


Using a Neural Network for displaced τ-jet tagging

Bjorn Kerby Dimayuga

Supervised by: Mykyta Schedrolosiev, Isabel Melzer-Pellman and Dirk Krucker

September 7, 2022

Abstract

In this report we explore different deep neural networks that can be exploited in the context of jet tagging problems. We specifically compare Space Neural Networks (SNN), a distance-weighted graph neural network (weighted GNN), and ParticleNet, the current state of the art for jet tagging. Their performances are evaluated in a beyond the Standard Model framework (BSM), where the super-symmetrical partner of the τ lepton, the *stau* $\tilde{\tau}$, is displaced from the primary vertex and decays into a pair of τ and charged neutralino $\tilde{\chi}_1^0$ or gravitino ($\tilde{\mathrm{G}}$). We discuss the tuning of the hyperparameters and the different trial procedures done to enhance the SNN's performance. The goal is to obtain the best performance of SNN with minimal computational costs and complexity by using floating point operations (FLOPs), trainable parameters and CPU/GPU time as benchmarks.

Contents

1. Introduction

1.1. SUSY

Supersymmetry is a speculative extension of the Standard Model of particle physics that contributes to the resolution of a number of unresolved issues related to our present knowledge of the basic principles governing the behavior of the universe. This extra proposed symmetry between fermions and bosons, which is the subject of an intense search effort at the Large hadron Collider, in its simplest form predicts the presence of a variety of new particles.
In searches for the direct production[1] of scalar tau leptons or the production of supersymmetric particles, e.g. tau slepton or SUSY tau $\tilde{\tau}$, thought to decay through scalar tau leptons (Fig. 1), significant progress has been made. Scalar tau lepton decays to their Standard Model counterparts produce difficult experimental signals, necessitating specialized reconstruction algorithms, which are frequently built using machine learning methods.

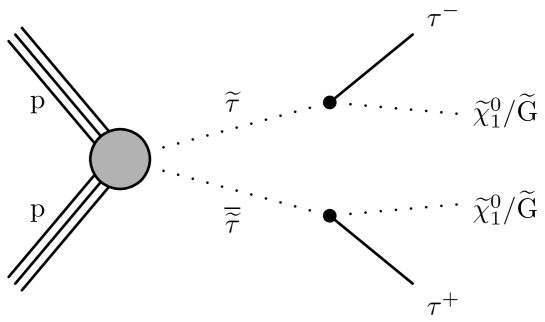


Figure 1: Decay of the SUSY tau $\tilde{\tau}$ to Standard Model tau and charged neutralino or gravitino

1.2. Jets and hadronic decays

In recent years, particle physics research has grown extremely active in the field of understanding the internal structure of hadronic jets. The Large hadron Collider collaborations are increasingly using jet substructure approaches in their experimental analysis as they look for both novel physics and Standard Model measurements.
In collider experiments, jets are collimated sprays of hadrons that are frequently connected to the generation of an elementary particle carrying a color charge, such as quarks and gluons. The strong force, which is quantized by quantum chromodynamics in the Standard Model of particle physics, controls how they evolve (QCD). The quark or gluon that starts a jet may emit more partons, creating a parton shower, or collimated shower of quarks and gluons, which eventually decay into the hadrons (π, K, p, n,...) seen in the detector. Jets are present in the great majority of LHC events (that one is interested

[1]For a more detailed summary of tau slepton production, see the Appendix

in). They are the most numerous and intricate objects measured by ATLAS and CMS, two multifunctional LHC experiments.

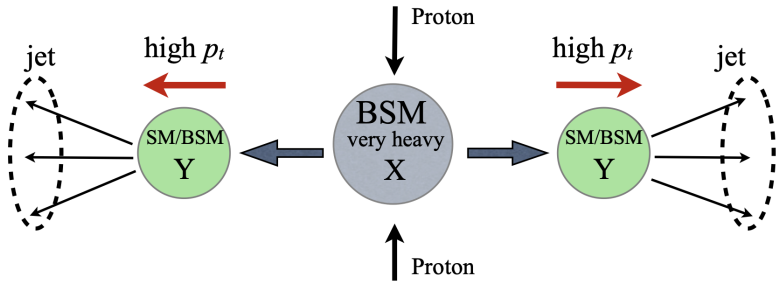


Figure 2: Generic interaction sequence for the search of a very BSM resonance that decays into electroweak-scale particles that subsequently decay hadronically.

Figure 2 depicts a typical circumstance of interest for BSM searches employing jet substructure: a proton-proton collision produces a hefty new resonance X with a mass of O(1) TeV. This heavy BSM resonance swiftly decays into lighter states Y, e.g. W/Z/H bosons or lighter BSM particles, with masses near the EW scale. Because their mass is substantially lower than the mass of the decaying particle X, particles Y often have a large transverse momentum (p_T). Because their mass is substantially lower than the mass of the decaying particle X, particles Y often have a large transverse momentum. Finally, if a particle Y decays hadronically due to its enormous boost, its decay product is collimated and reconstructed into a jet in the lab frame. The goal of jet substructure is therefore to identify a signal jet, which is generated by boosted heavy particles such as Y, from background jets, which are generally QCD jets generated by quarks and gluons. The majority of techniques for completing this classification task take a two-step approach: first, the jet is cleaned up (groomed), which involves removing soft radiation that is unlikely to originate from the decaying resonance. Next, one computes observables (e.g. jet-shape observables and prong-finders) specifically made to distinguish between signal and background jets based on the energy distribution among the remaining jet constituents. These observables aim to disentangle the different topologies that characterise signal and background jets [1].

1.3. CMS detector

A superconducting solenoid with a 6 m internal diameter and a 3.8 T magnetic field is the centerpiece of the CMS equipment shown in Figure 3. Within the solenoid volume, there are silicon pixel and strip trackers, lead tungstate crystal electromagnetic calorimeters, and brass and scintillator hadron calorimeters, each consisting of a barrel and two endcap sections. The pseudorapidity (η) coverage offered by the barrel and endcap detectors is expanded by forward calorimeters. Gas-ionization chambers built into the steel flux-return yoke outside the solenoid are used to detect muons. A two-tiered

trigger mechanism is used to choose events of interest. The first level (L1), which is made up of specialized hardware processors, selects events at a rate of about 100 kHz with a fixed latency of about 4 μs [2] using data from calorimeters and muon detectors. The second stage, referred to as the high-level trigger (HLT), lowers the event rate to about 1 kHz before data storage and consists of a farm of processors running a version of the entire event reconstruction software that has been tuned for quick processing [3].

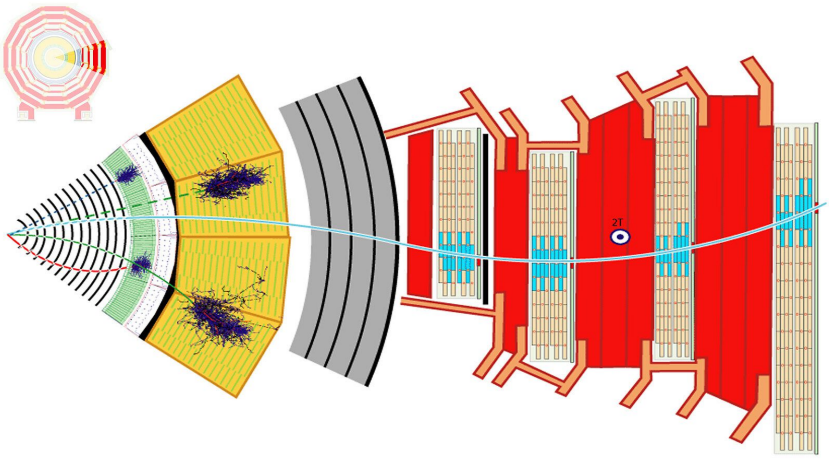


Figure 3: CMS detector cross section

1.4. Object reconstruction

A jet definition can be seen as made of a few essential building blocks: the jet algorithm, which is the recipe itself and a set of parameters associated with free knobs in the algorithm. A typical parameter, present in almost all jet definitions used in hadron colliders is the jet radius which essentially provides a distance in the rapidity-azimuth plane above which two particles are considered as no longer part of the same jet, i.e. no longer considered as collinear.

Over the past few decades, a number of jet algorithms have been proposed. They typically fall under two big categories: cone algorithms and sequential-recombination algorithms. Sequential recombination algorithms are based on the concept that, from a perturbative QCD viewpoint, jets are the product of successive parton branchings. These algorithms therefore try to invert this process by successively recombining two particles into one. This recombination is based on a distance measure that is small when the QCD branching process is kinematically enhanced. Thus, one successively recombine particles which minimise the distance in order to mimic the QCD dynamics of the parton shower. Most of the recombination algorithms used in the context of hadronic collisions belong to the family of the generalised-k_t algorithm.

Searches for new phenomena that consider signatures with τ leptons have gained great interest in proton-proton (pp) collisions at the CERN LHC. This particle is the most

massive among leptons ($\approx 1.8\,\text{GeV}$) and decays 35% of the time in leptons and 64% in hadrons, typically into either one or three charged mesons (predominantly $\pi^{\pm}$) plus up to two neutral pions. As the electrons and muons originating from τ decays are difficult to distinguish from those produced in the primary pp interaction, the algorithms for τ reconstruction and identification aim at hadronic tau decays (τ_h). Object reconstruction in the CMS detector starts from particle-flow (PF) algorithm [4] that combines data from the CMS subdetectors to recognize and reconstruct the charged hadrons, neutral hadrons, photons, muons, and electrons (pfCandidates) that emerge from proton-proton collisions. The missing transverse energy vector, the jets, the τ_h candidates, and the lepton isolation are all then recreated using these particles. hadronic decays of τ leptons in CMS are reconstructed and identified by the "Hadrons plus Strips" (HPS) algorithm [5], designed to reconstruct individual decay modes of the τ. The algorithm looks into the constituents of the jets to reconstruct the neutral pions that are present in most τ_h decays. The algorithm achieves an identification efficiency of 50 – 60% with a probability for quark and gluon jets, electrons, and muons to be misidentified as τ lepton between per cent and per mille levels. Standard CMS reconstruction/identification workflow works very poorly for the τ particles that are not coming from the primary vertex but from some displaced vertex in the CMS detector. Such τ particles can come from some long-lived SUSY particle (SUSY tau, $\tilde{\tau}$), for instance.
The use of PF objects allows for the reconstruction of jets. Using $R = 0.4$ as the distance parameter, the anti-k_T jet clustering technique [6] is employed with the application of typical jet energy corrections. A multivariate-based jet detection technique is used to eliminate jets produced by pileup collisions. This algorithm benefits from variations in the configuration of energy deposits in a jet cone.

2. Machine learning in Particle Physics

Machine Learning (ML) techniques have been an important ingredient in particle colliders for a quite a while. Event processing, particle identification and classification, energy tracking and direction measurements in calorimeters are just a few of the applications where these techniques are used. Boosted Decision Trees became state of the art and played a crucial role in the discovery of the Higgs boson by the ATLAS and CMS workgroup [7]. There are currently many studies to demonstrate the advantage and potential applications of Deep Learning (DL) algorithms on LHC's data taking and data processing workflows. Nevertheless, the actual numbers of DL models deployed in LHC experiments are quite low[2] and most of the studies are proof-of-concept demonstrations. One of these models is called *DeepTau* [8], which is a multiclass τ identification algorithm based on a convolutional deep neural network (DNN). In order to achieve an optimal tau identification performance, DeepTau combines information from the high-level reconstructed tau features together with the low level information from the inner tracker, calorimeters and muon sub-detectors using particle flow candidates, electrons and muons reconstructed within the tau isolation cone.

[2]Examples of DNNs for particle identification in CMS are B Tagger, Top Tagging, DeepAK8

The potential physics content of displaced τ particles decaying hadronically is substantial: by using DL architectures, one can possibly deploy an algorithm that has a very high accuracy and minimal computational costs. It can substitute the two way classification used in the current jet reconstruction algorithm. The current state of the art for jet tagging is based on particle clouds, called ParticleNet [9]. Deep residual learning [10], graph neural networks (GNNs) [11] and their distance weighted version [12] and Convolutional Neural Networks (CNNs) [13] can be combined to construct a powerful tool for the purpose of jet tagging displaced particles.

2.1. SNN

Space Neural Network, or SNN, is a DNN developed for τ particle reconstruction in CMS. It follows the same principles as GravNet/GarNet [12]: it receives as input a $B \times V \times F_{\text{IN}}$ dataset, which is fed to a few layers of convolution mechanism with which collective information from the vertex and its surroundings is gathered to produce the F_{OUT} output. This architecture is what we also call a dynamic GNN where the connections between the nodes and vertices of the graph $\mathcal{G}$ is not a fixed constant before the network is evaluated but rather a network the learns itself how to construct them in each layer. This SNN also implements the residual learning in its layers.

Base architecture of SNN The base architecture that is hypertuned is composed of $B = 500$ batch examples, $V = 50$ detector hits and $F_{\text{IN}} = 30$ features. The feature vector is represented by the vector

$$\vec{x}_i = \{x_1, x_2, \ldots, \Delta\eta, \Delta\phi\} \tag{1}$$

where η is the pseudorapidity and ϕ is the azimuthal angle.
The architecture is composed of:

- feature scaling layers: a scaling based on jet-momentum is applied to the features. The three methods used are
  1. **normal**
  $$x' = \text{clamp}\left(\frac{x - x_{mean}}{\sigma}, -5, 5\right) \tag{2}$$
  2. **linear/interval**:
  $$x' = \text{clamp}\left(\frac{x - x_{mean}}{x_{max} - x_{min}}, -1, 1\right) \tag{3}$$
  3. **categorical**

  To reduce the disbalance between the p_T and η of the jets, a non uniform bin reweighting was also implemented.

- SNN message passing mechanism (MPM): we index the batch number with $b = 1, \ldots, 500$, the number of detector hits with $i = 1, \ldots, 50$, the connections of a hit with itself and another hit as $j = 1, \ldots, 50$ and the features with k. With these indexing, the $k-$th feature vector of $i-$th hit connected with the $j-$th hit in the $b-$th batch is f^b_{ijk}, which is a 4th rank tensor. The SNN MPM acts on the feature vector $\mathbf{f}^b_{ij}$ to produce a new learned feature vector $\tilde{\mathbf{f}}^b_{ij}$. Mathematically, this is expressed by

$$\tilde{\mathbf{f}}^b_{ij} = \text{MLP}\left[\text{concatenate}\left(\mathbf{f}^b_{ij}, \bigoplus_{j=1}^{50} \boldsymbol{\mathcal{W}}^b_{ij} \odot \mathbf{f}^b_{ij} \odot \mathbf{M}^b_{ij}\right)\right] \tag{4}$$

where M^b_{ijk} is the masking vector referring to either the presence or absence of the $i-$th hit in $b-$th batch[3], $\mathcal{W}^b_{ijk} = \exp(-10 d^b_{ijk})$ is the weight tensor calculated from the distance tensor d^b_{ijk} between the $i-$th hit and all other j hits in the $b-$th batch[4], $\bigoplus_{j=1}^{50}$ is the chosen aggregation procedure, i.e. simple sum $\sum_{j=1, j\neq i}^{50}$, $\odot$ is the element-wise multiplication and MLP is simply the multilayer perceptron. This layer is represented in figure

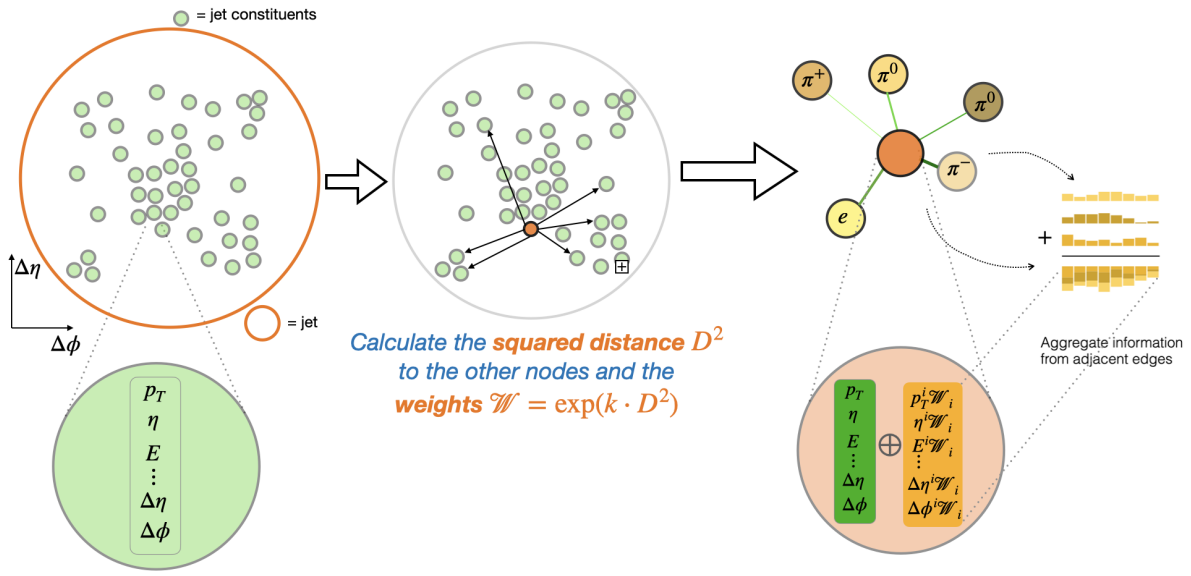


Figure 4: The message passing mechanism where the feature vectors inside the nodes are concatenated with the aggregated weighted feature vectors of all remaining nodes

- Wiring connections
- Batch Normalization layer, activation layer with ReLU and dropout layers with a rate of 0.1

[3]it is therefore the same value in the index j and k
[4]same value in the k index

- Aggregator layer
- Dense layers with batch norm, activation and dRopout layers
- Output layer with sigmoid activation

The overall learning rate of the architecture is 0.001.

3. SNN Tuning

3.1. Objectives and input space

The main objective of the neural networks we discuss here is to separate the jet τ_h from QCD jets. Our background data are composed of light quark and gluon jets from p_T-binned QCD samples that are combined without their cross-section weights; our signals are AK4 jets matched to τ_h from the LLSTau sample.
The selection criteria used for the background, signal and both of them is in Table 1. As for the input space, only jet daughters are taken (pfCand_jetDaughter==True).

Signal	**Background**	**S+B**
gen_tau_pt = 20	genJet_pt = 20	jet_pt $\geq$ 20
gen_z < 100	|genJet_eta| < 2.4	|jet_eta| < 2.4
gen_xy < 50	$dR < 0.4$	
$dR < 0.4$	no e, μ, τ_e, τ_μ	

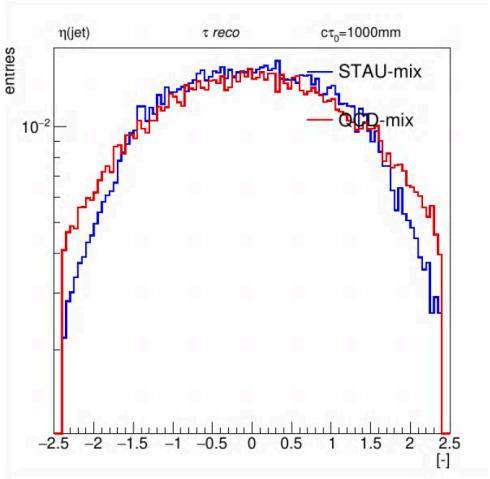
Table 1: Selection criteria for the signal and background jets

The pfCandidates are taken in a sequence with $n_{max} \leq 50$: $\{pf_1, pf_2, \ldots, pf_{n_{max}}\}$. If $n_{max} < 50$, the remaining empty $50 - n_{max}$ are masked by a vector. Every pfCandidate pf_n is a feature vector $\mathbf{x}_n = \{x_1, x_2, \ldots\} + \{\Delta\eta, \Delta\phi\}$. The comparison of the p_T and η of the background and signal is in Figure 5 while the decay distance of different $\tilde{\tau}$ masses are in Figure 6.
The final sample contains 13 million events evenly divided to signal and background, where 20% is preserved for testing. τ from STAU_M100_10cm and jets from TT-ToSemiLeptonic is used for evaluation of ROC curves. We have big amounts of data, so we observe the performance of the architecture for every modifications in just one epoch, and by looking at the batch AUC and loss trends.
The list of the feature vector and their scaling is in Table 3 in the Appendix.

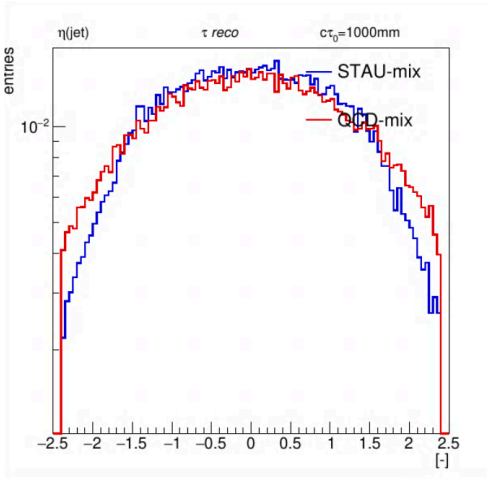
3.2. Hyperparameter tuning

Before the actual optimization of the hyperparameters, we studied the effects of:

1. changing the wiring layer after the batch normalisation and activation layers
2. eliminating the embedding layers of the categorical variables



(a) η distribution in log scale of signal (STAU) and background (QCD) jets



(b) p_T distribution in log scale of signal (STAU) and background (QCD) jets

Figure 5: Comparison between signal and background jets

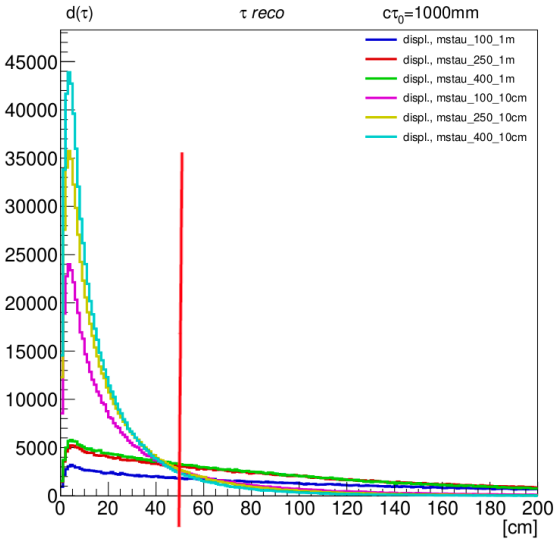


Figure 6: decay distance of different $\tilde{\tau}$ masses: only distances < 50 cm are considered

The results of these changes are: 1) doing the wiring after the normalisation and activation layers and 2) eliminating the embedding layers make the DNN perform worse. We also looked at the performance when the wiring periods are changed to $T_{\text{wiring}} = \{0, 1, 2, 3, 4, 5\}$. The results are in the Appendix, where we observe that $T_{\text{wiring}} = 3$ performs the best. Lowering or raising the T_{wiring} doesn't help the NN gain considerable performance enhancement.
The base architecture of $\mathcal{O}(160k)$ trainable paraemeters with which we compare the

results of the hyperparameter optimizations is presented in Figure 7

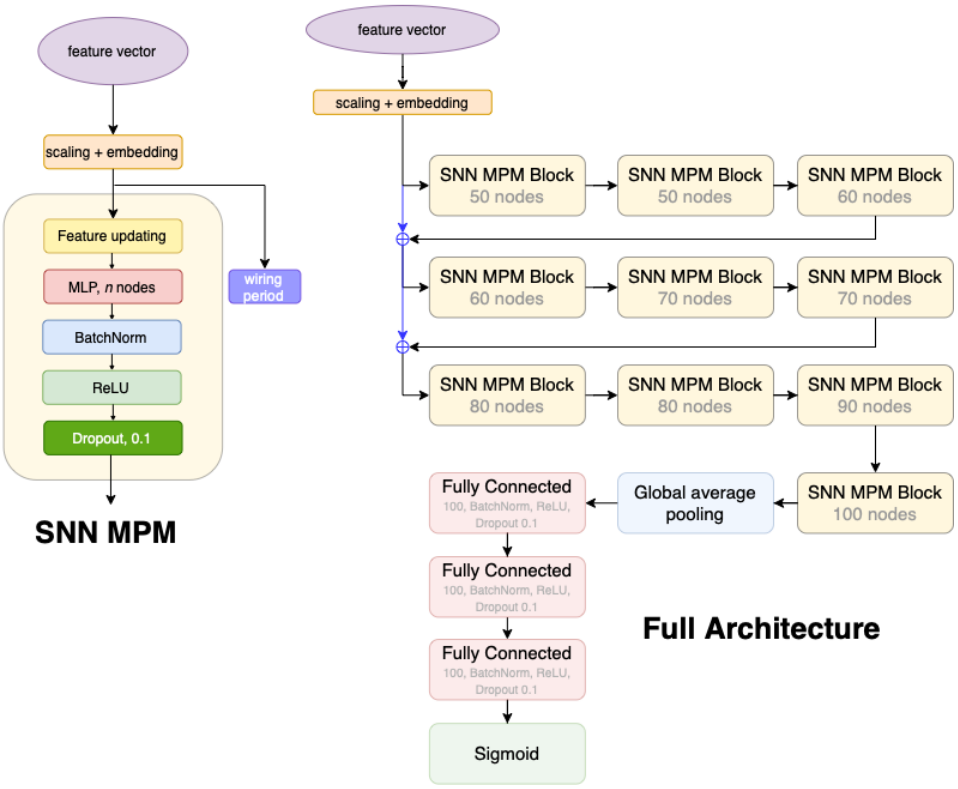


Figure 7: Base architecture of SNN with the message passing mechanism (MPM) implementation and the wiring connections (blue)

3.2.1. Embedding layer

I eliminate the embedding layers and add $n = 1, 2$ layers of 1×1 convolution layers with $nodes = 50$. Comparing it our base architecture, I observe a gain in performance, mostly due to the fact that it has more trainable parameters.

3.2.2. Feature updating layer

I study the performance of the architecture for different aggregation procedures $\bigoplus$:

- max pooling: $\bigoplus_j = \max_j(\dots)$

$$\bigoplus_{j=1}^{50} \boldsymbol{\mathcal{W}}_{ij}^b \odot \mathbf{f}_{ij}^b \odot \mathbf{M}_{ij}^b = \max_j \left(\boldsymbol{\mathcal{W}}_{ij}^b \odot \mathbf{f}_{ij}^b \odot \mathbf{M}_{ij}^b \right) \tag{5}$$

- mean pooling: $\bigoplus_j = \frac{\sum_j}{n_{max}}$

$$\bigoplus_{j=1}^{50} \boldsymbol{\mathcal{W}}_{ij}^b \odot \mathbf{f}_{ij}^b \odot \mathbf{M}_{ij}^b = \frac{\sum_{j=1,j\neq i}^{50} \boldsymbol{\mathcal{W}}_{ij}^b \odot \mathbf{f}_{ij}^b \odot \mathbf{M}_{ij}^b}{50} \tag{6}$$

- global sum: $\bigoplus_j = \sum_j$

$$\bigoplus_{j=1}^{50} \boldsymbol{\mathcal{W}}_{ij}^b \odot \mathbf{f}_{ij}^b \odot \mathbf{M}_{ij}^b = \sum_{j=1,j\neq i}^{50} \boldsymbol{\mathcal{W}}_{ij}^b \odot \mathbf{f}_{ij}^b \odot \mathbf{M}_{ij}^b \tag{7}$$

- Weighted average with the valid number of pfCandidates:

$$\bigoplus_{j=1}^{50} \boldsymbol{\mathcal{W}}_{ij}^b \odot \mathbf{f}_{ij}^b \odot \mathbf{M}_{ij}^b = \frac{\sum_{j=1,j\neq i}^{50} \boldsymbol{\mathcal{W}}_{ij}^b \odot \mathbf{f}_{ij}^b \odot \mathbf{M}_{ij}^b}{\sum_{j=1}^{50} M_{ij1}^b} \tag{8}$$

The results for these modifications indicate that the best aggregator is the weighted one. The complete weighted average, i.e. considering the weight tensor $\boldsymbol{\mathcal{W}}$

$$\bigoplus_{j=1}^{50} \boldsymbol{\mathcal{W}}_{ij}^b \odot \mathbf{f}_{ij}^b \odot \mathbf{M}_{ij}^b = \frac{\sum_{j=1,j\neq i}^{50} \boldsymbol{\mathcal{W}}_{ij}^b \odot \mathbf{f}_{ij}^b \odot \mathbf{M}_{ij}^b}{\sum_{j=1}^{50} \mathcal{W}_{ij1}^b M_{ij1}^b} \tag{9}$$

is not an optimal aggregating procedure. As the NN goes deeper in its SNN MPM layers, some of the nodes in the abstract learned space probably spread far away from the other nodes. This means that distances $D^2 \to \infty$ and the weights $\mathcal{W} \to 0$. In fact, we observe that the training stops abruptly because of the presence of *nan*s after a few batches.

Changing the scale of distance metric I looked at different scales ($k = 1, 10, 100$) of the exponential function $\exp(-k \cdot D^2)$ used in the weighting operation. The results tell us that the best scale of the distance between the nodes is $\mathcal{O}(10)$: with this constant we collect the right amount of information from nearby nodes without it being contaminated with noise and useless data contained in farther nodes.

Changing the dimensionality In the base architecture, distances between the nodes are always calculated in the 2D plane determined by the last two features in the feature vector, e.g. $\Delta\phi$ and $\Delta\eta$ in the first layer of SNN MPM block. I changed this distance calculation by considering only the last feature (1D case), the last three features (3D) and the last four (4D). The results of these modifications tell us that we can potentially gain performance enhancement if we calculate in higher dimensions. However, I observe an anomaly where the 2D case seems to be better than the 3D case, whereas I expected the result to be the other way. By repeating the experiments with the same configuration, we obtain the performances where it's visible that the possible gain in performance is largely hindered by the gradient descent variance and statistical fluctuations. This must be studied in dimensions higher than 4D to look if an enhancement could be achieved.

Performing dimensional wise aggregation Looking at the equation 4, we concatenate the weighted feature vector to the original feature vector before passing it to the MLP layer. This weighting is done in the 2D plane. I modified this operation by considering the dimensions singularly: instead of calculating the 2D distance, I calculated the 1D distance for both of the features. In this case, instead of having just one weighted vector, I have two of them that will be concatenated to the original feature vector

$$\text{concatenate}(\mathbf{f}_{ij}^{b}, \mathbf{f}_{ij,weighted_1}^{b}, \mathbf{f}_{ij,weighted_2}^{b}) \tag{10}$$

This operation was done to observe if we can gain performance enhancement by aggregating information in single dimensions. One could do this for all the features obtaining

$$\text{concatenate}(\mathbf{f}_{ij}^{b}, \mathbf{f}_{ij,weighted_1}^{b}, \ldots, \mathbf{f}_{ij,weighted_n}^{b}) \tag{11}$$

I studied this operation just for the 2D case and it performs worse than the base architecture.

Changing the potential function I also tried different distance metrics with varying properties. func1 $= 1-\sigma(10D^2-5)$ is approximately a constant function in the vicinity of the nodes and decays exponentially[5] after, while the other two functions, func2 $= 1 \cdot \sigma(10D^2)(1-\sigma(10D^2))$ and func3 $= \exp(-D^4/0.01)$, decay more rapidly and less rapidly to 0 than the base exponential function, respectively. The exponential distance metric with $k = 10$ is still the best metric to calculate the weights. Further hypertuning this constant k can be potentially beneficial for the SNN architecture. For now, the exponential function seems to capture the most information from the neighbouring nodes and eliminating the useless information from farther nodes.

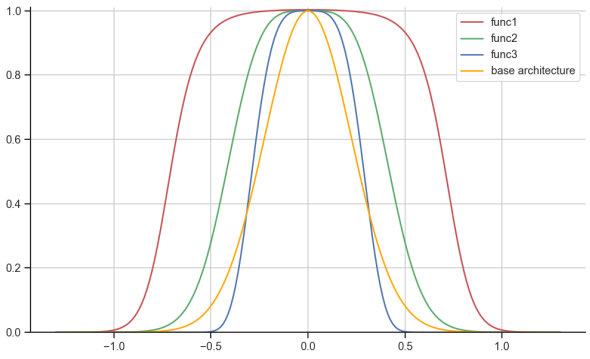


Figure 8: The three different distance metrics tested: the yellow is the exponential metric, the green one is a slower converging function, the blue one is faster converging to 0 and the red one is the constant function in some region

[5] $\sigma(x) = 1/(1+e^x)$ is the sigmoid function

3.2.3. Aggregator layer

We study the effects of changing the aggreation procedure after the SNN MPM blocks. We implement the same changes present in the feature updating layer, i.e max pooling, mean pooling, global sum and normalizing the sum by the number of valid pfCandidates. We found that global sum is the best aggregation procedure for this layer, followed by global average pooling. This indicates that we potentially gain from the architecture being sensitive to the number of valid pfCandidates.

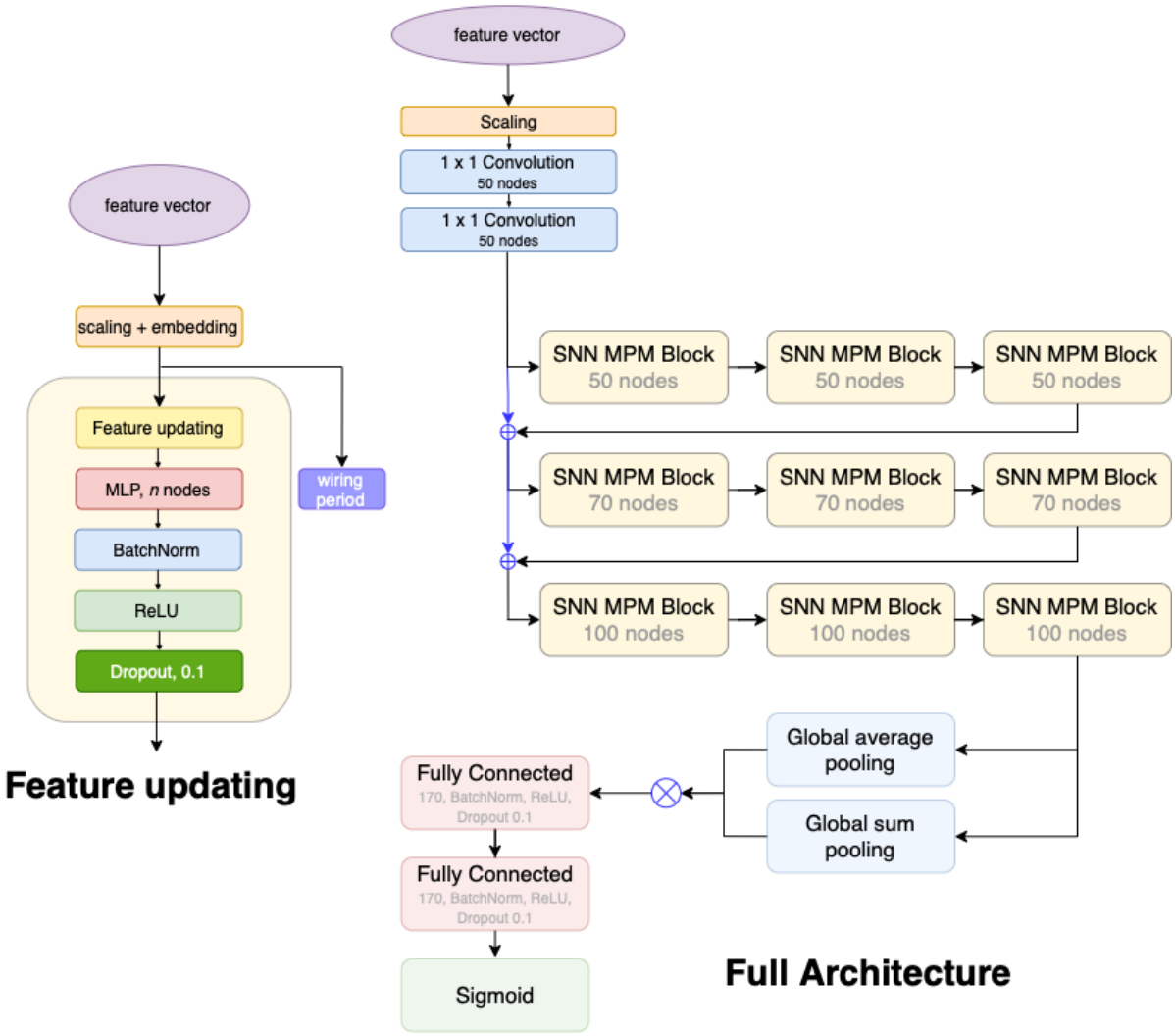


Figure 9: Full architecture of SNN after the hypertuning with the added 2 layers of 1×1 Convolution and two different aggregating procedures

3.3. Performance comparison

The final version of SNN (Figure 9) after the hypertuning has $\mathcal{O}(370k)$, which is comparable to ParticleNet's trainable parameters. We train both of this architecture for 5 epochs and we observe, through the number of floating point operations (FLOPs), that the SNN architecture is ten times lighter than the ParticleNet. During the SNN tuning, our background data are composed of light quark and gluon jets from p_T-binned QCD samples that are combined without their cross-section weights: they're summed up and fed to the neural network during the training. Similarly for the signal, τ_h jets from three different samples with different $\tilde{\tau}$ masses are combined. We evaluate the performance of our architecture by looking at the ROC-AUC per batch. However, we must take a further step to evaluate the performance of our NN. The kinematic properties of the training dataset are not physically realistic, and the AUC/ROC is not entirely representative of the performance in a physics analysis. In order to do so, we evaluate the AUC/ROC using signal jets from the $\tilde{\tau}_1(100)$ sample only, and background jets from a $t\bar{t}$ sample. We then use our architecture trained for 5 epochs to distinguish these jets and calculate the *jet misID probability* (the false positive rate) and *Tau ID efficiency* (the sensitivity, true positive rate) for the construction of the ROC curve. One other variable in which we can divide these ROC curves is the displacement between the decay vertex of the SM τ and the decay vertex of $\tilde{\tau}$, expressed by L_{rel}. If the DeepTau algorithm is also used after the HPS algorithm, the number of jets not from τ_h (NoHPS) must also be included in the denominator to be able to compare *HPS+DeepTau*'s and *SNN*'s or *ParticleNet*'s ROC curves. The HPS algorithm has a very limited tau signal efficiency that it doesn't even reach a sensitivity of 1 in the ROC curves, as can be seen in Figure 10 and Figure **??**.
We want a tagger that has low jet mis-identification probability for high tau identification efficiency. In Figure 10 and Figure **??**, a better architecture is nearer to the lower right part of the graph. These ROC curves are evaluated in different displacement regions to observe how our architectures perform. The visible results for our full SNN tagger are:

- competitive performance in comparison to DeepTau+HPS in low displacement regions ($0.2\,\text{cm} < L_{rel} < 1\,\text{cm}$)
- a decrease of mis-identification rate in comparison with the base SNN architecture, SNNlite,

misIDentification probability	**tauID efficiency**
$\approx 0.4 \times 10^{-3}$	60%
$\approx 1 \times 10^{-3}$	70%
$\approx 0.5 \times 10^{-2}$	60%
$\approx 0.9 \times 10^{-2}$	90%

Table 2: misID rate for different tauID efficiency of the full SNN architecture

- more visible performance gains in low displacement regions

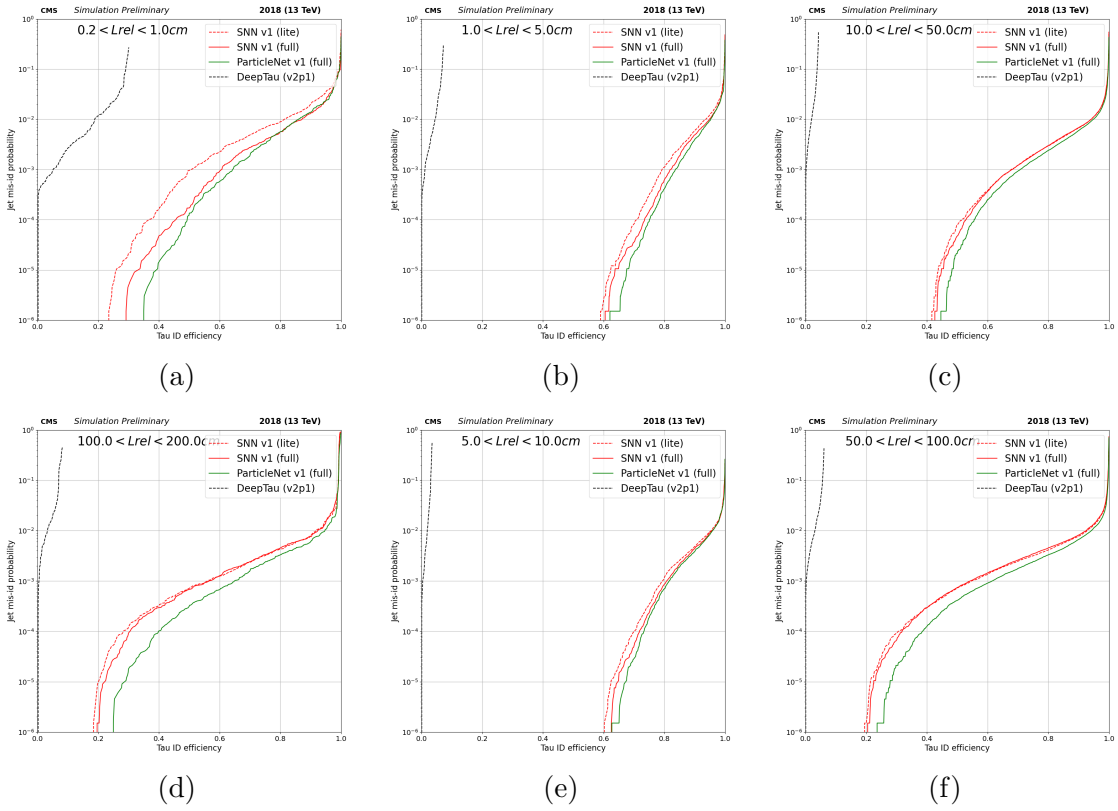


Figure 10: ROC curves for the DeepTau+HPS, SNNlite, SNNfull and ParticleNet architectures in the low p_T region and for six different displacement region L_{rel} in the CMS tracker

- comparable performance to ParticleNet, only slightly worse in some efficiency regions

3.4. Issues

There are a few issues that arose during the tuning of the architecture and must be addressed before any further procedures:

1. SNN (37 GFLOPs) appears to be ten times lighter than ParticleNet (417 GFLOPs), but the CPU run time indicates ParticleNet as the faster architecture.

2. The main bottleneck of SNN architecture is the tiling of the tensors inside the SNN MPM block, which is done two times. This might be directly correlated to the slower CPU run time of SNN

3. sigmoid activation function for the output layer was used instead of the theoretically correct softmax activation, because the network performs better with the latter

4. Conclusions and outlook

CMS continues its effort to search for Beyond-the-Standard-Model particles. None of these searches found any substantial excess in event yields over the predicted background from Standard Model processes, hence the search results are interpreted in terms of production cross section exclusion limits. The latest additions to the set of searches are the searches for direct production of tau sleptons, which are extremely difficult due to the small production cross sections and the challenging reconstruction of final states with tau leptons.

Machine learning models have already become ubiquitous in the field of high energy physics, and its discovery potential continues to grow each day. The current CMS two-step workflow for tau reconstruction, the HPS+DeepTau algorithm, is poorly sensitive to the displaced tau coming from the tau slepton. A dedicated tau jet tagging procedure is necessary to study this SUSY interaction. The current state of the art for jet tagging is based on particle clouds, the ParticleNet.

A lighter architecture, the Space Neural Network, based on distance-weighted GNN, has been developed for the task of jet tagging. In this report we explored the modifications and the hyperoptimization of this network. We looked at the batch loss and batch AUC to discern what modifications were beneficial for the performance, and we trained the network for five epochs before comparing it to ParticleNet. With the use of floating point operations (FLOPs), we observed how our architecture SNN stands at 37 GFLOPs, which is ten times smaller than ParticleNet's 417 GFLOPs. We also observed how our optimization led to SNN being more competitive in terms of mis-identification rate decrease: it is now comparable to the state-of-the-art ParticleNet and just slightly worse in some efficiency regions. We push for the improvement of this SNN architecture because of its small computational complexity. A lot of improvement is still feasible, e.g. better feature engineering or addition of new machine learning models, and there are some issues to be addressed regarding CPU run time and discrepancies between softmax and sigmoid functions but, overall, SNN is a promising jet tagger.

References

[1] Simone Marzani, Gregory Soyez, and Michael Spannowsky. *Looking Inside Jets*. Springer International Publishing, 2019. DOI: 10.1007/978-3-030-15709-8. URL: https://doi.org/10.1007/978-3-030-15709-8.

[2] CMS Collaboration. "Performance of the CMS Level-1 trigger in proton-proton collisions at $\sqrt{s} = 13$ TeV". In: (2020). DOI: 10.48550/ARXIV.2006.10165. URL: https://arxiv.org/abs/2006.10165.

[3] CMS Collaboration. "The CMS trigger system". In: (2016). DOI: 10.48550/ARXIV.1609.02366. URL: https://arxiv.org/abs/1609.02366.

[4] Benjamin Kreis. *Particle Flow and PUPPI in the Level-1 Trigger at CMS for the HL-LHC*. 2018. DOI: 10.48550/ARXIV.1808.02094. URL: https://arxiv.org/abs/1808.02094.

[5] CMS Collaboration. "Performance of reconstruction and identification of τ leptons decaying to hadrons and ν_τ in pp collisions at $\sqrt{s} = 13$ TeV". In: (2018). DOI: 10.48550/ARXIV.1809.02816. URL: https://arxiv.org/abs/1809.02816.

[6] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. "The anti-k_t jet clustering algorithm". In: (2008). DOI: 10.48550/ARXIV.0802.1189. URL: https://arxiv.org/abs/0802.1189.

[7] Tianqi Chen and Tong He. "Higgs Boson Discovery with Boosted Trees". In: *Proceedings of the 2014 International Conference on High-Energy Physics and Machine Learning - Volume 42*. HEPML'14. JMLR.org, 2014, pp. 69–80.

[8] "Performance of the DeepTau algorithm for the discrimination of taus against jets, electron, and muons". In: (2019). URL: https://cds.cern.ch/record/2694158.

[9] Huilin Qu and Loukas Gouskos. "ParticleNet: Jet Tagging via Particle Clouds". In: (2019). DOI: 10.48550/ARXIV.1902.08570. URL: https://arxiv.org/abs/1902.08570.

[10] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. DOI: 10.48550/ARXIV.1512.03385. URL: https://arxiv.org/abs/1512.03385.

[11] Zonghan Wu et al. "A Comprehensive Survey on Graph Neural Networks". In: (2019). DOI: 10.48550/ARXIV.1901.00596. URL: https://arxiv.org/abs/1901.00596.

[12] Shah Rukh Qasim et al. "Learning representations of irregular particle-detector geometry with distance-weighted graph networks". In: (2019). DOI: 10.48550/ARXIV.1902.07987. URL: https://arxiv.org/abs/1902.07987.

[13] Zewen Li et al. "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects". In: *IEEE Transactions on Neural Networks and Learning Systems* (2021), pp. 1–21. DOI: 10.1109/TNNLS.2021.3084827.

Acknowledgements

As summer slowly fades to its slumber, your memories within will remain in its full wonder...
This experience was nothing short of amazing. Almost two months of physics, science, adventures and friendships. I've learnt a lot, not just about machine learning and deep neural networks, but mostly about how beautiful human relationships are. First of all, I want to take a moment to acknowledge my supervisor, Mykyta. Getting chosen as his student and learning a ton about computing and neural networks is really a privilege. He considered me as his equal, and that put me in a comfortable position to work in this beautiful project of τ jet tagging. I also wanted to acknowledge Soham; he was practically my second supervisor, helping me with a lot of technical problems and always available when I needed him. I also want to mention Isabel Melzer-Pellman and Dirk Krucker for their precious suggestions throughout the whole program. A few people also helped me get through the two month programme with a smile in the face: thanks to the people at my office, Gabriele, Benno and David.
Last and not the least, I want to acknowledge the people with whom I spent the most time eating, laughing, travelling, drinking, partying, cooking, talking, gossiping, singing and many other things: we have the iτlians, Jacopo (and his capybaras), Francesca (and her *Einmal ist keinmal* tattoo), Beppo (and his Bari accent), Matteo (and his humongous appetite), Edo (and his cool shirts), Matilde (and her camera) and Francesco (and his tallness); the non iτlians, Paul (and his frenchy english accent), Erik (and his Šakotis), Max (and his free beers), Alain (and his Sunset instead of Schanze) and David (and his Cuban charm). Thank you all from the bottom of my heart. Good luck in life and I hope we see each other again.

A. Variables and scaling

Variable	scaling
pfCand_pt	linear
pfCand_eta	linear
pfCand_phi	linear
pfCand_lostInnerHits	linear
pfCand_nPixelHits	linear
pfCand_caloFraction	linear
pfCand_hcalFraction	linear
pfCand_rawCaloFraction	linear
pfCand_rawHcalFraction	linear
pfCand_mass	normal
pfCand_nHits	normal
pfCand_dxy	normal
pfCand_dxy_error	normal
pfCand_dz	normal
pfCand_dz_error	normal
pfCand_track_chi2	normal
pfCand_track_ndof	normal
pfCand_valid	categorical
pfCand_charge	categorical
pfCand_hasTrackDetails	categorical
pfCand_particleType	categorical
pfCand_pvAssociationQuality	categorical
pfCand_fromPV	categorical
pfCand_deta	no scaling
pfCand_dphi	no scaling
pfCand_puppiWeight	no scaling
pfCand_puppiWeightNoLep	no scaling

Table 3: Feature vector and their scaling methods

B. Jet daughters features

C. Batch AUC and loss plots