Prototype of Machine Learning "as a Service" for CMS Physics in Signal vs Background discrimination

L. Giommi[*]
University of Bologna
E-mail: luca.giommi2@studio.unibo.it

D. Bonacorsi

University and INFN of Bologna

V. Kuznetsov

Cornell University

Big volumes of data are collected and analyzed by LHC experiments at CERN. The success of this scientific challenges is ensured by a great amount of computing power and storage capacity, operated over high-performance networks, in very complex computing models on the LHC computing grid infrastructure. Now in run-2 data taking, LHC has an ambitious and broad experimental programme for the coming decades: it includes large investments in detector hardware, and similarly it requires commensurate investment in the R&D in software and computing to acquire, manage, process and analyze the shear amounts of data to be recorded in the high-luminosity LHC (HL-LHC) era.

The new rise of artificial intelligence—related to the current big data era, to the technological progress and to a bump in resources democratization and efficient allocation at affordable costs through cloud solutions—is posing new challenges but also offering extremely promising techniques, not only for the commercial world but also for scientific enterprises such as HEP experiments. Machine learning and deep learning are rapidly evolving approaches to characterising and describing data with the potential to radically change how data is reduced and analyzed, also at LHC.

This work aims at contributing to the construction of a machine learning "as a service" solution for CMS physics needs, namely an end-to-end data-service to serve machine learning trained model to the CMS software framework. To this ambitious goal, this work contributes firstly with a proof of concept of a first prototype of such infrastructure, and secondly with a specific physics use-case: the signal versus background discrimination in the study of CMS all-hadronic top quark decays, done with scalable machine learning techniques.

Sixth Annual Conference on Large Hadron Collider Physics (LHCP2018)
4-9 June 2018
Bologna, Italy

[*]Speaker.

1. Introduction

Today machine learning (ML) is becoming ubiquitous in HEP applications, most notably in final offline analyses, but it is also increasingly used in both online and offline reconstruction and particle identification algorithms, in the classification of reconstruction-level objects, such as jets, as well as in sectors that did not exploit ML up to 2-3 years ago, like the analysis of computing metadata for resource usage optimization [1, 2]. It is difficult to predict how this will evolve in HEP. For sure, more and more powerful hardware and more performant ML algorithms will be developed in the next years, and the toolset may become effective enough to position itself to replace previously existing approaches in HEP [3].

2. Machine Learning "as a Service" for CMS

The construction and deployment of a ML project and its deployment for production use requires specific skills and it is a highly time-consuming task. There are no data science teams stably collaborating with CMS physicists and helping them to achieve their ML objectives. At the same time, the CMS physicists themselves rarely have specific data science skills to face such challenges alone. What is needed to design and run successfully a ML project is often not found in a basic CMS physicist expertise (or a HEP physicist, for what matters) whose primary competences are focused on high energy physics, data analysis (including statistics), and whose ultimate goal is work towards a physics publication. Facing the need to improve a physics data analysis (and also in view of the challenges of incoming high-luminosity LHC (HL-LHC)) and understanding that ML might be an interesting exploration is hence just a first step towards actually embracing ML in an analysis.

The work presented in this paper, done in the context of CMS computing, constitutes the first step to what is considered to be a very ambitious goal in the medium-long term: build a ML "as a service" solution for CMS physics needs [4, 5], namely build an end-to-end data-service to serve ML-trained model to the CMSSW framework. The basic idea is as simple as this: instead of asking each physicists who wants to exploit ML in their own task to just learn how to do it and do it themselves independently, each user would ultimately build a modified data analysis code where "calls" to an external service—as simple as calls to functions—would be added to return a trained ML model output that could be directly used in the analysis code (e.g. in loops over events) in a streamlined manner, thus hiding all the complexity related to the ML machinery via outsourcing this to an external service. The scope of this work does not cover such a large-scope project but it is just one step in that direction, focusing mostly on how to prepare a ML task on a specific physics use-case and how to implement it in a way that it can be dealt with by such ML "as a service" infrastructure in the medium-long term. In the context of this work, it was possible to set-up a fully working prototype to serve a specific, selected physics use-case.

3. Architecture and components of "TFaaS"

The popularity of the Tensorflow ML/DL framework [6] makes this an excellent choice to apply ML algorithms for use in the CMS workflow pipeline. For this reason Tensorflow was chosen for the architecture of the ML "as a service" developed that we called "TFaaS" [7, 8], where "TF" stands for Tensorflow. To ease the use of this ML framework, several other high-level APIs—where each one can be seen as a set of commands that allows developers to compress multiple lines of code in only one, so without writing the code from scratch, saving time and energy—have been built independently on top of Tensorflow, such as Keras [9] that is easy to use thanks to its friendliness, modularity, and extensibility. For these reasons it was used for a quick prototyping of a TF-based model for TFaaS.

The concept of "as a service" means that some tasks will be eventually performed on resources that are not on premises (namely, they are "on the cloud(s)"). The jargon terminology of "ML as a service" (MLaaS) is a sort of an umbrella definition for any idea that includes the exploitation of automated and semi-automated cloud platforms to cover most infrastructure issues in a ML project such as data pre-processing, model training and model evaluation, down to the final predictions. The TFaaS is designed to bridge the gap between what HEP ML users need and what cloud companies can provide.

The full architecture of TFaaS will be described in a paper being prepared, and is only briefly summarized in the following. The CMSSW pipeline would use REST APIs to call a data-service as a front-end, which communicates with the back-end, consisting of local and cloud resources. The local, on premises resources, will be primarily used as a storage cache. The model creation and training is actually outsourced to a "TF box" that can run on clouds, to which the needed ROOT files would be served from any storage system that allows XRootD connections, i.e. from virtually any storage system in the Worldwide LHC Computing Grid (WLCG), or from HDFS systems. Such TF box, sitting somewhere on a remote machine, will be able to train the model as requested with the training dataset that can be read remotely, and then users can profit of this model as it will be served back to CMSSW applications in form of the actual predictions that were originally requested (e.g. in an event classification task, the trained model will be able to predict the type of event based on its characteristics).

An important and primary task of TFaaS is to read ROOT files and convert them into a suitable form as input to ML/DL systems. For this task the *uproot* [10] software is chosen, that provides native access to ROOT IO without any particular knowledge of the ROOT file structure. *Uproot* is implemented as a pure Python ROOT reader that directly copies columnar ROOT data into (for example) *NumPy* arrays. Uproot has few positive aspects such as the throughput observed with *uproot* scales well with increasing number of events read and *uproot* allows not only the reading of local ROOT files, but also the reading of remote files via XRootD.

4. Build a ML model for the $t\bar{t}$ selection use-case and use of TFaaS

In order to use the potentialities of the TFaaS prototype, a particularly important physics use-case has been chosen: the signal versus background discrimination in the fully-hadronic top decay performed in CMS. In this particular physics process there is a huge background (mainly composed by general QCD background) that covers the signal making difficult the discrimination and here ML could give an improvement with respect to the current analysis. The signal is the all-jets (or fully hadronic) channel, in which both W bosons decay hadronically:

$$t\bar{t} \rightarrow W^+bW^-\bar{b} \rightarrow q'\bar{q}b \; q'\bar{q}\bar{b} \rightarrow j_1 j_2 j_3 \; j_4 j_5 j_6, \tag{4.1}$$

producing 6 jets ("j_i" above) in the final state. In order to build a ML model able to discriminate the signal from the background, it is necessary to teach how they are made: the signal is given by a Monte Carlo sample and the background is given by a depleted data sample coming from the detector. Prior to using TFaaS (whose uses models created with Keras/Tensorflow, that are artificial neural networks), different algorithms of different nature have been used in order to verify how much could be useful ML in this analysis. Ten algorithms from the *Scikit-learn* library were tried such as Linear Discriminant Analysis, k-Nearest Neighbors, Random Forest, AdaBoost, Stochastic Gradient Boosting and eXtreme Gradient Boosting from Google. After having fixed a proper metric used to discriminate different models, eXtreme Gradient Boosting resulted to be on the top two algorithms and thanks to its advantages, such as a high processing speed and the implementation of regularization to reduce overfitting, it was selected as the "best" algorithm and subsequent efforts were spent to tune it further for the case under study. After having fixed the "best" model for the selected use case, it was used to give predictions on a sample coming from the detector and these predictions were compared to those obtained from the current analysis in CMS (that uses the instruments of the multivariate analysis inside TMVA of ROOT). While the former predictions are 0 or 1 for each event, the latter predictions (called with the name *mva*, that stands for multivariate analysis) are real numbers living in range 0–1 where 0 stands for background and 1 for signal. Thus an event the closer is to 1 the more is considered as a signal. Consequently, the "best" ML model fix a point in the phase space efficiency vs purity given a data sample, while the multivariate analysis produces a curve in which it is possible to do cuts for different values of *mva* and so allow a comparison with the ML model. In Figure 1 efficiency vs purity in the case $N_{b-subjets} >= 1$ for different *mva* cuts is shown, that is the case in which one or both b-subjets produced by the two top quarks decay are identified. It is possible to see that, at fixed efficiency given by the ML model,

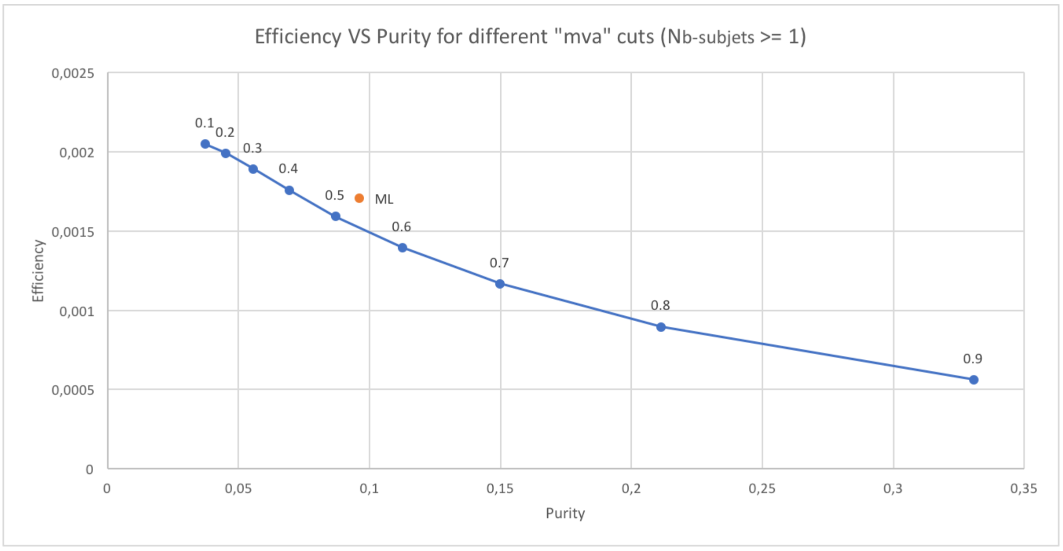

Figure 1: Efficiency and purity for different *mva* cuts (the values quoted above the dots), compared to the same obtained by ML, in the case of $N_{b-subjets} >= 1$.

the ML model has an higher purity (and thus not worse) respect to the result of the multivariate analysis, showing that an inspection of ML techniques outside the standard instruments used in the

current analysis is a good path to follow. The result obtained by the ML model is finalized to give a simple comparison and it is not the 'best" ML model in absolute, and much work can be done in order to improve this result.
After seeing that ML gives encouraging results, with the aim of using TFaaS infrastructure, it is necessary to build a model using Keras/Tensorflow, that explore the world of artificial neural networks. Presently the prototype of TFaaS created has the part of the creation and training of the model that works on local resources but it is planned that it will work on cloud resources, given the architecture construction. Currently, a user can use the TFaaS server from its local terminal or even on a CERN platform and, after loading the model, it is possible to query the server asking if an event, with the related features, is a signal or a background. Then the server gives back a number between 0 and 1 that represents the probability that the event is a signal.
As improvements of this prototype there are the usage of cloud platforms, the usage of GPUs and furthermore TFaaS can be used for other physics use case (also for regression problems). Another aspect that is already explored but not presented in this work is the usage of TFaaS through the application of image classification to classify HEP events.

References

[1] V. Kuznetsov, L. Giommi et al., *Predicting dataset popularity for the CMS experiment*. J. Phys. Conf. Ser., 762 (2016) 1.

[2] D. Bonacorsi, L. Giommi, V. Kuznetsov, T. Wildish, *Exploring patterns and correlations in CMS Computing operations data with Big Data analytics techniques*, in proceedings of International Symposium on Grids and Clouds (ISGC) 2015, March 2015, Academia Sinica, Taipei, Taiwan.

[3] A.A. Alves Jr et al., *A Roadmap for HEP Software and Computing R&D in the 2020s*. Tech. rep. HSF-CWP-2017-01. HEP Software Foundation, 2017. arXiv: 1712.06982 [physics.comp-ph].

[4] D. Bonacorsi, V. Kuznetsov, L.Giommi et al., *Progress on Machine and Deep Learning applications in CMS Computing*, in proceedings of International Symposium on Grids and Clouds (ISGC) 2018, March 2018, Academia Sinica, Taipei, Taiwan.

[5] L. Giommi, *Prototype of Machine Learning 'as a Service" for CMS Physics in Signal vs Background discrimination*. Master thesis, University of Bologna, March 2018.

[6] *Tensorflow*. https://www.tensorflow.org

[7] *TFaaS code*. https://github.com/vkuznet/TFaaS

[8] *First public version of TFaaS*. https://zenodo.org/record/1308049#.W6YVmS1aZ0t

[9] *Keras*. https://keras.io

[10] *Uproot*. https://github.com/scikit-hep/uproot