Machine Learning in ROOT

S. Gleyzer (University of Florida), Lorenzo Moneta (CERN)

OpenLab Machine Learning Workshop, 27th April 2017

Outline

- Present status and Overview of the ML tools in ROOT
- New Features
  - Deep Learning
- Future plans and outlook
- Summary

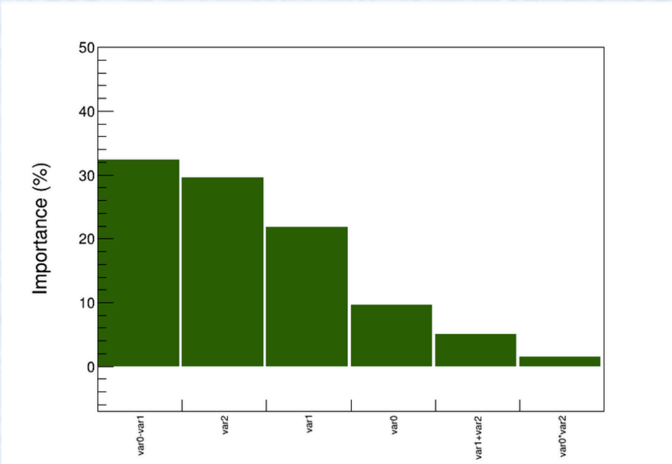
TMVA

- 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 analysis (e.g. Higgs studies)
- Easy interface for beginners, powerful for experts
- Several active contributors
- Various new features added last year (ROOT version 6.08)

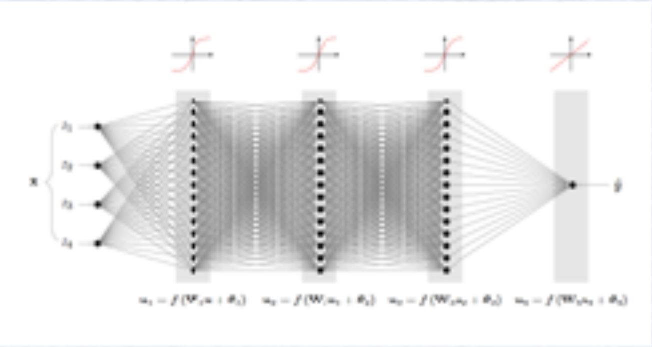
New Features

New features added last year:

- Improve software modularity
  - decoupling of algorithms/data set/input variables
- External Interfaces to ML tools in R
  - using ROOT-R interface
- Interfaces to Python tools
  - scikit-learn and then Keras (supporting both Theano and Tensorflow)
- Variable Importance algorithm
- Several improvements in SVM



New Features (2)

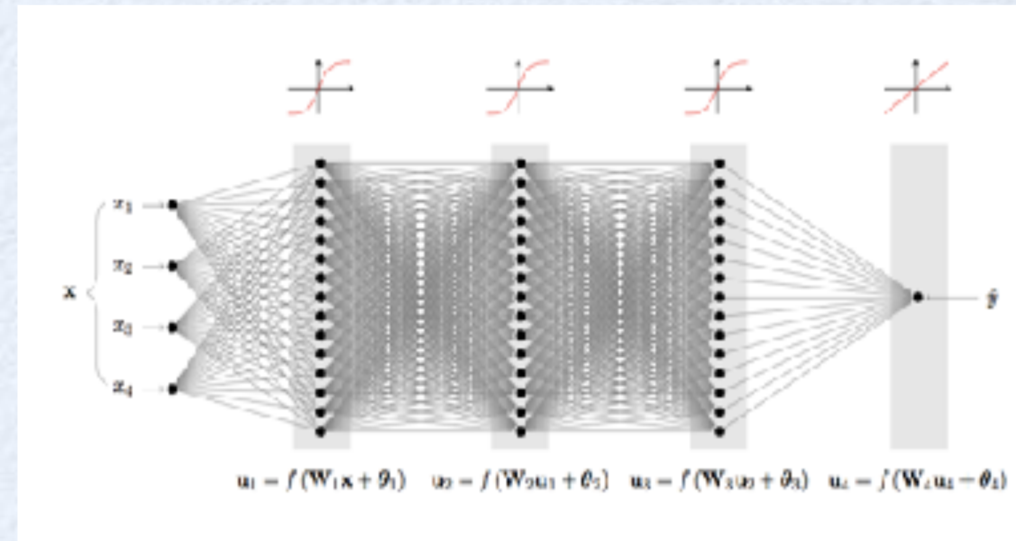


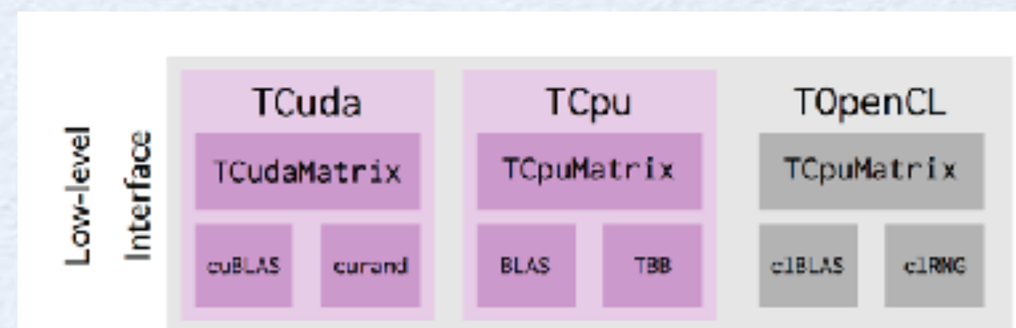
New features added last summer in 6.08:

- Deep Learning
  - support for parallel training on CPU and GPU (with CUDA and OpenCL)
- Cross Validation and Hyper-parameter optimisation
- Improved loss functions for regression
- Interactive training and visualization for Jupyter notebooks
- new pre-processing features (variance threshold)

Deep Learning

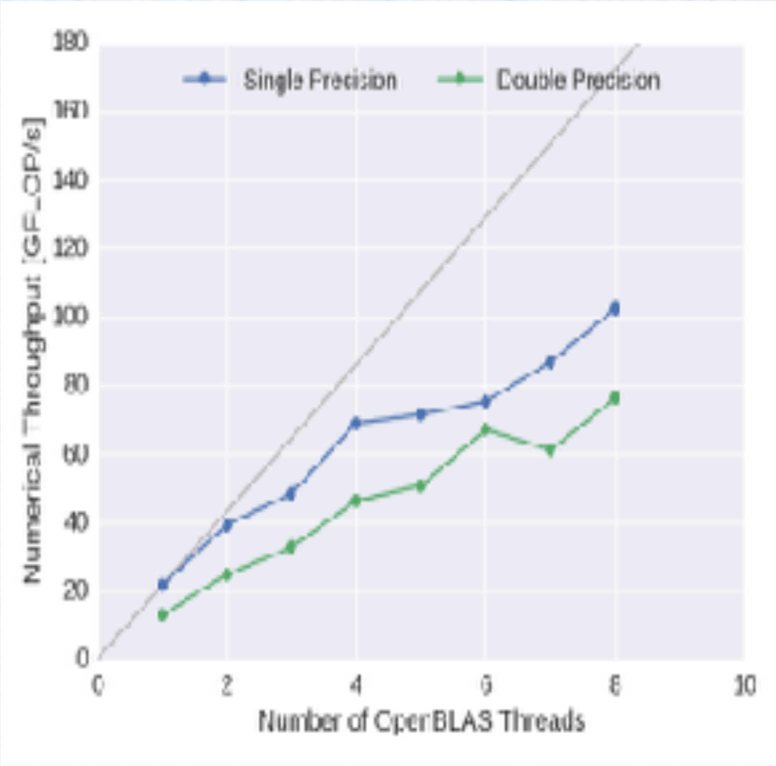
- Powerful ML method based on Deep Neural Network (DNN)
- New Deep Learning library in ROOT
  - parallel evaluation on CPU
    - implementation using OpenBLAS and TBB
  - GPU support
    - CUDA
    - OpenCL
  - Excellent performance and high numerical throughput
- For more information see
  - https://indico.cern.ch/event/565647/contributions/2308666/attachments/1345668/2028738/tmva_dnn_gpu.pdf



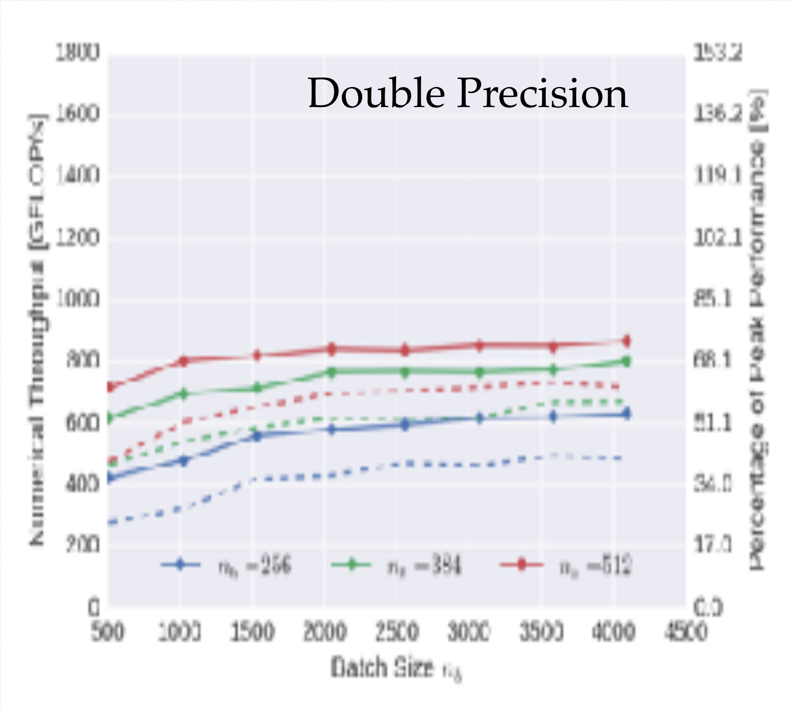


Deep Learning Performance

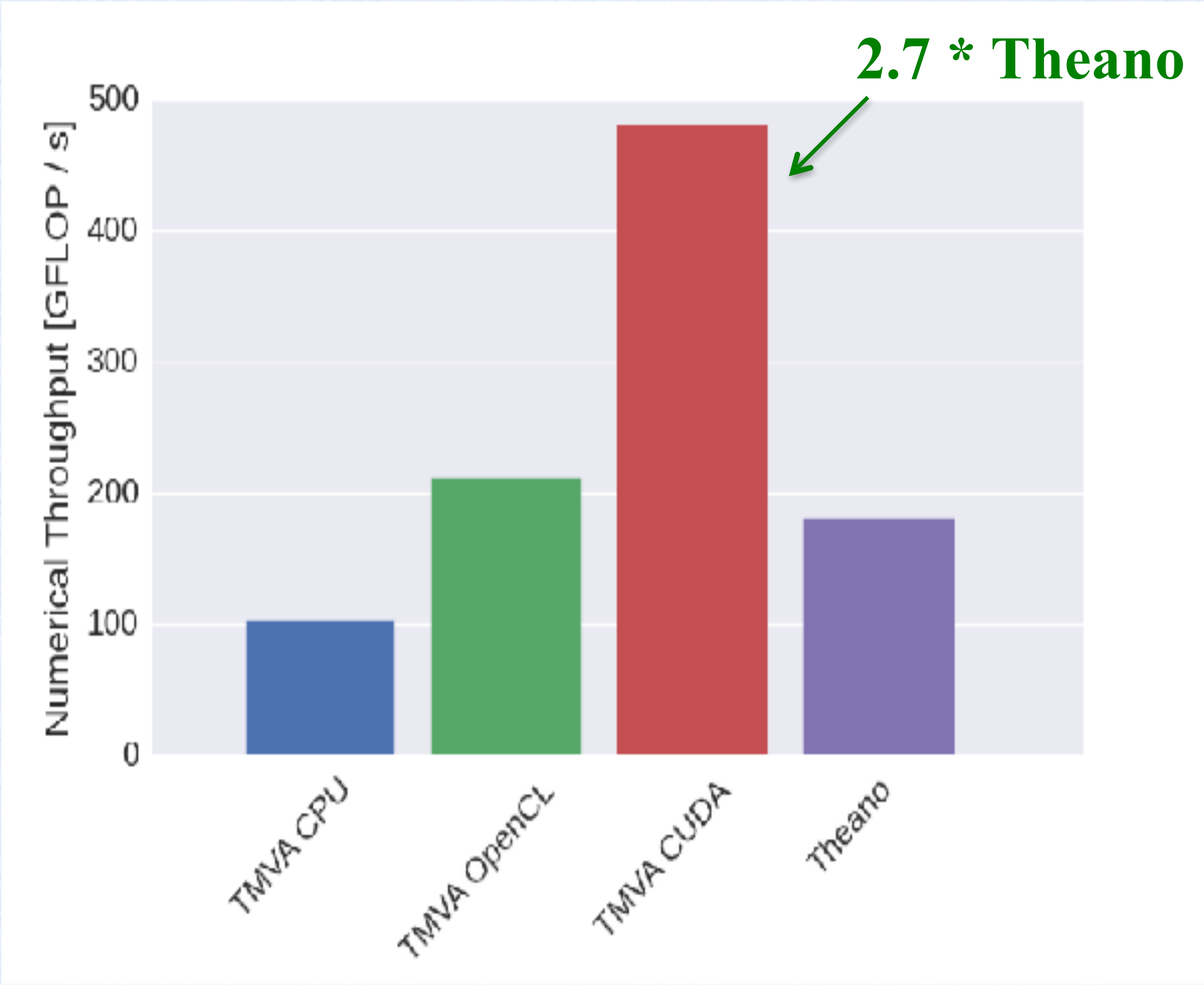
- CPU Performance
  - Intel Xeon E5-2650, 8 × 4 cores
  - Estimate peak performance:
    - 16 GFLOP/s / core



- GPU Performance
  - NVIDIA Tesla K20
  - Peak performance:
    - 1.17 TFLOP/s with double precision



Deep Learning Performance



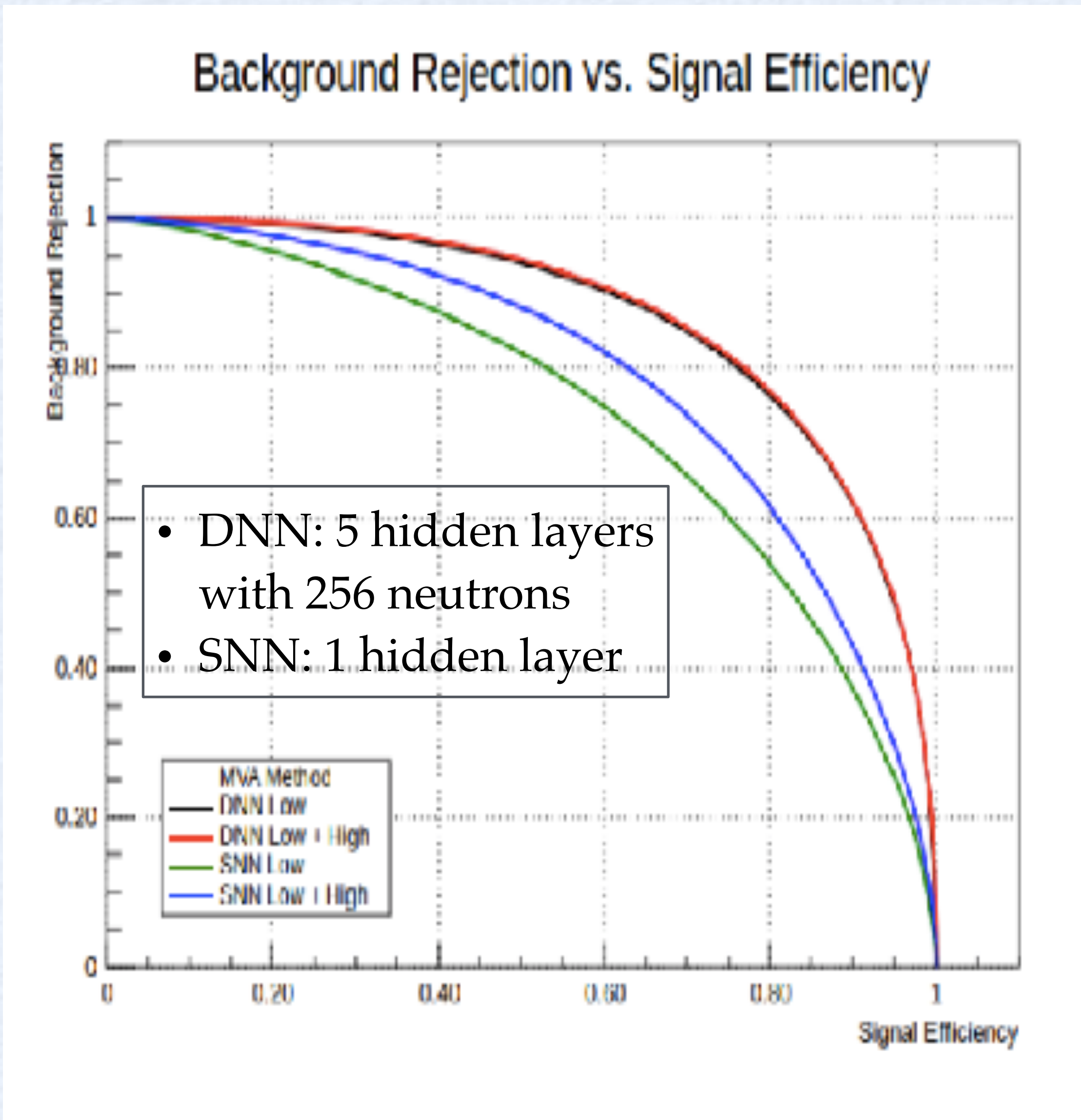
batch size = 1024

Single precision

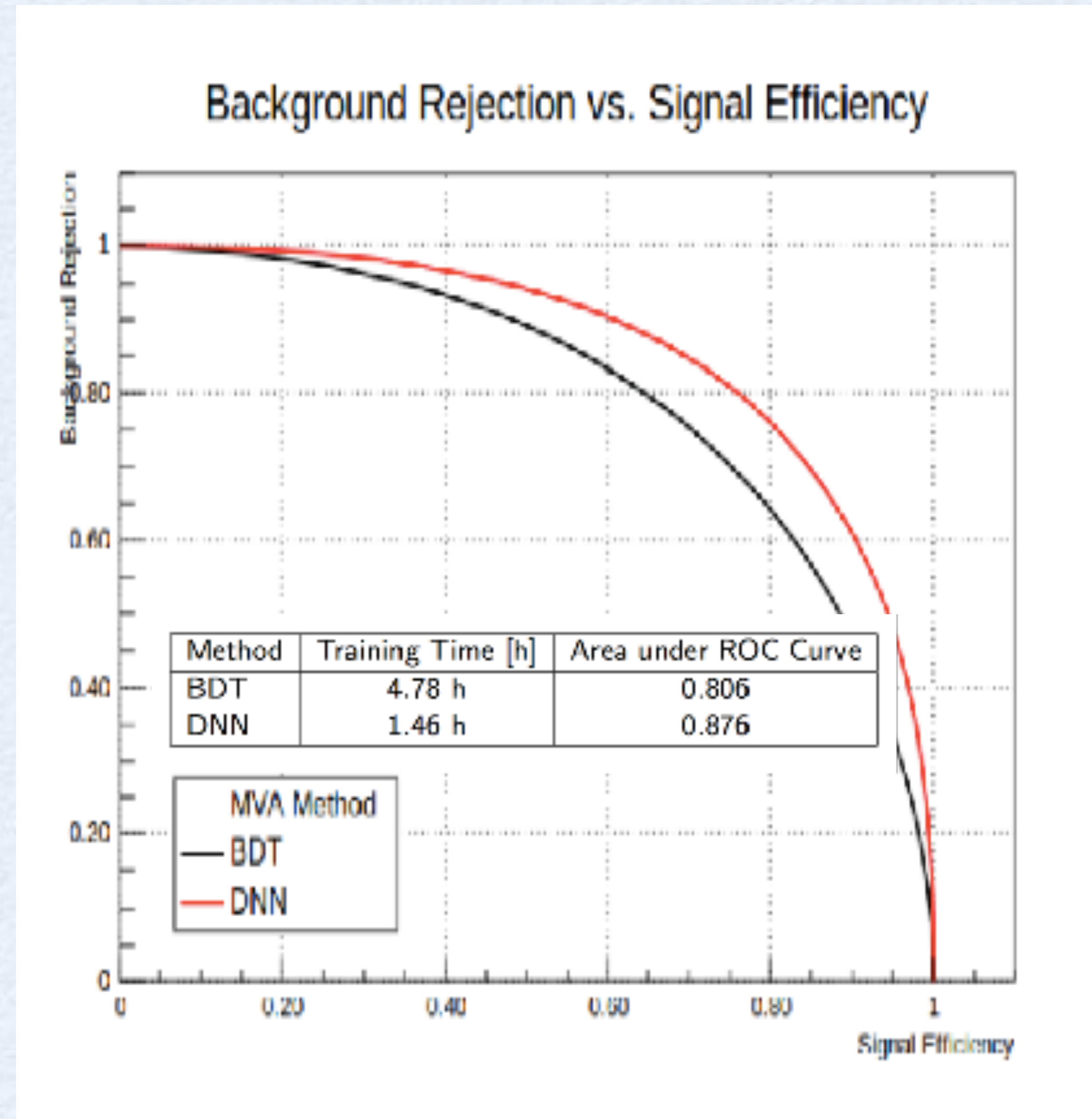
Excellent throughput compared to Theano on same GPU

Deep Learning Performance

DNN vs Standard ANN



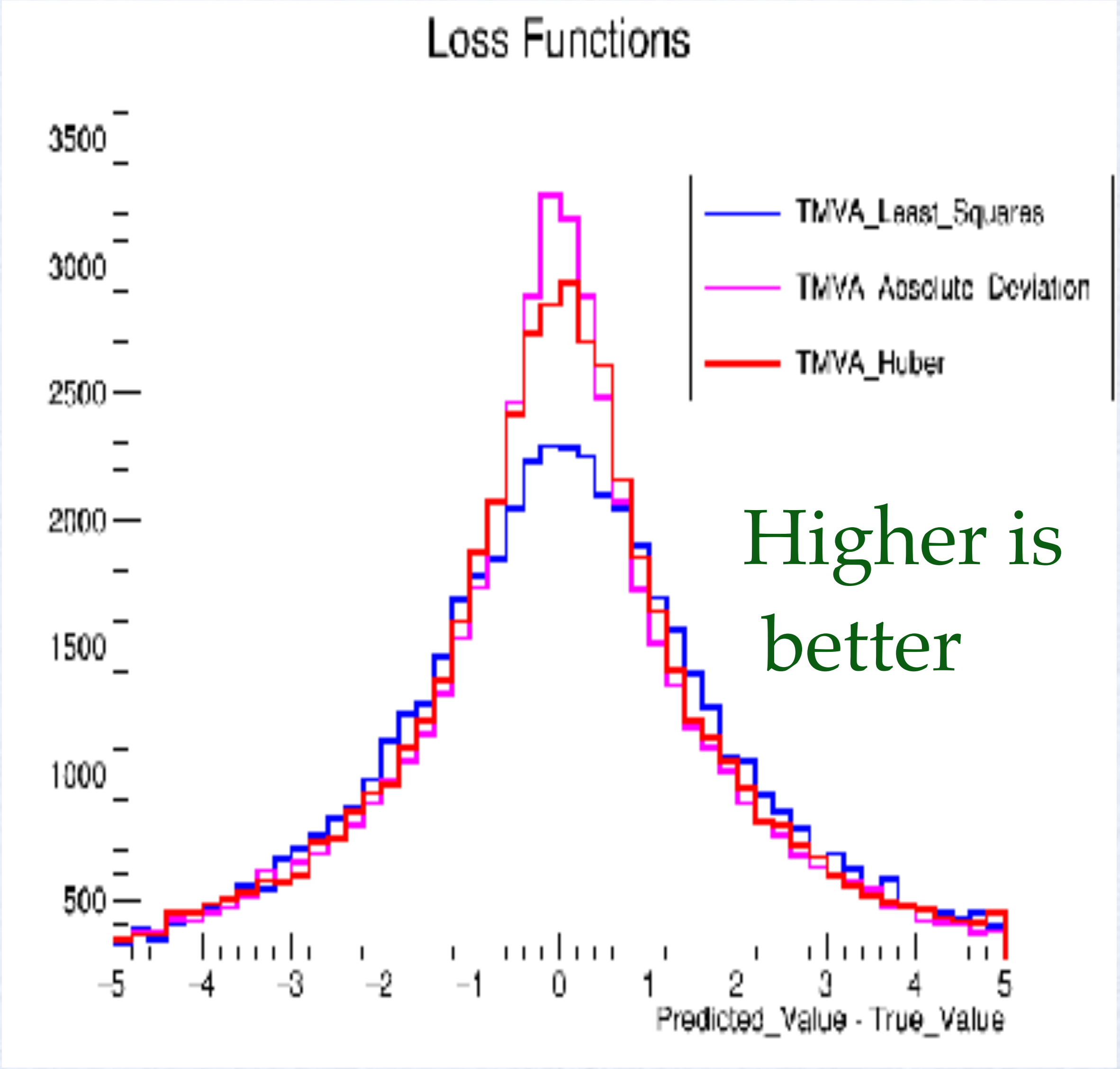
DNN vs BDT



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

Regression

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

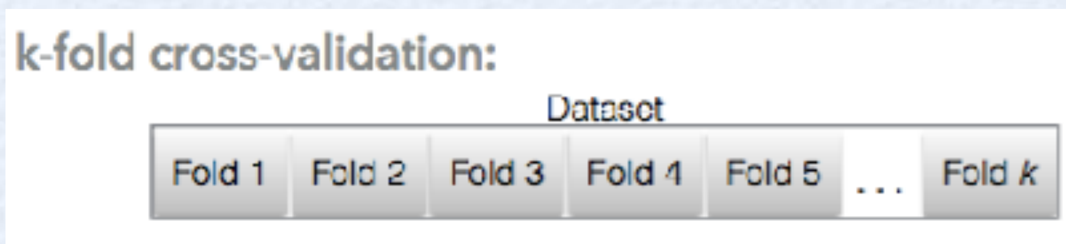


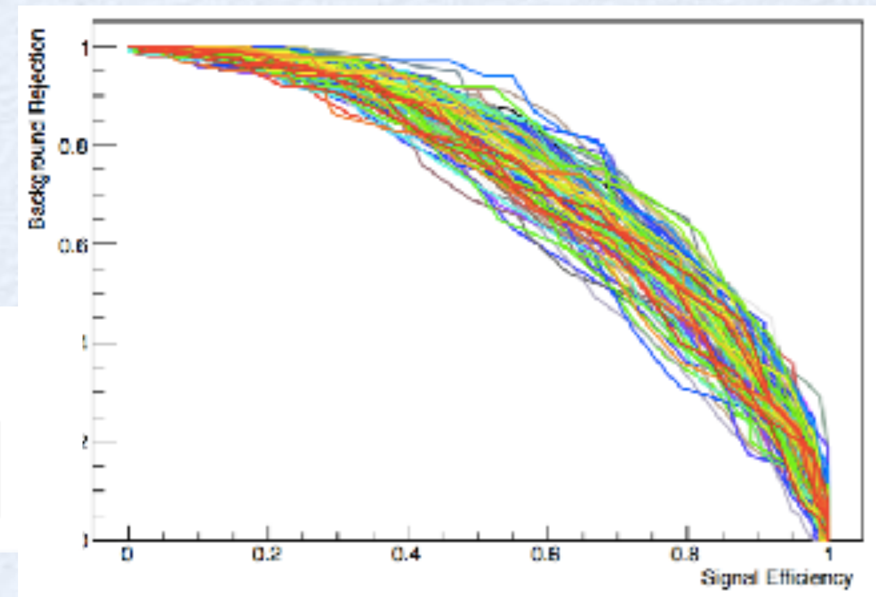
Higher is better

Important for regression performance

Cross Validation

- Added k-fold cross-validation



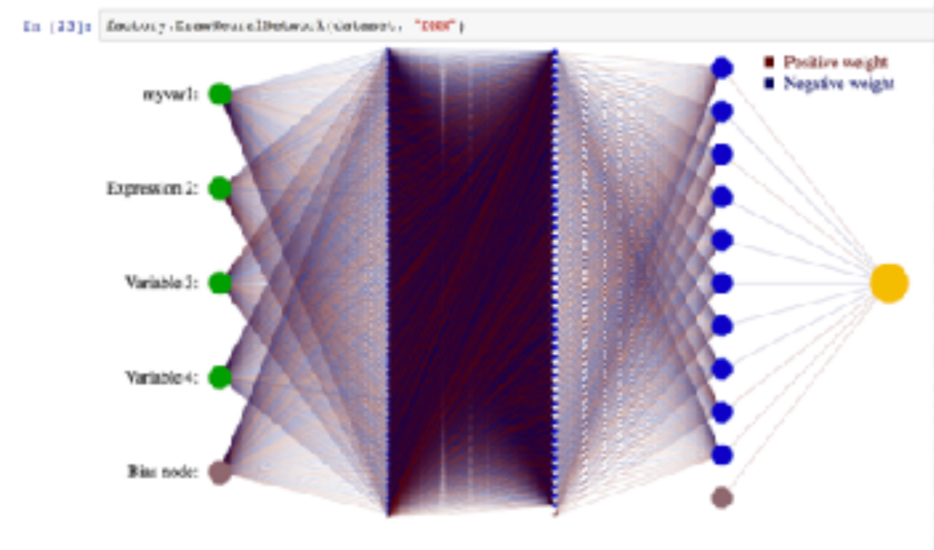


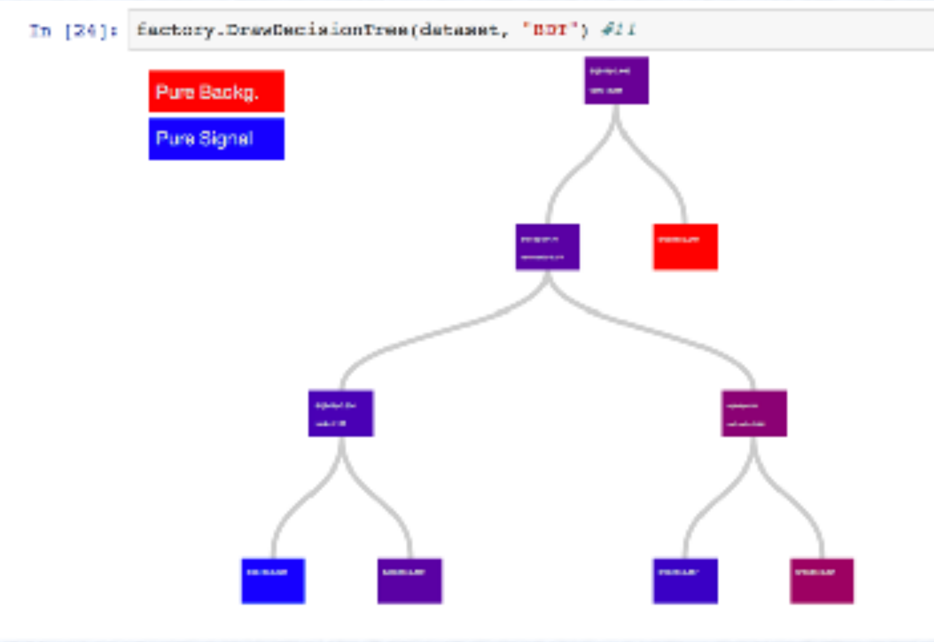
- Hyper-parameter tuning
  - find optimised parameters (BDT-SVM)
- Providing support for parallel execution
  - multi-process/multi-threads and on a cluster using Spark or MPI

Jupyter Integration

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

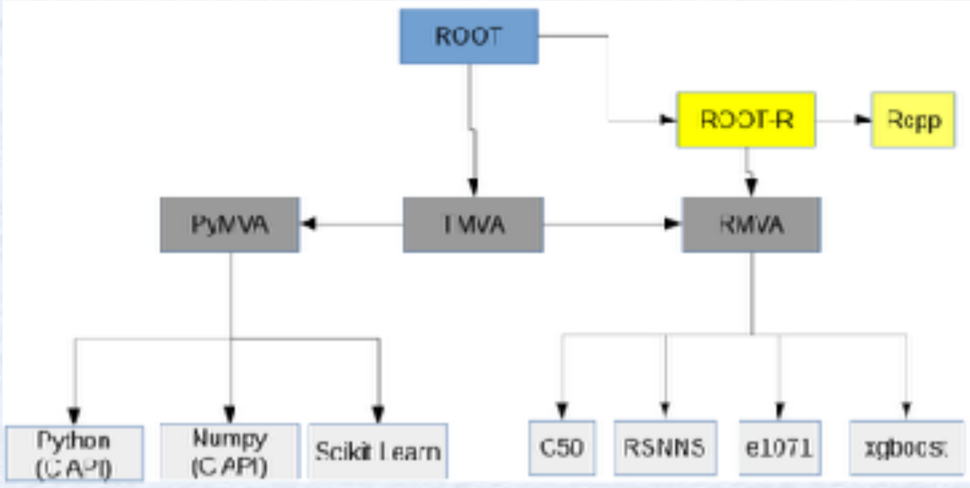
- Improved Python API for TMVA functions
- Visualisation of BDT and DNN
- Enhanced output and plots (e.g. ROC plots)
- Improved interactivity (e.g. pause/resume/stop of training)
- see example in SWAN gallery
  https://swan.web.cern.ch/content/machine-learning

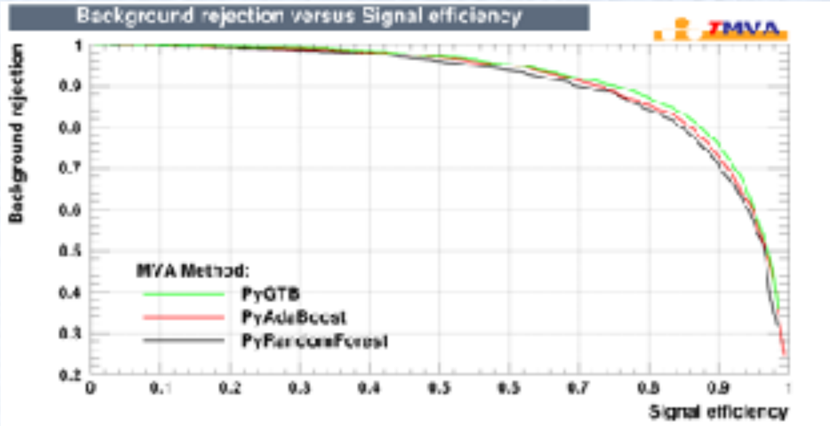




TMVA Interfaces

- RMVA: Interface to Machine Learning methods in R
  - c50, xgboost, RSNNS, e1071
  - see http://oproject.org/RMVA
- PYMVA: Python Interface
  - skikit-learn (RandomForest, Gradiend Tree Boost, Ada Boost)
    - see http://oproject.org/PYMVA
  - Keras (Theano + Tensorflow)
    - support model definition in Python
    - see https://indico.cern.ch/event/565647/contributions/2308668/attachments/1345527/2028480/29Sep2016_IML_keras.pdf
  - Data are copied from TMVA to Numpy array
  - C Python interface used



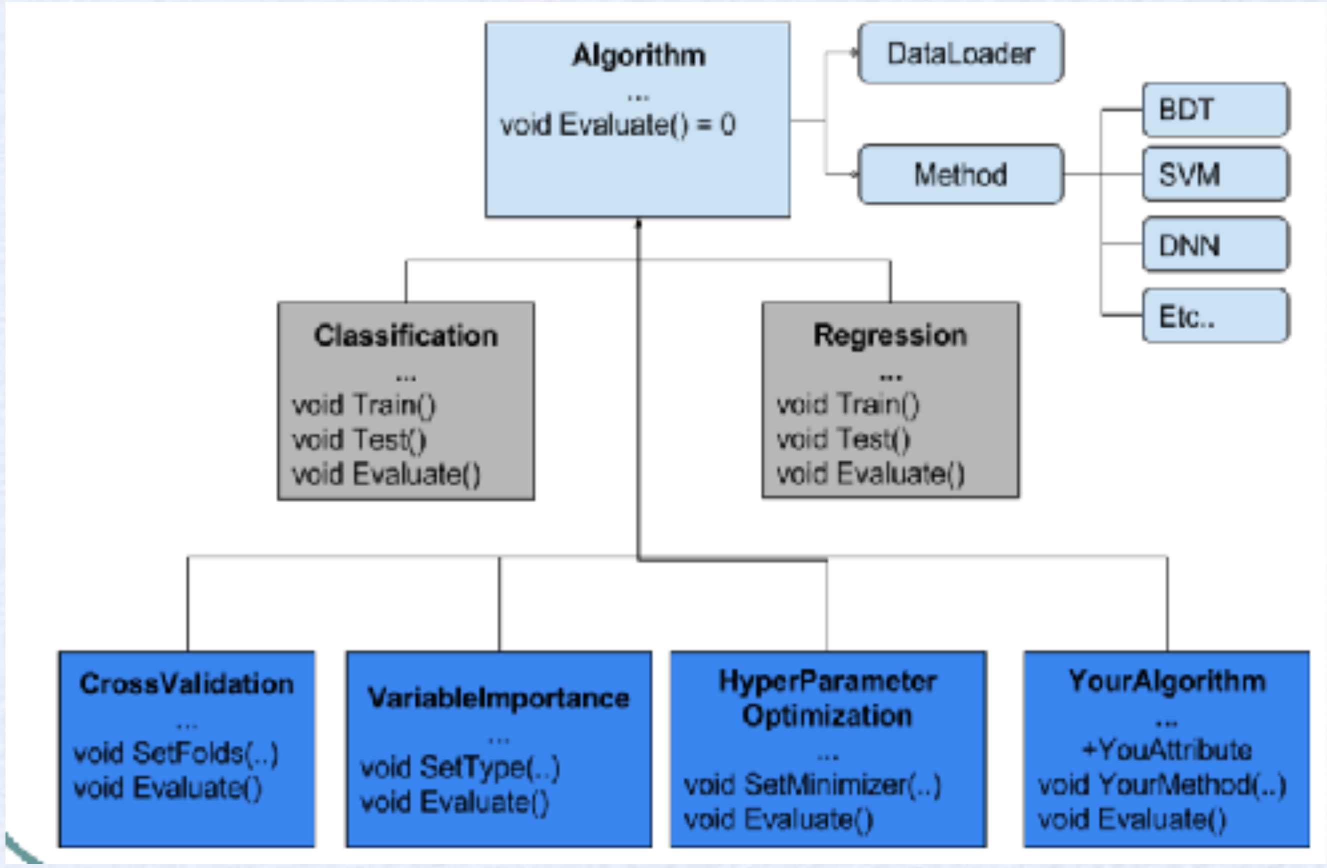


Upcoming Features

- Full support for parallelisation when using analyzer tools
  - e.g. cross-validation, hyper-parameter optimisation, variable importance
- Integrating deep auto-encoders and more unsupervised pre-processing tools (e.g. Hessian LLE)
- Improvements in deep neural networks
  - addition of Convolutional Neural Network (CNN) and Recurrent Neural Network planned for the summer (GSOC students)
- Adding support for multi-target regression
- Working on performance improvements of existing tool
- Facilitate handling of large data sets
  - minimise memory usage by minimizing data copy

Parallelisation

- Refactoring of high level classes in TMVA in order to support parallelisation
- Parallelize analyzer tools:
  - cross-validation
  - hyper-parameter optimisation
  - variable importance
- Support different paradigms:
  - multi-threads using TBB
  - multi-process using fork (TProcessExecutor)
  - multi-cluster using MPI or Spark
- Internal parallelization of the ML methods (multi-threads and GPU)
  - already supported for the DNN
  - on-going work for the BDT



Summary

- ROOT continues to provide a large set of ML tools
- Many new features added recently and even more will come soon
  - important to develop and maintain a core set of tools for HEP usage despite very good ML software exists outside
  - working also on integrating better external tools (handling large data sets)
- Many contributions from various people
  - excellent students from Google Summer of Code
- Feedback and further contributions welcome

More Information

Websites:

- http://root.cern.ch
- http://iml.cern.ch
- http://oproject.org

TMVA Contributors

- Lorenzo Moneta — Algorithm development, Parallelization, Integration and support
- Sergei Gleyzer — Analyzer Tools, Algorithm Development
- Omar Zapata Mesa — PyMVA, RMVA, Modularity, Parallelization
- Peter Speckmeyer — Deep-Learning CPU
- Simon Pfreundschuh — Deep-Learning CPU and GPU
- Adrian Bevan, Tom Stevenson — SVMs, Cross-Validation, Hyperparameter Tuning
- Attila Bagoly — Jupyter Integration, Visualization, Output
- Albulena Saliji — TMVA Output Transformation
- Stefan Wunsch — KERAS Interfance
- Pourya Vakilipourtakalou — Cross-Validation, Parallelization
- Abhinav Moudhil — Pre-processing, Deep Autoencoders
- Georgios Douzas — Spark, Cross-Validation, Hyperparameter Tuning
- Paul Seyfert — Performance optimization
- Andrew Carnes — Regression, Loss Functions, BDT Parallelization
- Kim Albertsson — Multi-class for BDT and various code improvements

- Continued invaluable contributions from Andreas Hoecker, Helge Voss, Eckhard von Thorne, Jörg Stelzer, and key support from CERN EP-SFT Group

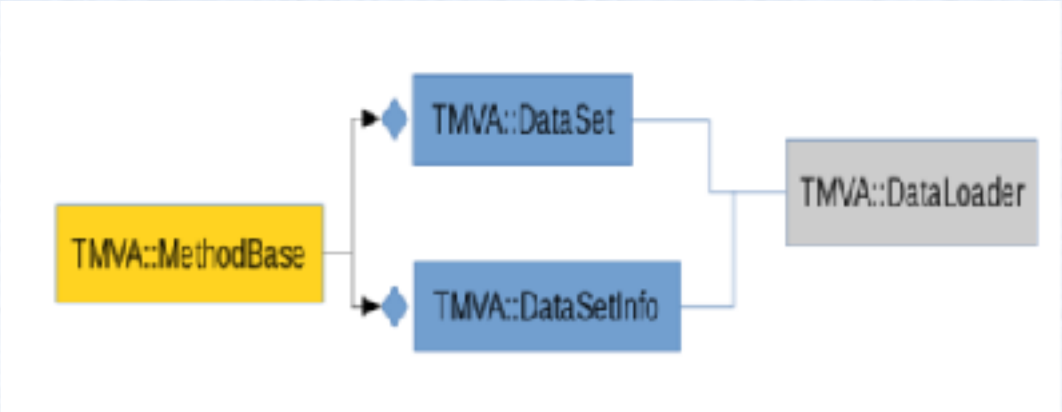
Backup slides

TMVA DataLoader

- DataLoader is a new class that allows greater flexibility when working with datasets. It is an interface to
  - load the datasets
    - root files (TTrees) but can be extended to other types
  - add variables
- TMVA Factory links DataLoader with a specific MVA method when booking

```
factory->BookMethod( DataLoader *loader, Types::EMVA theMethod,
                     const char * methodTitle, const char *option = "" );
```

- Obtained desired flexibility in de-coupling methods/dataset/variables

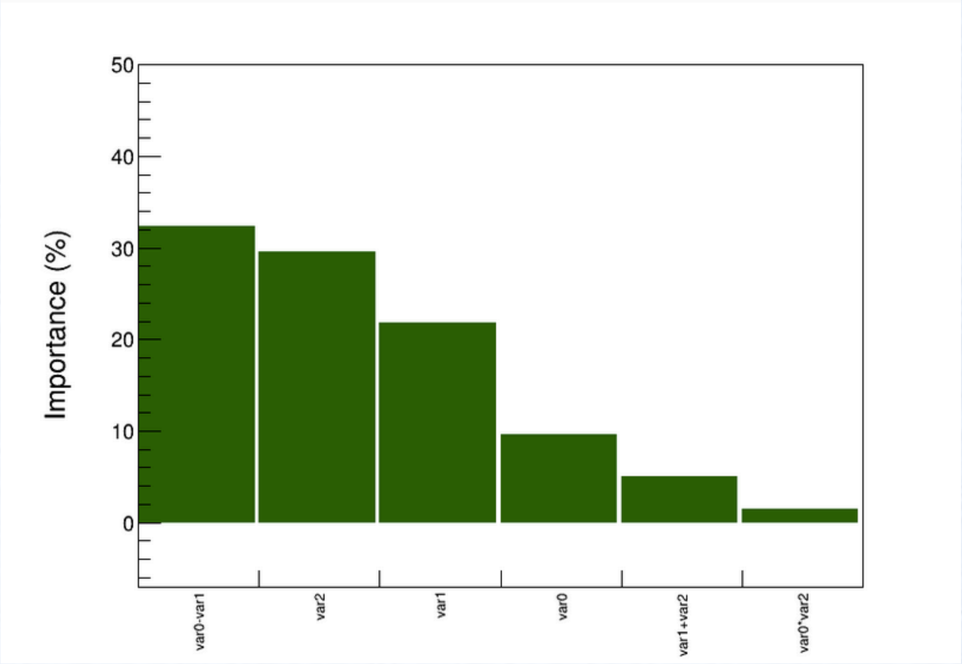


Feature Importance

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

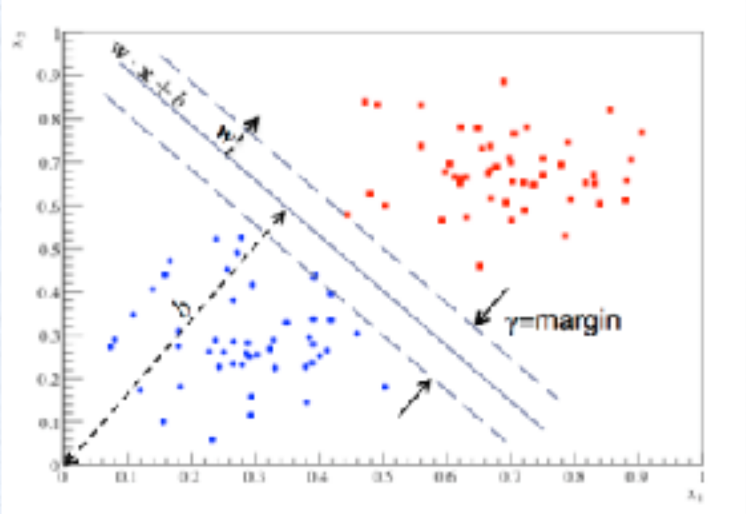
$$FI(X_i) = \sum_{S \subseteq V : X_i \in S} F(S) \times W_{X_i}(S) \qquad W_{X_i}(S) \equiv 1 - \frac{F(S - \{X_i\})}{F(S)}$$

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



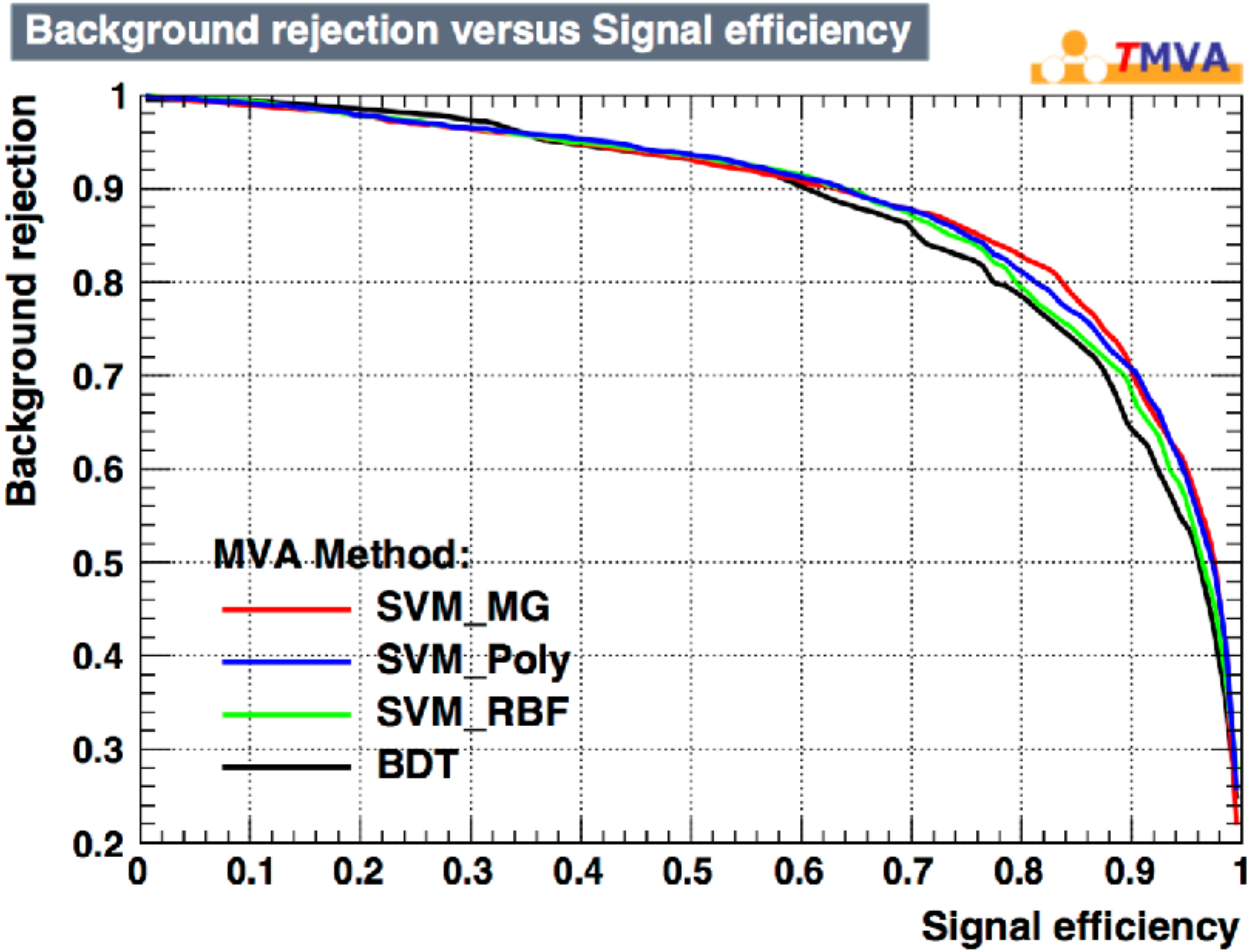
Improved SVM

- 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



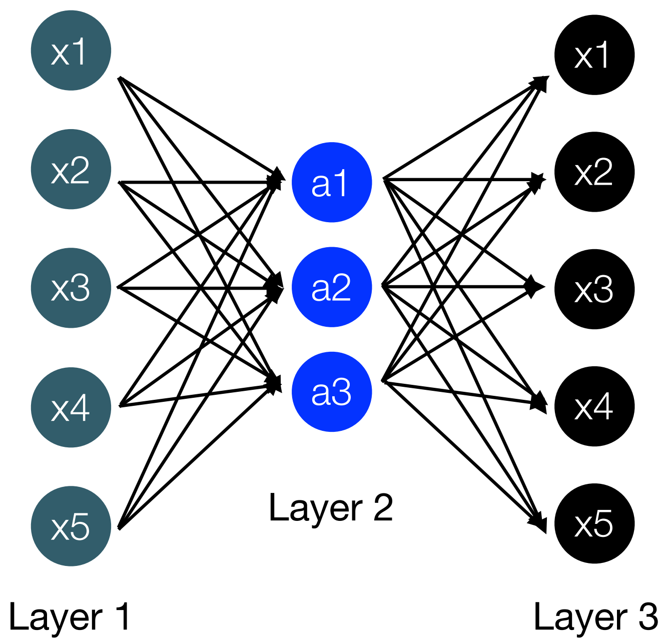
  - Implemented Parameter optimisation for kernel parameters and cost
    - Cost weighted to signal/background events
  - Loss function (implemented but not currently used)

EXAMPLES - HIGGS ML CHALLENGE DATASET



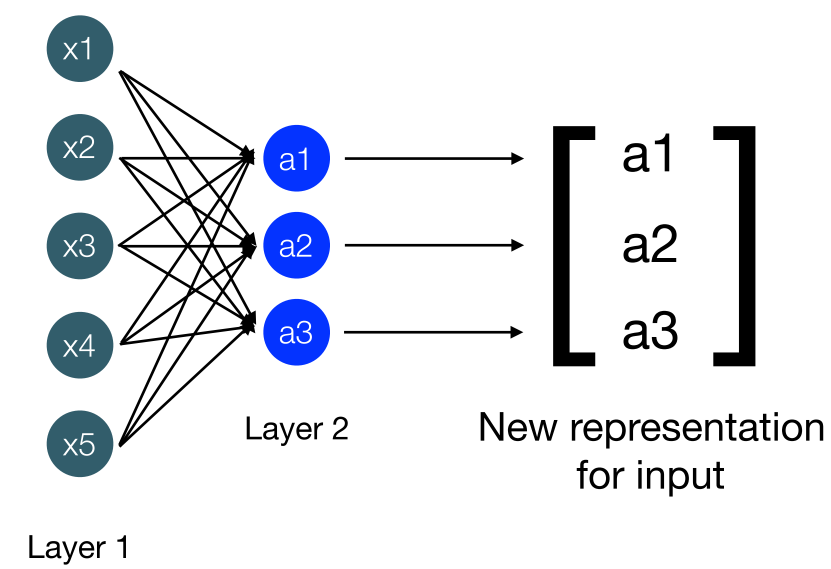
Deep Autoencoders

Deep Autoencoders



- Network is trained to output the input i.e. learn the identity functions.
- Constrain number of units in hidden layer, thus learning compressed representation.

3

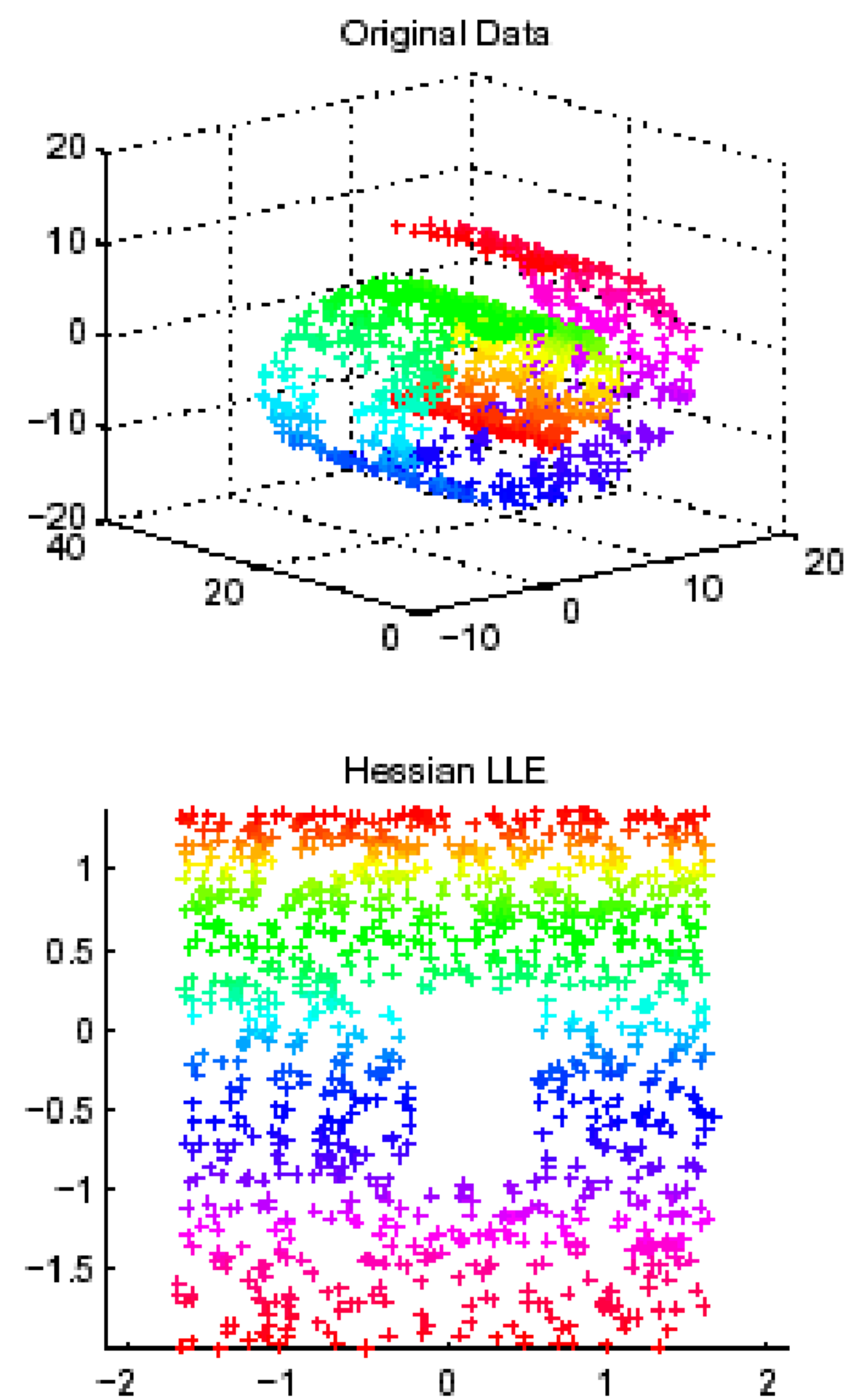


Reference:
Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." Science 313.5786 (2006): 504-507.

4

Hessian Linear Local Embedding

- A non linear dimensionality reduction method
- Embeds a set of points from high dimensional space to low dimensional space such that projected point should have the same neighbourhood as the original point



Reference:
Donoho, David L., and Carrie Grimes. "Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data." Proceedings of the National Academy of Sciences 100.10 (2003): 5591-5596.