EPJ Web of Conferences **214**, 06014 (2019) https://doi.org/10.1051/epjconf/201921406014
CHEP 2018

New Machine Learning Developments in ROOT/TMVA

Kim Albertsson[1,6,*], *Sergei* Gleyzer[4], *Marc* Huwiler[5], *Vladimir* Ilievski[5], *Lorenzo* Moneta[1], *Saurav* Shekar[8], *Akshay* Vashistha, *Stefan* Wunsch[1,7], and *Omar Andres* Zapata Mesa[2,3]

[1]CERN
[2]University of Antioquia
[3]Metropolitan Institute of Technology
[4]University of Florida
[5]EPFL
[6]Lulea University of Technology
[7]Karlsruhe Institute of Technology
[8]ETH Zurich

Abstract.
The Toolkit for Multivariate Analysis, TMVA, the machine learning package integrated into the ROOT data analysis framework, has recently seen improvements to its deep learning module, parallelisation of multivariate methods and cross validation. Performance benchmarks on datasets from high-energy physics are presented with a particular focus on the new deep learning module which contains robust fully-connected, convolutional and recurrent deep neural networks implemented on CPU and GPU architectures. Both dense and convolutional layers are shown to be competitive on small-scale networks suitable for high-level physics analyses in both training and in single-event evaluation. Parallelisation efforts show an asymptotical 3-fold reduction in boosted decision tree training time while the cross validation implementation shows significant speed up with parallel fold evaluation.

1 Introduction

Machine learning in high-energy physics research has a long standing tradition and is recently seeing increased adoption in a variety of applications. Boosted decision trees remains a popular baseline and different deep learning approaches are being investigated in areas including tracking, particle identification, jet reconstruction, and physics analysis.

This work describes recent developments included in the toolkit for multivariate analysis (TMVA) [1] and evaluates their performance on selected examples relevant to high-energy physics. Additionally, the future development direction is discussed based on identified trends in particle physics research.

In particular, deep learning has seen successful application, surpassing previous methods in classification performance in both ATLAS and CMS studies. Small-scale neural networks on the order of 5 dense layers has been proven beneficial in top-quark/W-boson tagging and Higgs searches respectively [2][3][4]. Both convolutional and recurrent networks have been applied to jet tagging. CMS's DeepJet [5] architecture uses a 1 by 1 convolution followed

*e-mail: kim.albertsson{at}cern.ch

Table 1: Features of the TMVA deep learning module and the ROOT version in which the features was implemented. A green color indicates that the feature is currently available while a yellow color indicates it is in the process of being integrated. Future plans include the integration GAN's, VAE's and support for LSTM layers.

	Dense	Conv	RNN	LSTM	GAN	VAE
CPU	6.08	6.12	6.12			
GPU	6.08	6.14				
	updated	new		upcoming		

by recurrent layers while an ATLAS Quark/Gluon tagger [6] uses three convolutional layers followed by a 128 unit dense layer. The convolutional layers used 128 filter banks, and 5 by 5 kernels.

The upcoming upgrade of the LHC, the High Luminosity LHC[7], will bring an unprecedented data rate with an expected 200 collisions per event, compared to the current 20 - 60. generating an order of magnitude increase in the number of generated particles and detector hits. Due to the algorithmic scaling complexity in e.g. tracking the increased number of detector hits will yield an exponential increase in required computational resources. New approaches that can scale to the required event complexity, while maintaining selection performance, need to be developed.

TMVA is a ROOT-integrated framework for machine learning. ROOT is a software suite designed to facilitate data analysis for high-energy physics experiments with TMVA providing an environment to evaluate and compare supervised learning methods. The toolkit is expanding to incorporate machine learning trends both through internal and external interfaces e.g. the integrated module for deep learning and better inter-operation with industry and open source community tools.

2 New Developments

The deep learning library of TMVA has been redesigned to incorporate more general network layer types; Convolutional and recurrent layers are now supported on CPU. Convolutional layers are also supported using a GPU backend. Table 1 summarizes the implementation status of the different features. Dense layers have been in TMVA since ROOT version 6.08 and are recently updated. Convolutional and recurrent layers with CPU support were introduced in ROOT 6.12 while from version 6.14 and onwards a GPU implementation is provided for convolutional layers.

Cross validation has received improved integration with the rest of TMVA, in particular with the GUI facilities for plot generation. Additionally, a cross validation technique called *cross validation in application* has been implemented.

Finally, as part of the ongoing efforts to improve the framework's performance, training of boosted decision trees and the evaluation of folds in cross validation has been parallelised.

Convolutional layer implementation

Convolutional layers provide the possibility of detecting the position of localized features in an input signal, often images or audio. The effectiveness of learning is increased through weight sharing; a small kernel is applied repeatedly to different patches of the input thus

receiving more updates compared to the non-convolutional case. TMVA currently supports 2-d convolution where the kernel is convolved over the two spatial dimensions, M and N. A third dimension, C, allows coupled inputs. For input colored images this third dimension can be interpreted as the color channels of the image. In subsequent layers, the third dimension corresponds to the output of a particular kernel in the previous layer. The output for a given kernel is given by

$$o_{i,j} = \sigma_o \left(\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \sum_{c=0}^{C-1} K_{m,n,c} I_{i+m,j+n,c} + b \right)$$

with the output at feature map location $o_{i,j}$ given by the convolution of the kernel and a patch of the input passed through an activation function, σ_o. TMVA allows configuration of both limits both and stride.

Recurrent layer implementation

Recurrent layers models the output of the layer as explicitly depending on past inputs with two central operations: The *hidden state* update, $h^{(t)}$, modelling the output dependence on past inputs; and the output $o^{(t)}$ calculation. In TMVA the layer type is implemented as described in [8] where these relations are given by

$$h^{(t)} = \sigma_h \left(W h^{(t-1)} + U x^{(t)} + b_h \right) \qquad o^{(t)} = \sigma_o \left(V h^{(t)} + b_o \right).$$

Here $h^{(t)}$ represents the hidden state of the layer at time t, which is updated given weight matrices W and U corresponding to the previous hidden state, $h^{(t-1)}$, and the current input, $x^{(t)}$, respectively. The output of a particular layer is given by $o^{(t)}$ where σ_o is the output activation function and V is the output weight matrix. TMVA implements backpropagation-through-time to train the recurrent layer as described in e.g. [9].

3 Performance benchmarks

Performance of the new developments of the deep learning module and the parallelisation of the TMVA boosted decision tree and cross validation implementations is reported in this section. Where relevant, comparisons between models achieve a similar classification performance, hence focus is placed on the run-time performance.

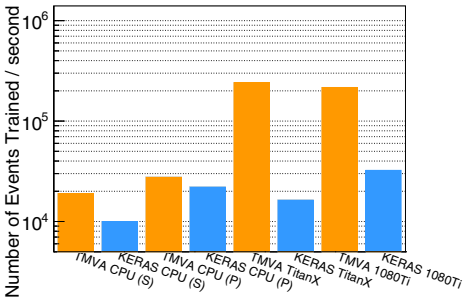
For performance evaluation of the deep learning module an Intel Xeon E5-2683 machine with 28 physical cores was used for the CPU benchmarks. GPU measurements were run using two different nVidia graphics cards, one GTX1080Ti and one TitanX card. Evaluation of BDT and cross validation performance used an Intel Xeon CPU E5-2695 v2 @ 2.40GHz machine with 24 physical cores.

The TMVA networks were compared against identical implementations using Keras [10], a high-level interface to Tensorflow and other backends. Tensorflow is a deep learning framework designed for efficient training and deployment of large scale models and datasets. It is also efficient for training models fitting on a single machine and compares favourably to other frameworks such as Caffe, Neon, and Torch for both use cases [11].

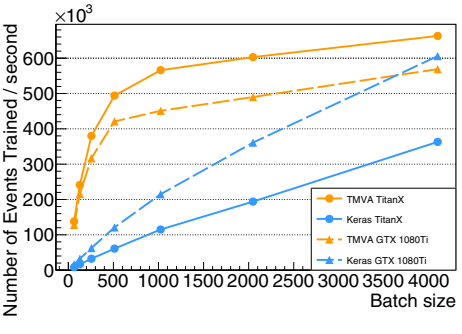
3.1 Deep learning module

In the following experiments architecture layouts are modeled after recent studies, discussed in Section 1, of the ATLAS and CMS experiments at CERN.

The updated dense layers were evaluated using the HIGGS UCI dataset [12] containing 21 low-level and 7 high-level features derived from a simulated Higgs boson decay process



(a) Event throughput for different backends for batch size 128.



(b) Scaling for GPU implementations with varying batch size, from 64 to 4096 at increasing powers of 2.

Figure 1: Number of events processed per second for training of a dense layer only network consisting of 5 dense layers, each 200 units wide, applied to 1M events with 28 features. 1a compares the performance using a single core (S), 28 cores (P) and GPU's (TitanX, 1080Ti) at a batch size of 128, while 1b shows the scaling with respect to batch size.

compared to a background top-quark decay process. The architecture setup used 5 dense layers with 200 hidden units each.

The performance of the TMVA convolutional layers was investigated using a non-public CMS data set containing images of simulated electromagnetic showers originating from photons and electrons and measured by the CMS calorimeter. The input images are cropped to 32×32 pixels, centered on active regions in the calorimeter. The evaluated architecture consisted of two convolutional layers with 12 kernels of size 3×3 followed by a max-pooling layer and two additional convolutional, max-pool layer pairs of the same specification. The final two layers were dense layers of width 64 and 32.

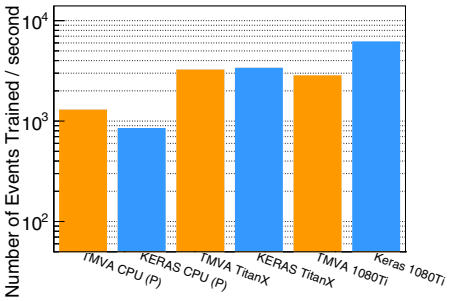
Single-event evaluation, i.e. evaluating on a single physics event at a time, is of interest in low-latency settings, where time for accumulating data into batches is insufficient. Applications include real-time processing of data streams such as in particle physics experiment triggers.

Where not otherwise specified TMVA is configured to use OpenBLAS as it matrix multiplication backend.

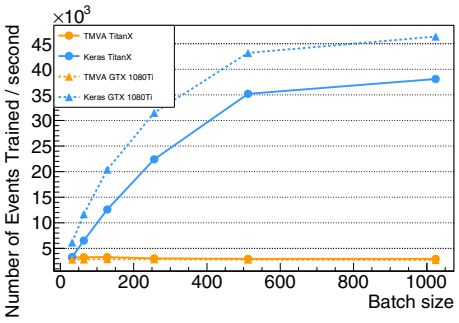
3.1.1 Training dense networks

In Figure 1a the throughput of different backends are compared with a fixed batch size. For both TMVA and Keras the single-threaded, multi-threaded and GPU performance is investigated. The multi-threaded test uses one thread per physical cores of the machine. The GPU throughput is evaluated using an nVidia TitanX and GTX1080Ti.

It is expected that there is a strict increase of performance going from serial to parallel, and from parallel to GPU execution, given that the data can be effectively transferred between the CPU and the GPU. The figure shows, for this particular batch size, TMVA performing between 1.3 (parallel) and 22 (GPU) times better that the corresponding network implemented in Keras.



(a) Event throughput for different backends for batch size 32.



(b) Scaling for GPU implementations with varying batch size, from 32 to 1024 at increasing powers of 2.

Figure 2: Training throughput of a convolutional network consisting of 4 conv layers and 2 max-pool layers followed by two dense layers of 64 and 32 units. The convolutional layers all use 12 kernels of dimension 3×3. 320k images of simulated calorimeter showers with size 32×32 was used as input. The y-axis shows number of events processed per second. 2a compares the performance for parallel (P), and GPU (TitanX, 1080Ti) execution. Single-core data was dropped since it seldom an interesting use case for convolutional nets. 2b shows the scaling with respect to the batch size.

In Figure 1b the throughput of TMVA and Keras, using the GPU backends, is compared with varying batch size. It is expected that the throughput quickly increases for small batch sizes since the studied architecture is small and neither the memory bus, nor the GPU is fully exploited. As the problem size increases, the throughput curve should flatten out as memory and GPU load saturates. Here TMVA outperforms Keras/Tensorflow on all investigated setups except for a batch size of 4096 using the GTX1080Ti card.

3.1.2 Training convolutional networks

In Figure 2a only parallel and GPU runs are included in the throughput study due to the heavy workload. As for dense networks it is expected that throughput increases as we move from CPU to GPU. For a batch size of 32, the TMVA and Keras GPU implementations both process about 3200 events per second. This is compared to the CPU backends which process 1285 and 840 images per second for TMVA and Keras respectively.

Figure 2b compares the throughput of TMVA and Keras, using the GPU backends, with varying batch size. The same scaling behaviour as for training dense layers is expected with one notable exception: TMVA currently does not parallelise over batches in convolving layers, hence the throughput is expected to be linear across all batch sizes.

Expectations are fulfilled. Keras starts flattening out at a batch size of 512 and achieves a throughput of 38000 processed images per second at batch size 1024. Meanwhile TMVA processes a consistent 3000 images per second, a factor 13 less.

Table 2: Throughput of the 5-layer dense network with batch size of 1, comparing TMVA backed by two different BLAS implementations and LWTNN. The evaluation is run on the CPU for both libraries with TMVA outperforming LWTNN with a factor of 1.5 using a tuned BLAS implementation.

	TMVA OpenBLAS	LWTNN	TMVA macos-BLAS
Events/sec	5195	12829	**18500**

3.1.3 Dense networks in application

The achieved throughput when comparing TMVA with LWTNN [13], a light-weight software package for evaluating neural networks especially geared towards the low latency setting, is tabulated in Table 2 with LWTNN achieving 12829 processed events per second.

TMVA is shown to process between 5195 and 18500 events per second depending on what matrix multiplication library is being used. The former result uses OpenBLAS while the latter uses the macos tuned Accelerate framework.

3.2 Parallelisation

ROOT version 6.14 introduced parallel training of the boosted decisions tree and parallel execution of independent folds in cross validation. To evaluate the benefits of these improvements, the software was evaluated using 1M events from the HIGGS dataset.

3.2.1 Boosted decision trees

The relative boosted decision tree training time with respect to the number of worker threads can be seen in Figure 3a where the time taken for the single threaded implementation serves as a baseline. The TMVA implementation reaches a top speedup of about 3.6 with 24 threads used; The speedup at 4 threads is close to 2. This can be compared to xgBoost[14], another popular implementation of boosted decision trees, achieving close to linear speedup across the measurement points.

3.2.2 Cross validation

The 1M events from the HIGGS dataset were split into 16 folds, with each fold trained with a single-threaded boosted decision tree using 100 boost iteration and trees fully expanded to a depth of 4 without subsampling.

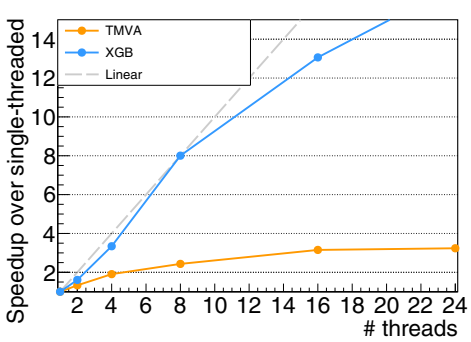
It is expected that the evaluation time decreases linearly with respect to the number of processes, with a start-up cost per process spawned.

Figure 3b shows the execution time dependence on the number of parallel processes used. Time taken for sequential execution is 3900 seconds and for fully parallel execution 390 seconds; a speedup of 10 using 16 processes. It is noteworthy that for up to 4 parallel processes, the scaling was found to be close to perfect.

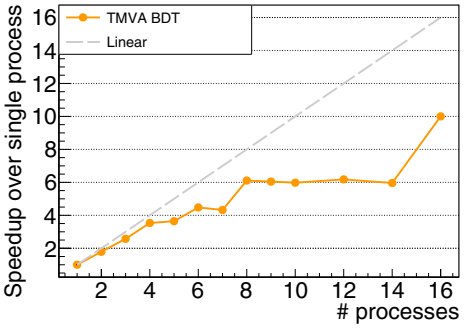
4 Discussion

4.1 Deep learning module

Comparing dense layer throughput, TMVA is performing as expected with throughput saturating at approximately 30000 events per seconds with a fully loaded GPU. Keras on the other



(a) Parallel training of BDT's.



(b) Parallel fold evaluation for cross validation with 16 folds.

Figure 3: Performance gain when using parallel execution for training of boosted decision trees (3a) and evaluation of all folds in cross validation (3b). The speedup over a baseline can be seen, for both plots, on the y-axis with the baseline being single-threaded performance in 3a and single-process performance in 3b. The x-axis shows the number of parallel workers used in the respective setup.

hand is performing below expectation and exhibiting linear scaling, indicating that there is a large constant overhead in the memory transfer rate. This is further suggested by the behaviour of the convolutional layers since there the increase in throughput is more rapid for increasing batch size than for dense layers. Further optimisation of the Keras setup is possible, should one use low level tensorflow constructs. However, since we focus on comparing high level interfaces, this is outside the scope.

In figure Figure 2b, it is clear that implementing parallelisation over the batches could benefit TMVA. What is promising is that the sequential evaluation of TMVA shows competitive throughput to the parallel evaluation of Keras up to a batch size of 32.

Single-event evaluation is, for TMVA, largely dependent on the backing matrix multiplication library. A well tuned implementation can significantly increase the throughput, enough to compete with software currently in active use in the ATLAS experiment.

4.2 Parallelisation

The parallelisation effort of the TMVA boosted decision trees shows an asymptotic speedup of 3.6 times is possible. However, as is shown in Figure 3a, better performance is achievable as evidenced by the scaling of xgboost. Workflows already employing training of boosted decision trees will benefit from the implicit multithreading.

The parallelised cross validation shown in Figure 3b shows linear speed up over sequential evaluation with overhead introduced due to process forking. The plateaus can explained by the workloads having similar duration. The final two folds cannot be started until any of the first 14 are done. A varying workload duration would smoothen the curve appropriately.

5 Future direction

A number of long-term trends in machine learning and in particular its adoption in high energy physics was identified in Section 1.

Firstly, the number of open source machine learning frameworks available has proliferated in recent years and while there exists many flourishing ecosystems built around different languages, the python ecosystem for scientific computing shows promising growth. By bridging the gap between ROOT/TMVA and external tools, physicists can easily integrate them in their workflows, leveraging e.g. multi-GPU training without explicit support in TMVA. Additionally, such an interface will make TMVA more easily accessible to a wider audience, potentially bringing new users to the High-Energy Physics ecosystem.

Secondly, deep learning is gaining increased adoption in a multitude of physics applications. In this environment TMVA aims to provide a framework for deep learning targeted specifically for high-energy physics applications including an easy-to-use high-level interface, performant training and evaluation on network sizes commonly used in physics research and fast single-event evaluation. This as a complement to the point above.

Thirdly, the future evolution of the LHC requires and order-of-magnitude increase in compute capacity. A final goal for TMVA is then to continue to increase performance of commonly used methods and incorporating new parallelism constructs from ROOT as they are made available.

In the near future, work is ongoing to expand the capabilities and performance of the deep learning module. New layer types are to be integrated and research to incorporate generative adversarial networks is ongoing. Furthermore improvements to existing layers remain, the parallelisation of convolutional layer batch evaluation being the most important.

References

[1] A. Hoecker, P. Speckmayer, J. Stelzer, J. Therhaag, E. von Toerne, H. Voss, PoS **ACAT**, 040 (2007), physics/0703039

[2] M. Aaboud et al. (ATLAS) (2018), 1808.07858

[3] A.M. Sirunyan et al. (CMS) (2018), 1804.03682

[4] M. Aaboud et al. (ATLAS), Journal of High Energy Physics **2017**, 141 (2017)

[5] M. Stoye et al. (CMS), Journal of Physics: Conference Series **1085**, 042029 (2018)

[6] Tech. Rep. ATL-PHYS-PUB-2017-017, CERN, Geneva (2017), http://cds.cern.ch/record/2275641

[7] G. Apollinari, O. Brüning, T. Nakamoto, L. Rossi, CERN Yellow Report pp. 1–19 (2015), 1705.08830

[8] D.E. Rumelhart, G.E. Hinton, R.J. Williams (MIT Press, Cambridge, MA, USA, 1986), chap. Learning Internal Representations by Error Propagation, pp. 318–362, ISBN 0-262-68053-X, http://dl.acm.org/citation.cfm?id=104279.104293

[9] P.J. Werbos, Neural Networks **1**, 339 (1988)

[10] F. Chollet et al., *Keras*, https://github.com/fchollet/keras (2015)

[11] M. Abadi et al., *TensorFlow: Large-scale machine learning on heterogeneous systems* (2015), https://www.tensorflow.org/

[12] P. Baldi, P. Sadowski, D. Whiteson, Nature Commun. **5**, 4308 (2014), 1402.4735

[13] D.H. Guest, J.W. Smith, M. Paganini, M. Kagan, M. Lanfermann, A. Krasznahorkay (2017)

[14] T. Chen, C. Guestrin, *XGBoost: A Scalable Tree Boosting System*, in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, New York, NY, USA, 2016), KDD '16, pp. 785–794, ISBN 978-1-4503-4232-2, http://doi.acm.org/10.1145/2939672.2939785