ROOT & TMVA
Data Analysis and Machine Learning

Lorenzo Moneta
(CERN)

ROOT in a Nutshell

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

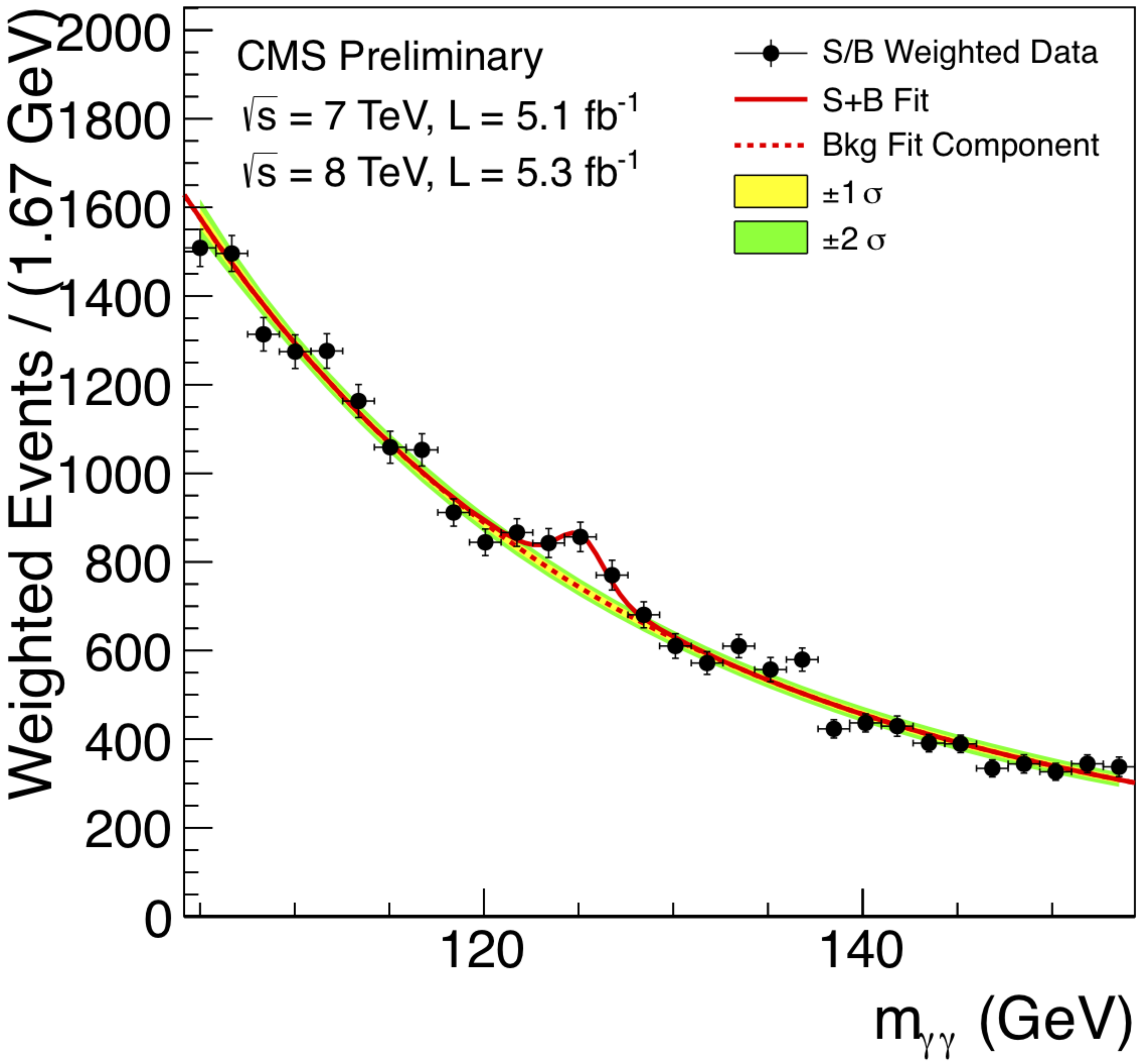
An Open Source Project
We are on github
github.com/root-project
All contributions are warmly welcome!

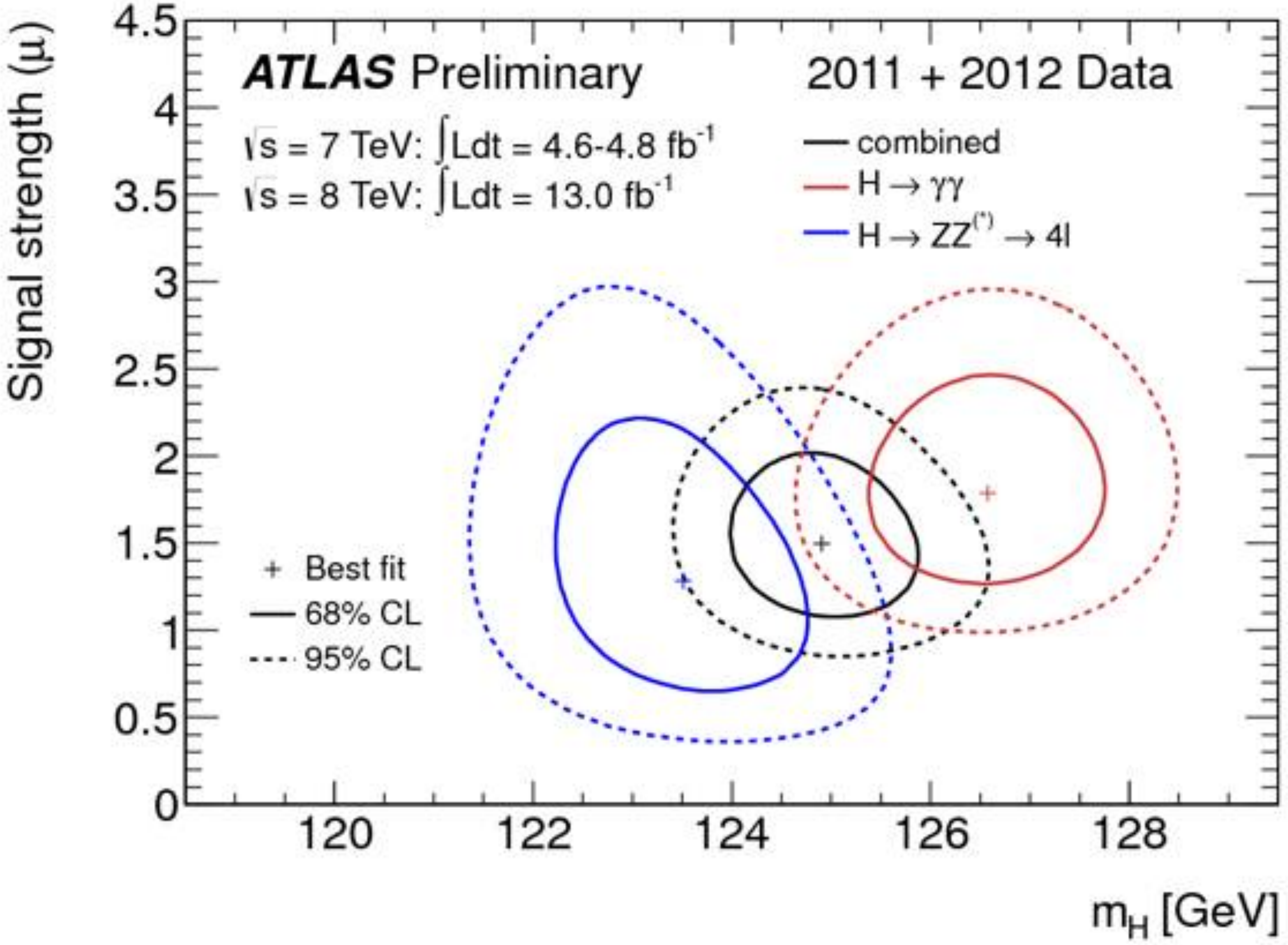
ROOT Components

ROOT can be seen as a collection of building blocks for various activities, like:

- **Data analysis**: histograms, graphs, functions
- **I/O**: row-wise, column-wise storage of any C++ object
- **Statistical tools**: rich modeling tools and statistical inference
- **Math**: math functions, linear algebra and minimisation algorithms
- **C++ interpretation**: full language compliance
- **Multivariate Analysis (TMVA)**: e.g. Boosted decision trees, Neural networks (including deep learning)
- **Advanced graphics** (2D, 3D, event display)
- **Declarative Analysis**: Data Frame for event filtering and selection
- And more: HTTP serving, JavaScript visualisation

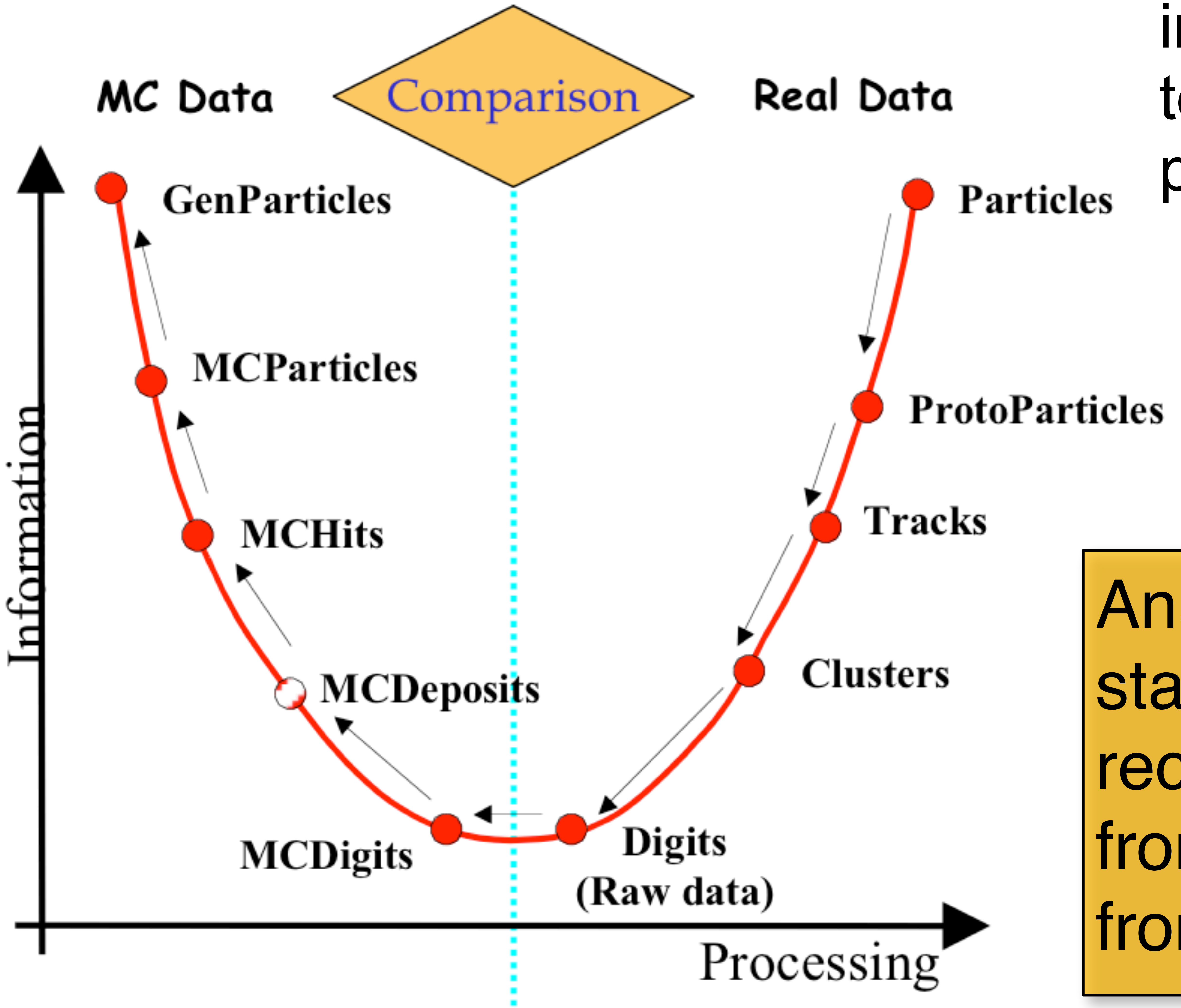
What can you do with ROOT?





HEP Data Interpretation

Monte Carlo Simulation follows the evolution of physics processes from collision to digital signals



Reconstruction "goes back in time" from digital signals to the original particles produced in the collision

Analysis compares (at statistical level) reconstructed events from real data with those from simulation

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

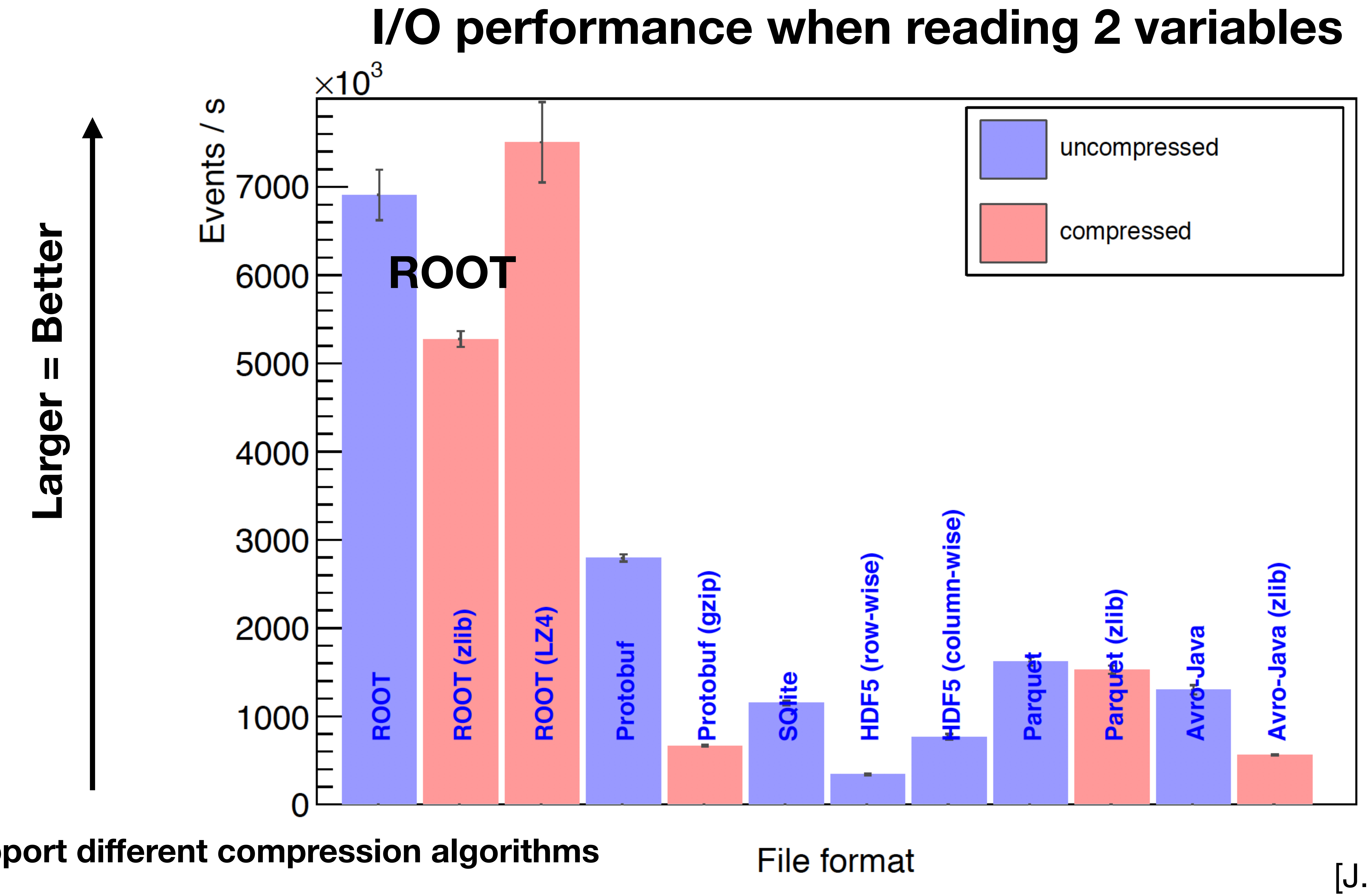
	ROOT	PB	SQlite	HDF5	Parquet	Avro
Well-defined encoding	✓	✓	✓	✓	✓	✓
C/C++ Library	✓	✓	✓	✓	✓	✓
Self-describing	✓	↯	✓	✓	✓	✓
Nested types	✓	✓	?	?	✓	✓
Columnar layout	✓	↯	↯	?	✓	↯
Compression	✓	✓	↯	?	✓	✓
Schema evolution	✓	↯	✓	↯	?	?

[J. Blomer, ACAT 2017]

✓ = supported
↯ = unsupported
? = difficult / unclear

- Unique capabilities of ROOT required for HEP data

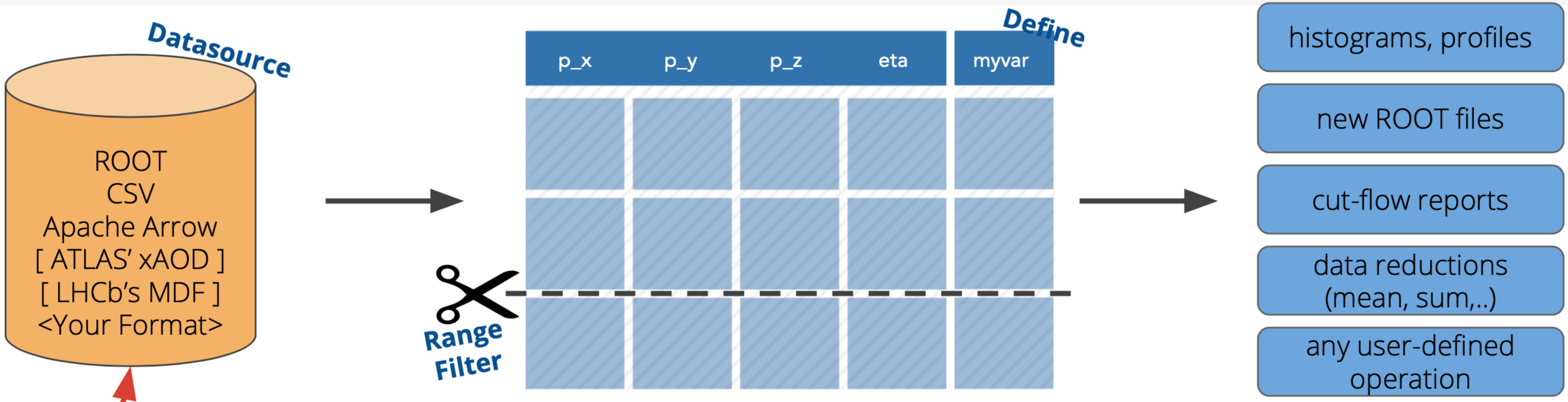
I/O Performance Comparison



Support different compression algorithms

[J. Blomer, ACAT 2017]

RDataFrame



Customisation point,
public interface!

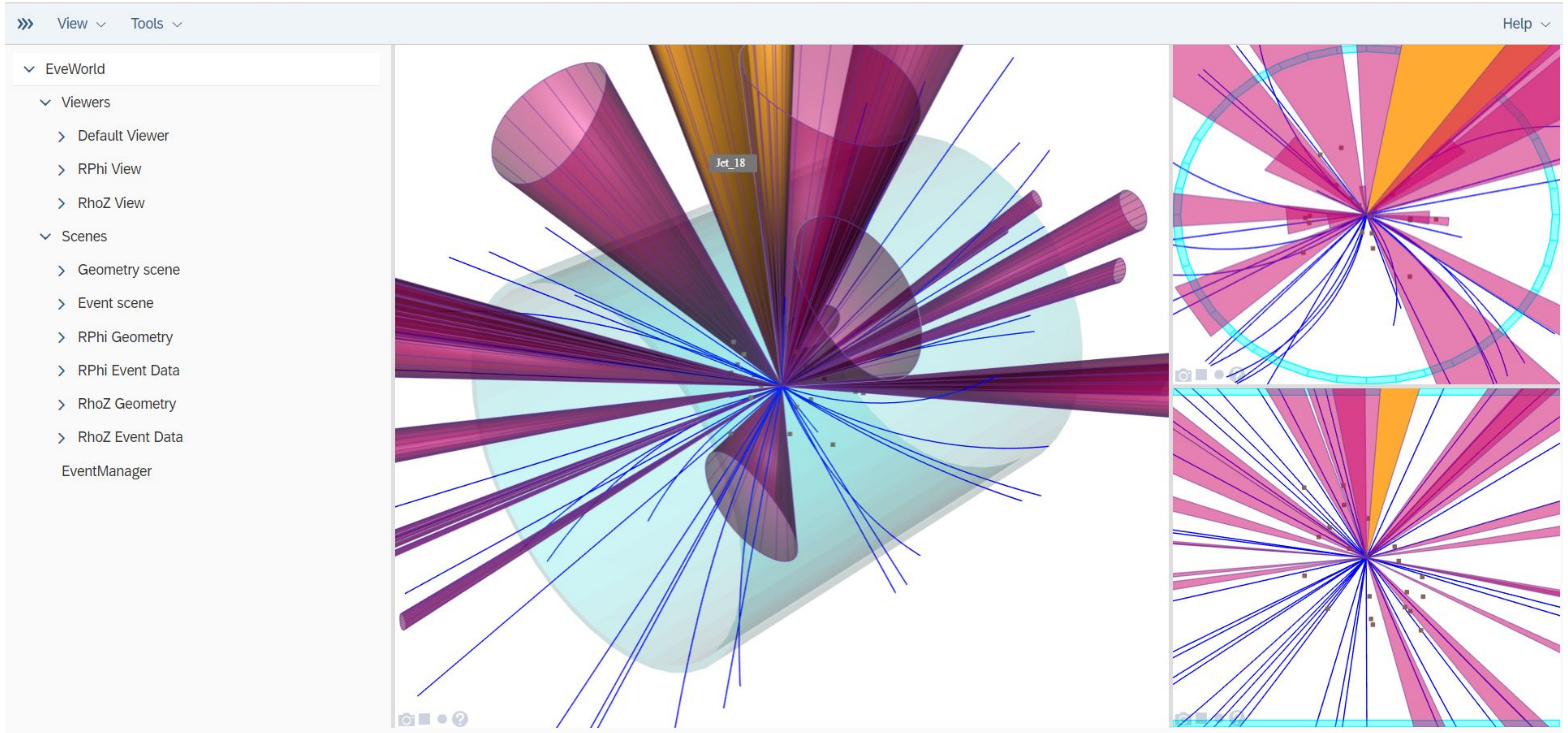
Goals:

- ➔ Be the **fastest** way to manipulate HEP data
- ➔ Be the **go-to ROOT analysis interface** from laptop to cluster
- ➔ Consistent interfaces in **Python and C++**
- ➔ Top notch documentation and examples

In production since last year (ROOT version 6.14)

New ROOT Graphics

- New Graphics of ROOT uses Web based technologies
  - able to run in the Web browser



WebEve

can run as
- a standalone application
- an existing browser
- embedded in other Web based GUI's (e.g. Jupiter notebooks)

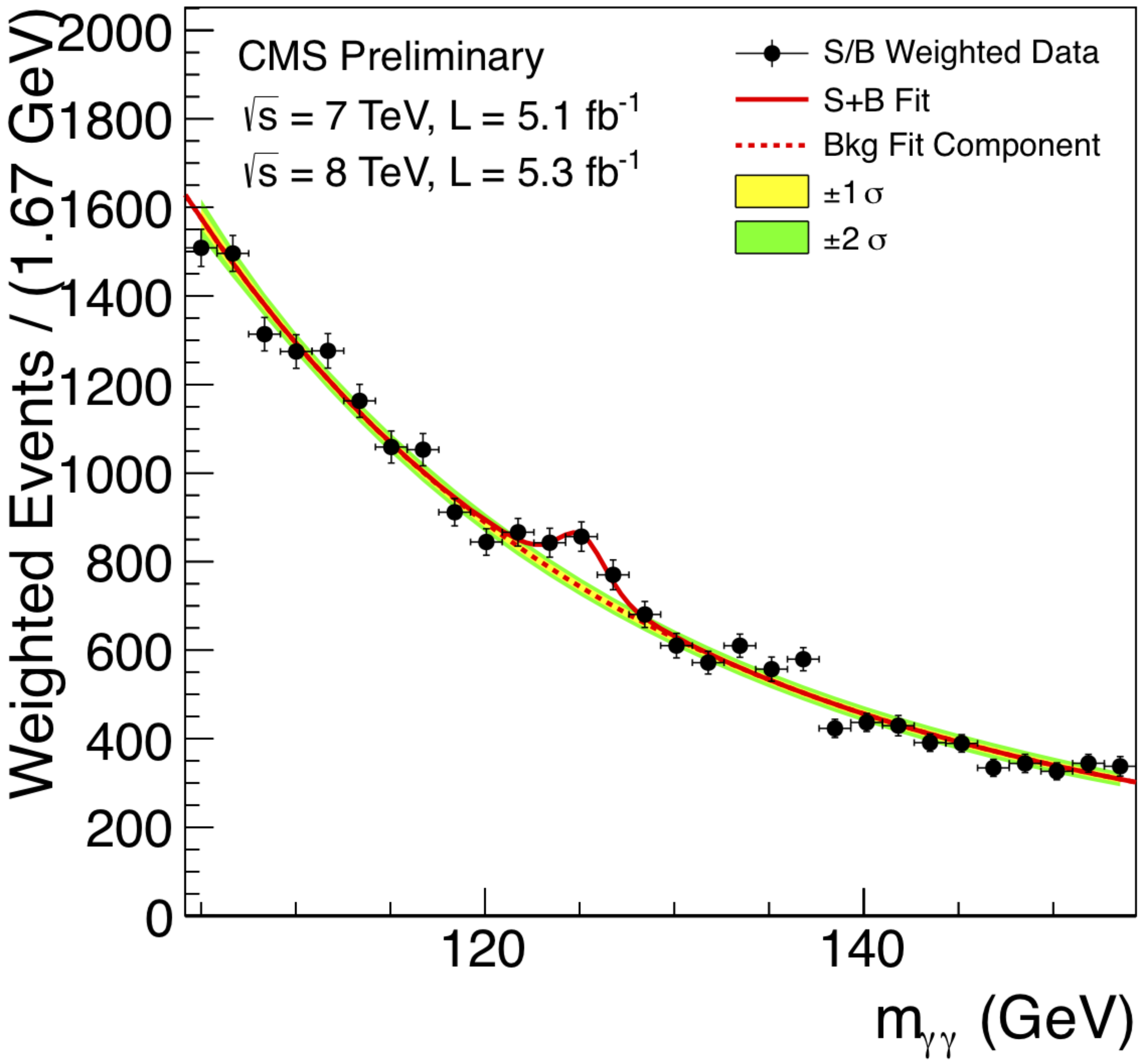
Modeling in Physics Analysis

- **Statistical modeling for physics parameter θ**
  - Estimate probability density functions $p(\mathbf{x} \mid \theta)$
    - typically using simulation (generative models)
  - compute likelihood function $p(\mathbf{x}_{obs} \mid \theta)$
  - parameter estimation (with uncertainties), hypothesis tests (frequentist)
  - Bayesian analysis to get $p(\theta \mid \mathbf{x}_{obs})$ using prior $p(\theta)$
- **Discriminative modeling (classification, regression)**
  - Model the $p(\mathbf{y} \mid \mathbf{x})$ using a training sample $\{\mathbf{x},\mathbf{y}\}$
    - e.g. $\mathbf{y}$ labels (signal events vs background events)
    - use typically simulated data for training
  - methods: neural networks, decision trees (boosted trees, random forest), etc..

Fitting and Data modeling

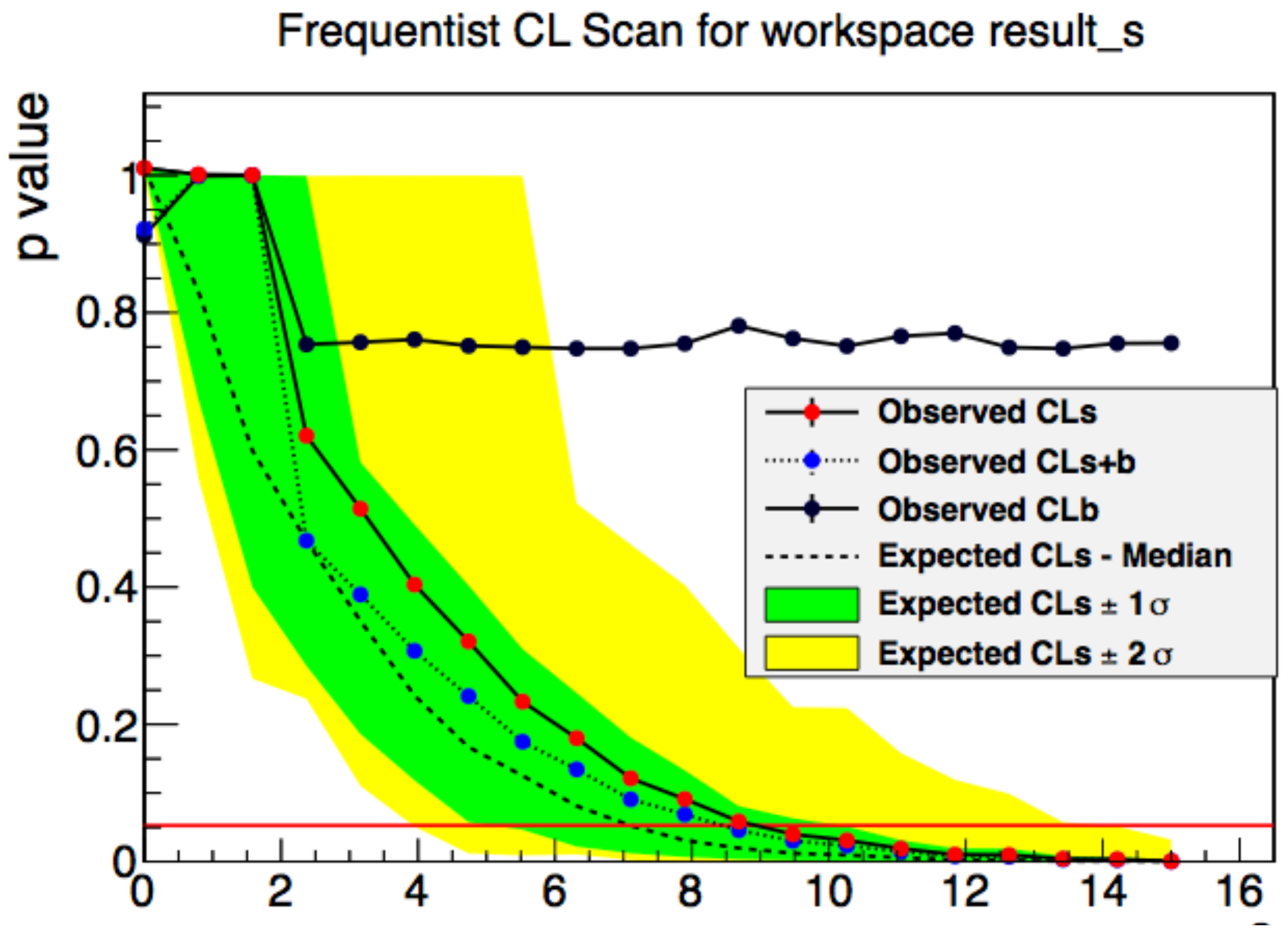
RooFit: ROOT Toolkit for Data Modeling and Fitting

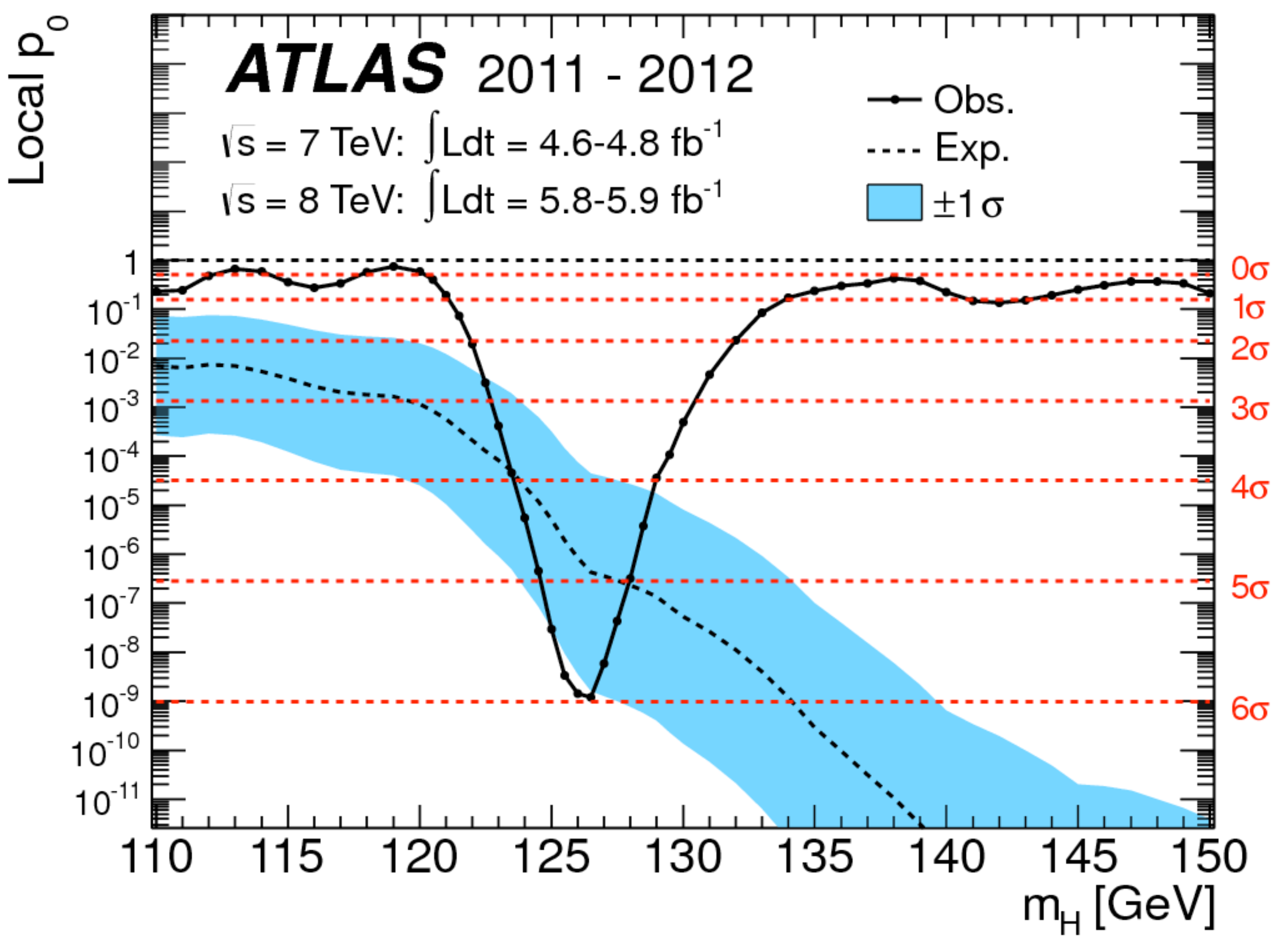
- functionality for building models: probability density functions (p.d.f.)
  - parametrize observed distributions **P(x;p)**
- capability for complex model building
  - e.g. composition, addition, convolution of distributions
- with functionality for
  - **maximum likelihood fitting** for parameter estimation
  - **bootstrapping** (simulate events from model distributions)
  - **visualisation**
  - **sharing and storing models**



Advance Statistics: RooStats

- Advanced statistical tool for HEP data analysis for :
  - confidence intervals estimation
    - parameter uncertainties
  - hypothesis tests
    - e.g. significance of discovery
- Provides both Frequentist and Bayesian statistical tools
- Used to publish majority of results obtained from LHC data

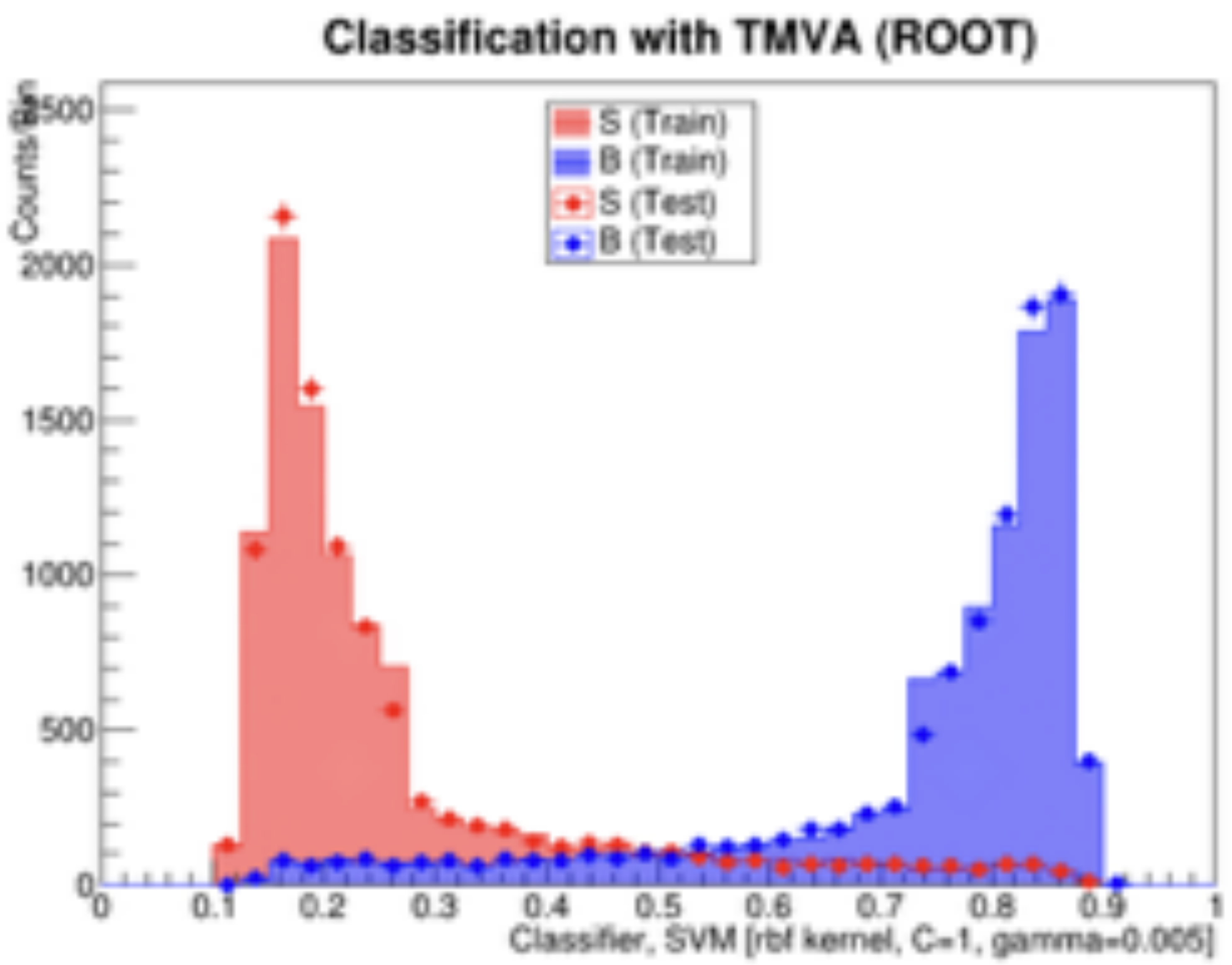




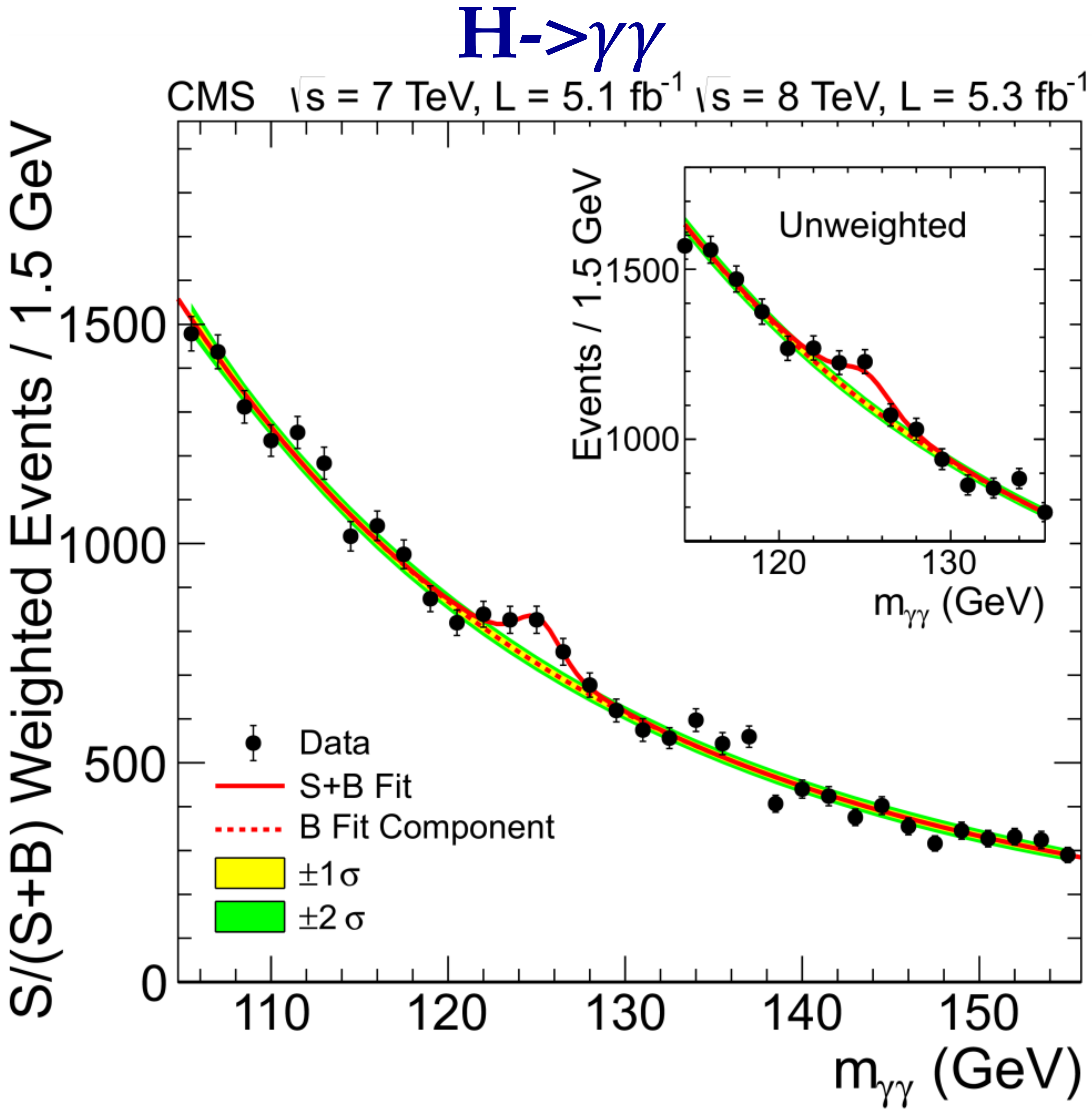
Machine Learning: TMVA

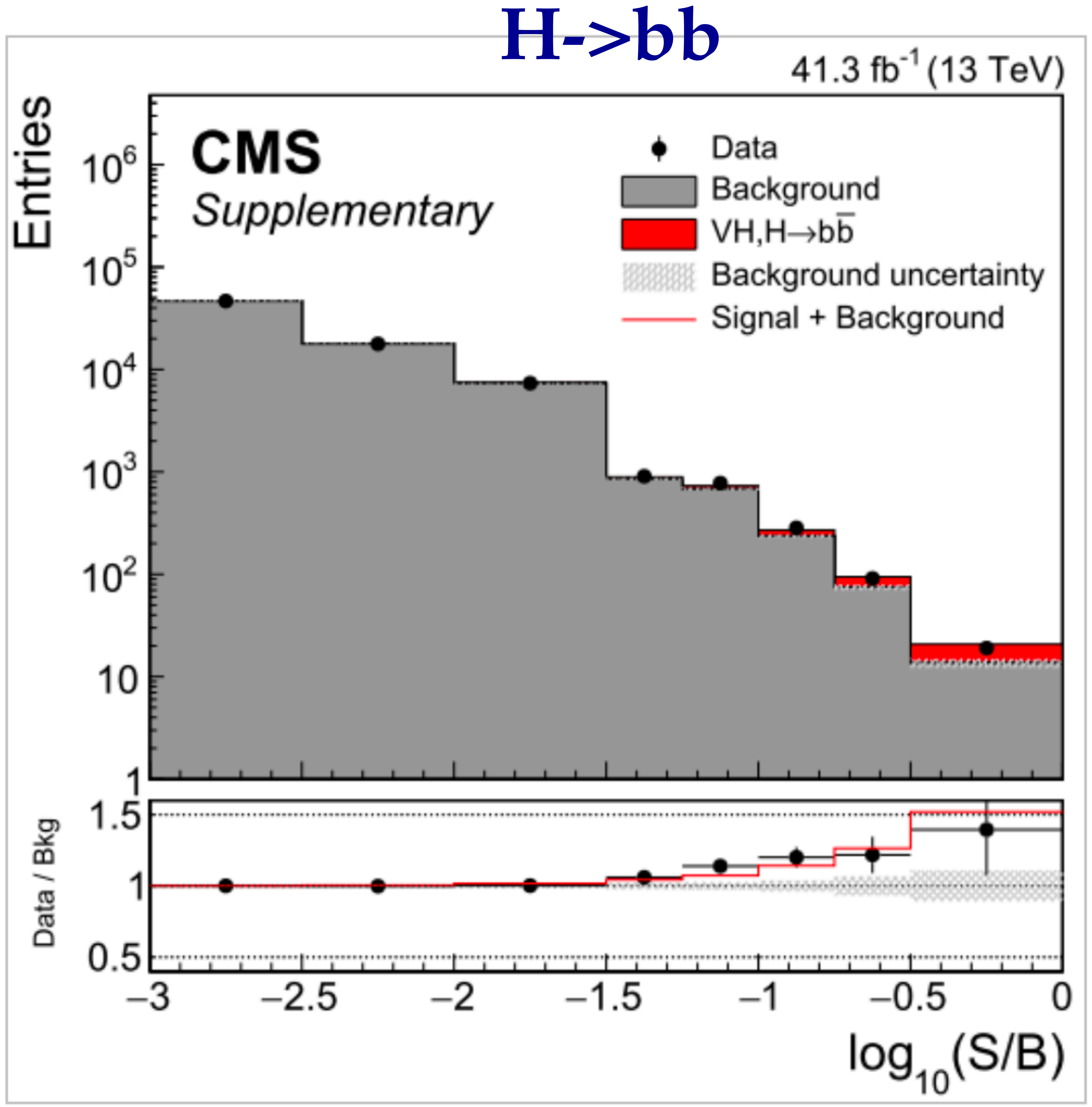
TMVA : Toolkit for Multi-Variate data Analysis in ROOT

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



Example: Higgs Discovery





Improvement in analysis from all areas using machine learning

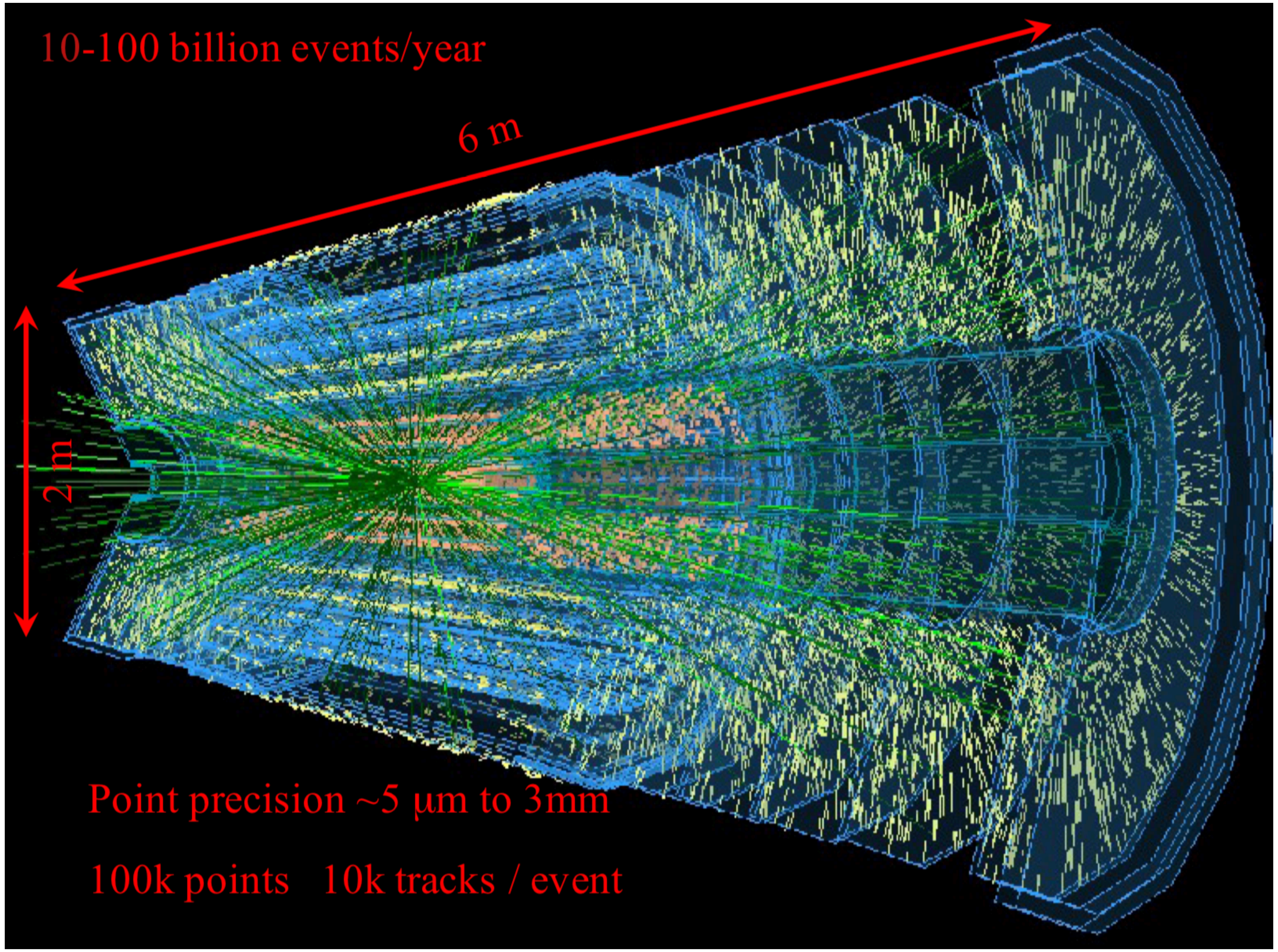
[S. Gleyzer]

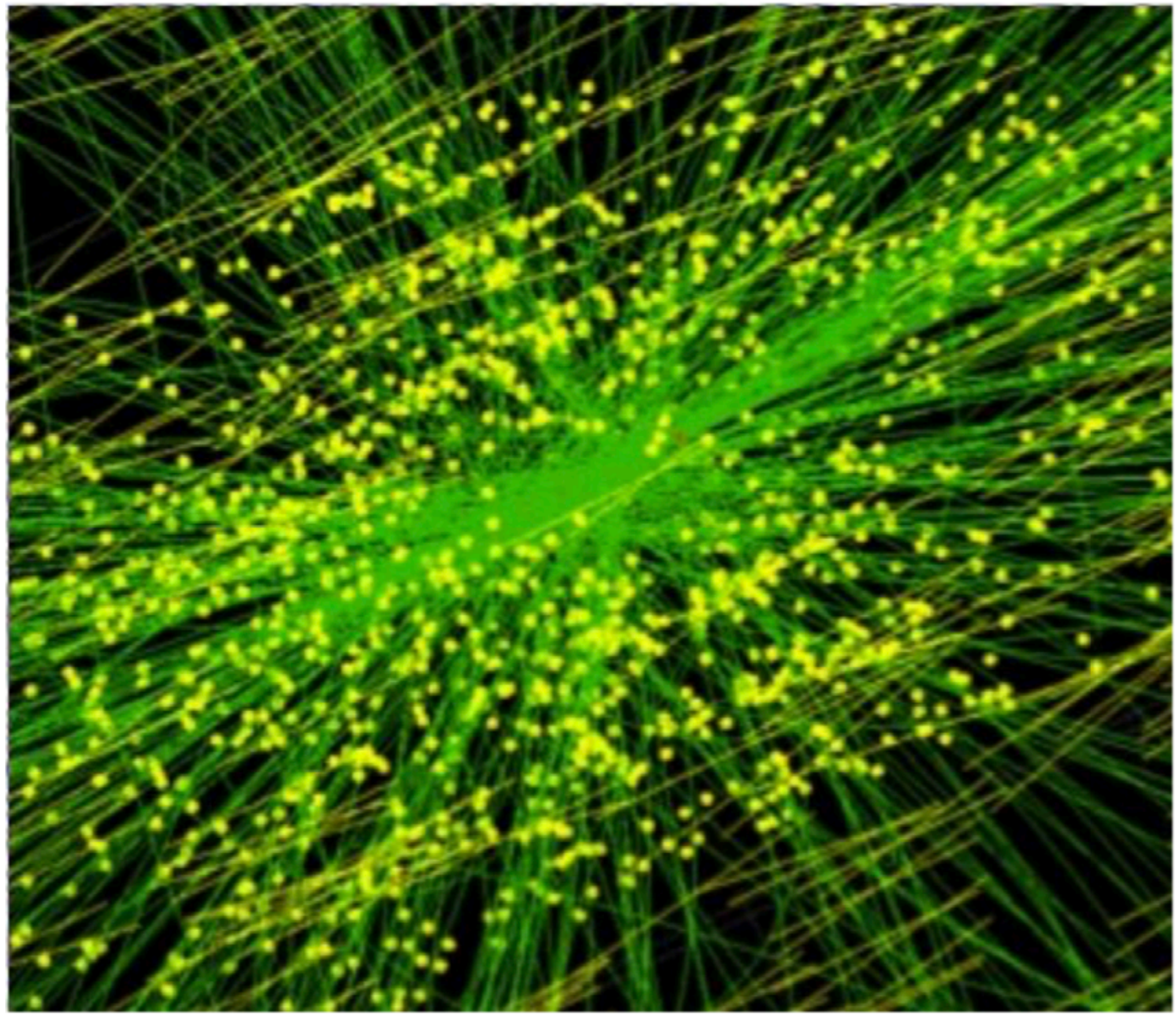
Deep Learning in HEP

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)

Example: Track Reconstruction at LHC





TMVA

- TMVA is not only a collection of multi-variate methods. It is a
  - **common interface to different algorithms**
  - **capability for both classification and regression**
  - training and testing of different methods on the same dataset
    - consistent evaluation and comparison
    - same data pre-processing
  - several tools provided for pre-processing
  - embedded in ROOT: **direct connection to the input data**

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**
  - including convolutional and recurrent networks
  - working on CPU and GPU

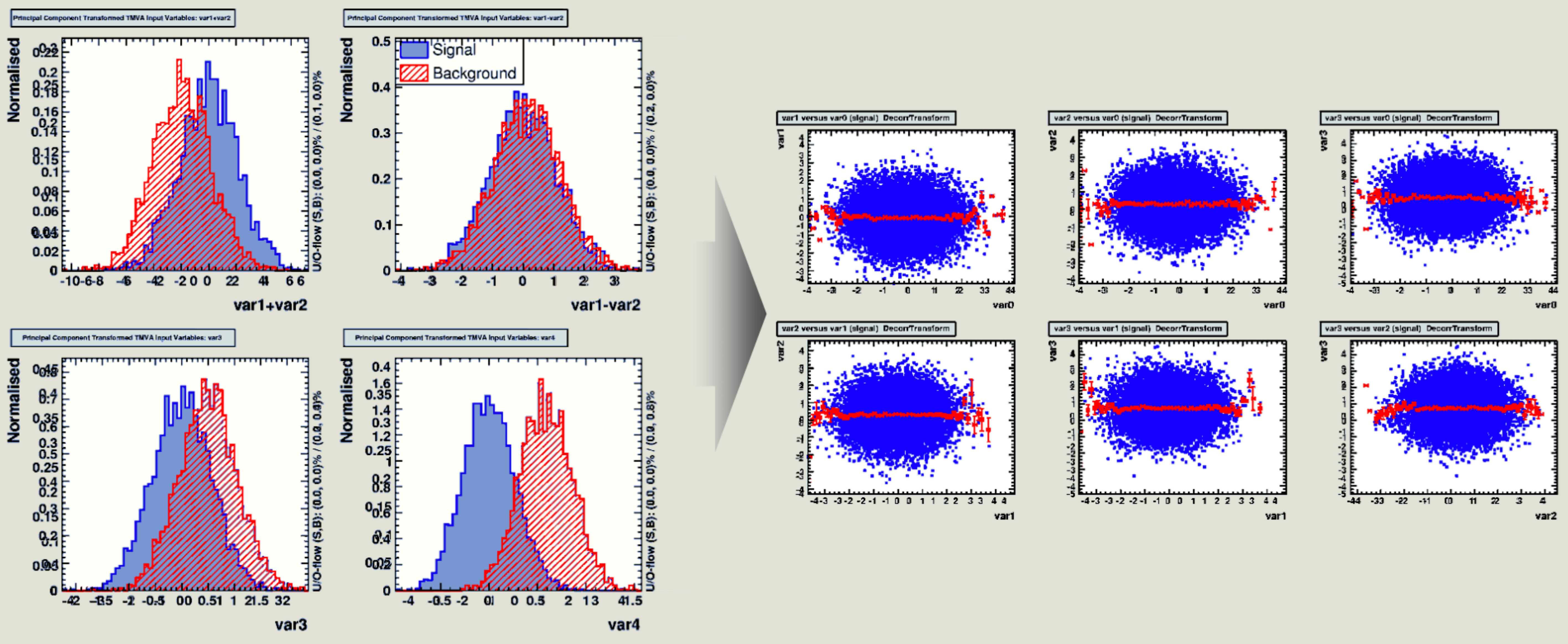
TMVA Workflow Features

TMVA supports:

- input data from ROOT data structures or ASCII data (e.g. csv)
- pre-selections on input data
- event weights
- various method for splitting training/test samples
- k-fold cross-validation and hyper-parameter optimisation
- algorithm to identify importance of input features
- GUI for output evaluation and analysis

Pre-processing of the Input

- Example: decorrelation of variable before training can be useful

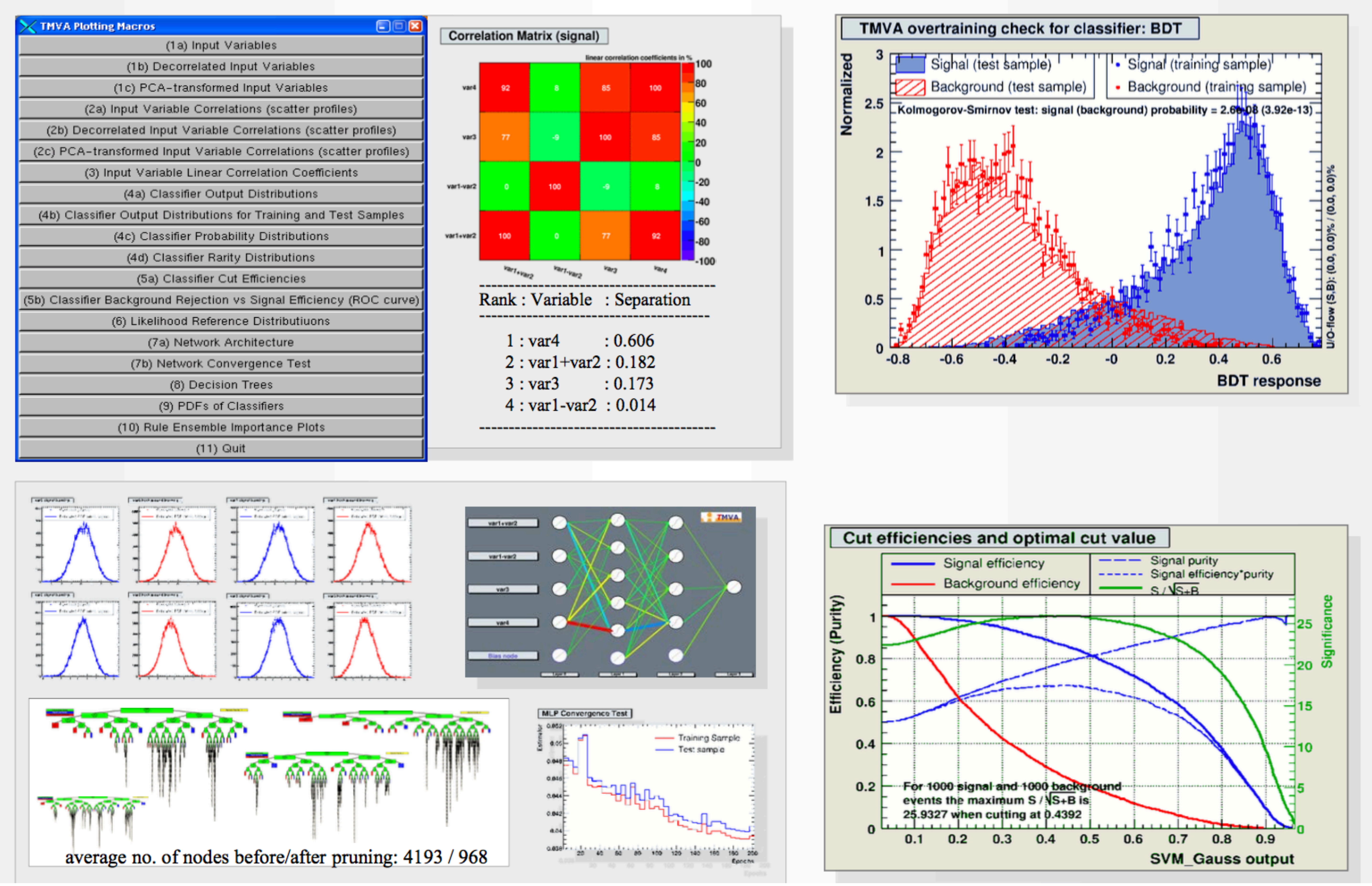


- Note that in cases with non-Gaussian distributions and/or nonlinear correlations decorrelation may do more harm than any good

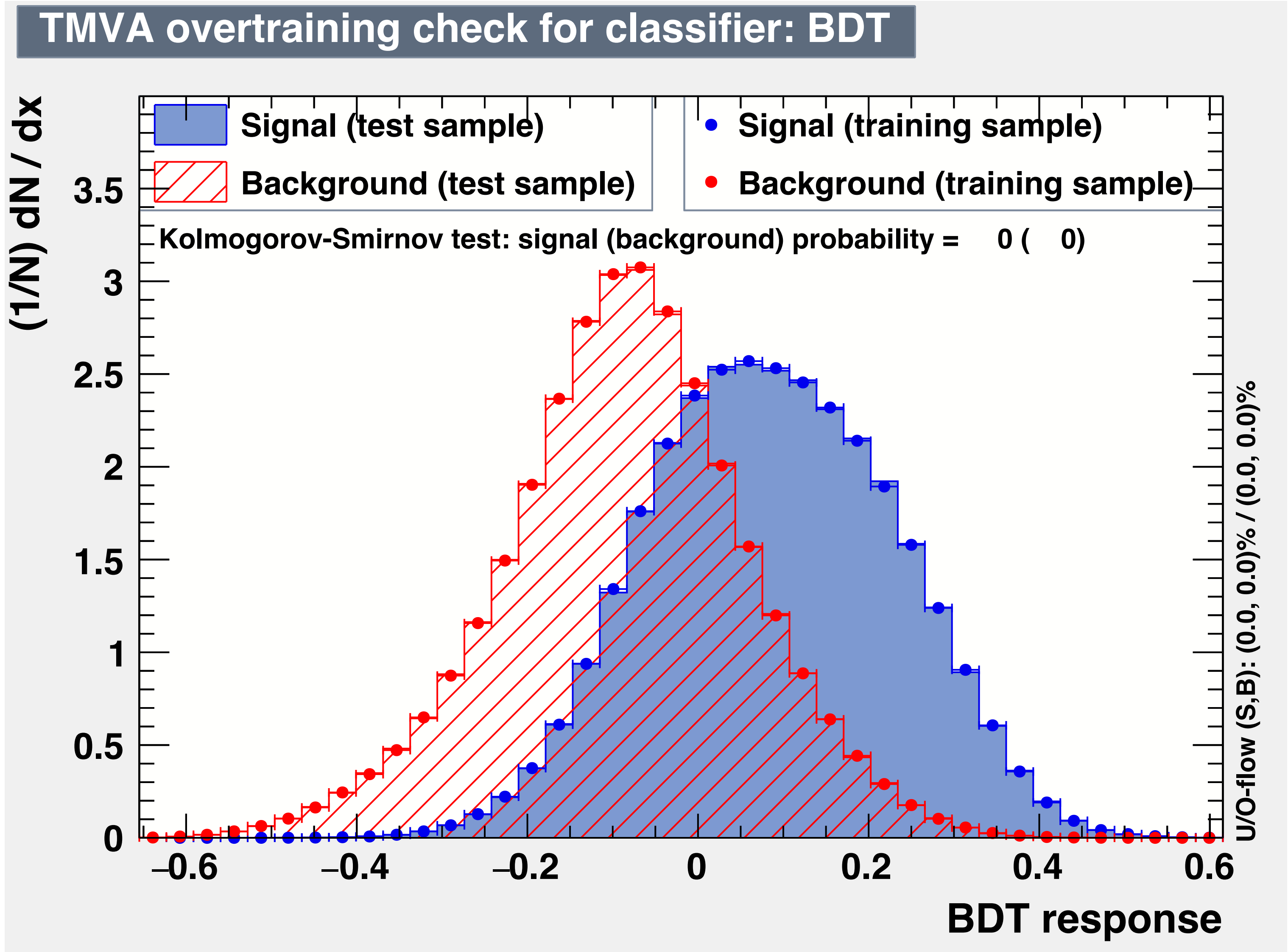
others pre-processing available (see Users Guide)

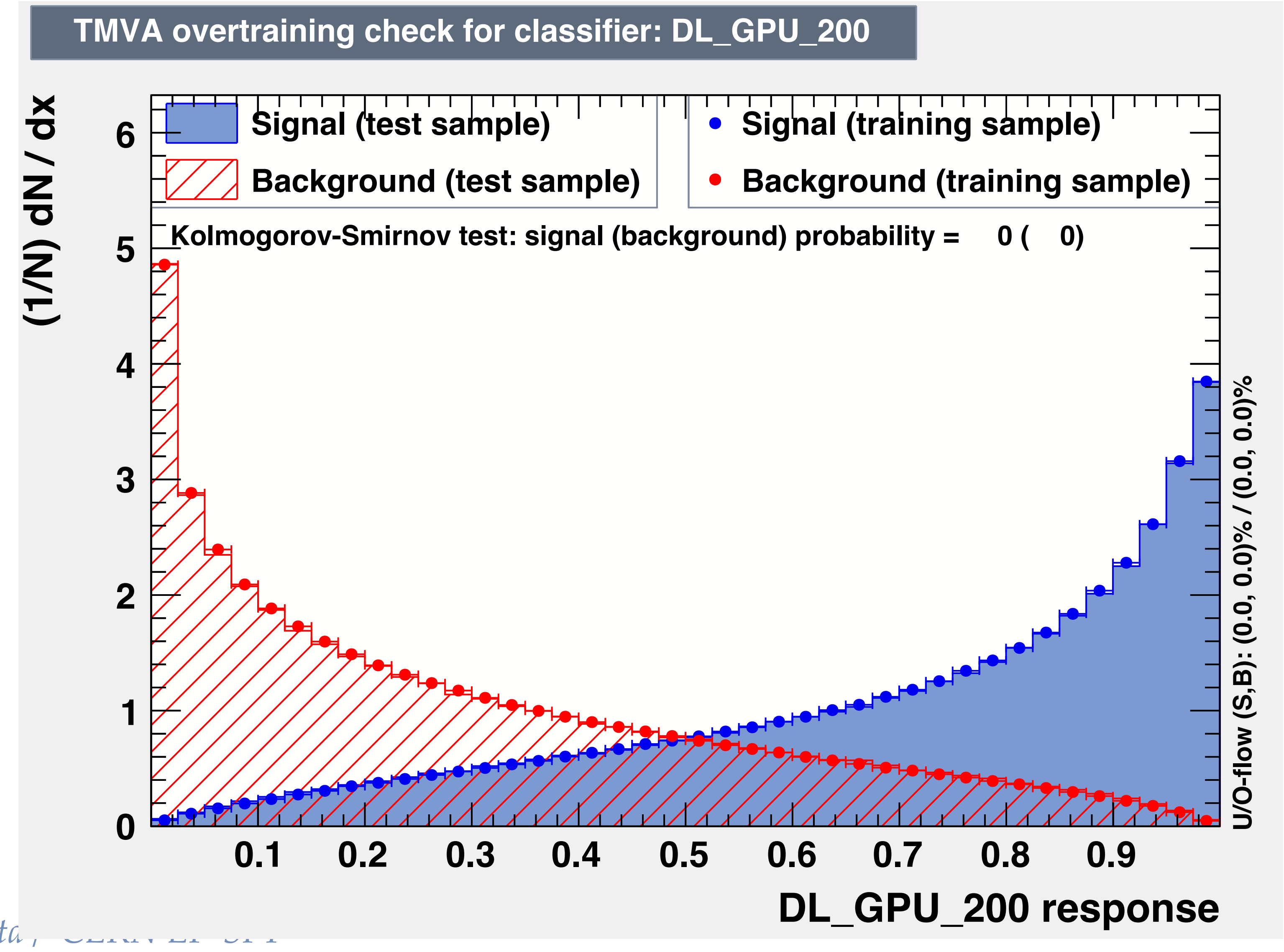
TMVA GUI

At the end of training + test phase, TMVA produces an output file that can be examined with a special GUI (**TMVAGui**)

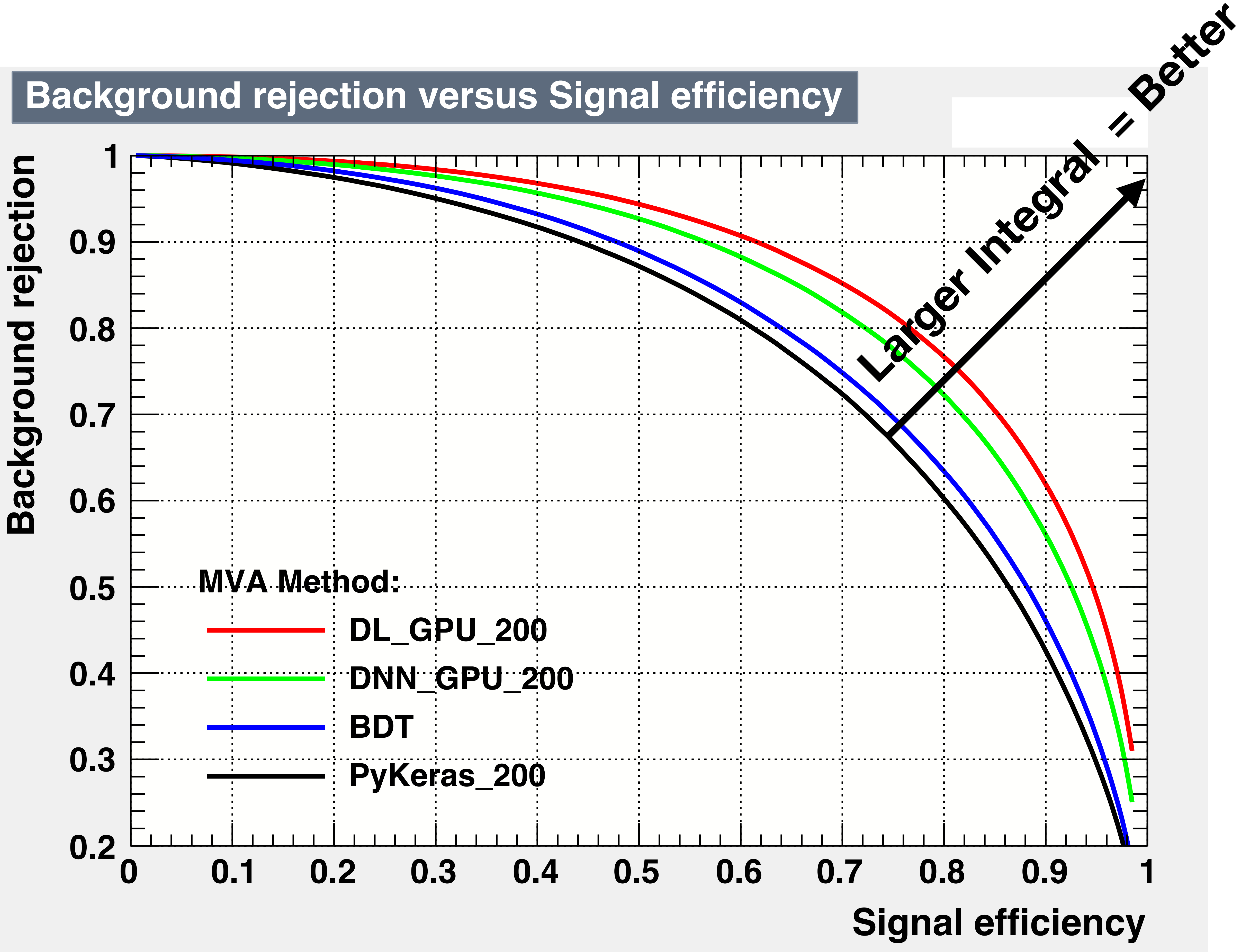


Event classification with TMVA



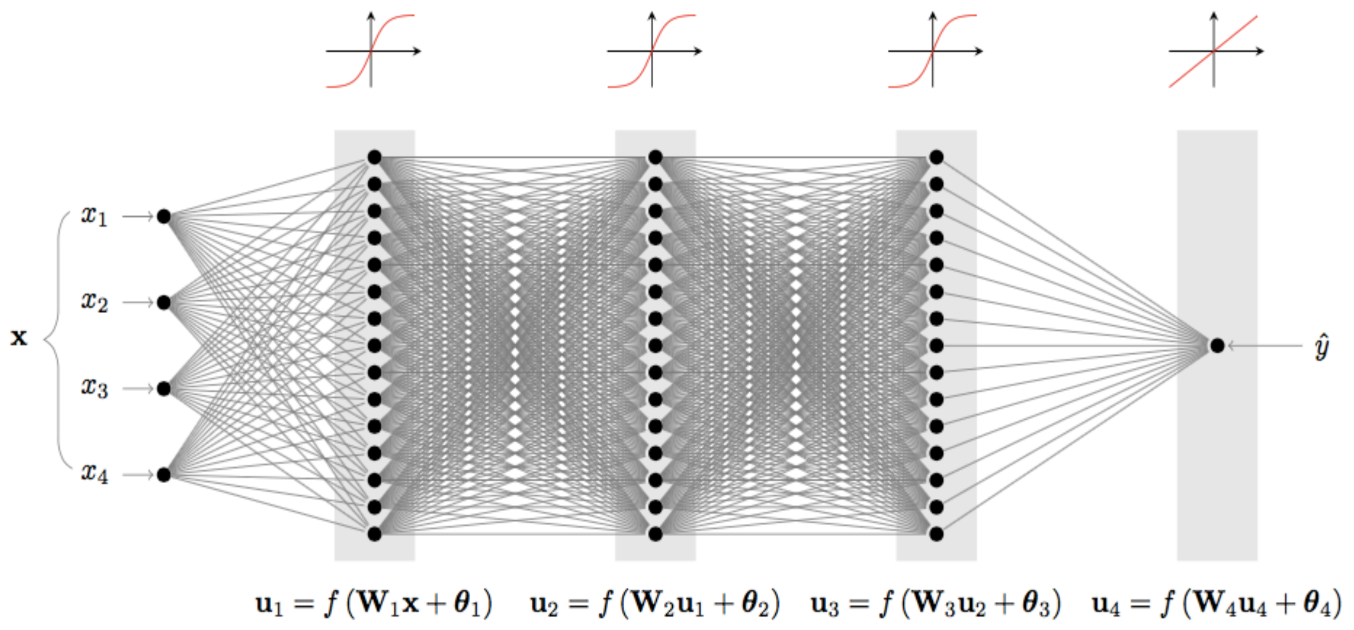


Receiver Operating Characteristic (ROC)

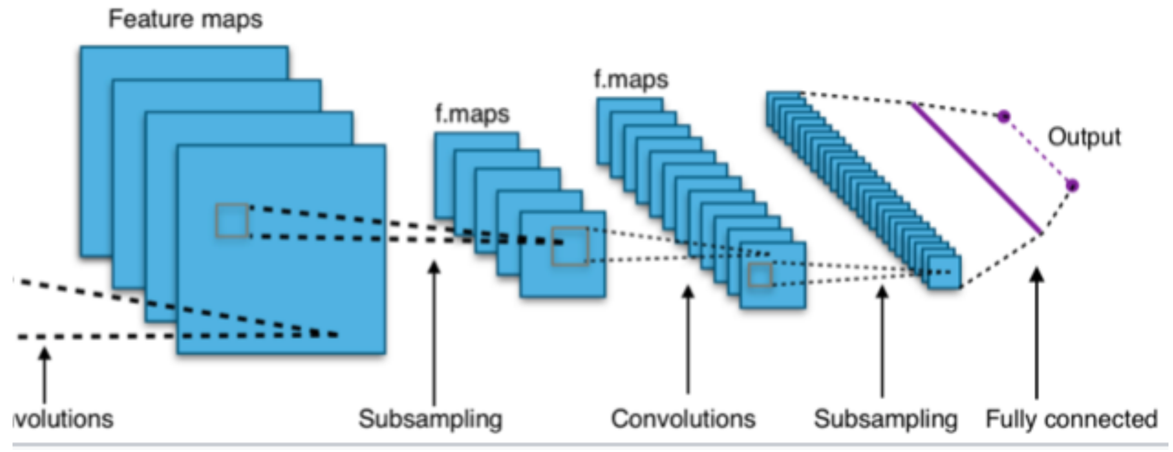


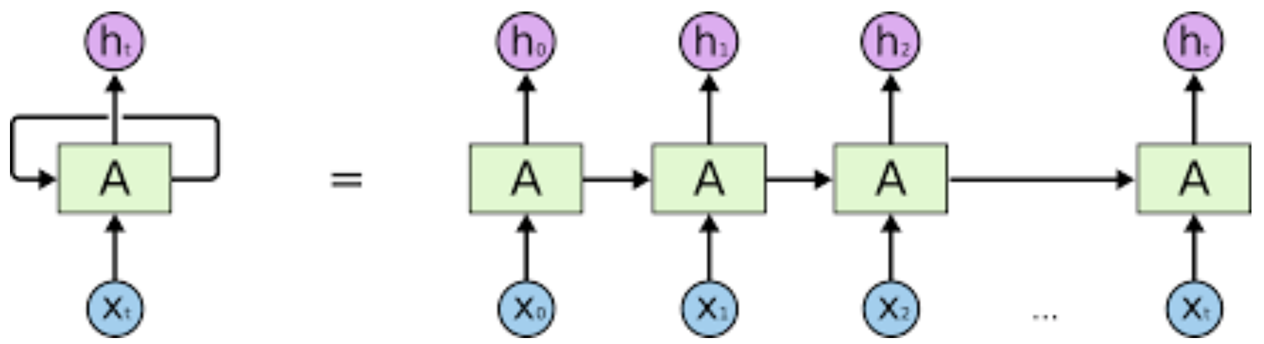
➜ Comparison of several methods possible in TMVA

Deep Learning in TMVA



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



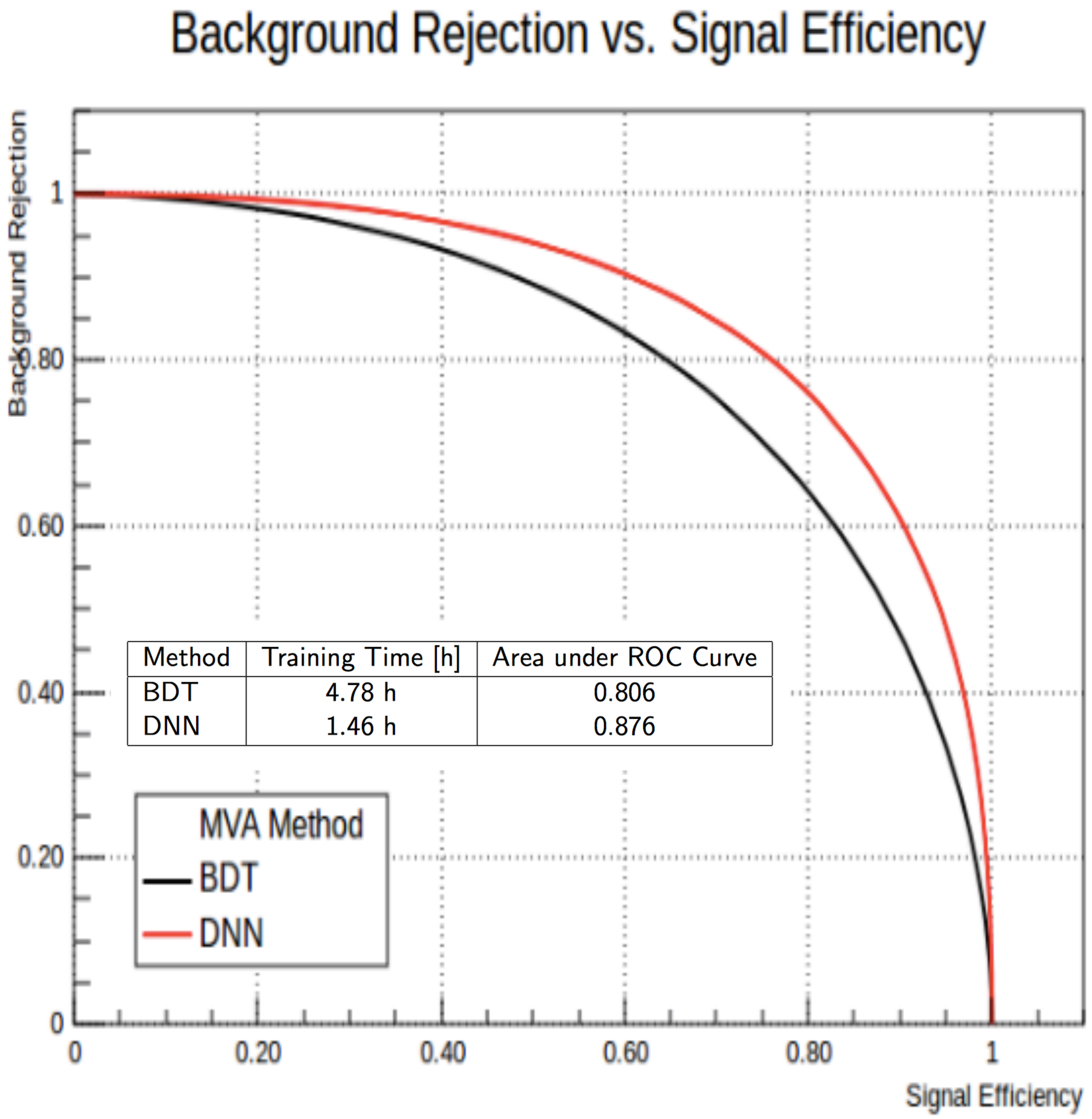


- Parallel implementation for both CPU and GPU
- Very efficient for training and inference of models

Deep Learning Performances

Example of Deep Learning for classification on a HEP data set (~ 10 M events)

DNN vs BDT



Method	Training Time [h]	Area under ROC Curve
BDT	4.78 h	0.806
DNN	1.46 h	0.876

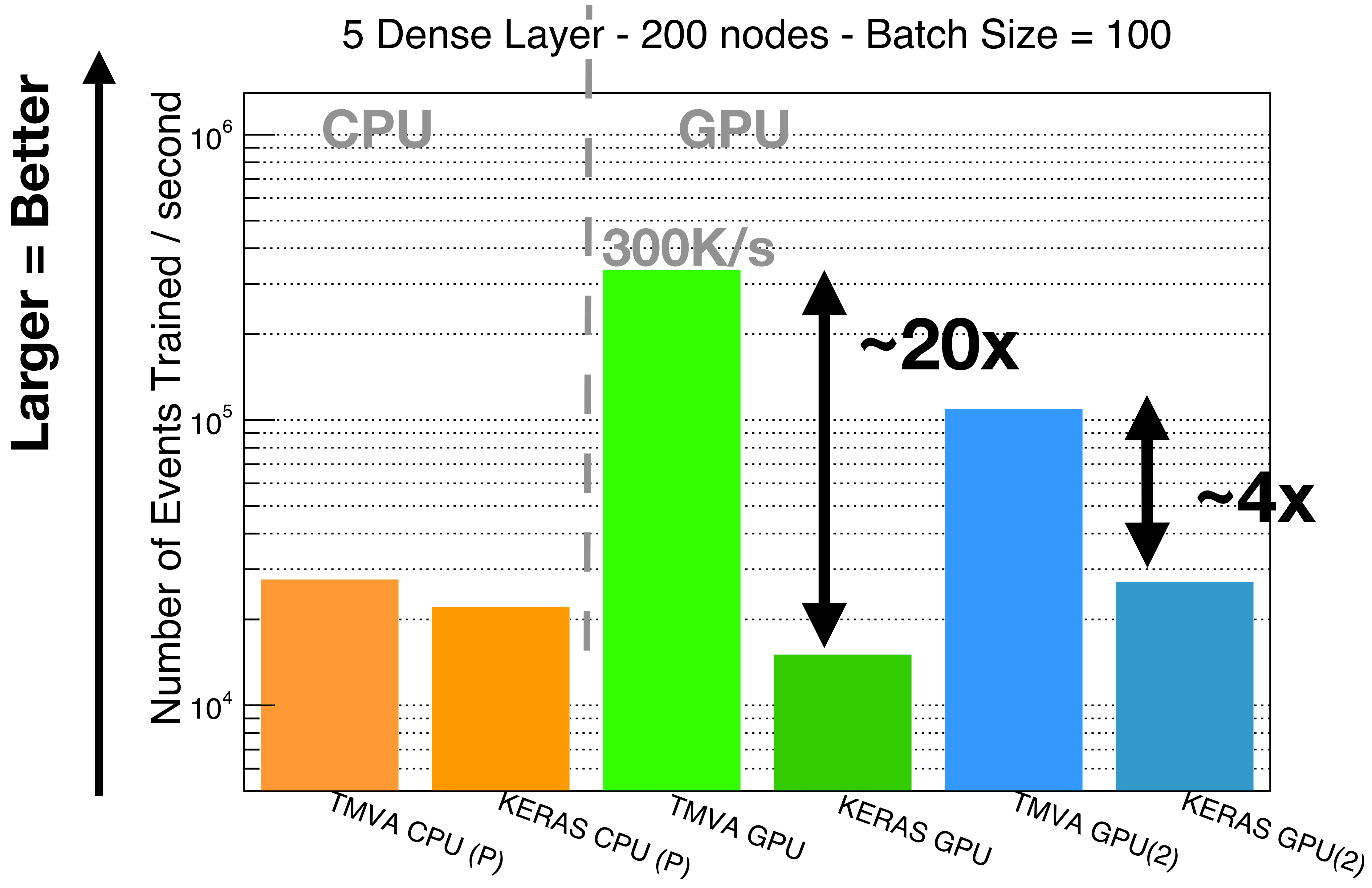
High classification performance compared to other ML methods (decision trees) when using many examples (events)

Fast train time possible thanks for parallelisation gains in using GPU

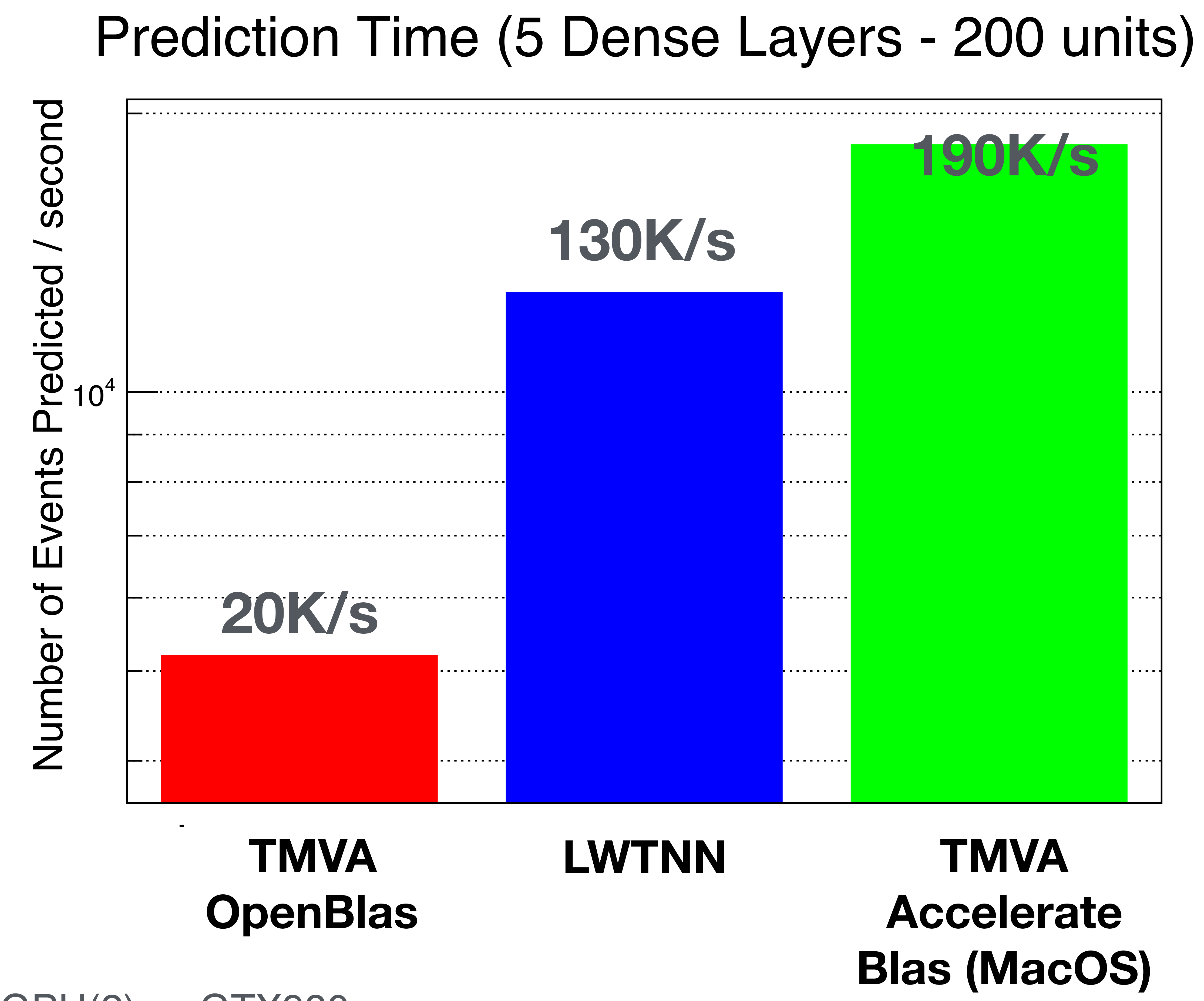
Deep Learning Performance

TMVA vs. Keras/Tensorflow on CPU and GPU using a typical HEP dataset

Training Performace (GPU)



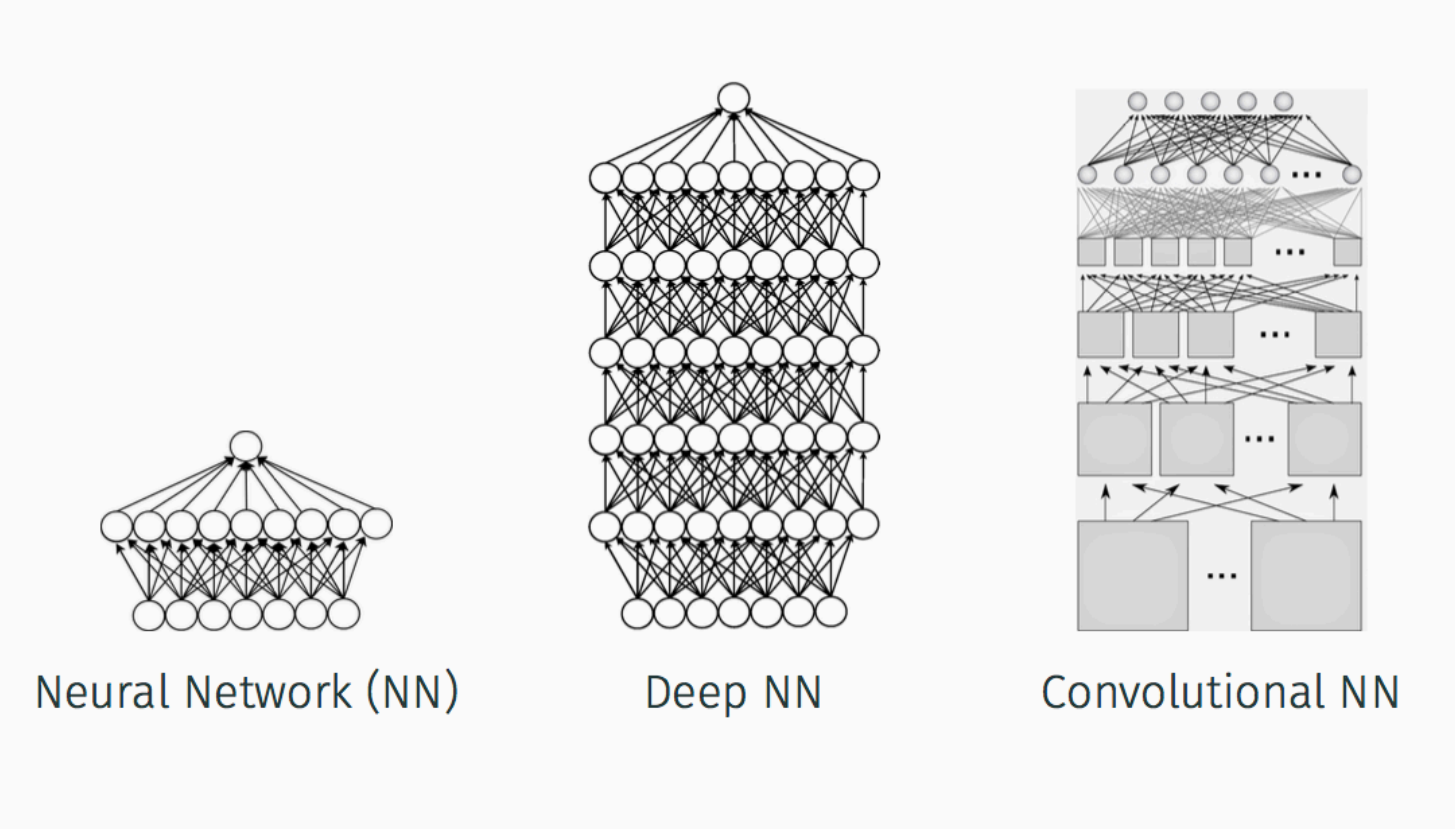
Single event inference Performace (CPU)



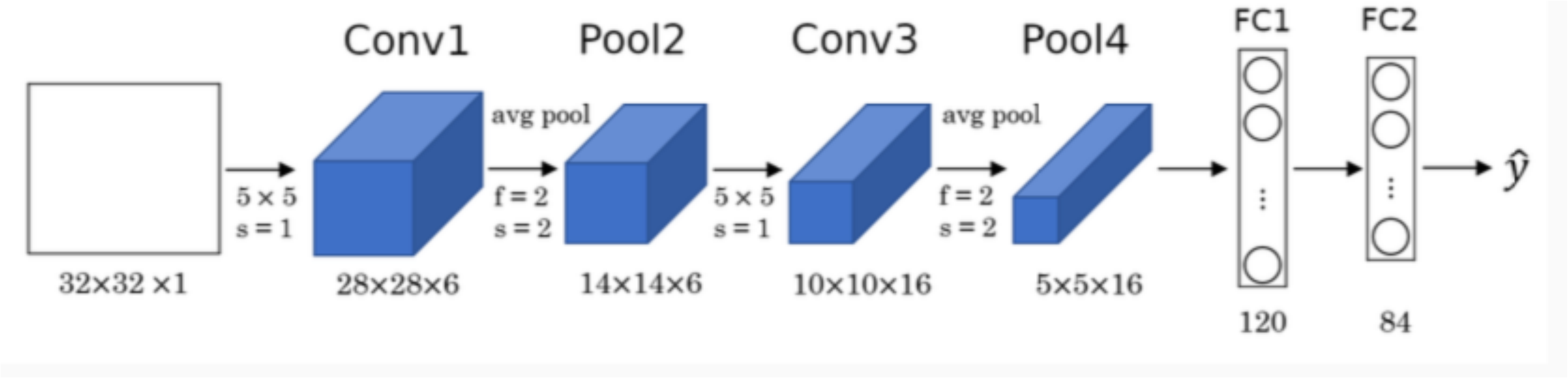
- Key difference is GPU utilisation
  - Tensorflow optimised for large operations

CPU— Intel Xeon E5-2683 (28 core) GPU — GTX1080Ti GPU(2) — GTX980

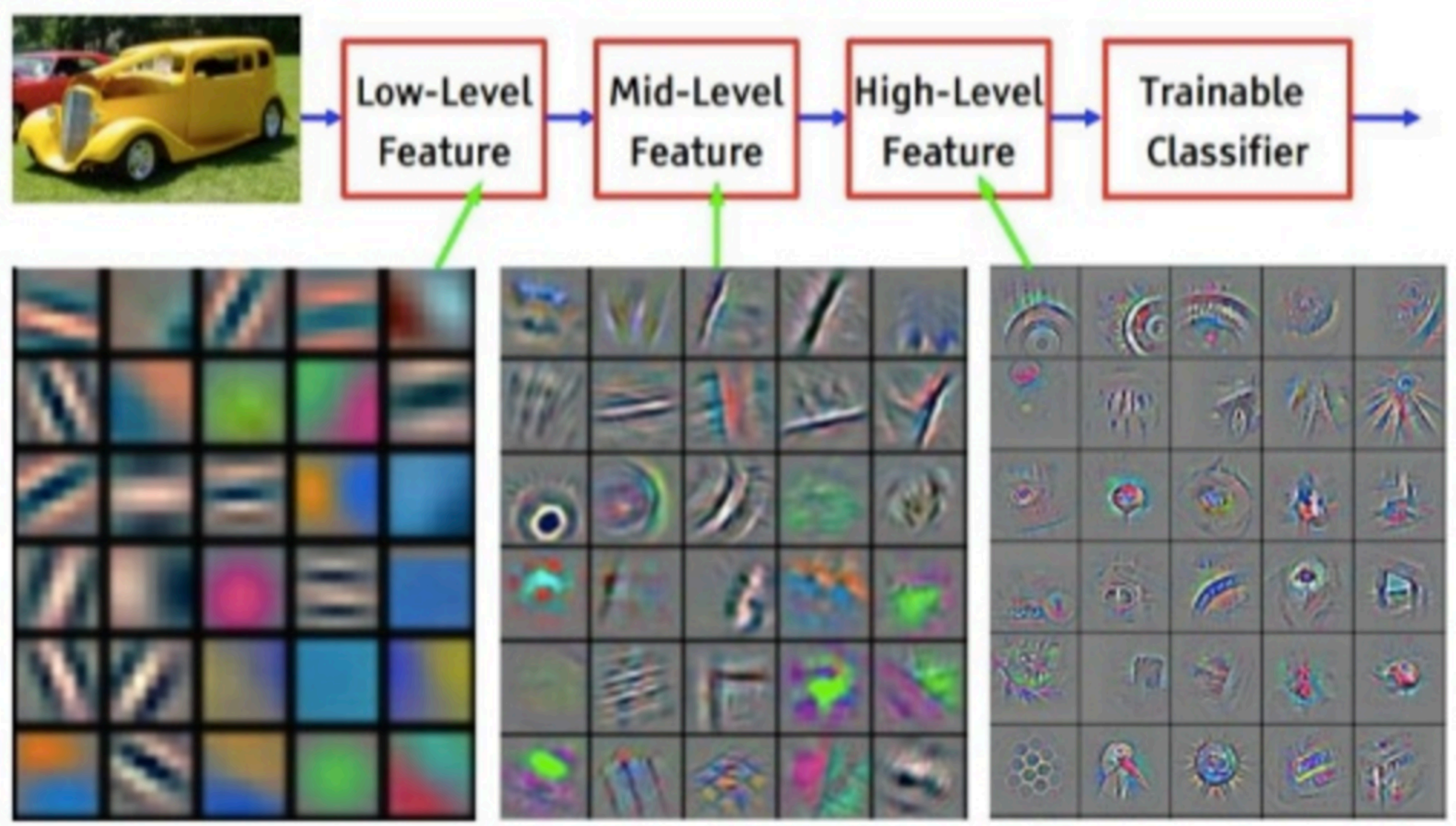
Convolutional Neural Networks



Convolutional Networks



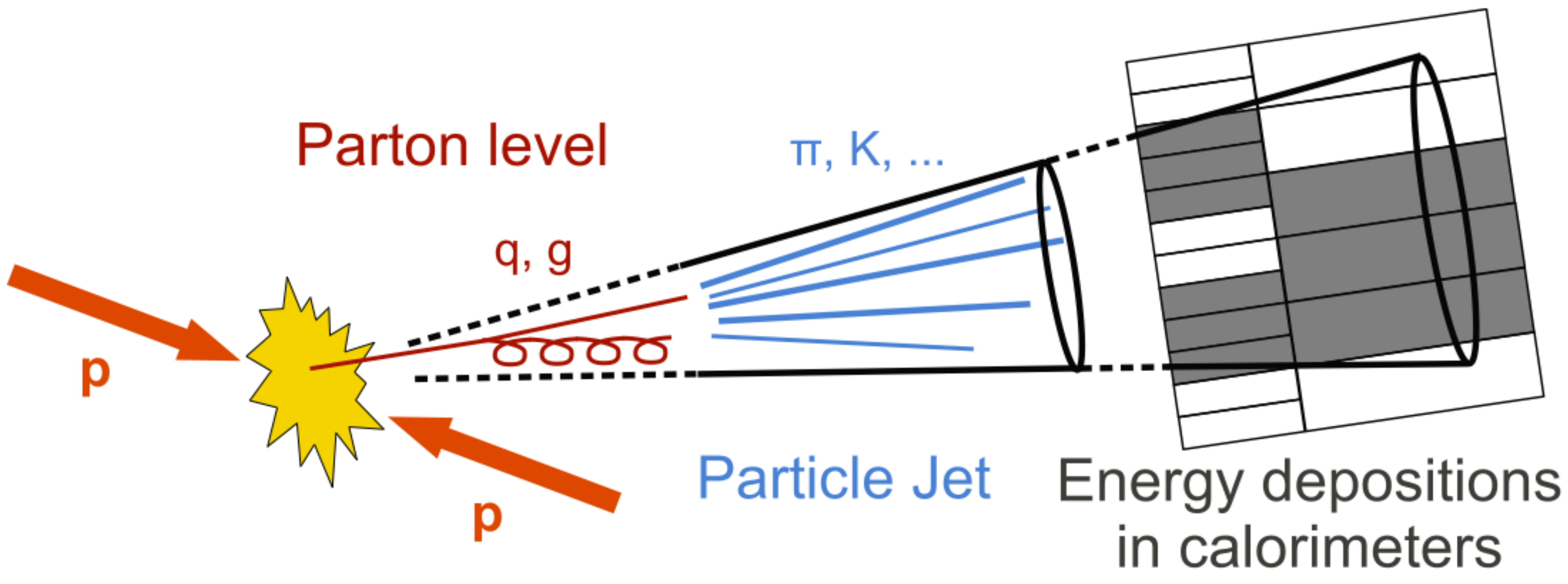
- Local structures captured at the early stage convolutions
- Long range structures in late stage convolutions and in the final dense layers

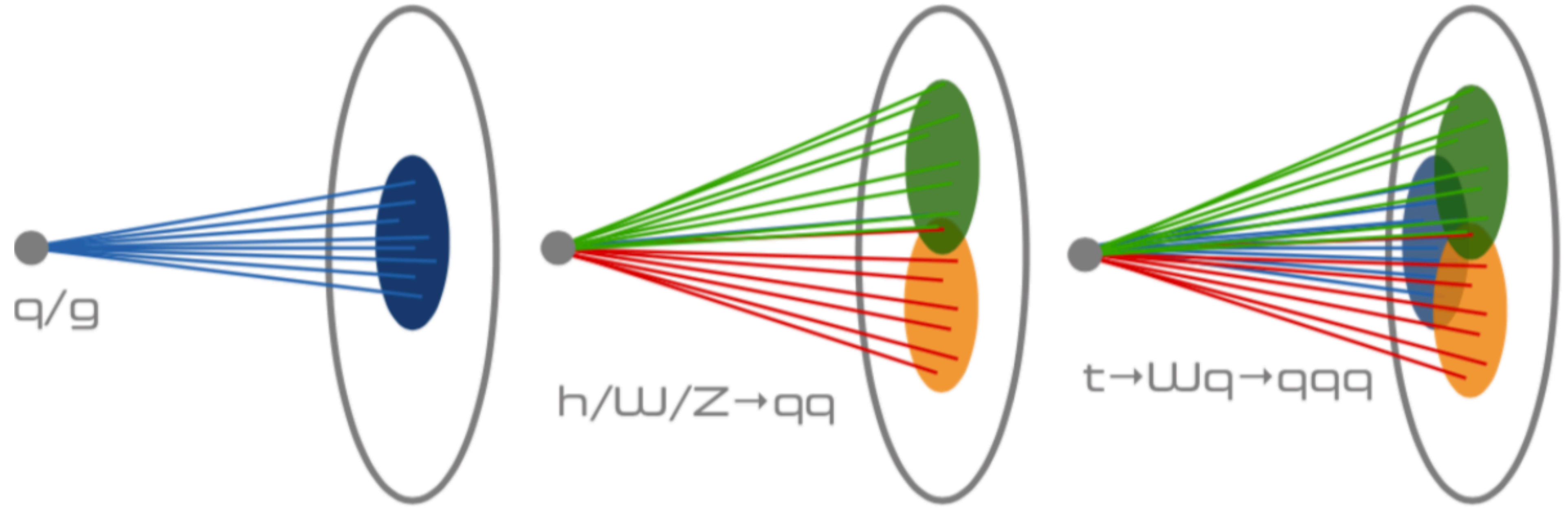


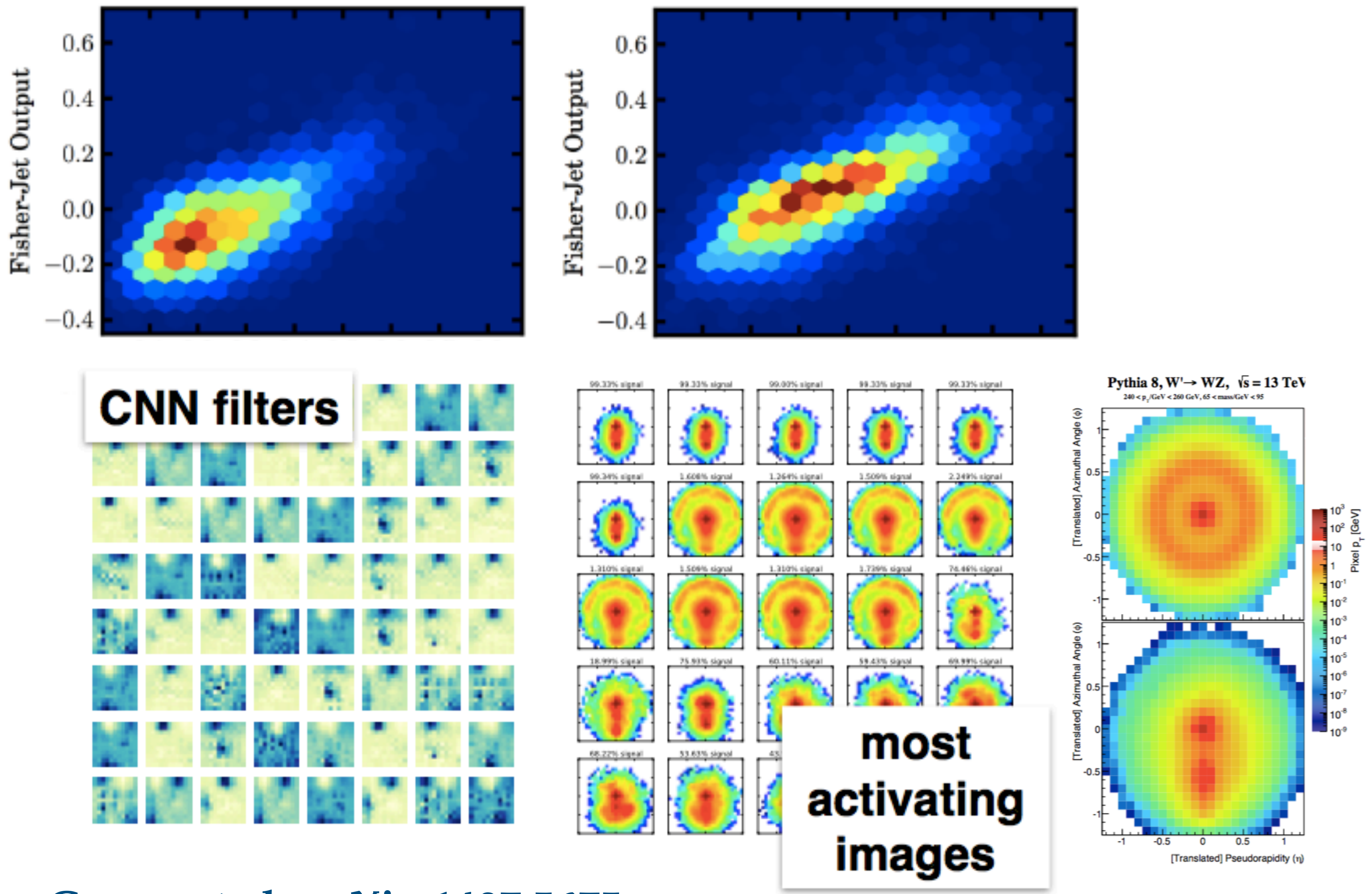
Feature Visualization of Convnet trained on ImageNet from [Zeiler & Fergus 2013]

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**





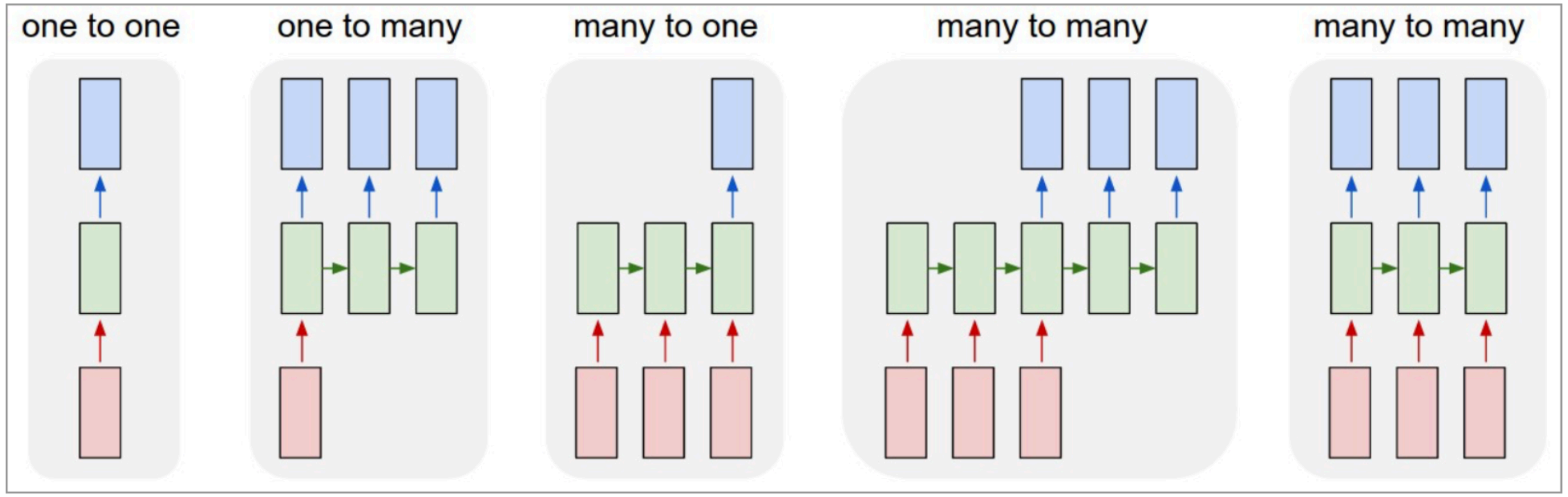


Cogan et al., arXiv:1407.5675

De Olivera et al., arXiv:1511.05190

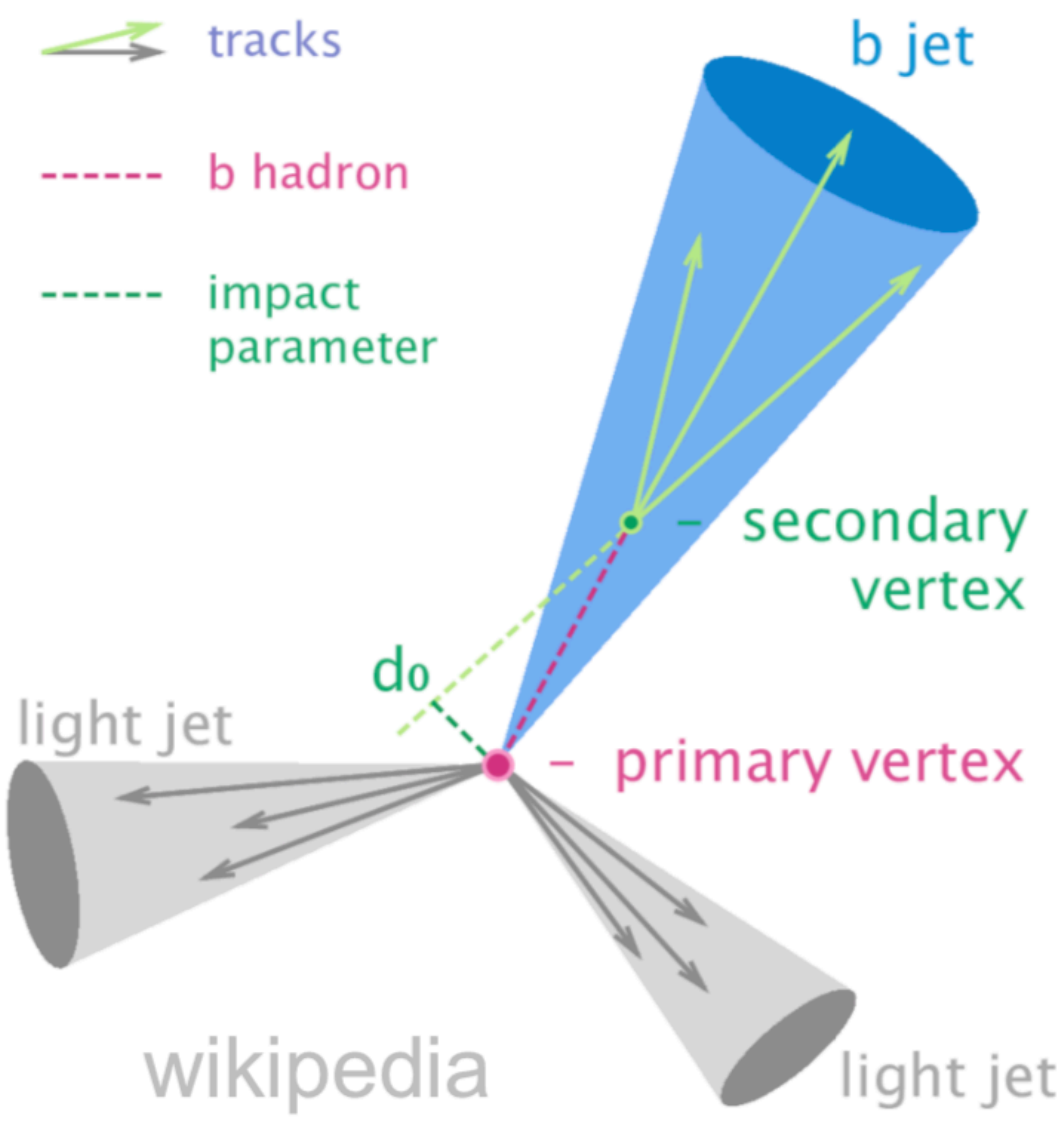
Recurrent Neural Networks

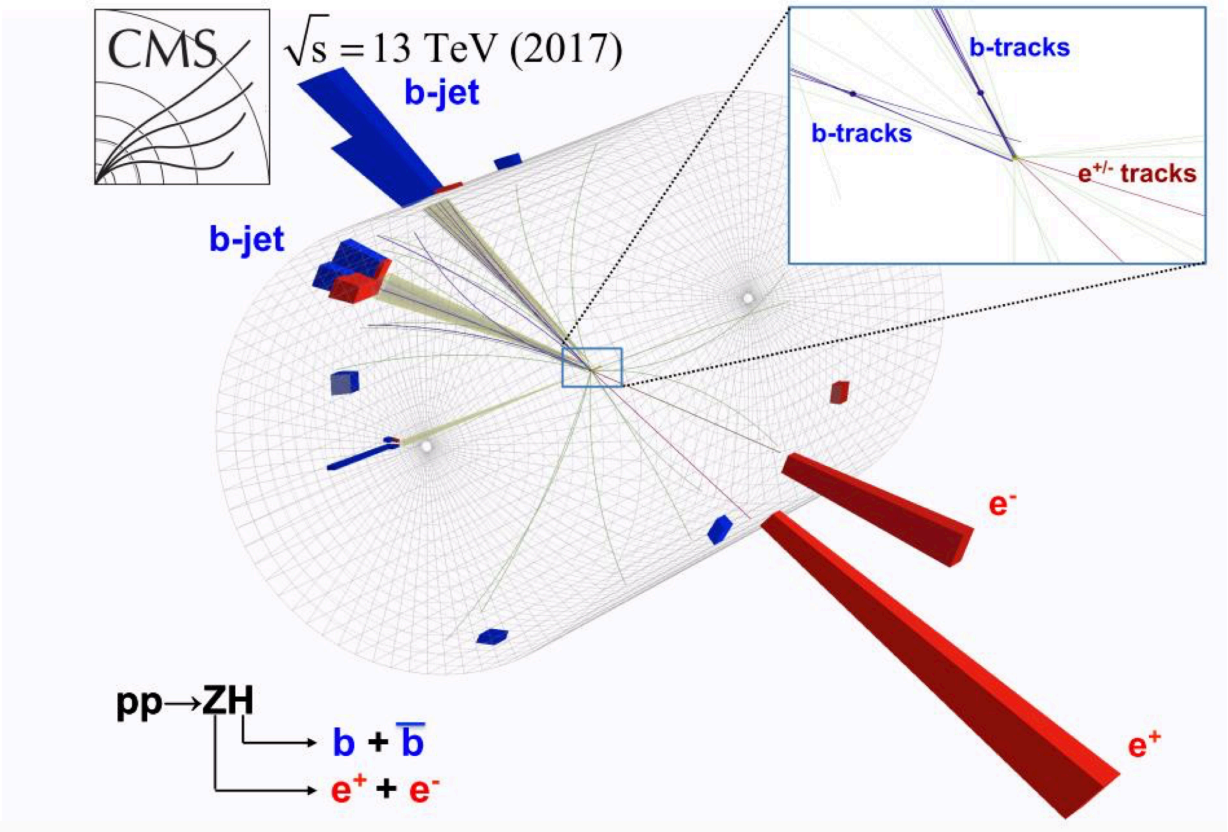
- What if we have variable length input data ?
- Or if input data is a sequence:
  - time series data (e.g. financial data)
  - audio or video data
  - natural language text
  - sequence of objects (e.g. particles sequence produced in a collision)
- Recurrent neural networks are special type of networks designed for these type of data

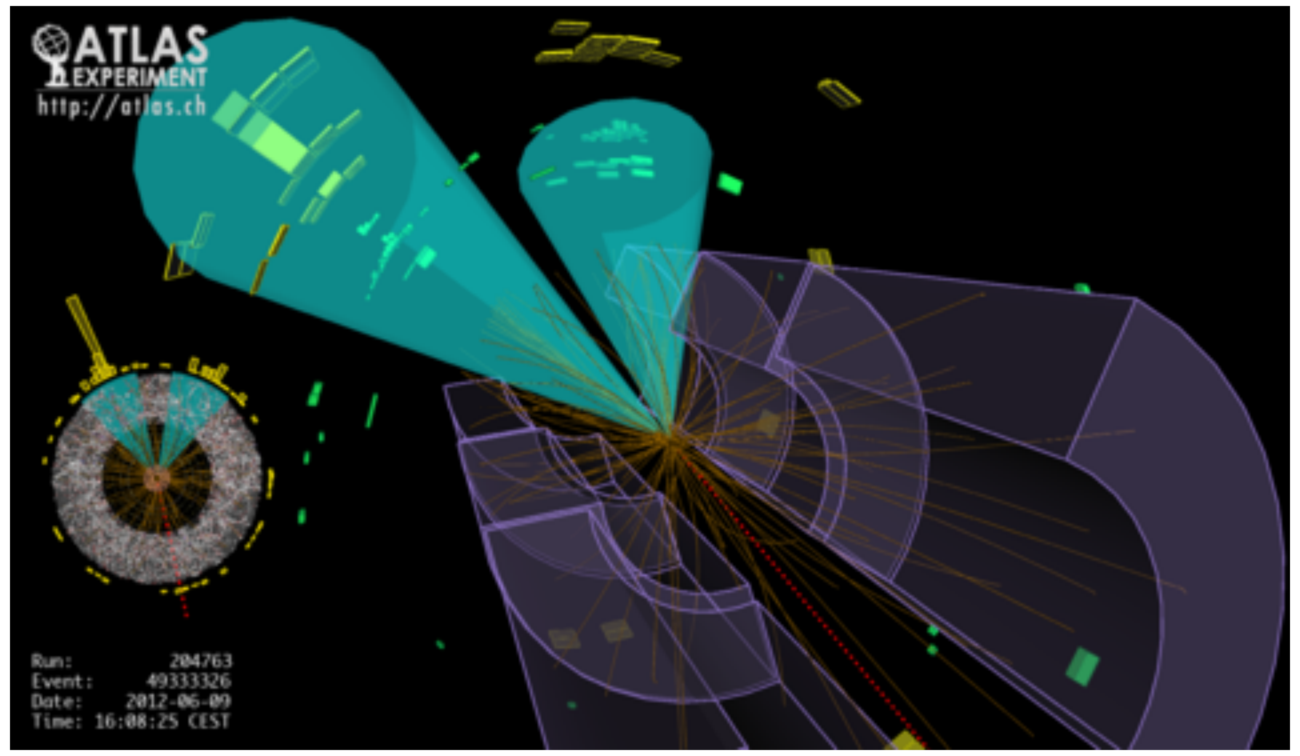


Flavour Jet tagging in HEP

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



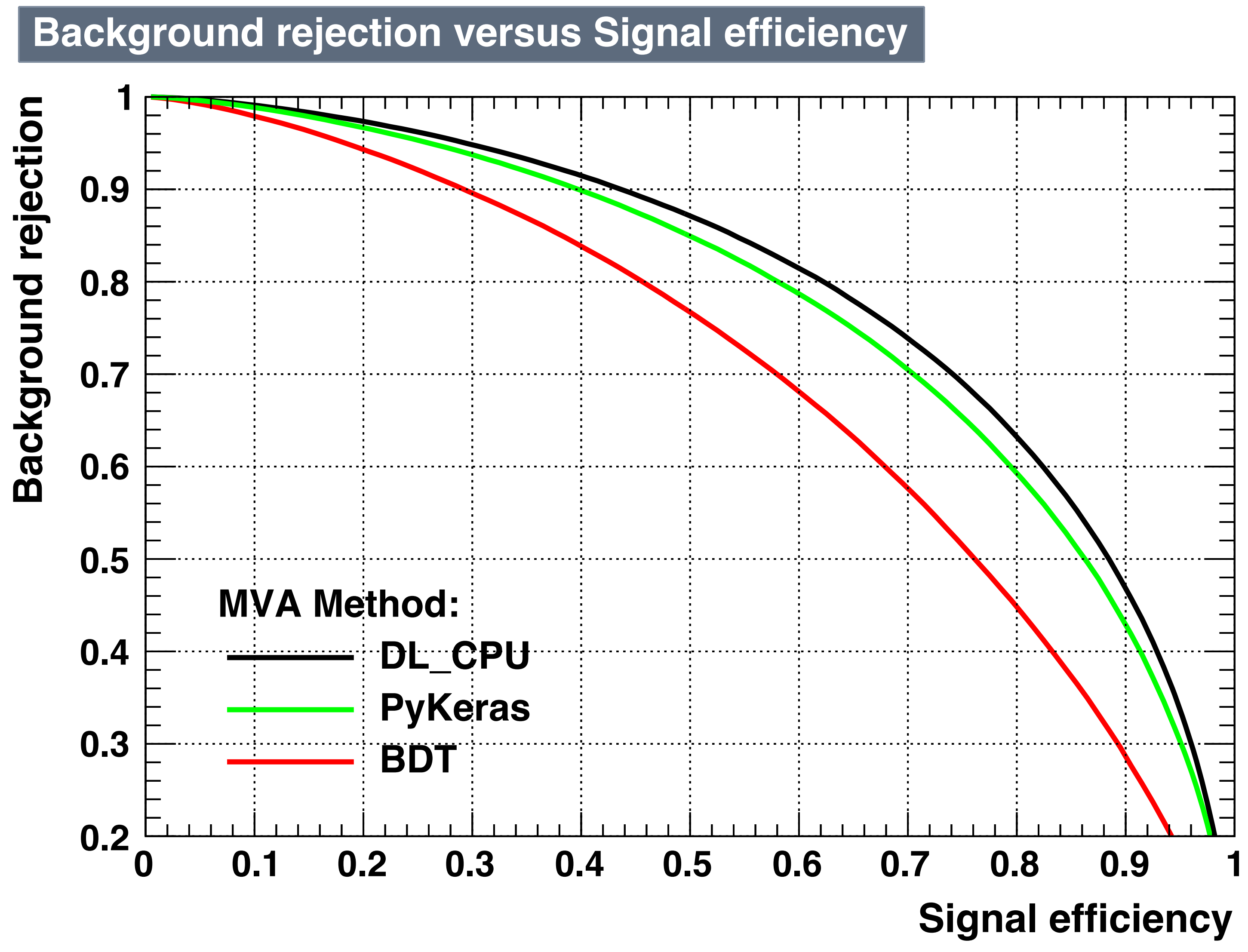




TMVA Interfaces

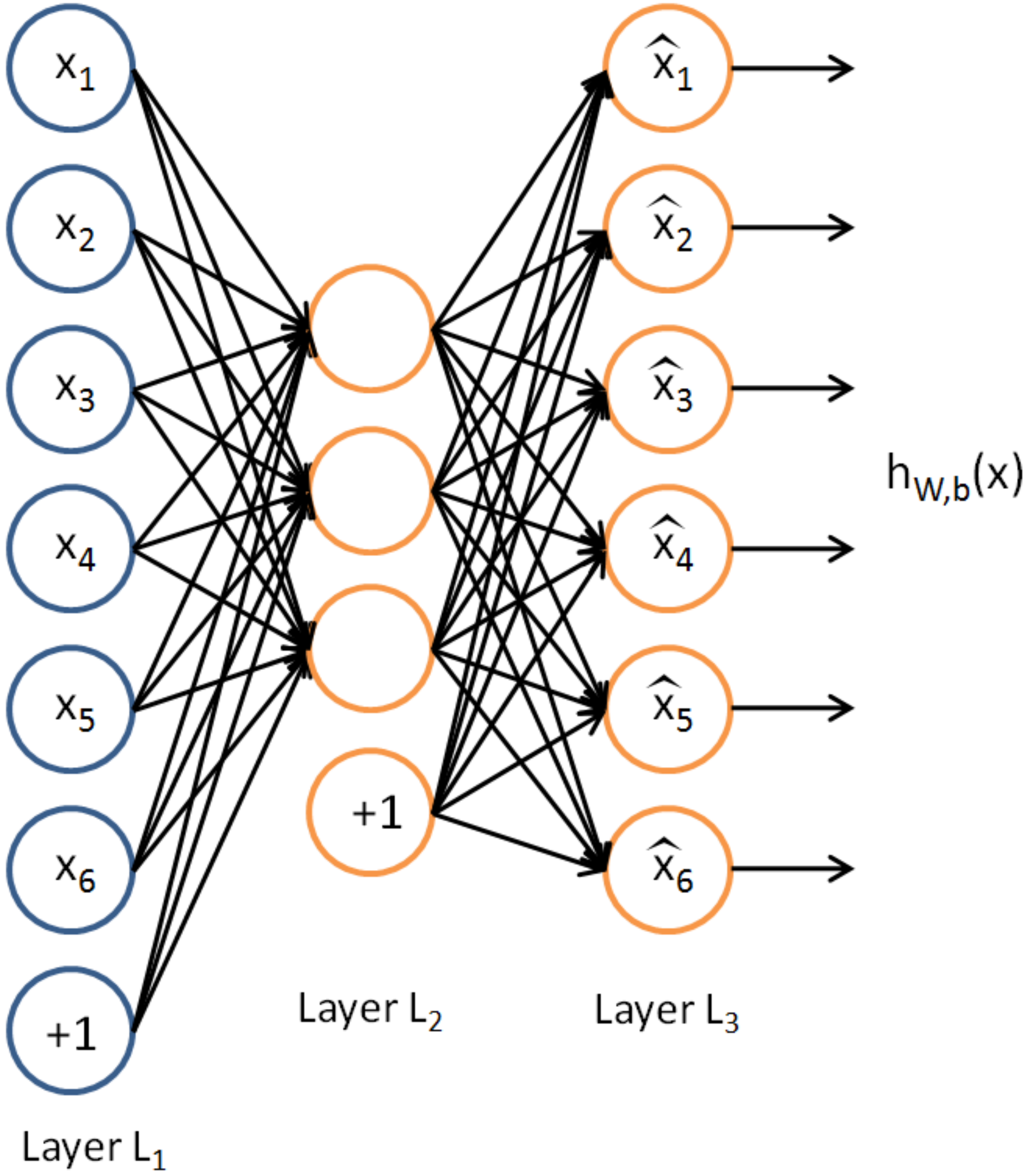
External tools are available as additional methods in TMVA and they can be trained and evaluated as any other internal ones.

- **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



Deep Autoencoder

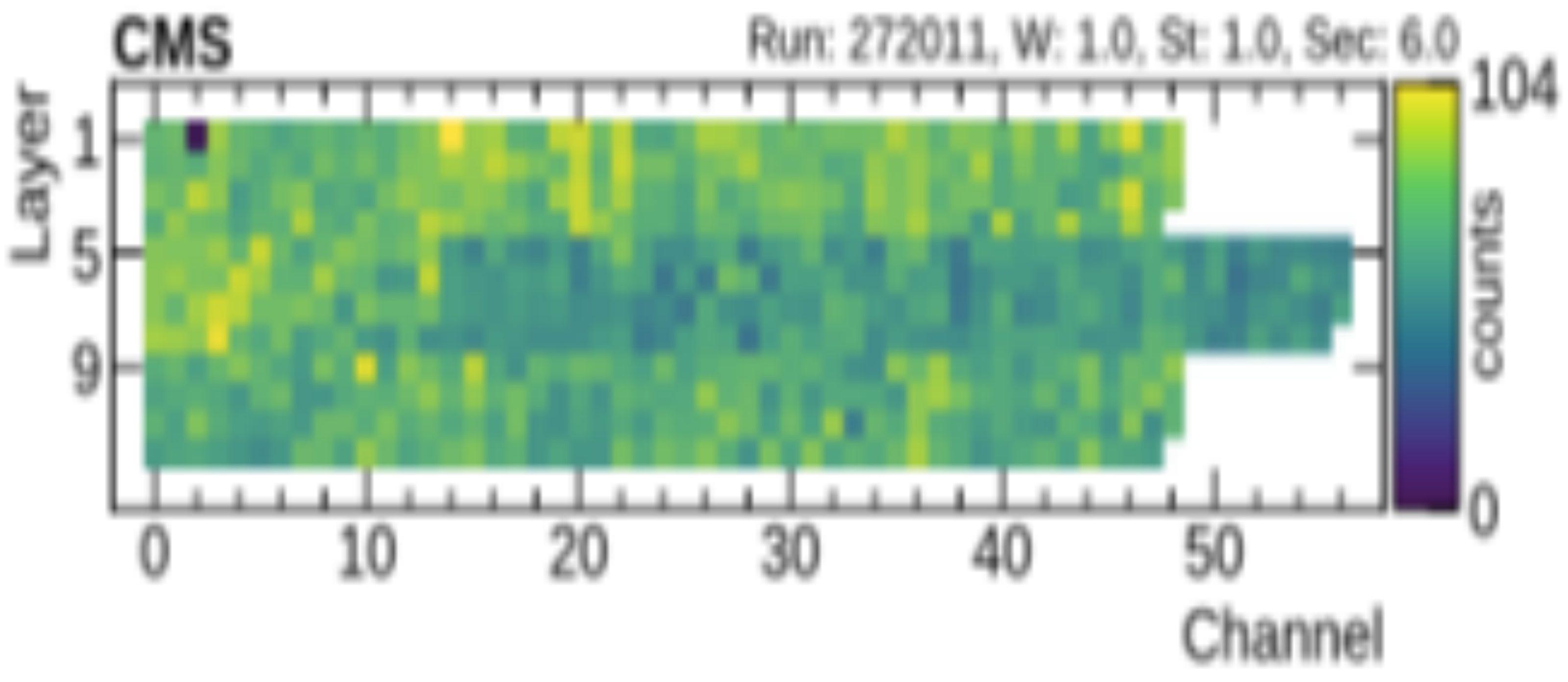
- An unsupervised neural network
- Trained by setting the target values equal to the inputs x_i
- Can be used for
  - **dimensionality reduction**
  - **anomaly detection**
  - and as a generator
    - **variational auto-encoders**

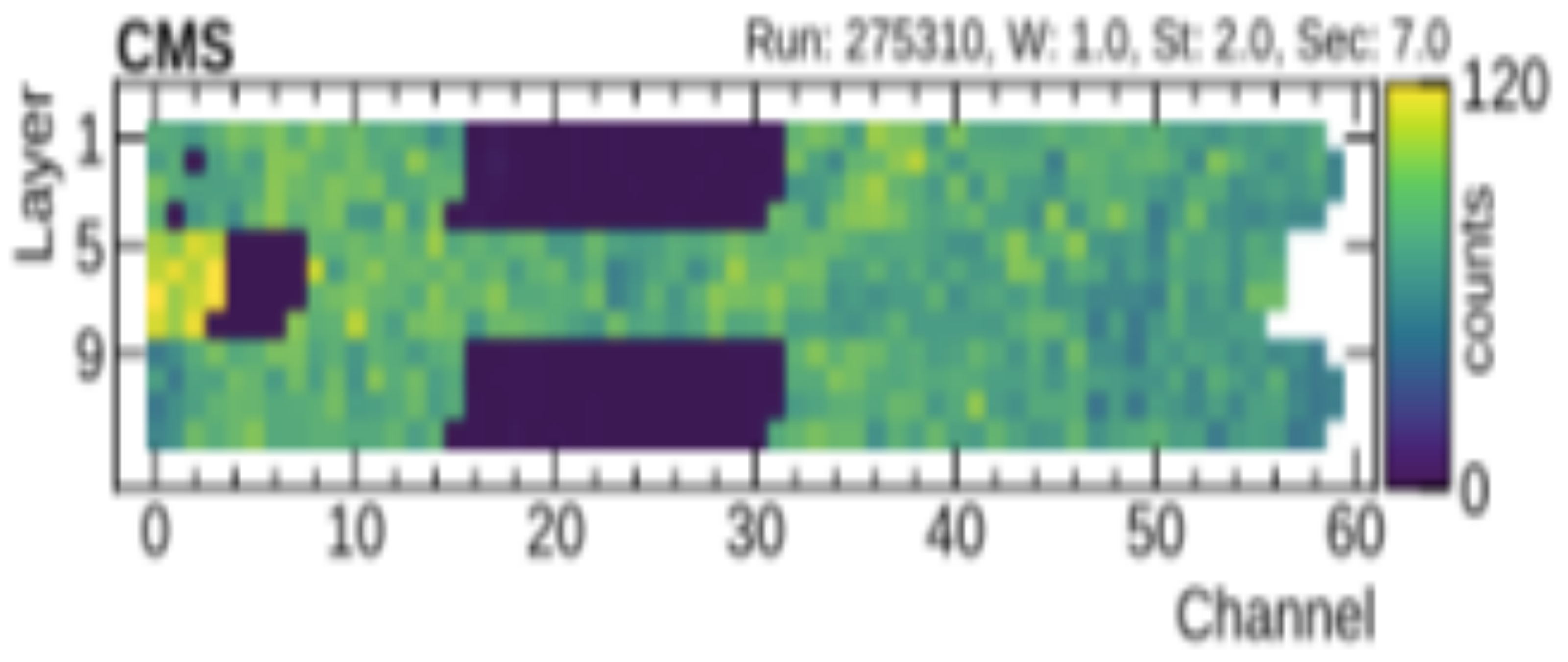


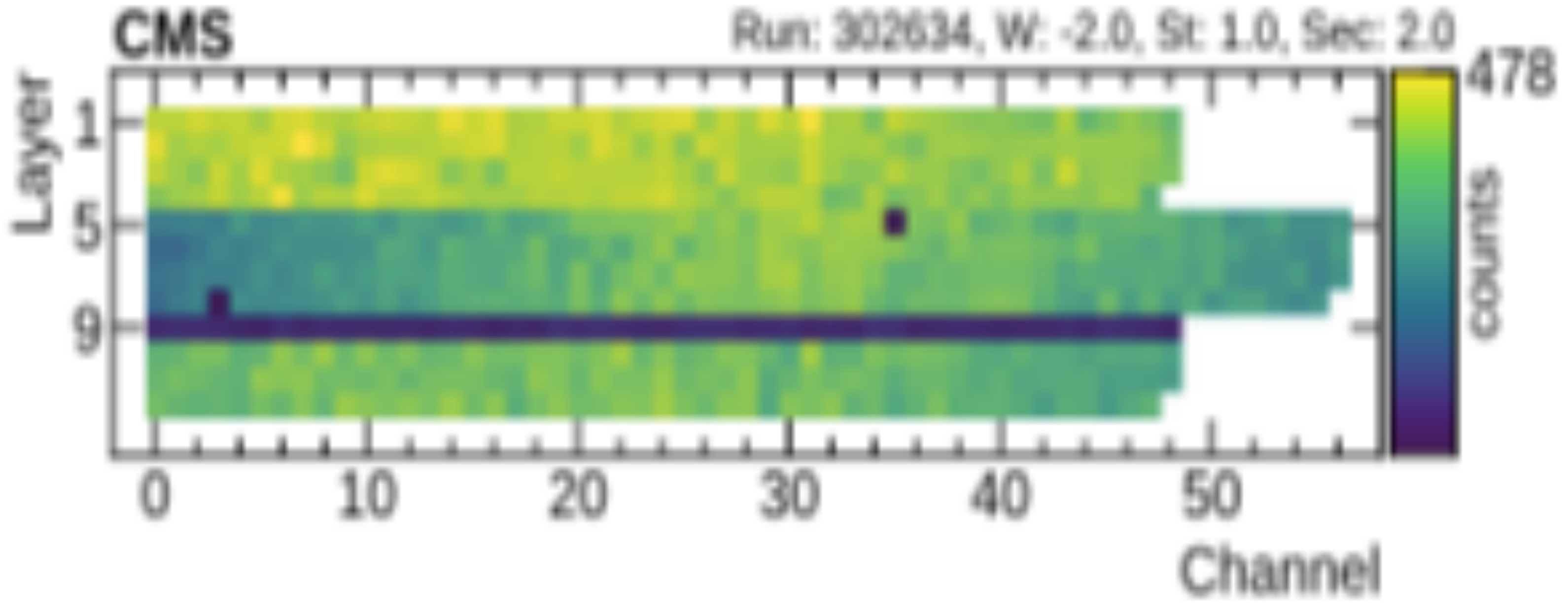
Data Quality Monitoring

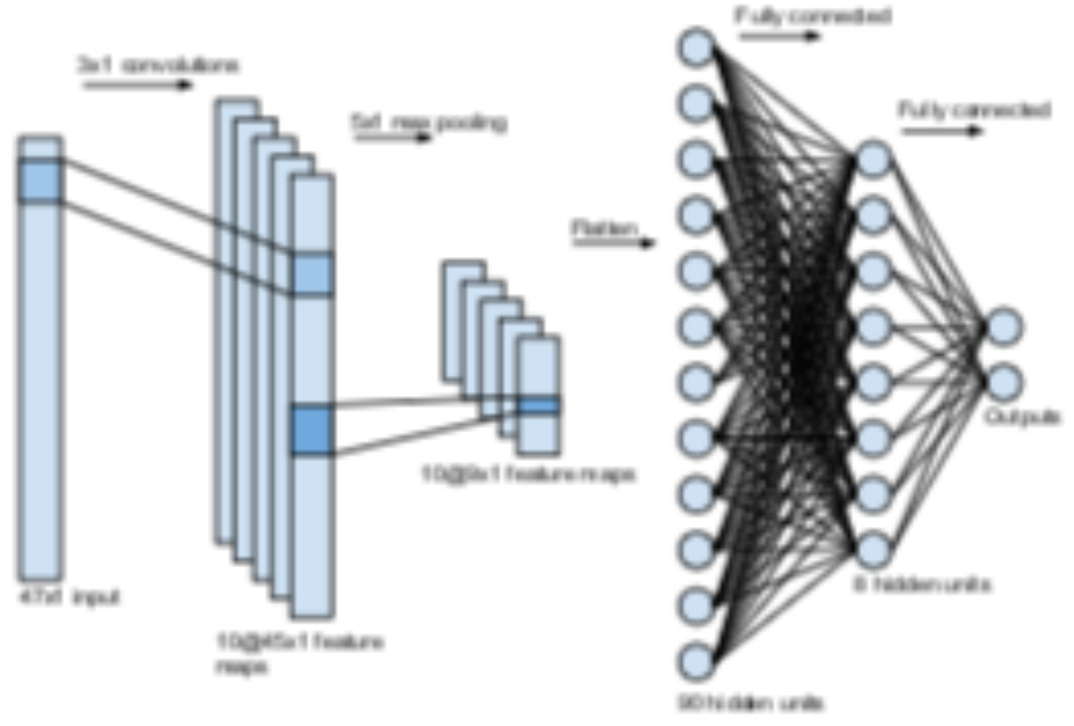
- Unsupervised ML used to spot anomalies

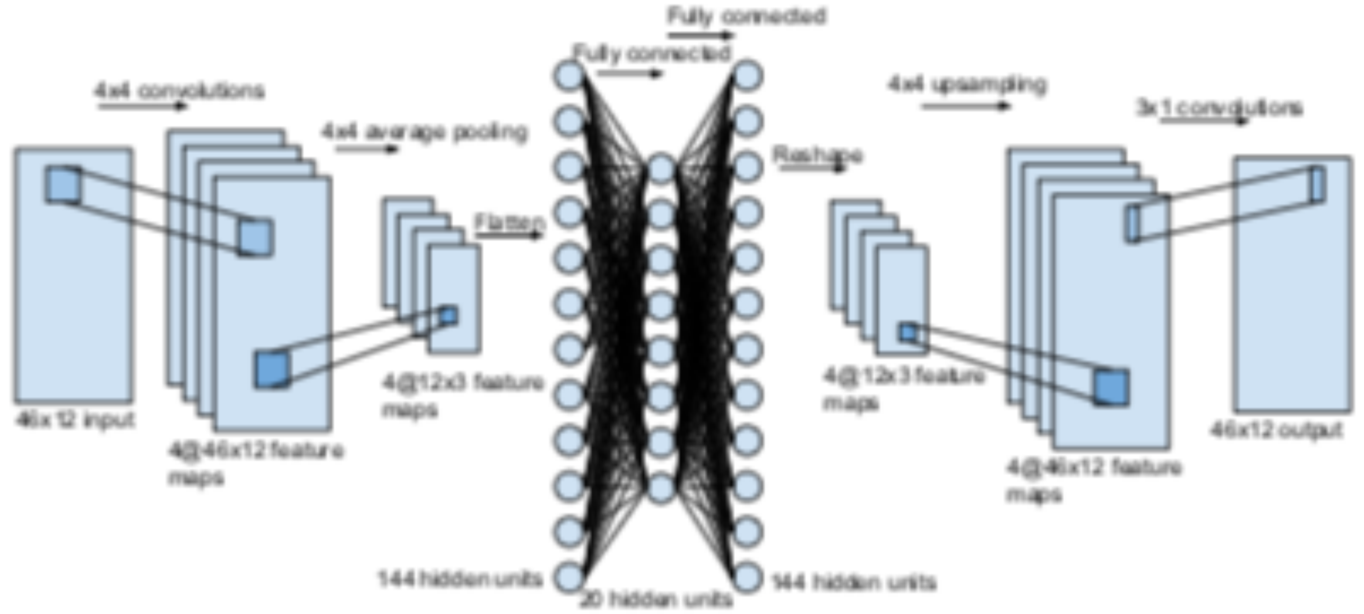
[Pol *et al.*, 2018, arXiv:1808.00911]

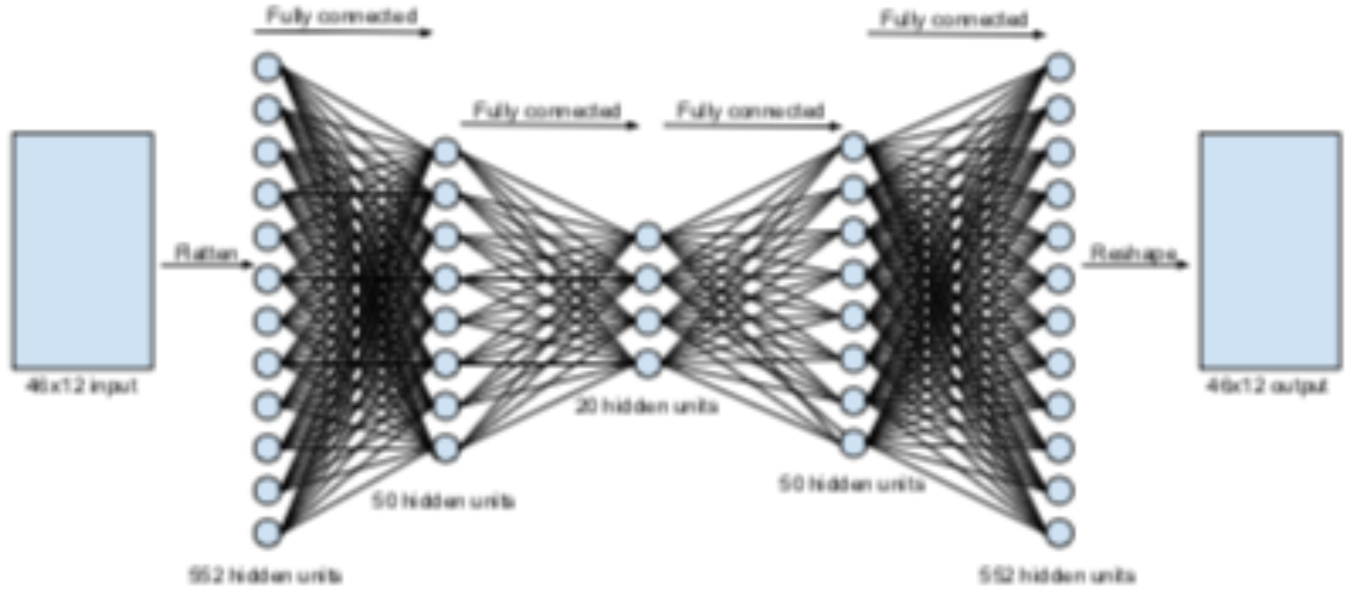


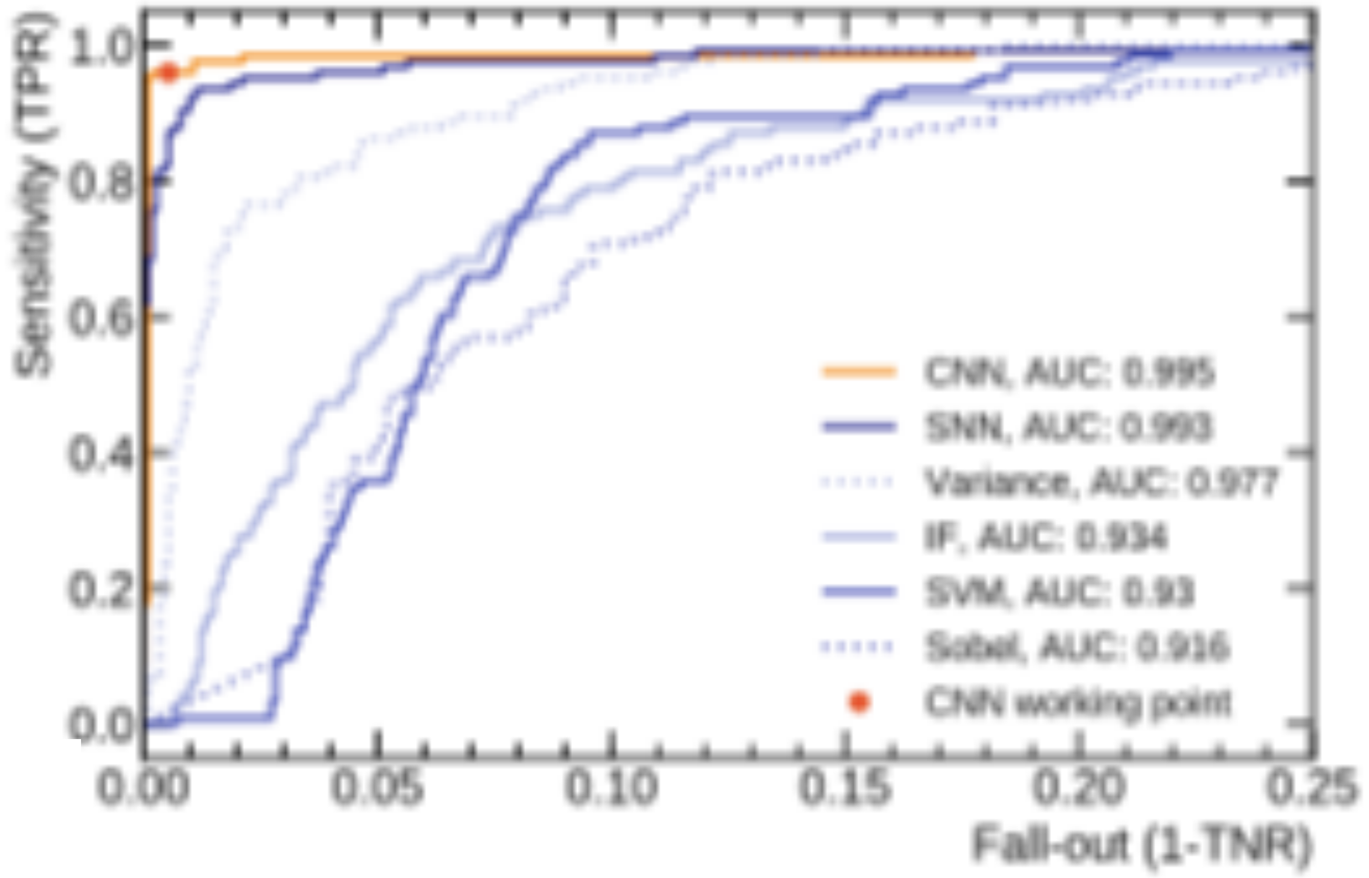


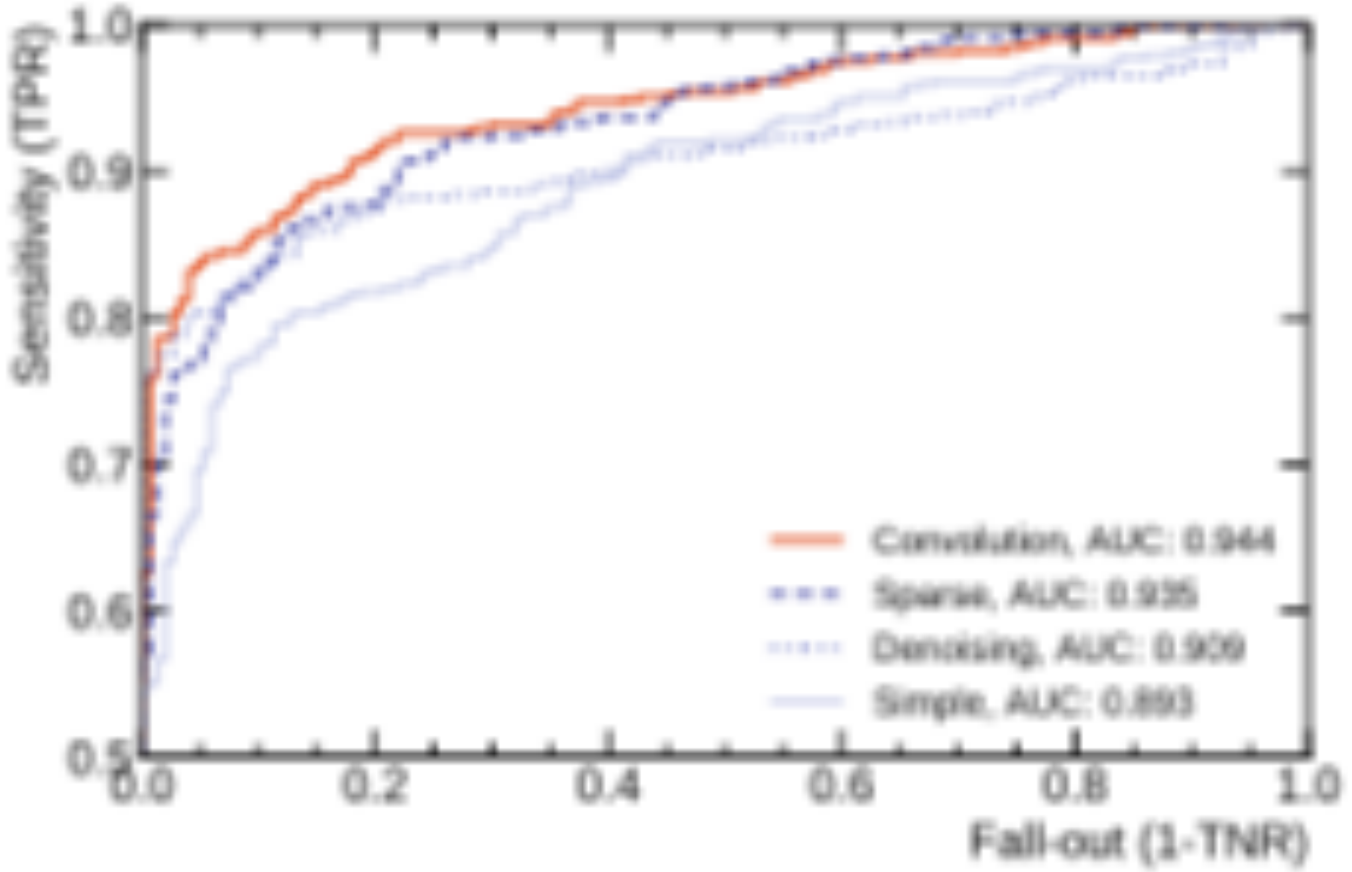








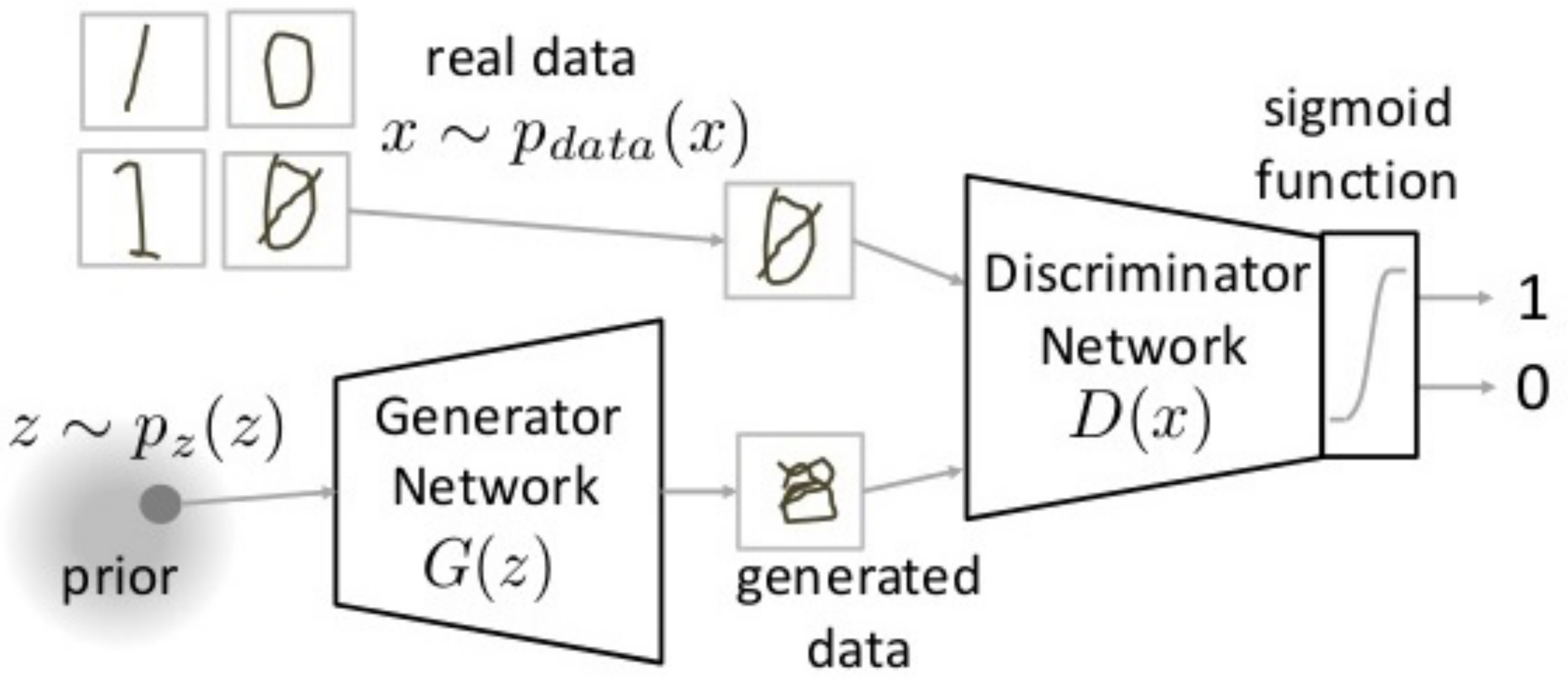




Generative Models

- 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

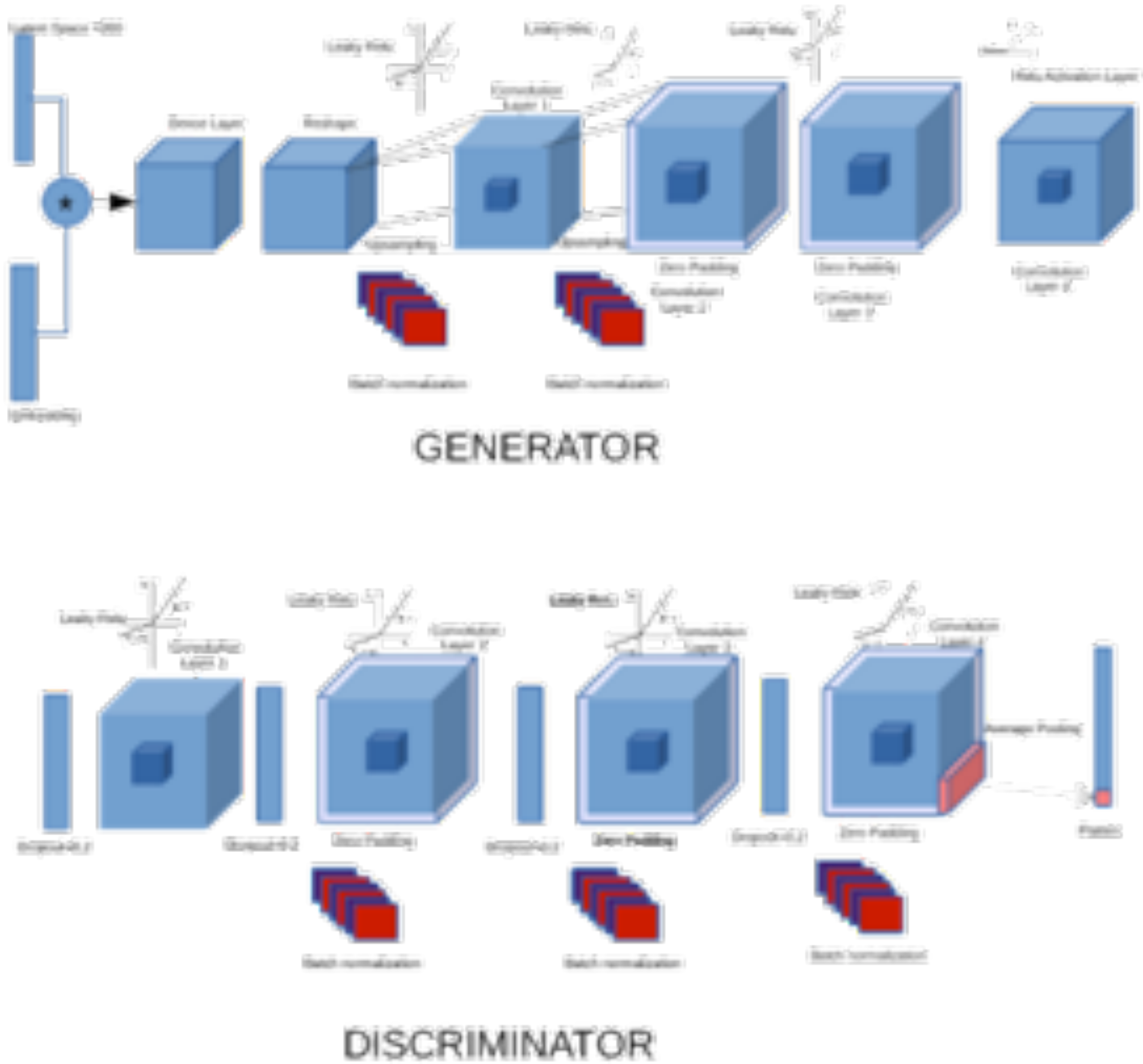


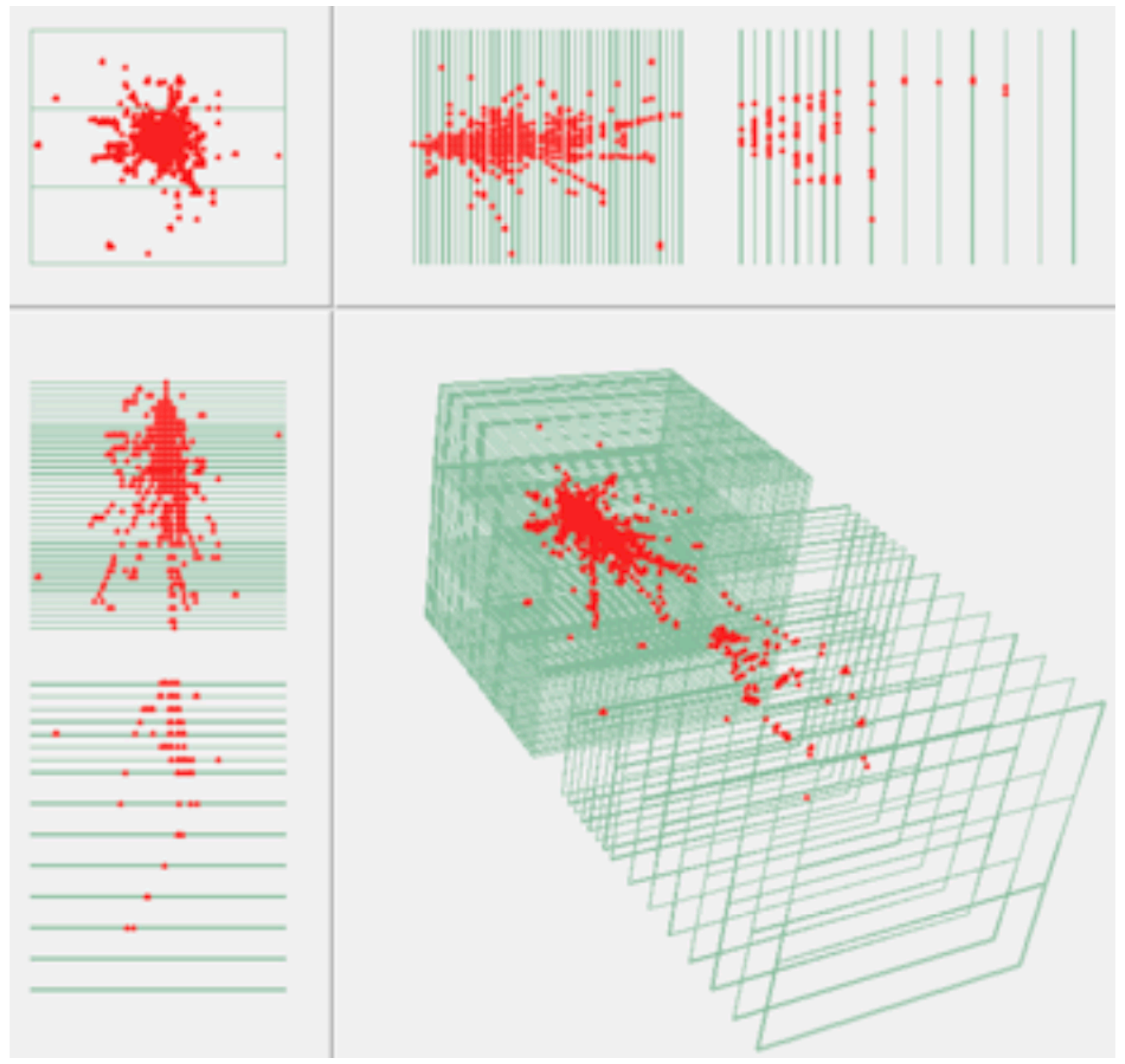
- **Generator network**:
  - output data from a random input G(x)
- **Discriminator network**:
  - discriminate the generated data from real ones
  - output probability D(x) that data are from real input

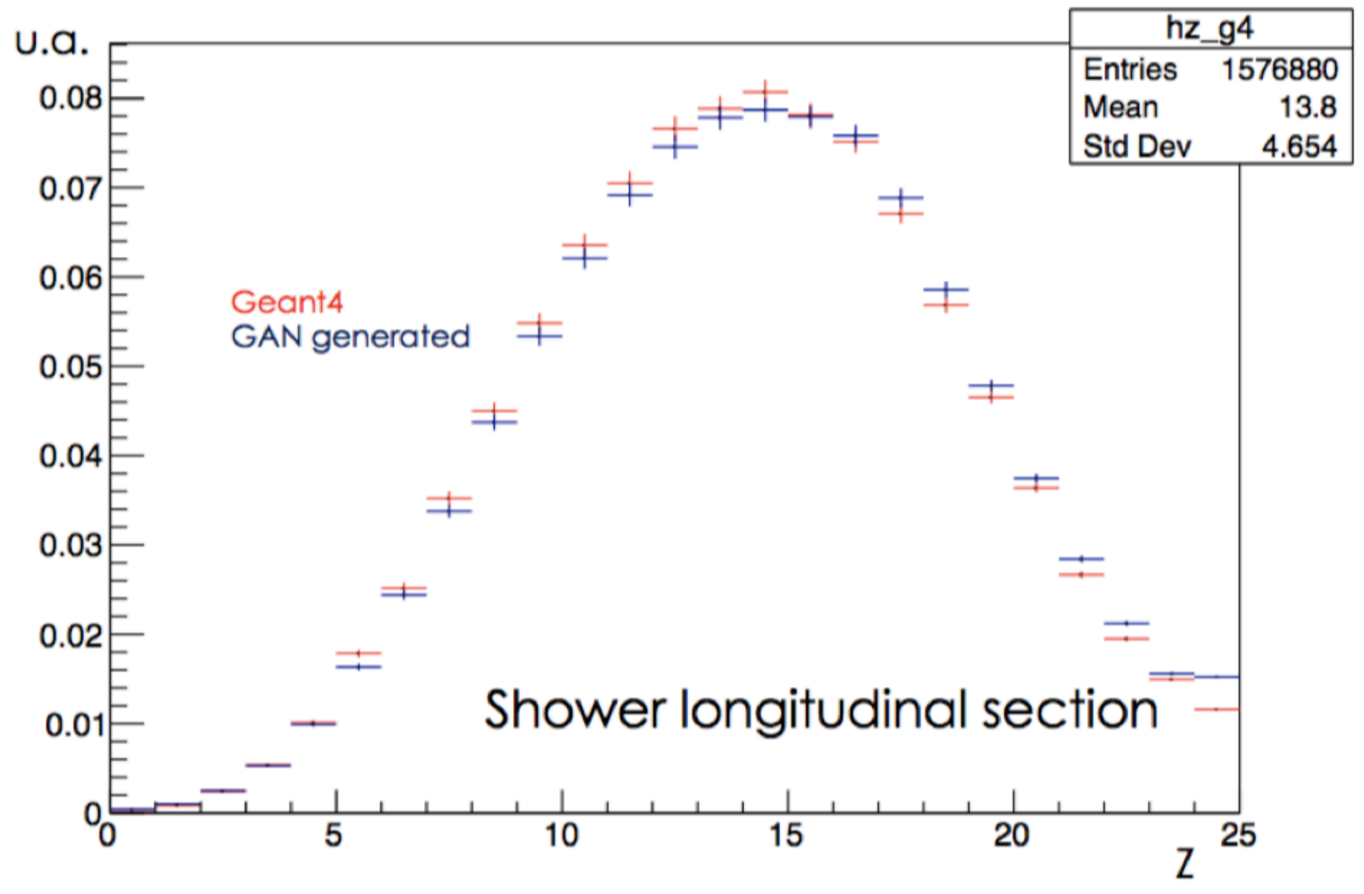
Example: 3d GAN for Calorimeter Images

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]

Interoperability with NumPy

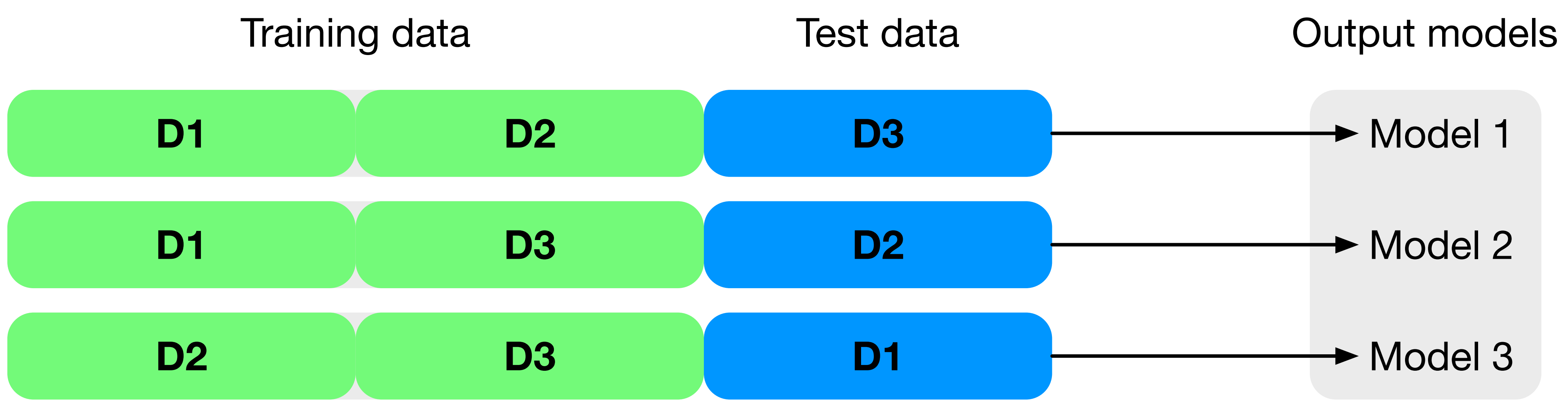
- Better interoperability in ROOT with Data Science Python tools
  - easier conversion to NumPy and Pandas
  - easier to use powerful machine learning Python tools
- Example: Reading ROOT data (TTree's) directly in NumPy arrays

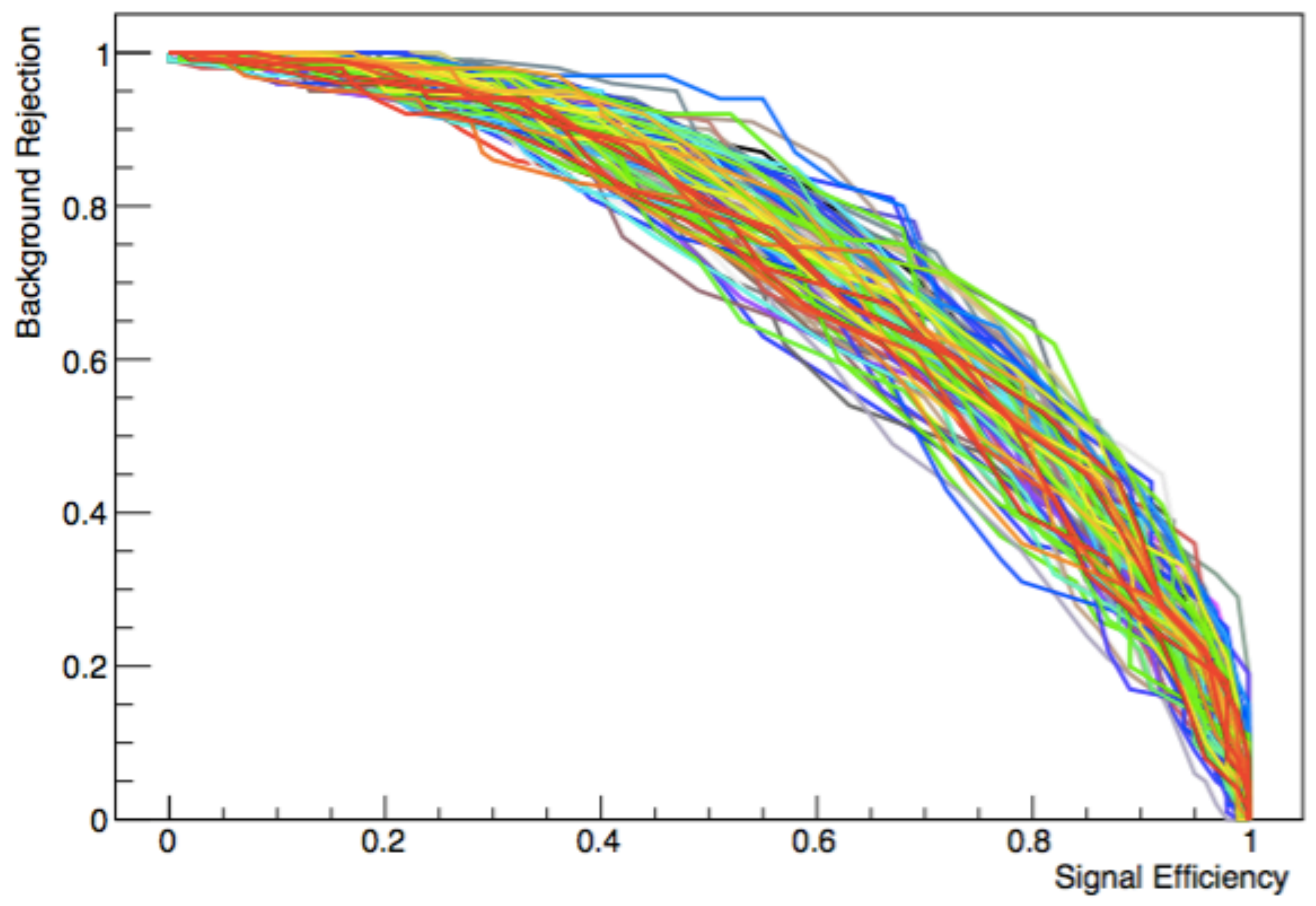
```
myTree # Contains branches x and y of type float

# Convert to numpy array and calculate mean values of all
# branches
myArray = myTree.AsMatrix()
m = np.mean(myArray, axis = 0)

# Read only specific branches
onlyX = myTree.AsMatrix(columns = ['x'])
```

- With **RDataFrame.AsNumpy(['v1','v2','v3'])** also from ROOT data to NumPy arrays

Cross Validation in TMVA

- TMVA supports k-fold cross-validation





  - Integration with TMVA analysis tools (e.g. GUI)
  - support for "CV in application"
- Hyper-parameter tuning
  - find optimised parameters (BDT-SVM)
- Parallel execution of folds in CV
  - using multi-processes execution in on a single node
  - foreseen to provide parallelisation in a cluster using Spark or MPI
- See Kim's presentation

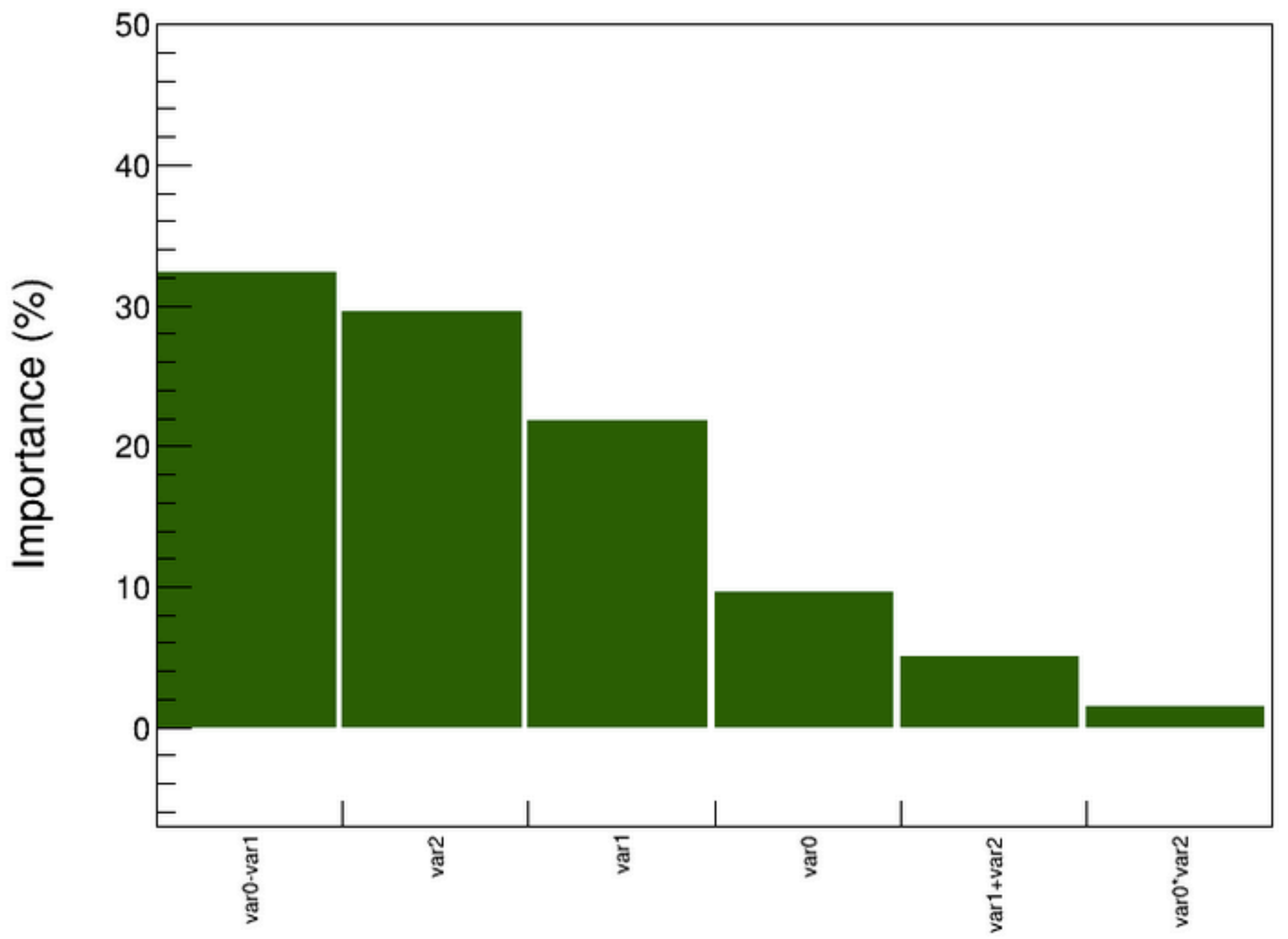
Feature Importance

- Ranks the importance of features based on contribution to classifier performance
  - A stochastic algorithm independent of classifier choice

$$FI(X_i) = \sum_{S \subseteq V : X_i \in S} F(S) \times W_{X_i}(S)$$

$$W_{X_i}(S) \equiv 1 - \frac{F(S - \{X_i\})}{F(S)}$$

- Feature set {V}
- Feature subset {S}
- Classifier Performance F(S)



Future Developments

- Aim is to provide to the users community efficient physics workflows
  - tools for efficient
    - data loading (using new RDataFrame)
    - integration with external ML tools
    - training of commonly used architectures
    - deployment and inference of trained models
- TMVA efficiently connects input data to ML algorithms
  - defining also new functional user interfaces

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

Conclusions

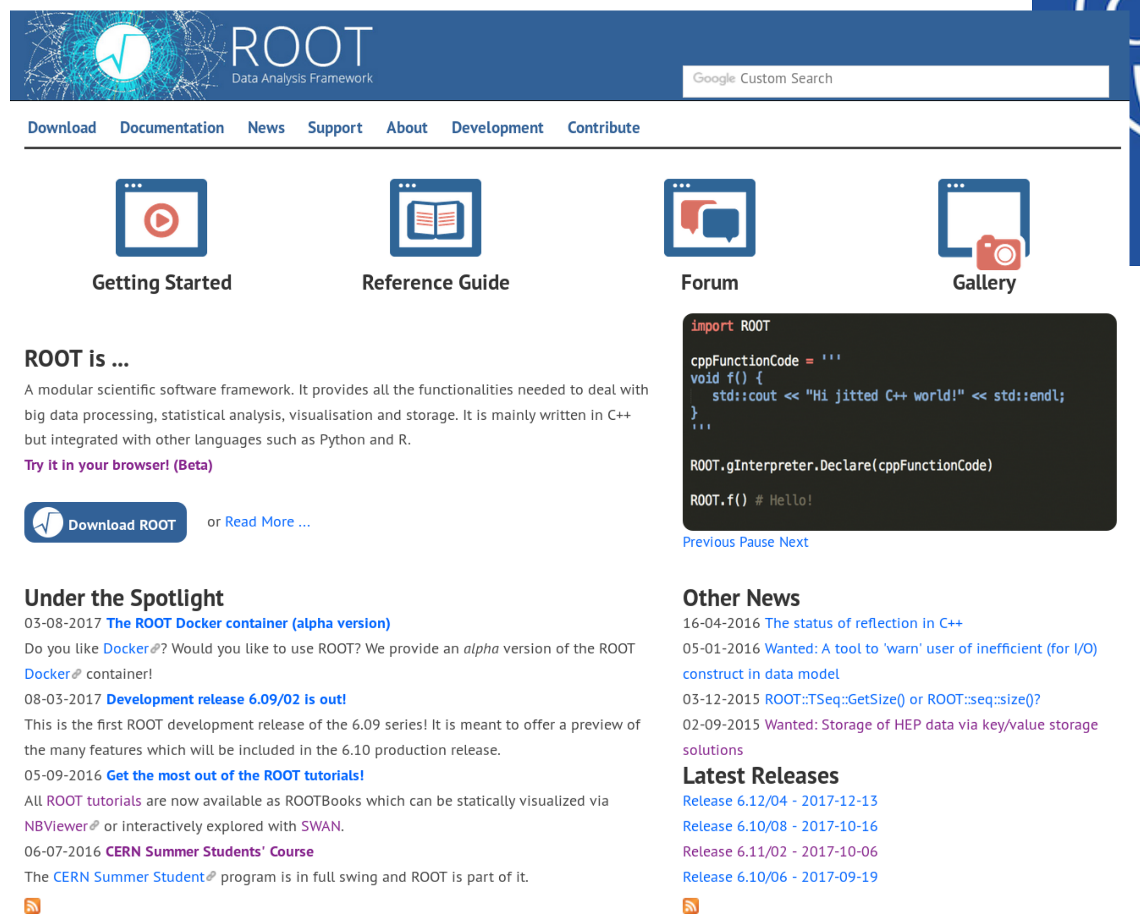
- ROOT /TMVA provides several Machine Learning tools for data analysis
  - direct connection to ROOT I/O data structures
  - provides several **new features** (modern ML algorithms)
  - excellent performances (training and inference of models)
    - parallelisation, GPU, faster I/O, etc.
- **Long term support and stability**
- **Large user community**
- **ROOT is an open source project**

TMVA Examples

- Notebooks in C++ or Python
  - https://swan.web.cern.ch/content/machine-learning
  - https://github.com/lmoneta/tmva-tutorial
- C++ examples:
  - https://root.cern.ch/doc/master/group__tutorial__tmva.html

ROOT

- 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



ROOT Users' Workshop:

https://cern.ch/root2018