UNIVERSIDADE D
COIMBRA

João Humberto Damião da Silva Sampaio Gomes

# MACHINE LEARNING IN QCD JETS

**Dissertação no âmbito do Mestrado em Física, no ramo de Física Nuclear e de Partículas, orientada pelo Professor Doutor José Ricardo Morais Silva Gonçalo e pela Professora Doutora Liliana Marisa Cunha Apolinário e apresentada ao Departamento da Física da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.**

Fevereiro de 2023

UNIVERSIDADE D
COIMBRA

# Machine Learning in QCD Jets


**Supervisor:**

José Ricardo Morais Silva Gonçalo

**Co-Supervisor:**

Liliana Marisa Cunha Apolinário


**Jury:**

Prof. Brigitte Anabelle Vaz Abreu Hiller

Prof. João Nuno Ramalho Gonçalves Pires

Prof. José Ricardo Morais Silva Gonçalo

Dissertation submitted in partial fulfillment for the degree of Master of Science in Physics.

Coimbra, February 2023

# Acknowledgments

First and foremost, I'd like to thank my supervisors, Ricardo and Liliana, for all the guidance they gave and all the patience they had with me. Without their fantastic help, this task would have been impossible.

I'd also like to acknowledge the financial support from FCT with a grant under project EXPL/FIS-PAR/0905/2021).

I'd also like to thank my parents and my sister, who always supported me even though they had no idea what I was working on.

Lastly, I want to thank my girlfriend, Maria, who gave me all the emotional support I needed and helped me draw some of the diagrams presented here.

# Resumo

Jactos são conjuntos de partículas colimadas que nascem de eventos de colisão de altas energias. São um paralelo experimental útil à fragmentação de partões altamente energéticos, os constituintes de toda a matéria hadrónica. Sendo os graus de liberdade fundamentais da Cromodinâmica Quântica (QCD), a evolução da fragmentação de partões em jactos pode ser compreendida como a sequência probabilística da emissão de partões. Os métodos de reagrupamento de jactos tentam recuperar fenómenos que ditam esta evolução de "Parton shower". Estas ferramentas de reconstrução têm sido muito utilizadas nas experiências de colisionadores de protão-protão com o objetivo de compreender os alicerces de QCD.

Os geradores de eventos Monte Carlo (MC) são amplamente usados na física de altas energias - simulam colisões experimentais e constroem técnicas essenciais para propor a estas experiências. Neste trabalho, fizemos uma análise comparativa das subestruturas criadas por dois mecanismos de parton shower implementados no gerador de eventos PYTHIA. Para isto geramos colisões de protão-protão a energias de colisão do LHC e futuros aceleradores, identificamos as subestruturas observáveis dos jactos $\Delta$ e $\kappa$, onde as distribuições eram afetadas pela escolha do mecanismo.

Tendo identificado estas observáveis, construímos uma rede neuronal que seria treinada para identificar qual dos mecanismos tinha sido usado para gerar um certo evento, usando as observáveis anteriormente mencionadas como input. Este treino não teve um sucesso sginficativo, visto que a exactidão da rede se manteve estagnada, em parte devido às limitações da rede neuronal, mas também é o resultado de uma concordância notável entre os dois algoritmos diferentes de simulação de chuveiro de partões.

# Abstract

Jets are collections of collimated particles that emerge from high-energy collision events. They are a useful experimental proxy for the fragmentation of a high-energy parton, the building blocks of all hadronic matter. Being the fundamental degrees of freedom of Quantum Chromodynamics (QCD), the evolution of parton fragmentation into jets can be understood as a probabilistic sequence of parton emissions. Jet reclustering methods attempt to recover QCD phenomena that govern this parton shower evolution. These reconstruction tools have been heavily used in proton-proton collision experiments to understand the QCD building blocks.

Monte Carlo (MC) event generators are widely used in High Energy Physics, to predict observable distributions in collider experiments and are essential tools for these experiments. In this work, we made a comparative analysis of the jet substructure created by two different parton shower algorithms implemented in the PYTHIA MC event generator. To this end, we generated proton-proton collisions at energies available at the LHC and future colliders, and identified the jet substructure observables $\Delta$ and $\kappa$, where distributions were most affected by the choice of the simulation algorithm.

Having identified such observables, we built a Neural Network which would be try to identify which mechanism was used to generate a particular event, using the aforementioned observables as input. This was unsuccessful, as accuracy was not improved, likely due to limitations with the neural network, but also because both algorithms show notable agreement in the simulation of parton showers.

*"Somewhere, something incredible is waiting to be known."*

*Carl Sagan*

# Contents

# List of Acronyms

| | |
|---|---|
| **QCD** | Quantum Chromodynamics |
| **IRC** | Infrared and Collinear |
| **LO** | Leading Order |
| **C/A** | Cambridge/Aachen |
| **UE** | Underlying Event |
| **MC** | Monte Carlo |
| **ISR** | Initial State Radiation |
| **FSR** | Final State Radiation |
| **DGLAP** | Dokshitzer–Gribov–Lipatov–Altarelli–Parisi |
| **MPI** | Multi-Parton Interactions |
| **RMSE** | Root Mean-Squared Error |
| **MSE** | Mean-Squared Error |
| **ROC** | Receiver Operating Characteristic |
| **FPR** | False Positive Rate |
| **TPR** | True Positive Rate |
| **ANN** | Artificial Neural Network |
| **MLP** | Multilayer Perceptron |
| **ML** | Machine Learning |
| **LHC** | Large Hadron Collider |

| | |
|---|---|
| **FCC** | Future Circular Collider |

# List of Figures

# List of Tables

# 1 Introduction

Monte Carlo (MC) event generators are today indispensable in the study of High Energy Physics, as they allow the study of high energy collisions in a much more accessible form than raw data from an experiment at a particle accelerator like the LHC. They work by executing a series of algorithms that simulate the various processes involved in these events, using random number generation in order to simulate the probabilistic effects of quantum mechanics. It is vital that these mechanisms are simulated in agreement with both theoretical calculations and experimental results.

In this work, we take a look at one of the mechanisms implemented in a event generator called PYTHIA. We generate proton-proton collisions at various energies and analyze how two different parton shower models, which simulate the behavior of radiation of QCD partons, and compare them using Machine Learning models by the reconstructed jet substructure. The goal is to develop a neural network that will be able to compare Monte Carlo simulations and real collision events, to infer on the properties of real jets. This will be done using reconstructed jets as a proxy for the QCD parton shower in order to assess this unobserved process and constrain further these models. The outcome would be a data-driven discriminator to use on real hadronic collider events. As such, we organized this text in the following way:

In chapters 2 and 3, a necessary contextualization of the problem is made. Chapter 2 discusses how jets relate with Quantum Chromodynamics; what jet algorithms are used; jet substructure; and Monte Carlo event generators, including a discussion on PYTHIA and its parton shower algorithms, which will be the main focus of this work. In Chapter 3, we give some basic notions about machine learning and neural networks.

In chapter 4, we look at the jet substructure of events generated with 2 parton shower models at various energies and kinematic cuts. A two-fold approach is done at this stage: first we attempt to find settings and observables where the choice of a parton shower has the greatest difference; then, we train various neural networks to distinguish between events generated by either model. New understanding about QCD building blocks.

# 2 QCD, jets, and parton showers

## 2.1 Quantum chromodynamics

Quantum Chromodynamics (QCD) describes the behavior of quarks and gluons with an SU(3) gauge theory, represented by the Lagrangian density [6]:

$$\mathcal{L} = \overline{\psi}(i\gamma^{\mu}D_{\mu} - M)\psi - \frac{1}{4}F^{a}_{\mu\nu}F^{a\mu\nu} \tag{2.1}$$

With $\psi, \overline{\psi}$ representing the quark and anti-quark fields, respectively, and $F^{a}_{\mu,\nu}$ is the gluon field strength tensor for a gluon with color index $a, a = 1, ..., 8$, given by:

$$F^{a}_{\mu\nu} = \partial_{\mu}A^{a}_{\nu} - \partial_{\nu}A^{a}_{\mu} + g_{s}f^{abc}A^{b}_{\mu}A^{c}_{\nu} \tag{2.2}$$

With $f^{abc}$ being the structure constant of SU(3) and $D_{\mu}$ is the covariant derivative in QCD, given by:

$$(D_{\mu})_{ij} = \delta_{ij}\partial_{\mu} - ig_{s}t^{a}_{ij}A^{a}_{\mu} \tag{2.3}$$

The SU(3) gauge theory is a consequence of the existence of 3 color charges, $r(\overline{r}),\ b(\overline{b}),\ g(\overline{g})$.

At high energy scales, $Q \gg \Lambda_{QCD}$, the coupling constant $\alpha_s$ is small. Because of that, matrix elements can be calculated with perturbation theory, at fixed orders in the strong coupling constant. Perturbation theory allows us to describe the dynamics of quarks and gluons at high energies. However, the particles which can be observed are hadrons, whose description requires non-perturbative techniques. These cannot be calculated as a function of fundamental parameters of the Lagrangian, but instead must fit to models with constraints to experimental data.

It is possible, in fact, to factorize the cross-section of a QCD process in perturbative and non-perturbative components. For a general hadron-hadron process, we have [7]:

$$d\sigma_{h_1h_2} = \sum_{i,j} \int_0^1 dx_i \int_0^1 dx_j \sum_f \int d\Phi_f f_{i/h_1}(x_i, \mu_F^2) f_{j/h_2}(x_j, \mu_F^2) \frac{d\hat{\sigma}_{ij\to f}}{dx_i dx_j d\Phi_f} \tag{2.4}$$

With i and j running for all possible parton types and f for all possible partonic final states (i.e. $d\hat{\sigma}$ is a partonic cross section). $f_{i/p}(x_i, \mu_F^2)$ is the parton density function, which is the probability of finding a parton $i$ inside hadron $p$, with momentum $p_i = x_i p_h$, when probing at a scale $\mu_F$.

## 2.2 Jets

In a high energy collision, the resulting partons will radiate repeatedly until eventually hadronization is reached. The result is that when such an event occurs, collimated beams of hadrons will be observed. Those beams form the basis for the notion of a jet [1].

Jets are extremely useful structures. They work as a proxy for the parton that originated them, bridging a gap between the predictions of perturbative QCD and the observed hadrons. Events can be studied by looking at the number of jets, jet kinematics, and jet substructure (i.e smaller jets within a certain jet).

While this notion is intuitive, jets need to be defined. Given the high number of particles involved, identifying jets is a task done by computers, according to a certain jet reclustering algorithm, which determines which particles belong to a certain jet and those that don't. This isn't, however, a trivial or uniquely task. A recombination scheme, which (re-)combines the particles' 4-momenta into a jet's 4-momentum, is required.

The Snowmass accord [8], established in 1990, gives a set of rules which a good jet algorithm should follow which still guide the study of jets today. It should be:

1. Simple to implement in an experimental analysis;
2. Simple to implement in the theoretical calculation;
3. Defined at any order of perturbation theory;
4. Yields finite cross sections at any order of perturbation theory;
5. Yields a cross section that is relatively insensitive to hadronisation.

If a jet follows these rules, then it will serve as a good proxy for the study of the partons.

One final property that jet algorithms should obey is Infrared and Collinear (IRC) safety. An IRC safe algorithm is one where the number of jets isn't altered with the addition of

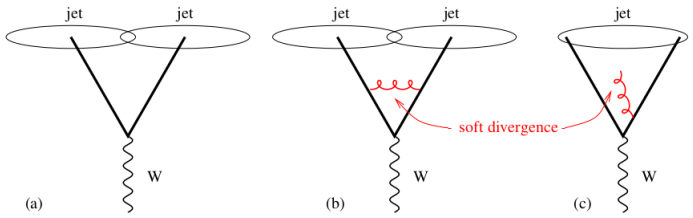



Figure 2.1: An infrared unsafe algorithm. The addition of a soft emission changes the number of jets. Taken from [1].

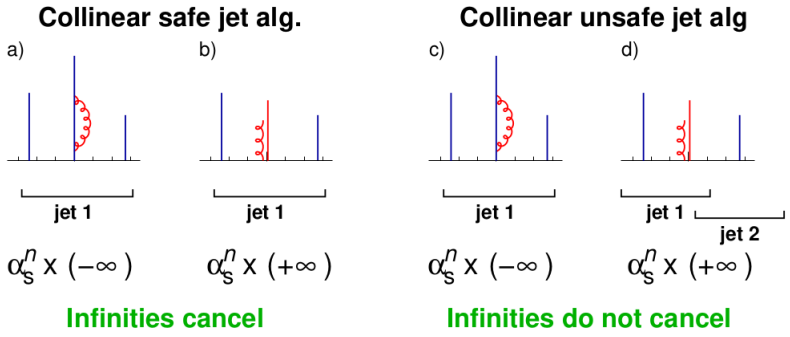



Figure 2.2: The effect of collinear safety. In the figure on the left, a soft emission does not alter the number of jets. Taken from [1].

arbitrarily soft or collinear radiation. In figures 2.1 and 2.2 illustrate infrared and collinear safety, respectively.

There are several reasons why this is a desired feature, namely that:

- Hard partons cause collinear splittings in the fragmentation process;
- QCD events go through other processes of both collinear and infrared radiation, mostly through non-perturbative processes;
- Collinear radiation can also be a consequence of other non-perturbative processes such as hadron decay;
- In order to agree with perturbative QCD, soft emissions and collinear splittings are associated with divergent tree-level matrix elements (quote from Towards Jetography, maybe adapt here), which should be compensated by divergent loop matrix elements which have the opposite sign. If those two elements in the diagram are not in the same jet, then they won't cancel, which means the jets obtained may not agree with pQCD.
- Experimental detectors have finite resolution, both in energy and angle, and therefore are naturally IRC safe.

The first jet algorithms, mainly a family called *cone algorithms*, were plagued with IRC unsafety. This is clearly undesirable, as the number of jets and related observables may have little relation with the partons which originated the jets, and therefore they won't agree with perturbation theory. Although it is possible up to determine the order of perturbation theory at which a jet algorithm will break (we can, for example, have an algorithm which is accurate up to Leading Order (LO) when a soft particle is added), but having to make these considerations for every jet is not practical, and comparison between theory and experimental results is limited regardless. Therefore, algorithms which don't have these limitations, i.e. IRC safe, are preferable.

## 2.3 Jet algorithms and relevant quantities

In this section, we will look at some jet algorithms, which will be relevant for this work, as well as recombination schemes.´ Another important notion is the recombination scheme, which determines how two particle's momenta, $p_i, p_j$ are combined into a new, combined momentum, $p_{ij}$. The simplest and most widely used recombination scheme is E-recombination, which just sums the two momenta, that is,

$$p_{ij} = p_i + p_j$$

.

### 2.3.1 Sequential recombination algorithms

The sequential recombination algorithms [9] are a family of jet algorithms which appeared as a result during the study of $e^+e^-$ events, but it's widely used in the study of $pp$ and heavy ion collisions. In general, that work by joining particles which are closest according to some distance.

One type of sequential recombination algorithms are the generalized-$k_t$ family, which works as following:

1. We take 2 particles, $i, j$, and set a quantity $R$ which we'll call the jet radius.

2. Work out the quantities:

$$d_{ij} = \min(p_{ti}^{2p}, p_{tj}^{2p}) \frac{\Delta R_{ij}^2}{R^2} \tag{2.5}$$

$$\Delta R_{ij} = (y_i - y_j)^2 + (\phi_i - \phi_j)^2 \tag{2.6}$$

$$d_{iB} = p_{ti}^2 \tag{2.7}$$

Where $y$ is the rapidity and $\phi$ is the azimuthal angle, and $d_{iB}$ is called the beam distance, and $p$ is a continuous parameter.

3. If $d_{ij} < d_{iB}$, we merge the particles i and j and start again from step 2.

4. If $d_{iB} < d_{ij}$ we say declare i to be a final state jet, remove it from the list of particles, and go back to step 2.

5. Repeat the process until no particles remain.

This family of algorithms allows us to establish a clustering history of the jet, by looking at which particles get merged first and which are merged later.

The behavior of the jet algorithm depends on the choice of the parameter $p$ . The case $p = 1$ generates the so-called $k_t$ algorithm [10], which will tend to merge soft particles first. [11] The choice p = - 1 gives the anti-$k_t$ algorithm [12]. This algorithm minimizes distances to hard particles, resulting in the creation of circular jets around hard particles. This makes this sort of algorithm ideal for identifying *signal* jets.

The p = 0 case generates the Cambridge/Aachen (C/A) algorithm [13], which clusters particles merely based on their distance in $(y - \phi)$ space. At leading logarithmic accuracy, QCD vacuum emission follows angular ordering, which means that subsequent emissions can't be emit at higher angles than previous ones. Because of this, the C/A algorithm mimics the QCD radiation pattern.

Another relevant choice is $p = 0.5$, called the $\tau$ algorithm [11] which is somewhat energy dependent, and will therefore tend to merge soft particles first, but not as strongly as the $k_t$ algorithm, having instead intermediary characteristics between it and the C/A algorithm. It also relates to formation time of an emission, the time it takes to behave as an independent source of radiation, since the distance measure is given by

$$d_{ij} \approx p_{tt}\Delta \sim \frac{1}{\tau_{form}} \tag{2.8}$$

The jet radius R is typically set between 0.2 and 1. The specific choice depends on the goal we propose: a smaller jet radius prevents the effects of background radiation, but it excludes more particles, creating a higher number of jets. By contrast, a larger radius includes more particles within the jet, but is also more susceptible to background effects.

Figure 2.3 illustrates how the these algorithms identify different jets in generated by similar events.

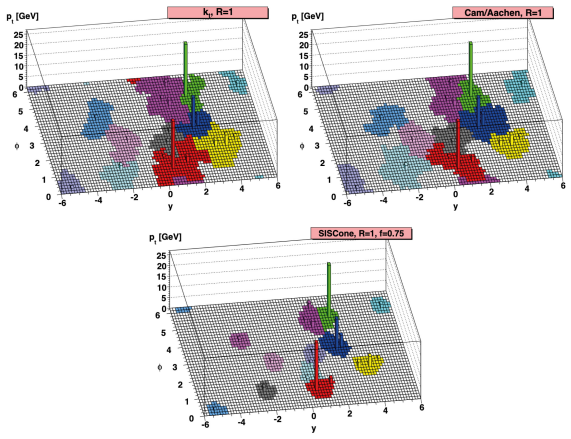



Figure 2.3: Parton-level event generated with Herwig showing the effects of different jet algorithms. Taken from [1].

### 2.3.2 Non-perturbative effects

Some effects observed in jets do not occur due to perturbative QCD, and are instead caused by low-$p_t$ non-perturbative radiation. Hadronization, that is the process of partons turning into hadrons, as well as interactions between spectator particles (called the Underlying Event (UE)), both play into this. As we shall see in the next section, and in order to keep the desired jets protected from non-perturbative effects (as per the Snowmass accord), techniques will be presented which remove this low $p_t$ radiation from the analysis.

## 2.4 Jet substructure

After having methods of identifying jets, we can move on to finding jet substructure, that is, looking at sub-jets within jets. In this way, in the same way that a jet works as a proxy for the hard parton that originated it, sub-jets work as proxies for the particles that it emits.

Jet substructure has had various applications, such as flavor tagging (discriminating jets initiated by light quarks, heavy quarks, or gluons). Another possible application is the study of Lorentz-boosted heavy particle decays (since the boost can cause the hadronic decays to lie all on one single jet instead of multiple ones).

### 2.4.1 The Lund Plane

One tool to look at the jet substructure is the Lund plane [14], which is a representation of phase space within jets. The following process is used to construct it [2]:

1. Decluster that jet into two sub-jets, $j_1$ and $j_2$, such that $p_{t1} > p_{t1}$, using the C/A algorithm.

2. Determine the quantities

$$\Delta = (y_1 - y_2)^2 + (\phi_1 - \phi_2)^2 \tag{2.9}$$

$$z = \frac{p_{t1}}{p_{t1} + p_{t2}} \tag{2.10}$$

$$\kappa = z p_{t2} \tag{2.11}$$

$$k_t = p_{t2}\Delta \tag{2.12}$$

$$m^2 = (p_1 + p_2)^2 \tag{2.13}$$

$$\tag{2.14}$$

3. Repeat the process for the sub-jets $j_1$ and $j_2$, if they have more than one particle.

By following this process, a tuple $\mathcal{T}^{(i)} = (\Delta, \kappa, z, ...)$ is obtained, which has information about the splitting into two sub-jets. The observables $\kappa_i, \Delta_i$ can be served to construct the Lund plane (it's possible, for example, to use the dimensionful $k_t$ instead of $\kappa$. Usually the logarithms $\ln(1/\Delta), ln(\kappa)$ are used instead, because emissions of soft/collinear gluons are uniformly distributed in $\ln(1/\Delta), \ln(\kappa)$ space [15]. This produces a triangle as the one shown in figure 2.6a, where the diagonal line corresponds to the kinematic limit $z = 0.5$.

The choice of the C/A is due to it using only angular separation in calculating the distance measure, which should follow angular ordering, such as what happens with QCD radiation in a vacuum. Therefore, sub-jets identified with this algorithm should correspond to the radiation created by the emission of a parton.

Often times, it may be useful to look at a single branch instead of the whole tree. One could, for example, look only at the hardest sub-jet at each declustering, thereby only focusing on the hardest emissions at each step.

The Lund planes consist of mapping these points $\Delta, \kappa$, or rather, their logarithms $\ln(1/\Delta)$, $\ln(\kappa)$. Each point corresponds to an emission from the original jet, as illustrated in figure 2.4. In figure 2.5, the different regions of the Lund plane are identified.

Infrared and collinear safety is also a point of concern. The tuples $\mathcal{T}^{(i)}$ are not necessarily IRC safe, since adding an infinitely soft emission can lead to another declustering, creating another tuple. However, in a finite portion of the Lund plane (which can be achieved with a cut in $\kappa$, IRC safety is recovered.

In figure 2.6a, we see an example of a Lund plane (trimmed by the soft drop procedure, which will be explained in the next section). In the $z$ axis we have the density of points per unit area, given by:

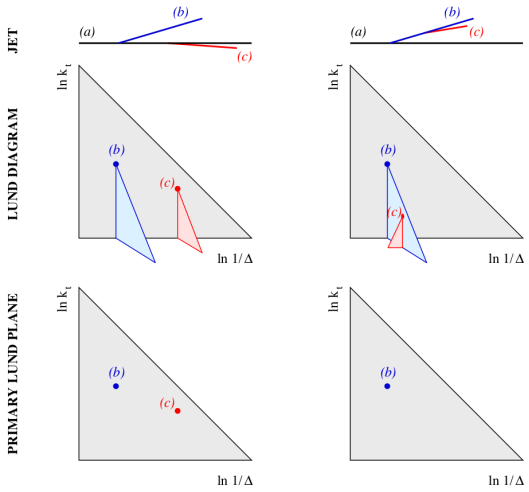



Figure 2.4: Particles inside of a jet and corresponding Lund plane. Each emission corresponds to a point in the Lund plane of the jet from which the particle was emitted. Taken from [2].

$$\rho(\Delta, \kappa) = \frac{1}{N_{jet}} \frac{dn_{emission}}{d\ln_\kappa \, d\ln(1/\Delta)} \tag{2.15}$$

### 2.4.2 The soft drop procedure

One thing that hasn't been considered so far in the creation of the Lund plane is the grooming of non-perturbative effects and soft radiation. These effects cannot be calculated with perturbative QCD, and may be undesirable in the study of hard partons. They are felt mostly in the form of soft and wide-angle radiation. As such, various techniques for jet grooming, which remove this type of radiation, have been established.

The Soft Drop procedure is one of those grooming techniques, and will be applied here: when declustering a jet into 2 components, we evaluate the condition: [16]

$$z_g \equiv \frac{p_{t2}}{p_{t1} + p_{t2}} > z_{cut} \left( \frac{\Delta R_{ij}}{R_{ij}} \right)^\beta \tag{2.16}$$

With $p_t$ being the transverse momentum (with $p_{t1} > p_{t2}$), $\Delta R_{ij}$ being the angular separation in $(y-\phi)$ space defined in equation 2.9, $R$ is the jet radius, and $z_{cut}$ (called the soft drop threshold and $\beta$ (called the angular exponent) are parameters. If the condition isn't met, then the softer sub-jet is removed.

For the purposes of this work, the soft drop procedure with $\beta = 0, z_{cut} = 0.1$ was used. With this choice, the jet splitting function $z_g$ satisfies an extended version of IRC safety. [17]

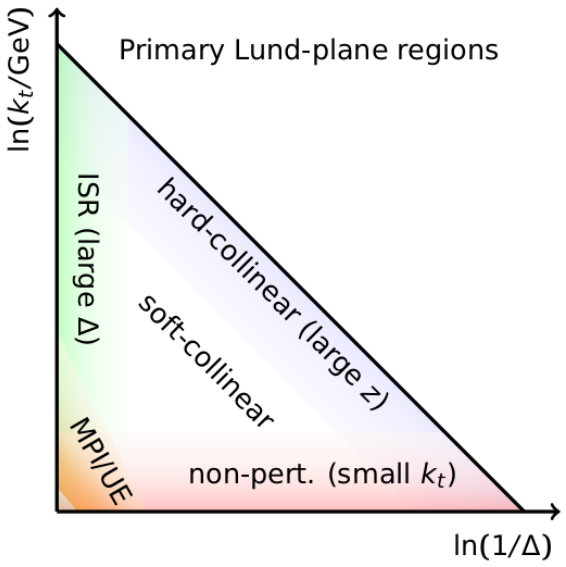
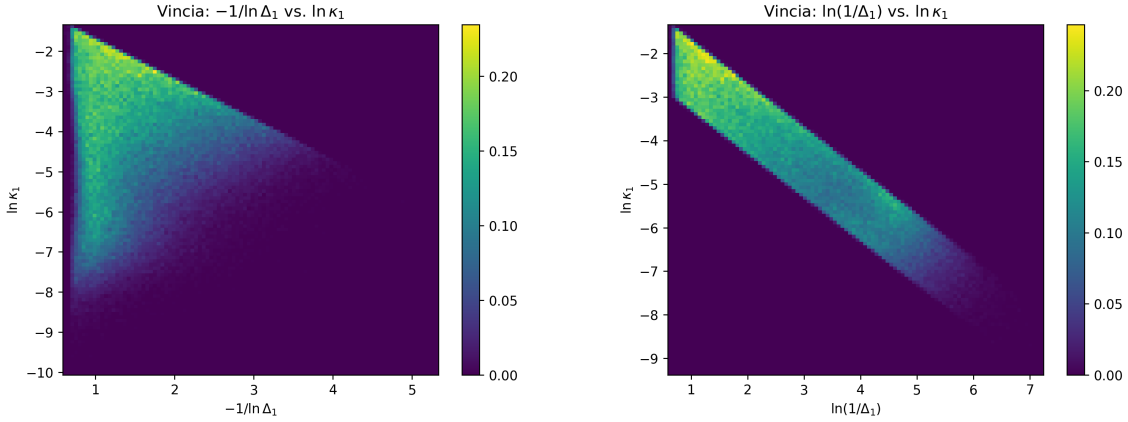



Figure 2.5: The different regions of the Lund plane. Taken from [2].

As shown in figure 2.6, the original triangle that constituted the Lund plane is turned into a trapezoid, with the conditions $z = 0.1$, $z = 0.5$ defining the diagonal lines.




(a) Without Soft Drop

(b) With Soft Drop

Figure 2.6: The effect of trimming the Lund plane with the soft drop procedure, at $z_{cut} = 0.1$, $\beta = 0$, for a $\sqrt{s} = 100TeV$ proton-proton collision, considering only Hard QCD, generated with Pythia8. Anti-$k_t$ jets reclustered with C/A algorithm.

## 2.5 Monte Carlo Event Generators

In order to study processes in particle physics, such as a collision of decay of a particle, it can be useful to computationally simulate such an event. Due to the random nature of quantum mechanics, such a simulation will require the use of random number generation to mimic the probabilistic effects. These programs are called Monte Carlo event generators.

MC event generators such as PYTHIA [18] , HERWIG [19] and SHERPA [20] are today widely used in particle physics. They can be used to study the background, allowing for better discrimination of a signal, but they can also be used for planning and designing new experiments.

In general, a MC event generator has 3 stages. One is the hard-scattering process, where calculations are done using perturbation theory. Another is parton level, which deals with the formation of a multitude of partons, and intends to create a realistic partonic structure, with models called parton showers. A parton shower can occur either before or after the hard scattering, in processes called Initial State Radiation (ISR) or Final State Radiation (FSR), respectively.

Lastly, there is hadron level, which deals with the hadronization of the partons which were created at parton level, using color confinement. Decay of unstable hadrons also happens at this stage.

### 2.5.1 PYTHIA

PYTHIA specifically is one of those MC event generators. A successor to a previous generator called JETSET [21], it stemmed from hadronization studies, and its most important features is the Lund string model of hadronization.

PYTHIA 8.3 includes a multitude of hard processes which can be selected, including Hard QCD, top quark and Higgs production, Electroweak processes, and even contains processes for supersymmetry, dark matter and other exotic processes. Within this work, we will use Hard QCD processes only, which contains 3 types of processes: (1) $2 \rightarrow 2$ scattering of light quarks and gluons, (2) production of heavy flavors (charm and bottom) in $2 \rightarrow 2$ processes, and (3) $2 \rightarrow 3$ processes involving light quarks and gluons. It is also worth noting that, at this stage, it's necessary to apply phase space cuts in order to avoid soft and collinear singularities in the limit $p_t \rightarrow 0$.

PYTHIA 8.3 has 3 different parton shower models implemented, with different mechanisms. The next section will explain how a parton shower works in general terms. This will be followed by a discussion about 2 of the 3 parton shower models implemented in PYTHIA.

### 2.5.2 Parton showers

When looking at high energy collisions, particularly pp collisions for the purpose of this work, the jets obtained contain a very complex substructure, caused by partons emitting other partons constantly until the hadronization scale is reached, below 1 GeV for light quarks. High energy events can have numbers of partons reaching into the hundreds, which makes computations of all those matrix elements computationally unviable. Instead, what Monte Carlo event generators like PYTHIA do is, after the hard scattering, apply a parton shower, which is nothing more than a model which attempts to describe how the hard scattering evolves towards softer and softer partons, until the hadronization scale is reached.

Historically, parton showers are usually divided into or initial state radiation (ISR), final state radiation (FSR) depending on whether they occur before or after the hard scattering process.

Let's start with FSR. Consider a splitting in the form $a \rightarrow bc$, with a,b,c being partons. The differential probability for this branching is:

$$d\mathcal{P}_a(z, Q^2) = \frac{dQ^2}{Q^2} \frac{\alpha_s(Q^2)}{2\pi} \sum_{b,c} P_{a \rightarrow bc}(z) dz \tag{2.17}$$

Where z represents the momentum fraction: b takes fraction z of the original parton's

momentum, and c takes 1-z.

Dokshitzer–Gribov–Lipatov–Altarelli–Parisi (DGLAP) evolution equations, which describe the variation of parton distribution functions with varying energy scales, provides the following splitting kernels [22].

$$P_{q\to qg}(z) = \frac{4}{3}\frac{1+z^2}{1-z} \tag{2.18}$$

$$P_{g\to gg}(z) = 3\frac{\left(1-z(1-z^2)\right)^2}{z(1-z)} \tag{2.19}$$

$$P_{g\to q\bar{q}}(z) = \frac{1}{2}\left(z^2 + (1-z^2)\right) \tag{2.20}$$

Following the condition $P_{a\to bc}(z) = P_{a\to cb}(1-z)$. The latter kernel is only for one quark color and has to be summed for all possible color channels.

Finally, we can integrate over the kinematically allowed $z$ range:

$$d\mathcal{P}_a(Q^2) = \frac{dQ^2}{Q^2}\frac{\alpha_s(Q^2)}{2\pi}\sum_{b,c}\int_{z_{min}(Q^2)}^{z_{max}(Q^2)} P_{a\to bc}(z)dz \tag{2.21}$$

For the ISR, we take the probability starting from the parton distribution function (eq. 77). Like with FSR, this equation is determined by splittings of the form $a \to bc$, but written in terms of the production rate of $b$ instead of the decay rate of $a$, having (eq. 78 of Pythia manual)

$$d\mathcal{P}_b(Q^2) = \frac{dQ^2}{Q^2}\frac{\alpha_s(Q^2)}{2\pi}\sum_{a}\int_{z_{min}(Q^2)}^{z_{max}(Q^2)} dz\frac{x' f_a(x', Q^2}{x f_b(x, Q^2)}P_{b/a}(z) \tag{2.22}$$

Where we used $z = x/x'$.

### 2.5.3 The Simple Shower

The "simple shower" model is the oldest parton-shower algorithm presenting in PYTHIA 8, and the default model. It stems from the older mass-ordered shower in previous iterations of Pythia, having changed to $p_t$ ordering with the goal of combining ISR and FSR evolution with Multi-Parton Interactions (MPI) [23]. Its name points to being more limited in its goals than its more advanced counterparts implemented in PYTHIA, such as the other model which will be studied in this work, Vincia.

This shower consists of one mechanism for ISR and one for FSR (as given by equations 2.22 and 2.21, respectively) but in terms of transverse momentum. This means that, instead of using a generic $Q^2$ scale, it uses the transverse momentum $p_{\perp\ evol}$. The parton shower is applied to a parton from a scale $p_{\perp\ max}$ downwards, until a cutoff energy $p_{\perp\ min}$ is reached.

### 2.5.4 The Vincia Parton Shower

VINCIA [24] is another parton shower model, which like the simple shower is ordered on $p_\perp$, but using the antenna formalism. In this formalism, a gluon emitted from a $q\overline{q}$ pair can be treated as radiation from the color dipole between $q$ and $\overline{q}$. As such, instead of using the $1 \to 2$ branchings that guide the simple shower (whose equations for FSR and ISR we've seen previously), it uses $2 \to 3$ parton branchings. In this model, both parent particles can acquire transverse recoil.

This type of shower was pioneered by the ARIADNE [25] model, the first parton shower model to include branchings of this sort. Vincia expanded on the ARIADNE model.

The evolution variable is based on a generalized version of the ARIADNE definition of transverse momentum. In a generalized branching $IK \to ijk$:

$$p_j^2 = \frac{\tilde{q}_{ij}^2 \tilde{q}_{jk}^2}{s_{max}} \tag{2.23}$$

These two different mechanisms for parton showers will share the recoil of the radiation differently.

### 2.5.5 The Dire shower

The DIRE parton shower [26] is the other parton shower model available in PYTHIA 8.3. It combines aspects of the $2 \to 3$ antenna showers with the "conventional" $1 \to 2$ showers. It will not be approached in this work, but it's mentioned for sake of completion.

# 3 A brief touch into Machine Learning

## 3.1 What is machine learning

Machine learning refers to a set of algorithms which can take data and use it in order to improve performance on a task. This idea can be summarized by the following quote:

> A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E." —Tom Mitchell, 1997 [27]

There is a multitude of applications for machine learning algorithms, not all of them as futuristic-looking. Besides the obvious examples of chess-playing computers, "basic" things like spam filters or a search engine use machine learning.

These sort of algorithms can be extremely useful. They can automate processes that wouldn't be possible with traditional computing. When there are a large number of rules that we couldn't explicitly give to a traditional computing program, for example, machine learning can be a useful tool.

Machine learning algorithms work by forming patterns: we sample some of the data and input it into the algorithm, part of the overall set we intend to study, and train it on that data. The algorithm will have a mechanism that allows it to identify improvements or decreases in performance, and will evaluate its predictions on that data. The process can be iterated and, by performing more and more adjustments, performance may be improved.

It is also important not to train an algorithm on the particular peculiarities of a randomly chosen set of data which can't be generalized. Therefore, it is common practice to have a separate set of data, called a test set (in opposition with the training set), and evaluate the final model on it. A program successfully trained should have similar performances between the training and test set. When the test set underperforms the training set, we call it overfitting.

We can distinguish machine learning algorithms based on what the sort of task they perform. In regards to the type of prediction, we can have either:

1. **Regression task**: The algorithm attempts to predict a numerical value for a given input. A very simple example of this would be a linear regression

2. **Classification task**: The algorithm attempts to assign a case to a class. An example of this would be a spam filter, which classifies e-mails as "spam" or "not-spam".

In classification algorithms, supervision is also an important concept. We can have either a supervised algorithm, where the data has a label attached, and the goal of the algorithm is to learn patterns that allow it to predict the label; or we can have an unsupervised algorithm, where it divides the data according to its own criteria.

As mentioned previously, we need a performance measure, a way of assessing the quality of the algorithm. In the next sections, further discussion will be done on the appropriate measure. For classifications tasks, a simple and widely used metric for performance, however, is accuracy, defined by

$$\text{Accuracy} = \frac{\text{Correct predictions}}{\text{Total predictions}} \tag{3.1}$$

For a regression task, the Root Mean-Squared Error (RMSE) is widely used as well:

$$\text{RMSE} = \sqrt{\frac{1}{\text{m}}\sum_{\text{i}=1}^{\text{m}}(\text{h}(\mathbf{x}^{(\text{i})}) - \mathbf{y}^{(\text{i})})^2} \tag{3.2}$$

Where $\mathbf{x}$ is the input data, $\mathbf{y}$ is the correct value, and $h(\mathbf{x})$ is the predicted value.

A cost function is also required. A cost function is a function that the algorithm intends to minimize, which should correlate with performance. By slightly changing parameters (according to a metric called the learning rate), and evaluating the cost function after each change, the algorithm can find patterns that work (as in, they minimize the cost function), and those that don't. Usually, the Mean-Squared Error (MSE) is used as cost function.

In this work, a cost function used for binary classifications, called binary cross-entropy, is used. It is given by: [28]:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} y_i \log(p(y_i)) + (1 - y_i)\log(1 - p(y_i)) \tag{3.3}$$

Where $N$ is the number of points, $y_i$ is the label to each (either 0 or 1), $p(y_i)$ is the probability of the label being correctly assigned.

Finally,there needs to be a method for optimizing a cost function. By having a parameter called the learning rate $\eta$, one can shift $\theta$ by a certain amount in order to minimize the function $\mathcal{L}(\theta)$. The optimizer used in this case is the Adam optimizer [29].

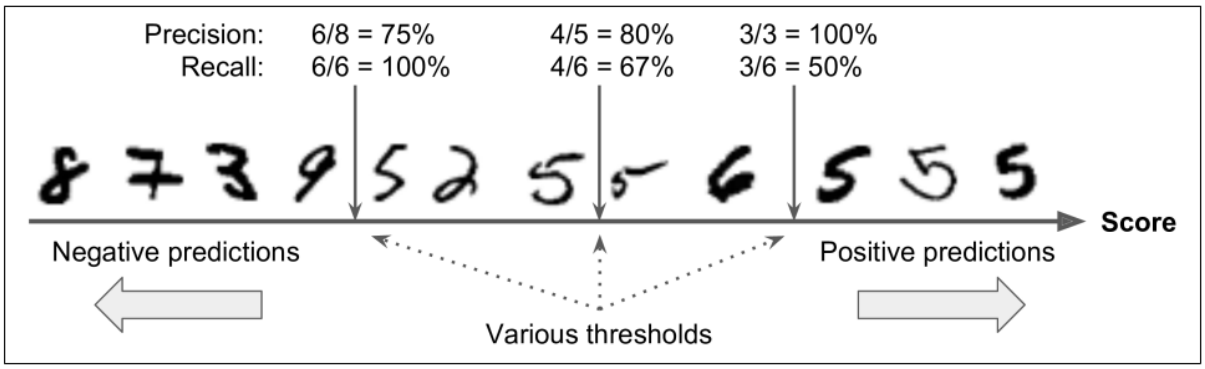



Figure 3.1: A machine learning algorithm that distinguishes handwritten 5 from non-5s. Changing the threshold means trading off precision for recall or vice-versa. Taken from [3].

## 3.2 Measures of quality

We've already looked at two performance measures, the RMSE for regression tasks and accuracy for classification tasks. But these can be insufficient. Think, for example, of a spam filter, in a case where a user receives 99% legitimate e-mails and 1% spam. A classifier that predicted all e-mails as legitimate would have a 99% accuracy. And yet, it would be completely useless at deleting any spam.

2 measures are quite useful when it comes to binary classification tasks: precision and recall, defined as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{3.4}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{3.5}$$

These measures both serve to represent measure quality. However, they usually come at a tradeoff, as shown in figure 3.1.

By varying the threshold (i.e the number at which the network classifies the output as positive instead of negative), then, it is possible to control whether to focus more on a higher precision or recall. One way of visualizing this tradeoff is by using the Receiver Operating Characteristic (ROC) curve [30], which plots the False Positive Rate (FPR) against the True Positive Rate (TPR) (which is the same as the recall). Each point of this curve corresponds to a threshold. An example of an ROC curve is plotted on figure 3.2.

A perfect machine learning algorithm, tuned to the ideal threshold would have a FPR = 0 and TPR = 1 which would put it on the top left corner. The area under the ROC curve itself

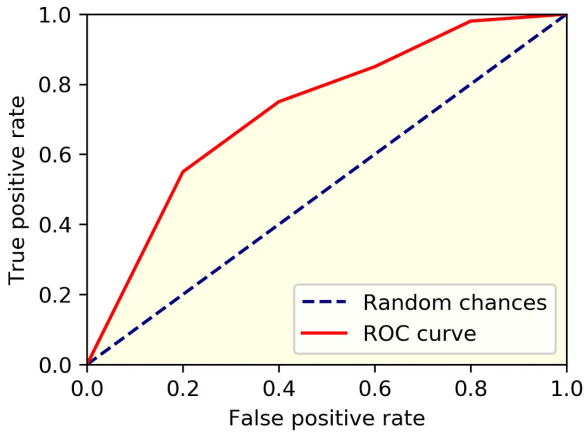



Figure 3.2: An example of an ROC curve. Notice the dashed diagonal line, which corresponds to a random guess. Taken from [4].

can be a measure of quality. [31] For a perfect algorithm, that area would be 1, while for a random guess it would be 0.5.

## 3.3 Neural Network

There are a multitude of machine learning algorithms, such as support vector machines, decision trees, random forests, and many more. However, one that has shown to be of particular interest, especially in recent years, are Artificial Neural Network (ANN). The name derives from their similarities to their biological counterparts.

The original neural networks were, like biological neurons, joined by layers. In biological systems, neurons from one layer connect to those from at the previous layer. These connections are either reinforced or weakened while learning some task. In order to understand an artificial neural network works, we will start by explaining the functioning of an artificial neuron. It takes the values $x_1, x_2, ...$ of one layer and computes a weighted sum:

$$z = w_1 x_1 + w_2 x_2 + ... \tag{3.6}$$

To the value $z$ a function $\phi$, called the activation function, is applied.

$$h_w(\mathbf{x}) = \phi(z) \tag{3.7}$$

Where $\phi$ is called an activation function. The initial models used the Heaviside or step function, but today other functions like the logistic ($\sigma(z) = 1/(1 + \exp(-z))$) or the rectified linear unit ($\text{ReLU}(z) = \max(0, z)$) are used instead. An illustration of this mechanism can be seen in figure 3.3

By adjusting the weights, performance can be improved. In order for this to happen, a rule must be implemented to reinforce connections that reduce the error. This is shown by the following equation.

$$w_{i,j}^{(\text{next step})} = w_{i,j} + \eta(y_- \hat{y}_j)x_i \tag{3.8}$$

Where $w_{i,j}$ is the weight of the connection between i and j, $\eta$ is the learning rate, $y_j$ is the target output of neuron j at a certain instance, $\hat{y}_j$ is the output of that neuron, and $x_i$ is the input value.

A Multilayer Perceptron (MLP) is a neural network architecture consisting of one input layer, one or more layers of artificial neurons (called the hidden layers), and one final layer of artificial neuron(s) which is the output layer. At every layer except for the output there is a bias neuron (which has a constant value of 1). An artificial neural network with a large number of layers is called a deep neural network, and it serves as the basis for the field of deep learning.

MLPs can be used for regression tasks, but also for classification. For a binary classification, one just needs a single output between 0 and 1.

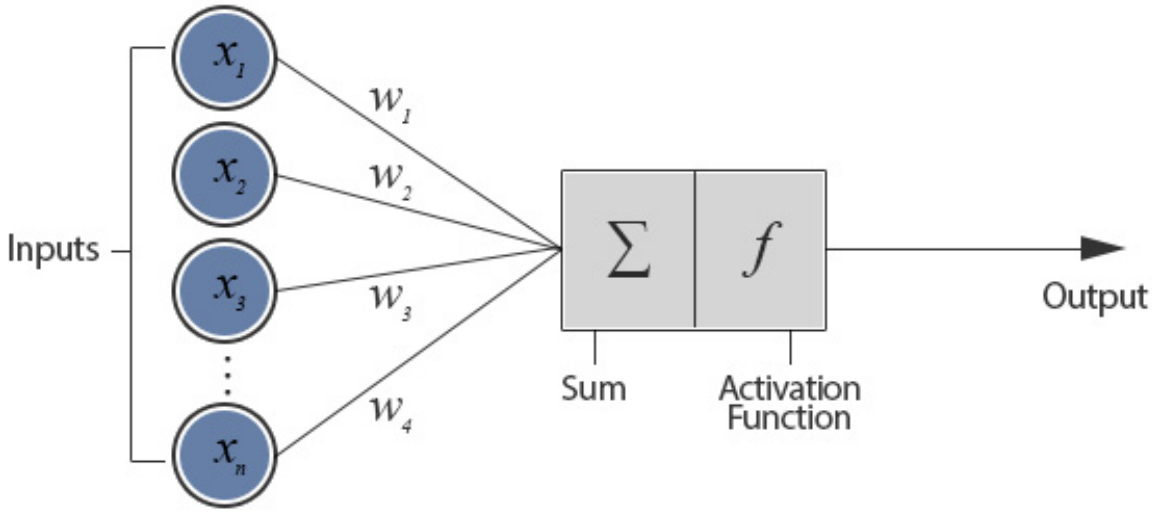



Figure 3.3: The behavior of an artificial neuron. Taken from [5].

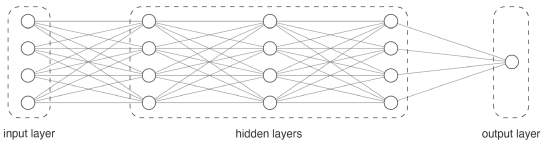



Figure 3.4: A multi-layer perceptron with 4 inputs, 3 hidden layers with 4 neurons each, and an output layer with 1 neuron.

## 3.4 Machine Learning in jet physics

### 3.4.1 The Lund Plane as a classification algorithm

Machine Learning (ML) techniques have also been applied in recent years to the study of jet physics [2], using variables associated with the Lund plane. The original Lund plane paper used a Long Short-Term Memory architecture for W tagging, that is, identifying jets initiated by a W decay from a background of QCD jets. Sequences of the form $(\ln(1/\Delta), \ln k_t)$, collected from declustering the leading branch (as shown in figure 4.1) were used as input, with an output predicting whether a given point in the Lund plane was generated by the W decay or by the QCD background.

In subsequent studies, [32], more refined networks using other neural network architectures, have also been used similarly for top tagging and for discriminating quark and gluon-initiated jets, using inputs of the form $(\ln(1/\Delta), \ln k_t, \ln z)$ or $(\ln(1/\Delta), \ln k_t, \ln z, \ln m, \ln \Psi)$

In the next chapter, we will move towards using the Lund Plane for another classification task: it will be used as a way of distinguishing between the Simple and Vincia parton showers on QCD jets. For this, it's necessary to find an appropriate set of variables, and train a neural network correctly. Training consisted in giving the network $N_{train}$ events with a label attached, indicating which parton shower model generated them. Being run by a number of epochs, it would be tested on a validation set at the end of each epoch, and after training was completely finished the test set was evaluated. Accuracy was used as performance measure, with the ROC curve being plotted at the end.

# 4 Comparing the models' jet substructure

The two parton shower models mentioned in chapter 2, having different mechanisms, can create different jet substructures for the same collision. In this chapter, the effects of the Vincia or Simple Shower models on jet substructure are studied. Proton-proton collisions, considering only Hard QCD processes (only light quarks and gluons are generated) are generated with both models, and a comparison of the various observables associated with the jet substructure are made.

After finding a set of observables where the models show different behavior, a neural network was trained in order to distinguish events generated by either jet. If successfully trained, this method could be expanded to compare jets generated by a Monte Carlo event generator with real jets, and could be used as a way of assessing the quality of the event generation. For each test, 2 simulations are run, under the same conditions, only varying the parton shower model. In each event, jets are identified with the anti-$k_t$ algorithm using Fastjet [33], a package used for identifying jets. The leading jet, the jet with highest transverse momentum, is picked. Kinematic cuts in transverse momentum and rapidity are then applied, in order to ensure that this jet results from a hard scattering process. One million events are run for each model.

The jet is then reclustered into two sub-jets using either the C/A or the generalized $k_t$ algorithm (with p = 0.5), constructing one of the Lund planes. Once the events are generated, a comparison is done between the histograms of the relevant observables. The Lund planes are also compared, by looking at the difference between both models. The soft drop procedure is applied, with $z = 0.1, \beta = 0$.

Finally, after selecting appropriate variables, a neural network is trained to distinguish events generated by either model.

## 4.1 Finding the appropriate variables

Our first task is to find appropriate observables. By appropriate, it is meant that they should: be impervious to non-perturbative effects, and there should be some distinctions between the

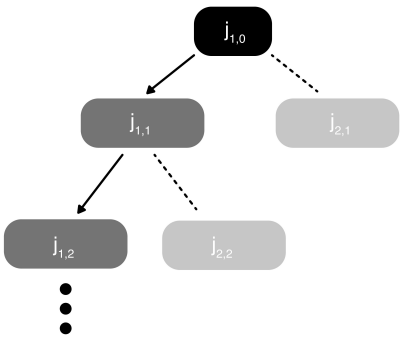



Figure 4.1: Collecting variables along the leading branch. We obtain a sequence of pairs $(\ln(1/\Delta), \ln(\kappa))$, by always picking the sub-jet with highest $p_t$. In dashed are the sub-jets that get discarded by picking this branch.

two models. The first requirement excludes immediately observables like the jet mass, which is more sensitive to soft and wide-angle radiation.

Our first test uses simulated data with with a center-of-mass energy $\sqrt{s} = 7TeV$. We look at a few variables to see which ones have potential in terms of finding any distinctions. The jets have a radius $R = 0.5$, to which we apply two kinematic cuts: they must have a minimum transverse momentum, $p_t > 300$ GeV/c and a maximum rapidity $\eta < 2.0$. Reclustering is done with the C/A algorithm, using a radius $R = 1$. At each step of declustering, instead of mapping the whole tree, we only look at the leading branch (that with highest $p_t$), as shown in figure 4.1, producing one tuple $T^{(i)}$ for each step of the declustering. This choice of taking the harder branch at every step allows for more radiation from the initial partons, and it's expected that different parton showers will have more distinct behaviors. Finally, it will be required that events go through at least 4 declusterings.

Applying these cuts implies the exclusion of a number of events. Out of the 1 million events generated for each model, only 183558 passed the cuts for the Simple Shower model, and 149534 passed it for the Vincia model. It is to be noted that the Simple Shower model accounted for 55% of the total number of events suitable to be analyzed. This fraction will remain constant for all the tests being performed henceforth, with jets obtained by Vincia failing the kinematic cuts and/or first declustering slightly more often. A review of the jet cuts is present in table 4.1, with the plots of the $p_t$ spectrum shown in figure 4.2.

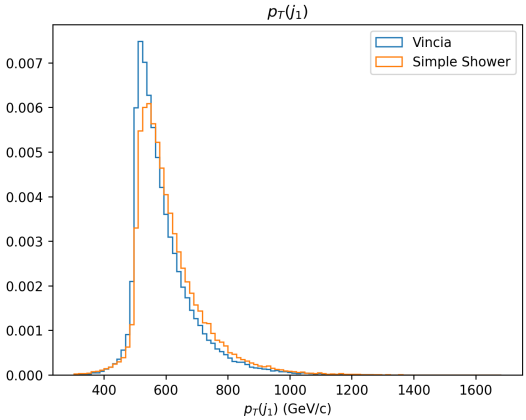



Figure 4.2: Distribution of the leading jet's momentum for the 7 TeV event with a $p_t > 300$ GeV/c cut.

Without jet mass, we are left with 3 possible jet substructure observables: $\Delta, \kappa, z, k_t$ (defined in equations 2.9 to 2.12. Given that $k_t$ is just a dimensionful version of $\kappa$, only the first 3 are analyzed.

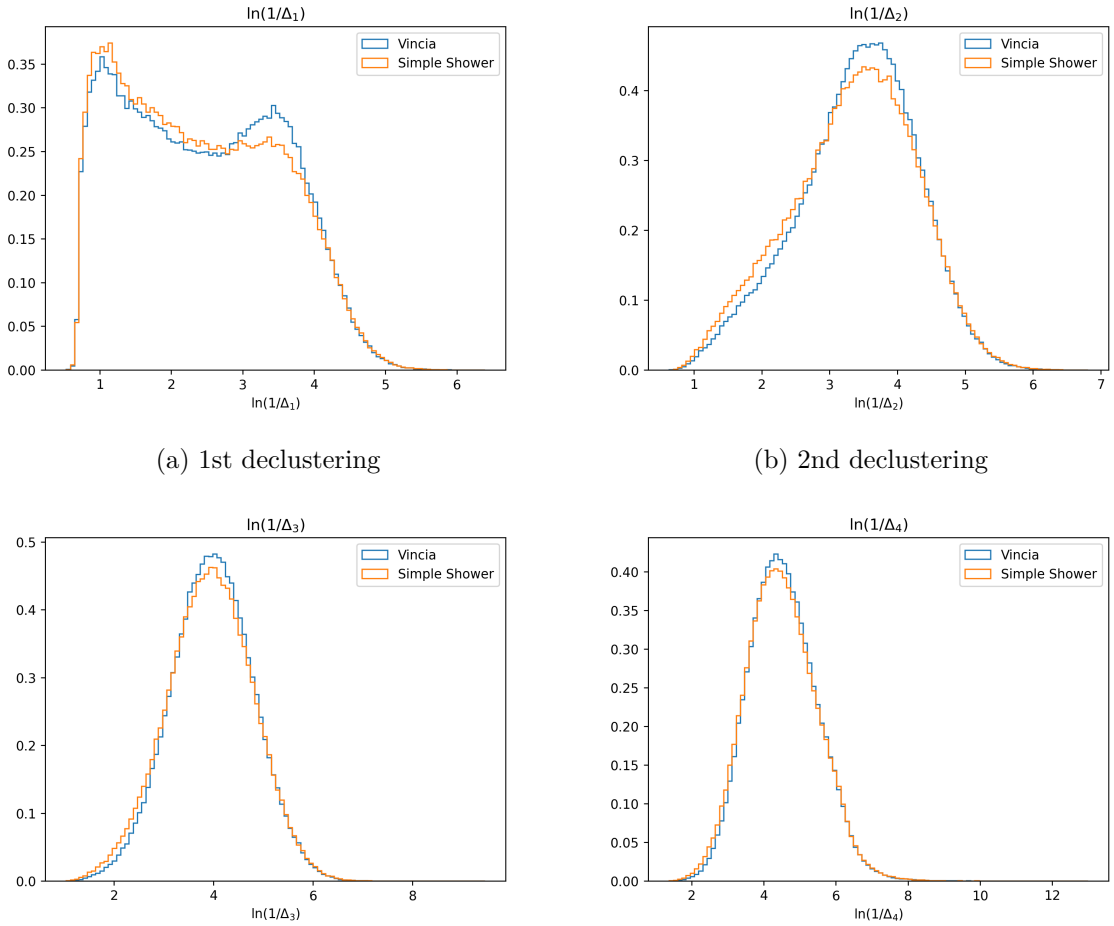
In figures 4.3, 4.4 and 4.5 are plotted the distributions in the first 4 declusterings for $\Delta, \kappa, z$, respectively. Just like done with the representation of the Lund plane, the comparison is done using the logarithms $\ln(1/\Delta_i), \ln \kappa$, for the same reasons as discussed in section 2.4.1. For the jet splitting function $z$, the logarithm $\ln z$ is used as well.

We start by analyzing the distribution of these variables, and then look at where the models show differences. In $\Delta$ and $\kappa$, we can observe a 2 peak distribution for the 1st declustering, with one of them disappearing in subsequent declusterings. These peaks, around $\ln(1/\Delta) = 1$ and $\ln(1/\kappa) = -3$, correspond to large angle radiation and is associated with ISR. The other peak is associated mostly with hadronization, as is discussed in sub-section 4.1.1 in more detail, and is shown in appendix A. Although these hadronization effects are not ideal, the parton showers still affect the distribution of both variables, which means they can be used for this comparison.

Regarding the differences between both models, we can look first at the two Lund plane variables $\Delta, \kappa$, which exhibit rather similar behaviors. It is clear that, for both, differences are decreased with each declustering. This is to be expected since the phase is decreased at each step, and therefore subsequent sub-jets will radiate less, leading to more consistency between the two models.

In both, we also observe the major differences around the peak not associated with ISR, with Vincia showing a sharper peak in every declustering, while Simple Shower exhibits a slightly more spread out distribution. The ISR peak is rather similar between both models.

The $z_i$ has a different behavior. One could expect to see similar behaviors to what is obtained




(a) 1st declustering

(b) 2nd declustering

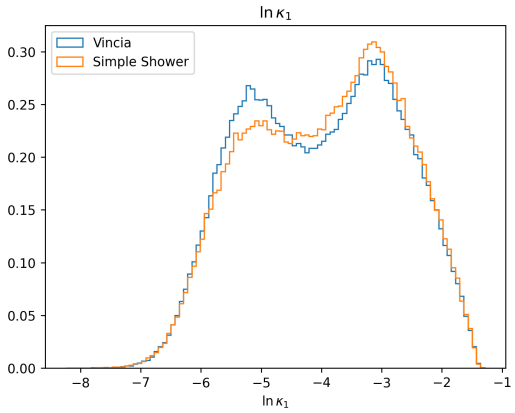
(c) 3rd declustering

(d) 4th declustering

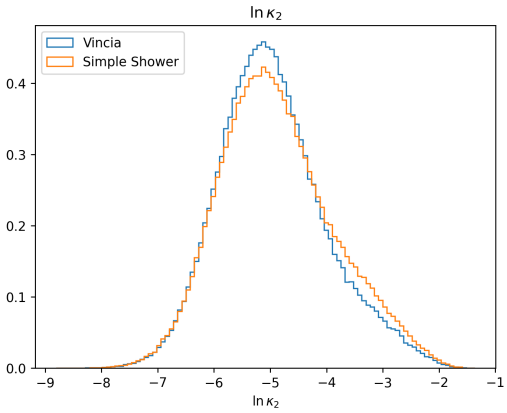
Figure 4.3: Histograms for the angular separation $\Delta$ between two sub-jets, for the leading branch of at a center of mass energy $\sqrt{s} = 7$TeV, with anti-$k_t$ jets (R = 0.5) reclustered with the C/A algorithm (R = 1). Full settings in table 4.1.

for $\Delta$ and $\kappa$, as the different splitting mechanisms could result in different distribution of momentum between sub-jets. Instead, $z$ shows a remarkable consistency between the two models, with distributions being virtually indistinguishable. It is therefore an unsuitable observable for distinguishing the two models.

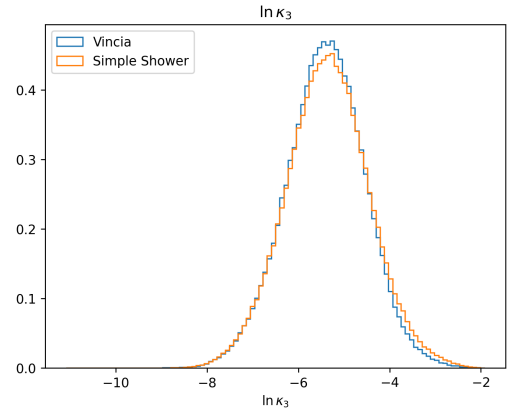
Given this, we are left with two sets of observables, $\Delta_i$ and $\kappa_i$ which may be suitable for training the neural network at distinguishing parton showers, which happen to be variables that can be used in the construction of the Lund plane . Because of this, we may also look at the Lund plane in order to find distinctions between models.



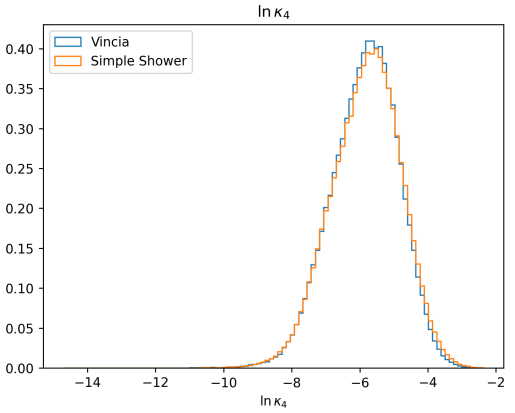

(a) 1st declustering




(b) 2nd declustering

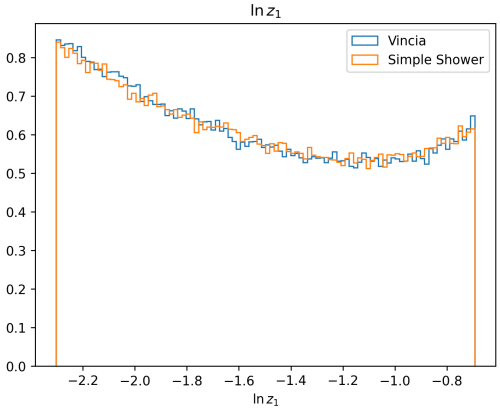



(c) 3rd declustering

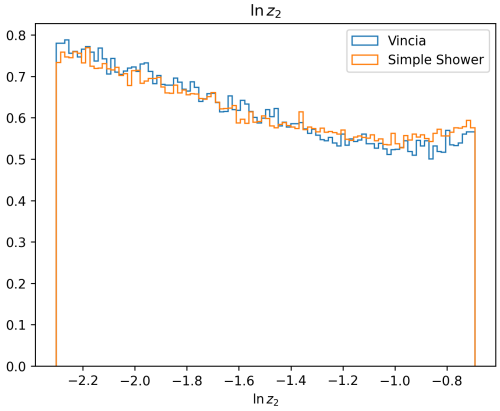



(d) 4th declustering

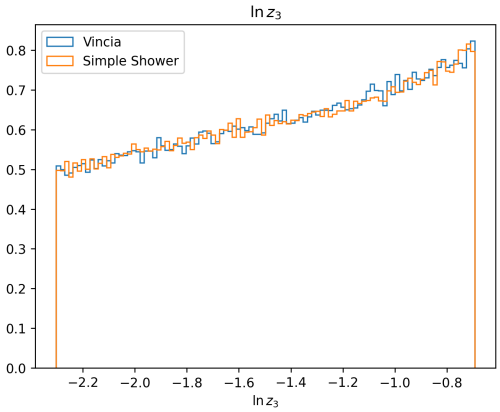
Figure 4.4: Histograms for $\kappa = z\Delta$, for the leading branch of at a center of mass energy $\sqrt{s} = 7$TeV, with anti-$k_t$ jets (R = 0.5) reclustered with the C/A algorithm (R = 1). Full settings in table 4.1.



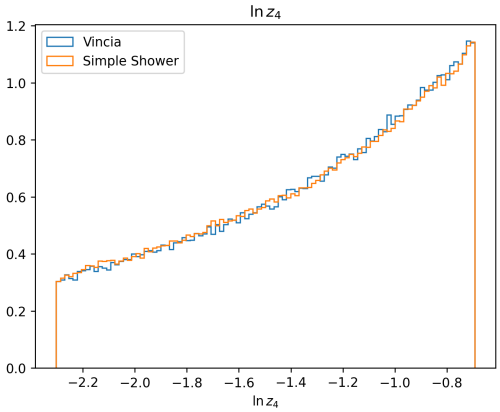

(a) 1st declustering




(b) 2nd declustering

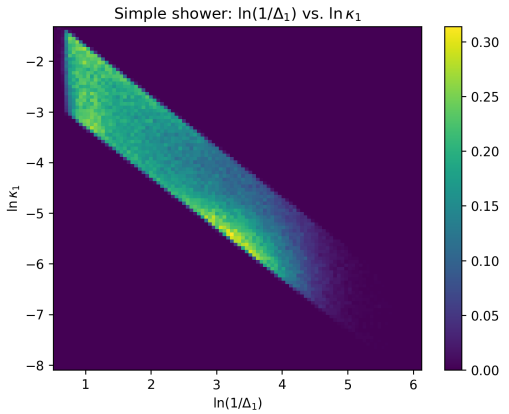



(c) 3rd declustering

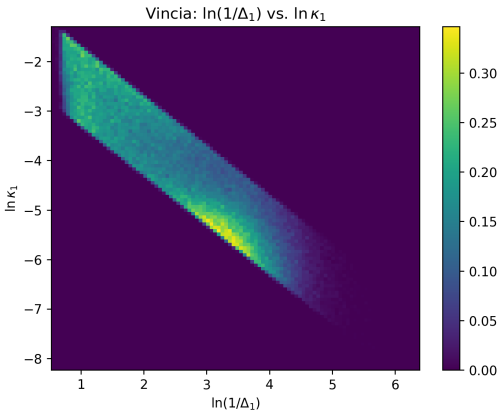



(d) 4th declustering

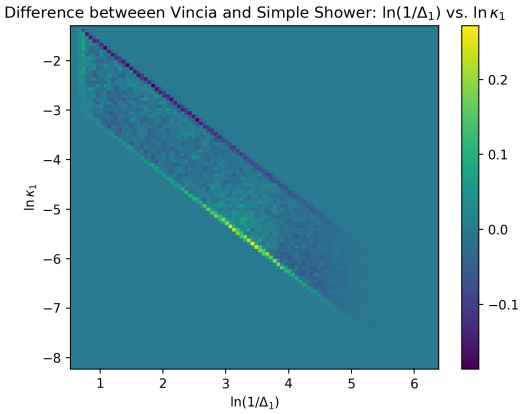
Figure 4.5: Histograms for $z$, for the leading branch of at a center of mass energy $\sqrt{s} = 7$TeV, with anti-$k_t$ jets (R = 0.5) reclustered with the C/A algorithm (R = 1). Full settings in table 4.1.




(a) Simple Shower




(b) Vincia




(c) Difference

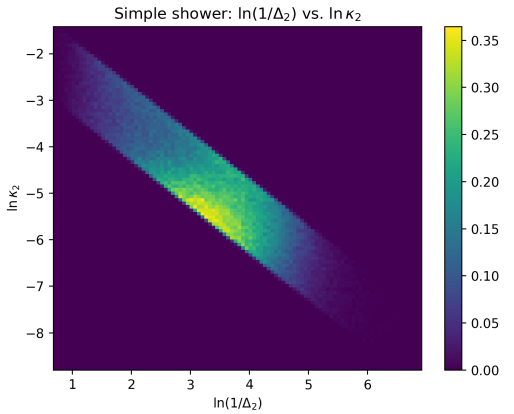
Figure 4.6: The Lund Plane for the first declustering in 7 TeV events (see table 4.1). a) Generated with Simple Shower; b) Generated with Vincia; c) Difference in density between both models.

### 4.1.1 Looking at the Lund Plane

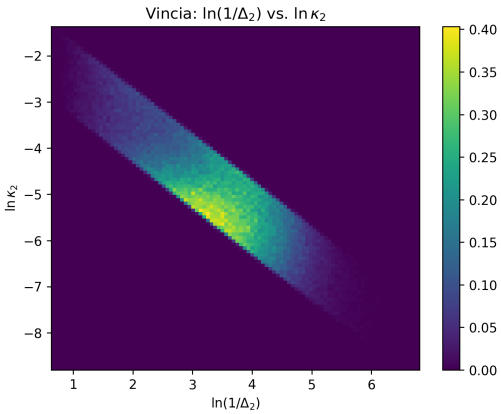
In the previous exercise, we showed how $\kappa, \Delta$ are most suitable for finding disparities. The Lund plane will therefore be a better way of making this comparison, as one can identify which regions cause said differences. Having plotted the first two pairs of $\Delta_i, \kappa_i$, and realizing that there are some small differences between the two models, it might also be enlightening to look at the Lund Plane, which plots the density of the $\Delta$ and $\kappa$ observables simultaneously. In this way, we are able to observe any correlations between the two variables which may behave differently in the two models.

The Lund planes for the first declustering obtained for the Vincia and Simple Shower are present in figure 4.6, as well as the difference in density between them. In figure 4.7 the same is done for the second declustering.

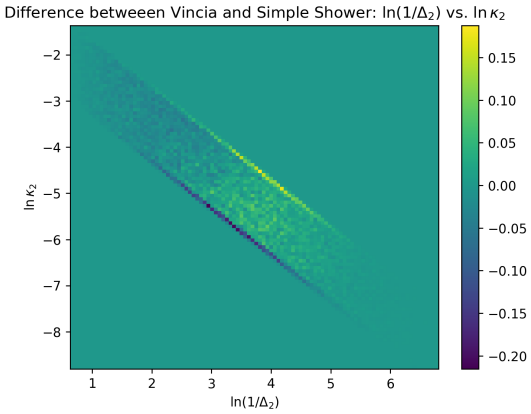
As seen for the histograms, the first declustering shows a higher density zone on the top left,




(a) Simple Shower




(b) Vincia




(c) Difference

Figure 4.7: The Lund Plane for the second declustering in 7 TeV events (see table 4.1). a) Generated with Simple Shower; b) Generated with Vincia; c) Difference in density between both models.

associated with ISR.

In these figures, some important differences can be observed. At the edges, in the diagonal lines corresponding to $z = 0.1$ and $z = 0.5$, there is a significant difference in density between the two models. By contrast, the regions in between these two thin stripes show good consistency between the two models.

## 4.2 Distinction of parton shower models using Machine Learning

Having found a set of appropriate observables, which happen to be the Lund plane variables $\ln(1/\Delta), \ln(\kappa)$, it is possible to train a machine learning algorithm to distinguish events generated from either model.

Since the Lund planes become more and more similar as we move along the tree, and in order

to keep dimensionality low, training was done initially using only $\ln(1/\Delta_1), ln\kappa_1, \ln(1/\Delta_2), \ln\kappa_2$ as input in a neural network, alongside with a binary label (0 or 1), representing whether the event was generated with the Simple Shower (0) or Vincia (1) model. This is therefore a supervised, binary classification task.

### 4.2.1 Network architecture

The neural network was compiled using the package Keras [34] and Tensorflow 2.7.0 [35]. As recommended by [3] for a binary classification task, the Rectifier Linear Unit (ReLU) activation function was used, with Cross-Entropy [36] being used as the loss function, with the optimizer Adam at a learning rate of 0.01. The network was run for either 10 or 30 epochs.

In order to provide better performance, the input went through standard scaling [37]. This is a process where sets are re-scaled so that they have a standard deviation of 1 and a mean of 0. This is common practice because, in general, neural networks have better performance when dealing with numbers of not too distinct magnitudes.

Table 4.1: A summary of settings used for the 7TeV events, and of the neural network trained to compare parton showers.

| **Name** | 7TeV-1 |
|---|---|
| **Event and jet settings** | |
| $\sqrt{s}$ | 7 TeV |
| Jet algorithm | Anti-$k_t$ |
| Jet reclustering algorithm | C/A |
| Jet $p_t$ | $>$ 300 GeV/c |
| Jet $\lvert\eta\rvert$ | $<$ 2.0 |
| Events generated | 1M Vincia + 1M simple |
| % of events kept after cuts | 15.0% (Vincia); 18.3% (simple) |
| **Neural network** | |
| # of declusterings used | 2 |
| Number of layers | 3 |
| Neurons per layer | 4 |
| Activation function | ReLU |
| Optimizer | Adam |
| Learning rate | 0.01 |

### 4.2.2 Measuring the network's performance.

In order to train and evaluate the network, out of the sample of 333092 events, 64% was be reserved for the training set, 16% for the validation set (which is used to compare with the training set at the end of each epoch), and 20% for the test set. The output is a number, between 0 and 1, which predicts what is the appropriate label. As a default setting, it will be assumed a threshold of 0.5. That is, if the output is below 0.5, the event will be predicted as Simple Shower; otherwise, it will be predicted as Vincia. This is merely a default setting on

which to train the neural network. It's possible to, after training, find a new threshold which can have a better performance. A neural network was constructed, using 3 dense layers with 4 neurons each, and a final output neuron. There wasn't any particular reason to choose this configuration, but optimization may be done later on, in order to find the ideal number of neurons and layers. Henceforth, we will call it Neural Network (NN) 1, just like the one in figure 3.4. The collection of jet settings and NN architecture will be referred as 7TeV-1, and can be checked in the appendices.

This network was trained for 30 epochs. In order to evaluate it, we looked at 3 different graphs. The first shows the evolution of the accuracy for both the training and validation set, as well as the evolution loss function; the second is the ROC curve; and the third is showing the networks output for each sort of event.

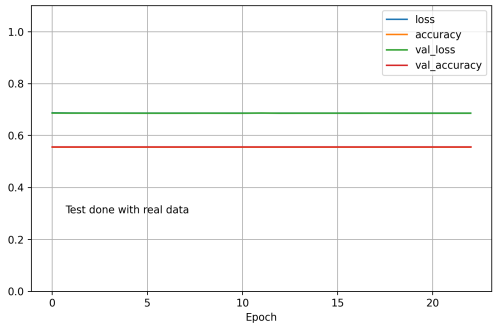
As we can see by figures 4.8a, 4.8b, and 4.8c, this method was incapable of making reliable predictions. The ROC curve is barely above a random guess. Accuracy stays at around 55% from the first epoch onward. Looking at the network output (where we would ideally see a spike around 0 for the simple shower events and a spike around 1 for the vincia events), 2 observations can be done: the two models have almost indistinuishable output curves, and all the values are far from either 0 and 1, with most events having an output around 0.46. With this, it's also possible to see where the 55% accuracy comes from. As mentioned previously, this is the original fraction of simple shower events in the total sample. And, since all events have an output below 0.5, they are all classified as being simple shower.

### Increasing transverse momentum

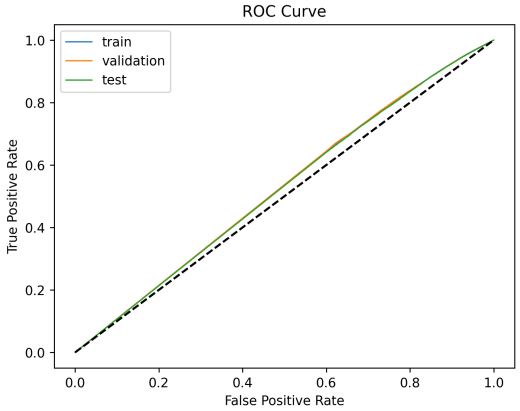
With these dissatisfactory results, instead of trying to improve the neural network's architecture immediately, it might be more enlightening to instead look at higher energies, where partons radiate a lot more and thus slight differences in the mechanisms of the parton shower can be propagated and cause increased discrepancies between models. They are compared again, now using events with center-of-mass energy of 14 TeV, expected for the the high-luminosity phase of the Large Hadron Collider (LHC).

For the same intent of looking at higher energy partons, the jet's minimum transverse momentum was increased to 1 TeV, with the distribution being seen in figure 4.9. The same neural network was considered, and we'll call these settings 14TeV-1, which are fully described in table 4.2. Similarly to what we did in the 7 TeV case, histograms of the selected observables are plotted in figures 4.10 and 4.11-

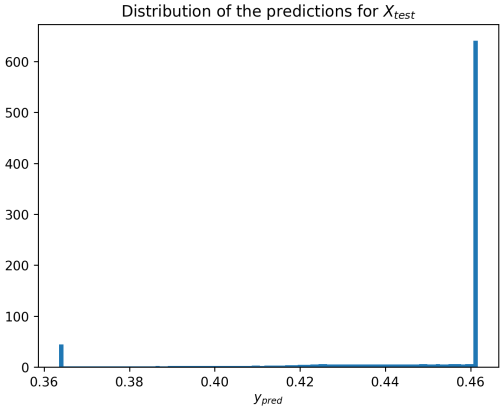
By choosing this higher energy, we notice a slightly different behavior. The rightmost peak




(a) Evolution of the accuracy and loss function




(b) ROC Curve




(c) Neural network output

Figure 4.8: Evaluation of the neural network (7TeV-1), for 7 TeV events using 2 declusterings. In a), we can observe the value of the loss function and accuracy for both the training and validation sets; b) shows the ROC curve for training, validation and test sets; c) plots a histogram of the neural network's output (a number between 0 and 1) for events with either label. In a perfectly trained network, 2 peaks would be observed: one at 0 for the Simple Shower curve, and one at 1 for the Vincia curve.

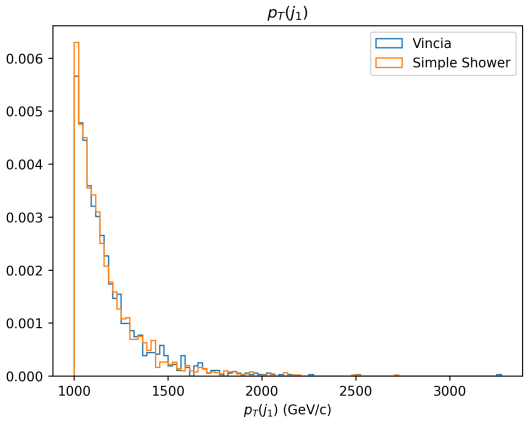
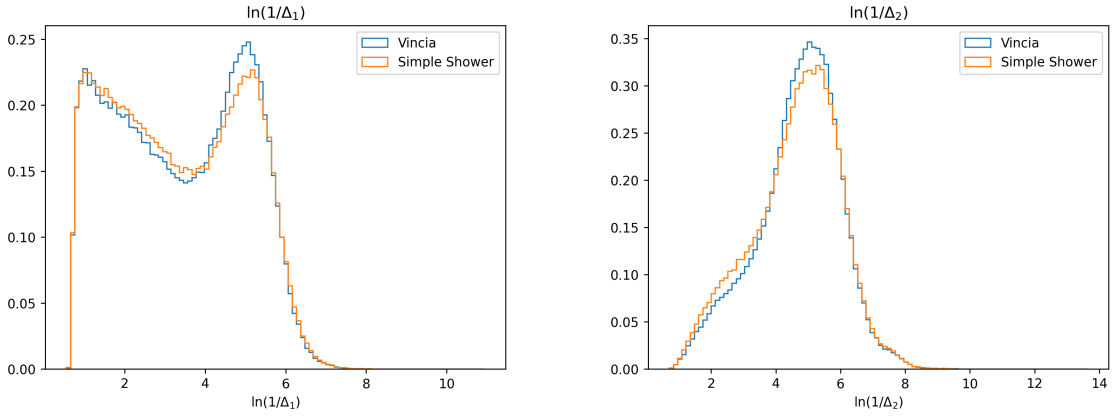


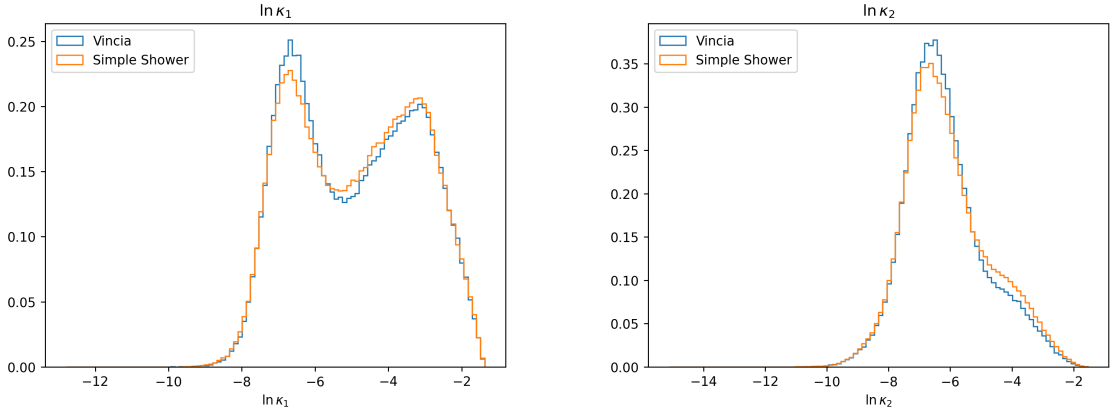

Figure 4.9: Distribution of the leading jet's momentum for the 14 TeV event with a $p_t > 1$ TeV/c cut.




(a) 1st declustering

(b) 2nd declustering

Figure 4.10: Histograms for the angular separation $\Delta$ between two sub-jets, for the leading branch of at a center of mass energy $\sqrt{s} = 14$TeV, with anti-$k_t$ jets (R = 0.5 $p_t > 1$ TeV/c) reclustered with the C/A algorithm (R = 1). See table 4.2.




(a) 1st declustering

(b) 2nd declustering

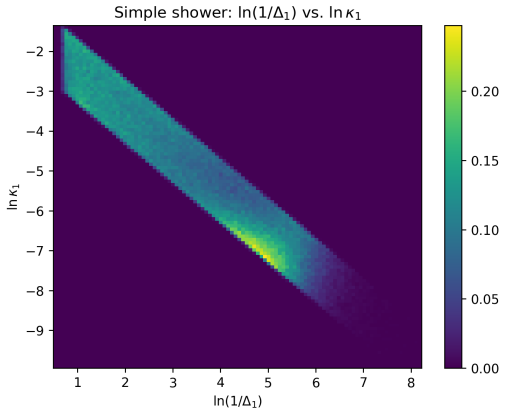
Figure 4.11: Histograms for $\kappa$, for 14 TeV events. See table 4.2.

in $\Delta_1$, associated with initial state radiation (see figure 2.5, is now the greatest peak. However, discrepancies don't seem to become more clear just by these plots. At this energy, and choosing harder jets, the benefit in discrimination doesn't seem obvious. An interesting effect happened, however, with the ISR peak stopped dominating the distribution.

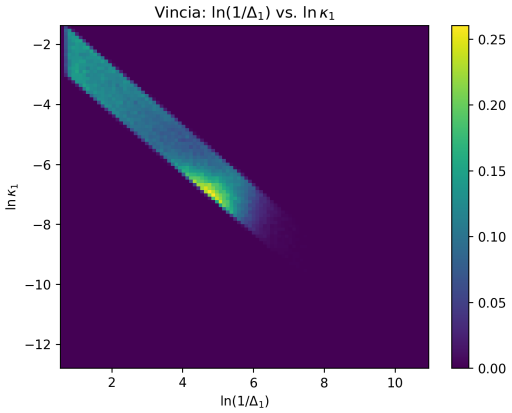
Table 4.2: A summary of settings used for the 14TeV events, and of the neural network trained to compare parton showers.

| **Name** | 14TeV-1 |
|---|---|
| **Event and jet settings** | |
| Branch | Lead branch of leading jet |
| $\sqrt{s}$ | 14 TeV |
| Jet algorithm | Anti-$k_t$ |
| Jet radius | 0.5 |
| Jet reclustering algorithm | C/A |
| Sub-jet radius | 1 |
| Jet $p_t$ | $>$ 1 TeV/c |
| Jet $\mid \eta \mid$ | $<$ 2.0 |
| Events generated | 1M Vincia + 1M simple |
| # of declusterings used | 2 |
| % of events kept after cuts | 24.3% (Vincia); 30.3% (simple) |
| **Neural network** | |
| Number of layers | 3 |
| Neurons per layer | 4 |
| Activation function | ReLU |
| Optimizer | Adam |
| Learning rate | 0.01 |

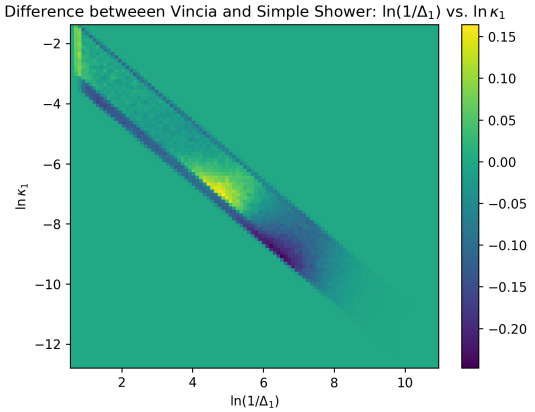
Looking at the Lund planes, however shows more clearly that higher energies can improve discrimination. The thin region around $z = 0.1$ and $z = 0.5$ where discrepancies were seen is now widened, especially for the first declustering. We therefore have a larger region where to find distinctions. In spite of this, performance stays exactly the same when compared with the 7 TeV configuration.




(a) Simple Shower

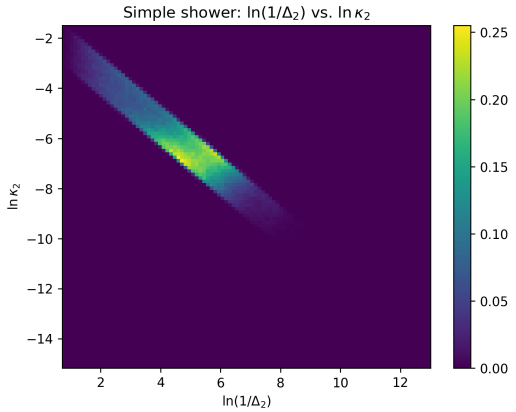



(b) Vincia

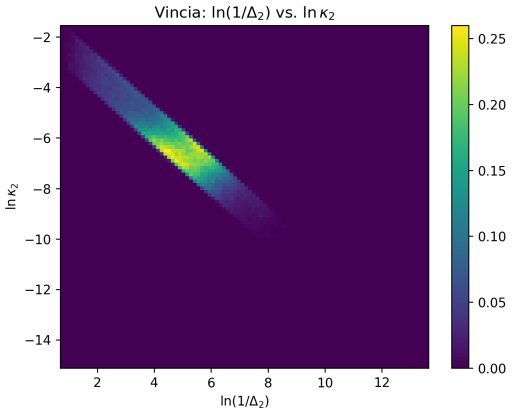



(c) Difference

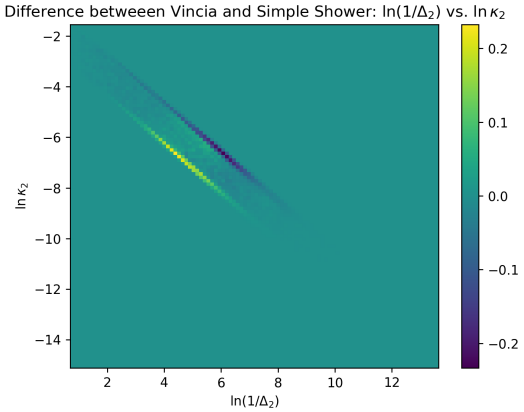
Figure 4.12: The Lund Plane for the first declustering at 14 TeV events. See table 4.2. a) Generated with Simple Shower; b) Generated with Vincia; c) Difference in density between both models.




(a) Simple Shower




(b) Vincia




(c) Difference

Figure 4.13: The Lund Plane for the first declustering at 14 TeV events. See table 4.2. a) Generated with Simple Shower; b) Generated with Vincia; c) Difference in density between both models.

## 4.3 100 TeV events

Though, as shown in the previous sections, the Lund planes show some slight differences between the parton shower models, the neural network hasn't been able to distinguish between the two models with the settings tested.
Although no optimization has been done up to this point, the low accuracy reveals at least that similar jet substructures are produced, at least at these energies and along the leading jet. Therefore, both models are consistent at least in these conditions, which is to be expected.
For much higher energies, however, this consistency is not guaranteed. Higher energy partons will radiate more, and therefore the different parton showers could potentially create more distinct jets and sub-jets, as well as their corresponding observables.

Continuing along the previous section, a set of ideal settings will be searched before optimizing the neural network.

The center-of-mass energy was at this point increased from 14 to 100 TeV. Kinematic cuts on the jets have remained the same, as well as the jet algorithm and radius, as shown in table 4.3, and the $p_T$ distribution seen in 4.14. This energy was picked because it will be the maximum energy allowed by the Future Circular Collider (FCC) [38] if/when it gets built. No other changes have been made, meaning we are still working with R = 0.5 anti-kt jets, reclustered with R = 1 C/A algorithm.

Similarly to what was done for lower energies, the histograms in figures 4.15 and 4.16a show the distributions of $\Delta$ and $\kappa$ at this energy, and can indicate whether the models become more distinct.

As espected, the effects of the different parton shower mechanisms become more evident at higher energies, both in the first and second declusterings of the jet. The peak associated with ISR, around $\ln \kappa = -3, \ln(1/\Delta) = 1$, now becomes the greatest peak in the distribution. However, in this region the two models are more consistent with one another, indicating that their behaviors for ISR are quite similar.

A more illustrating image can be reached through the use of the Lund planes, as shown in figures 4.17 and 4.18. Similarly to the distributions we had seen for the 14TeV example (figures 4.12 and 4.13), disparities between the two models tend to show differences in the diagonal lines of z = 0.1 and z = 0.5, as well as some smaller differences around the hadronization region.

After having made these considerations, and given the fact that the 100 TeV case appears to show a greater chance of success than the lower energy cases, the conditions are set up to input these events into the neural network. Once again, so far no changes have been done to

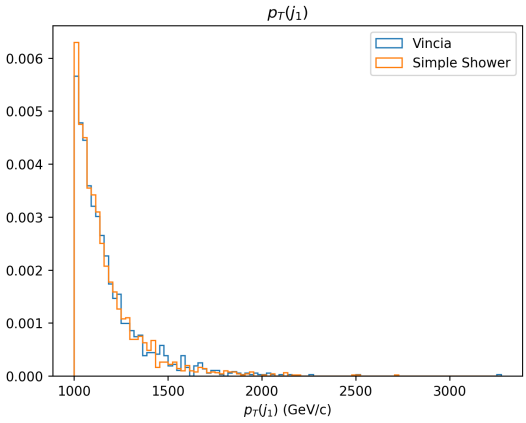
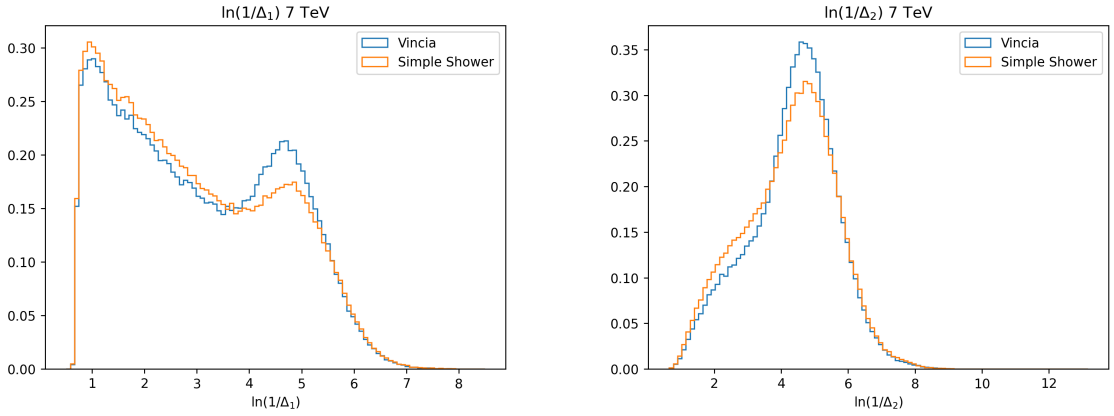


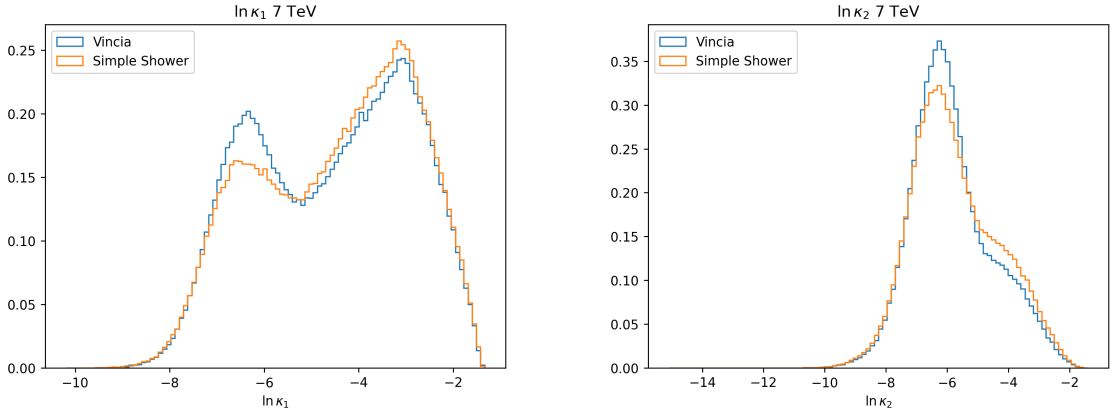

Figure 4.14: Distribution of the leading jet's momentum for the 14 TeV event with a $p_t > 1$ TeV/c cut.




(a) 1st declustering

(b) 2nd declustering

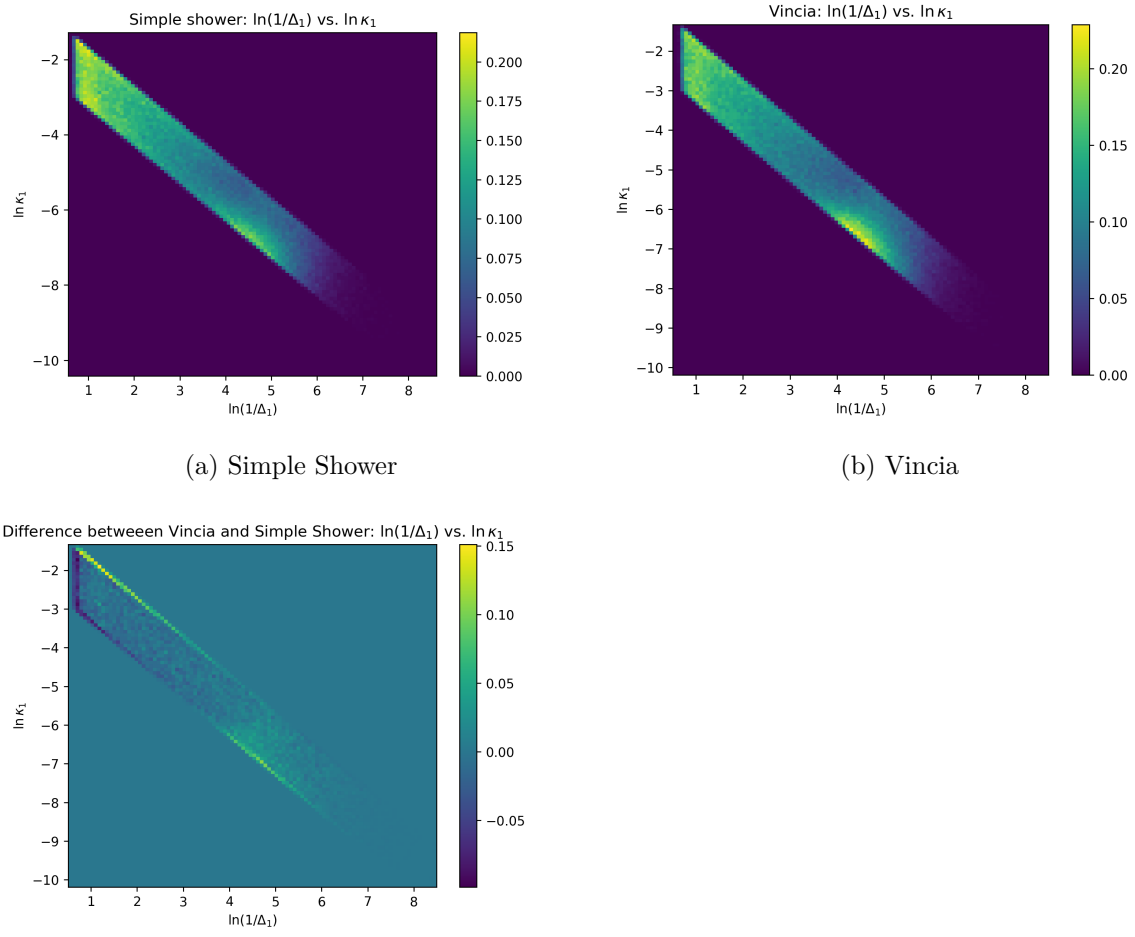
Figure 4.15: Histograms for the angular separation $\Delta$ between two sub-jets, for the leading branch of at a center of mass energy $\sqrt{s} = 100$TeV, with anti-$k_t$ jets (R = 0.5 $p_t > 1TeV$) reclustered with the C/A algorithm (R = 1). See table 4.3.




(a) 1st declustering

(b) 2nd declustering

Figure 4.16: Histograms for $\kappa$ for 100 TeV events. See table 4.3.

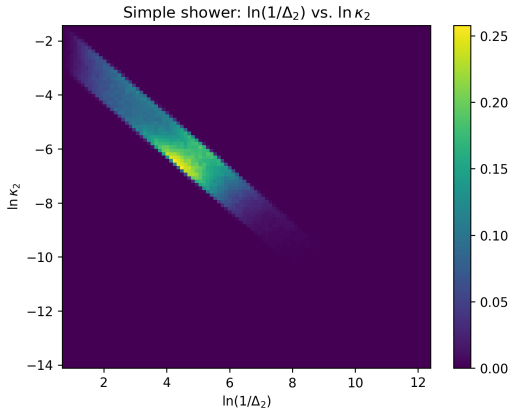



(a) Simple Shower

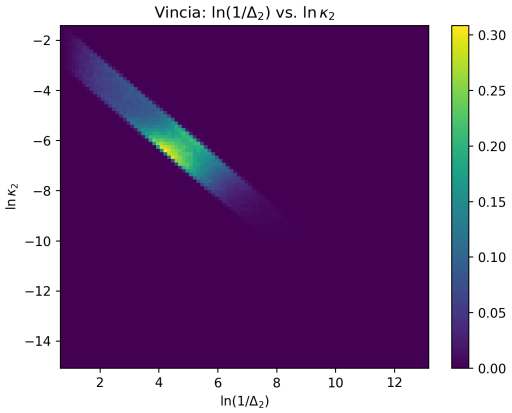
(b) Vincia

(c) Difference

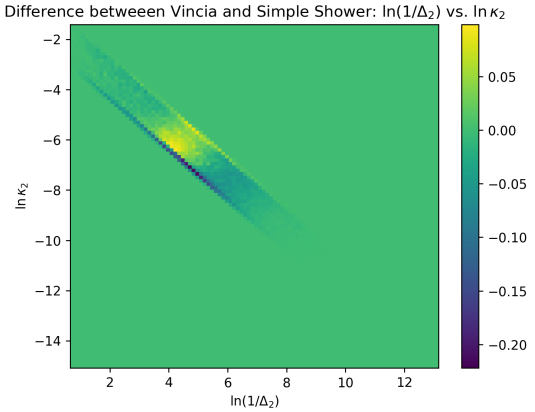
Figure 4.17: The Lund Plane for the first declustering, in a 100 TeV event (table 4.3). a) Generated with Simple Shower; b) Generated with Vincia; c) Difference in density between both models.




(a) Simple Shower




(b) Vincia




(c) Difference

Figure 4.18: The Lund Plane for the first declustering, in a 100 TeV event (table 4.3). a) Generated with Simple Shower; b) Generated with Vincia; c) Difference in density between both models

Table 4.3: A summary of settings used for the first test at 100TeV, and of the neural network trained to compare parton showers.

| **Name** | 100TeV-1 |
|---|---|
| **Event and jet settings** | |
| Branch | Leading branch of leading jet |
| $\sqrt{s}$ | 100 TeV |
| Jet algorithm | Anti-$k_t$ |
| Jet radius | 0.5 |
| Jet reclustering algorithm | C/A |
| Sub-jet radius | 1 |
| Jet $p_t$ | $>$ 1 TeV/c |
| Jet $\|\eta\|$ | $<$ 2.0 |
| Events generated | 1M Vincia + 1M simple |
| # of declusterings used | 2 |
| % of events kept after cuts | 32.8% (Vincia); 39.1% (simple) |
| **Neural network** | |
| Number of layers | 3 |
| Neurons per layer | 4 |
| Activation function | ReLU |
| Optimizer | Adam |
| Learning rate | 0.01 |

the network itself, except that the number of epochs of training was reduced from 30 to 10.

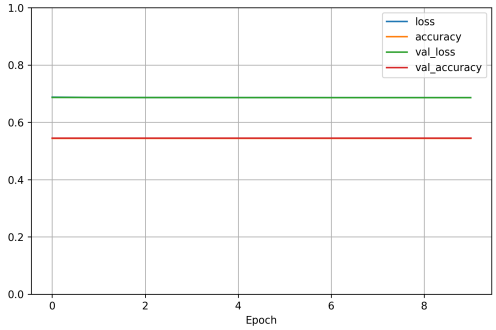
As shown in figure 4.19, in spite of the increased differences virtually no improvement was achieved in terms of network performance. The ROC slightly improved compared with the 7 and 14 TeV cases, but still is just barely above a random guess. The output of the network has indistinguishable curves, having 2 peaks (one around 0.41 and the other around 0.50) which may just be abnormal behavior by the algorithm.

Having the network failed in spite of the increased differences, some assumptions have to be re-evaluated. Optimization has already been mentioned, but going farther down the jet substructure. So far, only the first 2 splittings have been analyzed for 2 reasons. One is that the models become more similar to each other when progressing along the substructure, as shown in figures 4.20 and 4.21 for the 100 TeV events. The other reason is that by increasing the number of splittings, the number of samples that an algorithm uses a lot more computation time and, above all, it needs exponentially more samples in order to make a similar prediction. This phenomenon is called the curse of dimensionality [3].

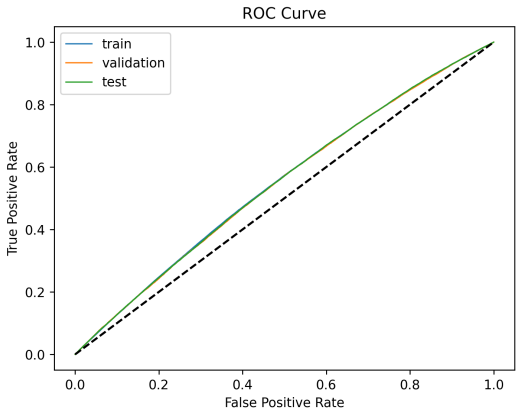
With no improvement in performance at this point, however, a new attempt can be done by using the first 4 splittings and the corresponding observables $\Delta_i, \kappa_i$ (in figures 4.20, 4.21, and Lund planes in figure 4.22. Additionally, this neural network goes from 4 to 8 neurons in each layer (see table 4.4).

Inputting these new sub-jets into the neural network, the output is as shown in figure 4.23.

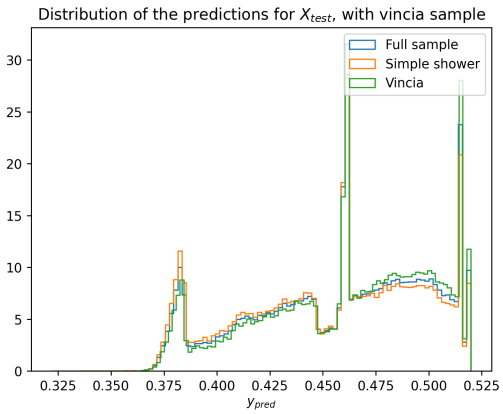
The inclusion of the 3rd and 4th splittings along the leading branch has, however, not created a decrease in the already poor performance, in spite of the increase in the number of dimensions.




(a) Evolution of the accuracy and loss function

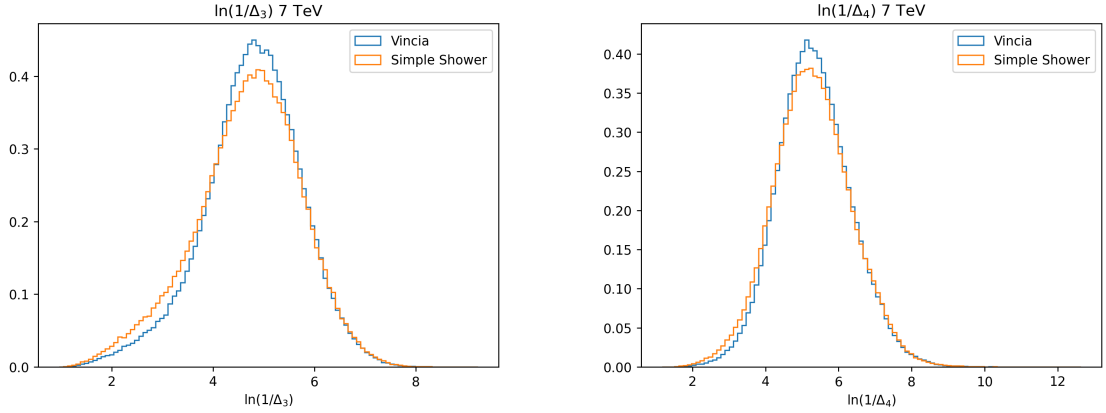



(b) ROC Curve




(c) Network output for the test set

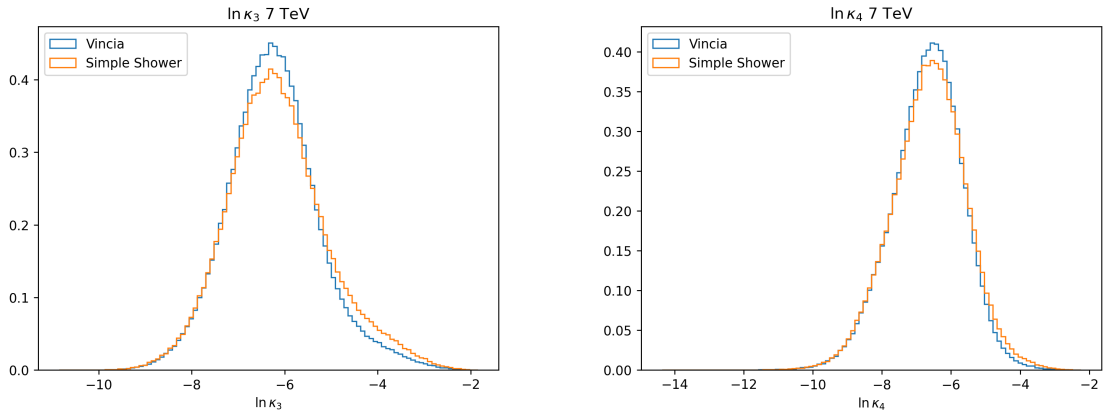
Figure 4.19: Evaluation of the neural network for the 100 TeV event, along the leading branch, using the Lund plane variables for the first 2 declusterings as inputs.




(a) 3rd declustering

(b) 4th declustering

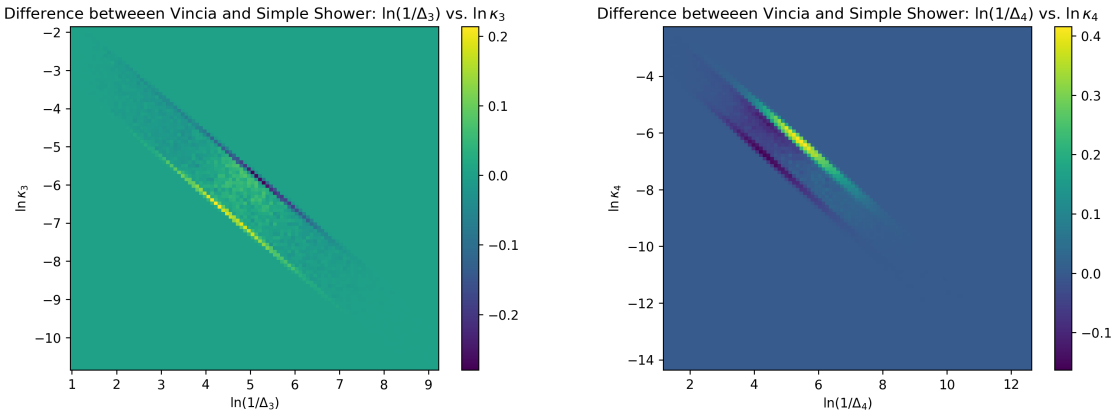
Figure 4.20: Histograms for the angular separation $\Delta$ between two sub-jets, for the leading branch of at a center of mass energy $\sqrt{s}$ = 100TeV, with anti-$k_t$ jets (R = 0.5 $p_t > 1TeV$) reclustered with the C/A algorithm (R = 1), for the 3rd and 4th declusterings.




(a) Distribution of $\kappa_1 = p_{t2}\Delta_1$ for an event at 100 TeV CM energy (see directory for settings).

(b) Distribution of $\kappa_1 = p_{t2}\Delta_1$ for an event at 100 TeV CM energy (see directory for settings).

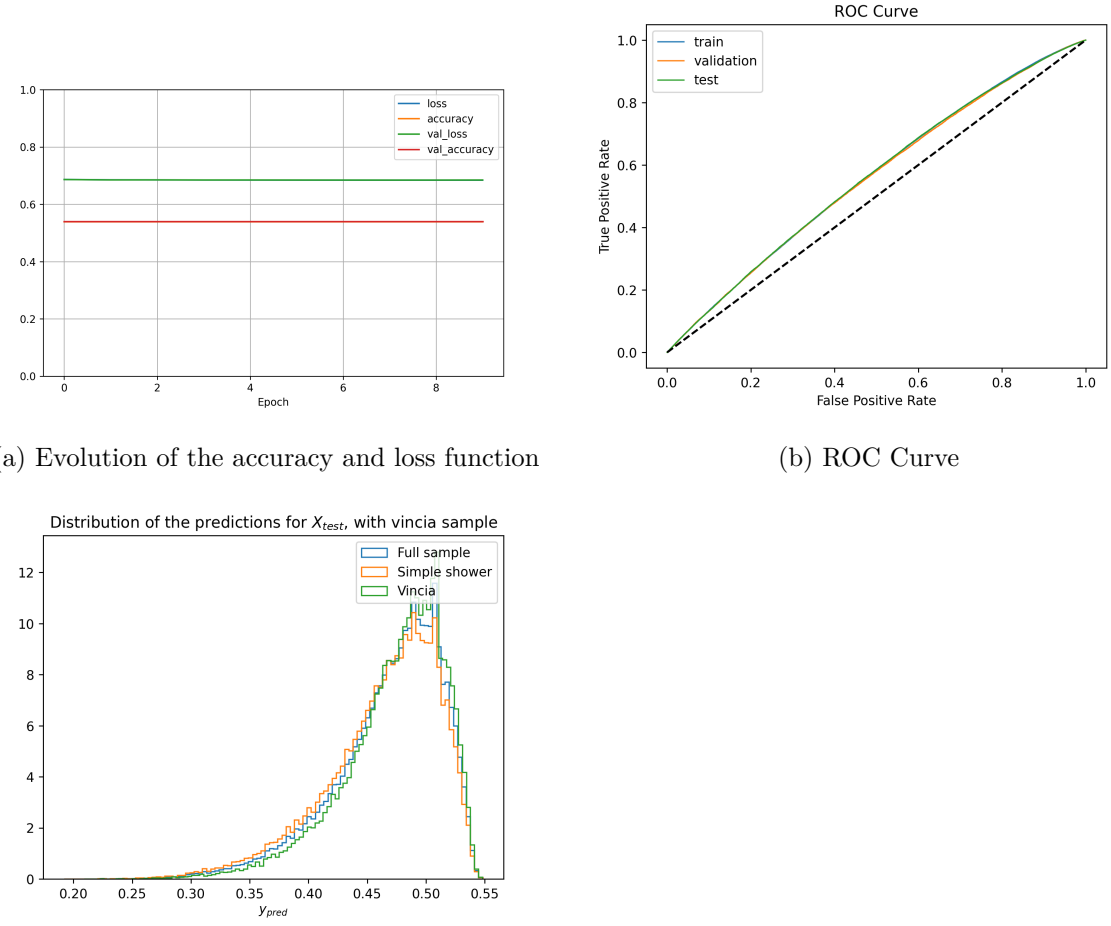
Figure 4.21: Histograms for $\kappa$, for the same jet substructure as that in figure 4.20.




(a) 3rd Lund Plane

(b) 4th Lund Plane

Figure 4.22: The 3rd and 4th Lund planes for the leading branch 100 TeV event (setting 100 TeV-2).




(a) Evolution of the accuracy and loss function

(b) ROC Curve

(c) Network output for the test set

Figure 4.23: Evaluation of the neural network for the 100 TeV event, along the leading branch, using the Lund plane variables for the first 4 declusterings as inputs.

Table 4.4: A summary of settings used for the second test at 100TeV, using 4 declusterings as input, and of the neural network trained to compare parton showers.

| **Name** | 100TeV-2 |
|---|---|
| **Event and jet settings** | |
| Branch | Leading branch of leading jet |
| $\sqrt{s}$ | 100 TeV |
| Jet algorithm | Anti-$k_t$ |
| Jet radius | 0.5 |
| Jet reclustering algorithm | C/A |
| Sub-jet radius | 1 |
| Jet $p_t$ | $>$ 1 TeV/c |
| Jet $\|\eta\|$ | $<$ 2.0 |
| Events generated | 1M Vincia + 1M simple |
| # of declusterings used | 4 |
| % of events kept after cuts | 27.8% (Vincia); 32.4% (simple) |
| **Neural network** | |
| Number of layers | 3 |
| Neurons per layer | 8 |
| Activation function | ReLU |
| Optimizer | Adam |
| Learning rate | 0.01 |

On the contrary, and likely because increases in the Lund plane's density become greater with the reduction of phase space, there is actually a very small increase in performance. The output graph now has a slightly higher peak for the Vincia model at slightly above 0.5. There is little reason to conclude that adding these extra sub-jets is necessary in constructing a neural network that succeeds in distinguishing parton showers.

### 4.3.1 Neural Network Optimization

The increase in energy has brought some increased differences between models, but accuracy has remained around 55% nonetheless. This was, however, done with a number of neurons and layers chosen for no particular reason. In machine learning, it's often necessary to fine-tune the algorithms' parameters (called hyperparameters), with various techniques being developed for it [39]. In the case of a neural network, one hyperparameter that can be tuned is the number of layers and the number of neurons in each layer.

In order to find this ideal number of neurons and layers, the package Optuna was used [40]. Optuna is a hyperparameter optimization framework, which allows for the automation of hyperparameter searches. It searched within the a phase-space of 1 to 3 layers with 1 to 128 neurons each. It then outputs the accuracy of the best network, as well as the number of neurons and layers.

Using the same data preprocessing as previously (standard scaling, and 64%, 16%, 20% split between training, validation, and test set), an Optuna study was created, consisting of 20 trials. The optimizer used was Adam, at a learning rate of 0.01. Running this program, using as input

the jet substructure observables of the previous simulation (100 TeV-2, as can be consulted in table 4.4) gave the following output:

1. Accuracy (validation set): 0.547
2. Configuration: 2 layers. 1st layer with 125 neurons; 2nd layer with 59 neurons.

This is a clearly unsatisfactory result, with performance being virtually identical to the one obtained before any optimization and again comparable to a random guess.

In the next sections, we take a look at other branches of the jet substructure tree and configurations, looking for cases where the parton shower models may differ even more, and once again training a neural network with that new data instead.

## 4.4 Increased radius and the τ algorithm

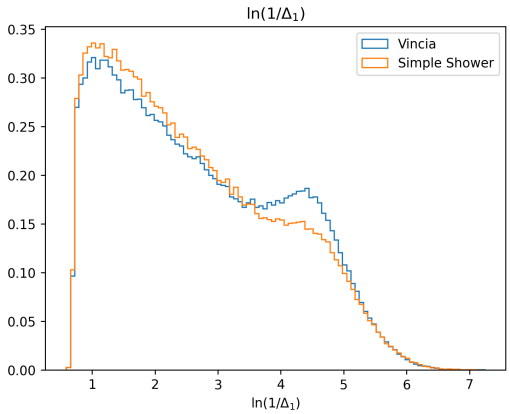
### 4.4.1 The τ algorithm

Another possible method for finding distinctions can come from changing the jet algorithm. The $\tau$ algorithm [41], having some energy dependency, may suffer effects of the different parton shower mechanisms slightly differently. As such, a new test was made, using 100 TeV events but with jets reclustered with the $\tau$ algorithm instead of C/A, as shown in table 4.5. The distributions of $\Delta$, $\kappa$ can be seen in figures 4.24 and 4.25, respectively, with the Lund planes seen in 4.5.

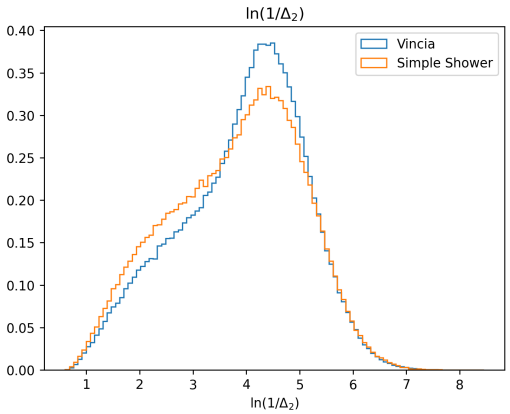
It appears that there can be indeed some advantages related to taking a larger radius. While for the first declustering we don't see greater distinctions than for the R = 0.5 case, there might be some very slight improvement for subsequent declusterings, as seen in figures 4.24 and 4.25. In the Lund Planes (figure 4.26), it's also possible to observe that differences aren't located in such thin strips (around $z = 0.1$ and $z = 0.5$) and can now be observed in a larger region in the Lund plane. That being said, by using the same neural network, results were indistinct from the previous cases.

### 4.4.2 Radius 1

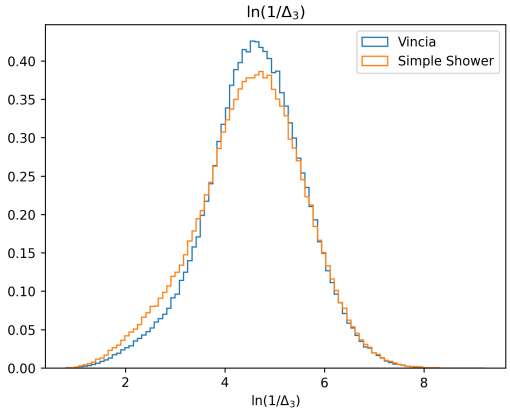
Only jets with radius R = 0.5 have been considered up to this point. This is because, as mentioned previously in Chapter 2, jets should have a jet radius that doesn't exclude too many final state particles (which would result in the identification of a large number of jets), but they shouldn't include too much background radiation.



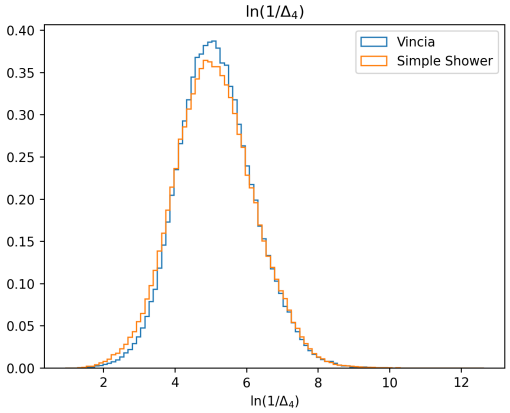

(a) 1st declustering




(b) 2nd declustering

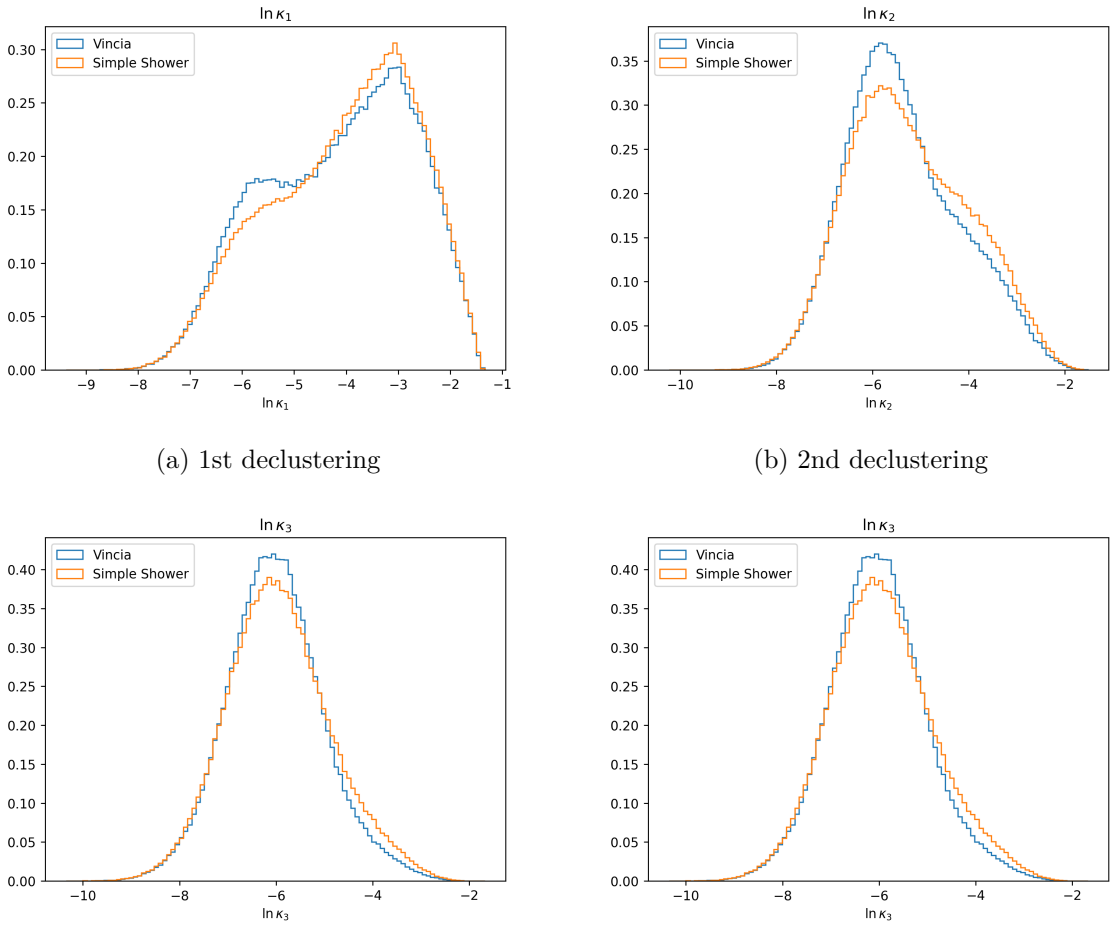



(c) 3rd declustering




(d) 4th declustering

Figure 4.24: Distributions of $\Delta$ for jets reclustered with the $\tau$ algorithm. See table 4.5.



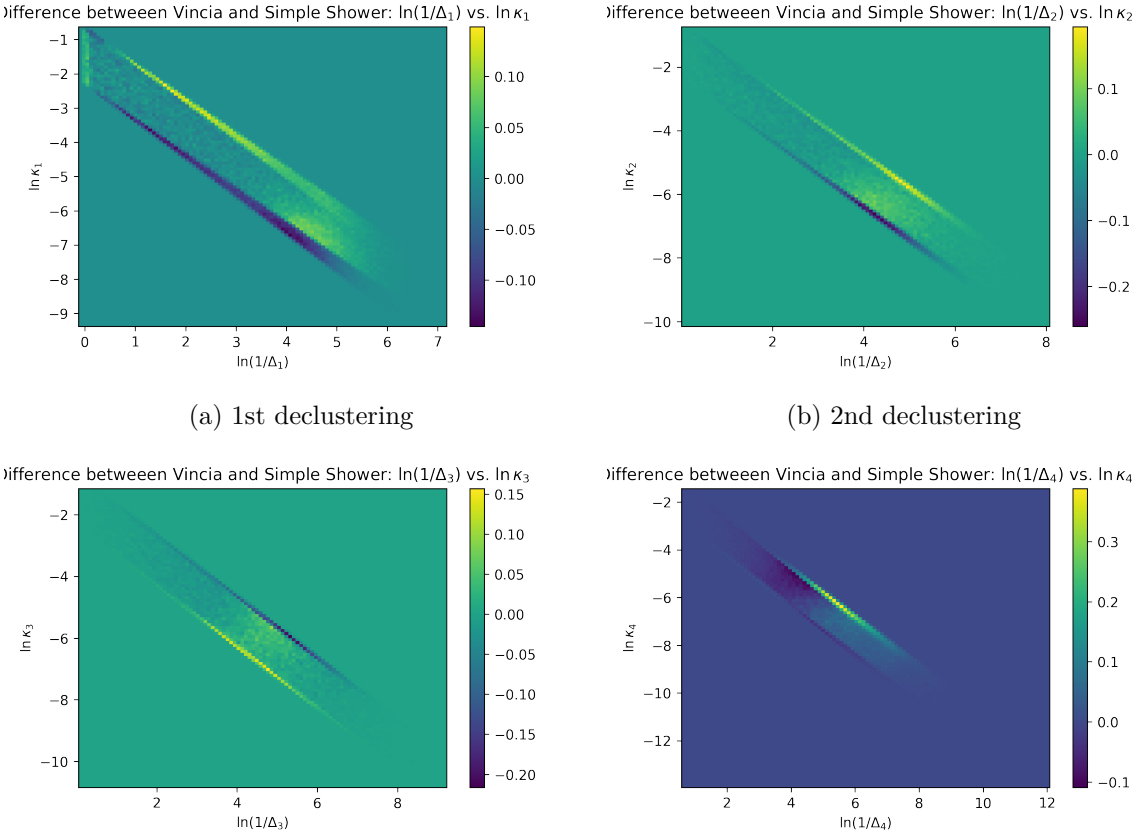

(a) 1st declustering

(b) 2nd declustering

(c) 3rd declustering

(d) 4th declustering

Figure 4.25: Distributions of $\kappa$ for jets reclustered with the $\tau$ algorithm. See table 4.5.




(a) 1st declustering

(b) 2nd declustering

(c) 3rd declustering

(d) 4th declustering

Figure 4.26: Lund planes for the difference in density for jets reclustered with the $\tau$ algorithm. See table 4.5.

Table 4.5: A summary of settings used for the second test at 100TeV, with jets reclustered with the $\tau$ and of the neural network trained to compare parton showers.

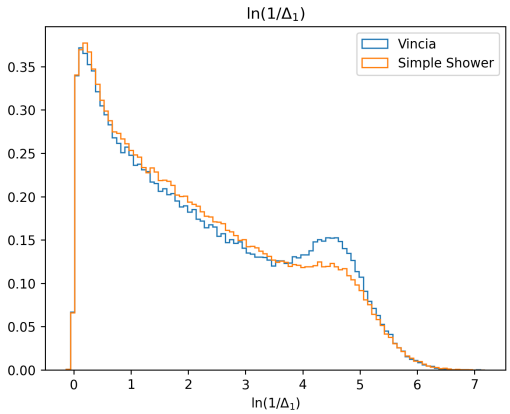
| Name | 100TeV-4 |
|---|---|
| Event and jet settings | |
| Branch | Leading branch of leading jet |
| $\sqrt{s}$ | 100 TeV |
| Jet algorithm | Anti-$k_t$ |
| Jet radius | 0.5 |
| Jet reclustering algorithm | $\tau$ |
| Sub-jet radius | 1 |
| Jet $p_t$ | $>$ 1 TeV/c |
| Jet $\eta$ | $<$ 2.0 |
| Events generated | 1M Vincia + 1M simple |
| # of declusterings used | 2 |
| % of events kept after cuts | 28.7% (Vincia); 33.6% (simple) |
| Neural network | |
| Number of layers | 3 |
| Neurons per layer | 8 |
| Activation function | ReLU |
| Optimizer | Adam |
| Learning rate | 0.01 |

One possibility, however, is that the parton showers are increasingly different at higher angles $\Delta$. The reasoning behind this would be that Vincia's $2 \rightarrow 3$ splittings could result in one particle being emitted to greater angles than what would happen with the Simple Shower model. By including that particle in the final jet, differences in the sub-jet observables might become apparent with a larger jet, which wouldn't be noticeable otherwise.

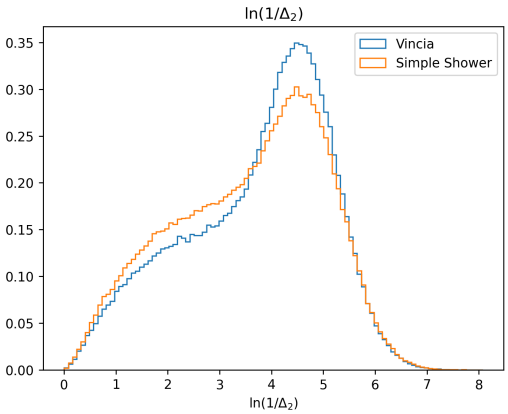
The jet radius was here increased from 0.5 to 1. The radius of the sub-jets is consequently also increased, from 1 to 1.5, as can be seen in table 4.6.

Taking in account the fact that the leading branch, considering 4 declusterings, appears to show larger differences between parton showers, tests are done following that branch. All other settings are kept the same, namely the center-of-mass energy stays at 100 TeV, the jet's $p_T$ cut at 1 TeV, and the soft drop condition at $z_{cut} = 0.1, \beta = 0$, with C/A reclustered anti-$k_t$ jets.

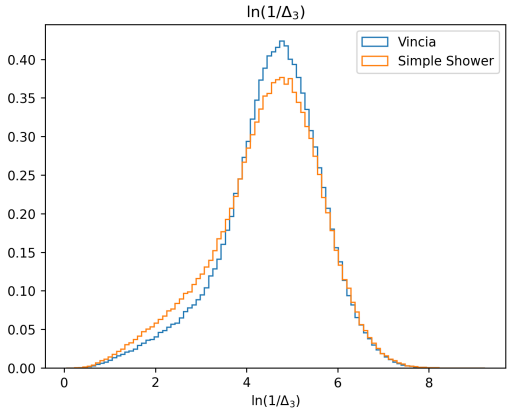
It appears that there can be indeed some advantages related to taking a larger radius. While for the first declustering we don't see greater distinctions than for the R = 0.5 case, there might be some very slight improvement for subsequent declusterings, as seen in figures 4.24 and 4.25. In the Lund Planes (figure 4.26), it's also possible to observe that differences aren't located in such thin strips (around $z = 0.1$ and $z = 0.5$) and can now be observed in a larger region in the Lund plane. That being said, by using the same neural network, results were indistinct from the previous cases.



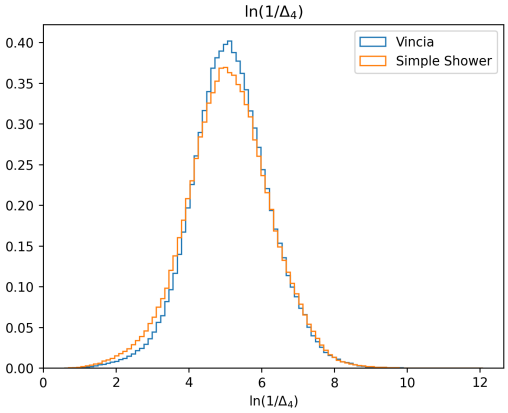

(a) 1st declustering




(b) 2nd declustering

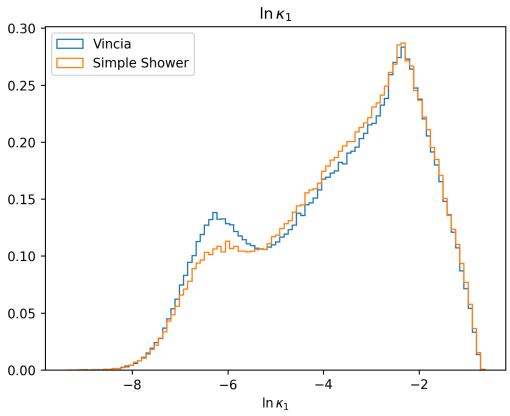



(c) 3rd declustering

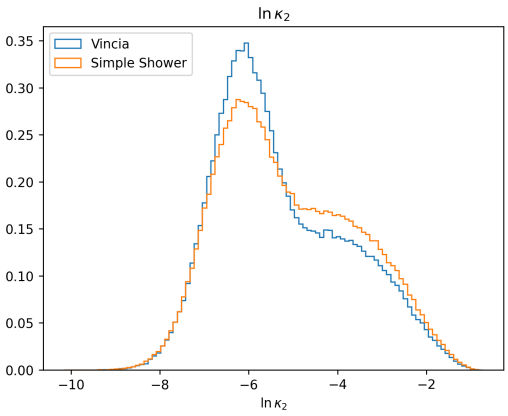



(d) 4th declustering

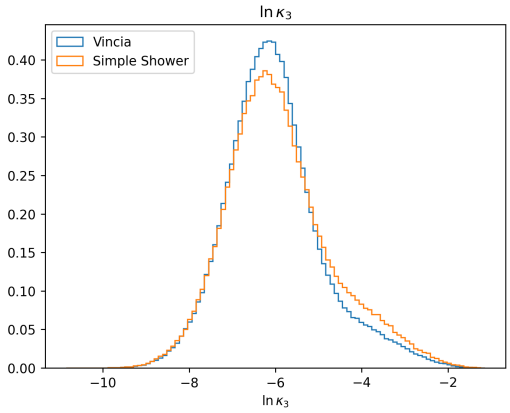
Figure 4.27: Distributions of $\Delta$ along the leading branch when using jet radius $R = 1$. See table 4.6.



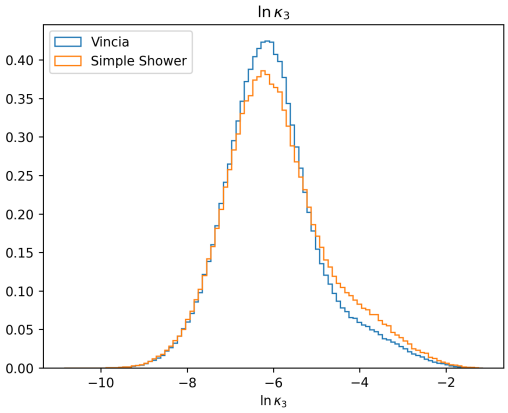

(a) 1st declustering




(b) 2nd declustering

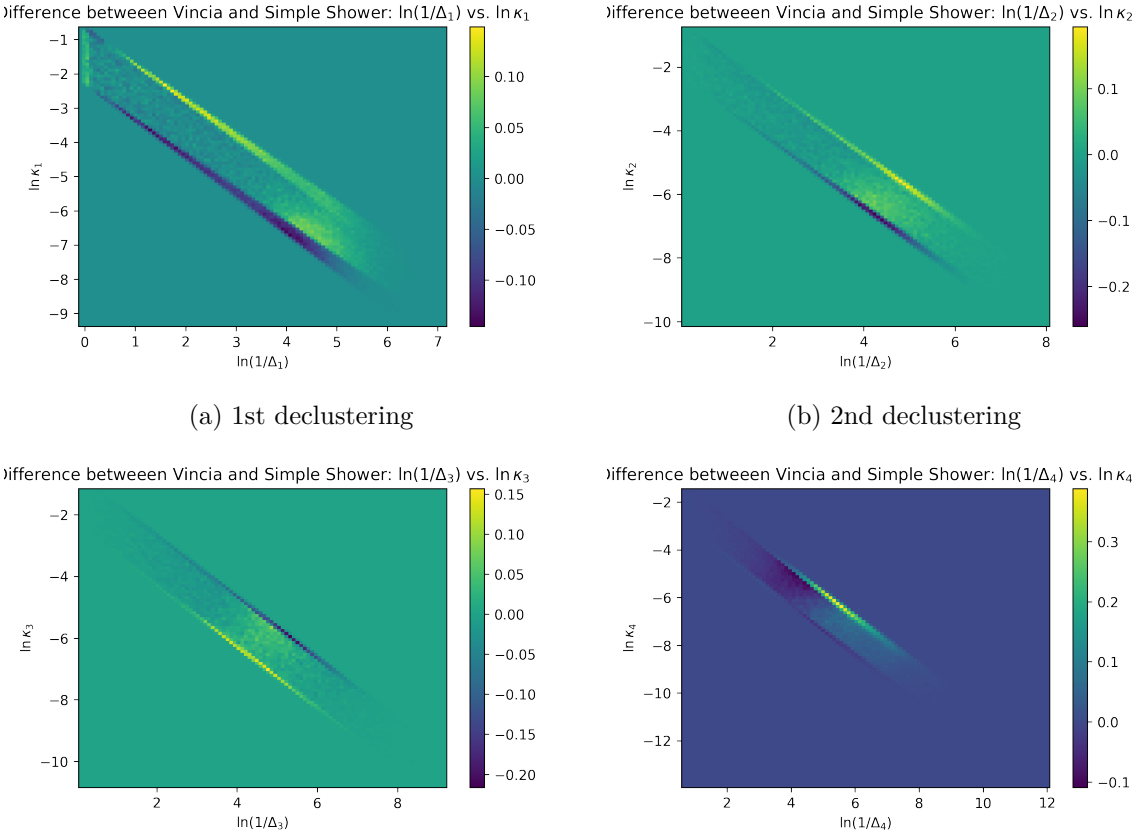



(c) 3rd declustering




(d) 4th declustering

Figure 4.28: Distributions of $\kappa$ along the leading branch when using jet radius $R = 1$. See table 4.6.




(a) 1st declustering

(b) 2nd declustering

(c) 3rd declustering

(d) 4th declustering

Figure 4.29: Lund planes for the difference in density along the leading branch, for 100 TeV events, with radius $R = 1$. See table 4.6.

Table 4.6: A summary of settings used for the second test at 100TeV, with a jet radius R = 1 and of the neural network trained to compare parton showers.

| | |
|---|---|
| Name | 100TeV-5 |
| Event and jet settings | |
| Branch | Leading branch of leading jet |
| $\sqrt{s}$ | 100 TeV |
| Jet algorithm | Anti-$k_t$ |
| Jet radius | 1 |
| Jet reclustering algorithm | C/A |
| Sub-jet radius | 1.5 |
| Jet $p_t$ | $>$ 1 TeV/c |
| Jet $\|\eta\|$ | $<$ 2.0 |
| Events generated | 1M Vincia + 1M simple |
| # of declusterings used | 4 |
| % of events kept after cuts | 33.8 %(Vincia); 39.4% (simple) |
| Neural network | |
| # of declusterings used | 4 |
| Number of layers | 3 |
| Neurons per layer | 8 |
| Activation function | ReLU |
| Optimizer | Adam |
| Learning rate | 0.01 |

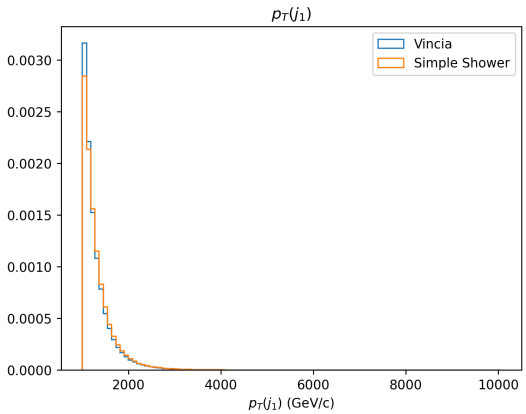
## 4.5 The subleading jet

Finally, instead of looking at the leading jet, we observe the sub-leading one, and look at the leading branch of that jet. These softer jets should suffer from more ISR effects than FSR, but it can be useful to compare them.

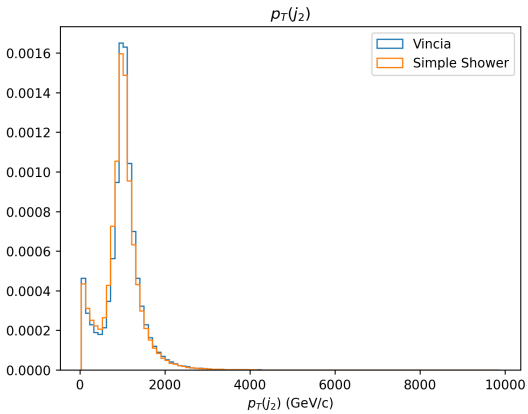
This comes with added care when it comes to selecting jets. Up until this point, kinematic cuts were only applied to the leading jet. It's worth remembering those set in order to identify jets which corresponded to a hard scattering, and therefore excluding events where the products of a hard parton were scattered between multiple jets.

When looking at the sub-leading jet, i.e. the jet with 2nd largest transverse momentum, we must ensure that similar events are being selected, meaning that $p_t$ cuts on both the leading and sub-leading jet has to be applied. As such, the 1 TeV $p_t$ cut for the leading jet is kept, and a 500 GeV cut for the subleading jet is applied simultaneously. In order to ensure the same hard scattering condition, we should finally ensure that the angular separation between the two jets is ensured. As such, a cut in the angle between the two jets, $\phi$, is also applied, with $\phi > \frac{2\pi}{3}$. These conditions are visualized in figure 4.30, with the remaining settings in table 4.7.

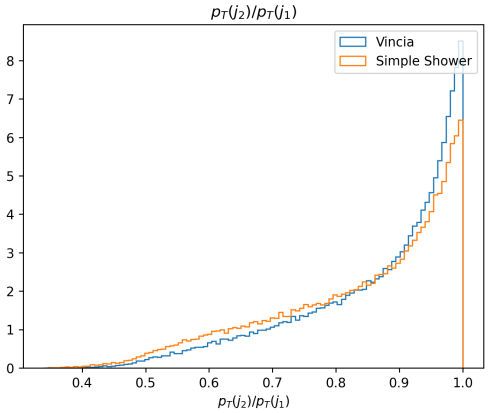
In figure 4.31, we can see the Lund plane for events generated with the Simple Shower model. Like predicted, in the first declustering there is a large prevalence of large angle radiation. Moving along the leading branch, we see a decrease towards lower angles but, similarly to the parton level test, emissions occur nearer to the $z = 0.5$ line.



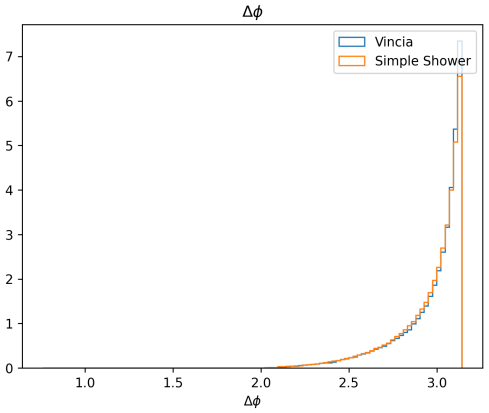

(a) Leading jet $p_t$




(b) Sub-leading jet $p_t$

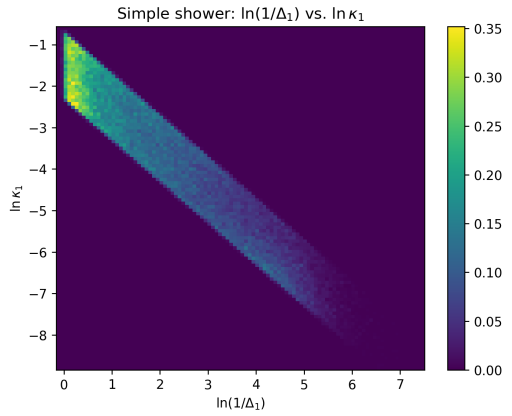



(c) Assymetry

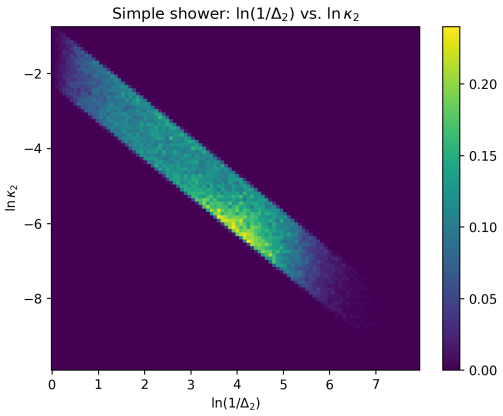



(d) Angle between jets

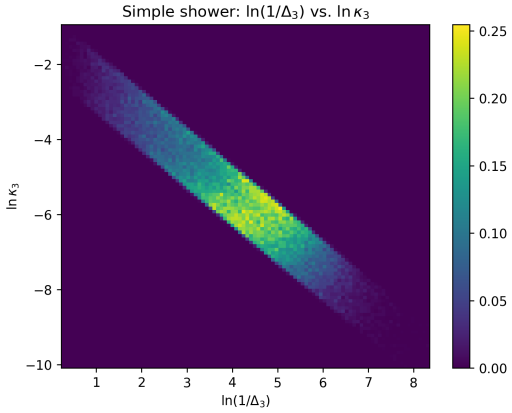
Figure 4.30: The kinematic cuts in the sub-leading jet and the assymetry between jets.



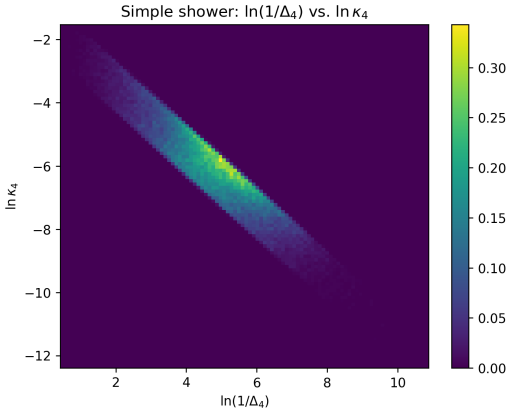

(a) 1st declustering




(b) 2nd declustering




(c) 3rd declustering




(d) 4th declustering

Figure 4.31: Lund planes for the difference in density along the leading branch of the subleading jet, for 100 TeV events, with radius $R = 1$

Table 4.7: List of settings for 100 TeV events generated along the leading jet of the sub-leading jet

| | |
|---|---|
| Name | 100TeV-subleading |
| Event and jet settings | |
| Branch | Leading branch of leading jet |
| $\sqrt{s}$ | 100 TeV |
| Jet algorithm | Anti-$k_t$ |
| Jet radius | 1 |
| Jet reclustering algorithm | C/A |
| Sub-jet radius | 1.5 |
| Jet $p_t$ | $>$ 1 TeV/c |
| Jet $\eta$ | $<$ 2.0 |
| Events generated | 1M Vincia + 1M simple |
| # of declusterings used | 4 |
| % of events kept after cuts | 33.8 %(Vincia); 39.4% (simple) |
| Neural network | |
| Number of layers | 3 |
| Neurons per layer | 8 |
| Activation function | ReLU |
| Optimizer | Adam |
| Learning rate | 0.01 |

The use of these substructure observables in the neural network brought no improvement either.

# 5 Conclusions

In this work, we looked at the differences between two different parton shower models implemented in PYTHIA 8.3 in the substructure of jets created by proton-proton collisions and Hard QCD partons. Jets were identified with the anti$k_t$ algorithm and reclustered with the C/A algorithm; events were generated with varying center of mass energies between 7 and 100 TeV.

We started by finding observables related with the effects of the parton shower and which showed discrepancies between the simple shower and Vincia models. It was shown that some differences exist when mapping the Lund planes and when looking at the histograms of the respective observables, $\ln(1/\Delta_i), \ln \kappa_i$. These discrepancies are increased at higher energies and when following the leading branch, that is, the branch with highest transverse momentum at each stage of the declustering of the jet, both for the leading and subleading jets. It was also shown that for other observables such as the jet splitting function $z$ there is no observable difference between the two models.

Having seen the points of discrepancy, a neural network was set up to classify unknown events according to the model which was used to generate them. However, in spite of attempts to optimize the network and finding conditions where disparities were more apparent, performance has barely been better than the tossing of a coin.

Given the visible distinction that can be done between the models, we believe that a working model can be developed, with the use of a more sophisticated neural network.

# Bibliography


[1] G. P. Salam, "Towards jetography," *The European Physical Journal C* **67** no. 3-4, (May, 2010) 637–686. https://doi.org/s10052-010-1314-6.

[2] F. A. Dreyer, G. P. Salam, and G. Soyez, "The Lund Jet Plane," *JHEP* **12** (2018) 064, arXiv:1807.04758 [hep-ph].

[3] A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* O'Reilly Media, Inc., 2nd ed., 2019.

[4] H. Bui, "Roc curve transforms the way we look at a classification problem." Mar, 2020. https://towardsdatascience.com/a-simple-explanation-of-the-roc-curve-and-auc-64db32d75541.

[5] H. Bui, Mar, 2020. https://www.theprojectspot.com/tutorial-post/introduction-to-artificial-neural-networks-part-1/7.

[6] M. E. Peskin and D. V. Schroeder, *An Introduction to Quantum Field Theory.* Westview Press, 1995. Reading, USA: Addison-Wesley (1995) 842 p.

[7] P. Skands, "Introduction to QCD," in *Searching for New Physics at Small and Large Scales.* WORLD SCIENTIFIC, Sep, 2013. https://doi.org/10.1142%2F9789814525220_0008.

[8] J. E. Huth *et al.*, "Toward a standardization of jet definitions," in *1990 DPF Summer Study on High-energy Physics: Research Directions for the Decade (Snowmass 90)*, pp. 0134–136. 12, 1990.

[9] S. Moretti, L. Lönnblad, and T. Sjöstrand, "New and old jet clustering algorithms for electron-positron events," *Journal of High Energy Physics* **1998** no. 08, (Aug, 1998) 001–001. https://doi.org/10.1088%2F1126-6708%2F1998%2F08%2F001.

[10] S. Catani, Y. Dokshitzer, M. Seymour, and B. Webber, "Longitudinally-invariant $k_\perp$-clustering algorithms for hadron-hadron collisions," *Nuclear Physics B* **406** no. 1, (1993) 187–224. https://www.sciencedirect.com/science/article/pii/055032139390166M.

[11] L. Apolinário, A. Cordeiro, and K. Zapp, "Time reclustering for jet quenching studies," *Eur. Phys. J. C* **81** no. 6, (2021) 561, arXiv:2012.02199 [hep-ph].

[12] M. Cacciari, G. P. Salam, and G. Soyez, "The anti-$k_t$ jet clustering algorithm," *JHEP* **04** (2008) 063, arXiv:0802.1189 [hep-ph].

[13] Y. L. Dokshitzer, G. D. Leder, S. Moretti, and B. R. Webber, "Better jet clustering algorithms," *JHEP* **08** (1997) 001, arXiv:hep-ph/9707323.

[14] B. Andersson, G. Gustafson, L. Lonnblad, and U. Pettersson, "Coherence Effects in Deep Inelastic Scattering," *Z. Phys. C* **43** (1989) 625.

[15] A. J. Larkoski, "An Unorthodox Introduction to QCD," arXiv:1709.06195 [hep-ph].

[16] A. J. Larkoski, S. Marzani, G. Soyez, and J. Thaler, "Soft drop," *Journal of High Energy Physics* **2014** no. 5, (May, 2014) . https://doi.org/10.1007%2Fjhep05%282014%29146.

[17] A. J. Larkoski, S. Marzani, and J. Thaler, "Sudakov Safety in Perturbative QCD," *Phys. Rev. D* **91** no. 11, (2015) 111501, arXiv:1502.01719 [hep-ph].

[18] C. Bierlich *et al.*, "A comprehensive guide to the physics and usage of PYTHIA 8.3" arXiv:2203.11601 [hep-ph].

[19] M. Bahr *et al.*, "Herwig++ Physics and Manual," *Eur. Phys. J. C* **58** (2008) 639–707, arXiv:0803.0883 [hep-ph].

[20] **Sherpa** Collaboration, E. Bothmann *et al.*, "Event Generation with Sherpa 2.2" *SciPost Phys.* **7** no. 3, (2019) 034, arXiv:1905.09127 [hep-ph].

[21] T. Sjostrand, "PYTHIA 5.7 and JETSET 7.4: Physics and manual," arXiv:hep-ph/9508391.

[22] G. Altarelli and G. Parisi, "Asymptotic freedom in parton language," *Nuclear Physics B* **126** no. 2, (1977) 298–318. https://www.sciencedirect.com/science/article/pii/0550321377903844.

[23] T. Sjostrand and P. Z. Skands, "Transverse-momentum-ordered showers and interleaved multiple interactions," *Eur. Phys. J. C* **39** (2005) 129–154, arXiv:hep-ph/0408302.

[24] P. Skands, N. Fischer, S. Prestel, and M. Ritzmann, "The VINCIA Antenna Shower for Hadron Colliders," 9, 2016. arXiv:1609.07205 [hep-ph].

[25] L. Lönnblad, "Ariadne version 4 — a program for simulation of qdc cascades implementing the colour dipole model," *Computer Physics Communications* **71** no. 1, (1992) 15–31. https://www.sciencedirect.com/science/article/pii/001046559290068A.

[26] S. Höche and S. Prestel, "The midpoint between dipole and parton showers," *Eur. Phys. J. C* **75** no. 9, (2015) 461, arXiv:1506.05057 [hep-ph].

[27] T. M. Mitchell, *Machine learning*, vol. 1. McGraw-hill New York, 1997.

[28] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," *Advances in neural information processing systems* **31** (2018) .

[29] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv e-prints* (Dec., 2014) arXiv:1412.6980, arXiv:1412.6980 [cs.LG].

[30] T. Fawcett, "Introduction to roc analysis," *Pattern Recognition Letters* **27** (06, 2006) 861–874.

[31] J. Hanley and B. Mcneil, "The meaning and use of the area under a receiver operating characteristic (roc) curve," *Radiology* **143** (05, 1982) 29–36.

[32] F. A. Dreyer and H. Qu, "Jet tagging in the Lund plane with graph networks," *JHEP* **03** (2021) 052, arXiv:2012.08526 [hep-ph].

[33] M. Cacciari, G. P. Salam, and G. Soyez, "FastJet User Manual," *Eur. Phys. J. C* **72** (2012) 1896, arXiv:1111.6097 [hep-ph].

[34] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[35] M. Abadi, P. Barham, *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283. 2016.

[36] Z. Zhang and M. R. Sabuncu, "Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels," *arXiv e-prints* (May, 2018) arXiv:1805.07836, arXiv:1805.07836 [cs.LG].

[37] M. M. Ahsan, M. A. P. Mahmud, P. K. Saha, K. D. Gupta, and Z. Siddique, "Effect of data scaling methods on machine learning algorithms and model performance," *Technologies* **9** no. 3, (2021) . https://www.mdpi.com/2227-7080/9/3/52.

[38] G. Bernardi *et al.*, "The Future Circular Collider: a Summary for the US 2021 Snowmass Process," arXiv:2203.06520 [hep-ex].

[39] M. Feurer and F. Hutter, *Hyperparameter Optimization*, pp. 3–33. Springer International Publishing, Cham, 2019. https://doi.org/10.1007/978-3-030-05318-5_1.

[40] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631. 2019.

[41] H. A. Andrews *et al.*, "Novel tools and observables for jet physics in heavy-ion collisions," *J. Phys. G* **47** no. 6, (2020) 065102, arXiv:1808.03689 [hep-ph].

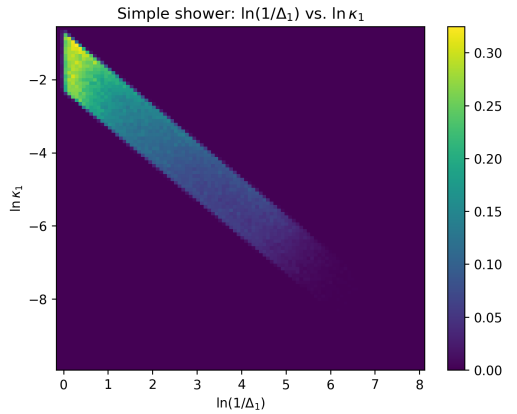
# Appendix A

# Events without hadronization

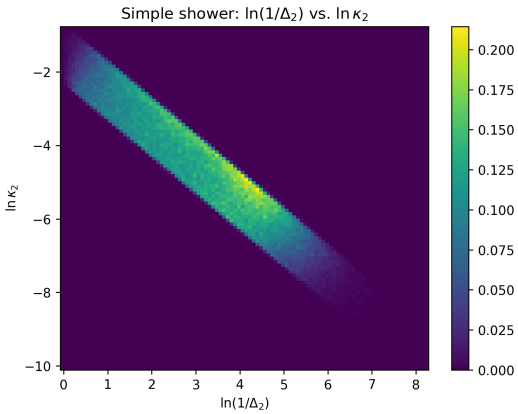
Over the course of this work, we repeatedly referenced the peaks created by the effects of hadronization in jet substructure, visualized in the Lund planes.

PYTHIA also allows toggling hadronization off, using (HadronLevel:all = off). One can see how the peak around $\ln(1/\Delta) = 5, \ln\kappa = -6$ corresponds to hadronization effects by looking at the distributions of events generated at parton level exclusively. Similarly to the previous section, we generate events with the settings in table 4.3, with hadronization toggled off. We plot here the Lund planes for both the Simple Shower and Vincia models for the first 4 declusterings, in figures A.1 and A.2.

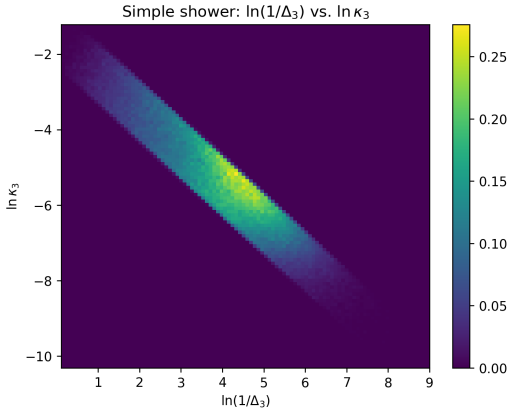
For the 1st declustering, we immediately see that the highest density region that we identified as associated with hadronization disappears. Moving into smaller and smaller sub-jets, we approach progressively smaller angles, nearing the aforementioned region. However, without hadronization emissions are much closer to the $z = 0.5$ line, whereas with hadronization they were close to the $z = 0.1$ line.



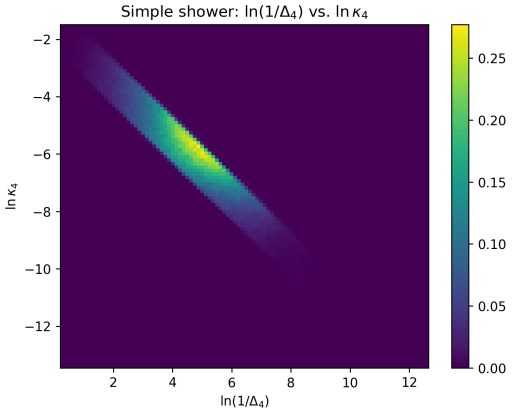

(a) 1st declustering




(b) 2nd declustering

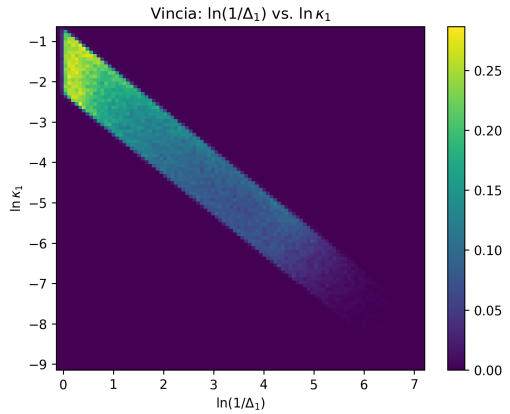



(c) 3rd declustering

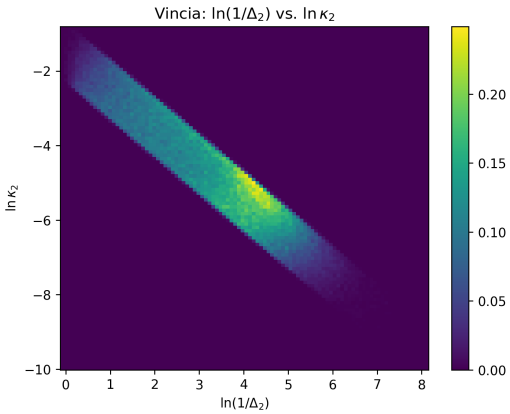



(d) 4th declustering

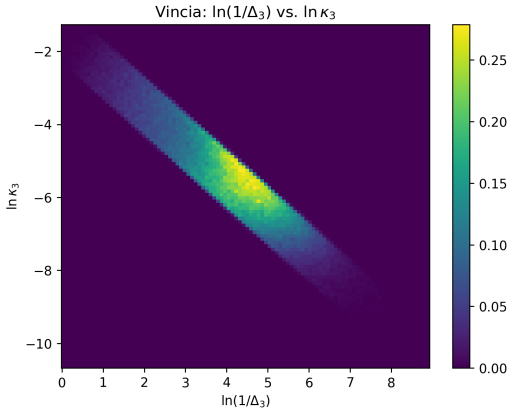
Figure A.1: Lund planes generated with the Simple Shower at parton level.



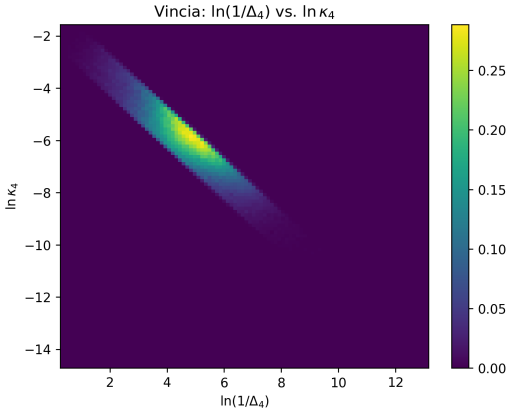

(a) 1st declustering




(b) 2nd declustering




(c) 3rd declustering




(d) 4th declustering

Figure A.2: Lund planes generated with Vincia at parton level.

# Appendix B

# The subleading branch

Up until this point, when progressing in the jet substructure, we always took the leading (highest $p_T$) sub-jet. This approach was picked because, following the hardest $p_T$ sub-jet implies looking at sub-jets produced by the most energetic partons. This is a branch that should be more dominated by wide-angle radiation, but it may be worth analyzing nonetheless.

One can, however, look at the secondary or subleading branch instead,as shown on figure B.1, producing the so-called secondary Lund plane. On the first iteration of the splitting the softer sub-jet is picked to continue declustering, and after that the hardest branch is always picked.

For the sake of simplicity, the same nomenclature is kept. This means that we'll keep using the reference for the tuples, $\mathcal{T}_1, \mathcal{T}_2, ...$, but they refer to another branch of the tree created by the jet substructure.

The same process is applied as for the previous cases: first analyzing the histograms (figures B.2 and B.3) and the Lund planes (B.4, and then looking at the neural network output. Following the same approach as for the leading branch, we started by comparing the histograms and the Lund planes. After that, the neural network was trained with these new events.

Before making any further analysis, it's worth looking at the histograms related with the first declustering: at a first glance, it looks like it should be similar to that obtained in the leading branch. After all, those observables relate to the first two sub-jets, so there shouldn't be any differences. However, we are only looking at jets that have gone through 4 declusterings, at least, along the branch we choose.

Looking at the histograms in figures B.2 and B.3, we notice smaller discrepancies between the two models following this branch. It is also worth noting that, for the first declustering, the distribution is now heavily dominated by the ISR region.

Since optimization didn't show any improvement, the same neural network of 3 layers with

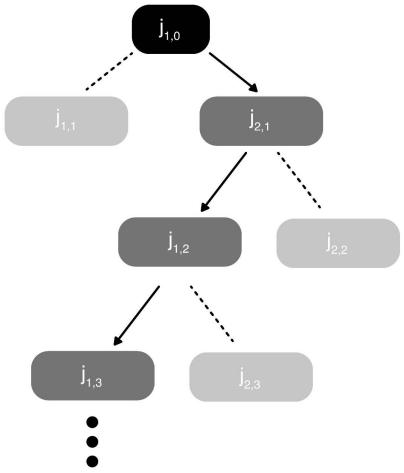
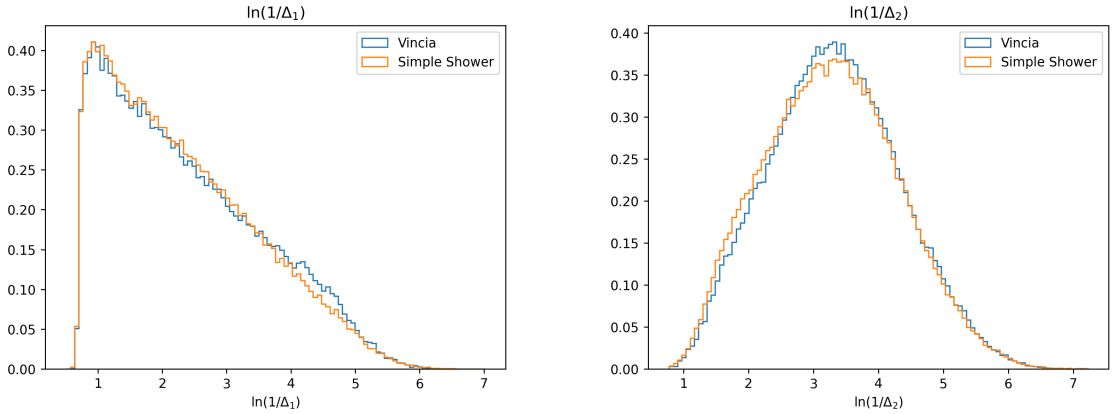


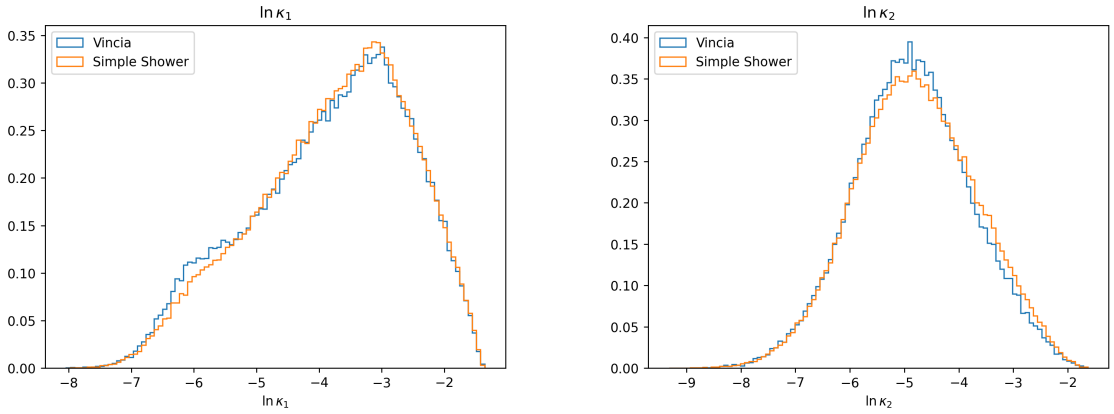

Figure B.1: The subleading branch. Higher $p_T$ sub-jets are placed on the left. After splitting the jet into two, we initially follow the lower $p_T$ sub-jets, and after that proceed to follow the higher $p_T$ sub-jets.




(a) 1st declustering

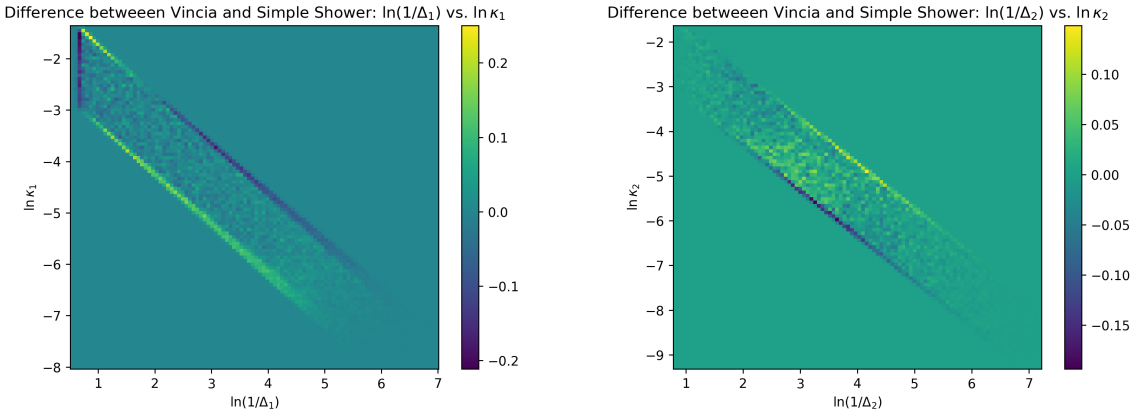
(b) 2nd declustering

Figure B.2: Distributions of $\Delta$ for 100 TeV events along the subleading branch. See table B.1



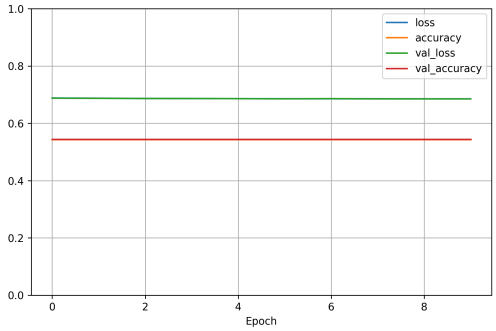

(a) 1st declustering (b) 2nd declustering

Figure B.3: Distributions of $\kappa$ for 100 TeV events along the subleading branch. See table B.1.

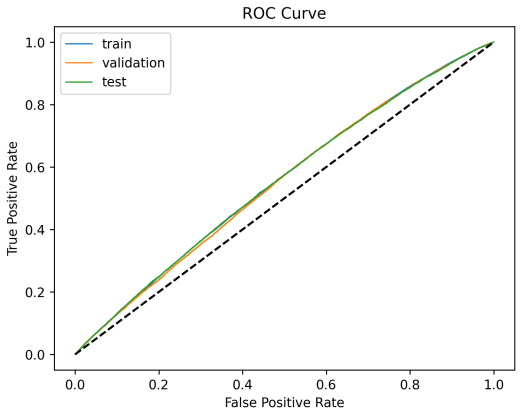



(a) 1st declustering (b) 2nd declustering

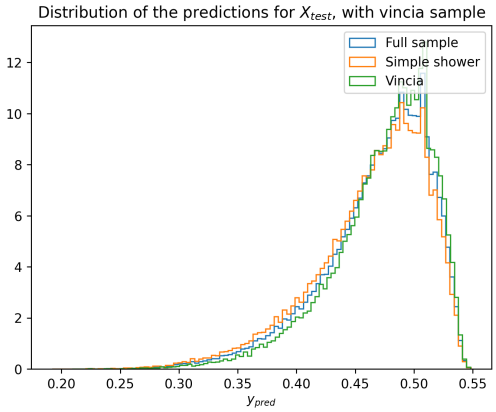
Figure B.4: Lund planes for the difference in density along the subleading branch, for 100 TeV events. See table B.1.




(a) Evolution of the accuracy and loss function




(b) ROC Curve




(c) Network output for the test set

Figure B.5: Evaluation of the neural network for the 100 TeV event, along the subleading branch, using the Lund plane variables for the first 4 declusterings as inputs.

8 neurons each was used, with outputs being shown in figure B.5.

Table B.1: A summary of settings used for the second test at 100TeV, using 4 declusterings as input, and of the neural network trained to compare parton showers.

| **Name** | 100TeV-3 |
|---|---|
| **Event and jet settings** | |
| Branch | Subleading branch of leading jet |
| $\sqrt{s}$ | 100 TeV |
| Jet algorithm | Anti-$k_t$ |
| Jet radius | 0.5 |
| Jet reclustering algorithm | C/A |
| Sub-jet radius | 1 |
| Jet $p_t$ | $>$ 1 TeV/c |
| Jet $\lvert\eta\rvert$ | $<$ 2.0 |
| Events generated | 1M Vincia + 1M simple |
| # of declusterings used | 2 |
| % of events kept after cuts | 23.0% (Vincia); 27.3% (simple) |
| **Neural network** | |
| Number of layers | 3 |
| Neurons per layer | 8 |
| Activation function | ReLU |
| Optimizer | Adam |
| Learning rate | 0.01 |