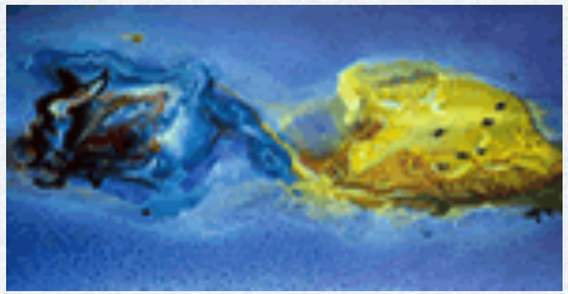
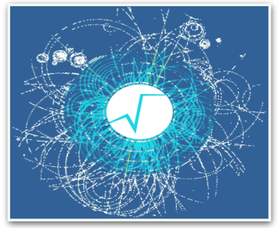
New Machine Learning Developements in ROOT/TMVA

S. Gleyzer, L. Moneta (CERN EP-SFT)
on behalf of the TMVA development team



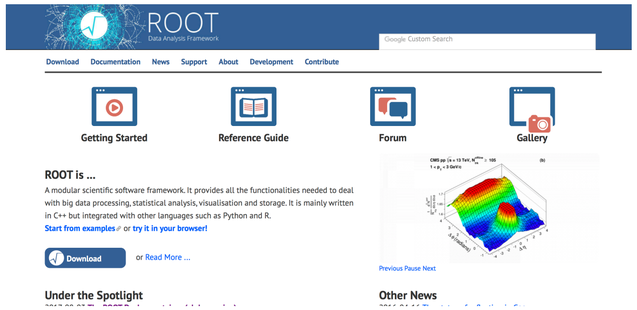
XIIIth Quark Confinement and the Hadron Spectrum

31 July to 6 August 2018
Maynooth University (Ireland)



ROOT

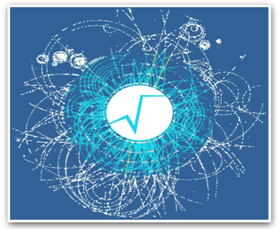
- ROOT is a software toolkit which provides building blocks for:
- Data processing
- Data analysis
- Data visualisation
- Data storage



- ROOT is written mainly in C++ (C++11 standard)
- Bindings for Python are provided.

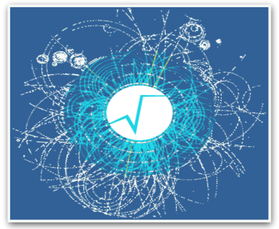
http://root.cern.ch

- Adopted in High Energy Physics and other sciences (but also industry)
- ~250 PetaBytes of data in ROOT format on the LHC Computing Grid
- Fits and parameters' estimations for discoveries (e.g. the Higgs)
- Thousands of ROOT plots in scientific publications



TMVA

- ROOT Machine Learning tools are provided in the package TMVA (Toolkit for MultiVariate Analysis)
- Provides a set of algorithms for standard HEP usage
- Used in LHC experiment production and in several analyses
- Key features
  - Facilitates HEP research, from detector to analysis
  - Easy to use, good performance
  - Long term support
  - Several features added recently (e.g. deep learning)
- Development done in collaboration with CERN experiments and HEP community
  - see HSF community Machine Learning white paper

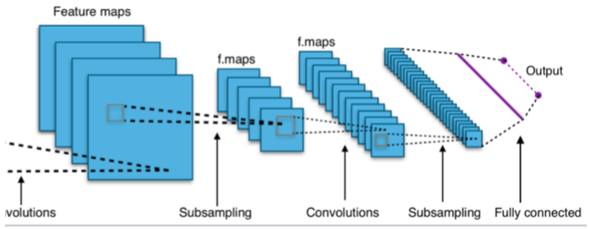


Outline

- Machine learning methods
  - Status of deep learning library
  - BDT parallelisation
- Workflow improvements
  - Cross validation
- A look into the future

Most Recent Features

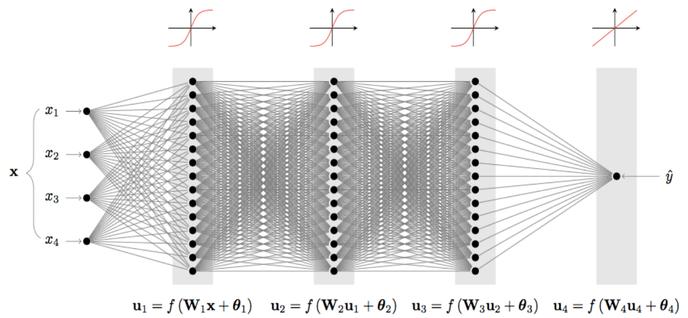
- New features available in latest ROOT version 6.14:
  - Deep Learning Module with
    - Convolutional Layer
    - Recurrent Layer
  - improved Cross Validation
  - improved BDT performance using multi-thread parallelisation

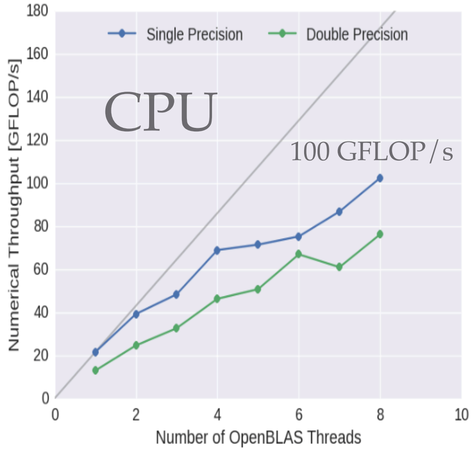


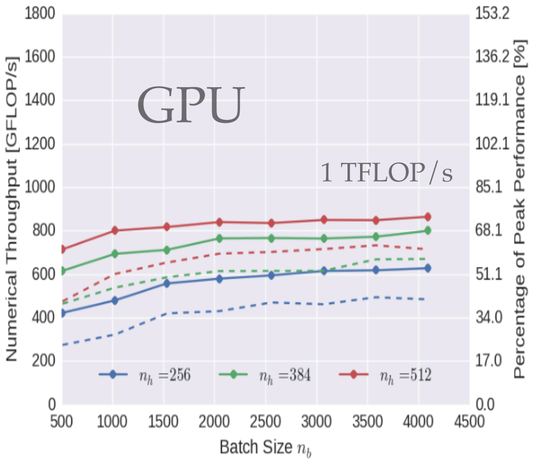
- And also available since ROOT 6.12:
  - new interfaces to external tools (scikit-learn, Keras, R)
  - hyper-parameter optimization and variable importance
  - interactive training and visualisation for Jupyter notebooks

Deep Learning in TMVA

- Deep Learning library in ROOT/TMVA
  - parallel evaluation on CPU
    - implementation using OpenBlas and Intel TBB library
  - GPU support using CUDA
  - Excellent performance and high numerical throughput
- For more information see
  - https://indico.cern.ch/event/565647/contributions/2308666/attachments/1345668/2028738/tmva_dnn_gpu.pdf

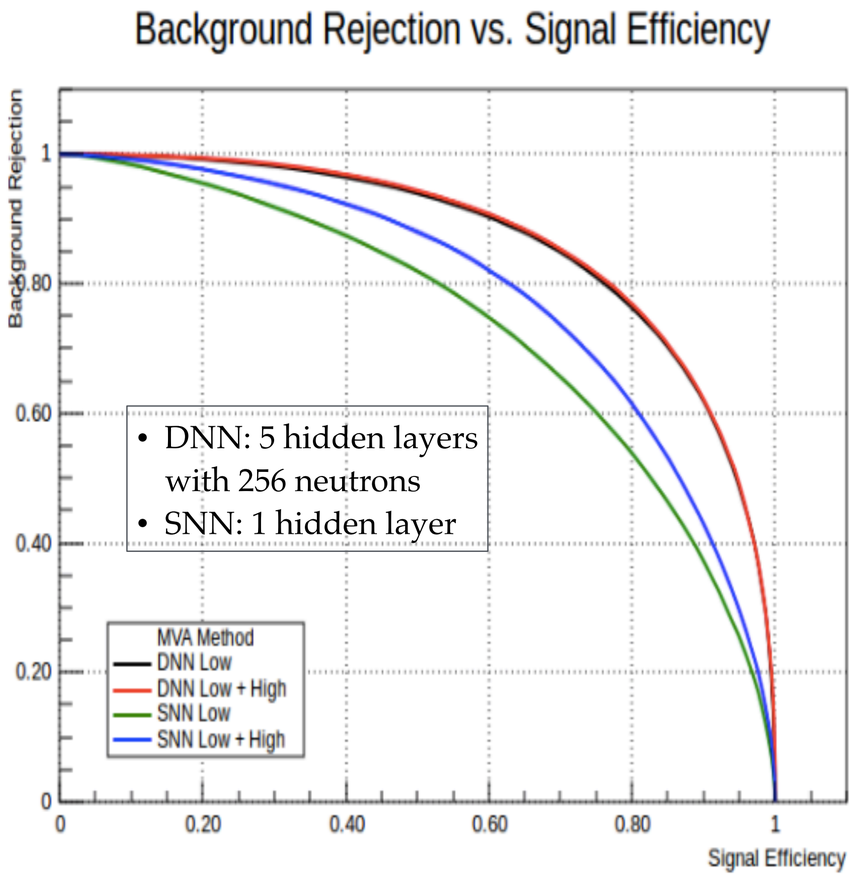




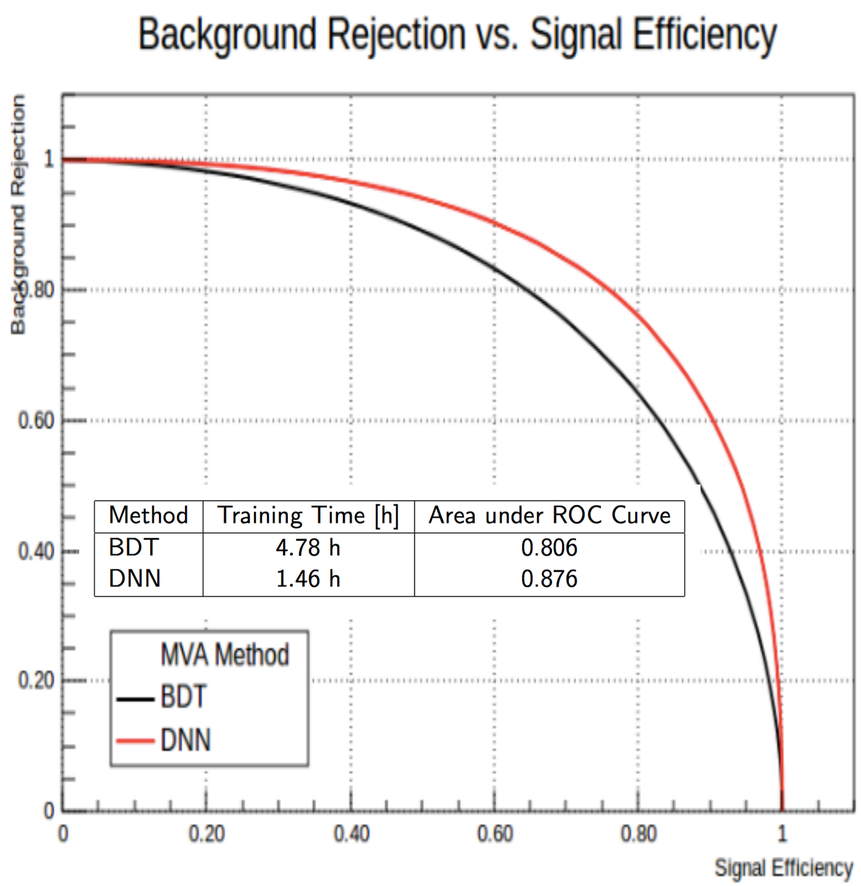


Deep Learning Performance

DNN vs Standard ANN



DNN vs BDT



- Using Higgs public dataset (from UCI) with 11M events
- Significant improvements compared to shallow networks and BDT

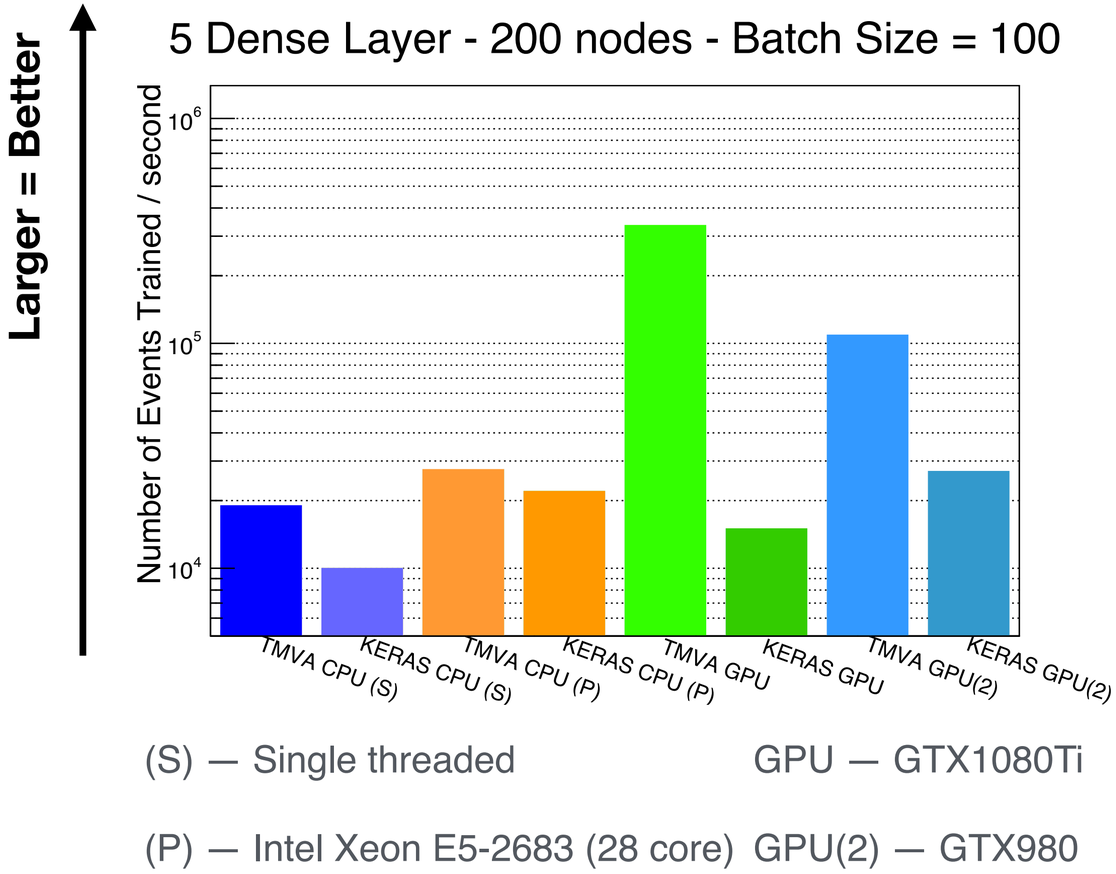
DNN Training Performance

Training time — Dense networks

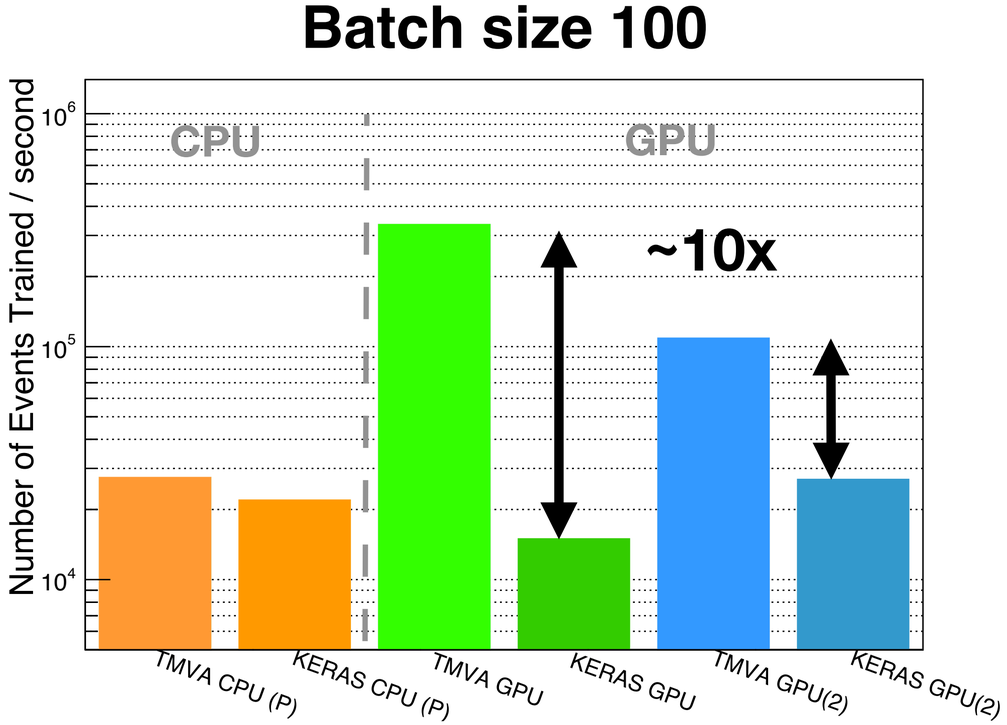
- Higgs UCI dataset with 11M Events
- TMVA vs. Keras/Tensorflow
- "Out-of-the-box" performance

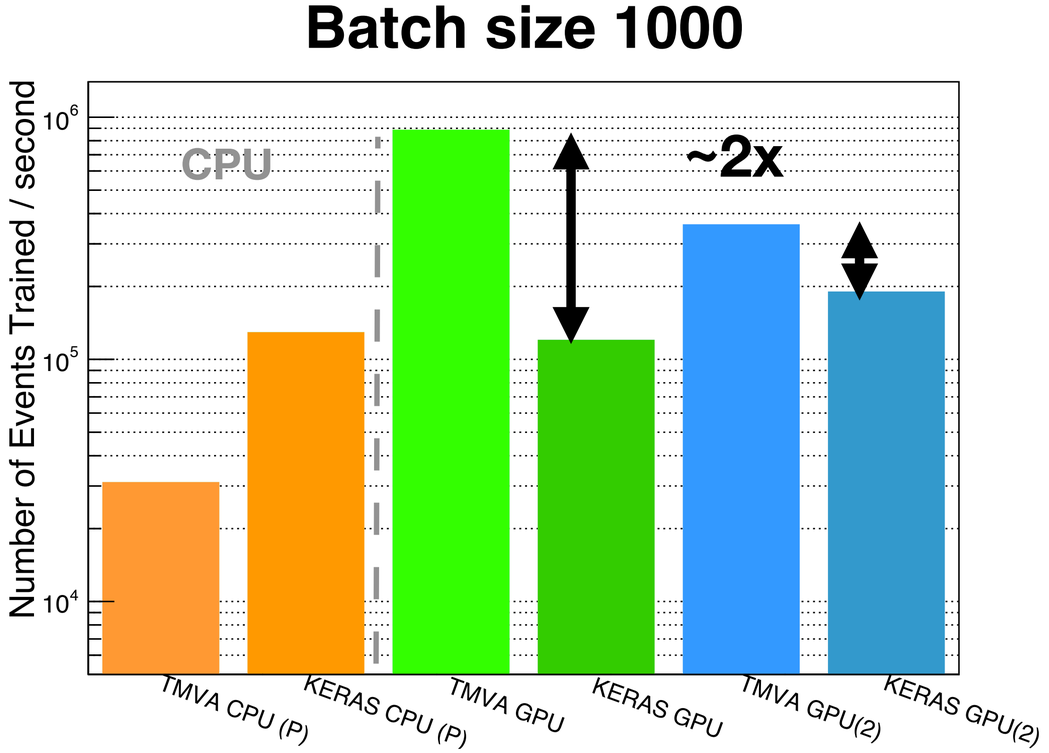
Excellent TMVA performance !

- How does it scale?



DNN Training Performance





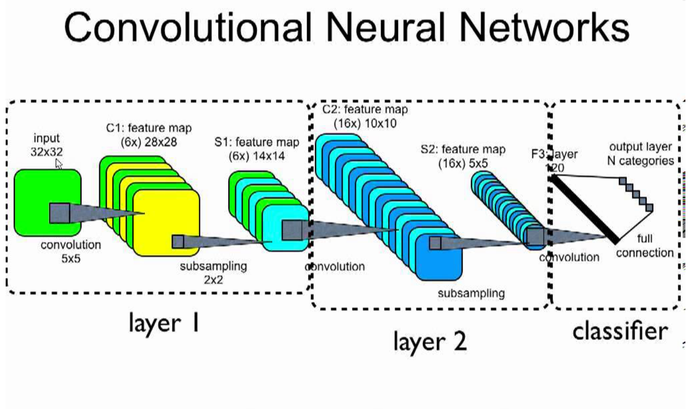
- Key difference is GPU utilisation
  - Tensorflow optimised for large operations

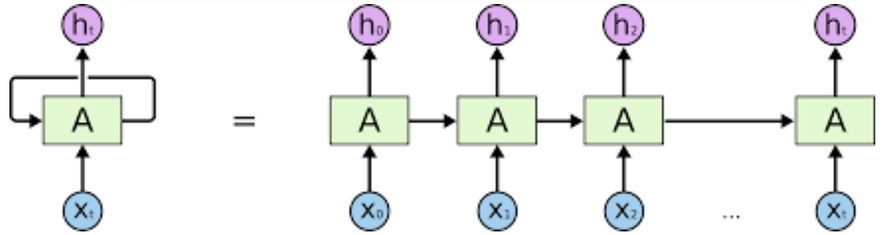
(S) — Single threaded GPU — GTX1080Ti

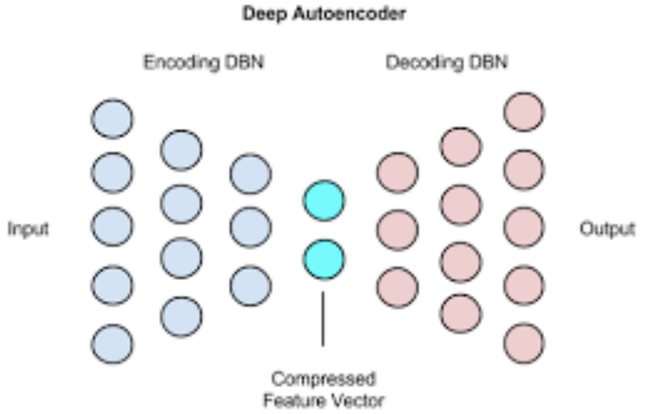
(P) — Intel Xeon E5-2683 (28 core) GPU(2) — GTX980

New Deep Learning Developments

- Extended Deep Neural Network classes by adding:
  - Convolutional Neural Network
    - very powerful for image data sets
  - Recurrent Neural Network
    - useful for time-dependent data
  - Deep Auto Encoder
    - useful for dimensionality reduction (pre-processing tool)
    - can be used as unsupervised tool (e.g. for anomaly detection)
    - usable also for generating models, Variational Auto Encoder (VAE)
  - Generative Adversarial Network (GAN)

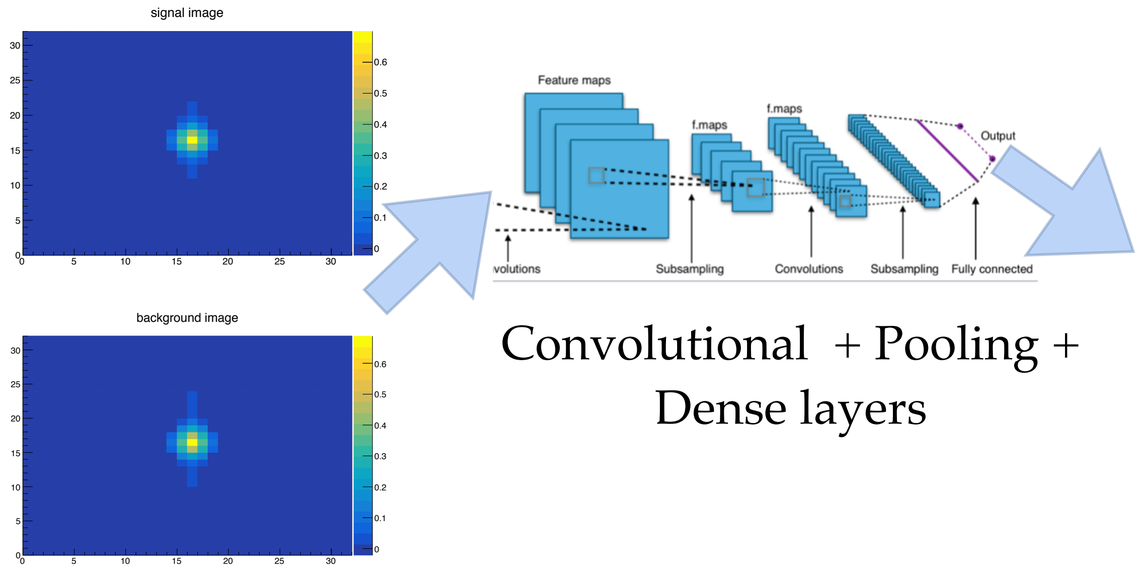






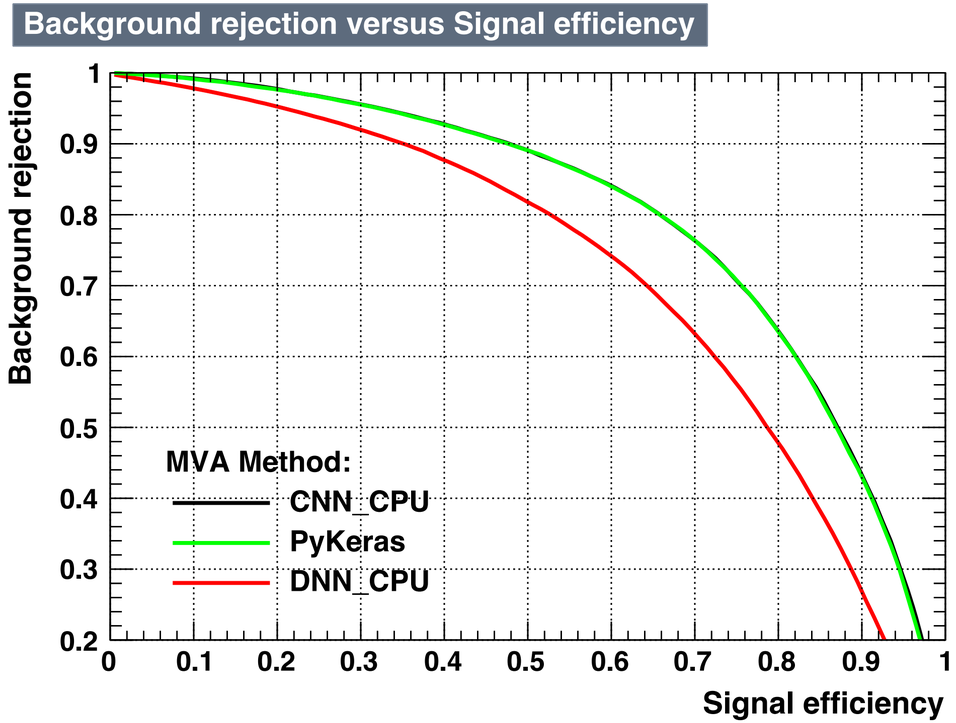
Convolutional Neural Network

- Available in latest ROOT release (6.14)
- Supporting CPU parallelization, GPU is available since few days in ROOT master
  - parallelisation and code optimisation is essential



input 32x32 images

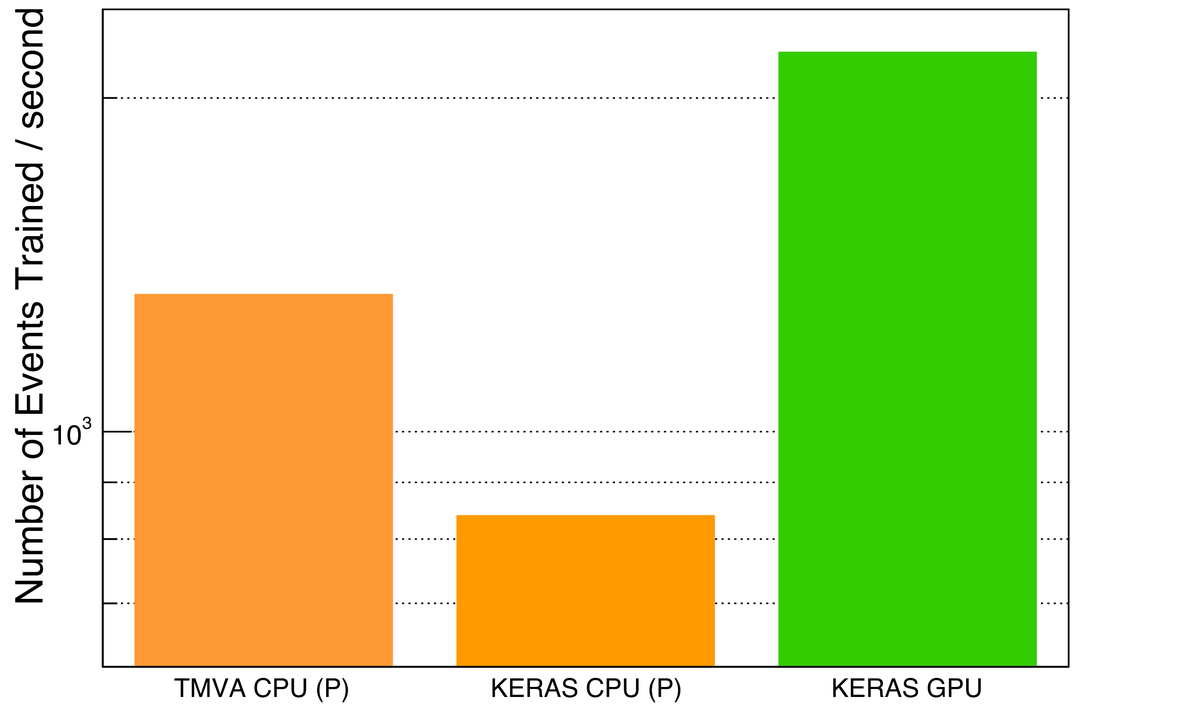
Convolutional + Pooling + Dense layers



CNN Training Performance

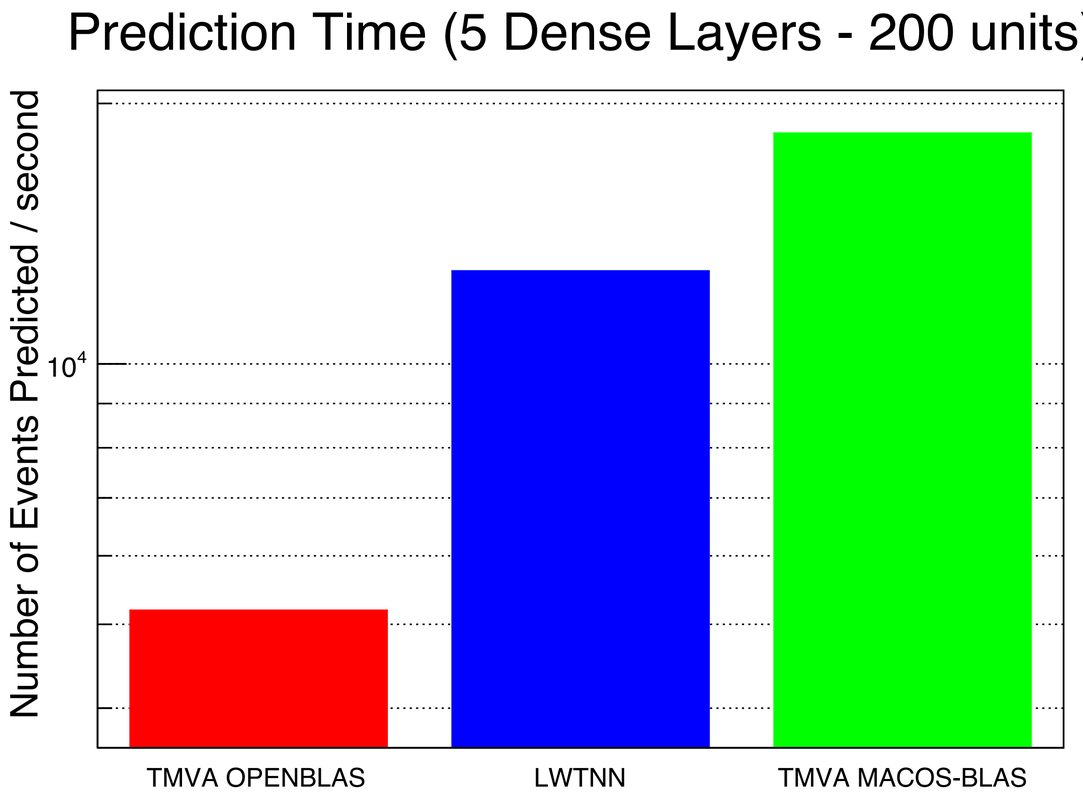
CNN performance for TMVA CPU

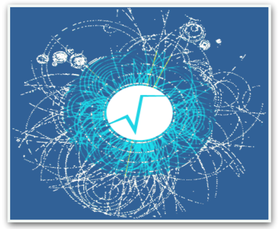
- Simulated particle showers from electromagnetic calorimeter image dataset
- again excellent TMVA performance for typical HEP networks !
- TMVA GPU is becoming available. Code is in the process of being optimised



DL Evaluation Performance

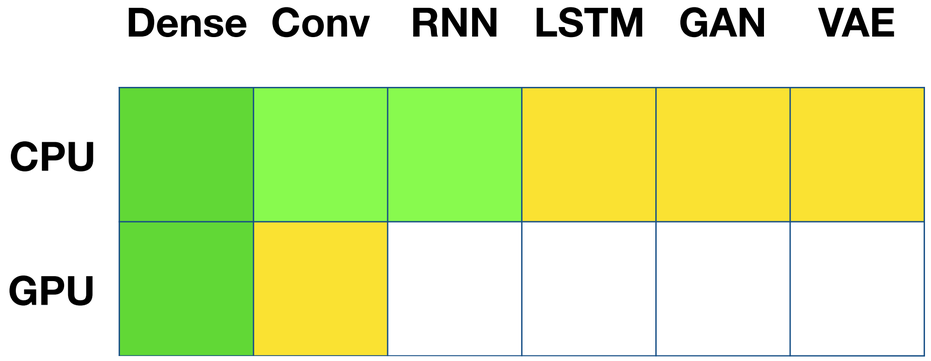
- **Single event evaluation time** for 5 layer network
  - For time critical applications — e.g. on-line reconstruction
  - Fast! 1.5 times speedup over specialised libraries like LWTNN
- For batched evaluation, same story as training





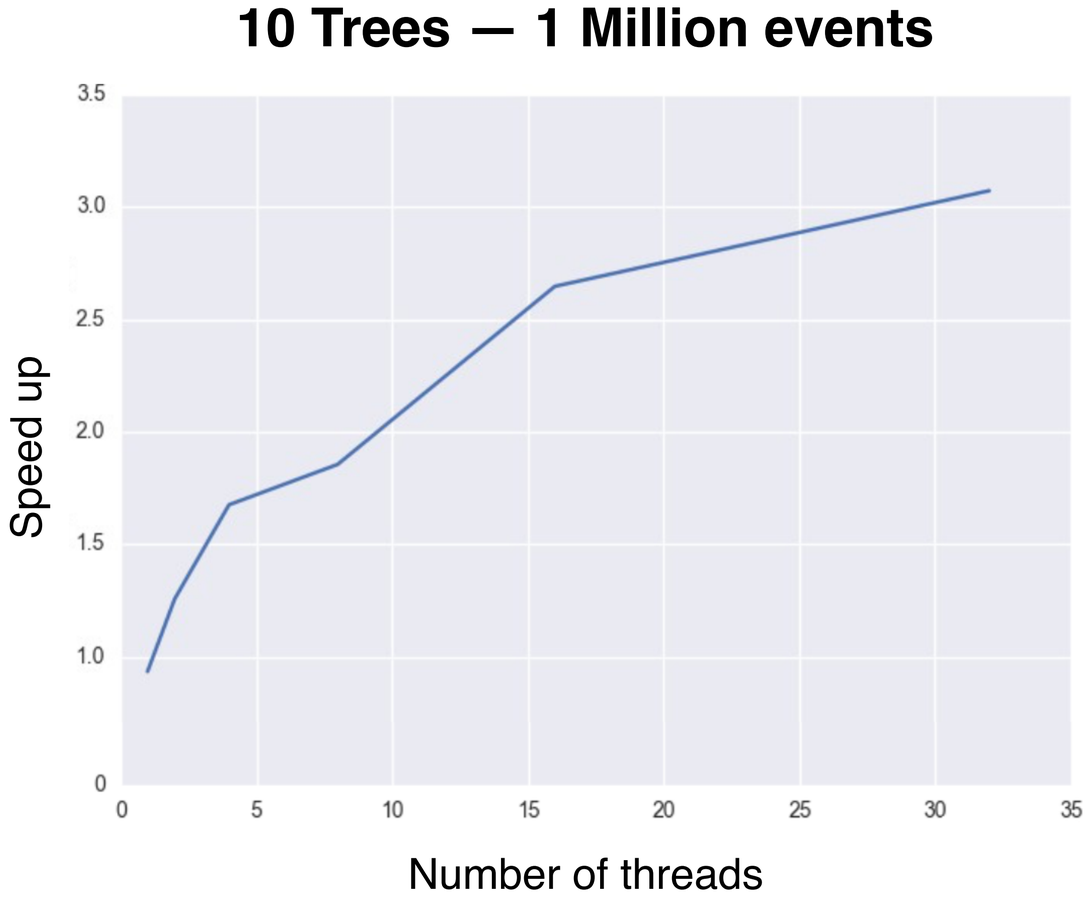
Status Deep Learning library

- Deep learning library since 2016
- Recent additions
  - Convolutional and recurrent layers
- Development ongoing!
  - LSTM (and also GRU) layers
  - GAN and VAE for generation
  - new optimisers complementing SGD



Boosted Decision Tree

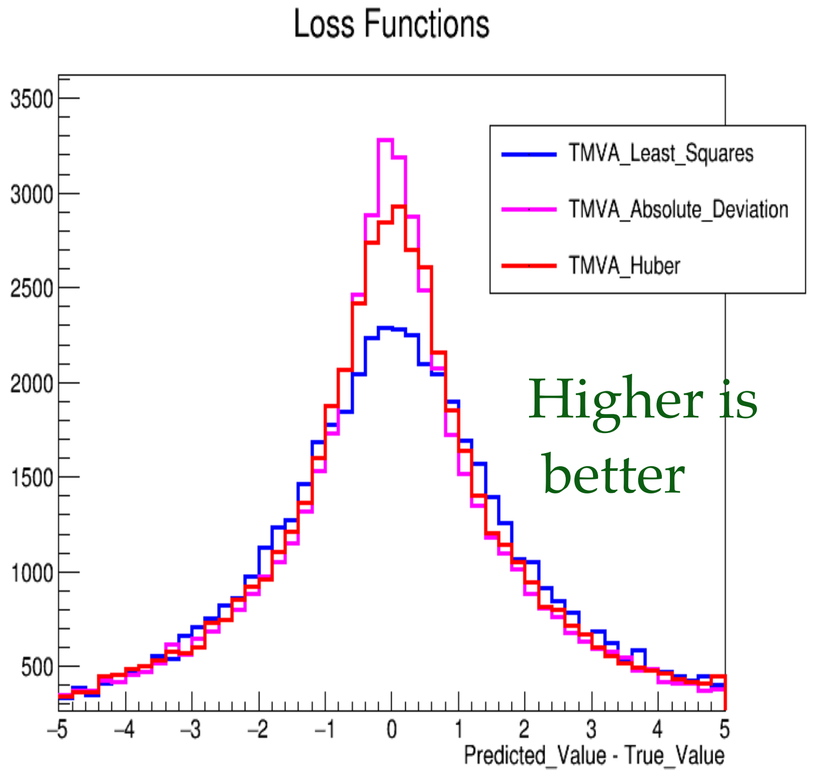
- Boosting is serial ⟶ Can't construct all trees in parallel
- Training time speed up ~1.6x with 4 threads approaching ~3x asymptotically
- To use, just add
  ROOT::EnableImplicitMT()
  to your code



Original slide by Andrew Carnes

Regression in TMVA

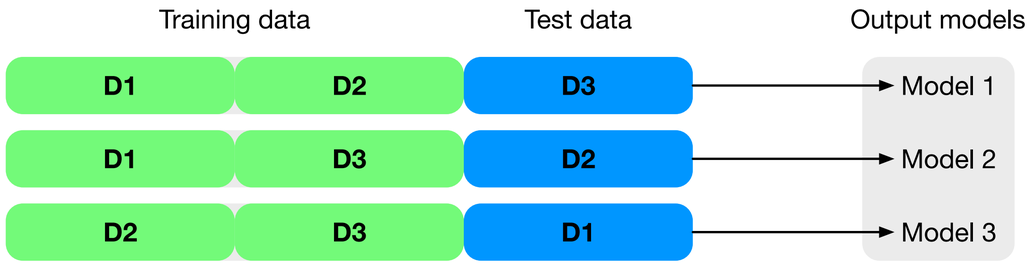
- **New Regression Features**:
  - Loss function
    - Huber (default)
    - Least Squares
    - Absolute Deviation
    - Custom Function

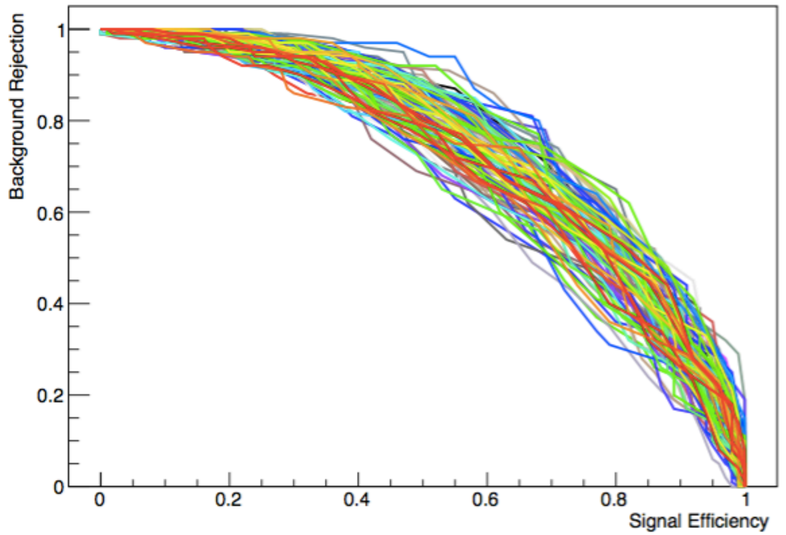


Important for regression performance

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

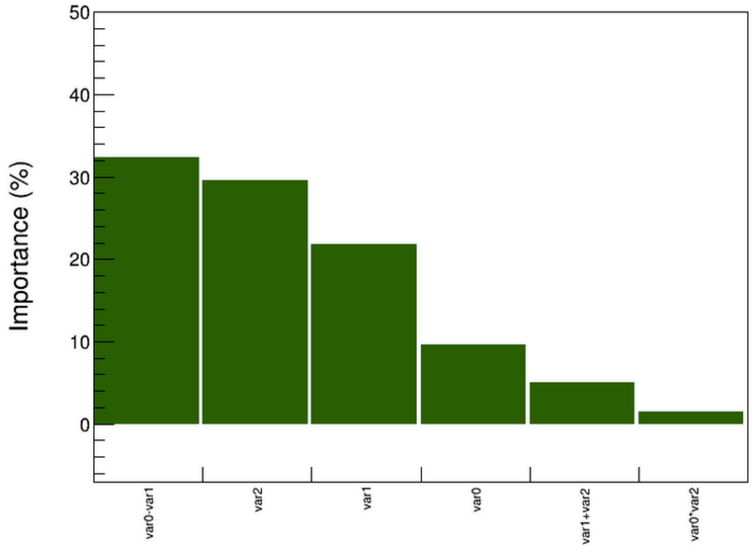
Feature Importance

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

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

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

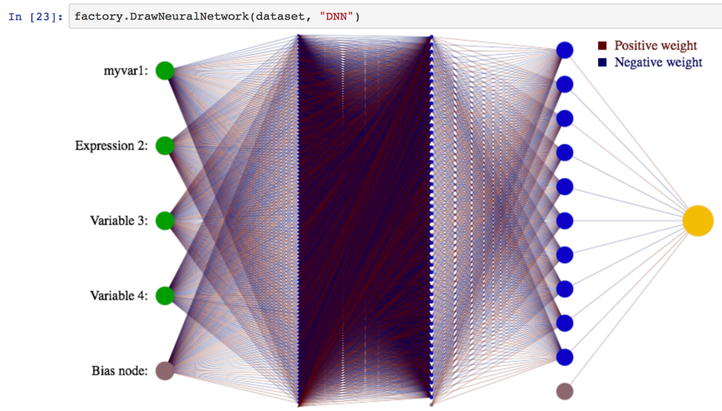
- Feature set {V}
- Feature subset {S}
- Classifier Performance F(S)

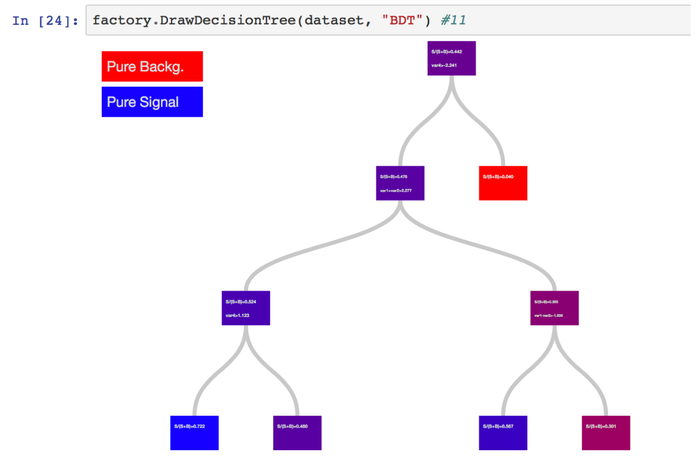


TMVA Jupyter Integration

New Python package for using TMVA in Jupyter notebook (jsmva)

- Improved Python API for TMVA functions
- Visualisation of BDT and DNN
- Enhanced output and plots (e.g. ROC plots)
- Improved interactivity (e.g. pause/resume/stop of training)
- see example in SWAN gallery
  https://swan.web.cern.ch/content/machine-learning

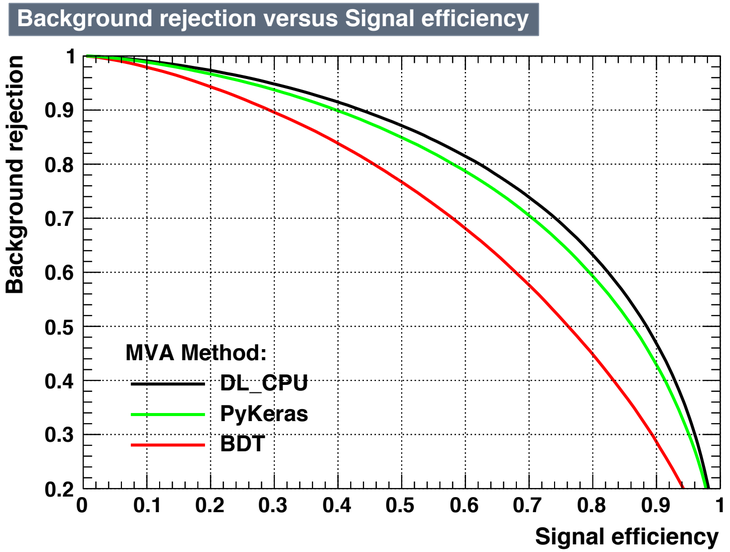




TMVA Interfaces

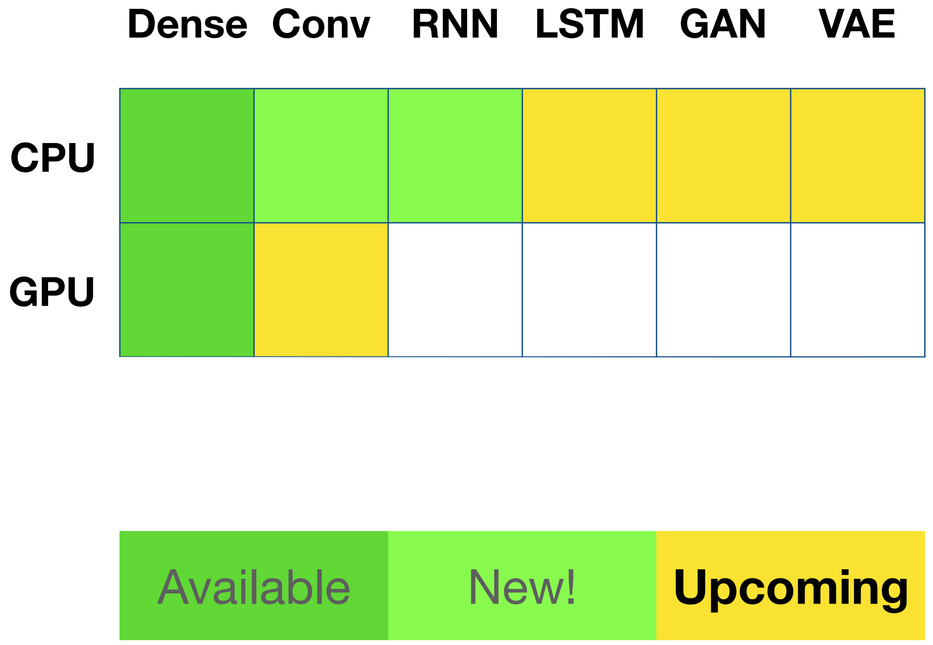
External tools are available as additional methods in TMVA and they can be trained and evaluated as any other internal ones.

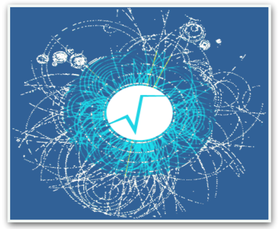
- **RMVA**: **Interface to Machine Learning methods in R**
  - c50, xgboost, RSNNS, e1071
- **PYMVA**: **Python Interface**
  - **scikit-learn**
    - with RandomForest,
      Gradiend Tree Boost, Ada Boost
  - **Keras** (Theano + Tensorflow)
    - support model definition in Python and then training and evaluate in TMVA
  - input data are copied internally from input ROOT trees to Numpy arrays



Future — Short term

- **Complete Deep learning library**
  - Conv layers on GPU
  - Add LSTM, GRU, GAN, VAE
  - And more!
- **Workflow improvements**
  - Modernised interfaces
  - integration with ROOT RDataFrame for better input data handling
  - optimize network evaluation
    - inference of trained Tensorflow models using native TMVA

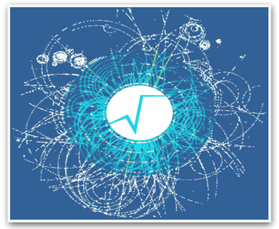




Future — Long term

- Lesson from the HSF Community white paper — Efficient physics workflows
- Provide tools for efficient
  - data loading (e.g. using new RDataFrame)
  - integration with external ML tools
  - deployment and inference of trained models
- TMVA efficiently connects input data to ML algorithms

HSF Community white paper — https://hepsoftwarefoundation.org/organization/cwp.html



Summary

Machine learning methods

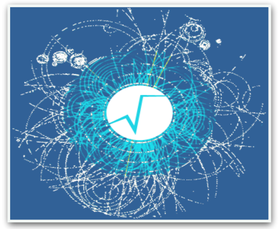
- Dense, Convolutional and Recurrent networks in TMVA
- Excellent training + evaluation time performance
- Parallel boosted decision trees and improved trees for regression

Workflow

- Cross validation analysis + parallelisation

Future

- Efficient physics workflows

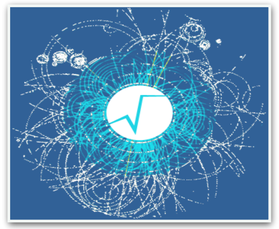


Conclusions

- Very active development happening in TMVA
  - several new features released recently and more expected for next release
  - thanks to many student contributions (e.g. from Google Summer of Code)
- Users contributions and feedback from users are essential
  - ROOT is an open source project
  - best way to contribute is with Pull Request in GitHub

    https://github.com/root-project/root
- For user support we have the ROOT Forum :

    https://root.cern.ch/phpBB3/
  - with a category dedicated to TMVA
- For reporting ROOT bugs: https://sft.its.cern.ch/jira or just contact us directly



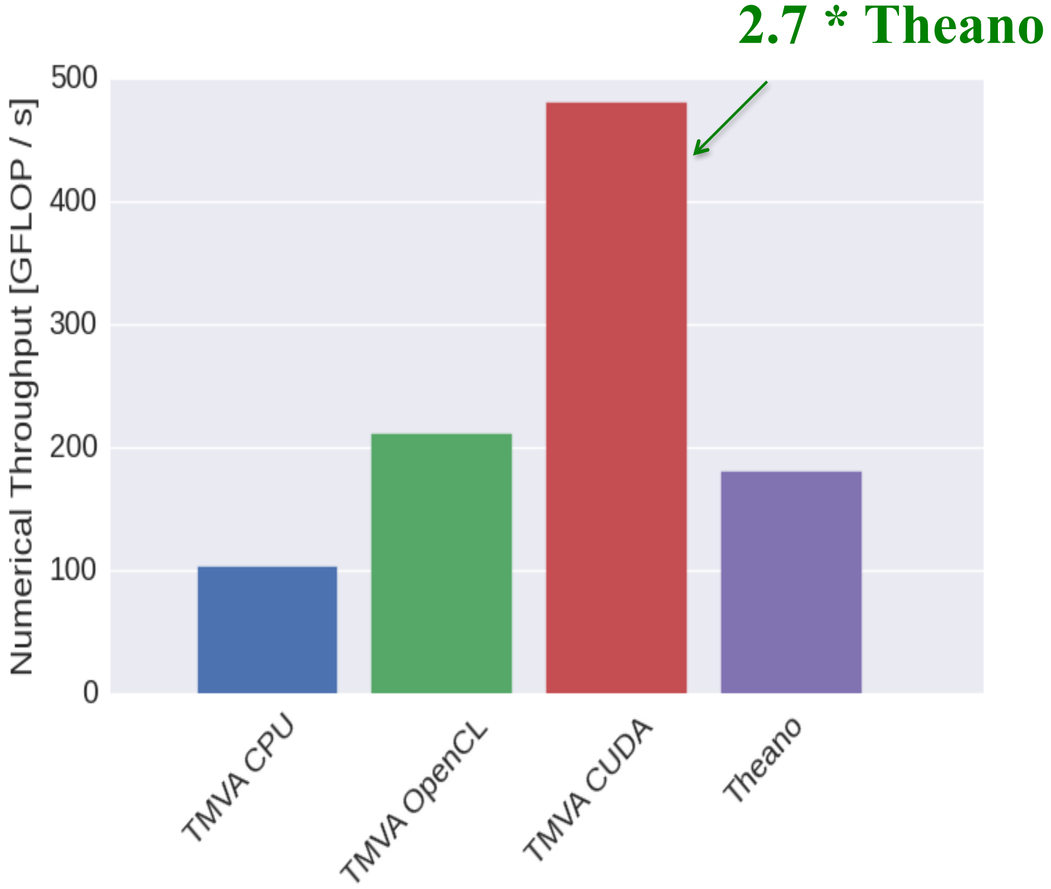
TMVA Contributors

Lorenzo Moneta	Algorithm development, Integration and support
Sergei Gleyzer	Analyzer Tools, Algorithm Development
Omar Zapata Mesa	PyMVA, RMVA, Modularity, Parallelization and Integration
Kim Albertsson	Multi-class for BDT, cross validation/evaluation and support
Stefan Wunsch	Keras Interface, integration, improved data handling
Peter Speckmeyer	Deep Learning CPU
Simon Pfreundschuh (GSOC 2016)	Deep Learning CPU and GPU
Vladimir Ilievski (GSOC 2017)	New Deep Learning module, Convolutional layers
Saurav Shekkar (GSOC 2017)	New Deep Learning module and Recurrent layers
Manos Stergiadis (GSOC 2018)	GPU support for CNN
Ravi Selvam (GSOC 2018)	New optimisers for deep learning
Adrian Bevan, Tom Stevenson	SVMs, Cross-Validation, Hyperparameter Tuning
Attila Bagoly (GSOC 2016)	Jupyter Integration, Visualization, Output
Paul Seyfert	Performance optimization
Andrew Carnes	Regression, Loss Functions, BDT Parallelization

Continued invaluable contributions from Andreas Hoecker, Helge Voss, Eckhard v.Thorne, Jörg Stelzer, and key support from CERN EP-SFT group and other ROOT team members

Backup Slides

Deep Learning Performance



Network:

- 20 input nodes,
- 5 hidden layers with 256 nodes each,
- *tanh* activation functions,
- squared error loss
- batch size = 1024
- Single precision

Training Data:

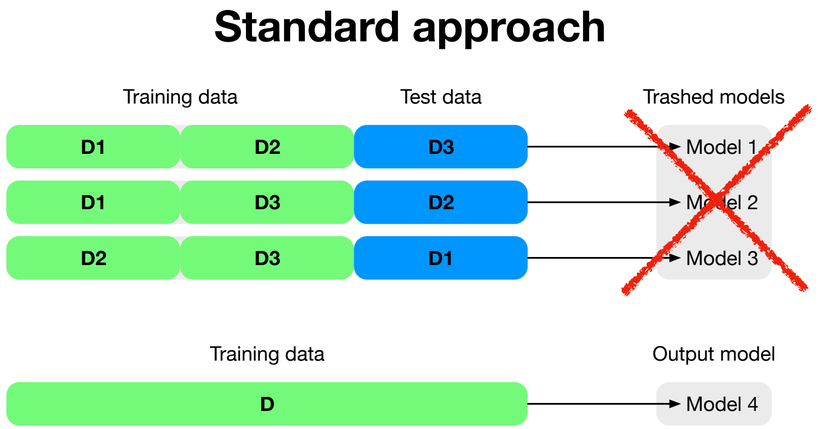
- Random data from a linear mapping $\mathbb{R}^n \rightarrow \mathbb{R}$

Excellent throughput compared to Theano on same GPU

Cross Validation in Application

"CV in application"

- Workflow in HEP analysis
- Deterministic assignment of events to folds + save all trained models
- Performance estimation holds for collection of models



Cross validation in application

Used in e.g.

- Evidence for the H → bb^{-} decay with the ATLAS detector (2017) — arxiv:1708.03299
- Search for the bb decay of the Standard Model Higgs boson in associated (W/Z)H production with the ATLAS detector (2015) — arxiv:1409.6212

Example PyMVA with Keras

Define model for Keras

Define the Keras model in Python

In [5]:
```
# Define model
model = Sequential()
model.add(Dense(32, init='glorot_normal', activation='relu',
        input_dim=numVariables))
model.add(Dropout(0.5))
model.add(Dense(32, init='glorot_normal', activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2, init='glorot_uniform', activation='softmax'))

# Set loss and optimizer
model.compile(loss='categorical_crossentropy', optimizer=Adam(),
        metrics=['categorical_accuracy',])

# Store model to file
model.save('model.h5')

# Print summary of model
model.summary()
```

Book the method as any others of TMVA

Book methods

Just run the cells that contain the classifiers you want to try.

In [6]:
```
# Keras interface with previously defined model
factory.BookMethod(dataloader, ROOT.TMVA.Types.kPyKeras, 'PyKeras',
        'H:!V:VarTransform=G:FilenameModel=model.h5:'+\
        'NumEpochs=10:BatchSize=32:'+\
        'TriesEarlyStopping=3')
```

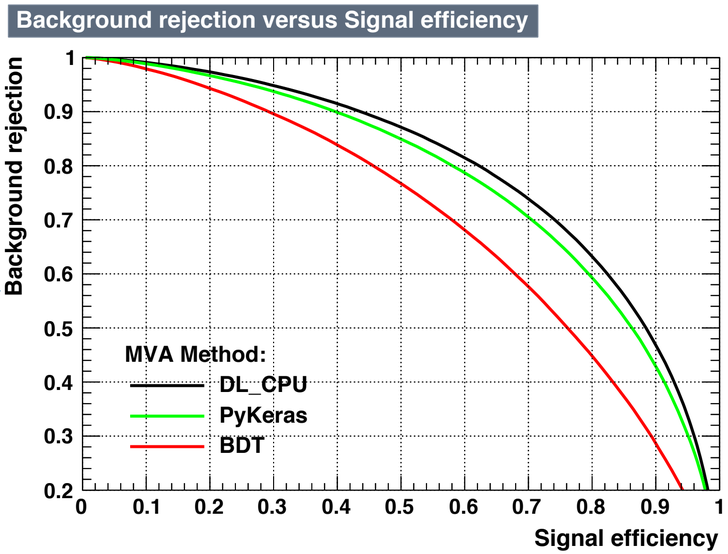
Out[6]:
```
<ROOT.TMVA::MethodPyKeras object ("PyKeras") at 0x77e48b0>
```

PyMVA with Keras

Train,Test and Evaluate inside TMVA (using TMVA::Factory)

Run training, testing and evaluation

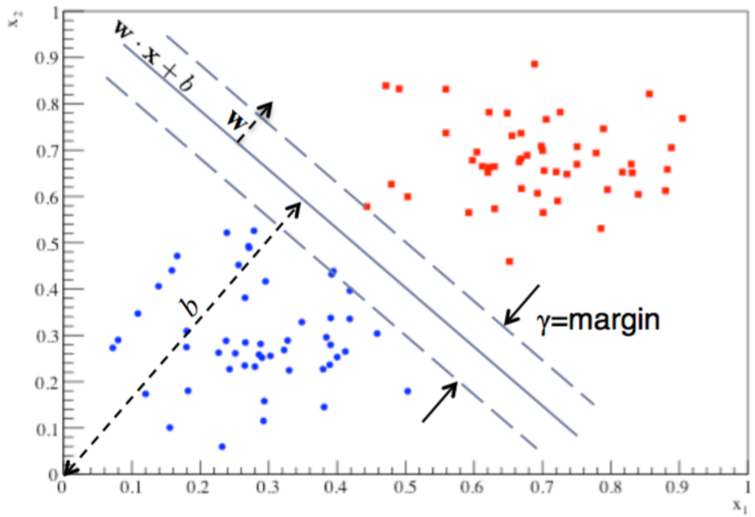
```
In [8]: factory.TrainAllMethods()

Factory                  : Train all methods

In [9]: factory.TestAllMethods()

Factory                  : Test all methods
Factory                  : Test method: PyKeras
:

In [10]: factory.EvaluateAllMethods()

Factory                  : Evaluate all methods
Factory                  : Evaluate classifier: PyKeras
:
```



Examine result with TMVA GUI

Improved SVM

- Include in TMVA additional functionality for SVM: (work by T. Stevenson and A. Bevan)
  - New Kernel functions:
    - Multi-Gaussian, Polynomial and support for product and sum of kernel functions



  - Implemented Parameter optimisation for kernel parameters and cost
    - Cost weighted to signal/background events