UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Fisica e Astronomia "Galileo Galilei"

Corso di Laurea Triennale in Fisica

Tesi di Laurea

Optimization of the training of Deep Neural Networks for signal-background discrimination at the LHC

Relatore

Prof. Marco Zanetti

Laureando

Stefano Mancone

Anno Accademico 2017/2018

ABSTRACT

The latest developments in the field of machine learning (ML) can have profound consequences on how to process the information recorded by the fundamental physics experiments like the ones preformed at the Large Hadron Collider (LHC). The dataset output by the latter are indeed characterised by a very large size and tremendous complexity, features which made ML techniques particularly suited for direct application. As shown in previous works, deep networks can tackle problems of signal vs background discrimination starting from low level quantities (e. g. particle four-momenta). The training of such networks requires though a non negligible computing power, which in the end turns into the main limitation of such an approach. The smart usage of computing resources, in particular of clusters of computing nodes, exploiting modern "big data" architectures could overcome this limitation. The proposal for this thesis work consists indeed in setting up, test and benchmark various computing infrastructure options for the training of a deep neural network aiming at the discrimination between signal (New Physics) and background (Standard Model) in a dataset from the CMS experiment at the LHC.

SOMMARIO

Gli ultimi sviluppi nel campo del machine learning (ML) possono avere profonde conseguenze in come vengono processate le informazioni ottenute dai più importanti esperimenti di fisica come quelli che vengono effettuati con il Large Hadron Collider (LHC). I dataset ottenuti in questo modo sono caratterizzati da un ampio e complesso insieme di features, il che rende le tecniche del ML particolarmente adatte ad un'applicazione diretta. Come visto in lavori precedenti, le deep networks possono affrontare problemi di discriminizzazione segnale contro rumore partendo da quantità primitive (e. g. i quadrimomenti delle particelle). L'allenamento di queste reti richiede una potenza di calcolo non indifferente, il che è la principale limitazione di questo approccio. L'uso intelligente delle risorse di calcolo, in particolare l'utilizzo di cluster di computer, può essere utilizzato insieme a moderne tecniche di "big data" per superare queste limitazioni. Lo scopo di questo lavoro di tesi consiste infatti nello sviluppare, testare e confrontare diverse infrastrutture di calcolo per l'allenamento di una deep neural network considerando un problema di discriminizzazione tra segnale (Nuova Fisica) e rumore (Modello Standard) con un dataset dall'esperimento CMS all'LHC.

CONTENTS

Part I

INTRODUCTION

1

THE PHYSICS OF DATA

2 THE PROBLEM

The scope of this thesis is considering a classic signal-background discrimination task by following the path outlined by [2] and [8] to find a reasonably good machine learning classifier built on top of a Deep Neural Network model. The intention is to use the results in the just cited works to fix some hyper-parameters and reduce our degrees of freedom. Then to develop a program to explore the time performance in training such models using clustering and GPU (graphics processing unit) architectures.

2.1 DEEP NEURAL NETWORKS

Machine Learning (ML) is a field of artificial intelligence that uses statistical techniques to give computer system the ability to learn from data without being explicitly programmed. [13] One example of machine learning task is the classification problem. You have, for instance, a dataset with many records. Each of them composed of some features that describe it and one label that indicates in which class among two or more the record belong. Using these information, in the training phase, the ML model learns to classificate the records in the correct class. After the training phase, there is a test one when the classifier performance is been tested with unseen records.

Deep Learning is a sub-field of Machine Learning that deals with algorithms inspired by the structure and function of the brain called deep neural networks (DNNs). More concretely a neural network is a function $f\colon \mathbb{R}^n \to \mathbb{R}^m$ from the space of the features to the space of the labels. It is structured in layers and consist of an input layer of size n, an output layer of size m and several hidden layers of various sizes. Each element of layers is called neuron and each neuron is connected with all previous layer neurons and with all those of the next (Figure 2.1). These connections are mathematically described by weights, such as the input of a neuron is determined by

$$y_i = \mathbf{w}_i \cdot \mathbf{x} + b_i = \sum_{j=1}^{q} w_{ij} x_j + b_i$$

where $\mathbf{x}$ is the input values vector or the outcome of the previous layer, $\mathbf{w}_i$ is instead the weight vector that corresponds to the i-th neuron. A function h is then applied to each of the i neurons, it is called layer *activation function* and it yields as an output $\mathbf{z} = \{z_i = h(y_i)\colon i =$

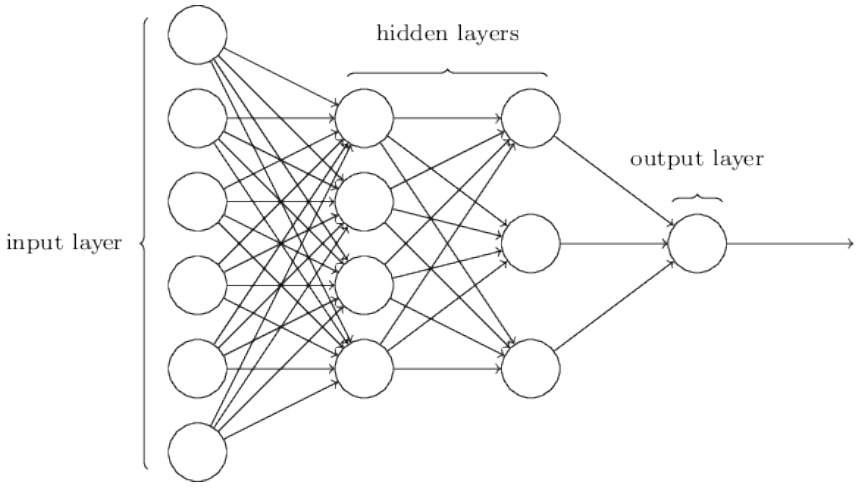


Figure 2.1: Neural Network structure

$1, \ldots, q\}$. This process is being repeated through the network until the output layer is reached. There are some options for initializing weights

- all initialized to a fixed value
- campionated from standard gaussian distribution
- campionated from uniform distribution
- other distributions
- etc.

it has been chosen to campionate them from a standard gaussian distribution.

The next step is understand how well the model predicts labels from data. To achieve this we choose a *loss function* that has the characteristic to become smaller and smaller as the model learn, e. g. perform better. Hence, the job is minimizing this function. In order to achieve it several optimizing algorithms, or optimizers, exist. The description of (some of these/the smoothest one: sgd) is given in the next section.

Had obtained the neural network model, it is remarkable to note that there is a theorem that states [10]

Theorem. *Feedforward networks are capable of arbitrarily accurate approximation to any real-valued continuous function over a compact set.*

This is clearly excellent for us. Despite the fact that under certain hypothesis almost every function can be represented by a neural network, the maximum complexity achievable, however, depend on the choice of some of the hyperparameters. Such as the number of layers, the number of neurons for layer and the activation functions. A bad choice of these hyper-parameters leads to a poor model that can not generalize well the problem (underfitting) or a too complex model

that does not fit well (overfitting). On the other hand, the learning process itself must be performed taking into account a series of issues that can lead to underfitting or overfitting. Refer to [7] regarding this (vast) topic.

2.1.1 *SGD*

The algorithm. A brief derivation. Some formulas.

2.1.2 *others..?*

Same.

2.2 THE PHYSICS OF LHC

The *European Organization for Nuclear Research*, known as *CERN*, is a European research organization that operates the largest particle physics laboratory in the world. *CERN* is the host of the *Large Hadron Collider* (*LHC*), the largest particle accelerator in the world. Along its circumference four detectors are arranged. *CMS*, *ATLAS* and *LHCb* detect only proton-proton collisions while *ALICE* is used with collisions between lead ions too. These four experiments carried on at *LHC* detect a great number of hadrons collisions per second and for each event are record various forms of data. The large size, the variety and the high rate of collisions production at *LHC* are the reasons are named *big data* and these are one of the most suitable dataset for testing efficiency of Neural Networks techniques. On the other hand, employing more and more efficient machine learning algorithms to *LHC* datasets will let high energy physics community to extract from data more information. This would reduce the waste of data.

The *LHC* accelerator itself consists of a 27-kilometer ring of superconducting magnets with a number of accelerating cavities to make beams focused in order to maximize the cross section. Also for curve the trajectory and increase the energy of the particles circulating inside its rings. The preparatory phase pf acceleration begin at the Linac 2 where the protons from hydrogen are accelerated to 50 MeV. The beam is then get injected into the Proton Synchrotron Booster (PSB) where the proton bunches are formed and accelerated further to 1.4 GeV. The next accelerator is called the Proton Synchrotron (PS) which forms the final shape of the beam bunches and kicks the beam up to 25 GeV. The particles are later sent to the Super Proton Synchrotron (SPS) where they are accelerated to 450 Gev, from which they are injected into two pipes of the *LHC*. This is where the beams go in opposite directions to be collided at the experimental sites with up to a center-of-mass energy of $\sqrt{s} = 14$ TeV.

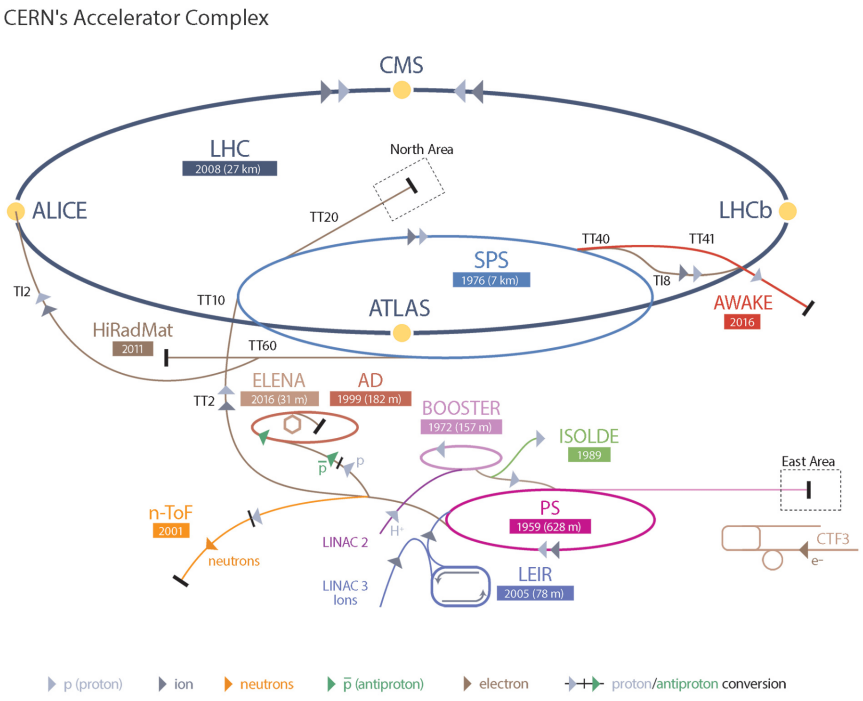


Figure 2.2: *LHC* complex

Taking the CMS detector as an example it is formed by a series of cylindrical sub-detectors able to detect different interactions. This for the purpose of study many aspects of the high energy interactions between protons, from the Standard model parameters to the New Physics theories and their associated particles. Different technologies are used to observe different features of the particles going across them. The closest one is a vertex detector of silicon pixels and strips used to track charged particles near to the point of interaction. Then there is the charged particle tracking chamber which distinguishes positive and negative charges measuring the curvature of the particles in a magnetic field. Next layer consists in two calorimeters: one absorbing the electromagnetic-showers and the other the hadronic-showers. Finally, the muon chambers which absorb the heaviest particles called muons. All these components together are able to absorb all the energy issued by the collision except that carried out by neutrinos. From the pieces of information collected by each layer of the detector, high energy physicists are able to go back to the nature of the particles produced by the collision and study the laws which regulate these processes.

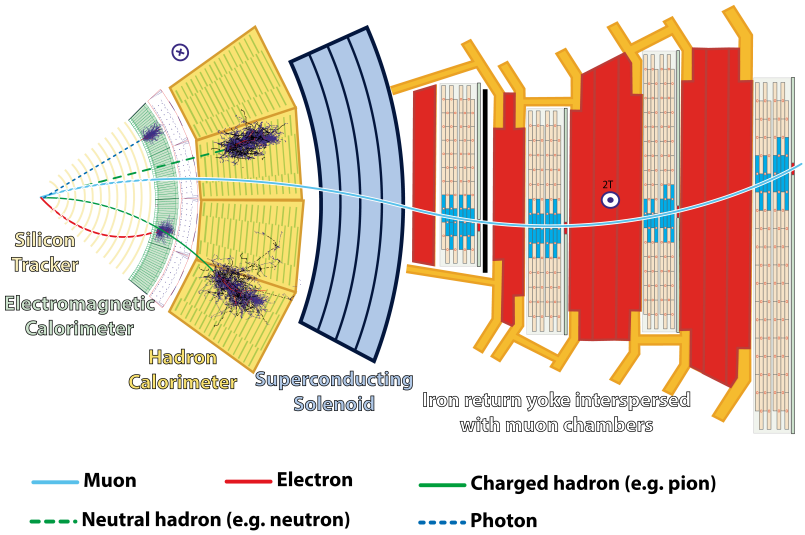


Figure 2.3: A transverse section of the CMS detector

2.3 THE BENCHMARK DATASET HIGGS

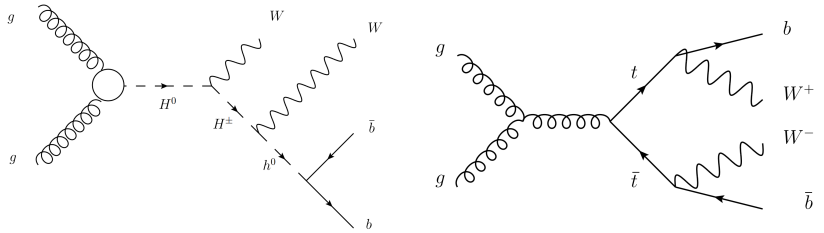
The dataset subject of this work involves a signal process where new Higgs bosons are produced and a background process with identical decay products but distinct kinematic features. The signal process is the fusion of two gluons into a heavy electrically-neutral Higgs boson ($gg \to H^0$), which decays to a heavy electrically-charged Higgs boson ($H^\pm$) and a W boson. The $H^\pm$ subsequently decays to a second W boson and in a light Higgs boson, h^0. The light Higgs boson then decays predominantly to a pair of bottom quarks. The entire process is then:

$$gg \to H^0 \to W^\mp H^\pm \to W^\mp W^\pm h^0 \to W^\mp W^\pm b\bar{b} \tag{2.1}$$

which leads to $W^\mp W^\pm b\bar{b}$. The background process, which mimics $W^\mp W^\pm b\bar{b}$ without the Higgs boson intermediate state is the production of a pair of top quarks, each of which decays to Wb (see figure 1.):

$$gg \to g \to t\bar{t} \to W^\mp W^\pm b\bar{b} \tag{2.2}$$

accordingly, to [2] the events are simulated assuming 8 TeV collisions of protons at LHC. For the benchmark case here, $m_{h^0} = 125$ GeV, $m_{H^0} = 425$ GeV and $m_{H^\pm} = 325$ GeV has been assumed. In order to simulate events for this dataset the semi-leptonic decay mode has been chosen, which is one W boson decaying to a lepton and a neutrino($l\nu$) and the other W boson in a pair of jets (which correspond to a couple of up-type and down-type quarks). Thus, the final products of our decays are $l\nu bjjb$. In [2] has been considered events which satisfy (and we do accordingly):



(a) (b)

Figure 2.4: (a) Diagram describing the signal process involving new exotic Higgs bosons H^0 and $H^\pm$. (b) Diagram describing the background process involving top-quarks (t). In both cases, the resulting particles are two *W* bosons and two b-quarks.

- exactly one electron or muon, with $p_T > 20$ GeV and $|\eta| < 2.5$
- at least four jets, each with $p_T > 20$ GeV and $|\eta| < 2.5$
- b-tags on at least two of the jets, indicating that they are likely due to b-quarks rather than gluons or lighter quarks

where p_T is the momentum transverse to the beam direction and (referred to a polar coordinate system) the polar angle θ is substituted by

$$\eta = -\ln \operatorname{tg} \frac{\theta}{2}$$

there is then the azimuthal angle ϕ. The above requirements are then summed up by 21 low-level features:

- 4 jets, each of them described through 4 variables: p_T, η, ϕ and the b-tag.
- 1 lepton described through 3 variables: p_T, η, ϕ
- 1 neutrino indirectly described by: missing energy magnitude and missing energy ϕ

Theoretical considerations allow constructing new high level features which better highlight the differences between these two processes. The features are the invariant masses of the metastable particles. In particular, for signal processes the following resonant decays have been theorized and the related invariant masses have been computed:

- $W \to l\nu$: the invariant mass $m_{l\nu}$ should show in the known mass of the *W* boson m_W
- $W \to jj$: the invariant mass m_{jj} should show a peak at the known mass of the *W* boson m_W
- $h^0 \to b\bar{b}$: $m_{b\bar{b}}$ should show a peak at m_{h^0}

- $H^{\pm} \to W^{\pm}h^0$: m_{Wbb} should show a peak at $m_{H^{\pm}}$
- $H^0 \to WH^{\pm}$: $m_{WWb\bar{b}}$ should show a peak at m_{H^0}

there represent 5 high-level features. On the other hand, regarding $t\bar{t}$ background, it is expected that:

- $W \to l\nu$ shows a peak in $m_{l\nu}$ at m_W
- $W \to jj$ shows a peak in m_{jj} at m_W
- $t \to Wb$ shows a peak in $m_{jl\nu}$ and $m_{jb\bar{b}}$ at m_t

thus two more invariant masses $m_{l\nu b}$ and m_{jjb} have been computed. In total there are 7 high-level features. Before being passed to the neural networks the dataset had been standardized removing mean and standard deviation from features. The just described sample with 11 million events is available in the UCI machine learning repository.[1]

2.4 THE SETUP (BIGDL?)

The used hardware is composed by a six nodes (computers) cluster hosted by Cloudveneto[2] and a GPU machine under the same domain. In a cluster, one node is called cluster manager or master, while the others workers or slaves. In this case, the master is a computer with 2 cores and 4 GB of RAM while the 5 nodes are computers with 8 cores and 16 GB of RAM, for a total of 40 cores and 80 GB of RAM for the slaves. The master role is to dispatch individual small tasks to the slaves that send back the processed results. This paradigm is carried out by Apache Spark[14] which is a *unified analytics engine for large-scale data processing*[3]. Spark applications run as independent sets of processes coordinated by the `SparkContext` object in the main program (*driver program*). Specifically, to run on a cluster, the SparkContext connect to a *cluster manager* which allocate resources across applications. Once connected, Spark acquires *executors* on nodes in the cluster, which are the processes that run computations and store data to process. Next, it sends the application code to the executors. Finally, SparkContext sends *tasks* to the executors to run.

The driver program has been written in Python[4] which is a high level programming language widely used nowadays for data science topics. In the program, through Spark application program interface (API), instructions are sent to workers. Spark use as backend Java, meaning that the execution of the Spark code is committed to a Java interpreter which is a lower level programming language. This is a common programming paradigm, the code is written in a higher level

1 https://archive.ics.uci.edu/ml/datasets/HIGGS
2 https://cloudveneto.ict.unipd.it/
3 https://spark.apache.org/
4 https://www.python.org/

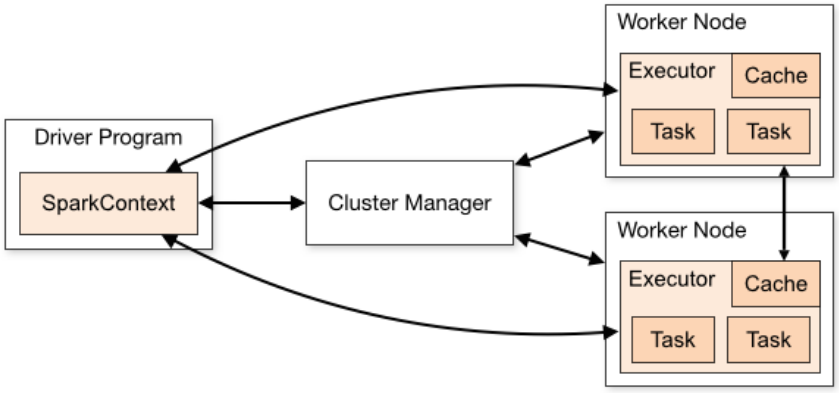


Figure 2.5: The Spark architecture

language while the actual execution is performed by a lower one. Python itself is written in C++ which is one of the oldest and still very used low level programming language.

While the framework used is Spark, the utilized library for implementing the distributed Deep Neural Network models is BigDL [4] from Intel Corporation. Instead, for the standalone and the GPU version has been used Keras [3] with Tensoflow [1] backend.

The work done in [8] and [2] has been used for fix some hyperparameters:

- Number of layers: 3
- Neuron for layer: 300
- Dropout: 0
- Layer activation function: *tanh*
- Output activation function: *sigmoid*
- Learning Rate: Default value of the optimizer

For what concern the hidden layers activation function the tanh function has been chosen. Instead, for the output neuron a sigmoid has been taken.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$\mathrm{sig}(x) = \frac{1}{1 + e^{-x}}$$

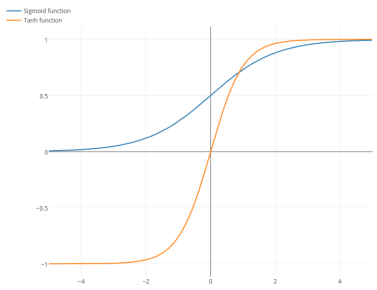


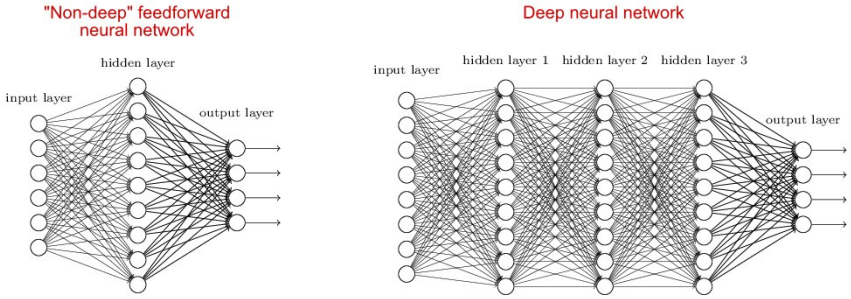
Figure 2.6: tanh and sigmoid functions comparison

The sigmoid function has the property of returning an output in the interval $(0, 1)$ it can be seen as the confidence the network has regarding the 0 or the 1 classification. It means that the closer to 1 is the output the more confident the network is about the event classification as a signal, vice versa a result near 0 means a background event. It is a very common choice for an output function for a Machine Learning classifier.

3

NEURAL NETWORKS PERFORMANCE ANALYSIS

Different variety of Neural Networks exist. The used one in this thesis, and the description that was given, is about Feedforward Networks. Depending on the number of hidden layers they are called Shallow Neural Networks (1) and Deep Neural Networks (> 1).



The main focus of [2] and [8] is to demonstrate that performance of the two kinds is comparable when using high level features for the shallow one and low level features for the deep one. To achieve this it is need a *metric*, a measure of how well the classificator is performing. Evaluating the metrics over the training, validation and test sets is the primary technique for point out how the training process is going. Therefore, is crucial to use the most significant ones.

The default metric provided by Keras [3] is the binary accuracy. Given an event if the output is greater than 0.5 then it is classified as signal, background otherwise. The right label is then compared and the prediction correctness is determined. Done it for the entire considered subset and found the right prediction percentage the (binary) accuracy is obtained. Another metric widely used in Machine Learning is the result loss function considered as a mean over a subset of events. This is the number that the optimizer itself is trying to minimize so is a direct performing rate. Despite this, it is significant in the training process only. It can not be used in the final classifier rating not having a statistical meaning.

A more statistically significant approach is to find the *AUC* score, i. e. the *Area Under the ROC Curve*, by drawing and integrating the *ROC Curve* that stands for *Receiver Operating Characteristic*. Drawing the ROC curve is a graphical method to observe binary classifiers efficiency. It is done by plotting percentage successfully signal prediction (signal efficiency) versus the percentage successfully background discovery #(classification?) (background rejection). The various points of the graph are obtained by varying the threshold in the interval $[0, 1]$

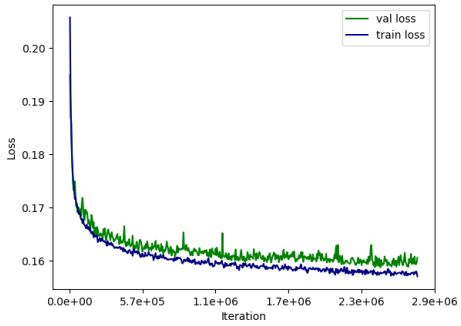


Figure 3.1: loss function during training

beyond which the Neural Network output it considered signal, note that the binary accuracy is essentially the ROC point which threshold is 0.5.

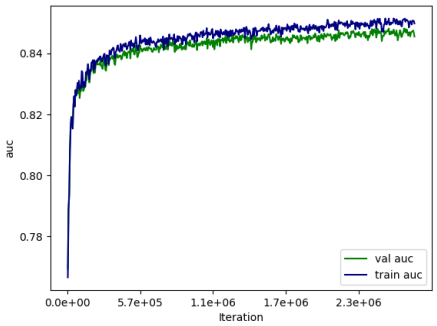


Figure 3.2: auc score during training

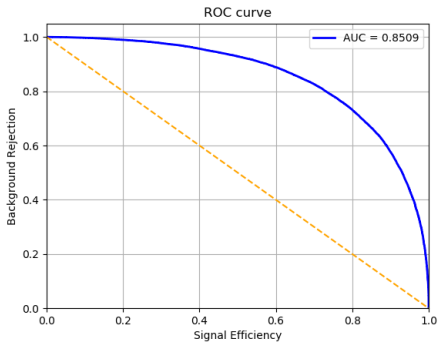


Figure 3.3: test set auc score

One more quantity that highlights the model behavior is called *Figure Of Merit* (*FOM*). It is defined as $\mathrm{FOM} = \frac{S}{\sqrt{B}}$ where S is the right classified signal events total number and B is right classified background events total number. The term $\sqrt{B}$ represents the error on the number of background events assuming they follow a Poisson distribution. More precisely, to state whenever signal events S are actually a resonance or only statistical fluctuations, the effective error should take into account also the error on the number of signal events, that is $\sqrt{S}$, as signal follow poissonian distribution too. So the correct expression for the error would be $\sigma = \sqrt{B+S}$. Nonetheless since $S \ll B$ the error can be approximated as $\sigma \approx \sqrt{B}$.

The data were simulated as described in Section 2.3 and they were built such that signal and background would have the same probability. The consequence is that the number of signals predicted events should be quite the same of background predicted. In data collected by experiments, a signal of new physics is usually a rare phenomenon which competes with lots of standard processes included in background. Thus, the real cross section of signal events may be so small

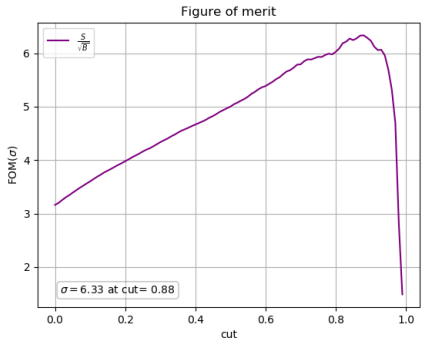


Figure 3.4: Figure of merit of the test set

that it may be confused with background. FOM computation allows finding the optimal cut point to put in evidence the presence of signals. In fact, maximize this quantity considered as a function of the classifier acceptance threshold means maximizing signal and background distribution over the phase space expected by the theoretical model. A good cut point choice allows physicists to determine whenever a difference in experimental distributions can be identified as a new signal or not.

The last performance indicator that has been considered is the histogram of the predicted events over the possible model outcomes. The network discriminates well if the predicted signal distribution present a peak towards 1 and the predicted background distribution towards 0.

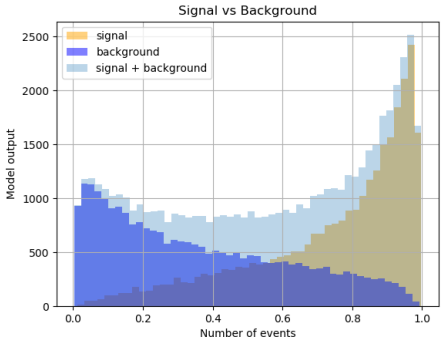
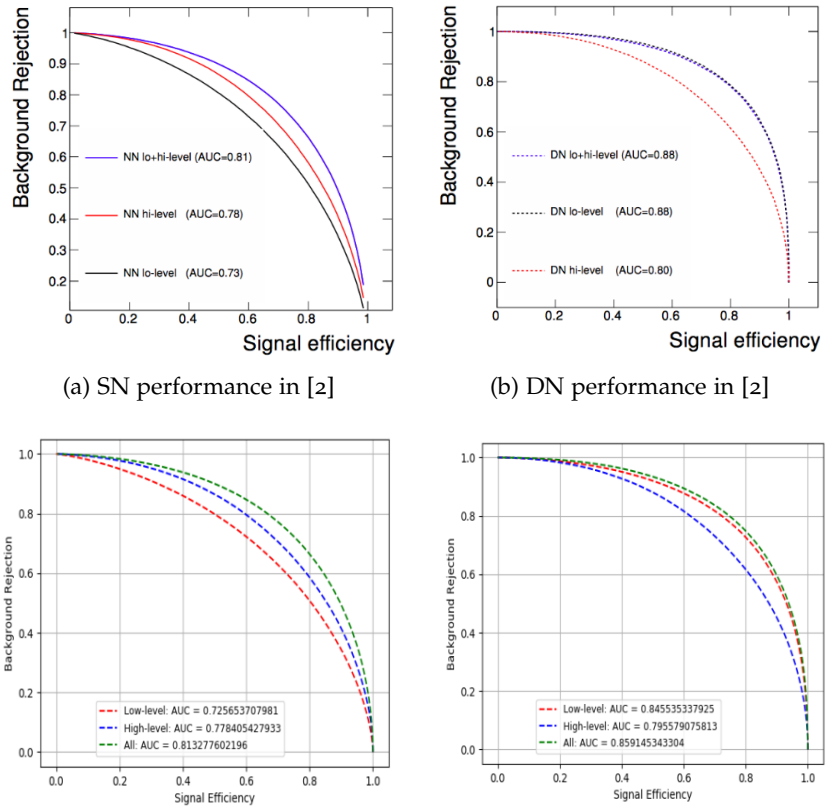


Figure 3.5: Signal versus background in test set

3.1 SHALLOW AND DEEP NEURAL NETWORKS PERFORMANCE

In [8] some experiments have been done varying the number of layers, neurons for layer and using various regularization techniques. The most interesting result obtained is that the Deep Neural Networks trained and tested with low level features perform the same (or better) than the Shallow Neural Networks with high level features, which is



(a) SN performance in [2]

(b) DN performance in [2]

(c) SN performance in [8]

(d) DN performance in [8]

Figure 3.6

excellent given the not negligible work needed to develop high level features. The same results are also found in [2]. In Fig. 3.6 and in Tab. 3.1 are shown a comparison of the performance with shallow neural networks (SNN) and deep neural networks (DNN) in [2] and [8] for three sets of input of features: low level features, high level features and the complete set of features. As can be seen, a shallow NN trained using only the low level features performs significantly worse than one trained with only the high level features. This is a well-known problem with shallow learning methods, and motivates the calculation of high level features. Methods trained with only the high level features, however, have a weaker performance than those trained with the full set of features, which suggests that despite the insight represented by the high level features, they do not capture all the information contained in the low level features. The deep learning techniques show close performance using the low level features and the complete features, suggesting that they are automatically discovering the insight contained in the high level features. The slightly different performance in [2] and [8] are explainable due to the fact in [8] the algorithms can handle the training for about 40-60 epochs before overfitting, instead [2] are able to train for 200-1000 which is a great difference.

Technique	Low level	High level	Complete
SNN_{auc}	0.733	0.777	0.816
DNN_{auc}	0.880	0.800	0.885
SNN_{DS}	2.5σ	3.1σ	3.7σ
DNN_{DS}	4.9σ	3.6σ	5.0σ

(a) AUC and discovery significance in [2]

Technique	Low level	High level	Complete
SNN_{auc}	0.675	0.763	0.812
DNN_{auc}	0.846	0.796	0.858
SNN_{DS}	3.34σ	3.69σ	4.11σ
DNN_{DS}	4.44σ	3.85σ	4.66σ

(b) AUC and discovery significance in [8]

Table 3.1

Even if the quantitative results differ, the qualitative ones are the same.

To understand if the obtained results can help to discriminate new signals assumption was made that for 100 signals events were 1000 background events as can be seen in Fig.3.7. In order to simulate the physical reality prediction histograms were re-normalized. The Tab.3.1 show then that the discovery significance, which is a standard metric in high-energy physics, can significantly improve even with small increases in AUC.

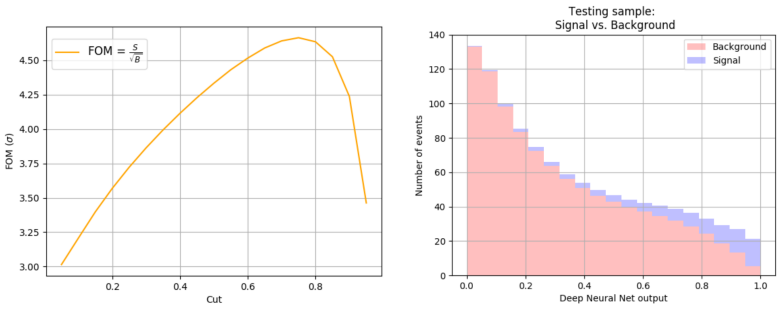


Figure 3.7: Figure of merit obtained after normalization in [8]

Part II

RESULTS

4

CLUSTER SCALABILITY

In the thesis early stages, experiments have been done with the Distributed Keras framework [11] developed at CERN but BigDL seemed more promising so we continued with this library. Initially training was performed with a cluster formed of an 8 cores master node and 16 GB (*xlarge*) and 5 slaves with 2 cores each and 4 GB (*medium*). Then it became clear that, for the task, it was better an architecture with a *medium* master and 5 *xlarge* slaves due to the fact that all the heavy lift is performed by workers. On the GPU architecture side we had available an *Nvidia Titan Xp*[1] and an *Nvidia RTX 2080 FE*[2] graphics cards.

We tested our model with several optimizers implemented by BigDL and Keras libraries. The standard optimizer SGD, Adam[12], Adamax[12], Adadelta[15] and Adagrad[5]. Cluster performance is evaluated between different numbers of nodes and available GPUs. Results are shown in Fig. 4.1 and Fig. 4.2.

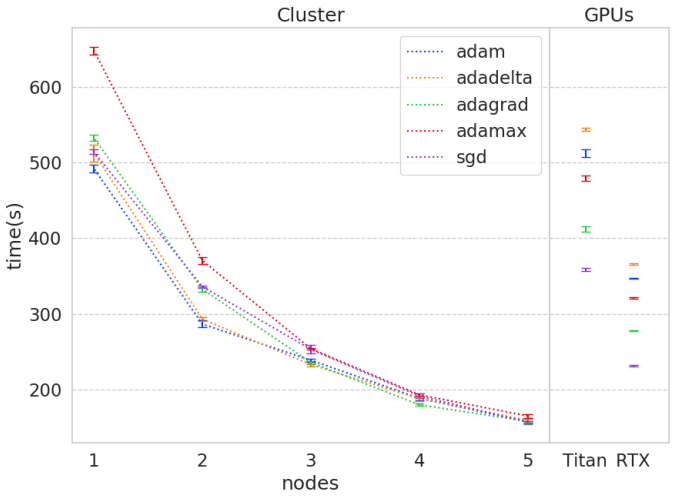


Figure 4.1: Number of nodes versus execution time for 1 epoch

1 https://www.nvidia.com/en-us/titan/titan-xp/
2 https://www.nvidia.com/en-us/geforce/graphics-cards/rtx-2080/

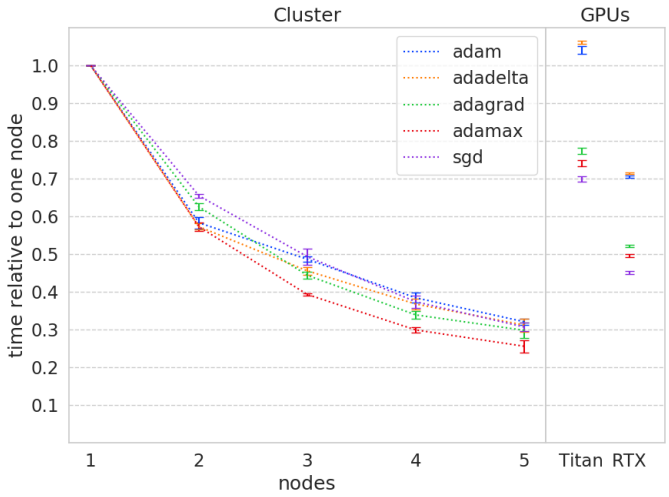


Figure 4.2: Relative time regard execution with one node

As shown in the Fig. 4.1 the performances improve by adding nodes, as intended. The improvement is not linear, the difference in time performance between optimizers decrease with increasing number of nodes. A simple interpretation could be that with more nodes the training bottleneck is on the network side so the difference in time performance between optimizers become less significant. To support this hypothesis in Fig. 4.2 it can be seen that performance increase approximately by 30% - 40% for each added node but relative improvement shrink with the increment of the nodes. This may be due to the fact that the increase in the number of nodes is limited by the time required for their communication with the master.

The graphs also show that latest *Nvidia* consumer-grade GPUs have comparable performance with multi-cores CPUs. Remembering that every node has 8 it can be seen that in this test the *Titan Xp* can compete with an 8 - 16 cores cluster, depending on the chosen optimizer. The *rtx 2080* gets better results and is comparable with the performance of a 16 - 32 cores cluster. This difference could be related to the different GPUs *Pascal* [9] and *Turing* [6] architectures. It could be also noted that in this case the performances are strongly influenced by the chosen optimizer. Not easily explainable this aspect should be further explored.

BIBLIOGRAPHY

[1] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. https://www.tensorflow.org/. Software available from tensorflow.org. 2015.

[2] P. Baldi, P. Sadowski, and D. Whiteson. "Searching for exotic particles in high-energy physics with deep learning". In: *Nature Communications* 5, 4308 (July 2014), p. 4308. DOI: 10.1038/ncomms5308. arXiv: 1402.4735 [hep-ph].

[3] François Chollet et al. *Keras*. https://keras.io. 2015.

[4] Jason Dai et al. "BigDL: A Distributed Deep Learning Framework for Big Data". In: *CoRR* abs/1804.05839 (2018). arXiv: 1804.05839.

[5] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research.* 12. 2121-2159. (2011). eprint: 12.2121-2159..

[6] Nick Stam Emmett Kilgariff Henry Moreton and Brandon Bell. *NVIDIA Turing GPU Architecture*. https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf. 2018.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. 2016.

[8] Gaia Grosso. "Deep Learning techniques to search for New Physics at LHC". Thesis. University of Padua, 2017.

[9] Mark Harris. *GP100 Pascal Whitepaper*. https://nvidia.com/content/pdf/tesla/whitepaper/pascal-architecture-whitepaper.pdf. 2018.

[10] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. *Multilayer feedforward networks are universal approximators*. Vol. 2. 5. 1989, pp. 359 –366. DOI: https://doi.org/10.1016/0893-6080(89)90020-8.

[11] CERN IT-DB Joeri R. Hermans. *Distributed Keras: Distributed Deep Learning with Apache Spark and Keras*. https://github.com/JoeriHermans/dist-keras/. 2016.

[12] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2014). arXiv: 1412.6980.

[13] Arthur L. Samuel. "Some studies in machine learning using the game of Checkers". In: *IBM journal of research and development* (1959), pp. 71–105.

[14] Matei Zaharia et al. "Spark: Cluster Computing with Working Sets". In: HotCloud'10 (2010), pp. 10–10.

[15] Matthew D. Zeiler. "ADADELTA: An Adaptive Learning Rate Method". In: *CoRR* abs/1212.5701 (2012). arXiv: 1212.5701.