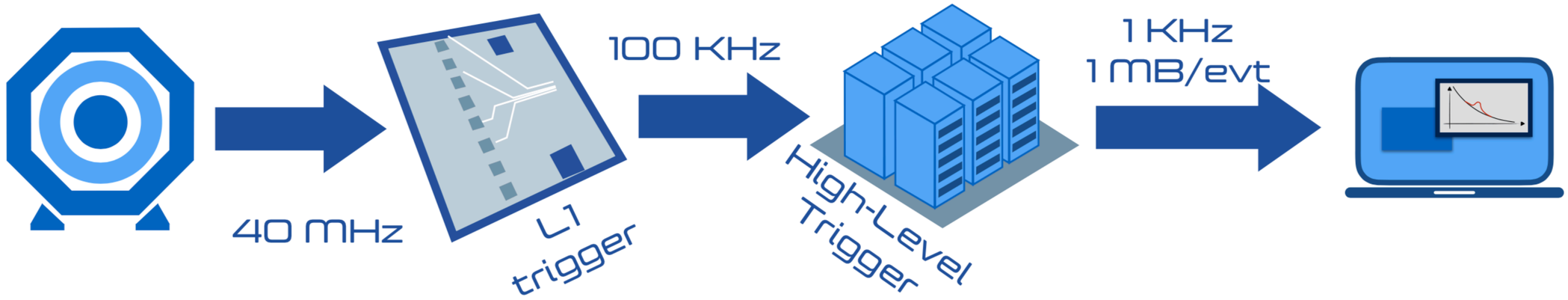
6.5 TeV Proton Proton Colliding Beams

Selection of 1 event in 10,000,000,000,000

The LHC: Big Data Problem

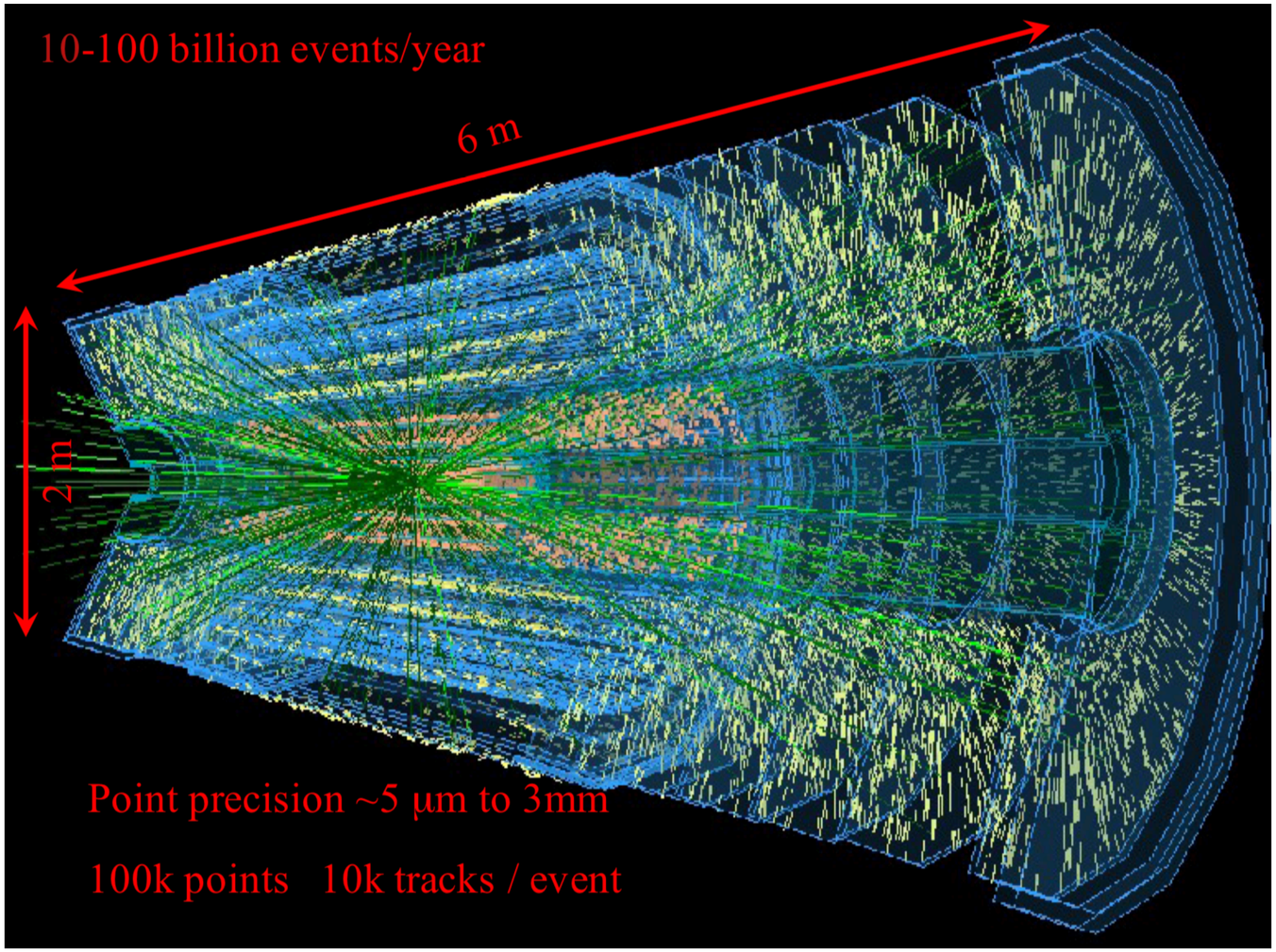
LHC Today	HL_LHC (~ 2025)
~40 collisions/event	~200 collisions/event
~10 sec/event processing time	~minute/event processing time(*)
(at best) Same computing resources as today	(at best) Same computing resources as today

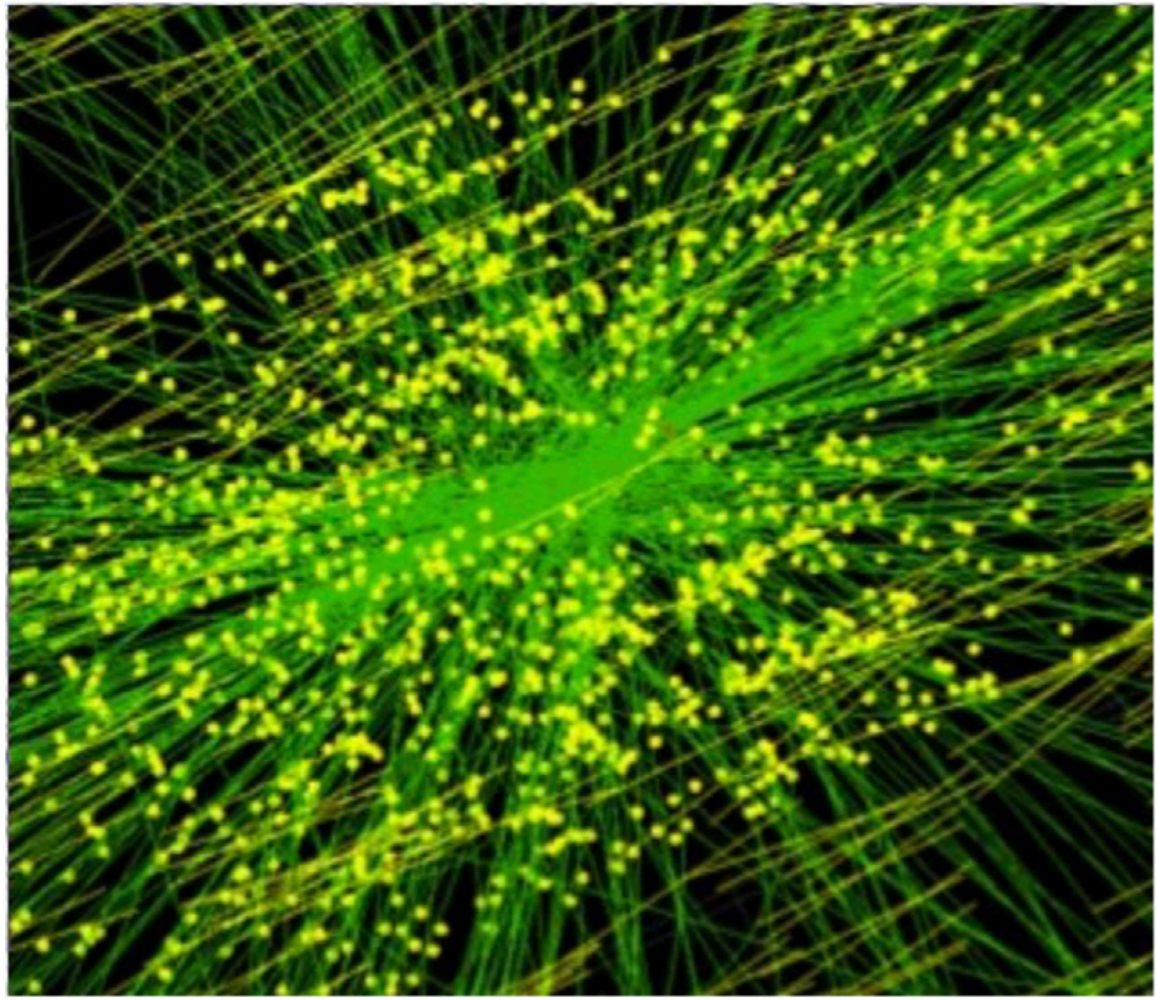
- need to filter online
- **Deep Learning as possible solution to the problem**

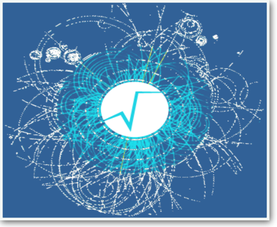


[M. Pierini]

Example: Track Reconstruction at LHC







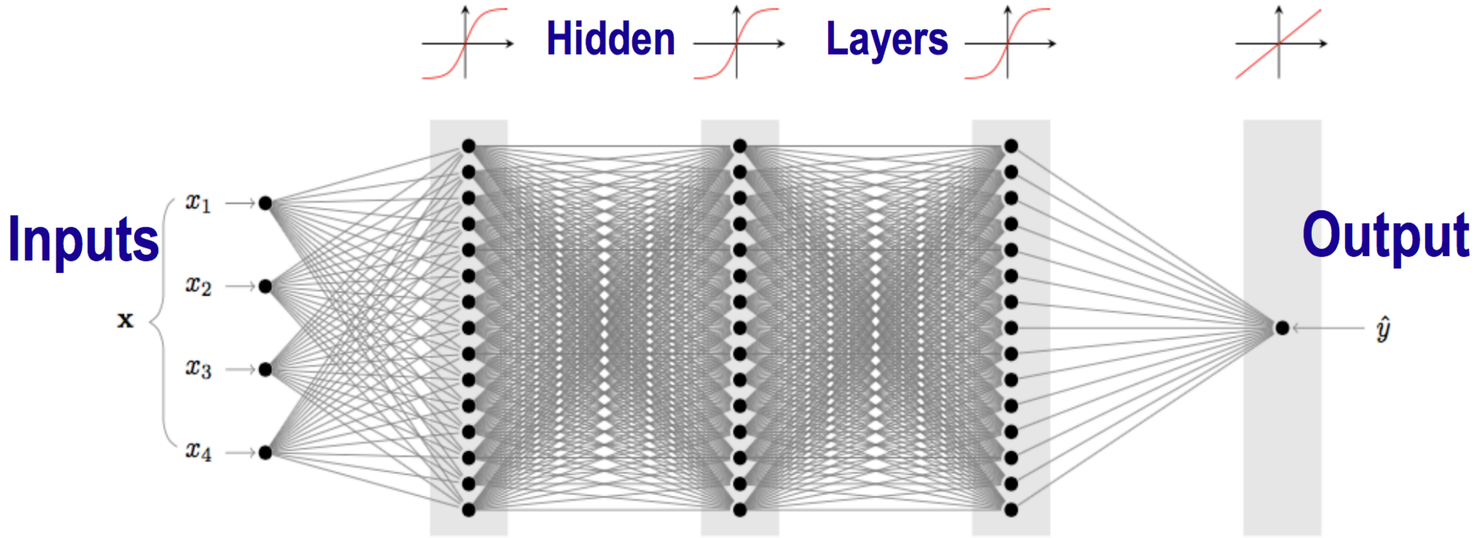
Deep Learning in Physics

In last years large growing interest in applying Deep Learning to problems in High Energy Physics:

- **Deep Neural Networks** for optimal event classification and tagging
- **Computer vision techniques** (Convolutional NNs) applied to particle detector images
- **Natural Language processing** (Recurrent NNs) to particle sequences
- **Anomaly detection with auto-encoders**
- **Graph networks** for particle tracking and irregular geometries
- **Generative models** for fast simulation (Generative Adversarial Networks and Variational Auto-encoders)

Deep Neural Networks (DNN)

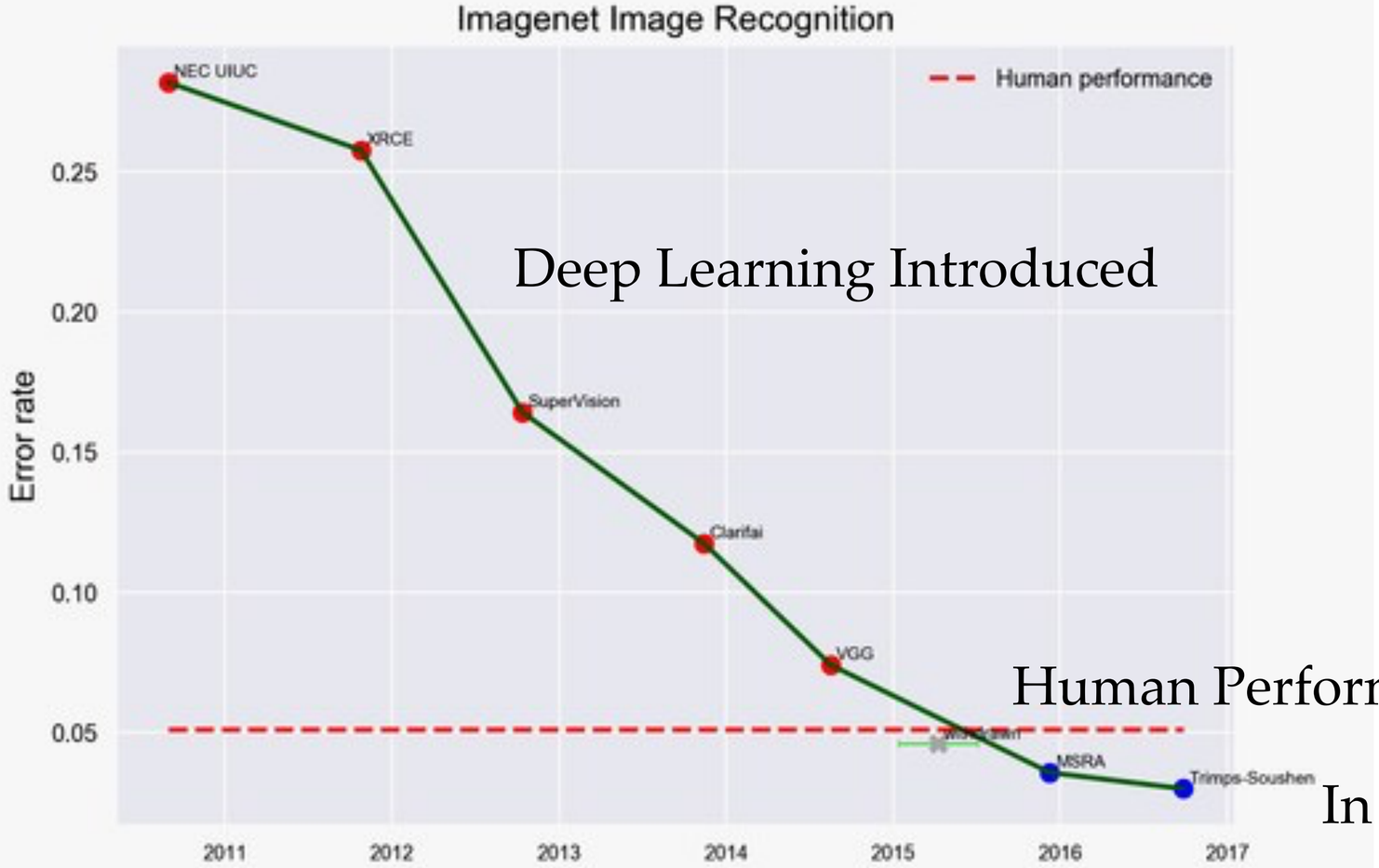
- An artificial neural networks using several hidden layers
- Can provide significant performance improvements
  - hidden layers help in learning data representations
- Very popular in recent years thanks to improvement in computing performances
  - thanks to usage of **Graphics Processing Units (GPU)**



$$\mathbf{u}_1 = f(\mathbf{W}_1\mathbf{x} + \boldsymbol{\theta}_1) \quad \mathbf{u}_2 = f(\mathbf{W}_2\mathbf{u}_1 + \boldsymbol{\theta}_2) \quad \mathbf{u}_3 = f(\mathbf{W}_3\mathbf{u}_2 + \boldsymbol{\theta}_3) \quad \mathbf{u}_4 = f(\mathbf{W}_4\mathbf{u}_4 + \boldsymbol{\theta}_4)$$

Imagenet Challenge

Image Recognition Challenge



Deep Learning Introduced

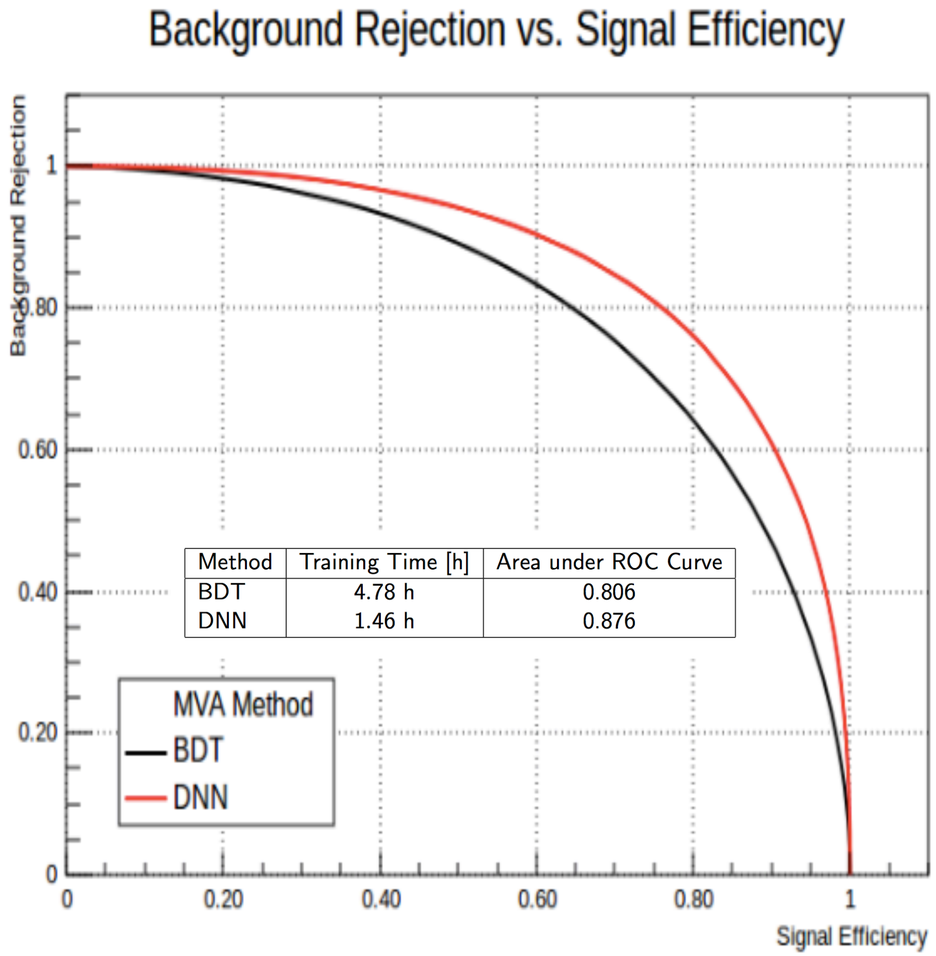
Human Performance 5%

In 2017 ~ 3%

Deep Learning Performances

Example of Deep Learning for classification on a HEP data set

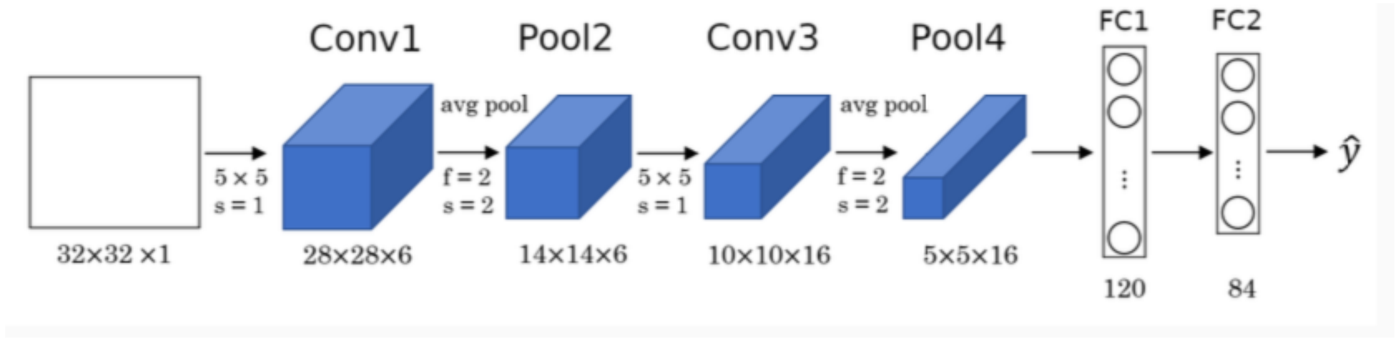
DNN vs BDT



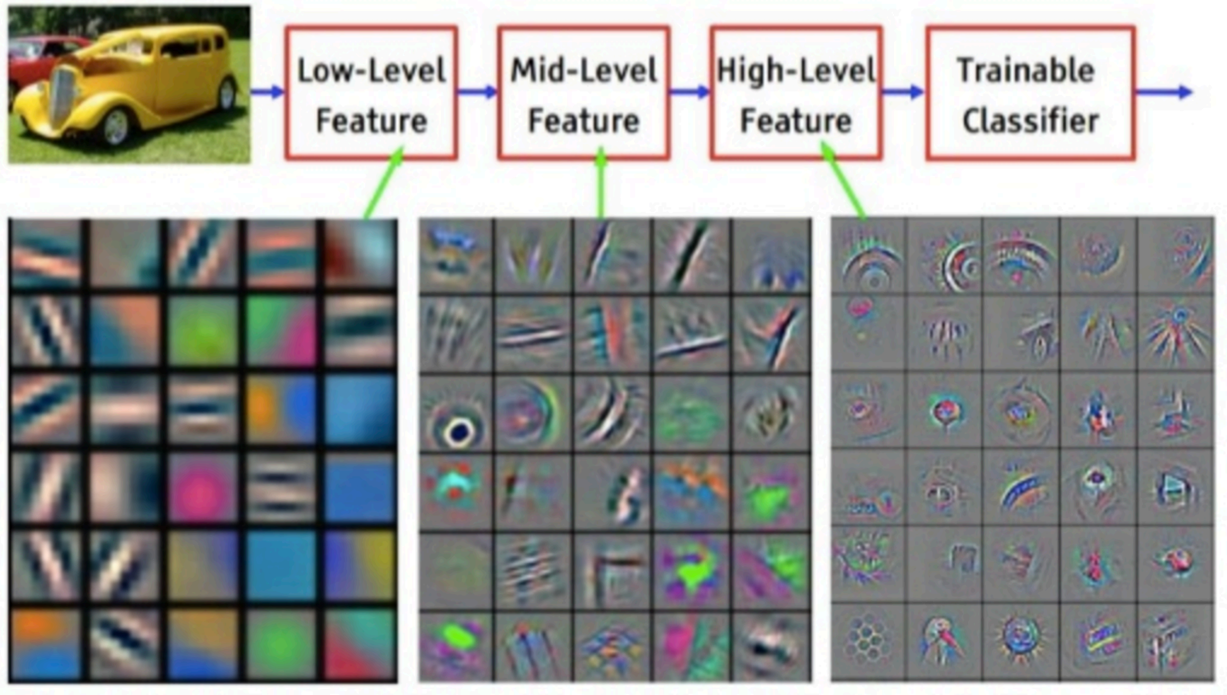
from searches for new types of Higgs bosons
[P. Baldi et al., Nat. Comm. 5:4308(2014)]

- High classification performance compared to other ML methods (e.g Boosted Decision Trees)
- Fast training time possible thanks to parallelisation gains in using the Graphical Processing Units (GPU)

Convolutional Networks



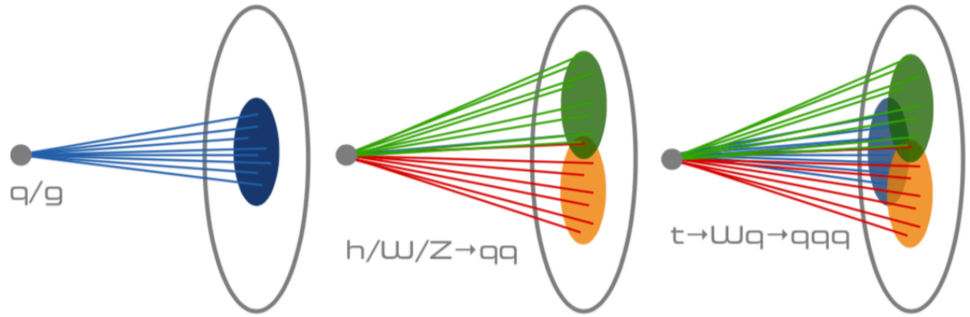
- Local structures captured at the early stage convolutions
- Long range structures in late stage convolutions and in the final dense layers

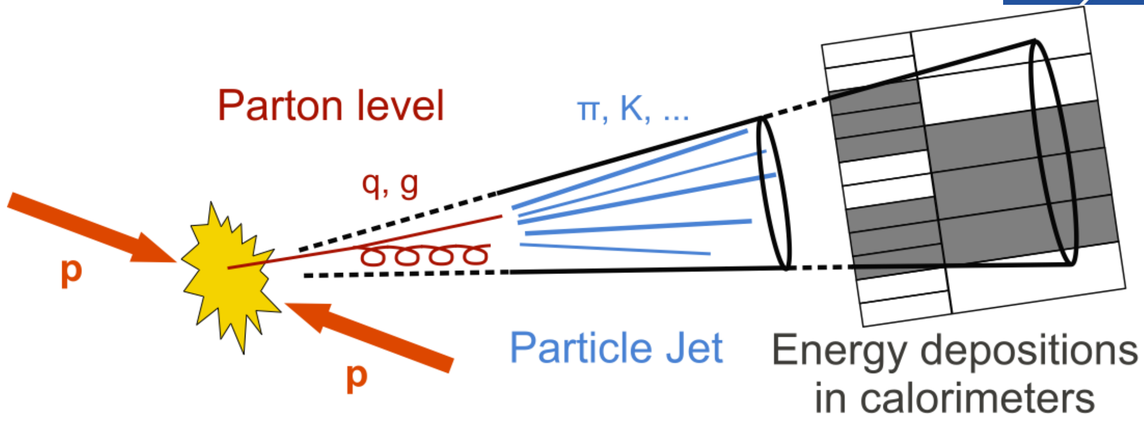


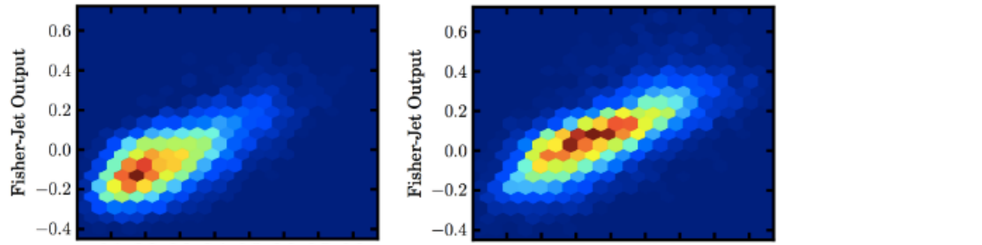
Feature Visualization of Convnet trained on ImageNet from [Zeiler & Fergus 2013]

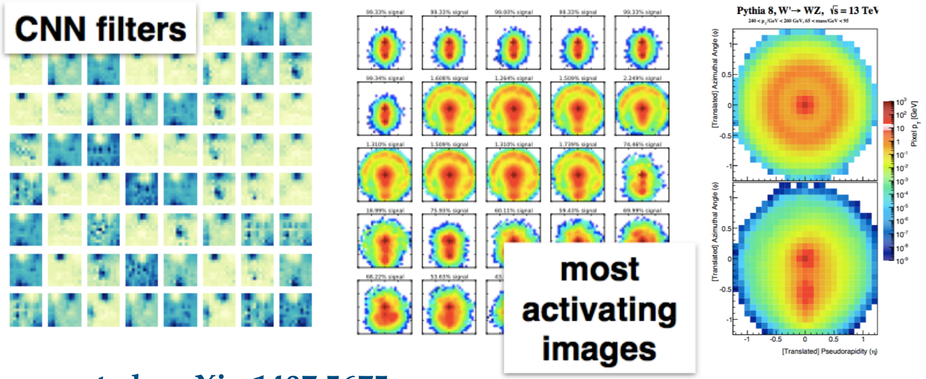
Jet Identification with CNN's

- **Applying computing vision techniques to jet identification tasks**
- Discriminate standard one-prong particles jets (q/g) from jets originating from overlap of sub-jets from boosted decay of heavy objects
  - **h/W/Z → qq or t → Wq → qqq**







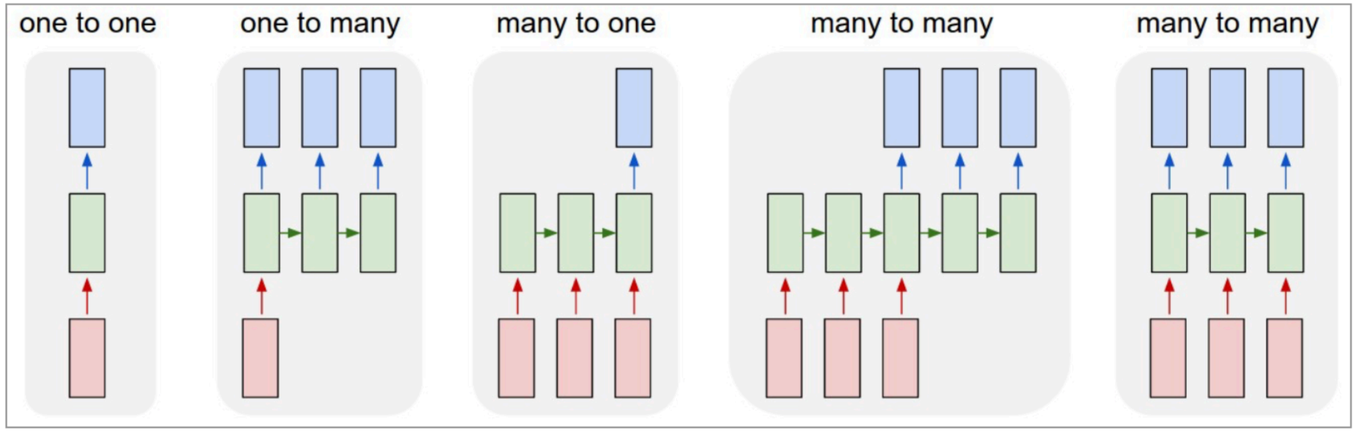


Cogan et al., arXiv:1407.5675

De Olivera et al., arXiv:1511.05190

Recurrent Neural Networks

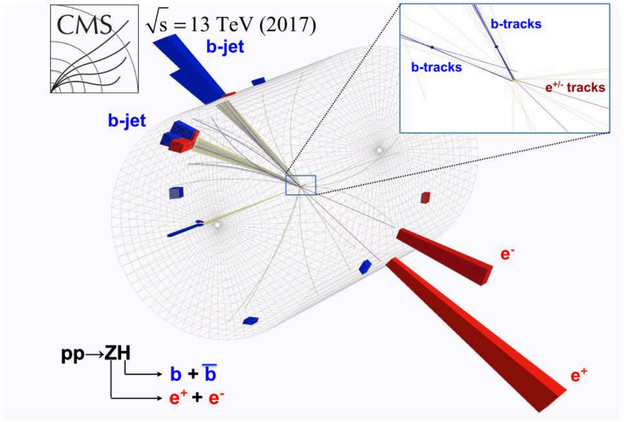
- What if we have variable length input data ?
- Or if input data is a sequence:
  - time series data
    (e.g. financial data)
  - audio or video data
  - natural language text
  - sequence of objects
    (e.g. particles sequence produced in a collision)

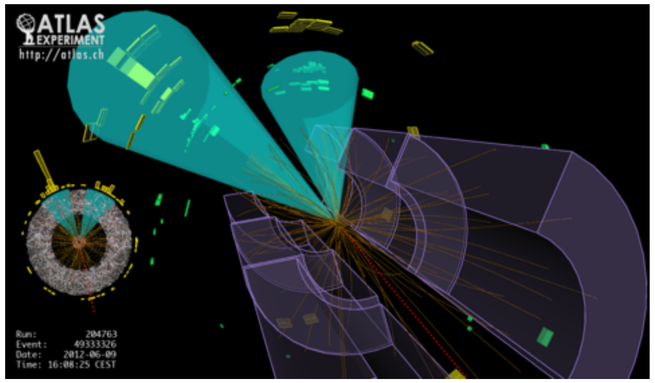


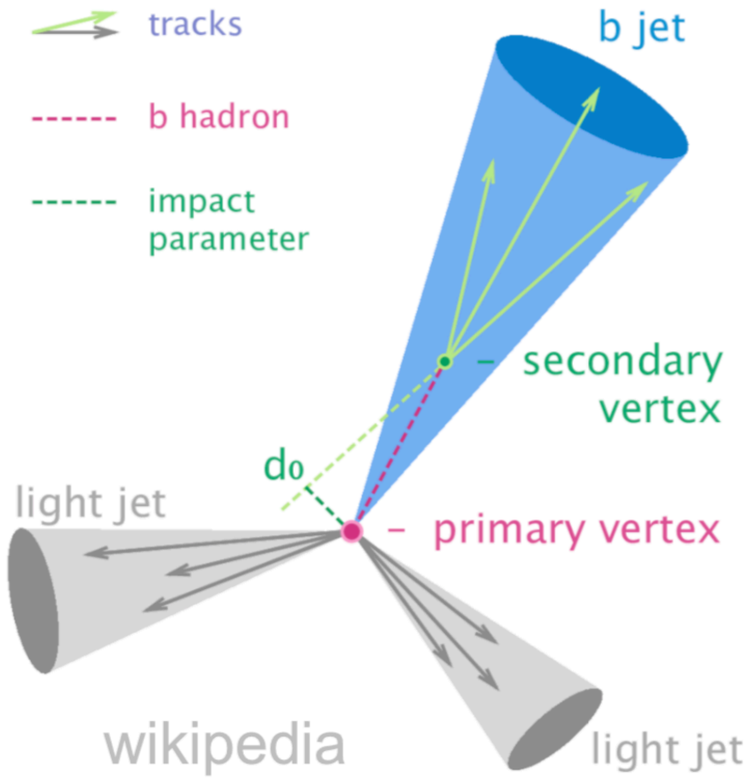
- Recurrent neural networks are special type of networks designed for these type of data

Flavour Jet tagging in HEP

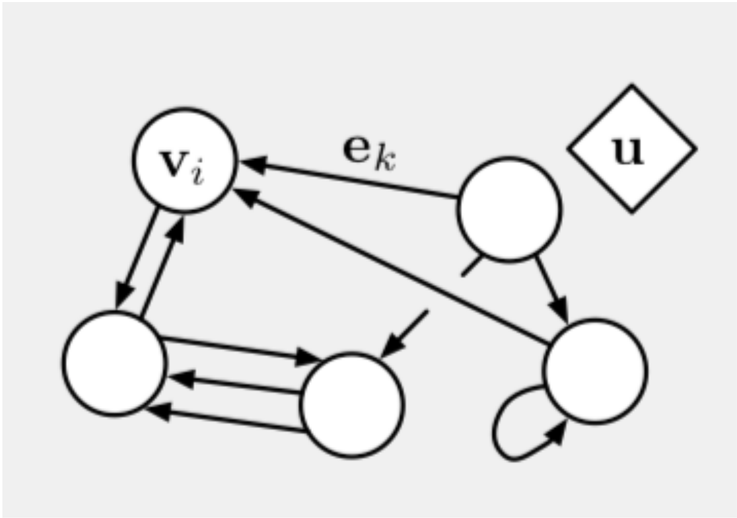
- Particle reconstruction using natural language processing algorithms
  - particle as words in a sentence
  - physics theory (e.g. QCD) as gramn
- Example: heavy flavour jet tagging in ATLAS and CMS







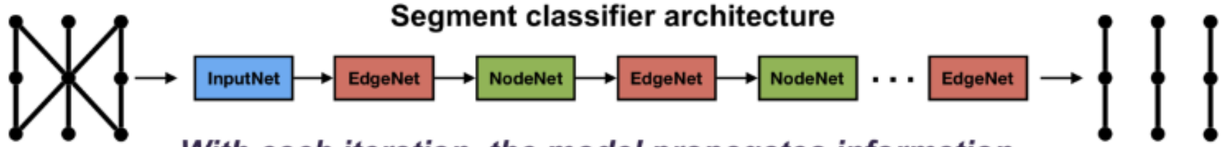
Graph Neural Networks



- Graph: {Edges, Nodes}
- Each of edges/nodes carry attributes
- At each step: information flow along direction of edges from node to node
- Network learns its optimal structure from the data
- In HEP are currently investigated for :
  - tracking: find optimal connections between reconstructed detector hits
  - irregular geometries: optimal mapping for reconstructed particle detector images

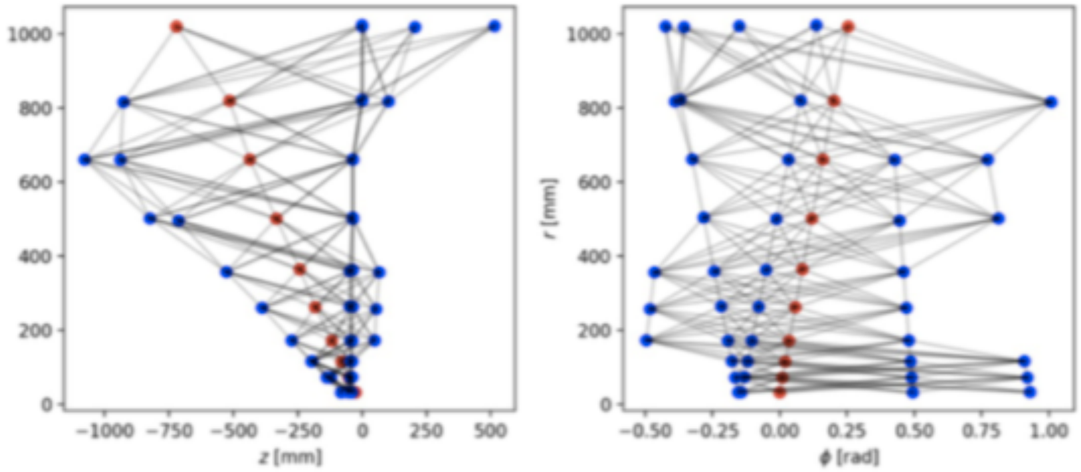
Applications of Graph Net's

Tracking of particles

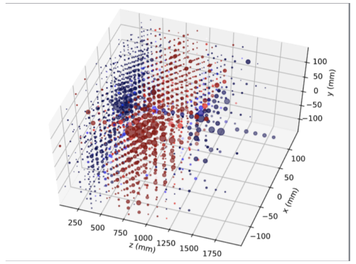


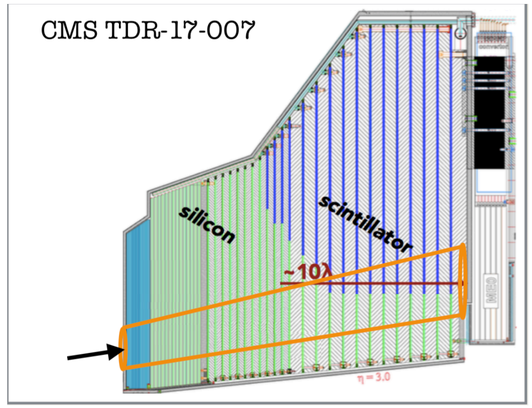
With each iteration, the model propagates information through the graph, strengthens important connections, and weakens useless ones.

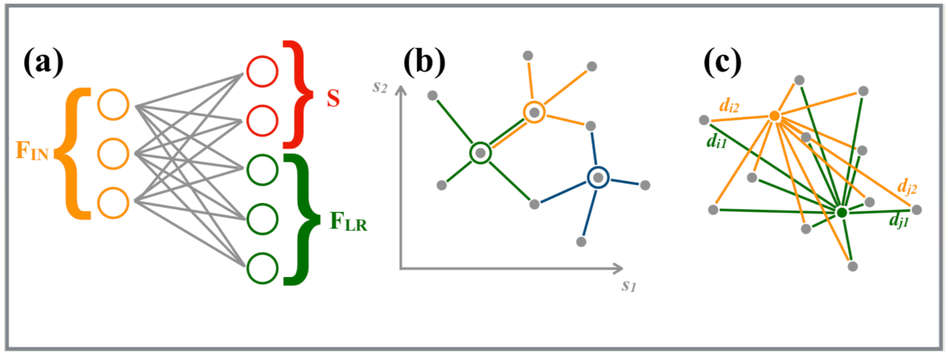
- Unseeded hit-pair classification
- Model predicts the probability that a hit-pair is valid



Learn representation of irregular particle detector geometries



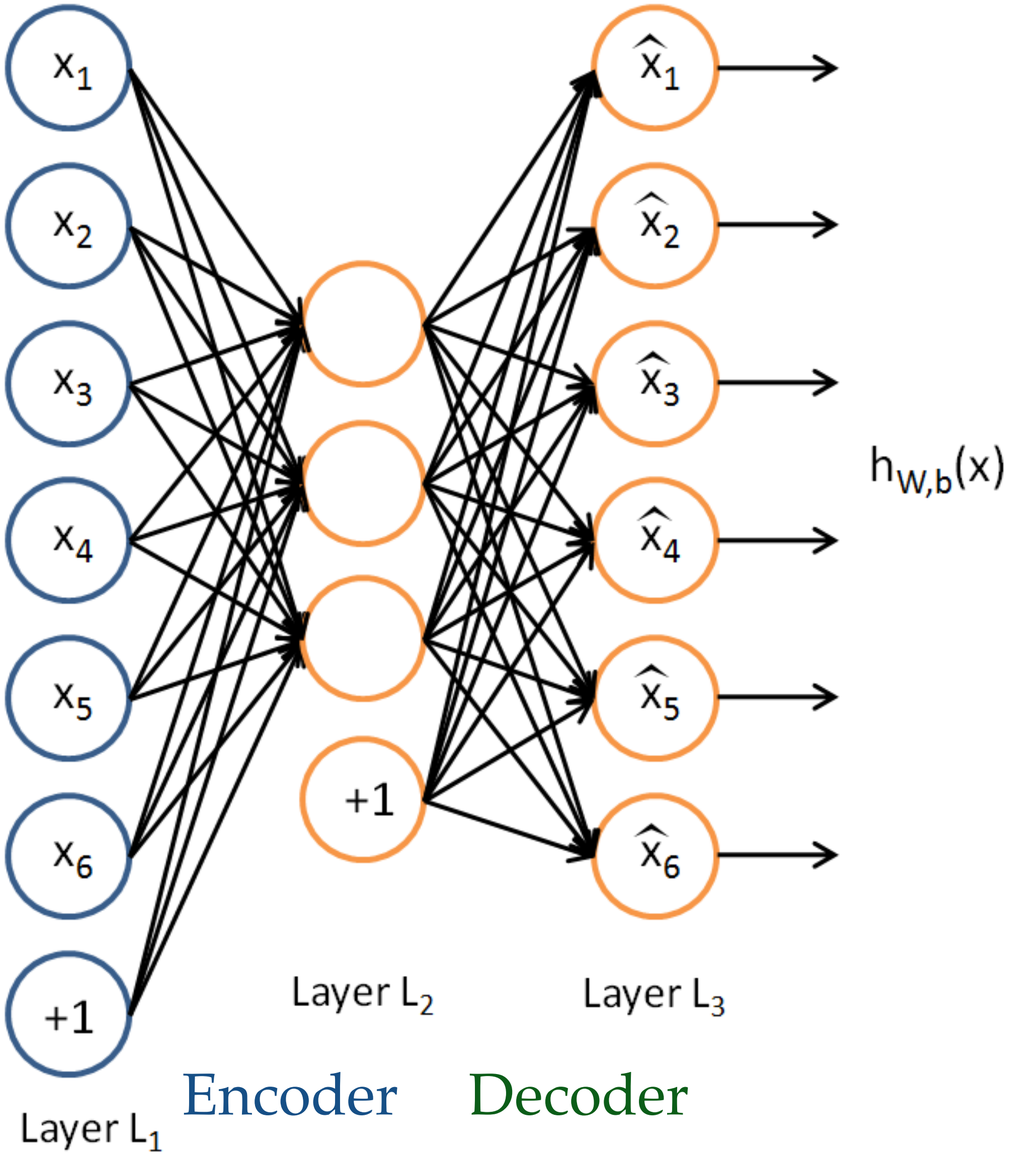




S.R.Quasim at al., arXiv:1902.07987

Deep Autoencoder

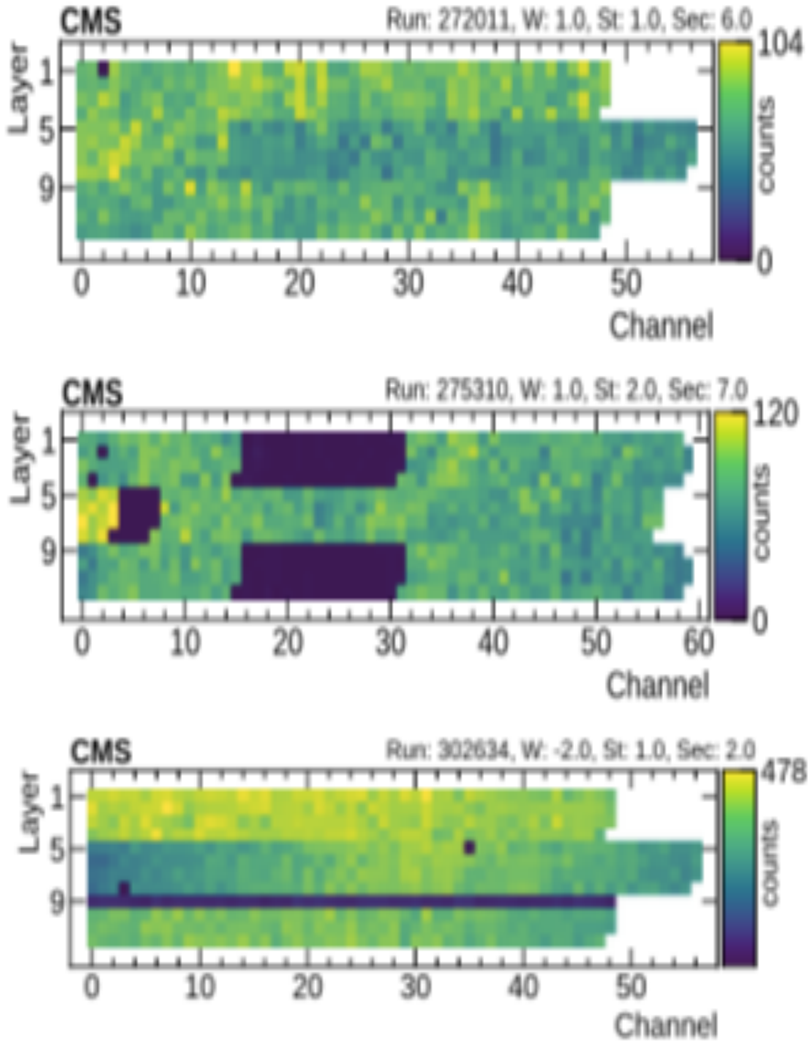
- An unsupervised neural network
- Trained by setting the target values y_i equal to the inputs x_i
- Can be used for:
  - **dimensionality reduction**
  - **anomaly detection**
  - and as a generator: **variational auto-encoder**

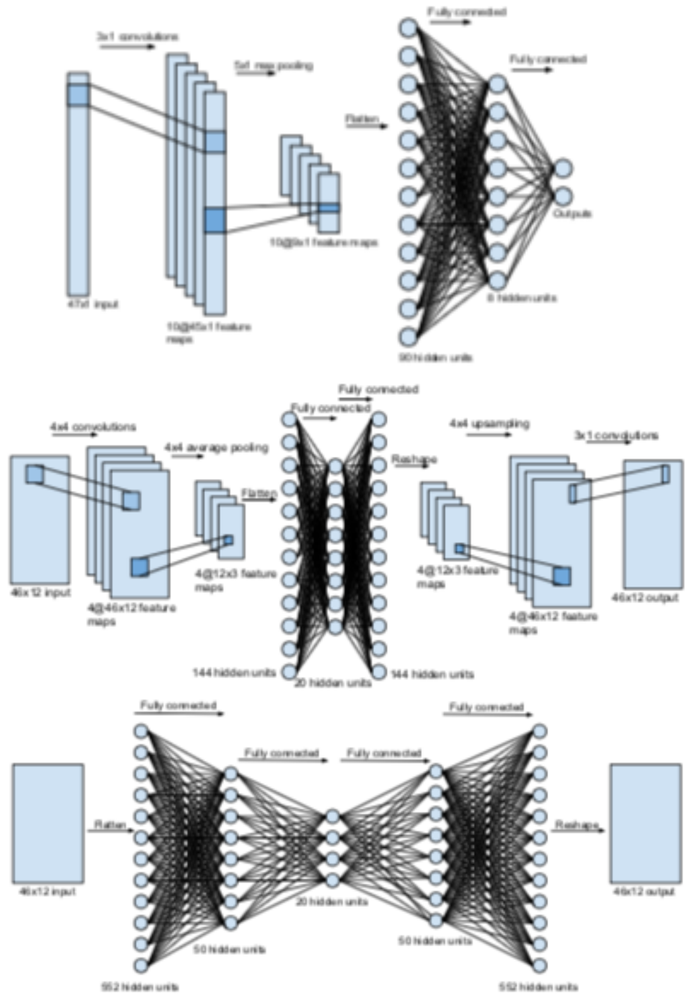


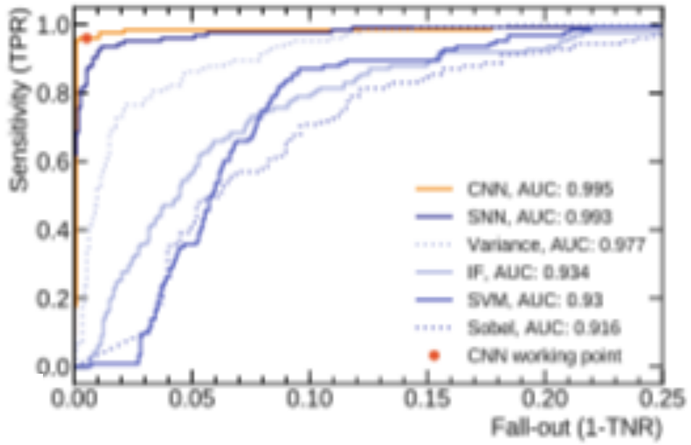
Data Quality Monitoring

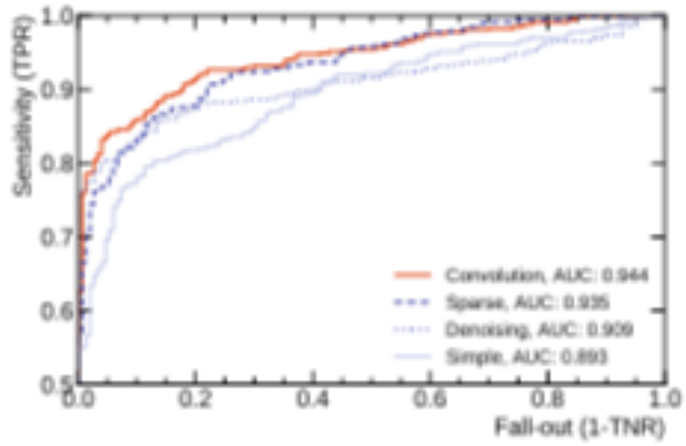
- Unsupervised ML used to spot anomalies

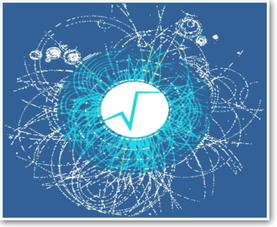
[Pol *et al.*, 2018, arXiv:1808.00911]







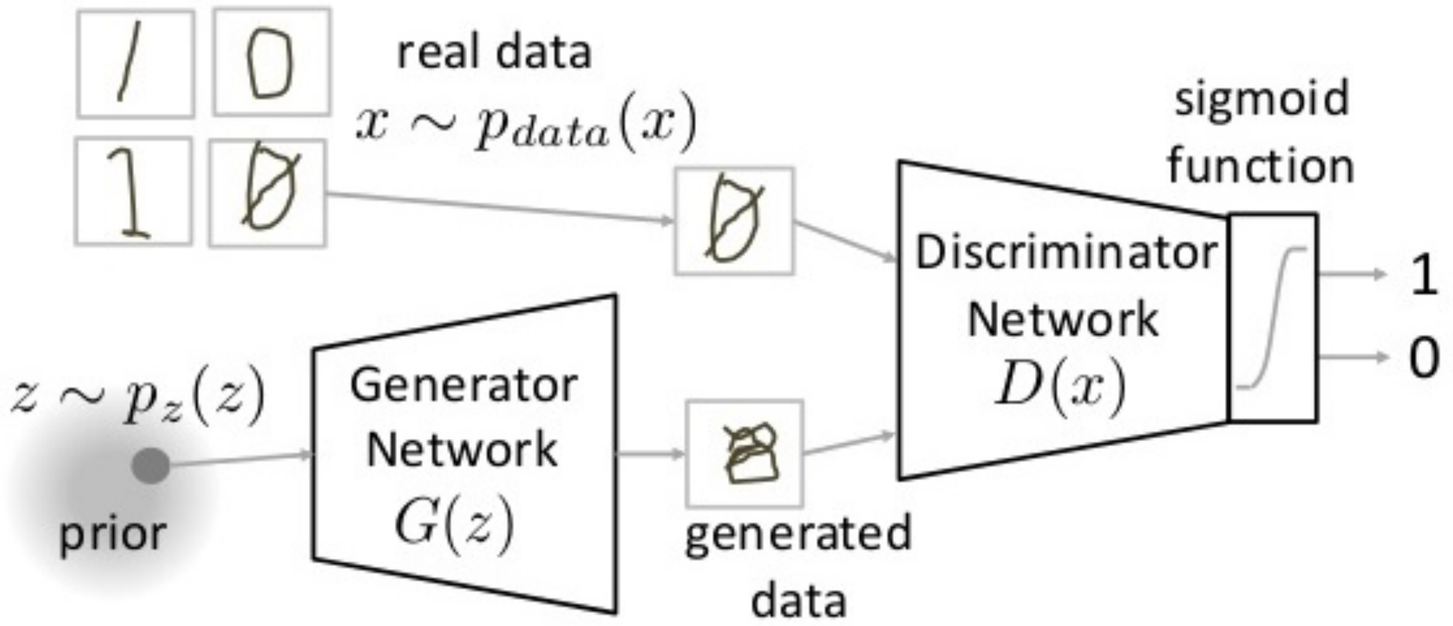




Generative Models

- Classification and regression predict a target Y given the input data X
- **What if we want instead to predict the density p(X) ?**
  - useful for
    - data sampling (simulation)
    - data augmenting
    - outlier detection
    - ….
- Difficult to model data distribution if data highly dimensional
- **Deep Generative models**
  - Variational Auto-Encoder (VAE)
  - Generative Adversarial Networks (GAN)

GAN: Generative Adversarial Network



- **Generator network**:
  - output data from a random input **G(z)**
- **Discriminator network**:
  - discriminate the generated data from real ones
  - output probability **D(x)** that data are from real input

GAN Progress for Image Generation

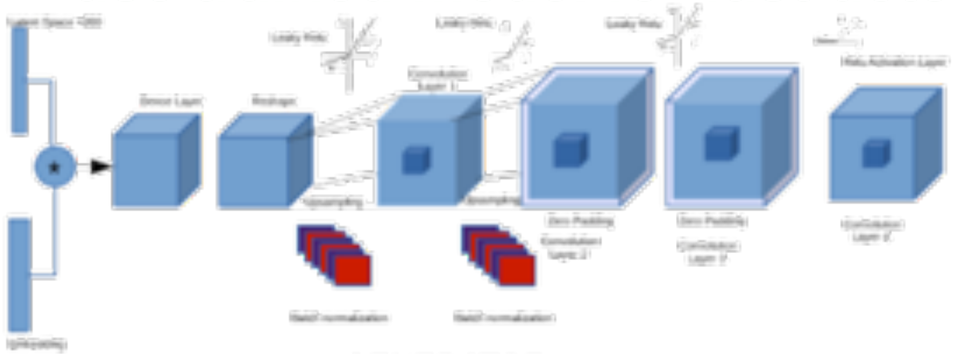
[M. Kagan]

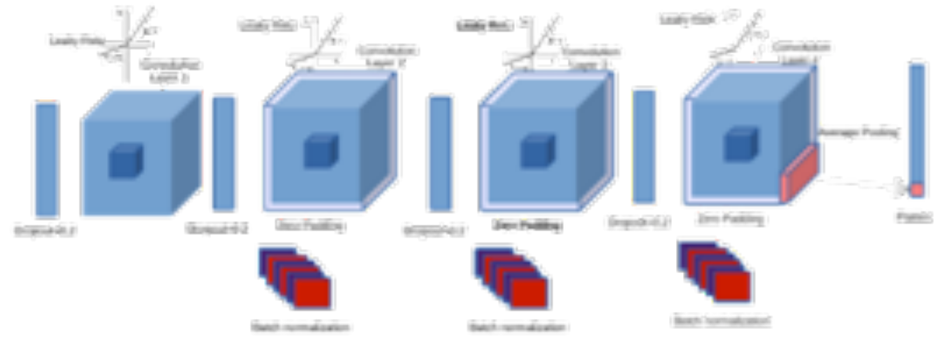
- **Large progress and developments in recent years**
  - still challenging to have GAN converging to good solution

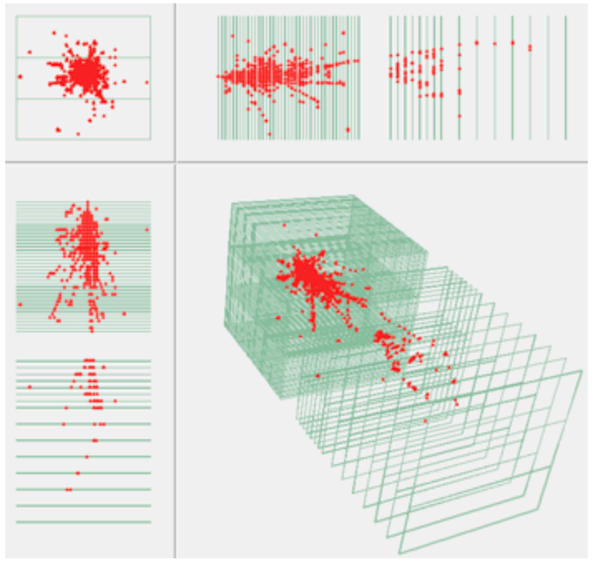
Example: 3d GAN for Calorimeter Images

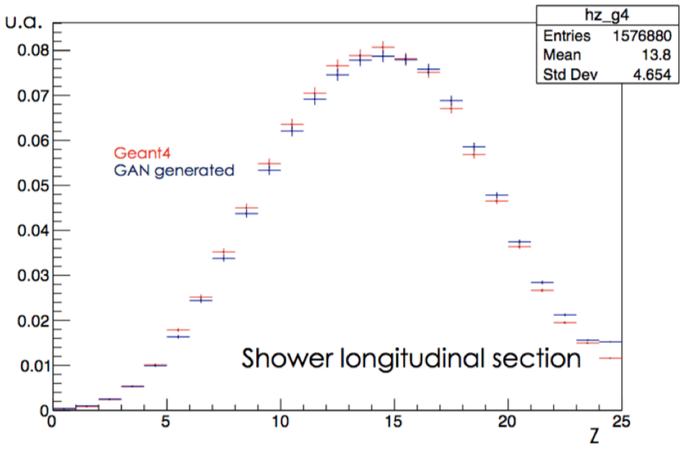
GAN as possible fast simulations of calorimetric images

- Discriminator and generator network built using convolutional layers
- Train with full simulated (Geant4) images
- **use trained model for fast detector simulation**









[S. Vallecorsa]

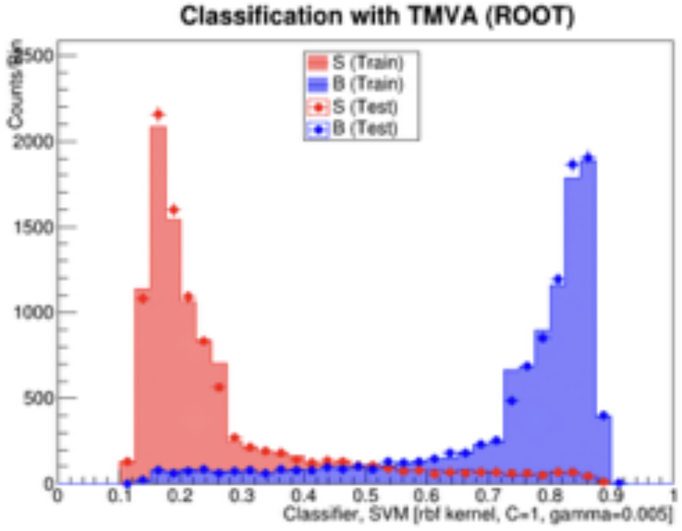
Deep Learning Software

- **Many open source tools developed by large data science communities written in Python**
- Very rich functionality allowing to build all possible deep learning models
  - use power of Python language to have a very rich and convenient API
  - support efficient back-end deployment for both GPU and CPU
- Most popular tools for build and train Deep Learning models:
  - **TensorFlow**
  - **Keras** (only an high level tool, use Tensorflow as backend)
  - **PyTorch**
- Specific HEP tool: **ROOT/TMVA**

Machine Learning: TMVA

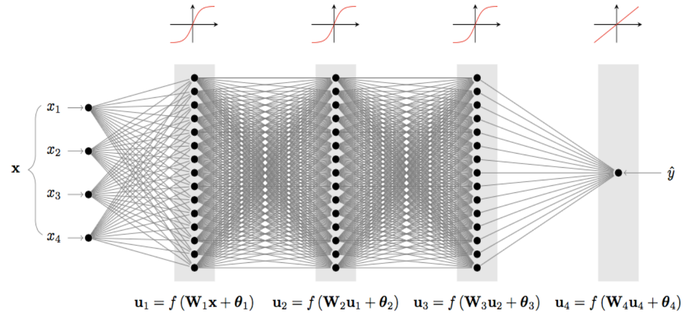
TMVA : Toolkit for Multi-Variate data Analysis in ROOT

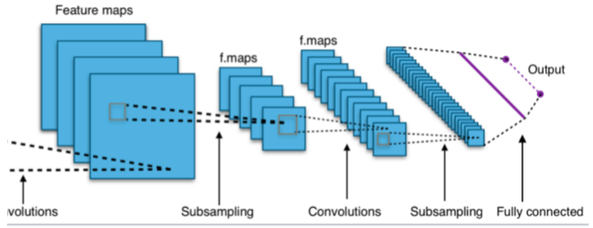
- provides several built-in ML methods for HEP usage including:
  - **Boosted Decision Trees**
  - **Deep Neural Networks**
- and interfaces to external ML tools packages
  - scikit-learn, Keras (Tensorflow), R

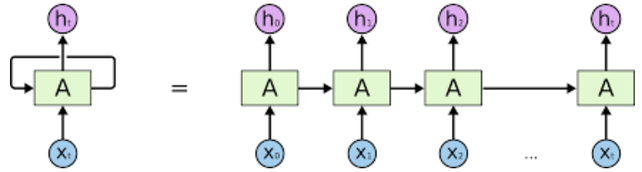


Deep Learning in TMVA

- Deep Learning Library in ROOT/TMVA with support for
  - Dense (fully connected) layers
  - Convolutional layers
  - Recurrent layers





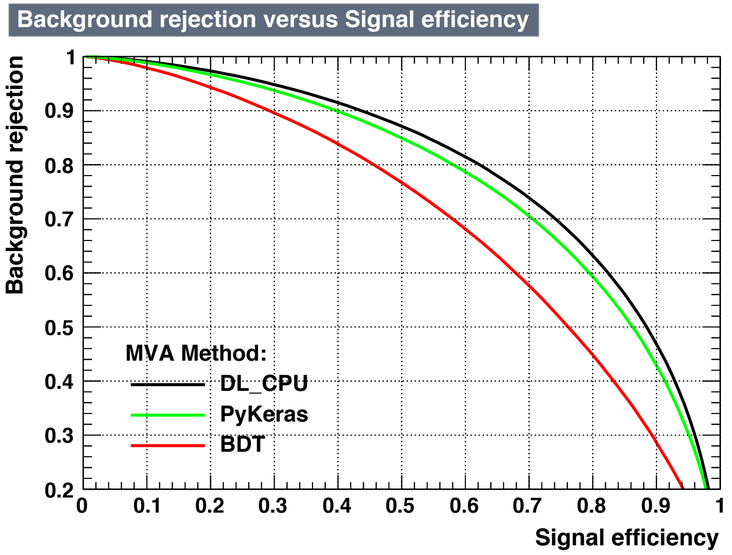


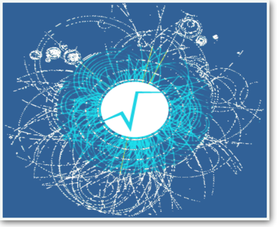
- Parallel implementation for both CPU and GPU
- Very efficient for training
- Focus especially for low latency inference of models

TMVA Interfaces

External tools are available as additional methods in TMVA and they can be trained and evaluated as any other internal ones.

- **RMVA**: **Interface to Machine Learning methods in R**
  - c50, xgboost, etc..
- **PYMVA**: **Interface to Python ML packages**
  - **scikit-learn**
    - with RandomForest,
      Gradient Tree Boost, Ada Boost
  - **Keras** (Tensorflow)
    - support external model definition (in Python)
    - training and evaluation within ROOT





ROOT/TMVA outside CERN

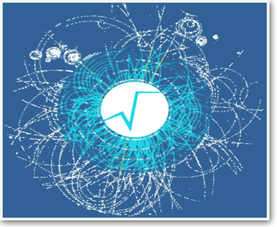
- Large interest for Machine Learning algorithms in industry
- Interest for ROOT/TMVA tools from different domains
  - e.g. finance, medical data, optimisation of vaccine production (Sanofi Pasteur)
- **ROOT provides ML tools with other powerful data analysis capabilities**
  - visualization, statistical modeling
  - powerful I/O system for storage and filtering of data
  - C++/Python and Web interfaces (Jupyter)

Experience with Sanofi

- Training course of Machine Learning techniques and ROOT/TMVA to Sanofi-Pasteur
- Focus on techniques to improve vaccine production
  - data sets with large amount of variables
  - difficult to apply conventional methods

- **Interest in learning new methods** (as those provided in ROOT/TMVA) and in **applying them to their data**

More information in this article

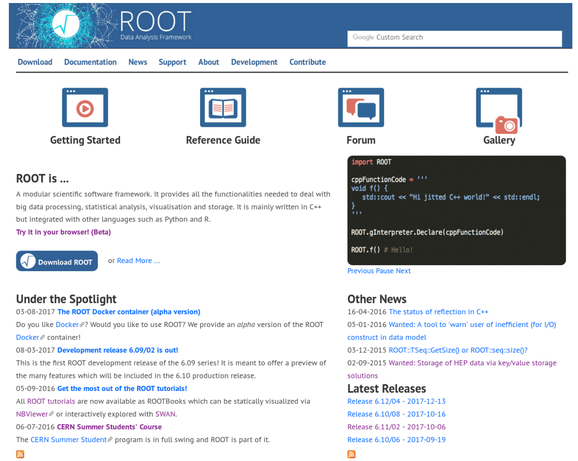


Summary

- **Deep Learning is a powerful set of AI tools**
  - Can be used in many different domains in data science
    - including new high energy physics applications
    - optimal usage of data collected of LHC (better sensitivity to new discovery)
  - Very exciting field with lots of new developments and applications
- ROOT /TMVA provides Machine Learning tools for data analysis
  - Long term support and stability
  - Excellent performance for training and inference
  - Large user community
  - **It is free (an open source project)**

ROOT/TMVA

- Web page: https://root.cern
  - TMVA: https://root.cern/tmva
- Forum: https://root-forum.cern.ch
- github: https://github.com/root-project
- @root-project
- https://www.linkedin.com/groups/1826455
- root-dev@cern.ch



Some Useful References

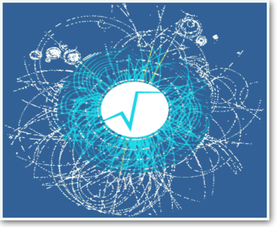
Useful materials for ML at CERN and HEP can be found in:

- *M. Kagan*: CERN Academic Training Lectures (2017)
- 3rd LPCC Inter-experimental Machine Learning Workshop (2019)
- Machine Learning Session at Phystat-nu 2019

Review Papers for Machine Learning applications in High Energy Physics

- A. Radovic *et al.* Machine learning at the energy and intensity frontiers of particle physics (2018) (https://doi.org/10.1038/s41586-018-0361-2)
- D. Guest et al. Deep Learning and Its Application to LHC Physics (2018), (https://doi.org/10.1146/annurev-nucl-101917-021019)

Additional Material

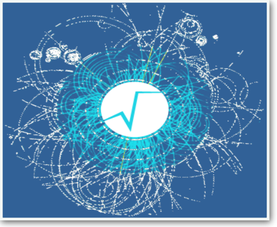


ROOT in a Nutshell

- ROOT is a software framework with building blocks for:
  - **Data processing**
  - **Data analysis**
  - **Data visualisation**
  - **Data storage**

An Open Source Project
We are on github
github.com/root-project
All contributions are warmly welcome!

- ROOT is mainly written in C++ (C++11/17 standard)
  - Bindings for Python available as well
- Adopted in High Energy Physics and other sciences and also industry
  - more than 1 Exabyte of data in ROOT format
  - Data analysis (machine learning), parameters estimations and discovery significances (e.g. the Higgs)
  - Thousands of ROOT plots in scientific publications



ROOT I/O

- ROOT offers the possibility to write C++ objects into files
  - This is impossible with C++ alone
  - Used by the LHC detectors to write several petabytes per year
  - seamless C++ integration: unique feature of ROOT
- Achieved with serialization of the objects using the reflection capabilities, ultimately provided by the interpreter
  - Raw and column-wise streaming
- As simple as this for ROOT objects: one simple method -

```
file->WriteObject(pObj, "name");
```

I/O Feature Comparison

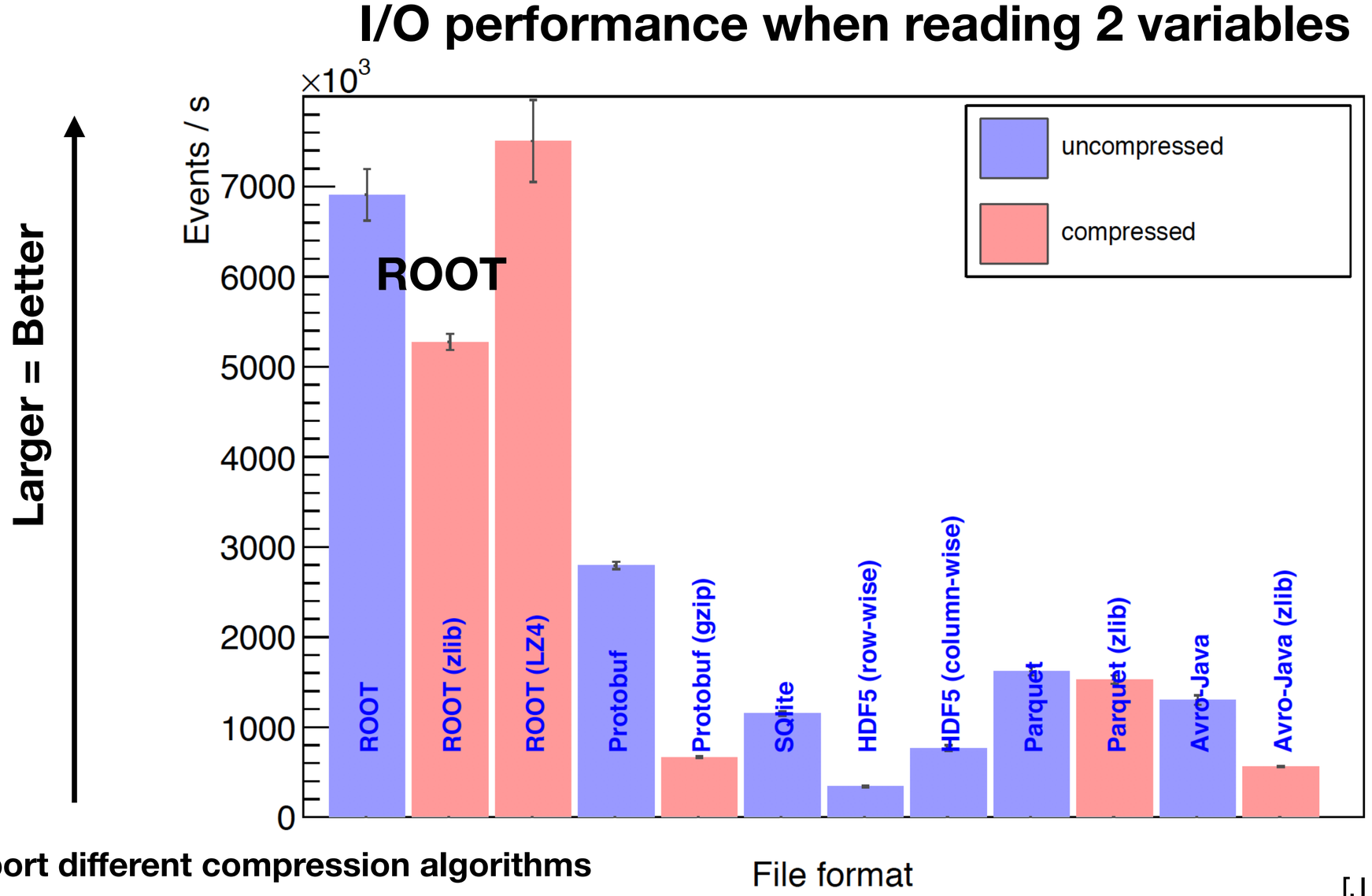
	ROOT	PB	SQlite	HDF5	Parquet	Avro
Well-defined encoding	✓	✓	✓	✓	✓	✓
C/C++ Library	✓	✓	✓	✓	✓	✓
Self-describing	✓	↯	✓	✓	✓	✓
Nested types	✓	✓	?	?	✓	✓
Columnar layout	✓	↯	↯	?	✓	↯
Compression	✓	✓	↯	?	✓	✓
Schema evolution	✓	↯	✓	↯	?	?

[J. Blomer, ACAT 2017]

✓ = supported
↯ = unsupported
? = difficult / unclear

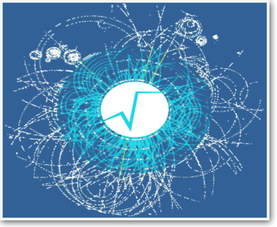
- Unique capabilities of ROOT required for HEP data

I/O Performance Comparison



Support different compression algorithms

[J. Blomer, ACAT 2017]

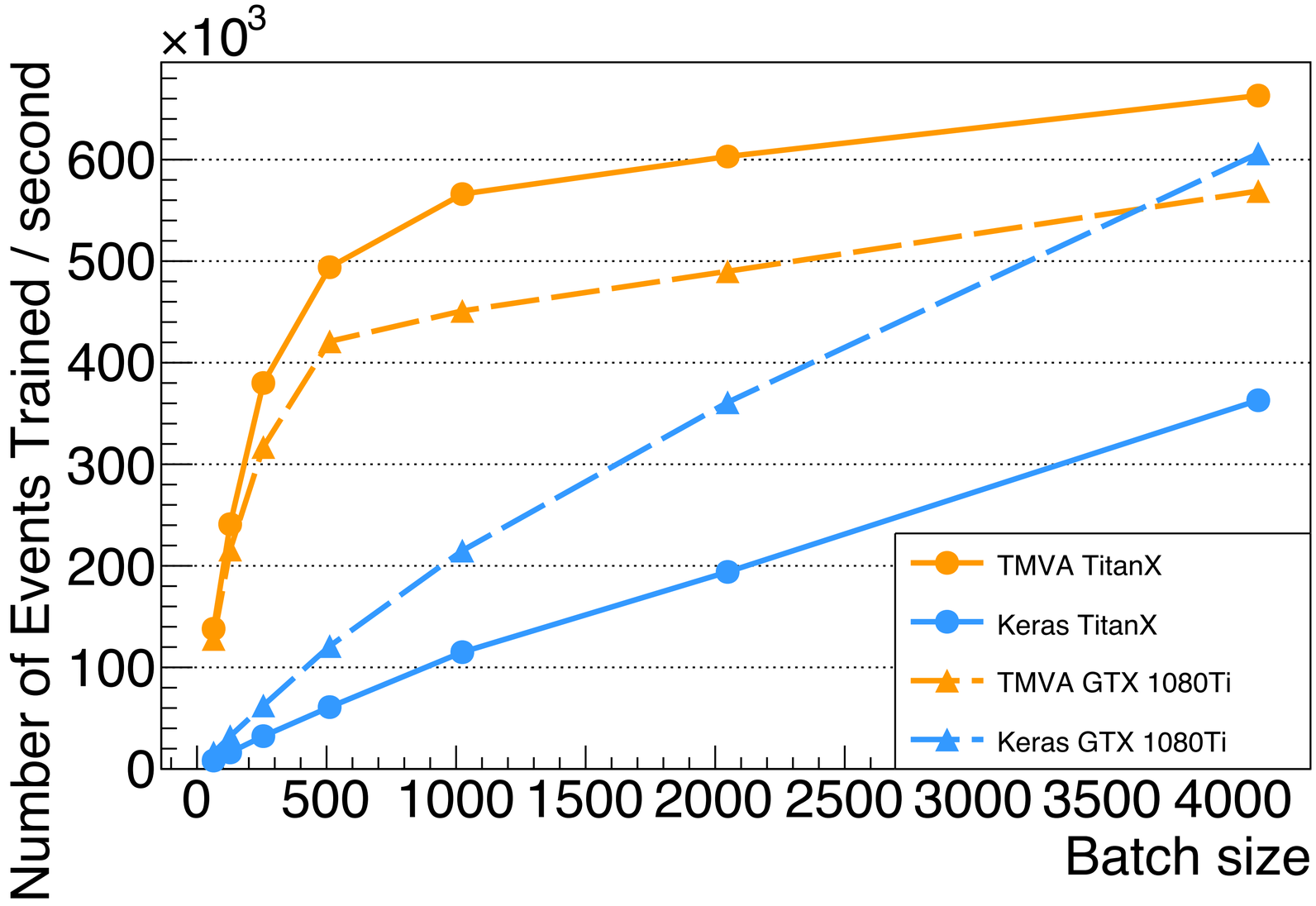


TMVA Methods

The major algorithms available in TMVA are :

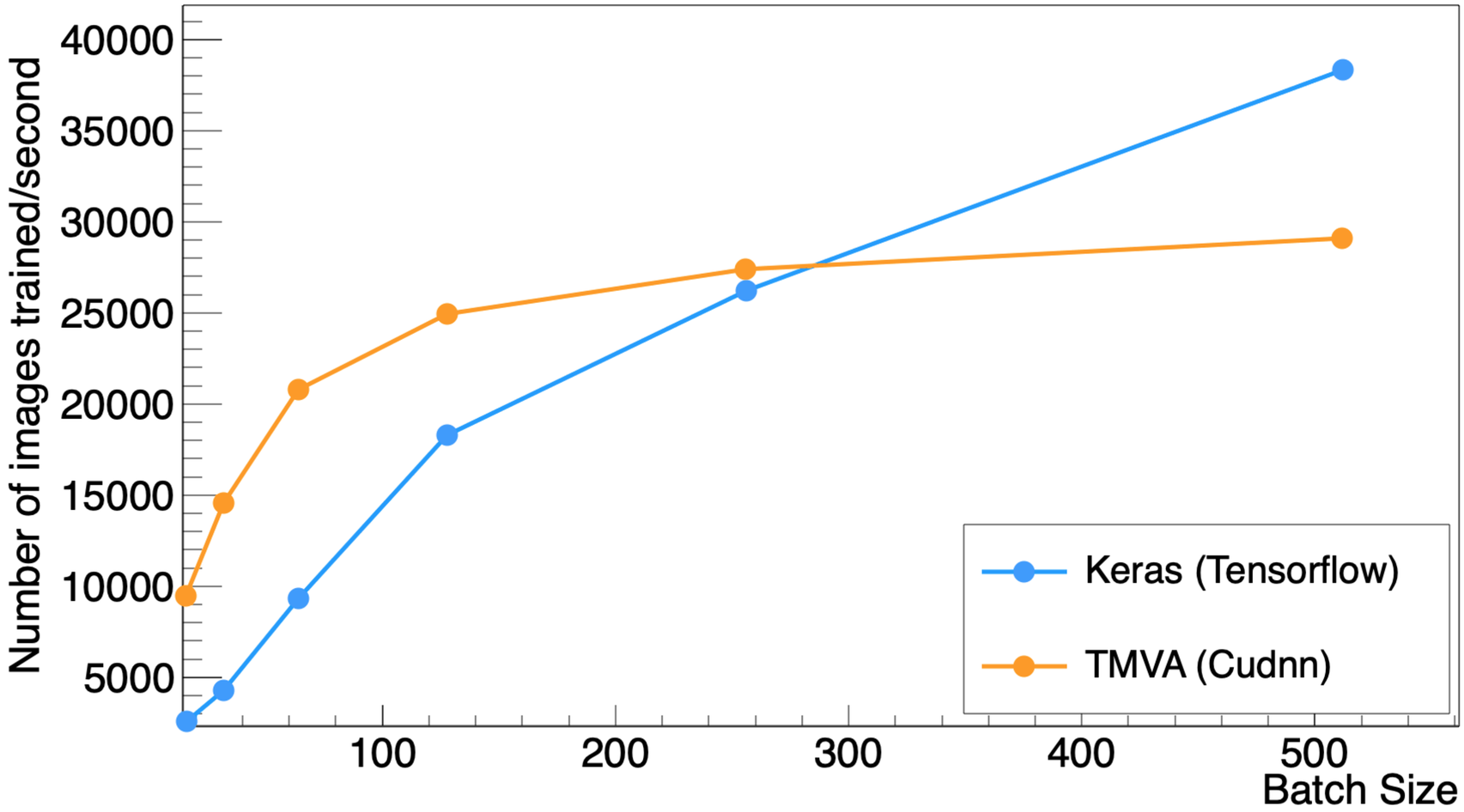
- Projective likelihood estimation (PDE approach)
- Multidimensional probability density estimation (PDE - range-search approach)
- Multidimensional k-nearest neighbour classifier
- Linear discriminant analysis (H-Matrix and Fisher discriminants)
- Boosted/Bagged decision trees
- Predictive learning via rule ensembles (RuleFit)
- Support Vector Machine (SVM)
- Artificial neural networks (various implementations for shallow networks)
- **Deep Learning**
  - with convolutional and recurrent networks
  - working on CPU and GPU

DNN Training Performance



CNN Training Performance

Training performance on GPU (RTX 2070), image size = 32x32



Evaluation Performance

- **Single event evaluation time** for 5 layer network
  - For time critical applications — e.g. on-line reconstruction
  - Fast! 1.5 times speedup over specialised libraries like LWTNN when using optimised Blas library exploiting vectorisation
- For batched evaluation, same story as training