Summer Student Intership Report

Jet classification with GarNet

June 2021 – August 2021

Julia Carvalho Leite

Local Supervisor: Thiago Tomei

Supervisor: Maurizio Pierini

Abstract

In this report, it is passed though all important aspects of Hadronic Jet produced in LHC collision events and collisions simulations approach using the main programs, all the way through tagging heavy particles in Fat jets algorithm process. Then, it is briefly discuss the Machine Leaning using Neural Networks through classification task.

The final and most important of this project development proposal is to use the learned Machine Leaning and Neural Network knowledge to adapt the GarNet layer of Neural Network on classification of fat jets. In this work, it was analyzed just the tagging classification between top quark jets and gluon jets data, taken from hls4ml dataset.

Summary

Development

Intro to Particle Physics and the LHC

In principle, It's always useful to understand the big picture of this work, and thus, improve the knowledge and understand better the context of experiment that this study takes place. Therefore, it's exposed a brief explanation of the study object and the experiment that this report has connection with.

Hadronic Jets

The Hadrons are composed by quarks, which can be divided in Baryons (ex: protons and neutrons) and Mesons (ex: pions). Quarks are elementary particles related to the strong interaction, which exchange gluons (the gauge boson of the strong interaction) intertwining the quarks. Each gluon carries colours charges. All Hadrons are unstable, and thus, decay. Strong decays are much faster than the decay related to the weak force [2], for example.

Hadrons interactions with other particles can be described by the Feynman diagrams, which is a pictorial representation to the mathematical expressions that defines the particles interaction behavior. In this interactions, every time a quark or gluon detaches from a Hadron it pulls particles from the vacuum because of the strong interaction, since they have a net colour charge and cannot exist freely due to colour-confinement. Thus, it looks like a spray of flying particles. This is called Hadronic Jets, which conventionally can be represented as a cone of particles spray which the tip is the initial interaction point and the cone base is where we can see the particle jets produced, on a lateral view of the cone (Figure 1[1]). Other common view of the particle jets is by the cross section of the particle collider (Figure 2[2]).

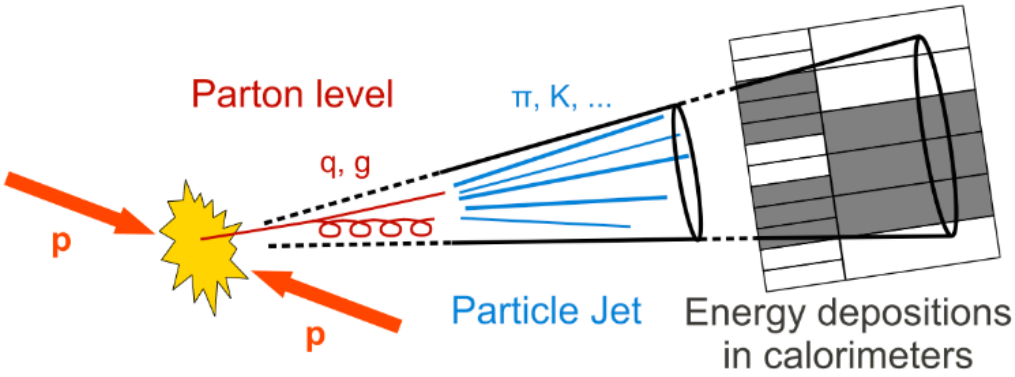

Figure 1: Sketch of a pp-collision and resulting collimated particles, a jet.[1]

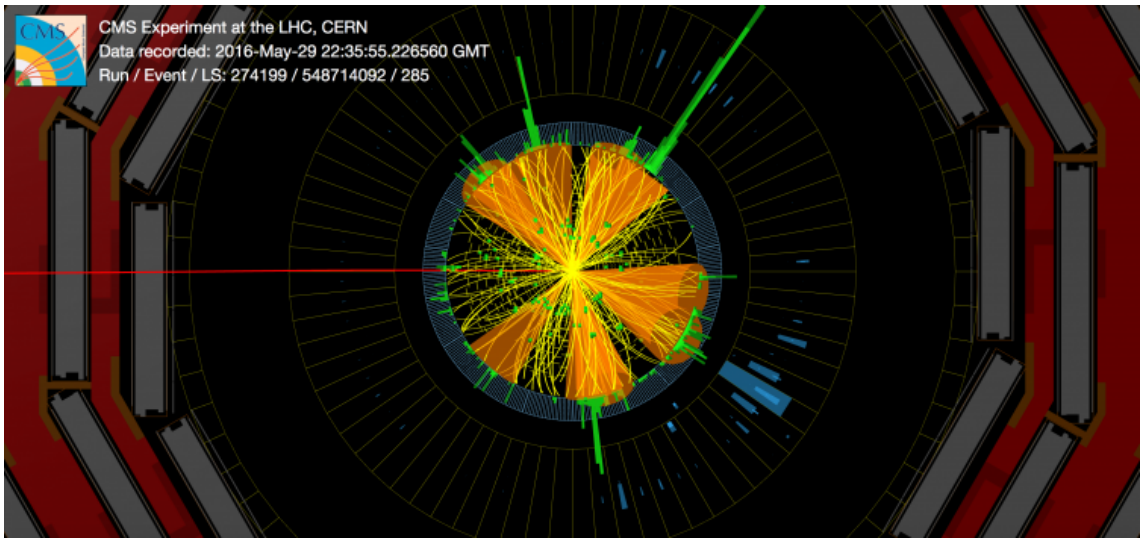

Figure 2: Cross section CMS view, with the cones representation in orange. Jet originating from a b quark (b jet), a muon and a neutrino (that cannot be directly detect).[2]

CMS experiment

Compact Muon Solenoid (CMS) experiment focus on detect muons, mainly because one of the clearest "signatures" of the Higgs Boson is its decays into four muons. It happens by the production of ZZ* boson followed by its decays in muons, which the Z* boson is a virtual particle given by the uncertainty energy and time principle. Thus, we can see that the H -> Z boson is the on-shell interaction (respect $E = \sqrt{p^2 + m^2}$ in natural units), and the Z* boson "appearing" is the off-shell interaction (do not respect the relativistic energy conservation in a small amount of time).

The CMS interesting also includes to study the antimatter by b quarks observations, since neutral particles oscillates between two eigenstates, like what happens with neutrinos, but obviously b quarks are easier to be observed and it involves new physics to be studied.

Atlas and CMS, both placed in the Large Hadron Collider (LHC) complex, have a cross-checking data to ensure the experiment results. Besides, both have different equipment and analysis methods, which certify the result reliability. For example, the Atlas' electromagnetic calorimeter detector is more sensitive to electromagnetic and neutral particles signals, while CMS has a more reliable silicon tracker to charged particles signal.

Nowadays, CMS experiment is in its third run preparation, which will increase the luminosity and energy levels of the experiment, producing more data to be observed and studied. This can provide new aspects of the collision experiments related to hadronic jets, since it is related to the Higgs production and its theoretical implications need to be better understood in the Standard Model.[3]

- **Detector: interesting points**

The Electromagnetic Calorimeter detects only lighter particles, since to one particle to be

[1] `https://cms.cern/news/jets-cms-and-determination-their-energy-scale`

[2] `https://cms.cern/news/lhc-powerlifting--searching-simultaneous-production-four-times-most-massive-elementary`

[2] David Tong in "Particle World" lecture, on Summer Student Lecture 2021

[3] Matthew Philip Mccullough in "Beyond The Standard Model" lecture, on Summer Student Lecture 2021.

detected it needs to shock with the detector, what makes the particle to be destroyed. Thus, heavier particles will shock with this detector, but if it is massive enough it will pass thought it without big energy changes (muon for example), and consequently, not being destroyed.

The magnet used on CMS experiment is a superconductive magnet because it needs to be powerful enough to curve the massive particle trajectories, like muons, to determine the momentum of each particle using the magnetic force equation that acts on a moving particle. If the cooling system of the experiment fails, the superconductive magnet suffers what is called Quench, which is a process that makes the magnet resistivity increase, losing its superconductivity property as the temperature increases on the magnet coils, eliminating the magnetic field and causing superheating, enabling an explosion.

Other interesting fact is the reason why the gas detector (muon detector) and the silicon detector (tracker) is placed on the end of the yoke and in the begin of the yoke, respectively. In essence, both has the same logic of detection: a polarized middle when the particle pass thought it. But the gas detector is less sensitivity than the silicon detector given its matter state and properties, for example the silicon matter properties enable micro signals detection. Therefore, considering the solid angle of the collision experiment, we can affirm that as closer the detector is from the collision, smaller is the solid angle, which makes necessary to have a more sensitive detector like silicon in the inner layer of the yoke. Because the outer layer of the yoke has a bigger solid angle, by the same logic, it doesn't require the detector to be extremely sensitive as the silicon layer, and thus, it's used the gas detector.

Simulating collisions with Pythia

To understand how the jets simulations work, it was introduced the Pythia[4] program. It is a standard tool for high-energy collision, which reproduce the physical aspects of "the evolution from a few-body high-energy (“hard”) scattering process to a complex multihadronic final state".[10] Other examples of programs with the same proposal are Herwig[5] and Sherpa[6]. All this programs basically are categorized as a "General purpose Monte Carlo event generators", which in simple means they simulates all steps of the event going from the Feynman Rules for the hard process all the way thought to the set of hadrons that are actually seen in the detector[7].

The steps simulated by these programs, basically, are[7]:

- The Hard (scattering) process: it is defined by a high momentum transfer collision calculated through Feynman diagrams. In contrast there is the soft process, defined by a low momentum transfer. To define the difference between this two in the algorithm since both happens in collision event, it's needed to defined a cut to separate the two phenomenon.

- Parton shower: colour and electric charged particles radiates other particles if accelerated. So, for example: accelerating colour charged, it radiates a gluon and in that process, the

[4] `https://pythia.org/`
[5] `https://herwig.hepforge.org/`
[6] `https://sherpa.hepforge.org/doc/Sherpa.html`
[7] Mike Seymour in "Making Predictions at Hadron Colliders" lecture, on Summer Student Lecture 2019.

gluon itself is colour charged and it's accelerated as it's produced, then the gluon themselves can radiate further gluons. Thus, we get much more extending cascade of partons(quarks and gluons joined) in this process accompanying any colour charged particles. This cascade phenomenon is called parton shower.

- Hadronization: in short, since quarks and gluons have a net colour charge and cannot exist freely due to colour-confinement, they come together to form colour-neutral hadrons. That process is the hadronization (as we have discussed earlier), leading to the collimated sprays of hadrons.

- Underlying event: This is the formation of soft particles (particles not involved in the hard process) that makes it difficult to clearly observe the hard process that we are interested in. This happens because the protons remainings in the collision interact with each other, producing this soft particles that "hide" the total picture of the hard process. This process of parton-parton interact (proton remaining) that producing soft particles is called multi-parton interaction model.

- Unstable particle decays: unstable particles produced in the collision event will decay, therefore, the last step the algorithm reproduce is to simulate all the possible decays that these particles will produce.

These programs do not just simulated collision, but help us to identify the jets clustering on the collison though experiment data too, depending on the adaptations and information requested by the user available in the program.

Practice: What is the Jet localization?

Here, we use the Pythia associated to the FastJet algorithm (will be discussed in the next section) to identify the localization of jets. For practice, it was used the decay $W^- \rightarrow e^- + \bar{\nu}_e$ and its jets production resulted by $p\bar{p}$ collisions at $\sqrt{s} = 1.96$ TeV (correspondint to the main71 example of Pythia). After comprehend the code of this example and how it works, it was extracted important jet localization information, described below:

- Pseudorapidity η and Azimuthal Angle ϕ

  This two variable are used to define the localization related to the beam in the LHC complex. Both are defined by the spherical coordinates, where the azimuthal angle ϕ is identical to the spherical coordinate ϕ itself as indicated in Figure 5. However, instead of using the radius P or the θ variable, it's more useful the psedorapidity η defined by

$$\eta = -\ln\left[\tan\left(\frac{\theta}{2}\right)\right] \tag{1}$$

  Since pseudorapidity tends to the rapidity in the ultrarelativistic limit, it becomes useful to use it on hadronic jets because the rapidity is invariant under transverse boosts if the

[8] https://wiki.physik.uzh.ch/cms/latex:tikz

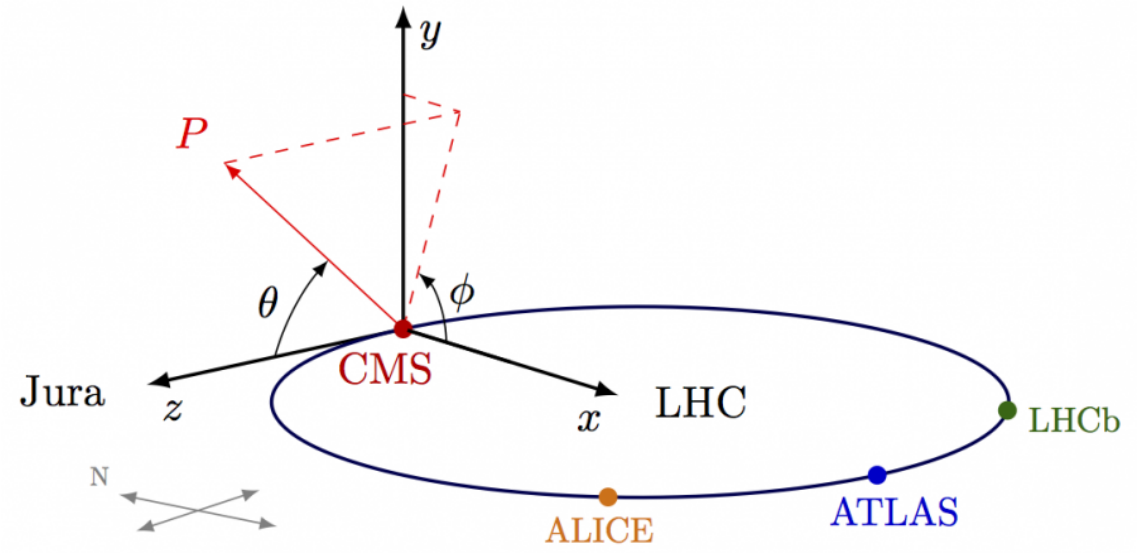

Figure 3: Referential coordinates used on CMS. The z axis points to the Jura mounts in the image and correpond to the direction of the beam. The three momentum vector is indicated by the red vector P.[8]

particles are massless. The rapidity, in this case, is defined by $\Delta R = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}$. Note that ϕ determines the jets localization under perspective of the cross section of CMS and η determines the localization under longitudinal perspective of the CMS.(Figure 4)

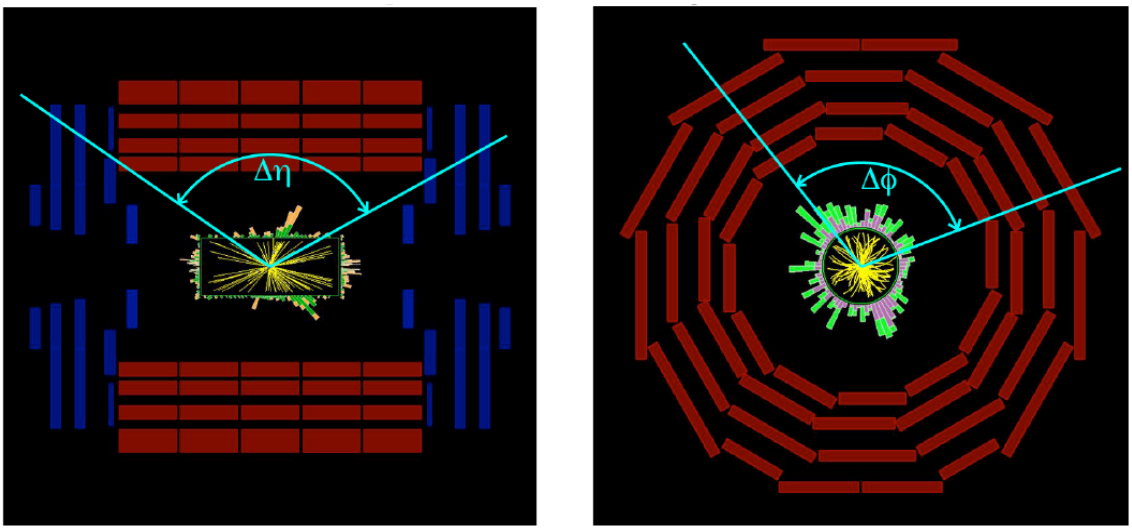

Figure 4: Perspectives of localization on the collision event in the detector.[9]

The same was data was extracted to each component of its respectively jet, then was plotted each component and its jets information to see the localization relation of them calculated by the program. It was taken one collision event with 4 jets and it was extracted the information just of this situation for better visualization (Figure 5). Analysing the print localization, we can presume that the jet separation though closer components clustering was well done by the program on this reduced group to exemplification.

[9] https://cornellmath.files.wordpress.com/2010/09/angles.png

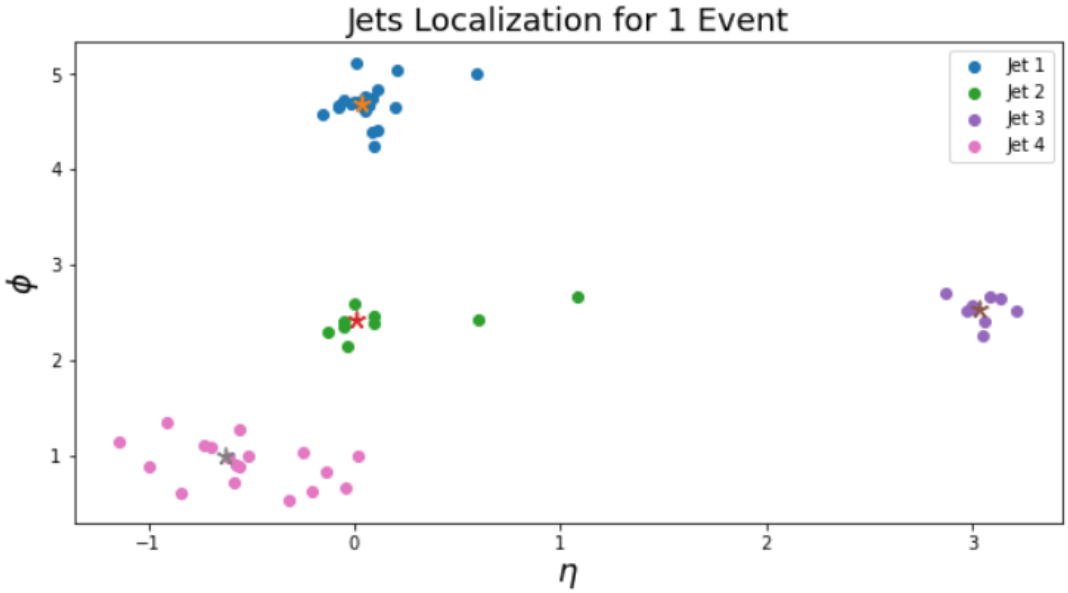

Figure 5: Each star represent jet information. The components of each jet are described by points with the same colour.

Practice: How does relative variables histograms of jet components are?

Furthermore, it was extracted the relative information of all component of the jets produced in 10.000 collision events to define histograms. The information taken of the program is:

- Transverse Momentum pt

  This variable is the last needed variable to determine the momentum of a hadronic event. The transverse momentum is a component perpendicular do the beam line define by

$$pt = \sqrt{p_x^2 + p_y^2} \tag{2}$$

  It's interesting to note that if case of the relativistic particles with pt>> m (mass of the particle), the pseudorapidity and the rapidity is equal. This reinforce the fact that is interesting to us to use the pt, eta and phi variables. Besides, we can see that the three momentum will be defined by p_x = pt cosϕ, p_y = pt sinϕ and p_z = pt sinhη.

- Relative variables

  Consequently, the relative variables are relevant because if we know jet components variables values relative to the jet itself variables values, we can analyse the energy-momentum conservation. Thus, the relative variables to a component i are:

$$pt_{relative,i} = \frac{pt_i}{pt_{jet}} \tag{3}$$

$$\eta_{relative,i} = \eta_i - \eta_{jet} \tag{4}$$

$$\phi_{relative,i} = \phi_i - \phi_{jet} \tag{5}$$

Note that pt variable has to be a fraction of the total transverse momentum of the jet itself, since its components compose the jet total transverse momentum. On the other hand, η and ϕ needs to be the difference since its related to the jet "middle" localization and its component distance with it (Figure 4).

.

The resulting histograms can be seen below (Figure 6, 7 and 8). We can see that the pt histogram looks like a exponential distribution (Figure 6), while the η (Figure 7) and ϕ (Figure 8) look like a delta distribution around zero. See that the three histograms shows that pt, η and ϕ are more frequent around zero. So, the energy-momentum is probabilistic conserved, since the zero value is the biggest frequency on all three variables that defines the three momentum P.

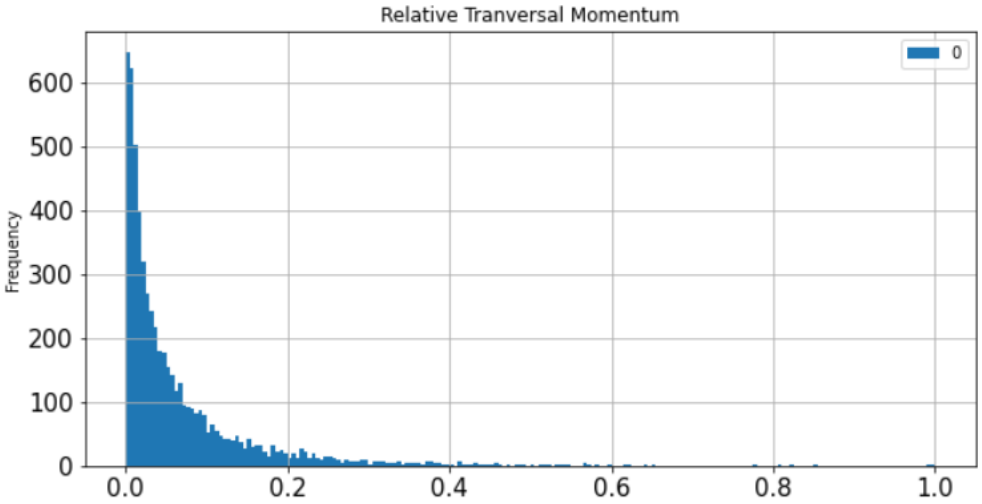

Figure 6: It seems there is a exponential distribution and the frequency of zero transverse momentum is much higher.

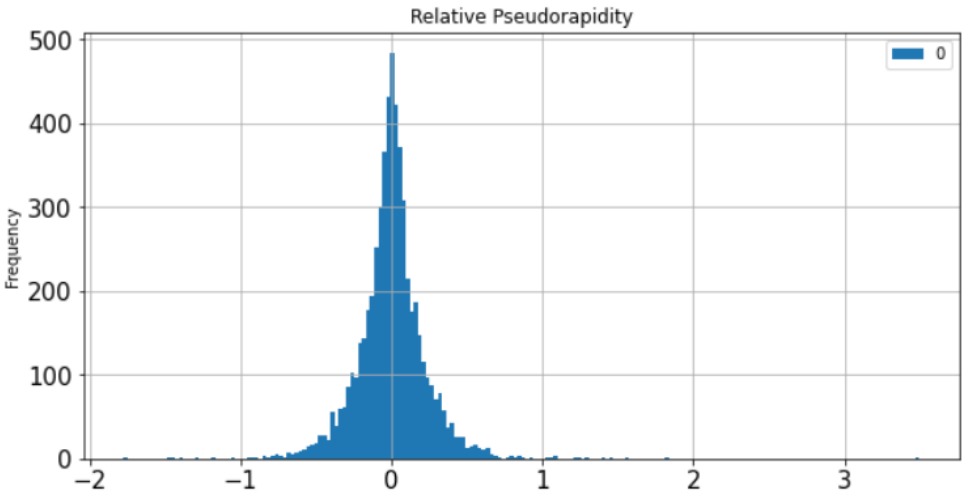

Figure 7: The distribution is similar to a delta around zero.

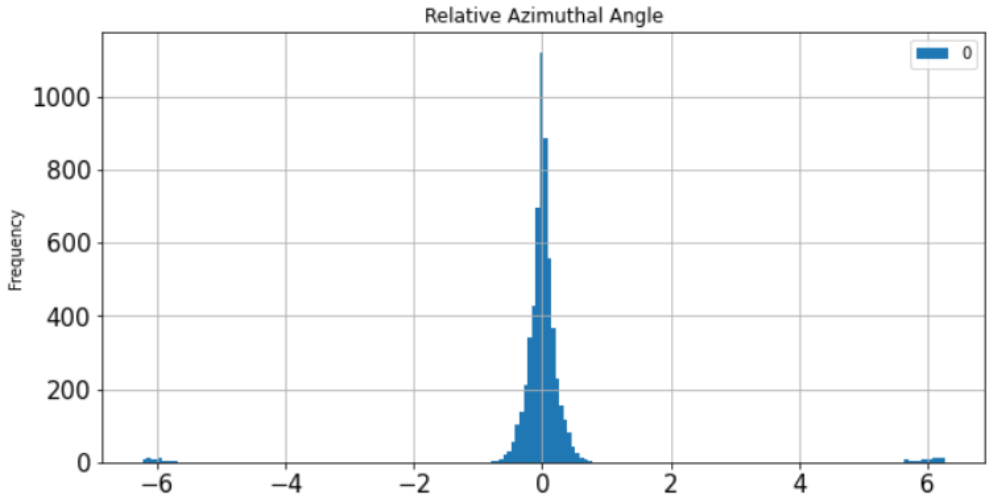

Figure 8: This histogram is done with azimuthal angle in range $(0, 2\pi, 4\pi, ...)$. It also seems with a delta distribution around zero.

Doing the same with hls4ml

The hls4ml [6] is a simulated dataset of gluon jets, top quark jets and light quark jets [1] and it is available in Zenodo [10]. Each jet set has around 30 particles and the data contain each particle relative pt, relative η and relative ϕ. Besides, each particle has a variable called *mask* which indicates if it is a "true" particle when the number is positive.

Using gluon jets data only, it was done the same above analysis to see the aspects of a big set jet. In Figure 9, we can see all jet set localization in one plot, which can be similar to what the detectors sees, demonstrating how difficult can be to separate the different jet clusters.

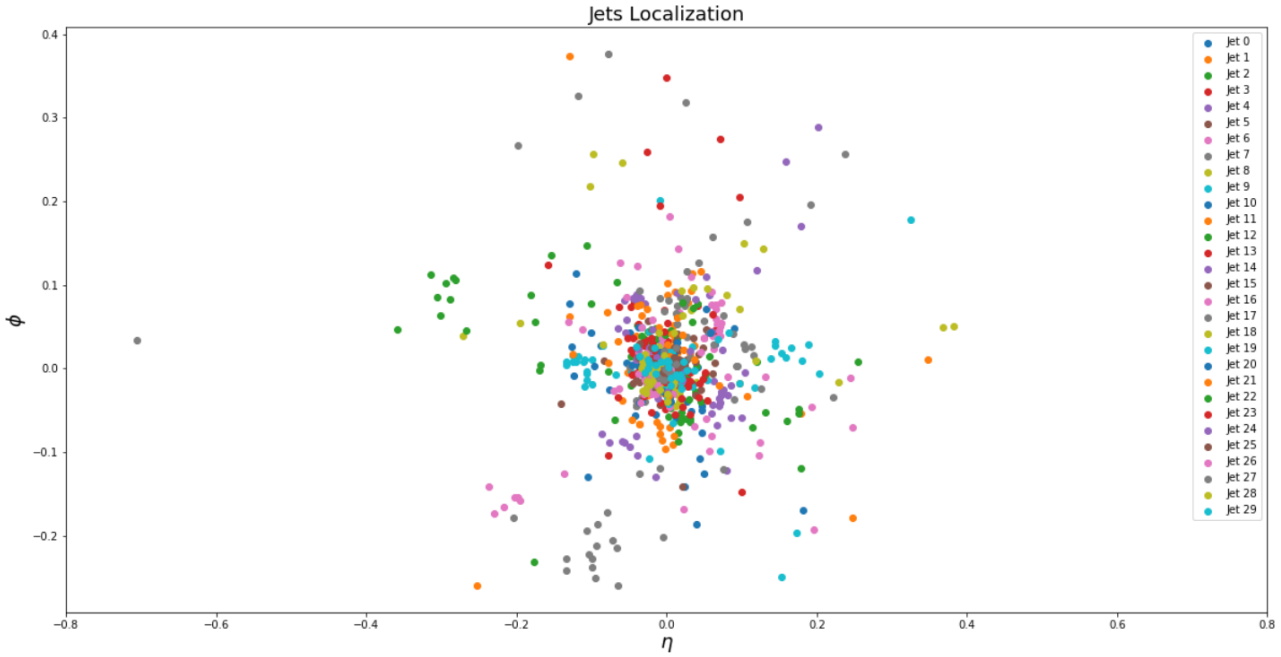

Figure 9: Gluon jets localization using hls4ml dataset.

By the histograms (Figures 10, 11 and 12), we can see that the hls4ml dataset follows that same

[10] https://zenodo.org/record/4834876#.YSKjGvfQ-XL

previous conditions discussed in the relative variables histograms of the jet components (Figures 6, 7 and 8).

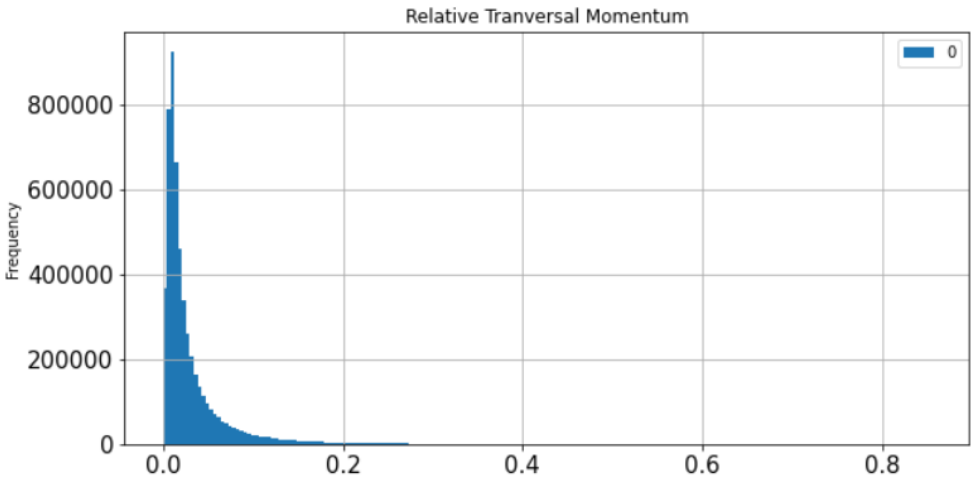

Figure 10: The distribution seems a exponential with the biggest frequency much closer to zero.

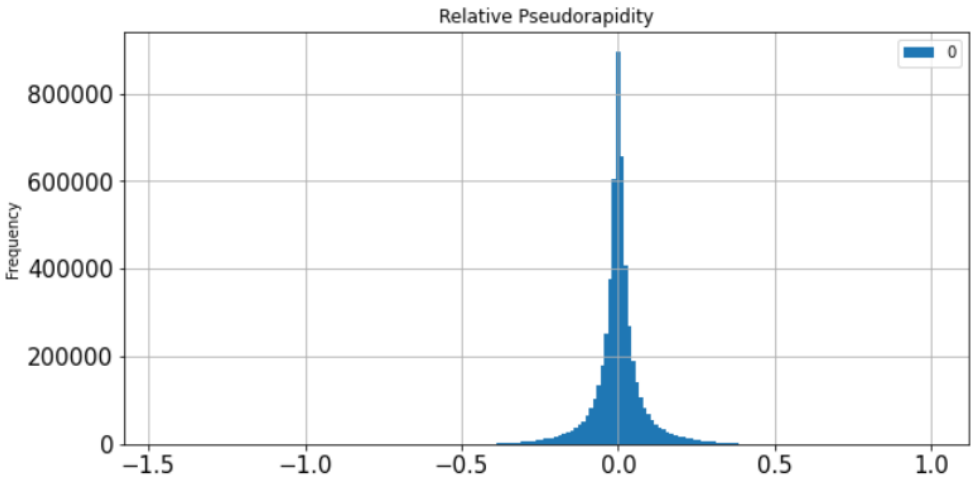

Figure 11: It seems a delta distribution around zero.

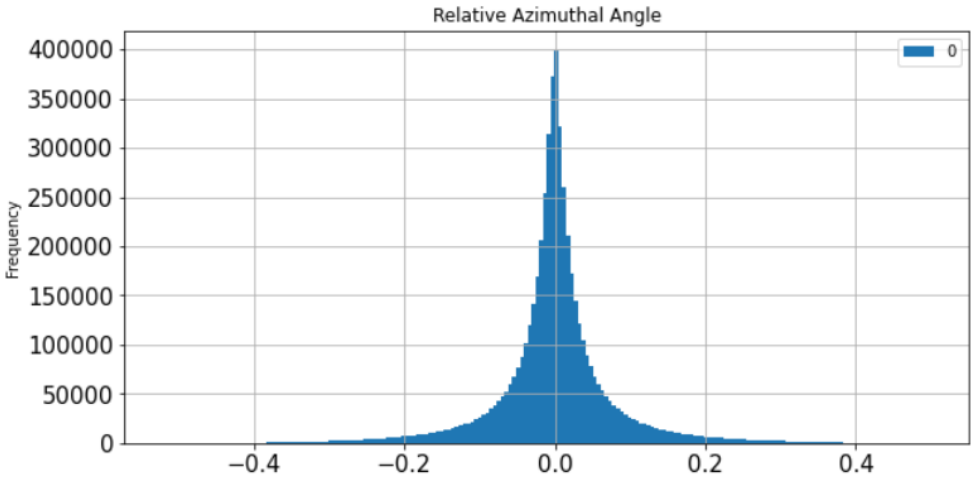

Figure 12: This histogram is with the azimuthal angle in range $(-\pi, \pi)$. It also seems a delta around zero.

Making jets and fat-jets with FastJet

The FastJet[11] is a software package that works through a plugin mechanism that can be associated with event generators like Pythia. It provides a broad range of jet finding and analysis tools for pp and e^+ e^- collisions, also providing means to facilitate the manipulation of jet substructure, including some common boosted heavy-object taggers, determination of jet areas and subtraction or suppression of noise in jets.[3] That is why it was so useful to associate this package to Pythia in the previous practice chapter.

In fact, the heavy-object tagger is a very interesting feature since fat-jets are referred as jets of heavy particles like Higgs boson, top quark, boson W or Z. So, if the program can defined fat-jets (and jets) precisely, we can tag this heavy objects to studied it into hadronic decays of boosted heavy particles, allowing us to investigate the nature of electroweak symmetry and phenomena beyond the the standard model, for instance. The study of jet definitions and its methods can be called "jetography".[9] Essentially, the approach is:

> "For example,looking for boosted top quarks a fat jet algorithm will try to distinguish between two splitting histories, where we mark [tag] the massive splittings from boosted top decays:

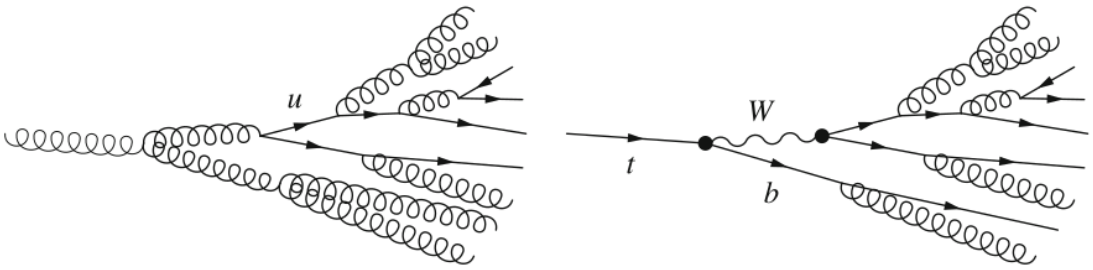

Figure 13: Image from [7], page 295. On the left, a top quark jet without its components identified. On the right, is how this top quark jet components should be identified by fat jet algorithm.

> The splittings inside the light–flavor jet ["common" jets] are predicted by the soft and collinear QCD structure. The splittings in the top decays differ because some of the particles involved have masses. This is the jet substructure pattern a fat jet algorithm looks for."[7]

This demonstrate how the jet substructure is so important to fat jet search. The biggest difficulty is to split an apparently one single big jet (like in Figure 9) into more than one jet, since heavy particles have very collimated decays due to their relativistic boost. Different codes were developed to identify clustering jets by the final states of particles decays detected by the detectors. Famous examples are the *kt* algorithm, the Cambridge/Aachen algorithm and the anti-*kt* algorithm. To this report were are going to focus on the anti-*kt* algorithm[2], because we are dealing with hard jets and it's a good strategy to extract different jets from inside a big cluster sequence defined by a large radius R of the base of the spray cone (forward discuss).

[11] `http://fastjet.fr/`

Practice: What are the differences between gluon jets and top quark jets?

Using FastJet and the hls4ml dataset, it was proposed to do histograms of both gluon jets and top quark jets perhaps to see the differences between jets recognized by FastJet. Therefore, the hls4ml dataset was given as input to FastJet separating each jet of the set and giving its particles components to be rearranged by it. Important detail is that particles with negative *mask* were not given to the software. All data input were relative variables input, so it is reasonable to deal η and ϕ as non relative variables since they are invariant under transverse boosts. On the other hand, relative pt input were multiplied by 1000 because we considered it is a fraction of total transverse momentum, being around 1000 GeV. Besides, the data don't provide the mass of the jet components, thus it was used the standard mass value of the program.

The jet definition used was determined by the anti-kt algorithm and R = 0.99. The base cone radius R was chosen to be big to include all given particles in the same spray cone, in this way it was avoided that any given particle would be excluded from the analysis. This exercise was done with histograms of the pt, η, ϕ and the jet invariant mass taken from the program after rearranging the given particles.

By this four histograms (Figures 15, 16, 17 and 14) we can see that the two types of jets have different fat jet structures by visualizing the FastJet variables output. It give us some perspective of what features this program looks for to identify boosted particles like top quark and gluon, using this patterns to tag the massive splittings.

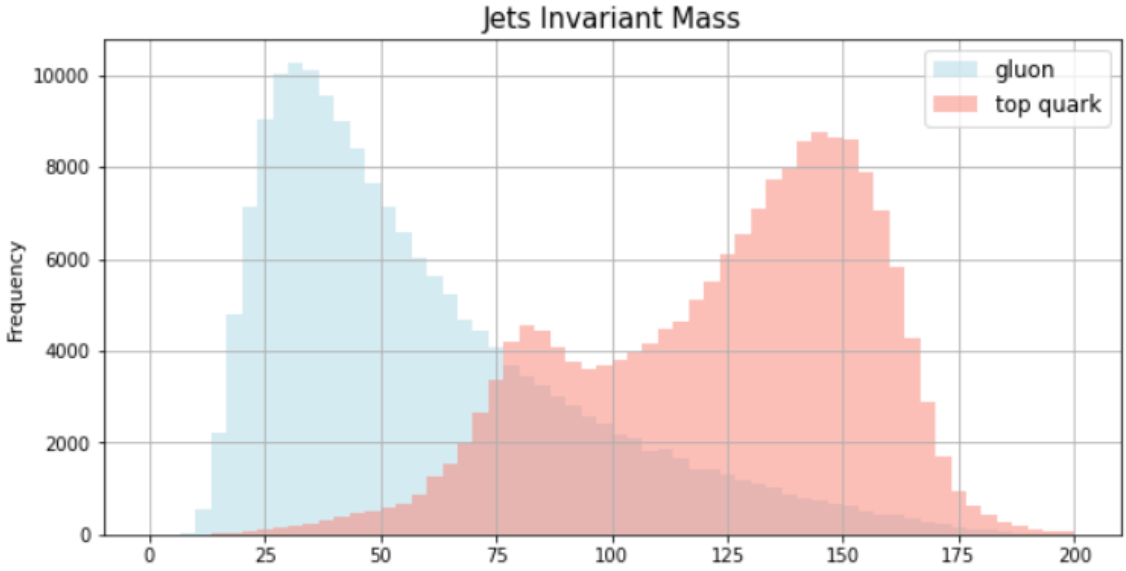

Figure 14: The invariant mass is the most interesting feature, since we can see that this jets are being separated by complete different mass conditions even we have not given any particle mass.

Simple fat-jet tagging: subjettiness

The previous practice have shows us interesting aspect of fat jet algoriths, how ever there are other method that could help us to separate better the substructures inside the spray cone and to tag the heavy particles. This method is called subjettiness.

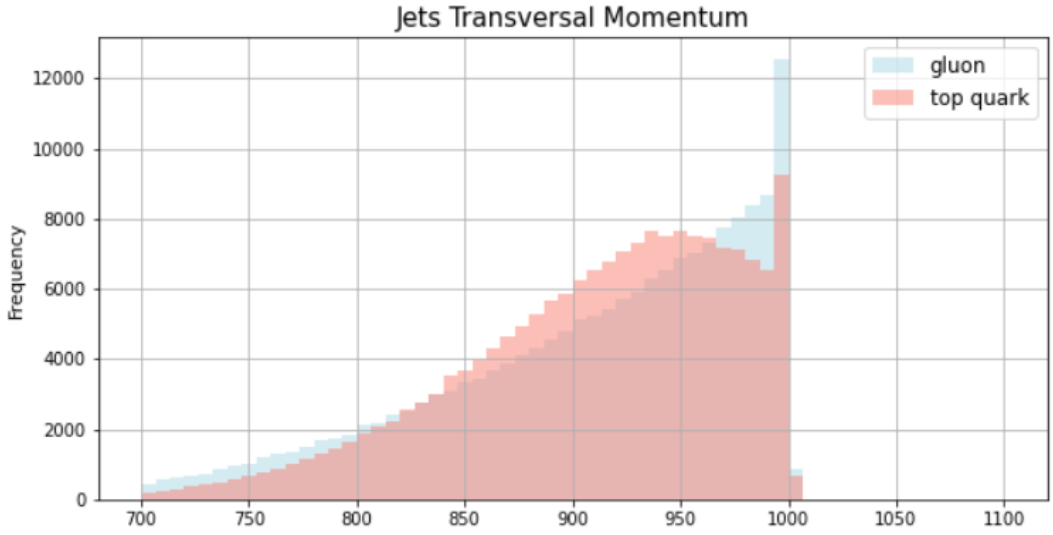

Figure 15: We see that there is a protuberance on top quark jets, which there isn't on gluon jets.

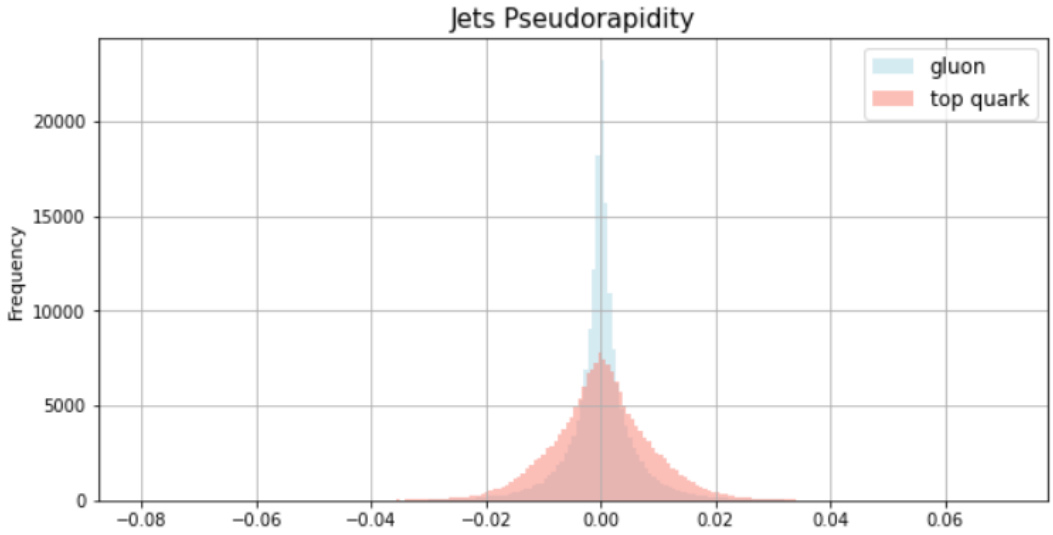

Figure 16: The top quark jets pseudorapidity seems more narrow than the gluon jet, but it has a wider base.

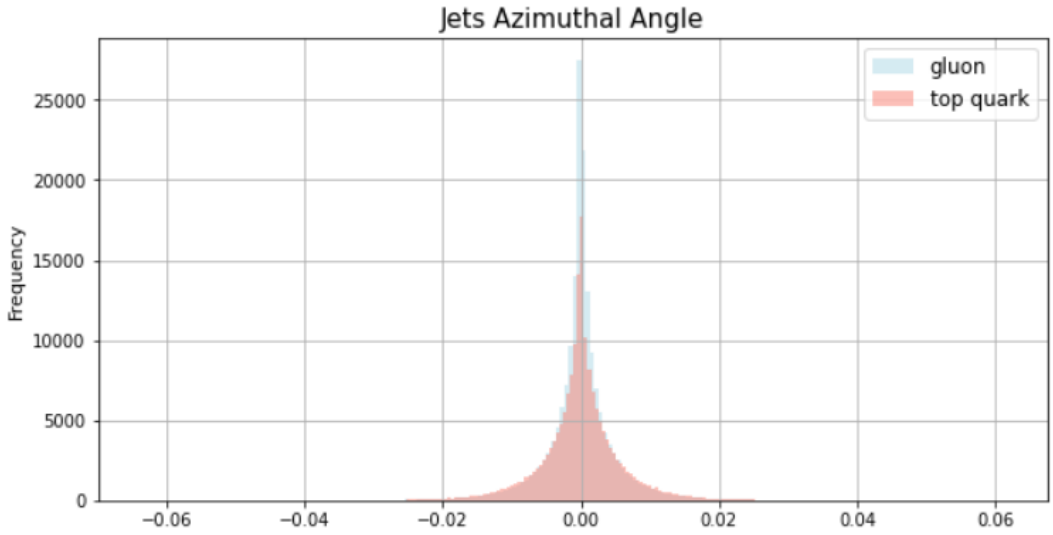

Figure 17: In the ϕ variable case, both looks like very similar distribution.

Briefly, the N-subjettiness tagger identify a highly boosted colour singlet particle decaying to 2 partons[3]. A better explanation example could be:

> "We start by defining an inclusive jet shape called “N-subjettiness” and denoted by τ_N. First, one reconstructs a candidate W jet using some jet algorithm. Then, one identifies N candidate subjets using a procedure to be specified [using FastJet, for example] [...]. With these candidate subjets in hand, τ_N is calculated via
>
> $$\tau_N = \frac{1}{d_0} \sum_k pt_{,k} \min(\Delta R_{1,k}, \Delta R_{2,k}, ..., \Delta R_{N,k}) \tag{6}$$
>
> Here, k runs over the constituent particles in a given jet, $pt_{,k}$ are their transverse momenta, and $\Delta R_{J,k} = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}$ is the distance in the rapidity-azimuth plane between a candidate subjet J and a constituent particle k. The normalization factor d_0 is taken as
>
> $$d_0 = \sum_k pt_{,k} R_0 \tag{7}$$
>
> where R_0 is the characteristic jet radius used in the original jet clustering algorithm. It is straightforward to see that N quantifies how N-subjetty a particular jet is, or in other words, to what degree it can be regarded as a jet composed of N subjets."[11]

In other words, this allows to define a cut to the number of subjets inside the spray cone, since depending on τ_N value we can determine if the cone spray cluster have exactly N or more than N subjets.

On this report, we propose that, instead of using this approximation models, the jets and fat jets definitions inside a spray cone cluster could be defined using the machine learning approach to tag and remark different features of hadronic jets.

Intro to Machine Learning: Neural Networks

Machine Learning (ML) technique are widely being used nowadays due its versatility and innovative way to approach problems that needs to deal with a large number of variables or large number of data. It’s based in algorithms that can adjust and improve mathematical models learnt from data that characterize the patterns, regularities, and relationships amongst variables in the system [12].

Although, ML can be can be defined in many different ways. Other form is to say:

> "Machine learning is essentially a form of applied statistics with increased emphasis on the use of computers to statistically estimate complicated functions and a decreased emphasis on proving confidence intervals around these functions." [4]

While the concept of a **learning** algorithm can be defined as:

[12] Michael Kagan in "Introduction to Machine Learning" lecture, on Open Lab Summer Student 2021.

> "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T, as measured by P, improves with experience E."[5]

It's said that the key element of this technique is the mathematical model dependence to its task[12]. In this report, we are going to concentrate on the classification task (T) of hadronic jets through the Neural Networks approach.

Classification task

This kind of task is based on determine thresholds to separate different classes of data. But these cuts must hold to other data sets of the same kind, and thus, to be a general feature to this type of data in such a way that avoids overfitting [12] (which is a model that exaggerates the fit compared to an "answer" function).

A well-known example of this task is the object recognition, where the input is an image (usually described as a set of pixel brightness values), and the output is a numeric code identifying the object in the image.[4] An first example and one of the most famous that we have first approached in was the digit recognizer problem based on the MNIST database. By this, the basic idea for the tagging process could be developed, which allows the computer to define an target object y depending on a finite set of labels (i.e. classes)[12].

Neural Network

The idea comes from neurons, which is based on the fact that it is connected with many others and each connection has a weight, defining which connection is more important or influential than another in order to recognize or identify something. In the case on Neural Network (NN), we have layers of nodes where each of the nodes in one layer is connected with all node in the next layer, defining a weighted net between the layers. The initial layer has the initial input and the last layer must contain the all possible answers of the computer based on the constructed neural network (output). The between layers are called "hidden layers", these is where we have the layers that answers some middle questions that defines the most probable characteristics of the input. (Figure 18[13])

The linear combination of all weighted connections of all the nodes of the previous layer respective to one node of the next layer (N_0) summed with its, so called, "bias" defines a value, which is inserted on a "activation" function (can be the logistic function, for example) determining how active or probable the node N_0 on layer h_n is, based on all its previous connections. The "bias" defines a threshold to N_0 be activated. By this, each of the nodes will have an "activation" value related to the characteristic that its layer should define about the given input.

Repeating this process in all layers will give an output answer that can be the not expected one. So, it's defined a cost function which will be the sum of all deviations between the "activation values" N_0 of the output and the expected value. The final propose of this "learning" method on

[13] https://wandb.ai/site/articles/fundamentals-of-neural-networks

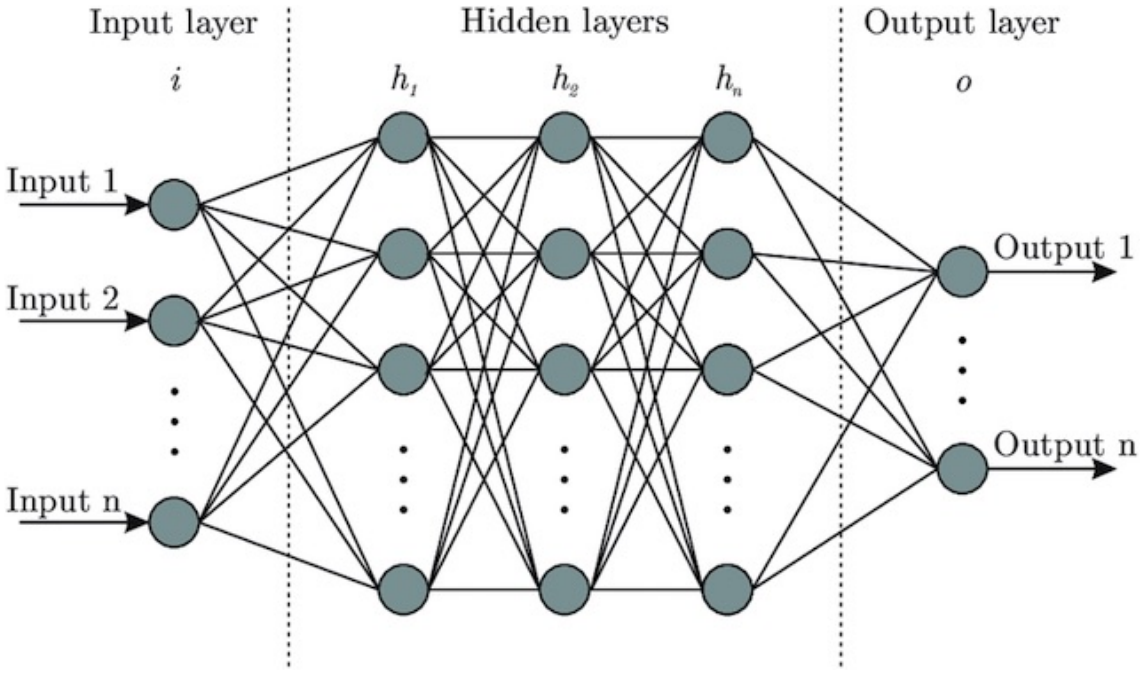

Figure 18: This is the most common Neural Networks structure. However, there are many different structures depending on the data kind input.

math means, it's to determine the minimum possible value of the cost function, which correspond to the most probable "right" answer. All this can be written in a matrix notation, which the negative gradient of the cost function determines the minimum point. Using this same logic, we can do the backwards process ("backpropagation") to define each one of the weights and bias of the nodes. After repeating this many times with some groups of data, that is "training" the network, we can get the better suitable values to weights and bias. This process is truly a stochastic process, which is defined by its randomness to reach the desired minimum point, in a statistic point of view.

One interesting proved fact is the **Universal Approximation theorem**, which the main state that we are going to use here is:

> "A neural network may also approximate any function mapping from any finite dimensional discrete space to another. [...] universal approximation theorems have also been proved for a wider class of activation functions [...]"[4]

Nevertheless, it's important to point that:

> "We are not guaranteed, however, that the training algorithm will be able to learn that function. Even if the MLP [multilayer perceptron] is able to represent the function, learning can fail for two different reasons. First, the optimization algorithm used for training may not be able to find the value of the parameters that corresponds to the desired function. Second, the training algorithm might choose the wrong function as a result of overfitting."

That is why we normally increase the complexity of the NN models adding more layers or activation functions to increase the possibility of a "universal approximation", what is called Deep Neural Networks, but it doesn't mean that the above problems disappear on this approach. It just tries to diminish them.

Practice: What is the standard deviation of a Jet cluster?

It was proposed to exercise the property of the **Universal Approximation theorem** through testing a Neural Network created to approximate the standard deviation function (std) of simulated gluon jets' relative pseudorapidity provided by hls4ml dataset [1].

Previously, we have seen that the relative pseudorapidity distribution of jets simulations (Pythia chapters) is described apparently by a delta distribution around zero. So, by this assumption, we can use this to check the NN result.

The NN was constructed using Keras [14] and TensorFLow [15]. The more effective and simple NN structure that provided good results was a basic three hidden layers structure ("dense layers" of 120, 32 and 1 units, respectively), which basically reduced an input jet of 30 particles ([1,30] size) into an output value (y_{output}) that needed to be near of the std real result (y_{actual}) of this jet. Besides, it was used the Relu activation function in all layers, the stochastic gradient descent (called "adam") and a custom loss function described by $((y_{output} - y_{actual})/y_{actual})^2$. Then, it was trained using 141802 jets with 10 epochs.

The result of the test was done on a group of 35450 jets. The below result show that the distribution of the NN is coherent with the expected result, since it seems a gaussian around zero. It's expected that the distribution is not necessarily a delta, since it is only an approximation to std function on a restrict group of jets. Maybe it could be refined the NN with a bigger training group and different layer constructions, but the simple NN described here have already presented a good enough result to the proposed exercise.

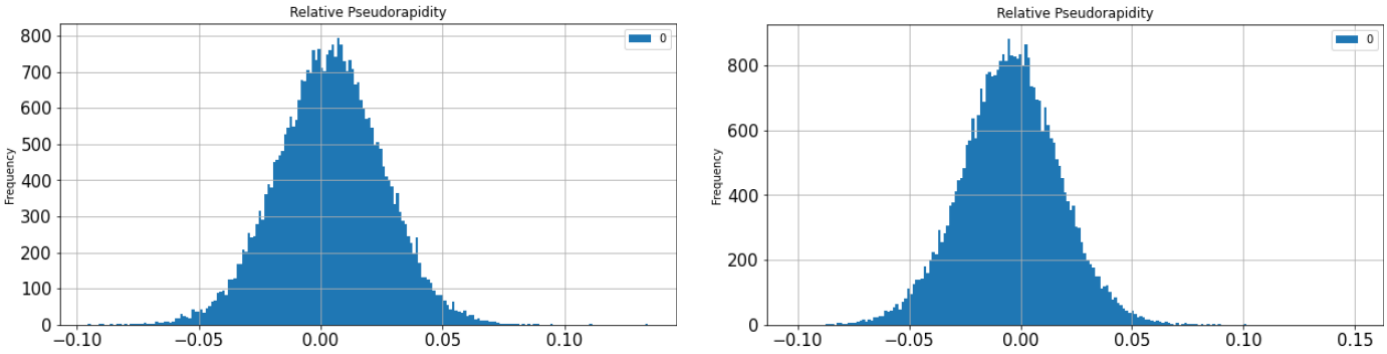

Figure 19: On left side is the first result histogram of the proposed neural network. For consistency, it was done a second run, which is the on the right side.

Fat-jet tagging with NN using GarNet layer

Finally, we discusse the use of a Neural Network to tag the features of fat jets in a way that it could identify from what particle that jet came from, depending on the given particles constituents. For this approach, we have used a special NN layer called GarNet[8] with the hls4ml dataset. It was designed to provide competitive performance on particle reconstruction tasks while dealing with data sparsity in an efficient way, exploring the use of graph networks and aiming to keep a trainable

[14] `https://keras.io/guides/`
[15] `https://www.tensorflow.org/api_docs`

space representation architectures at minimal computational costs. Because its architectures make no specific assumption on the structure of the underlying data, it can be employed for many other applications related to particle and event reconstructions, such as tracking and jet identification. That is why it was chosen to be used in this work, besides it is a very efficient approach for fast and accurate inference. A simplified explanation of how GarNet works:

- GarNet

  "In the case of the GarNet layer, the graph is built connecting each of the V vertices to a set of dim(S) aggregators. What is learned by S, in this case, is the distance between a vertex and each of the aggregators."

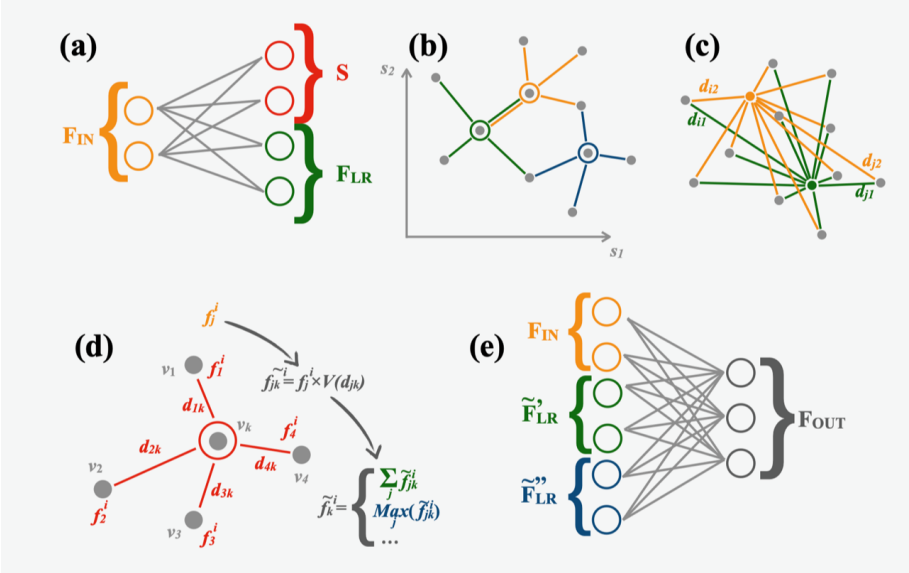

Figure 20: The (b) item it's about the GravNet layer, which is other approach NN layer discussed in [8] and it will not be use in this report.

"[Figure 20 is] Pictorial representation of the data flow across the GarNet and the GravNet layers. (a) The input features F_{IN} of each $v_i \epsilon V$ are processed by a dense neural network with two output arrays: a set of learned features F_{LR} and spatial information S in some learned representation space. [...] (c) In the case of the GarNet layer, the S quantities are interpreted as the distances between the vertices and a set of S aggregators in some abstract space. The graph is then built connecting each vi vertex to each a_j aggregator, and the S quantities are the d_{ij} euclidean distances. (d) Once the graph structure is established, the f_{ij} features of the v_j vertices connected to a given vertex or aggregator v_k are converted into the f_{ijk} quantities, through a potential (function of d_{jk}). The corresponding information is then gathered across the graph and turned into a new feature $\bar{f}_{ik}$ of v_k (e.g. summing over the edges, or taking the maximum). (e)For each choice of gathering function, a new set of features $\bar{f}_{ik} \epsilon F_{LR}$ is generated. The FLR vector is concatenated to the initial F_{IN} vector. The resulting feature vector is given as input to a dense neural network with tanh *activation, which returns the output representation F_{OUT}."*[8]

The Neural Network using GarNet was constructed using Keras and TensorFlow [16]. The input format is [x, n], which x is defined by a tensor of size (*vmax*, *info*). The *vmax* defines the number of particles of each jet and *info* defines the number of given variables per particle. The n is a tensor of size (1,) needed to give the correct input to GarNet, and thus keep graph structure information. The layers are composed by GarNet, Flatten and 3 Dense layers, respectively. The GarNet was used with three "sublayers" of 16 nodes each. The 3 Dense layes also used 16 nodes. All layers used the Relu activation function. The output was defined by a Dense layer with the number of nodes defined by the number of the option of jet classification, for example, if the given particle components were generated by a top quark jet or a gluon jet (as it will be seen). This output layer used the softmax activation function.

Moreover, the backpropagation process is done using the "categorical crossentropy" function as the loss function, and, the optimization also use "adam". This loss function computes the crossentropy loss between the labels and predictions.

To the learning process, it was given a input training of 236798 gluon and top quark jets shuffled with 30 (*vmax*) particles each. It contained the relative η, ϕ and of each particle, that is *info* equal 3. The input training is given by a tensor of ([*size train group*, *vmax*, *info*] , *vmax**[*size train group*,1]), which the second argument keep the graph connection weight to each particle of the jet. The predicted results were compared with a set of 236798 labels with the correct answer, that is if the jet was a gluon or top quark jet. Also, it was used a validation group of 59199 shuffled jets into the learning process, which verifies if the fitted model created using the train group fit the validation group too. The validation group had the same told features of the training group.

The test (prediction) process was done with 59198 shuffled jets (same input features). After this process, the given result was compared with its respective label set to recognize how many false and correct classification results were given. Below, we see the predict retult using the test group of this NN to classify jets between top quark jets and gluon jets, after 66 epochs.

In Figure 21, we see the comparing performances of the training and the validation datasets. From 0 to 5 epochs, we see that the training and validation performances are parallel lines that start to departure consistently. In 5 to 30 epochs, we see that the validation performances oscillates around the training performances, what might be a sign to stop training at an earlier epoch. Around 30 epochs we have a difference of loss between training and validation.

In Figure 22, we see the ROC curve and the AUC value of it jet type tagger. The Roc (“Receiver Operating Characteristic”) curve shows how much the model created by the NN is to distinguish the different classification options. It defines the "true positive rate" (sig. efficiency) and the "false positive rate" (bkg. mistag rate). The AUC value is the area below the ROC curves of each taggers, defining a value between 0,0 and 1,0. To facilitate analysis, it is showed in percentage. So, we can see that the AUC value to both tagger is high, close to 100%. It shows that the quality of the prediction model is very good to both taggers.

The result of this two taggers can be compared with the Jedi-NET results [6], Figure 23 in this report. We can see that ROC curve of the gluon and top quark taggers have similar contours with all example. The AUC value of the constructed NN is bigger than DNN, GRU and CNN models to gluon

[16] Git hub:`https://github.com/GasperJu/Jet-Classification-with-GarNet`

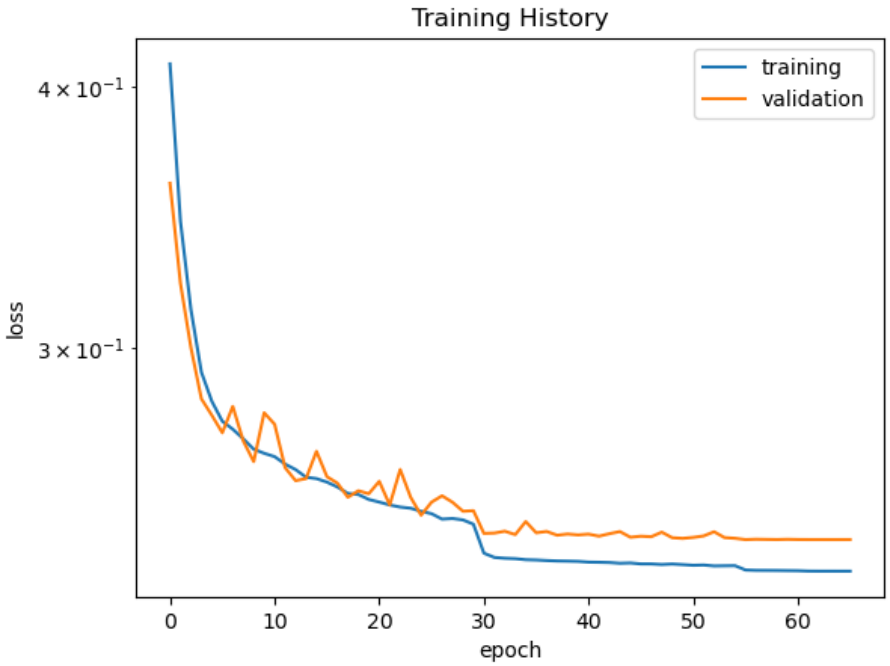

Figure 21: Plot of loss, comparing the performance of the train and the validation datasets.

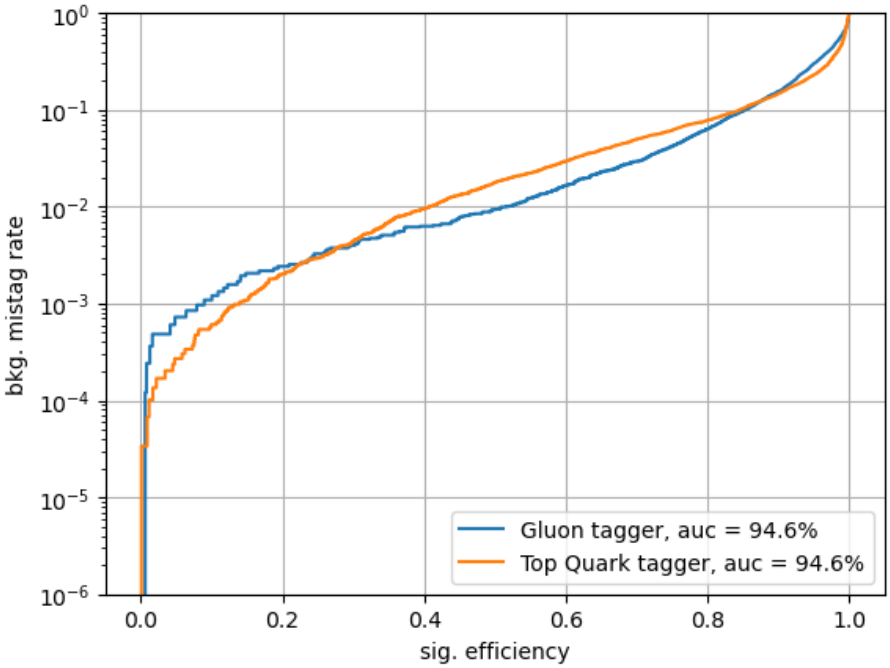

Figure 22: Plot of the ROC curve with its AUC value in percentage.

tagger. To top quark tagger our NN is better than DNN and GRU. The Jedi-NET gives the best AUC number to both taggers.

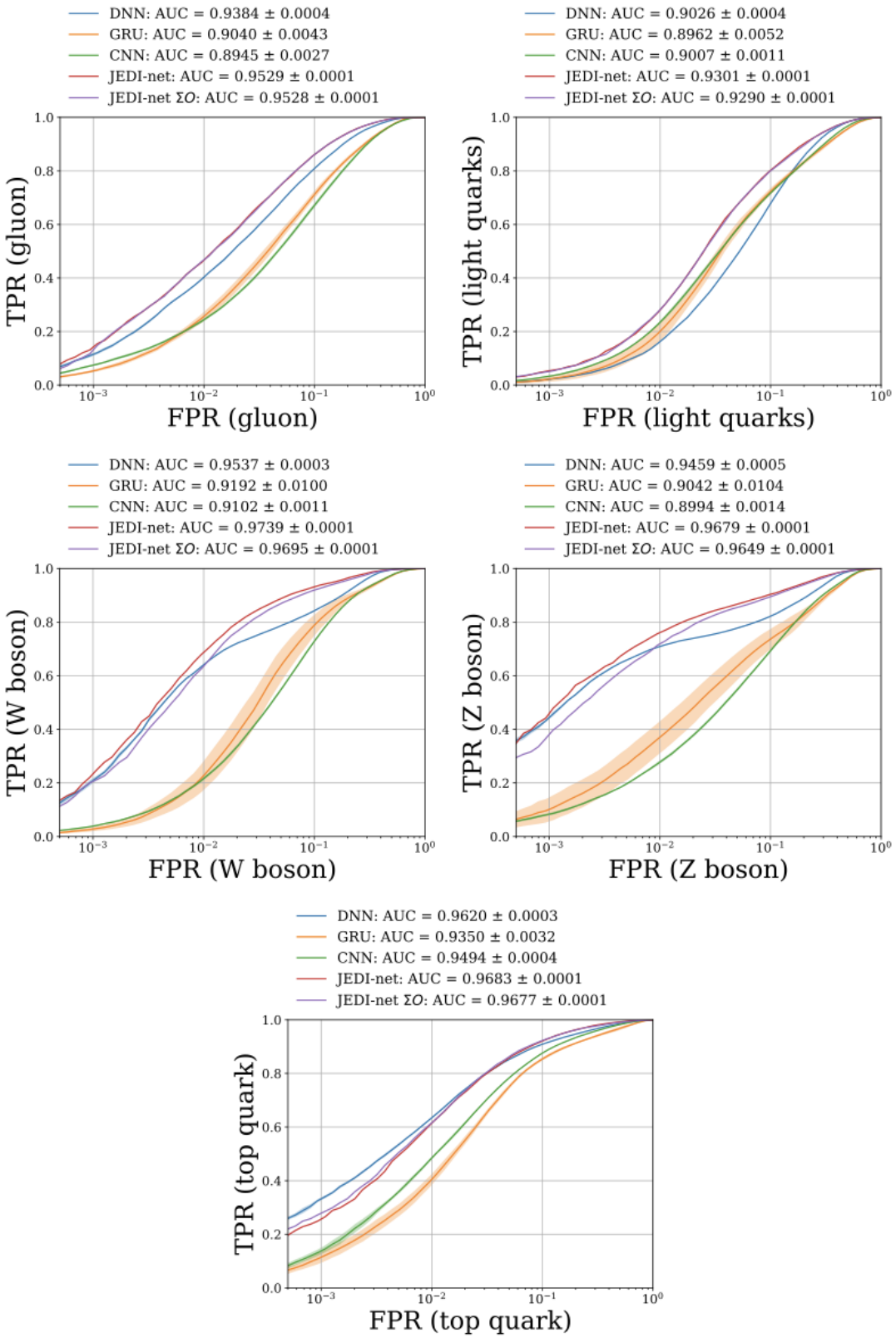

Figure 23: Figure 8 of the Jedi-Net paper[6], with the tagger results.

Conclusion

All steps developed on this Summer Student Internship were fundamental to understand the hole picture of this process and ensure the obtain knowledge of the final stage of this project.

The test using Neural Network developed for Jet classification between gluon jets and top quark jets can be easily modified to other types of tagging jets. It is interesting to do more test considering other situation to understand better the features of the GarNet layer associated to the hls4ml. Besides, it seems that minor number of epochs is better to fitting model. Since the AUC value is high to both tagger, try more taggers can be a good line to be study.

Acknowledgment

I appreciate all the help and time that was given to me, so I want to thank you Maurizio Pierini and Thiago Tomei for helping me on this journey. All the developed steps were fundamental to my knowledge and understanding. I have never thought that I could learning a beginning of Machine Learning and Neural Networks in so short time and fills that I comprehend much more than before the the internship.

Also, I want to thank you all colleagues of the Summer Student Internship, hls4ml group colleagues, professors of all lectures and all staff at CERN, specially Ana Dordevic.

Bibliography

[1] Anonymous. *JetNet*. Version 1.0. Zenodo, May 2021. DOI: 10.5281/zenodo.4834876. URL: https://doi.org/10.5281/zenodo.4834876.

[2] Matteo Cacciari, Gavin P Salam, and Gregory Soyez. "The anti-ktjet clustering algorithm". In: *Journal of High Energy Physics* 2008.04 (Apr. 2008), pp. 063–063. ISSN: 1029-8479. DOI: 10.1088/1126-6708/2008/04/063. URL: http://dx.doi.org/10.1088/1126-6708/2008/04/063.

[3] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. "FastJet user manual". In: *The European Physical Journal C* 72.3 (Mar. 2012). ISSN: 1434-6052. DOI: 10.1140/epjc/s10052-012-1896-2. URL: http://dx.doi.org/10.1140/epjc/s10052-012-1896-2.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016. ISBN: 0262035618.

[5] T.M. Mitchell. *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill, 1997. ISBN: 9780071154673. URL: https://books.google.com.br/books?id=EoYBngEACAAJ.

[6] Eric A. Moreno et al. "JEDI-net: a jet identification algorithm based on interaction networks". In: *The European Physical Journal C* 80.1 (Jan. 2020). ISSN: 1434-6052. DOI: 10.1140/epjc/s10052-020-7608-4. URL: http://dx.doi.org/10.1140/epjc/s10052-020-7608-4.

[7] Tilman Plehn. "Lectures on LHC Physics". In: *Lecture Notes in Physics* (2012). ISSN: 1616-6361. DOI: 10.1007/978-3-642-24040-9. URL: http://dx.doi.org/10.1007/978-3-642-24040-9.

[8] Shah Rukh Qasim et al. "Learning representations of irregular particle-detector geometry with distance-weighted graph networks". In: *The European Physical Journal C* 79.7 (July 2019). ISSN: 1434-6052. DOI: 10.1140/epjc/s10052-019-7113-9. URL: http://dx.doi.org/10.1140/epjc/s10052-019-7113-9.

[9] Gavin P. Salam. "Towards jetography". In: *The European Physical Journal C* 67.3-4 (May 2010), pp. 637–686. ISSN: 1434-6052. DOI: 10.1140/epjc/s10052-010-1314-6. URL: http://dx.doi.org/10.1140/epjc/s10052-010-1314-6.

[10] Torbjörn Sjöstrand et al. "An introduction to PYTHIA 8.2". In: *Computer Physics Communications* 191 (June 2015), pp. 159–177. ISSN: 0010-4655. DOI: 10.1016/j.cpc.2015.01.024. URL: http://dx.doi.org/10.1016/j.cpc.2015.01.024.

[11] Jesse Thaler and Ken Van Tilburg. “Identifying boosted objects with N-subjettiness”. In: *Journal of High Energy Physics* 2011.3 (Mar. 2011). ISSN: 1029-8479. DOI: 10.1007/jhep03(2011)015. URL: http://dx.doi.org/10.1007/JHEP03(2011)015.