Instituto Superior Técnico
Licenciatura Bolonha em Engenharia Física Tecnológica (LEFT)
- Projeto Integrador de 1º Ciclo em Engenharia Física Tecnológica (PIC1): "Scientific project"
Course responsible: Professor Ilídio Lopes
Examiners: Professor Ilídio Lopes and Professor Helena Santos

Deep Neural Networks applications in experimental physics data analysis

Systematic study of the importance of each of the low-level taggers used in ATLAS for b-jet tagging in Pb+Pb collisions using a deep neural network

Abstract

In this project, we set out to find which of the low-level taggers (*JetFitter*, *IP2D*, *IP3D*, *SV1*) utilized by the ATLAS (**A** **T**oroidal **L**HC **A**pparatu**S**) experiment in CERN had the greatest impact in the correct tagging of b-jets, in Pb+Pb collisions, when its outputs were directly fed to a deep feed-forward neural network (*DL1*). I found that, in general, although the differences between taggers were very subtle, *IP2D* and *IP3D* appear to have the most impact in the correct classification of the jets' flavours. Before feeding the Pb+Pb collision data to the DL1 network, sample preparation had to be realized, in which we performed down-sampling followed by scaling, so as to reduce eventual biases that could arise in the network's training. All of this was initially done for proton collision data (based on a tutorial from CERN), which I then generalized to the Pb+Pb case.

Sunday 19th June, 2022

Afonso Azenha - 96502

Contents

List of Figures

List of Tables

1 Introduction and context

One of the objectives of the heavy ion program of the LHC (**L**arge **H**adron **C**ollider) in Run 3 is the study of heavy flavour jets (collimated sprays of particles originating from the hadronization of charm and bottom quarks, together with gluons). Heavy flavour jets have proven to be excellent probes of the quark-gluon plasma (QGP), the primordial state of matter that existed in the Universe a tiny fraction of time after the Big Bang. The study of the quark-gluon plasma is fundamental to understand how our Universe was created. That's a lot of complicated words, so I'll begin unpacking them one by one. I've organized the rest of this section unconventionally in a sort of Q&A fashion, since I believe it makes it easier to follow.

1.1 Why study heavy-ions (Pb+Pb) instead of protons, like before?

First of all, CERN already has a long track record of experimenting with colliding protons (they started in 1971[5]!), so why did they suddenly decide to collide lead nuclei instead? (For reference, CERN began to accelerate heavy-ions in 1986[3].) The reason is that by colliding lead nuclei, we are able to obtain a volume (bigger than in a proton collision) with a very high energy density. In such a volume, this very high energy density allows for the formation of the aforementioned quark-gluon plasma.

1.2 The quark-gluon plasma (QGP)

The quark-gluon plasma is a "state of matter" in which the elementary particles that make up the hadrons[1] of baryonic matter[2] are freed of their strong attraction for one another under extremely high energy densities. These particles are the quarks and gluons[3], hence the name quark-gluon plasma. An early discovery about the quark-gluon plasma was that it behaves more like a perfect fluid with small viscosity than like a gas[4], as many researchers had expected, which I found curious. The characterization of the quark-gluon plasma is, thus, an important step for understanding the expansion of the Universe moments after the Big Bang occured.

1.3 What are b-jets and why should we want to tag them?

At this point, you might be wondering what are those b-jets I mentioned in the beginning. As a result of the collision between heavy-ions, jets are formed (dijets, in fact!). These are, as already mentioned before, collimated sprays of particles originating from the hadronization of quarks. Jets serve as probes. They are, by nature, strongly interacting, but moving so fast and with so much energy that they are often not completely absorbed by the surrounding quarks and gluons in the quark-gluon plasma. The degree of jet quenching (how much energy it loses to the

[1] Composite subatomic particles made of two or more quarks held together by the strong interaction.
[2] Baryons are hadrons with an odd number of valence quarks.
[3] Force carrier responsible for the strong interaction.

medium), plus the jet's orientation, directionality, composition, and how they transfer energy and momentum to the medium, allow us to infer the properties of the quark-gluon plasma. In other words, jets are our tool of choice that we use to study the QGP and its properties. They are classified as one of three types or flavours, depending on the flavour of the quark that originated it:

- **b-jets**, if they contain only b-hadrons (hadrons originating from a bottom quark);
- c-jets, if they contain c-hadrons, but not b-hadrons;
- light jets, if they don't contain neither b- nor c-hadrons (which means they come from one of the lighter quarks: up, down or strange).

Finally, why do we care so much about tagging specifically b-jets? In essence, b-tagging is utilised to study physics processes with b-jets in their final state. It just so happens that there's lots of interesting physics phenomena with this characteristic. Namely, b-tagging is important for (besides studies of the QGP):

- SM (**S**tandard **M**odel) Higgs sectors ($H \to b\bar{b}$, $HH \to b\bar{b}b\bar{b}$, ...);
- Top physics ($t \to Wb$);
- BSM (**B**eyond **S**tandard **Model**) searches ($X \to bY$).

Very recently, b-tagging was decisive in the observations of the Higgs boson decay into bottom quarks and of its production in association with a top-quark pair[14]. B-tagging has also been used to study CP (charge conjugation parity symmetry) violation[13]. With all that being said, I believe the importance of these jets cannot certainly be overstated.

1.4 How do we correctly tag these b-jets? What distinguishes them from the rest?

B-jets have a handful of characteristics that distinguish them from c- or light-jets. Namely, hadrons containing bottom quarks have sufficient lifetime that they travel some distance before decaying, leading to a (displaced) secondary vertex (see figure 1), which can be detected. In addition to this, the high mass of b-hadrons (few GeV) leads to decay products with a larger transverse momentum. This causes b-jets to be wider, have higher multiplicities (numbers of constituent particles) and invariant masses, and also to contain low-energy leptons with momentum perpendicular to the jet. All these features can be measured, and jets that have them are more likely to be b-jets.

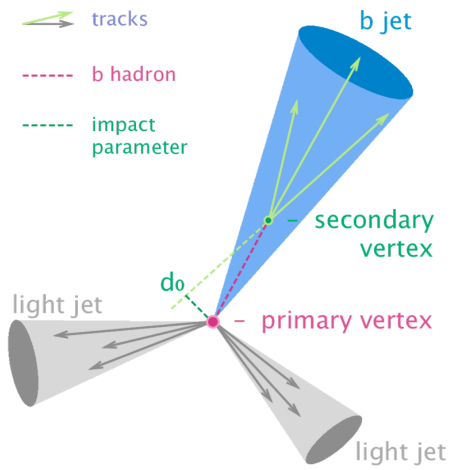


Figure 1: Diagram ilustrating the structure of a b-jet.

The actual tagging is first done by what we call the low-level taggers (LLTs). These were developped by the ATLAS Collaboration and can be divided into 3 major types, which I'll describe in the following subsubsections.

1.5 Low-level taggers

1.5.1 Algorithms based on impact parameters: IP2D and IP3D

There are two complementary impact parameter-based algorithms, IP2D and IP3D. The IP2D tagger makes use of the signed transverse impact parameter significance of tracks to construct a discriminating variable, whereas IP3D uses both the track signed transverse and the longitudinal impact parameter significance in a two-dimensional template to account for their correlation. Further down the line, this then allows us to compute the so-called log-likelihood ratio (LLR) discriminants[4], which are afterwards fed to the high-level taggers as input.

1.5.2 Secondary vertex finding algorithms: SV1

In short, SV1 runs iteratively on all tracks and attempts to reconstruct a single (displaced) secondary vertex. I won't be going into much detail on how this is done, as it goes way beyond the scope of my project (details in [14]). Nevertheless, SV1 outputs a bunch of variables, such as the vertex's mass and energy fraction, which then allow us to build LLR discriminants similar to the ones I mentioned for IP2(3)D.

1.5.3 Topological multi-vertex finding algorithms: JetFitter

Finally, and not going into much detail (once again refer to [14] for the full explanation), JetFitter attempts to reconstruct the whole b-hadron decay chain, outputting a similar set of variables as SV1, which are then used as inputs to high-level taggers.

1.6 High-level taggers

The output of these 3 types of low-level taggers already gives us a rough prediction of the jets' flavours. However, we can push this prediction's accuracy even further by utilising what we call the high-level taggers. These are more sophisticated algorithms, reliant on machine

[4]These LLR discriminants are calculated based on the per-track probability ratios for each jet-flavour hypothesis. Details on how this is done are in [14].

learning techniques, which receive the low-level taggers' outputs as inputs (alongside some other jet variables) and compute the probability of each jet-flavour hypothesis.

1.6.1 Machine learning: Deep feed-forward neural networks - DL1

In this work, I utilised an algorithm called DL1. In short, DL1 is based on a deep feed-forward neural network (NN) trained using Keras with the Theano backend and the Adam optimiser. The DL1 NN has a multidimensional output corresponding to the probabilities for a jet to be a b-jet, a c-jet or a light-flavour jet (refer to [14], section 4.3.2, for the specific NN architecture). Anyways, there's many variations of DL1: DL1r, DL1rmu and DL1d (see [8]). In my case, I am going to be training a slightly modified version of DL1. In doing many variations of its input layer, I'll attempt to find out which of the low-level taggers is the most important for the networks performance. In order to decide which jets are effectively classified as b-jets, a final discriminant is defined:

$$D_{\mathrm{DL1}} = \ln\left(\frac{p_b}{f_c \cdot p_c + (1-f_c)\cdot p_{\mathrm{light}}}\right), \tag{1}$$

where p_b, p_c, p_{light} and f_c[5] represent, respectively, the b-jet, c-jet and light-flavour jet probabilities (DL1's output), and the effective c-jet fraction in the background training sample. Utilising this discriminant, we will be able to assess the NN's performance under different conditions, but more on that later, once I present ROC curves and tagging efficiencies. Note that most of my work was based on a tutorial from CERN (see [2]), which was developed utilising proton collision data. As such, my job consisted, for the most part, in transforming this example, so as to make it work in the heavy-ion (Pb+Pb) case (which wasn't very easy, as the tutorial was fairly outdated!).

2 Computational work

I will now describe what I did in detail, from generating the original data files (which would then be used in the sample-preparation step) from the .h5 files provided by CERN for Pb+Pb collisions, to the training of the DL1 NN. Keep in mind that all of the code developed is in `python` (I also made some `bash` scripts to automate things).

2.1 Generating data files (before sample preparation)

This step offered lots of technical problems, provided I was working in CERN's SWAN[10] (**S**ervice for **W**eb based **AN**alysis), which is a platform to perform interactive data analysis in

[5] This f_c value can be tuned after the training of the DL1 NN so as to optimise the c- and light-jet rejection rates. This is something that I did not do in this analysis. I just maintained the value proposed in the initial tutorial: $f_c = 0.018$.

the cloud. As such, there were serious memory limitations that I had to go around in order to achieve reasonable results. In the end, I had to settle for a rather small number of jets (3 million in the training sample and 1.5[6] millions in the test/validation sample), which might impact the NN's performance negatively. However, since my goal is not to obtain the highest possible network accuracy, but to assess the low-level taggers' relative importances for the NN's overall performance, we can make do with this. In order to create the aforementioned training and testing samples, I utilised a modified version of the `create_hybrid.py` script[11] created by CERN. I was provided with data from Pb collisions divided in three folders. In each of these folders was collision data enriched in one of the three jet flavours, divided in 5 p_T intervals[7]. What I did was utilise the modified `create_hybrid.py` script (credit to Mariana Ribeiro who helped me with this) to select 1 million i-jets from the i-jet folder, for $i = b, c, u$, for the training sample, choosing only jets with even event number, which is a convention used in ATLAS. For testing/validation, jets with odd event number are used. The same procedure was done for the testing sample, but this time selecting only 500 000 of each jet flavour (using the odd flag!), for a total of 1.5 million. In hindsight, I realise that this wasn't the most brilliant idea. The testing sample is supposed to be something that we use to validate our model and to test its generalization capabilities. As such, it should be as close to real data as possible. And we know real data does not have a perfect $33.(3)\%$ split between the three jet flavours. If I had more time [8], I would rerun the whole experiment, but with a more realistic testing/validation sample. In order to be able to concatenate this many jets in a single file, without blowing up the terminal in SWAN, I also had to modify the `create_hybrid.py` script to allow for recursively opening the .h5 files (effectively storing 1 big vector (final data), instead of 2 (jet file data and final data), at any point in time. I read 50.000 jets at a time, iteratively). The reason being, some files had so many jets that they would fill all the available memory of the system, leading to the process being killed. Thankfully, I did have enough memory that I did not have to also save recursively. After all of this, we should have 4 files. Three for the training data-set, divided by jet type, and one for the test/validation data-set (all 3 flavours merged together).

2.2 Sample preparation

After this came the sample preparation step. For this, down-sampling followed by a scaling of the variables was performed. The down-sampling was done in order to avoid differences in the kinematic distributions of signal (b-jets) and background (c- and u-jets). That is, to make

[6] This number is smaller than the 3 million used for the training sample because, after the sample preparation step, we are left with around 1 million jets for training. I wanted to keep the ratio of jets for training and testing equal to 1, so there was no point in selecting more than around 1 million jets for testing. I did 1.5 just to be safe.

[7] I utilised only one of these intervals - JZ4. Initially, though, I hadn't done this and my p_T distributions were different for the different jet flavours, which then led to a lot of jets being lost in the sample preparation step! I've included images of the p_T distributions for the different jet flavours both before and after the down-sampling (in sample preparation) in the appendix.

[8] Running the script for selecting this many jets takes a really long time (hours!).

it so the training sample has the same distribution in the p_T, η[9] space for each of the three jet flavours, so our classifier is not biased to one of the three flavours, simply by virtue of it appearing more often in the training sample (in a certain (p_T, η) interval). Naturally, this cuts down the amount of jets in the training sample. Afterwards, a scaling of the jet variables was performed (and also of the track variables). For this, I had to develop my own scaling dictionary, adapting code from the Umami[7],[6] framework. This scaling was performed in order to center all the jet variables around 0 and to make it so their standard deviation was close to 1 (so I simply normalize them). This step is of the utmost importance for the successful training of the DL1 NN. If we hadn't done scaling, it could very well happen that one of the variables (features) dominates the loss function of the NN at the end of each epoch, simply due to it having a much broader range than any other feature. We do scaling to make sure that each of the features contributes approximately the same to this loss function, and so, to the evolution of the NN. Here I didn't have the same luck as in the subsection before, though: I had to save the train and test/validation files recursively in order to stay within the memory limit.

2.3 Training the DL1 neural network

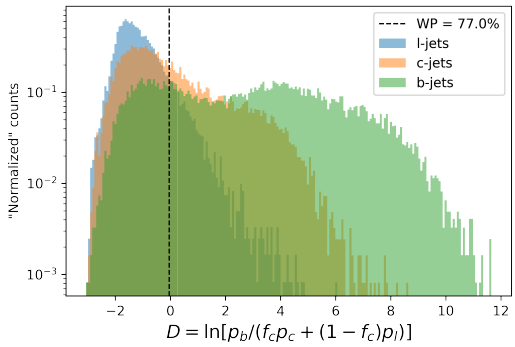
Now comes the fun part. I didn't change much from the tutorial's NN architecture: Input layer, followed by 8 hidden layers with 72, 57, 60, 48, 36, 24, 12, 6 neurons, respectively, with the 'ReLU' activation function and, finally, an output layer consisting of 3 neurons, with the 'softmax' activation function, which allows us to obtain probabilities as an output. I would also like to point out that the topology of the neural network output consists of a mixture of fully connected hidden and maxout layers, with batch normalisation added by default, and that the utilised loss function was the 'categorical cross-entropy'. The only things I changed were ADAM's learning rate as well as the batch size (I maintained the 130 epochs). Initially, I was training using CPU so it was rather slow, taking upwards of 25 seconds to train a single epoch, which led me to increase the learning rate from 0.01 to 0.03, so that the NN would converge faster[10]. The reason why I changed the batch size (from $2k$ to $10k$) was so I would obtain more 'stable' test accuracies (and losses). While training, I noticed the testing/validation loss was oscilating too much which led me to introduce this change. Now, as for my actual work, I trained the NN utilising various combinations of the low-level taggers (LLT) output variables as the inputs of my NN (in addition to the jet's p_T, η and $|\eta|$). I tested 8 combinations in total. Three of them in which I used only one of the 3 LLTs (I bundled IP2D and IP3D in the same category). Another three in which I removed only one of the 3 LLTs. And then two for control: one with all the LLTs and another with none (only the jet's p_T, η and $|\eta|$ were included by default, just like in all the cases above).

[9] This is the pseudorapidity: $\eta \equiv -\ln\left[\tan\left(\frac{\theta}{2}\right)\right]$, where θ is the angle between the particle's three-momentum $\mathbf{p}$ and the positive direction of the beam axis.

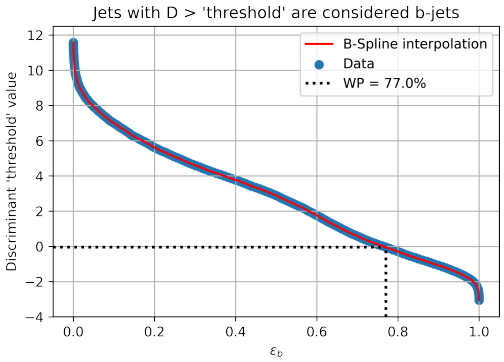
[10] Later on, I was able to train the NN using GPU, which significantly increased the training speed: from $\sim$20 s/epoch to $\sim$1 s/epoch! This meant that I no longer had to use 0.03 as the value for the learning rate, I could use a smaller step. Nonetheless, I mantained this value since it seemed to yield good results.

3 Results

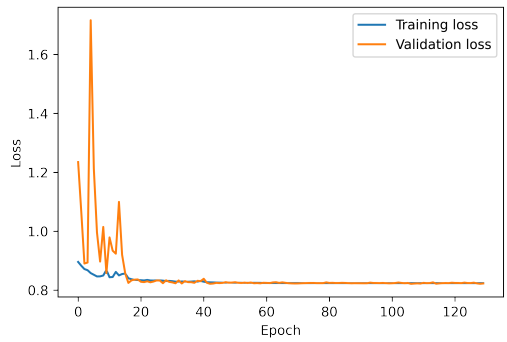
I am finally ready to present the obtained results! I generated lots and lots of plots for each combination, but they won't all fit in this report (due to limitations in the number of allowed pages). I will only present these plots for the case in which I used all the LLTs. I will, however, present the ROC (**R**eceiver **O**perating **C**haracteristic) curves comparing all 8 combinations mentioned before.



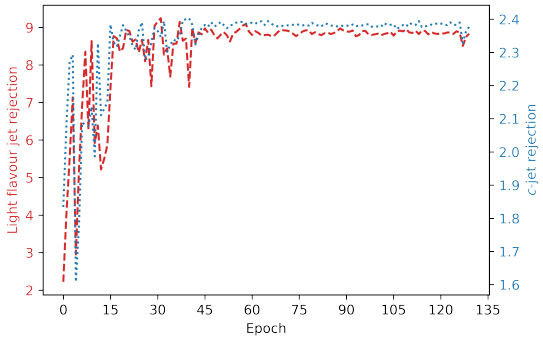
(a) Number of jets' distribution in the discriminant values.



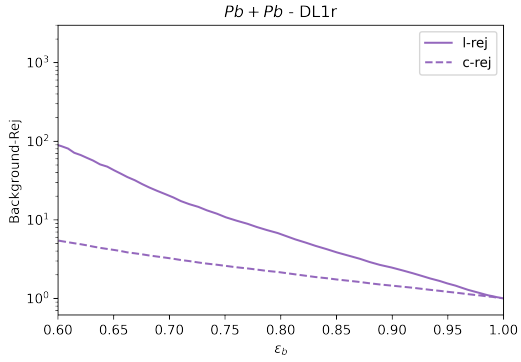
(b) Discriminant value as a function of ε_b.



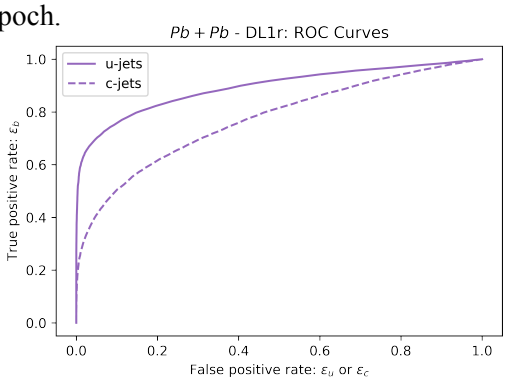
(c) Training and validation losses per epoch.



(d) Light- ($1/\varepsilon_u$) and c-jet rejections ($1/\varepsilon_c$) per epoch.



(e) Classifier's ROC Curve - ATLAS convention.



(f) Classifier's ROC Curve - usual convention.

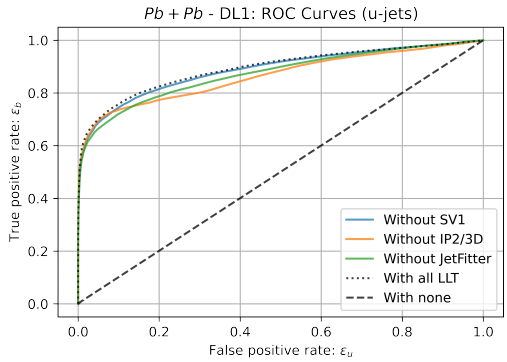
Figure 2: Graphs produced for the case in which I train the DL1 NN using all the LLTs.

Before anything else, I'd like to point out that I didn't use dropout in the end, as there was no clear evidence of overfitting, which can be seen in figure 2c: both the training and validation losses appear to converge to a single value. Before we interpret the rest of the results, I need to present some definitions: the variable ε_b represents the probability for a b-jet to be (correctly) tagged as a b-jet, whereas ε_c (ε_u) represents the probability for a c-jet (light-jet) to be (incorrectly) tagged as a b-jet. We also define the c-jet (u-jet) rejection as $1/\varepsilon_c$ ($1/\varepsilon_u$). With all that being said, for practical applications, we need to define the value of ε_b that we want to work with (the so-called working point (WP)). By choosing one value, we are compromising between our ability to correcly tag b-jets and effectively reject c- and light-jets. If we choose a WP that is too high, sure, we might correctly tag most b-jets as such, but we are also sure to incorrectly tag many c- and light-jets as b-jets! We must be mindful of this when choosing the WP to work at. In the figures above, I often represent the $WP = 77\%$ mark, since it is a common choice in ATLAS. Now, onto the rest of the graphs. After the definitions I've just presented, figure 2d is self-explanatory. Once again, we can see that the algorithm starts to stabilize after around 40 to 45 epochs (can also be seen in figure 2c, although not very clearly due to the superposition of the two graphs). Figure 2a represents the jets' distributions in the discriminant values (for each flavour). As can be seen, the b-jets' distribution is clearly skewed towards higher values of the discriminant (D), whereas the u-jets' is more focused on the left-hand side (smaller D values), with the c-jets' one being somewhere in the middle. This makes perfect sense, since we defined this discriminant so as to be the highest for the jets that the DL1 NN is most sure are b-jets. For $WP = 77\%$, jets with $D > -0.04007$ are classified as b-jets, so the jets on the right of the vertical line in figure 2a. If we look closely at this graph, we see that we are still mistagging many c-jets as b-jets (for this WP), which might be problematic. Depending on our goals, we might want to use a smaller WP, so as to guarantee greater c- and light-jet rejections[11]. Deciding on what WP to use can be done utilising the ROC curves[12] of graphs 2e and 2f. These ROC curves can be used to assess the classifier's quality. The closer the curve is to the top-left corner of the graph (for 2f only![13]), the better the classifier is. So, by looking at figure 2f, we can clearly see that our DL1 NN is much better at classifying (rejecting) light-jets than c-jets, which is in line with our previous realisation (from figure 2a) that we incorrectly tag way too many c-jets, but not so many light-jets (most are correctly rejected! - they are on the left of the vertical line). All that's left to do now is to repeat this analysis for all the 7 remaining combinations I mentioned before (so we can compare the results). I will present the results in a compact form, showing only the ROC curves and the area under these curves (AUC), which can be used as a quantitative parameter to gauge the classifier's quality: the greater the area, the better the classifier is (AUC = 1 would be the ideal classifier).

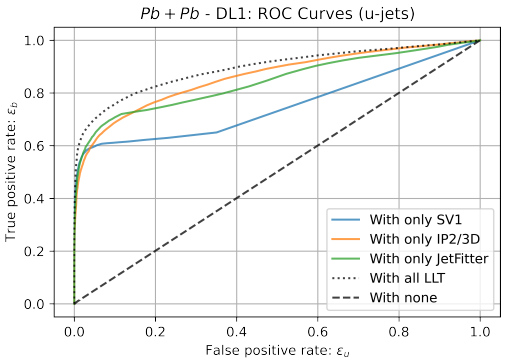
[11] We can also tune the value of f_c (*a posteriori*) to obtain greater rejection rates for c- and light-jets.

[12] Graphs of the true positive rate (also known as sensitivity) *vs* the false positive rate ($= 1 -$ specificity, if you will).

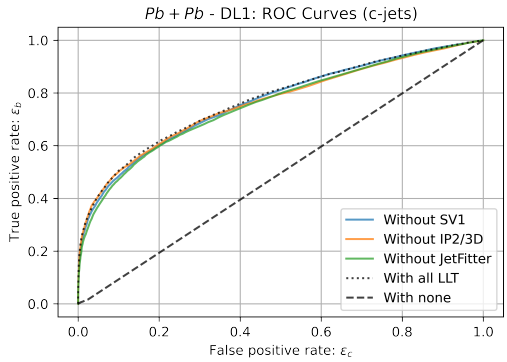
[13] For figure 2e, we use the top-right corner instead. I just find the usual ROC curves (of figure 2f) prettier and easier to compare against other ROC curves (utilising the area under the curve (AUC)).



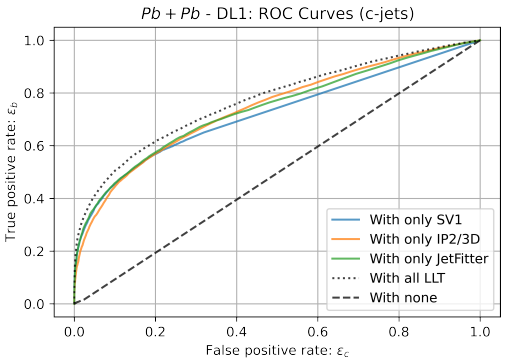
(a) ROC Curves for u-jets: cases without one of the three LLTs.



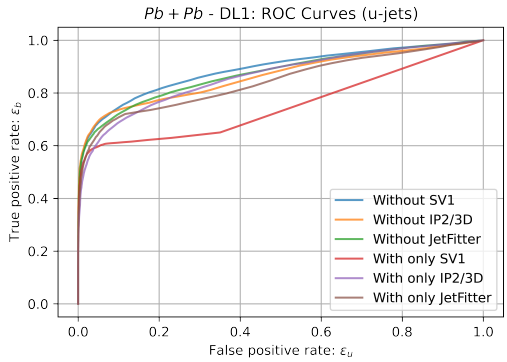
(b) ROC Curves for u-jets: cases with only one of the three LLTs.



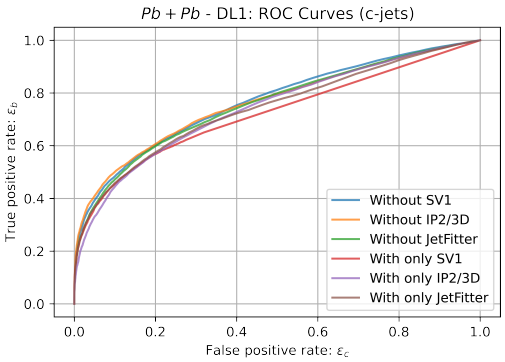
(c) ROC Curves for c-jets: cases without one of the three LLTs.



(d) ROC Curves for c-jets: cases with only one of the three LLTs.



(e) ROC Curves for u-jets: all cases, except 'All LLTs' and 'No LLTs' - 3a and 3b together.



(f) ROC Curves for c-jets: all cases, except 'All LLTs' and 'No LLTs' - 3c and 3d together.

Figure 3: ROC Curves obtained for the 8 mentioned combinations.

Let us now unpack all this information. The bottom line, though, is that IP2D and IP3D[14] appear to be the most important LLTs for the NN's performance, followed by JetFitter and, lastly, SV1. In order to understand this, we can look at the ROC curves presented above[15]. Note that there's two ROC curves for each of the 8 combinations: one for light- and another for c-jets! We'll start with the light-jets. In figure 3a, we can see that the ROC curve with the greatest area under the curve (AUC) is the one referring to 'Without SV1', which suggests this is the least important tagger (not in isolation!), given that the NN's performance is approximately the same with or without this LLT (compare with the dotted curve: 'With all LLT'). On the other hand, IP2/3D would be the most important, provided the NN's performance drops noticeably when these are not being used. This hypothesis (IP2/3D most important, SV1 least important) is also verified in figure 3b. Figure 3e is the combination of these two graphs we've just discussed. Now moving onto c-jets, focusing on figure 3c, there isn't much information we can take away from just looking at these curves, since they're almost completely superimposed. This means we have to look at the AUC values in table 1, in order to be able to infer anything of interest. Looking at this table, we can see that, from the three cases being discussed, 'Without SV1' has the greatest AUC value, but this time, 'Without JetFitter' shows the smallest (and not 'Without IP2(3)D'), suggesting that this tagger [JetFitter] is the most important for the rejection of c-jets, when paired with any other tagger (so, not in isolation!). Similarly, by looking at figure 3d, not much can be deduced. So, we need to check the table. By doing so, we can see that 'With only IP2(3)D' has the greatest AUC and 'With only SV1' has the smallest, which goes in line with the hypothesis we had formulated before (IP2/3D most important, SV1 least important). Figure 3f is the combination of these last two graphs we've just discussed. One thing I found curious was that the DL1 NN's performance for c-jets seemed to be better in the case where we used 'IP2(3)D Only' than in the case 'With all LLTs' (check AUC values in table 1)! I can't quite put my finger on as to why. What I can say, though, is that we must keep in mind that there's always some degree of randomness when training NNs. What I mean is that they won't always yield the exact same deterministic result every time you train them: there's some variance. So we must be able to know if the results that we're getting are consistent! For that I should have ideally trained the NN many times and checked the reproducibility of the results I just presented. Doing this would also let me know if the differences in AUC values are big enough to be considered meaningful. This is especially important provided that some of the conclusions I've drawn here were based on very tiny variations in the AUC values between curves. I would also like to point out the AUC values for the 'With no LLTs' case. These are very close to 0.5, which means this is a random classifier. So the NN cannot distinguish between jet flavours based solely on the p_T and η (and $|\eta|$) values. This makes sense, provided the down-sampling step we did! Overall, qualitatively, I believe we can say that IP2(3)D is the most important LLT, but not by much!

[14] I bundled IP2D and IP3D together for this analysis.

[15] I've also put in the 'Appendix' the ROC curves obtained utilising the ATLAS convention (same data, but different visualization), for completeness, as well as some extra ROC curves graphs.

	AUC	
	For u-jets	For c-jets
IP2(3)D Only	0.8617295673985709	0.7717993863355088
JetFitter Only	0.8432182356109175	0.7401102220083365
SV1 Only	0.7514398294916529	0.7202121200528215
Without IP2(3)D	0.864307842485108	0.7625360004157198
Without JetFitter	0.8731488688209466	0.7558781591150483
Without SV1	0.8876155403941507	0.7659239655139646
With all LLTs	0.8910500952460235	0.7711738714226132
With no LLTs	0.5008039407947135	0.49608459681932415

Table 1: AUC values for the 8 different combinations (for both u-jets and c-jets, relative to b-jets).

4 Conclusions and closing remarks

In conclusion, we've finally managed to achieve our goal of identifiyng the most important LLT for the DL1 NN's performance: IP2(3)D. Although the differences between taggers were very subtle, IP2(3)D seems to be the one that showed up on top the most. As such, we've managed to fulfil what we set out to do, which is great. In order to increase the confidence of this result, however, I would suggest re-training the NN many times for the same 8 conditions, so we could validate (or not!) our hypothesis. Another thing I could have done was separating IP2D from IP3D and evaluating each taggers' impact on the NN's performance individually, which could prove to be interesting. I would also like to point out that the trained NNs managed to achieve, at best, a 60% training accuracy[16] for overall tagging, which is not much! I believe some optimizations of the NN's architecture could also be done in order to obtain better accuracies. The parameters I used were (mostly) the ones optimized for the proton-proton case, which are, most likely, not the optimized parameters for the Pb+Pb case. This means there's (probably) still room for improvement! This work allows for a deeper comprehension of the LLTs influence in b-tagging when working with a deep neural network. This is useful for deciding which input parameters we want to choose for DL1. Had we found that one of the taggers was substantially less impactful in predicting the jets' flavours, we might have opted to not include it in the DL1 inputs, so as to have a less complex NN, which should allow for faster training times[17]. All of this justifies the work I've just presented, whose results I am fairly content with.

[16] This refers to the case in which all LLTs were used.

[17] Also, we wouldn't need to run this LLT's algorithm, since we wouldn't be using it, effectively saving even more time and effort.

Appendix

A Generating data files (before sample preparation):

A.1 p_T distributions before down-sampling (in sample preparation)

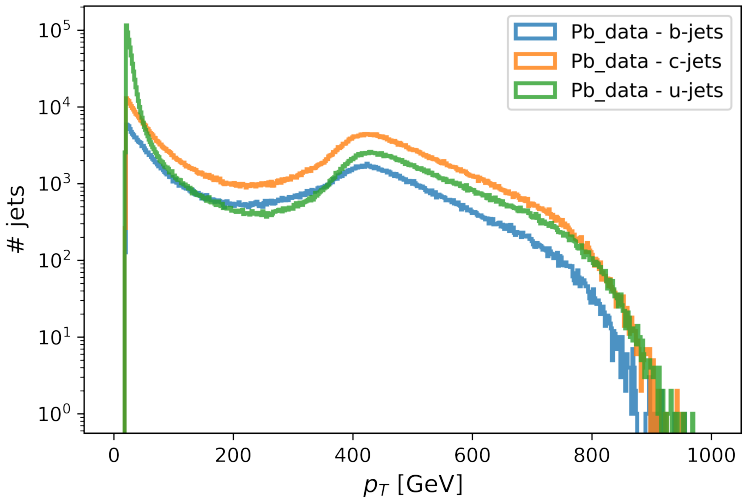


Figure 4: p_T distributions for the different jet flavours, before down-sampling (in sample preparation).

A.2 p_T distributions after down-sampling (in sample preparation)

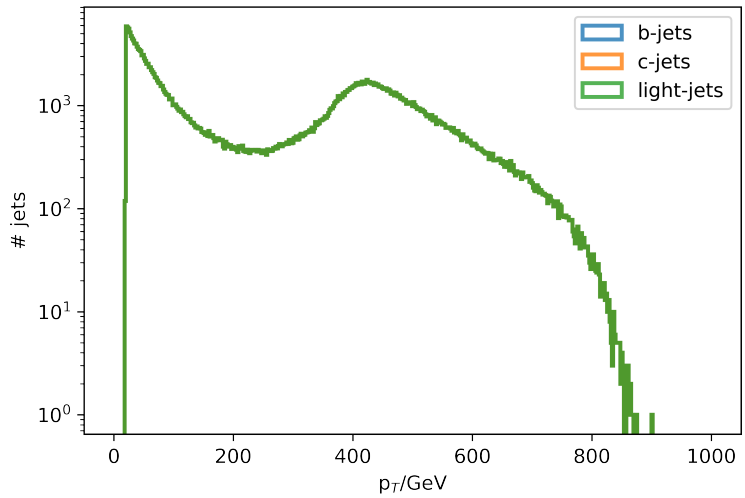
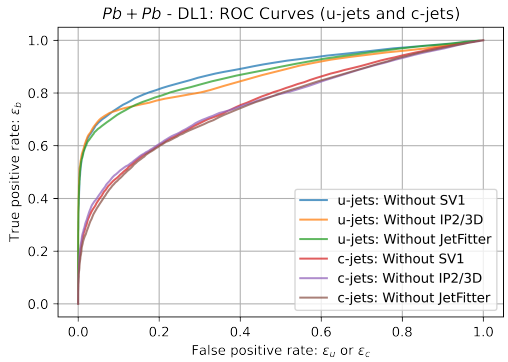


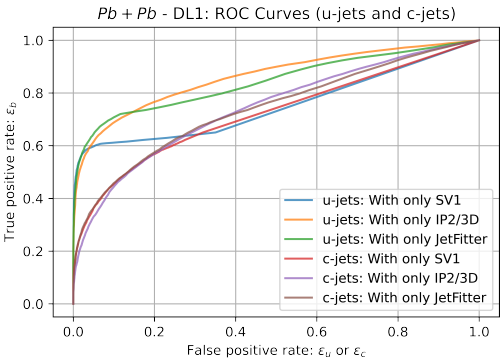
Figure 5: p_T distributions for the different jet flavours, after down-sampling (in sample preparation).

B More ROC Curves:

B.1 Using the usual convention:



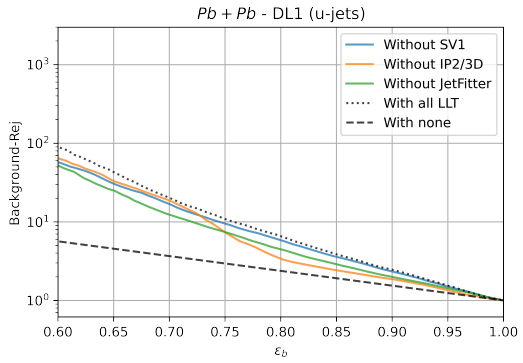
(a) ROC Curves for u-jets and c-jets: cases without one of the three LLTs - 3a and 3c together.



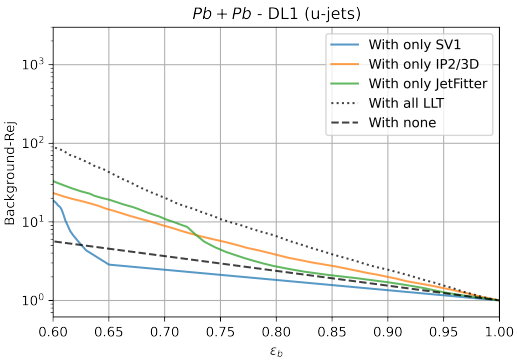
(b) ROC Curves for u-jets and c-jets: cases with only one of the three LLTs - 3b and 3d together.

Figure 6: ROC Curves obtained for the 8 mentioned combinations, except 'All LLTs' and 'No LLTs' (u- and c-jets together).

B.2 Using the ATLAS convention:

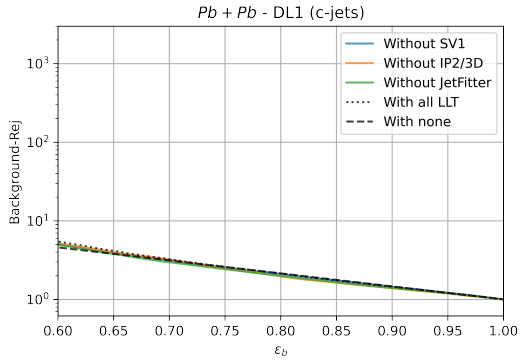


(a) ROC Curves (ATLAS convention) for u-jets: cases without one of the three LLTs.

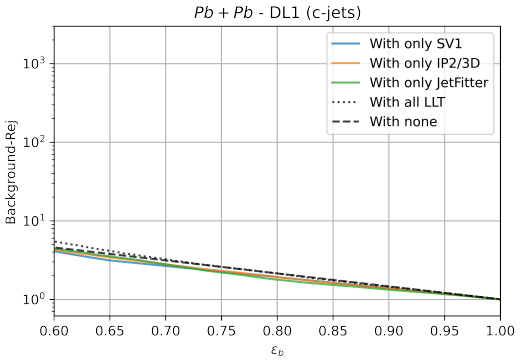


(b) ROC Curves (ATLAS convention) for u-jets: cases with only one of the three LLTs.

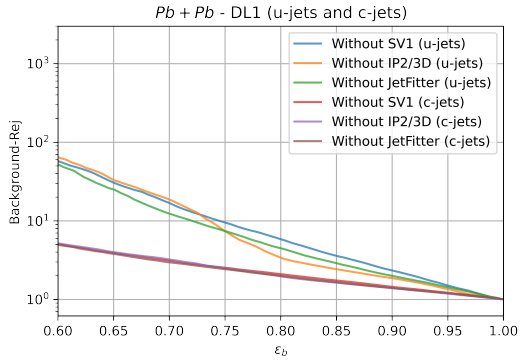
Figure 7: ROC Curves (ATLAS convention) obtained for the 8 mentioned combinations.



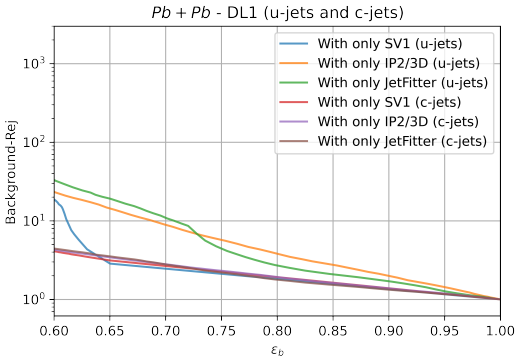
(a) ROC Curves (ATLAS convention) for c-jets: cases without one of the three LLTs.



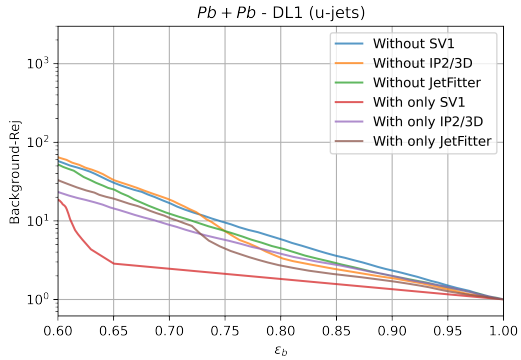
(b) ROC Curves (ATLAS convention) for c-jets: cases with only one of the three LLTs.



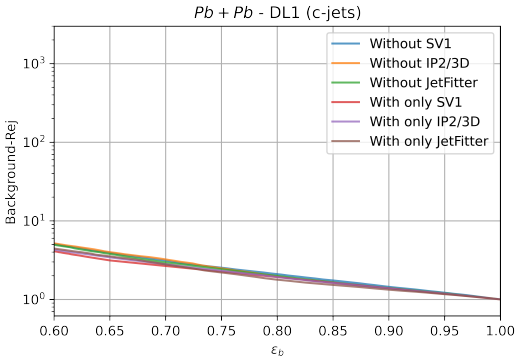
(c) ROC Curves (ATLAS convention) for u-jets and c-jets: cases without one of the three LLTs - 7a and 8a together.



(d) ROC Curves (ATLAS convention) for u-jets and c-jets: cases with only one of the three LLTs - 7b and 8b together.



(e) ROC Curves (ATLAS convention) for u-jets: all cases, except 'All LLTs' and 'No LLTs' - 7a and 7b together.



(f) ROC Curves (ATLAS convention) for c-jets: all cases, except 'All LLTs' and 'No LLTs' - 8a and 8b together.

Figure 8: ROC Curves (ATLAS convention) obtained for the 8 mentioned combinations (continuation).

References

[1] b-tagging. https://en.wikipedia.org/wiki/B-tagging. Accessed: 2022-06-19.

[2] b-tagging machine learning tutorial. https://gitlab.cern.ch/aml/tutorials/aml-workshop19-tutorials/-/tree/master/btagging-ml_tutorial. Accessed: 2022-06-20.

[3] Heavy-ion collisions begin. https://timeline.web.cern.ch/heavy-ion-collisions-begin. Accessed: 2022-06-19.

[4] Heavy ions and quark-gluon plasma. https://home.web.cern.ch/science/physics/heavy-ions-and-quark-gluon-plasma. Accessed: 2022-06-19.

[5] Today in 1971: First proton-proton collisions. https://home.cern/news/news/cern/today-1971-first-proton-proton-collisions. Accessed: 2022-06-19.

[6] Umami docs. https://umami.docs.cern.ch/. Accessed: 2022-06-20.

[7] Umami GiLab. https://gitlab.cern.ch/atlas-flavor-tagging-tools/algorithms/umami. Accessed: 2022-06-20.

[8] ATLAS FTAG algorithms documentation (DL1 section). https://ftag-docs.docs.cern.ch/algorithms/dl1/. Accessed: 2022-06-20.

[9] B-TAGGING: A REVIEW. https://indico.lip.pt/event/884/contributions/2660/attachments/2243/3083/tfpac_VladlenGaletsky.pdf. Accessed: 2022-06-19.

[10] The SWAN service. https://swan.web.cern.ch/swan/. Accessed: 2022-06-20.

[11] create_hybrid.py script. https://gitlab.cern.ch/dstarko/training-dataset-dumper/-/blob/master/create_hybrid.py. Accessed: 2022-06-20.

[12] Marie Lanfermann, on behalf of the ATLAS Collaboration. Deep Learning in Flavour Tagging at the ATLAS experiment. https://cds.cern.ch/record/2287551/files/ATL-PHYS-PROC-2017-191.pdf. Accessed: 2022-06-22.

[13] Olaf Steinkamp on behalf of the LHCb Collaboration. CP violation in b- and c-hadron decays at LHCb. *J. Phys.: Conf. Ser. 878 012013*, 2017. https://iopscience.iop.org/article/10.1088/1742-6596/878/1/012013/pdf.

[14] The ATLAS Collaboration. ATLAS b-jet identification performance and efficiency measurement with $t\bar{t}$ events in pp collisions at $\sqrt{s}$ = 13 tev. *Eur. Phys. J. C 79 (2019) 970*, 29th January 2020. arXiv:1907.05120.

[15] The ATLAS Collaboration. Performance of b-jet identification in the atlas experiment. *Submitted to: JINST*, 6th April 2016. arXiv:1512.01094.

[16] The ATLAS Collaboration et al 2008 JINST 3 S08003. The ATLAS Experiment at the CERN Large Hadron Collider. `https://iopscience.iop.org/article/10.1088/1748-0221/3/08/S08003/pdf`. Accessed: 2022-06-22.