Investigation of Machine Learning for Jet Momentum Reconstruction in Heavy Ion Collisions

Jordan D. Lang
Department of Physics, University of Colorado Boulder

Undergraduate Honors Thesis
Defended March 22, 2023

Committee Members

Jamie Nagle
Thesis Advisor, Department of Physics

John Cumalat
Honors Council Representative, Department of Physics

Divya Vernerey
Department of Mathematics

Contents

Dedication

To the people in my life who showed me
you're never too old to learn something new.

Barbara Lang

Daniel Lang

Lois Lang

Ted Alderson

Vanessa Zacarias

Acknowledgements

I could not have made it this far without the support of so very many people. I wish I could include them all here, but I think that would be another thesis in itself. Instead, I wish to recognize and thank the mentors who guided me along on my journey into physics:

Professors Jamie Nagle and Dennis Perepelitsa, who brought me into their lab group and provided me with this amazing opportunity to learn about heavy ion physics. Through them, I realized my passion for the field of nuclear and particle physics. I feel incredibly lucky to have had this undergraduate research experience, and to have had both of them as mentors.

Professor Shuo Sun, who gave me my first undergraduate research experience and introduced me to quantum optics and photonics. I learned so much from him about hands-on experimentation and starting a research lab, and it has been wonderful to see his lab grow.

Steve Swingle from City College of San Francisco, who taught all of my introductory physics classes. He was my first real mentor in physics, and his passion for teaching inspired me to pursue physics as a career.

Finally, I must thank Dr. Maria Falbo who taught my high school physics class. She saw my affinity for physics well before I did, and I still remember her telling me to consider it instead of design. It took me a few years, but I finally got the message.

1 Introduction

This thesis covers my undergraduate work in the Heavy Ions Group at the University of Colorado Boulder, investigating the application of machine learning methods to the reconstruction of transverse momentum of jets in heavy ion collisions. This project largely follows from the paper *Machine-learning-based jet momentum reconstruction in heavy-ion collisions* from Rudiger Haake and Constantin Loizides, in which they simulate heavy ion collisions with a toy model and use it to train three machine learning algorithms that reconstruct jet momentum [1]. Their objective was to show how relatively simple machine learning tools can halve the resolution of reconstructed jet momenta, and can enable reconstruction of both lower-momentum jets and quenched jets, as shown in Fig. 1.

My project investigates this methodology and the validity of these results, and considers whether machine learning is an appropriate tool for jet momentum reconstruction.

Section 2 of this thesis provides an overview of useful background information including: quark-gluon plasma, jets, and machine learning. I have tried to present this at a level that is broadly accessible. Section 3 follows this with a detailed review of the methodology, covering the software used and specific implementation. Section 4 presents results for three simulated jet momentum distributions, and provides further analysis of a baseline model using linear regression. Finally, Section 5 reflects on the implications of this project and suggests avenues for future investigations. Some additional plots, references, and information are included in Section 6.

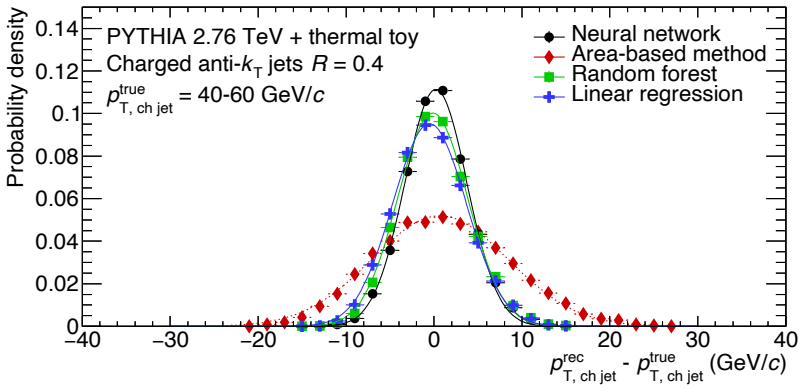

Figure 1: Key figure from Ref. [1] showing that machine learning and linear regression models (black, green, and blue) produce significant resolution improvements over the commonly-used area-correction method (red).

2 Background

2.1 Momentum and Angular Measures

This section provides a brief overview of momentum and angular measurements in the context of relativistic heavy ion physics. The terms and symbols included here are used frequently throughout this document.

Transverse and Longitudinal Momentum

Consider a particle with mass traveling away from a collision in a detector at relativistic speed. We can split the particle's vector momentum, $\mathbf{p}$, into two components[1]: **longitudinal momentum**, p_{L}, which is the momentum in the direction of the beam axis, and **transverse momentum**, p_{T}, which is the momentum in the transverse plane, perpendicular to p_{L}. Throughout this thesis we examine the transverse momentum of several objects, with "p_{T}" referring to general transverse momentum or the momentum of individual particles, "$p_{\mathrm{T,\ Jet}}$" referring to the transverse momentum of jets, and "$p_{\mathrm{T,\ Const.}}$" referring to the transverse momentum of constituent particles within jets [2, 3].

Azimuthal Angle, Rapidity, and Pseudorapidity

We describe collisions using a spherical coordinate system in momentum space, with the origin at the collision point, and the beam axis as the central axis. The simplest angular measure is the **azimuthal angle**, ϕ, which is just the angle around the barrel of the detector in the transverse plane, i.e. perpendicular to the beam axis.

There are two measures of the polar angle, θ, that are used in collider physics and throughout this investigation: rapidity and pseudorapidity. The necessity for these alternative angle definitions comes from the relativistic nature of objects produced in collisions. Differences in both rapidity and pseudorapidity are Lorentz invariant, meaning they are measured to be the same for any longitudinal reference frame. Thus these measures relate to p_{L} [2, 4].

Rapidity, y, is defined as:

$$y = \frac{1}{2}\ln\left[\frac{E + p_{\mathrm{L}}c}{E - p_{\mathrm{L}}c}\right] = \ln\left[\frac{E + p_{\mathrm{L}}c}{\sqrt{m^2c^4 + p_{\mathrm{T}}^2c^2}}\right]$$

[1] The vector momentum is the last three components of the four-momentum. The four-momentum is defined as $p^\mu \equiv \{E/c, p_x, p_y, p_z\}$, and is used for relativistic physics since we can easily transform it to different reference frames. Note that $E = \sqrt{m^2c^4 + |\mathbf{p}|^2c^2}$, so if a particle's mass is very small compared to its momentum, $E \approx |\mathbf{p}|c$. This condition is ideal for using pseudorapidity with massive particles.

While **pseudorapidity**, η, is defined as:

$$\eta \equiv -\ln\left[\tan\left(\frac{\theta}{2}\right)\right] = \tanh^{-1}\left(\frac{p_{\mathrm{L}}}{|\mathbf{p}|}\right)$$

Perpendicular to the beam axis, $\eta = 0$, and it increases as the polar angle gets closer to the beam axis[2], as shown in Fig. 2. One advantage of using pseudorapidity is that $\eta \approx y$ for particles traveling close to the speed of light, and $\eta = y$ if the particles are massless [4, 5].

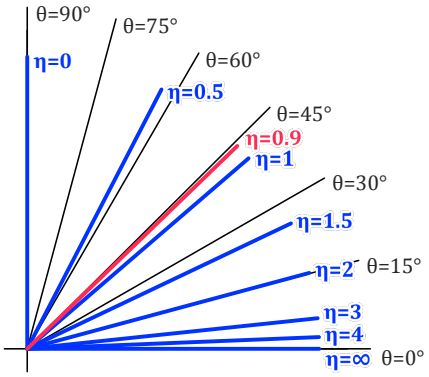

Figure 2: Diagram of pseudorapidity values in the lab frame compared to polar angle. Produced with reference to figures from Ref. [4].

The software packages we use store particle angles in $\eta - \phi$ space (pseudorapidity), while jets angles are stored in $y - \phi$ space (rapidity).

2.2 Partons

Quarks and Gluons

We start with the fundamental particles of this field: the **quarks**. Table 1 lists the six "flavors" of quarks, across three "generations", organized from lightest to heaviest. Quarks possess fractional electric charge, as well as color charge, a unique property of quarks and gluons which is covered later. Matter we encounter in everyday life contains only the up, u, and down, d, quarks (plus other non-quark particles). These are the lightest quarks and are stable. Heavier quarks have short lifetimes and will quickly decay into lighter particles, with their lifetime inversely proportional to their mass. Quarks couple to the **strong force**, for which **gluons** are the force-carrier [3, 7].

[2] An angle of $\theta = 45°$ roughly corresponds to $\eta = 0.9$. This is the acceptance of the ALICE detector and the pseudorapidity range used in this project [6].

Quark Flavor		Generation	Elec. Charge [e]	Mass [MeV]
Up	u	1	$+\frac{2}{3}$	2
Down	d	1	$-\frac{1}{3}$	5
Strange	s	2	$-\frac{1}{3}$	93
Charm	c	2	$+\frac{2}{3}$	1270
Bottom	b	3	$-\frac{1}{3}$	4180
Top	t	3	$+\frac{2}{3}$	173,000

Table 1: Quark flavors, generations, electric charge (in fundamental charge), and approximate mass (in electron-volts) [8].

Each quark has a corresponding antiquark, which is denoted by a bar over its symbol, like the anti-up quark, $\bar{u}$. Antiquarks have the opposite electric charge of their corresponding quark and carry anti-color, but have the same mass and generation. A quark and its antiquark can annihilate, forming a gluon or a high-energy photon. The process can also be reversed, generating a quark-antiquark pair from a gluon or photon[3] [3].

Quarks can carry any color charge, and there are three types of color charges that we call red, green, and blue. Color charges are not related to actual color, but do share properties with color theory. Combining all three colors produces a colorless object: red + green + blue = white (for light) [7]. This allows multiple quarks of the same flavor to exist in the same state, such as the two up quarks in the proton, uud. Just as placing a positive charge and negative charge next to each other results in a neutral net charge, a blue quark and an anti-blue antiquark have no net color [3].

Gluons, the carrier particles for the strong force, carry color charge and anti-color charge simultaneously (but are *not* colorless). When gluons are exchanged between quarks, they change the colors of those quarks in that interaction. While one might compare gluons to photons, the electromagnetic force carrier, gluons are quite different in that they couple to colored particles and have color themselves. Because of this, gluons can interact with other gluons and produces complex strong force interactions [7].

Quarks must always exist in a bound state, i.e., there are no "free" quarks as there are electrons or photons. The reason for this lies in the way the strong force interacts with quarks. If we examine two quarks that are close together – within the space of a proton, roughly 10^{-15} m or 1 fm – then the strong force is minimized and they can effectively move freely within that space. However, if we try to separate our quarks we will find the energy

[3]There additional interactions mediated by Weak force carriers, but these will not be discussed.

contained in their strong force interaction increases. If the separation is large enough, there is sufficient energy in the interaction to generate a new quark-antiquark pair, producing two pairs of bound quarks in a phenomenon we call **quark confinement** [9].

Fermions are a class of fundamental particles which includes quarks and electrons. Fermions possess half-integer spin, and obey the Pauli Exclusion Principle which states there cannot be two identical fermionic particles that occupy the same state[4]. All quarks are spin-$\frac{1}{2}$ particles (as are electrons) [3]. The details of spin will not be discussed beyond this.

Hadrons: Baryons and Mesons

Bound states of quarks and gluons create **composite particles** or **hadrons**, and we refer to quarks and gluons as **partons** (the "parts" of larger particles). We are all familiar with some of these composite particles, such as the proton and neutron. Since they make up atomic nuclei, we often refer to the proton and neutron as **nucleons** in contexts where the distinction between them is less important [3].

The nucleons – and all hadrons – are defined by their quark combinations[5]. As noted before, the proton is two up quarks and one down quark, uud, and the neutron is one up quark and two down quarks, udd. At low energies or long time scales, we consider a nucleon to be made of only these three quarks, bound together by gluons. However, as we increase the energy of a nucleon or interact with it on very short time scales[6], we find that it becomes a sea of gluons and virtual quark-antiquark pairs being produced and annihilated, as in Fig. 3. The nucleon's properties are determined by their **valence quarks**, the three quarks that we see at low energies [3].

There are two main classifications within hadrons. **Baryons** have three valence quarks, so the proton and neutron are baryons[7]. **Mesons** are comprised of a quark and an antiquark, with the most common mesons being the pions: π^+ ($u\bar{d}$), π^- ($\bar{u}d$), and π^0 ($\frac{1}{\sqrt{2}}(u\bar{u}+d\bar{d})$). Since

[4]A particle's state includes its spin angular momentum, its orbital angular momentum, and its spacial wave-function. Some particles may have additional components to their state. Electron orbitals are a common example of the Exclusion Principle, where the lowest ground state of an atom has no orbital angular momentum and can hold two electrons if they have opposite spin, since electrons are fermions. The nuclear shell model uses a similar structure for the protons and neutrons within an atomic nucleus.

[5]Technically we must consider spin states to distinguish higher energy hadrons

[6]This is a consequence of the Heisenberg Uncertainty Principle, which has a few well-known forms. $\Delta E \Delta t \geq \frac{\hbar}{2}$ is the form of interest here. Note how this requires that as Δt (time) gets smaller, ΔE (energy) must get larger to maintain this inequality. This is what allows for virtual particles!

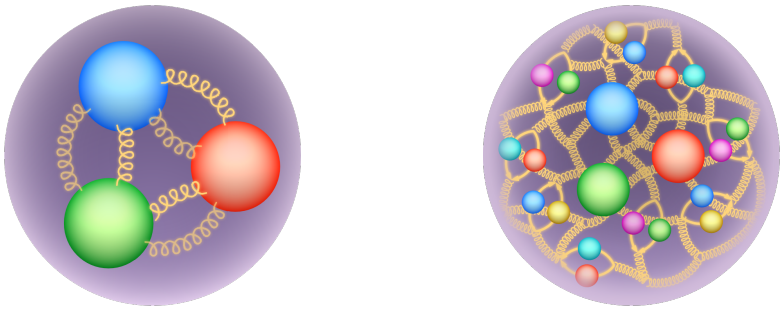

Figure 3: My illustration of quarks in a nucleon. Left: A nucleon at low energies, where only valence quarks are relevant. Right: At high energies the nucleon becomes a sea of gluons and virtual quarks and anti-quarks.

spins are additive, baryons (three quarks) are also fermions, but mesons (quark-antiquark) have integer spins and are called **bosons** [3].

All hadrons (baryons and mesons) have net integer electric charges from the fractional charges of their constituent quarks. This is how a proton can be made of three quarks but has a total charge of $+1$ e, where e is the magnitude of the electron charge. Similarly, all hadrons are colorless, regardless of the number or combination of valence quarks. We never encounter bare color charges at larger scales – there are no unbound quarks or gluons [3].

Parton Scattering

When hadrons collide, partons from opposing nucleons can interact and transfer their energy away from the beam axis. In a **hard scattering** event, partons collide directly and there is a large momentum transfer. The partons travel away from the collision point transverse to the beam axis and **fragment**, radiating gluons to produce a shower of partons (Fig. 6 in Section 2.4 visualizes this). Since quarks and gluons must be confined, this shower will **hadronize** in which quarks will combine or new quarks will be spontaneously generated, forming hadrons. This produces a branching structure of outgoing hadrons. The large initial energy of the scattered parton boosts these hadrons such that this structure is usually narrow or collimated. Partons can also undergo **soft scattering**, still transferring energy away from the beam axis but without the energy to produce one of these showers. These interactions

[7] They are the most stable baryons. The proton is extremely stable, and may exist indefinitely. The neutron is not stable in isolation, and will decay after roughly 15 minutes into a proton, an electron, and an anti-electron-neutrino: $n(udd) \rightarrow p(uud) + e + \bar{\nu}_e$.

produce diffuse collections of lower energy particles [10, 11]. In heavy ion collisions, however, there are other ways that particles can be produced.

2.3 Quark-Gluon Plasma (QGP)

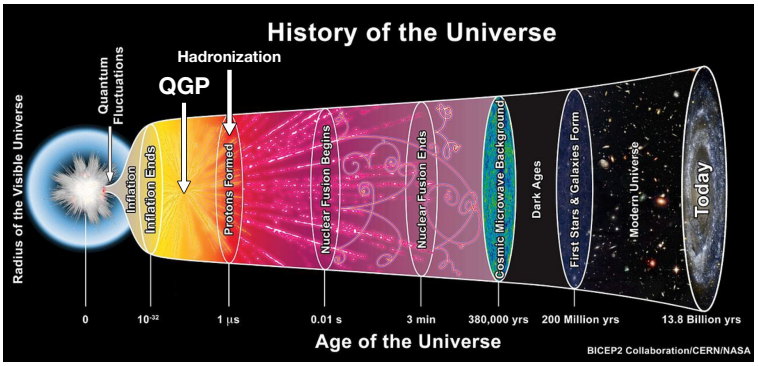

Figure 4: Diagram of the expansion of the universe. Quark-gluon plasma existed in the period between 10^{-32} s and 10^{-6} s. Image reproduced from Ref. [12].

Nanoseconds after the Big Bang, the universe existed as an extremely hot, extremely dense matter of deconfined quarks and gluons which we call **quark-gluon plasma** or **QGP**. Within microseconds, it had expanded and cooled, condensing into the first hadrons. Today, we can only create QGP in heavy ion collisions, where it lasts on the order of 10^{-23} seconds. In this state, quarks and gluons are no longer confined to hadrons and can move freely through the matter. Despite its incredible density, QGP flows like a near-perfect fluid governed by strong force interactions.

Heavy Ion Collisions

A lead nucleus, one of the heavy ions used in collision experiments, contains 208 nucleons, each with three valence quarks. At modern accelerator energies the virtual particles also become relevant, so in Pb + Pb collisions we get *far* more than $200 \times 3 = 600$ interacting particles. If the nuclei collide with enough overlap, QGP is temporarily created, as in the diagrams in Fig. 5. The QGP is a tiny fireball with enormous pressure that explodes outward. It rapidly expands from the initial collision point and cools down to confinement temperatures where quarks and gluons reform bound states in the process of **hadronization** [13].

The production of QGP requires that colliding nuclei have sufficient overlap. We measure the degree of overlap as the **centrality** of a collision. This investigation uses a simplified

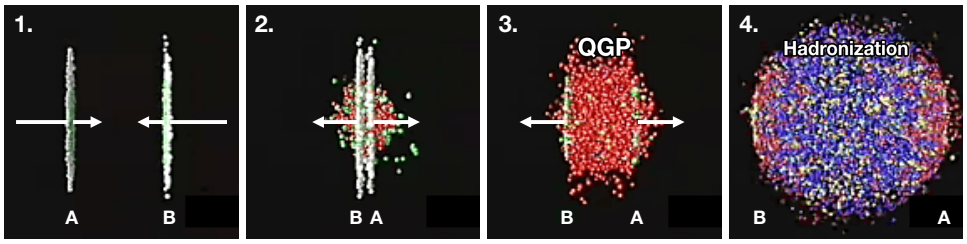

Figure 5: The steps of a heavy ion collision. 1) Two large nuclei, A and B, move towards each other at relativistic speeds. Due to length contraction, these normally spherical objects have been "flattened" into disks in our reference frame. 2) The nuclei pass through each other (note A and B have switched places), and QGP starts to form at the overlap of their dense centers. 3) As the nuclei remnants continue traveling apart, QGP forms in the region between them. 4) As it rapidly expands and cools, QGP undergoes hadronization as quarks recombine. These hadrons will spread out from the collision, and are what we see in detectors. Images reproduced from Ref. [14].

model of **central** Pb+Pb collisions in which nuclei completely, or nearly-completely, overlap (as in Fig. 5). If the nuclei have a small overlap, the collision is said to be **peripheral** [2, 9].

Since QGP is formed within the volume of a large atomic nucleus and only exists for 10^{-23} seconds, it is far too small and too short lived to detect directly. We must use indirect methods to measure QGP, such as how it effects "things" produced by the initial heavy ion collision that pass through it. We call these things **probes**. One key probe is jets, but there are others, such as hadrons containing heavy quarks, photons, and even leptons produced from Weak force interactions.

2.4 Jets

Recall the hard scattering interactions at the end of Section 2.2. If we were able to observe all of these final state particles produced by the scattered partons, then we could reconstruct information about the original partons.

Jet Definition

Unfortunately, detectors cannot perfectly measure every particle in this structure and its properties. Instead, we cluster what we *can* measure into **jets** with algorithms built around **jet definitions**. There are many jet definitions used in high-energy physics, but their common purpose is to combine seemingly correlated final-state particles in ways that provide information about the initial scattered parton and the branching structure formed from it [10].

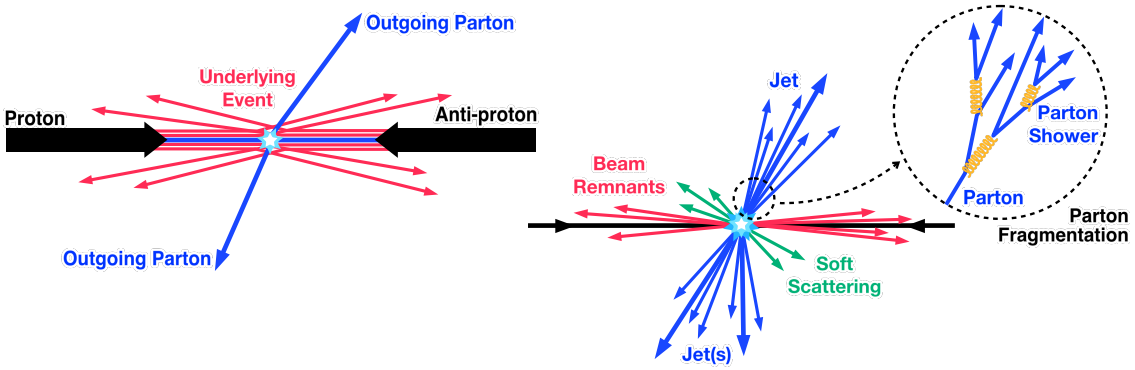

Figure 6: Diagram of hard scattering (left) and the resulting particle showers which are clustered into jets (right). Images adapted from Ref. [11].

Each algorithm defines how to determine jet centers, typically starting with a single high-p_{T} particle, how to cluster surrounding particles into the jet, and how to add the momentum or energy of those particles. Two common sources of data that is clustered into jets are **charged particle tracks** and **calorimeters**. This project is based on the ALICE detector which uses charged particle tracks [6]. These have the advantage of revealing the constituent particles within a jet and the jet structure, but lack information about neutral particles which make up roughly a third of a given jet. Calorimeters collect nearly all of the energy from a jet, but may lose some individual particle information. It is also possible to combine both methods for a more complete picture [10, 15].

Jet Quenching

In central heavy ion collisions, partons from hard-scattering interactions may pass through the quark-gluon plasma as they fragment and hadronize. The strong interactions within QGP affect these particles, and their initial energy is dispersed through the QGP medium in a phenomenon we call **jet quenching**. Quenched jets have a lower momentum and are broader than vacuum jets (like jets produced in $p+p$ collisions). Thus, the jet observables and structures contain information about QGP interactions, through which we can learn about the properties of QGP. We often do this by comparing spectra of jet observables from different collision types, like Pb + Pb and $p+p$ collisions. Every collision event, and every jet, is different, but we can use statistics to examine these QGP effects on large number of jets. [15].

Jet Clustering Algorithms

As noted in Section 2.3 section, the QGP does not simply disappear. It produces an abundance of lower energy particles as it expands, cools, and hadronizes. These hadrons, along with soft-scattered particles and beam remnants, produce a not-quite random noise over the measurements: the **background event** or **underlying event**. A great challenge in heavy ion physics is the identification of jets from hard scattering process from the background [15].

To do this we use jet finding algorithms, such as the anti-k_t algorithm. The anti-k_t algorithm looks for the highest p_T particles in an event and expands outwards within a given radius to categorize particles into conical jets around each high-p_T particle. Fig. 12 in Section 3.3 shows this clustering applied to our simulated data.

If additional high-p_T particles are found within the jet area, then the jet shape and area are modified. If two jets are near each other, the anti-k_t algorithm blends their borders based on the jet momentum. This is an improvement over older algorithms that assume circular jets or that prioritize jet borders based on momentum sorting [16].

Note that all particles in every event, whether it is a $p+p$ collision with relatively clear true jets or a Pb + Pb collision with a large background, are clustered into jets. Thus in heavy ion physics we face a separate challenge of deciding which jets are likely to be "true" jets, in addition to the challenge of reconstructing the true values of jet observables, namely the jet momentum.

2.5 Machine Learning

Machine learning is a broad field with a few major sub-fields. This section provides a brief and non-comprehensive overview of machine learning in order to clarify the context of this investigation early on. We only utilize regression estimators with supervised learning methods, since we wish to match a selection of jet observables to the true jet momentum, $p_{T,\,Jet}^{True}$, which has a continuum of possible values.

Broadly speaking, there are two main uses of machine learning: regression and classification. **Regression** produces a continuous function such that small variations in input data will gradually change the output value. For example, if inputting the value 1.0 outputs 8.5, inputting 1.1 might output 8.7. **Classification** sorts input data into discrete bins. Revisiting our example, an input of 1.0 may be classified as A, while 1.1 may be classified as B. Regression and classification thus solve fundamentally different problems, and techniques used for one may not apply to the other [17, 18].

Before discussing learning styles, let us cover some key terms. The machine learning algorithm is referred to as the **estimator**. When we train a model on data, we are training the estimator, and we then use the estimator to predict values from new data. In order to train an estimator (at least with supervised learning, which is explained shortly), we need two types of related data: the thing that we want to predict, which we call the **target**, and the data we want to use as inputs in the future, which we call **features**. Once an estimator is trained with a specific set of features, we cannot change those features. To make predictions, new data must include all of the same features [18, 19].

In the context of this project, our target is $p_{\text{T, Jet}}^{\text{True}}$, the true momentum of each jet. Our features vary, but include the raw jet momentum, $p_{\text{T, Jet}}^{\text{Raw}}$, the jet area, and the p_{T} of the constituent particles within each jet.

As hinted at earlier, there are different styles of training. Three main categories are supervised learning, unsupervised learning, and reinforcement learning. **Supervised learning** requires that all input data is split into features and targets. For regression, targets are the true values we wish to estimate on a continuum. With classification, targets are manually sorted into categories for training. **Unsupervised learning** models take in features and then attempt to find relationships between them. There are no target values or categories so the estimator has free reign to find connections which may or may not be useful. **Reinforcement learning** is almost a blend of the two. It uses a reward system to build connections between input data, and can have some manual oversight without requiring all data to be tagged as features and targets [17, 18].

Linear Regression

Linear regression is a basic form of machine learning, relying primarily on algorithms that optimize the determination of the line of best fit between each feature and the target data. This creates a hyperplane in N-dimensional space, where N is the number of input features. Though it is relatively simple, linear regression outputs the coefficients for each feature, and a y-intercept if needed, which gives an explicit and understandable function [18, 19]. However, linear regression has several limitations, as demonstrated in Fig. 7.

For example, if we want to reconstruct the momentum of a jet using the raw momentum, $p_{\text{T, Jet}}^{\text{Raw}}$, area, A_{Jet}, and the momentum density from the collision event, ρ_{Event}, we might get a linear regression function like this:

$$p_{\text{T, Jet}}^{\text{Reco}} = a_0 + a_1(p_{\text{T, Jet}}^{\text{Raw}}) + a_2(A_{\text{Jet}}) + a_3(\rho_{\text{Event}})$$

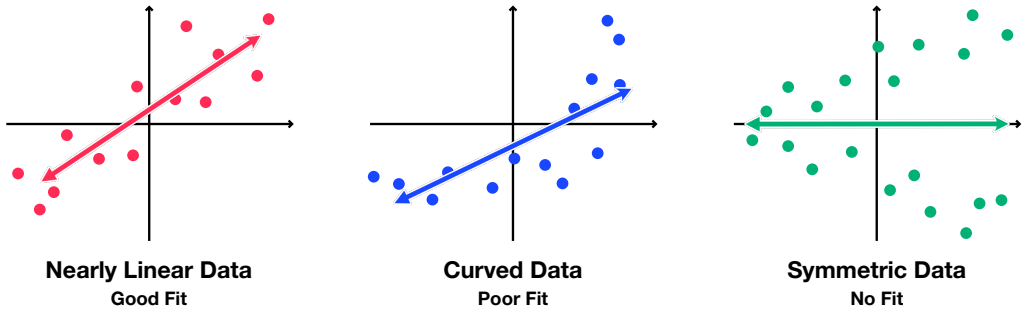

Figure 7: Diagrams of linear regression fits. If the data is reasonably linear (red plot), then the fit will be good. If the data shows a curve (blue plot), then the fit will be poor, depending on the degree of the curve. Finally, if the data is symmetric about either axis (green plot), then there will be no fit even though there may be strong linear dependence within the data.

In fact, these are the features used for the area-correction method which has the form $p_{\mathrm{T,\,Jet}}^{\mathrm{Reco}} = p_{\mathrm{T,\,Jet}}^{\mathrm{Raw}} - (A_{\mathrm{Jet}})(\rho_{\mathrm{Event}})$, but linear regression is unable to reproduce this relationship. Each feature has a unique coefficient and a purely linear relationship to the target value. More advanced machine learning models – like random forests and multilayer perceptrons – are able to find nonlinear relationships between features, but we cannot directly observe the relationships they build.

Random Forest Regression

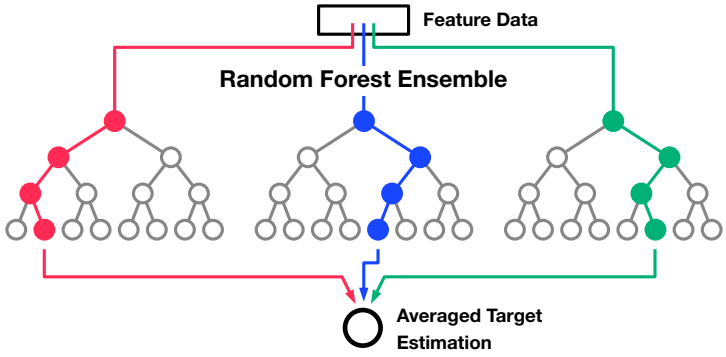

Figure 8: A simplified diagram of random forest estimator structure. Data is passed through multiple decision trees (the ensemble) for training. The output from all trees is averaged to produce the final value. This makes random forests decently robust to small gaps in data. Images adapted from Ref. [20].

Random forest regression uses an ensemble of decisions trees to find connections between features that improve the estimation of targets, as represented in Fig. 8. Combinations of features on subsets of data may be explored in order to identify these relationships.

Random forest estimators can determine **feature importance** on a scale of 0 to 1, which can be used to understand how relevant each feature is to estimation. Features that have the strongest impact on estimation relative to others will branch earlier in the process and are assigned a higher feature importance. The sum of all feature importance values is 1. No explicit function is output from a random forest estimator, so one must rely on the feature importance to guess at the feature relationships it is finding [18, 19].

Multilayer Perceptron Regression (Shallow Neural Network)

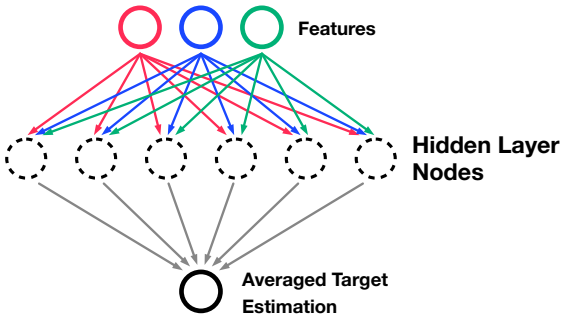

Figure 9: A simplified diagram of multilayer perceptron structure. Features are passed to a hidden ensemble of nodes, each of which has an activation function. The outputs of activated nodes are averaged to produce the target value. Images adapted from Ref. [21].

Multilayer perceptron estimators are a basic type of neural network. In our implementation, they are a shallow network with only three layers of nodes which features are passed between, as in Fig. 9. Neural networks can construct intricate nonlinear correlations between features, and claim to be capable of finding nuanced relationships that can drastically improve the resolution of estimation. However, they are black boxes and provide no outputs for discerning the connections they have made or glimpsing the complex algorithms created. They can be powerful, but their obscurity puts them at a greater risk of suggesting relationships that do not exist. Furthermore, a poorly constructed neural network may **overfit** to training data, suggesting a greater estimation power than it actually possesses [18, 19].

2.6 ROOT and Data Storage

The majority of the code for this project relies on the software package ROOT from CERN [22]. ROOT is implemented through C++ scripts, and has many useful functions and defined objects. It is excellent for efficiently handling large quantities of data, such as thousands of particles across hundreds of thousands of events. In addition, ROOT has built-in functions for random number generation and plotting. Because of this, most of the packages mentioned in Section 3 are implemented alongside ROOT.

One common data storage type used throughout this project is the **TTree**. TTrees organize data using a hierarchy structure: information is stored by "entry", and variables are assigned to "branches" which are filled by entry. Branches may be a single value, a vector, or a multidimensional array. For example, a single Pb + Pb collision would constitute an entry, and the p_{T}, η, ϕ, and mass of each particle would be branches with arrays. Each event can have thousands of values per branch, all organized by their particles.

The plots throughout this document were also produced in ROOT.

3 Implementation

3.1 Methodology

This section describes the process used to investigate and experiment with machine learning in the context of jet momentum reconstruction. Each step is explained in depth here, but some additional figures can be found in the appendix. Our workflow generally follows the methodology outlined by Haake and Loizides, members of the ALICE collaboration, and so we use the constraints of the ALICE detector [1, 6].

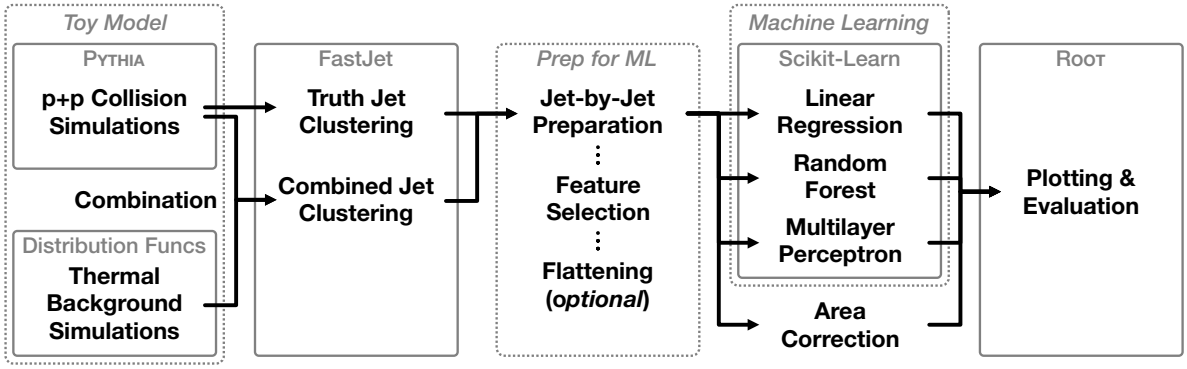

Figure 10: Overview of the process. Toy model events are generated by combining charged particles from PYTHIA $p+p$ collisions with particles produced by the thermal model distribution functions. Jet clustering is done on PYTHIA events and combined events using FASTJET. Features are stored per-jet in files for machine learning, and $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ distributions may be flattened. Three machine learning estimators are trained and tested with scikit-learn to produce machine learning $p_{\mathrm{T,\,Jet}}^{\mathrm{Reco}}$ values, in parallel with area-correction. Outputs are compared and plotted in ROOT.

We simulate $p+p$ PYTHIA events at a beam energy of 2.76 TeV. To model the acceptance of the ALICE detector, we only record final-state charged particles within the pseudorapidity of $|\eta| \leq 0.9$[8] [6]. These particles are clustered into jets with FASTJET to determine $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$. Event particles are saved to a ROOT TTree for further study if at least one of the PYTHIA jets has a transverse momentum with 10 GeV $\leq p_{\mathrm{T,\,Jet}}^{\mathrm{PYTHIA}} \leq$ 90 GeV. This provides the "true" jets, from which we get the target value $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$.

An underlying event is built from the thermal model. Parameters for thermal particles are randomly selected from distribution functions, as detailed in Section 3.4. PYTHIA particles and thermal particles are combined and stored in a second TTree. These combined events

[8]Note that other detectors, such as ATLAS and CMS, may have a larger acceptance. These methods can be adapted to other acceptances and detector constraints in future investigations. The sPHENIX detector has a similar acceptance, and was a motivation for this study.

are clustered again with FastJet, producing the raw jets which are closer to what we expect from real data. Several observables from these jets, including $p_{\mathrm{T,\,Jet}}^{\mathrm{Raw}}$, are used as features for machine learning to estimate $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$.

Importantly, we diverge from the reference paper in our definition of $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$. We define $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ as the transverse momentum of the Pythia jet, as opposed to the authors' method of calculating $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ as the sum of the Pythia particle momenta within the raw (combined event) jet [1]. The difference in $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ definitions is small, and the definition in the reference is non-standard and more computationally complex.

For this investigation, we generate 500,000 events for each training and testing distribution. This is *not* the number of jets used for training. In most distributions that number is above 600,000 jets, but our flattened data sets have $350,000$ jets. Finally, note that we are not splitting a larger data set into testing and training data sets. These are two *separately generated* files in order to maintain truly statistically independent training and testing samples.

3.2 PYTHIA

Pythia is a software package used for generating high energy particle collisions, or "events." It uses Monte Carlo simulations based on physics models to replicate experimentally observed phenomena [23–25]. Pythia is effective for producing vacuum jets, like those from $p+p$ collisions, but does not model jet quenching processes in heavy ion collisions. Thus our initial study does not incorporate quenching effects, only vacuum jets embedded in a thermal background.

We implement Pythia as part of a larger C++ script for event generation. This script also includes jet clustering via FastJet (Section 3.3), and our thermal model. To emulate the ALICE detector at the energies simulated in the reference paper we run Pythia p+p collisions with center-of-mass energy of 2.76 TeV and record only charged[9], final state particles within a pseudorapidity of $|\eta| < 0.9$ [6].

We configure $p+p$ collisions with the Pythia setting `HardQCD:all` to produce only hard scattering events and enhance our statistics for events containing jets. This also requires that parameter `pTHatMin` be nonzero, which defines the lowest allowed transverse momentum of the initial collision. and affects the jet branching and final state particles (the evolution of the parton shower resulting from the collision) [23]. To balance a "realistic-enough" particle p_{T}

[9]Neutral particles constitute roughly one-third of a jet. This is the trade-off of using charged particle tracks [15].

distribution with the lengthy simulation time, we set the parameter `pTHatMin` to $\frac{3}{4}\left(p_{\mathrm{T,\,Jet}}^{\mathrm{Min}}\right)$. This is 7.5 GeV in our case because we select jets with 10 GeV $\leq p_{\mathrm{T,\,Jet}}^{\mathrm{PYTHIA}} \leq$ 90 GeV.

A realistic $p_{\mathrm{T,\,Jet}}$ distribution is rapidly falling and, in our nominal statistical sample, has fewer than 10 jets in the highest $p_{\mathrm{T,\,Jet}}$ bins, as shown in Fig. 11. We address this by applying a bias to event generation with the parameter `bias2SelectionPow`. This applies an exponential bias to the particle p_{T} distribution in order to increase the multiplicity of high-p_{T} jets, which has the form $\left(\hat{p_{\mathrm{T}}}/p_{\mathrm{T}}^{\mathrm{Ref}}\right)^{[\mathrm{Bias\ Power}]}$, where $p_{\mathrm{T}}^{\mathrm{Ref}}$ is the realistic p_{T} cross section given other parameters and $\hat{p_{\mathrm{T}}}$ is the transverse momentum of scattered partons immediately after the initial collision [23]. This allows us to create smooth distributions with additiona high-p_{T} jets in a single event generation process[10].

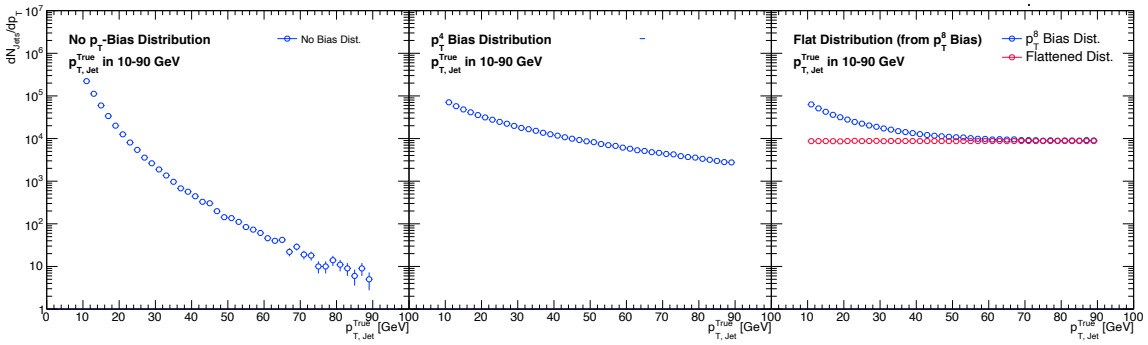

Figure 11: The three $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ distributions we investigated, with multiplicity on a log scale. The no bias distribution (left plot) does not use the PYTHIA parameter `bias2SelectionPow`. The p_{T}^{4} bias distribution (middle plot) uses `bias2SelectionPow = 4`. The flat distribution starts with a p_{T}^{8} bias (right plot, blue), `bias2SelectionPow = 8`, which is then flattened (right plot, red) using the randomized flattening function described earlier.

Finally, randomness between events is ensured with the parameters `Random:setSeed = on` and `Random:seed = 0`. These settings incorporate the device time into the randomization seed so no two data sets are identical, regardless of event generation parameters. Without this explicit randomization, sets of events may be repeated over multiple PYTHIA runs if they are executed in the same order.

[10] A common technique to increase high-p_{T} statistics is to produce several sets of events with different p_{T} bins, adjusting $\hat{p_{\mathrm{T}}}$ for each bin. These distributions are combined to increase statistics at higher-p_{T}, and may be re-weighted to ensure uncertainties are appropriate. It is our understanding that Haake and Loizides used this method to produce their distribution, but the original distribution is no longer accessible.

3.3 FastJet

FastJet is a software package that clusters particles in collision events into jets [26]. We perform clustering on Pythia $p+p$ collision data to determine the truth jets and $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$, which is our target for machine learning. We also cluster the combined event data which includes thermal background particles detailed in Section 3.4, and use observables from combined jets as features for machine learning estimators. Both of these clustering steps are visualized in Fig. 12.

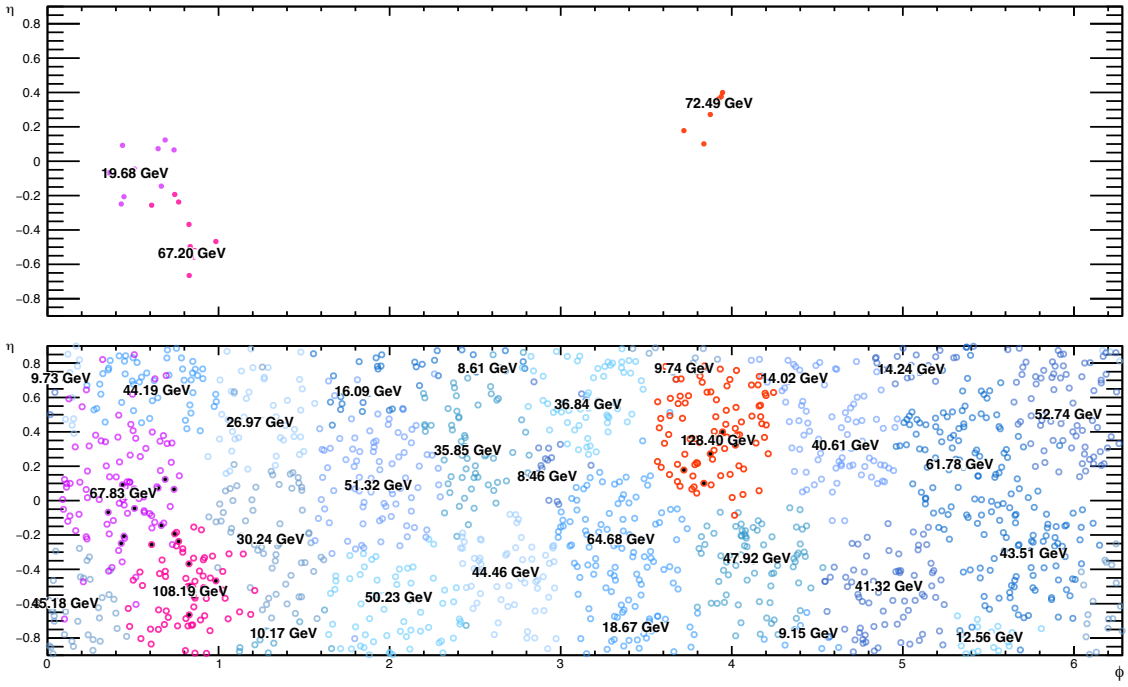

Figure 12: Clustered jets and their constituent charged particles in $\eta-\phi$ space, labeled with $p_{\mathrm{T,\,Jet}}$. Top: Jets clustered from a Pythia $p+p$ event. These are our truth jets, and their p_{T} is $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ [11]. Bottom: Jets clustered from a toy model event where the Pythia particles (black dots) have been combined with thermal background particles. In this case, the three highest-p_{T} jets from the combined event nearly match the locations of the Pythia jets, and their p_{T} is $p_{\mathrm{T,\,Jet}}^{\mathrm{Raw}}$. The remaining jets form the background event[12].

[11] We can even see a potential clustering problem, as the proximity of the two left-most jets in the top plot suggests they may have originated from the same initial scattering event.

[12] This is an intentionally "ideal" event. The raw jets and the Pythia jets are well-aligned, the jets are roughly 2π apart in ϕ space, the three highest-p_{T} jets in the combined event match the Pythia jets, and the background is well-behaved. This is not always the case. Since we select events that have at least one jet with $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ in 10-90, many events have jets with $p_{\mathrm{T,\,Jet}}^{\mathrm{True}} >> 90$ GeV which are not used for machine learning. If the thermal particles are particularly turbulent or include a high-p_{T} particle, jets may be clustered quite differently from the Pythia jets.

We use the anti-k_t algorithm to cluster all of our jets [16], using a jet radius of 0.4 in $y - \phi$ space[13] for PYTHIA and thermal model jets. This gives a mean truth jet area of approximately $\pi R^2 = 0.5$ for our combined jets. Jet area is calculated using the ghost particle method within FASTJET, with a ghost particle area of 0.05 [26]. Clustered jets include lists of constituent particles and parameters for each particle. After the jet clustering step, particle data is only stored per-jet.

3.4 Thermal Model

The toy thermal model is a highly simplified emulation of the thermal background event for central Pb+Pb events. Unlike real background events which can have fluctuations dependent on the collision parameters, this toy model is essentially uniform in distribution.

The number of thermal particles is defined by a Gaussian function centered at 1800 with a standard deviation of 200. Particle η and ϕ are randomly selected from flat distributions. The p_T of each particle comes from a Modified-Hagedorn function fit to HEP data. The Modified Hagedorn function is used because it describes the p_T distribution of an underlying event in a heavy ion collision, specifically the thermal particles and particles from soft-scattering interactions [27]. We left all parameters in the Modified-Hagedorn function undefined[14]and applied a fit to ALICE data for Pb+Pb collisions at $\sqrt{s_{NN}} = 2.76$ TeV [28, 29]. We obtained the following fit parameters, and Fig. 13 shows the fit line compared to HEP data.

$$A_0 \frac{p_T^2}{(p_T^2 + A^1)^{1/2}} \left(1 + \frac{p_T}{A_2}\right)^{-A_3}$$

A_0	64547
A_1	3.076
A_2	1.126
A_3	8.491

We merge PYTHIA particle data and thermal model particle data into a single TTree to produce the "combined event" data. In practice, the thermal model and PYTHIA particles are saved as individual TTrees, and the combined event is a third TTree, allowing for more effective debugging and the ability to explore the individual components prior to merging.

[13] Fastjet uses rapidity, y, for the polar angle of jets, while particle polar angles use pseudorapidity, *eta*. For more information, see Section 2.1.

[14] Note that parameter A_1 should be particle mass, such that $A_1 \approx m_\pi$, but we obtained a tighter fit to the data by leaving this unfixed. Since this is for a toy model application, this adjustment was deemed acceptable. Additionally, parameter A_3 is set as positive for the function in code, and the input value is negative. This prevented peculiar fit behavior.

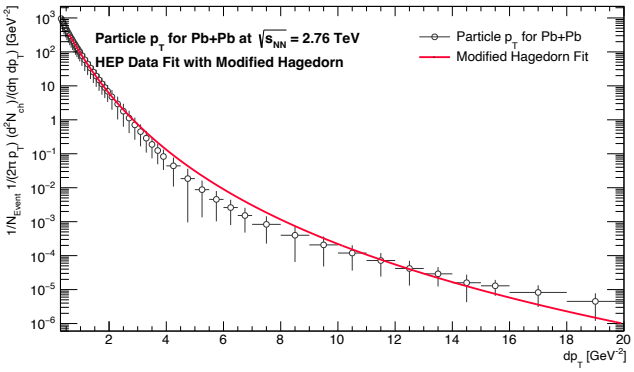

Figure 13: Fit of Modified Hagedorn function to HEP data of central Pb+Pb collisions at $\sqrt{s_{NN}} = 2.76$ TeV [28]. Our fit is not perfect, but is well-aligned for p_{T} in 1-4 GeV which encompasses the majority of the thermal particles. Since this toy model excludes other effects, such as jet quenching, this fit is sufficient for our investigation.

Note that our thermal model differs from the thermal model of the reference paper [1]. We contacted the author Constantin Loizides, but were unable to obtain the exact parameters used for the original thermal model. The paper references using a power law function for p_{T}, but we chose to fit to Pb + Pb HEP data that was also referenced in that paper.

3.5 Preparation for Machine Learning

We match truth jets to the nearest high-p_{T} jet from the combined event within a radius of 0.3 units in $y - \phi$ space. This is done by satisfying the following relationship between combined jet angles and PYTHIA jet angles:

$$|\eta_{\mathrm{Jet}}^{\mathrm{PYTHIA}} - \eta_{\mathrm{Jet}}^{\mathrm{Combined\ Event}}|^2 \ + \ |\phi_{\mathrm{Jet}}^{\mathrm{PYTHIA}} - \phi_{\mathrm{Jet}}^{\mathrm{Combined\ Event}}|^2 \ \leq \ (0.3)^2$$

Data is prepared for machine learning by filtering jets and storing them in a "flat" TTree, where each entry is a single jet and its measured values (Note that this is *not* the same as flattening the distribution). This converts our analysis from an event-by-event to jet-by-jet basis. Finally, this TTree is converted to a `.csv` format for machine learning in Python, using the PyROOT library which enables ROOT functions within Python [22]. After this point, Python functions only use the `.csv` files. All recorded input features are included in these "master" data files.

Distribution Flattening

We can flatten the $p_{\text{T, Jet}}^{\text{True}}$ distribution of the jet TTree using a custom flattening function. This function determines the minimum number of jets in any $p_{\text{T, Jet}}^{\text{True}}$ bin, or takes a minimum number of jets per bin to flatten to as a parameter. It determines the ratio of total jets to minimum jets, and uses that ratio to randomly determine whether each jet should be copied jets from the original TTree into a new flat distribution TTree. This alters the distribution such that the number of jets is the same across all bins, on average.

As a direct example, consider the third plot in Fig. 11 (Section 3.2). We take a p_{T}^{8} bias distribution (in blue) as an input, and apply the flattening function. The resulting flat distribution (in red) has some small variations due to using a random generator, but has nearly the same number of jets per 1 GeV $p_{\text{T, Jet}}^{\text{True}}$ bin.

3.6 Area Correction Method

The resolution improvements of machine learning methods are compared to an existing $p_{\text{T, Jet}}$ reconstruction method which we refer to as the "area-based correction method" or "area correction" for short. This method calculates the background momentum density, ρ, of each collision event and subtracts the density times the jet area from each $p_{\text{T, Jet}}^{\text{Raw}}$ value.

Our implementation of this method calculates the jet momentum density for all regions but the areas corresponding to the highest two jets in the event. We take the median of all of these jet densities and use that as our event-wide background density for the area correction method. If we sort the jets from highest-p_{T} to lowest-p_{T} with the highest at index 1, we can express this as:

$$\rho = \text{median}\left\{\frac{p_{\text{T, Jet}}^{3}}{A_{\text{Jet}}^{3}}, \frac{p_{\text{T, Jet}}^{4}}{A_{\text{Jet}}^{4}}, \ \ldots\ , \frac{p_{\text{T, Jet}}^{\text{n}}}{A_{\text{Jet}}^{\text{n}}}\right\}$$

Using the same notation, we can express the area correction calculation for the i^{th} jet as:

$$p_{\text{T, Jet}}^{\text{i, Area Corr.}} = p_{\text{T, Jet}}^{\text{i, Raw}} - (A_{\text{Jet}}^{\text{i}})(\rho_{\text{Event}})$$

We compare our area correction distribution to that of the paper in Fig. 14, and find a difference in the distribution widths. This is likely due to the difference in our thermal model from that used for the reference paper, as we were unable to obtain the original parameters and functions used.

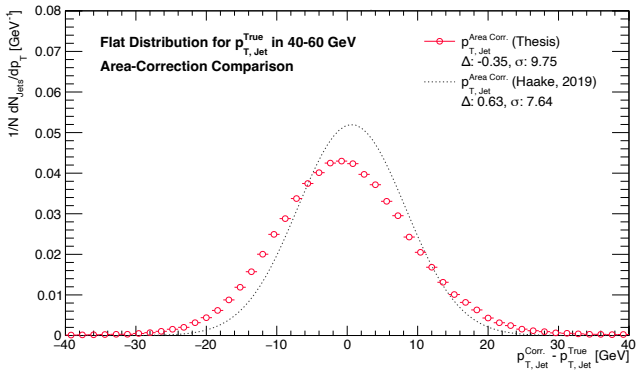

Figure 14: Comparison of our area-correction distribution with that of the reference paper [1]. Note that our area correction distribution has a larger width and is offset in the opposite direction.

3.7 Machine Learning with scikit-learn

Machine learning is implemented through the scikit-learn Python library [19]. Functions for training and testing machine learning models are kept in Python scripts and called by a Jupyter notebook [30]. This workflow is chosen to leverage Jupyter's flexibility for parameter modification, allowing for iteration and easier control over individual steps.

Three regression models from scikit-learn are used: linear regression (LR), random forest regression (RF), and multilayer perceptron regression (MLP). The specific features used for training and testing are given in Table 2.

The Machine Learning Process

First, specific entries of the master `.csv` files are added to Python arrays - one array for input features, another for targets. These are kept in memory and not saved externally, enabling quick iteration of features used for training. These feature and target arrays are generated for each training data set and testing data set.

A training data set with a given number of features is passed to a machine learning estimator pipeline. Pipelines in scikit-learn link estimators with processing tools into a single Python object that can be trained and referenced. Our pipelines incorporate the preprocessor `StandardScaler`, followed by the machine learning estimator. This re-centers the mean of each feature distribution to 0, and then divides feature values by the mean such that the standard deviation is unity [19]:

$$x_i^{ML} = \frac{(x_i - \bar{x})}{\sigma_x} \ , \qquad \sigma_x = \sqrt{\frac{(x_i - \bar{x})}{N_x - 1}}$$

Linear regression feature coefficients and y-intercepts, as well as random forest regression feature importance scores, are written to `.csv` files for future reference.

The Python library `joblib` is used to save estimators for future use. Since estimators can take a long time to train, this streamlines the process of testing on multiple distributions or repeating non-training parts of the Python scripts.

Testing data is passed to the trained pipeline and the results are saved to a `.csv` file for analysis, storing the features, target, and estimated target values. These files are imported back into ROOT for analysis.

All plotting is done within ROOT [22].

4 Results

4.1 Evaluation Metrics

Our jet momentum reconstructions are nearly Gaussian in nature, so we evaluate the performance of each method by considering the offset of the mean from zero and the width of the distribution as fit with a Gaussian function. Improvements over the area-correction method for nearly-centered distributions are determined by the width ratios $\sigma^{\text{Reco}}/\sigma^{\text{Area Corr.}}$. Histograms are plotted with the difference between the reconstructed and the true jet momentum in GeV on the horizontal axis, $p_{\text{T, Jet}}^{\text{Reco.}} - p_{\text{T, Jet}}^{\text{True}}$, and are normalized such that the integral over the distribution is unity, so the vertical axis is the probability density for each bin.

Since we are evaluating this methodology in the context of the paper produced by Haake and Loizides, plots are made with similar axes and color codes to the key figure from their paper [1]. Note Fig. 15, which shows a width improvement of $\sigma^{\text{Reco}}/\sigma^{\text{Area Corr.}} \approx 0.5$ for linear regression and random forest regression, with a greater improvement using multilayer perceptron regression (neural network).

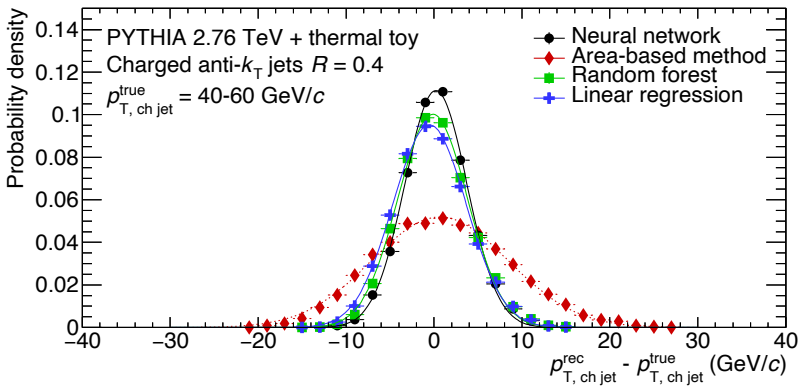

Figure 15: Reproduced from Ref. [1].

4.2 Baseline Model

We choose a flat $p_{\text{T, Jet}}^{\text{True}}$ distribution to train our baseline model. The training distribution used by Haake and Loizides has been described as a falling distribution with additional high-p_{T} events to provide enough statistics at high-p_{T}, but specifics about this distribution or the parameters used to produce it are unknown [1]. We seek to remove all potential biases from the $p_{\text{T, Jet}}^{\text{True}}$ distribution, and ensure our machine learning models are identifying relationships between features only.

To compare the effects of the training distribution on jet momentum reconstruction, we also train the estimators with a no bias distribution and a p_{T}^4 bias distribution. These distributions, and our baseline, are produced by generating 500,000 toy model events. The outputs are shown in Fig. 11. To obtain our flat baseline we first generate events with a p_{T}^8 bias. This results in roughly 660,000 jets. The flattening function, described in Section 3.5, randomly removes jets above the minimum bin size. This reduces our total number of jets to roughly 350,000, or a little more than 4000 jets per 1 GeV bin over the $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ range of 10-90 GeV.

The testing data set also has a flat $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ distribution to ensure roughly testing statistics across the whole training range. This is generated separately to ensure the data is independent. Additionally, we test on narrow bins in $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ so that the reconstruction distributions are not sensitive to the testing distribution. Only the training distributions influence our results[15].

We select 11 features to train with. The reference article uses 20 features, which we initially reduced to the top 12 based on the feature importance listed in that paper, from the authors' random forest estimator [1]. One of these is $p_{\mathrm{T,\,Jet}}^{\mathrm{Area\ Corrected}}$, which is determined from $p_{\mathrm{T,\,Jet}}^{\mathrm{Raw}}$, jet area, and background p_{T} density (See Section 3.6). Since this thesis is an investigative exercise, we removed $p_{\mathrm{T,\,Jet}}^{\mathrm{Area\ Corrected}}$ from the feature list to avoid obviously correlated features. This influences the feature importance and linear regression coefficients of our baseline model to be more informative. With 12 features, $p_{\mathrm{T,\,Jet}}^{\mathrm{Area\ Corrected}}$ had an importance of 0.829 and $p_{\mathrm{T,\,Jet}}^{\mathrm{Raw}}$ an importance of 0.022, while with 11 features $p_{\mathrm{T,\,Jet}}^{\mathrm{Raw}}$ has an importance of 0.742. Furthermore, reconstruction distributions with 11 features versus 12 features are nearly indistinguishable.

The linear regression coefficients and feature importance values from random forest regression are included in Table 2. Feature importance is a measure of how useful a feature is for distinguishing between potential target values, so a high or low feature importance does not correspond to the magnitude of linear regression coefficients. Still, comparing these values suggests intriguing differences between the mechanisms of these methods. For example, jet rapidity has an importance of 0.015 and a regression coefficient of -0.007. The regression coefficient suggests jet rapidity is essentially inconsequential, while feature importance suggests this offers some valuable information. Unfortunately, we cannot examine the random forest decision tree directly, but we know that rapidity should have a symmetry about $y = 0$ due to the symmetric nature of Pythia $p + p$ collisions (see Fig. 21 in Section 6). The random

[15] See Fig. 18 towards the end of this chapter for evidence of this.

Training Features		Feature Importance	Lin. Regression Coefficient	Feature Set 1	3	11
Raw Momentum	$p_{\mathrm{T,\,Jet}}^{\mathrm{Raw}}$	0.742	16.288	○	○	○
Background p_{T} Density	ρ_{Event}	0.008	-1.350		○	○
Jet Area	A_{Jet}	0.005	-1.428		○	○
Jet Mass	m_{Jet}	0.009	-2.354			○
Jet Rapidity	y_{Jet}	0.015	-0.007			○
Constituents in Jet	$N_{\mathrm{Const.}}$	0.003	-1.801			○
Constituent Mean p_{T}	$p_{\mathrm{T,\,Const.}}^{\mathrm{Mean}}$	0.149	1.133			○
Constituent 1 p_{T}	$p_{\mathrm{T,\,Const.\,1}}$	0.015	3.561			○
Constituent 2 p_{T}	$p_{\mathrm{T,\,Const.\,2}}$	0.022	1.928			○
Constituent 3 p_{T}	$p_{\mathrm{T,\,Const.\,3}}$	0.017	1.341			○
Constituent 4 p_{T}	$p_{\mathrm{T,\,Const.\,4}}$	0.015	3.553			○
(y-Intercept)*		–	50.013			

Table 2: Table of features used for machine learning. Columns 3 and 4 list the feature importance for the random forest and the linear regression coefficient computed during training on 11 features with a flat distribution. The remaining columns indicate which features are included in each feature set. Feature importance and coefficients for 1 and 3 feature sets are in Table 4 in Section 6.1.
*Note that y-intercept only applies to linear regression.

forest and multilayer perceptron estimators may find relationships in this distribution that linear regression cannot. Other features that have a nonlinear relationship may be similarly under-weighted in linear regression.

4.3 Evaluation of Distributions and Features

To highlight the effects of the training distribution of $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ on machine learning, consider Fig. 16. Here we examine the effects of both the training distribution and the number of input features on a restricted test region of $p_{\mathrm{T}}^{\mathrm{True}}$ = 48-52 GeV, the center of our training $p_{\mathrm{T,\,Jet}}$ range. We compare three training distributions: no bias, p_{T}^4 bias, and flat (the left, middle, and right plots in Fig. 11, Section 3.2). We also compare three feature sets: 1 feature, 3 features, and 11 features (Table 2 lists which features are included in each feature set). Note that the 1 and 3 feature sets do not include information about constituent particles, only the overall jet observables.

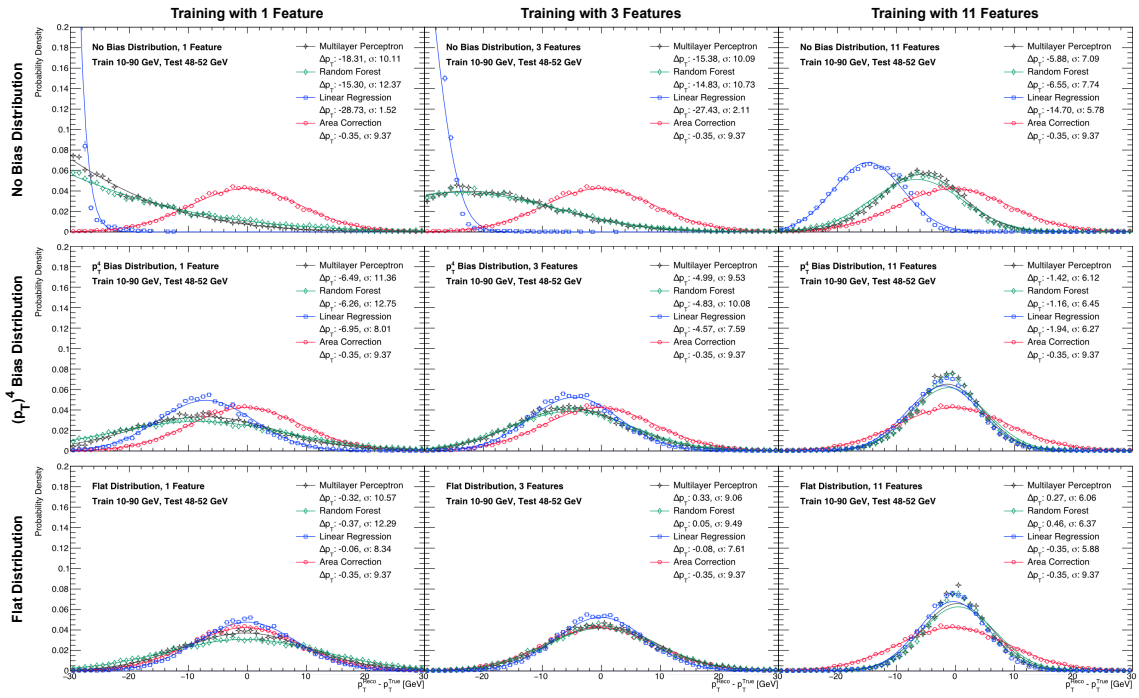

Figure 16: Training with a variety of distributions and features and testing with a flat distribution. Each row is trained on data with a different $p_{\mathrm{T}}^{\mathrm{True}}$ bias: Row 1 uses the no bias distribution. Row 2 uses the p_{T}^4 bias. Row 3 uses the flattened distribution. Each column is trained on a different number of features: Column 1 uses only $p_{\mathrm{T}}^{\mathrm{Raw}}$, Column 2 uses $p_{\mathrm{T}}^{\mathrm{Raw}}$, jet area, and background p_{T} density (i.e. no constituent particle information), and Column 3 uses 11 features with constituent particle data. (*See Fig. 22 in Section 6 for larger figure.*)

First, we consider the influence of training distribution on reconstruction. The no bias distribution clearly shows a strong offset for all feature sets. While this distribution is closest to what we expect from experimental data, it is poor for training. Note in Fig. 11 that the multiplicity of this distribution spans almost five orders of magnitude, so the majority of the data is associated with $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ in 10-20 GeV. There are very few high-p_{T} jets, with some 1 GeV bins having less than 10 jets. The linear regression fits primarily to the large number of low-$p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ jets, so features with smaller coefficients in our baseline model (Table 2) will be disproportionately corrected to a lower $p_{\mathrm{T,\,Jet}}$ value. The random forest and multilayer perceptron estimators are less clear, but we can assume they are optimizing for low-$p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ values and incorporating some high-$p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ data[16]. This could be investigated further by training on distributions with missing bins.

[16]The random forest reconstructed distribution is actually somewhat "lumpy" for the no bias training distribution, though this is clearer in other plots that were not included. This is likely due to it fitting with coarser bins at high-$p_{\mathrm{T,\,Jet}}$.

Due to the poor performance of the no bias training distribution, we will only consider the p_{T}^4 and flat distributions for the remainder of this analysis.

The p_{T}^4 bias underestimates for all feature sets and with estimators, but the widths are only slightly wider than for the flat distribution. It also shows agreement in offset across all three estimators. The flat distribution appears to have the best performance overall with the narrowest widths and smallest offsets for each category. However, we are evaluating the midpoint of the flat distribution so this might be expected (or a coincidence). Later in this analysis, we consider reconstruction performance at other p_{T} ranges to properly evaluate the flat training distribution.

Next, we examine the effects of the feature sets used for training. The 3 feature set improves the width for p_{T}^4 bias and flat distributions, but the p_{T}^4 model still estimates low, and has slightly wider widths than the area-correction method. Note how the 3 feature set with a flat training distribution almost perfectly matches the area correction method. Since this feature set contains the information used in the area-correction calculation, this result is not surprising. If anything, this demonstrates the efficacy of the area-correction method given these jet observables. Interestingly, with 3 features the linear regression model shows narrower widths than the three other models, including area-correction. This may be an artifact of the y-intercept term being a good average, but should be investigated further.

With the 11 feature set, which includes jet constituent information, both the offset and distribution width are reduced for all three estimators, improving them above the area-correction method. The flat distribution with 11 features has the smallest offset and the narrowest widths for all three estimators with a resolution of roughly two-thirds that of the area-correction method. However, this is still short of the resolution improvement of one-half shown in the referenced paper.

We now consider machine learning reconstruction for additional narrow $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ bins. Fig. 17 shows the results for p_{T}^4 bias and flat training distributions using only the 11 feature set and testing closer to the minimum and maximum of our training range ($10\ \mathrm{GeV} < p_{\mathrm{T,\,Jet}}^{\mathrm{True}} < 90\ \mathrm{GeV}$). Our testing bins are: 28-32 GeV, 48-52 GeV (the testing bin used in Fig. 16), and 68-72 GeV[17]. Performance is generally similar for random forest regression and multilayer perceptron regression, while linear regression is more susceptible to the distributions. The no bias distribution is included, but will not be discussed further.

[17]Tests are performed on seven 4 GeV wide bins centered on 20, 30, ... 80 GeV. The lowest bin is 18-22 GeV and the highest is 78-82 GeV. However, these bins show strong skewing and asymmetric reduction in width, indicating potential edge effects. Further investigation is needed, and care must be taken if using machine learning near the limits of the training data.

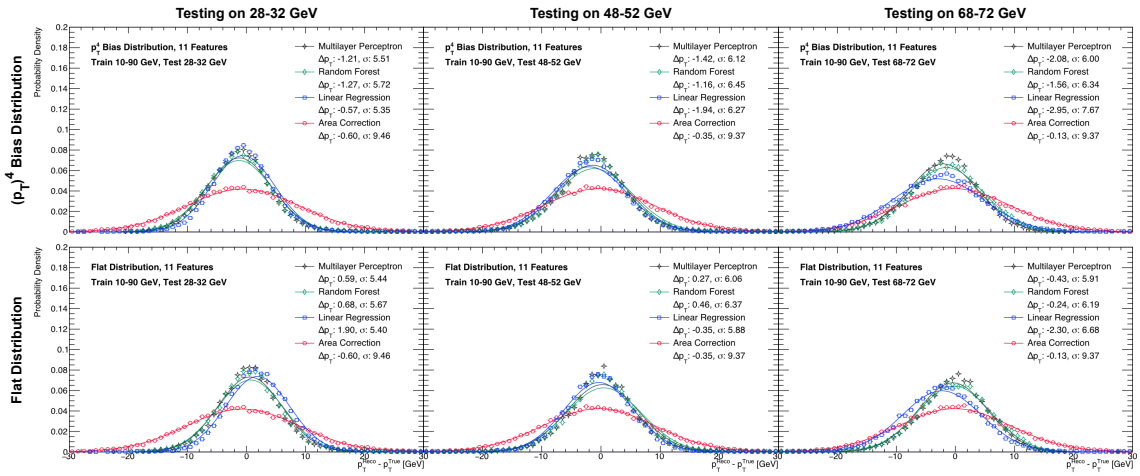

Figure 17: Testing p_{T}^4 bias (top) and flat (bottom) distributions with 11 features across three bins: 28-32 GeV, 48-52 GeV (the center of the training range), and 68-72 GeV.
(*See Fig. 24 in Section 6 for larger figure.*)

Random forest and multilayer perceptron regression perform nearly equally for both distributions and all test bins. However, the p_{T}^4 bias distribution gives low estimates at all ranges, while the flat distribution estimates higher for the 28-32 GeV bin and is nearly centered for the two higher bins. It is also consistently narrower (though only slightly) compared to the p_{T}^4 bias distribution. These small differences indicate that the underlying distribution *does* affect the quality of the estimator. The p_{T}^4 bias distribution has plenty of statistics at high-p_{T} yet still exhibits a larger offset than the flat distribution at all tested ranges. However, for a sufficiently "flat" distribution the significance of these differences is debatable.

We include Fig. 18 to address any concerns that training and testing on the same type of distribution is the reason the flat $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ distribution performs better than others. The no bias distribution is lacking statistics at higher-p_{T}, but the reconstructed distributions remain centered. The p_{T}^4 distributions are nearly identical to the bottom row of Fig. 17.

In summary, the flat training distribution with 11 features displays the best machine-learning-based reconstruction performance, though improvements over the p_{T}^4 bias are small. It remains nearly centered across the full training range and demonstrates a consistent improvement over the area-correction method.

4.4 Linear Regression Analysis

Though linear regression is the simplest of the estimators, this simplicity provides complete transparency through the regression coefficients. As shown in Fig. 16 and Fig. 17, lin-

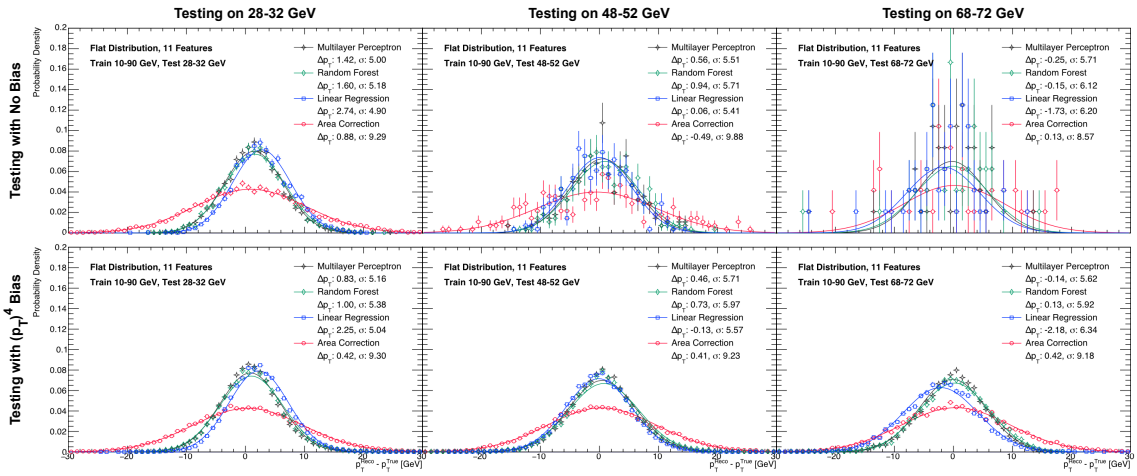

Figure 18: Testing estimators trained on a flat distribution with 11 features. Testing with no bias distribution (top row) and with p_{T}^4 bias distribution (bottom row) across three bins: 28-32 GeV, 48-52 GeV, and 68-72 GeV.

ear regression has nearly identical performance to random forest and multilayer perceptron methods when trained with a flat or p_{T}^4 bias distribution and using 11 features. Therefore, we can use linear regression to gain some insight into what the other estimators may be doing.

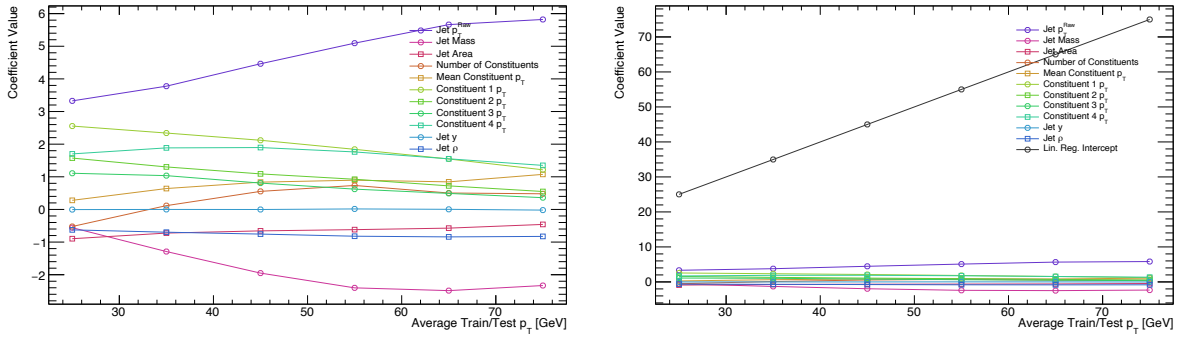

Figure 19: Linear regression coefficient values from training on 30 GeV-wide bins in $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ [18]. The horizontal axis is the central $p_{\mathrm{T,\,Jet}}$ of each bin, while the vertical axis is the coefficient value. For example, the first bin is for $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ in 10-40 GeV, centered on 25 GeV.
(*See Fig. 25 in Section 6 for larger figure.*)

Fig. 19 shows the linear regression coefficients for our baseline model[19]. Since the linear regression function has a form of $y = a_0 + a_1x_1 + a_2x_2...$, we can infer the order of the

[18]The 30 GeV training and testing process uses six bins in $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$: 10-40 GeV, 20-50 GeV, 30-60 GeV, 40-70 GeV, 50-80 GeV, and 60-90 GeV.

relationships between each feature and $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ based on the coefficient line. A flat line with constant coefficient a indicates a truly linear/first order relationship: $y = ax$ $(+\ b)$. As feature x increases, y increases by the same factor, potentially with some offset b. A straight line with *nonzero* slope therefore indicates a second-order relationship[20], $y = ax^2$ $(+\ bx\ +\ c)$, where the factor by which y increases (or decreases) also increases with x. Alternatively, one can think of this as a multiplication of two linear relationships: $y = a(x)\cdot x$ where $a(x) = a'x$. Extending this further, curved lines suggest higher order relationships.

We observe that few of the coefficients exhibit a truly linear behavior, since a linear relationship must have a constant coefficient over the full training range of 10-90 GeV (i.e. it must be flat). This explains why the $p_{\mathrm{T,\,Jet}}^{\mathrm{Reco,\,LR}} - p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ distributions show greater offsets when further from the mean of the training distribution. Machine learning specializes in establishing nonlinear relationships between features, so the small improvements in estimation from the multilayer perceptron and random forest are likely due to this ability to fit nonlinear behaviors.

The second plot of Fig. 19 shows how linear regression can nearly match more advanced forms of machine learning. The coefficients for each feature are small compared to the y-intercept term, so any nonlinear relationships are a comparatively small correction to the linear regression function for the full training range. Though the intercept term increases linearly with $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ in this exercise, this behavior is handled by the $p_{\mathrm{T,\,Jet}}^{\mathrm{Raw}}$ coefficient for linear regression trained on the whole 10-90 GeV range as seen by the large value for $p_{\mathrm{T,\,Jet}}^{\mathrm{Raw}}$ in Table 2.

Some observations on these relationships, as shown in Fig. 20, are given below.

Notable First Order Feature Coefficients

Jet rapidity, y, and background p_{T} density, ρ_{Event}, have nearly flat coefficients. y should be symmetric about $y = 0$ and thus the coefficient is not just flat but *is* almost 0 (this can be seen in Fig. 21 in Section 6). If we were to split our data by positive and negative rapidity, we expect this term to vary, but this will have to be explored as a future project.

[19] Per-jet values for each feature are centered on zero and re-scaled to have unit standard deviation prior to being applied to machine learning. Thus these coefficients represent the *relative strength* of deviation from the mean for each feature.

[20] The y-intercept value is *not* a second-order relationship, since it has no associated feature.

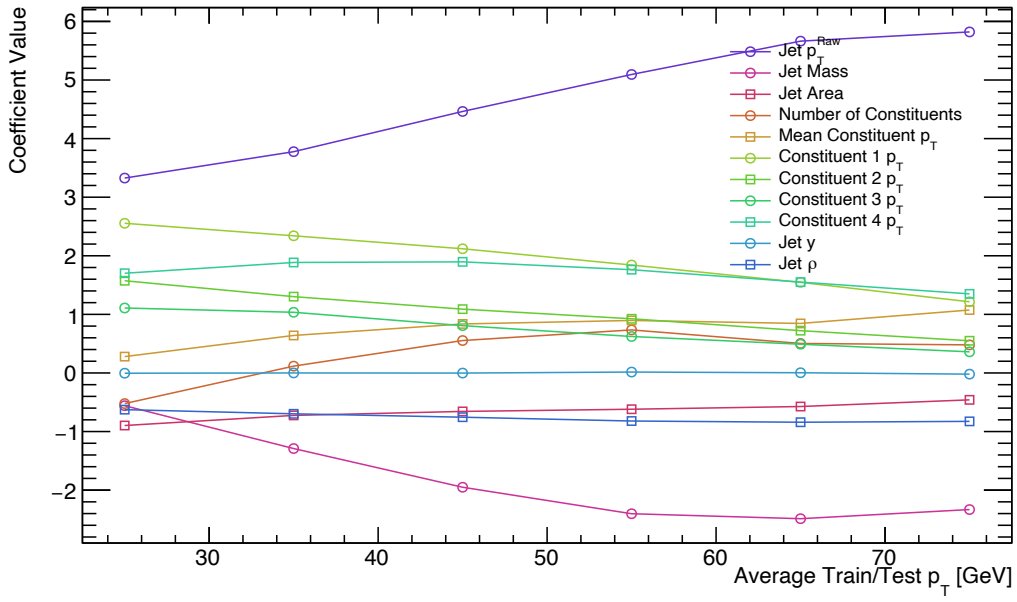

Figure 20: The left plot of Fig. 19 enlarged for clarity.

Since ρ_{Event} is generated randomly for our toy model, this should also be essentially flat, as we expect most events to have dominant truth jets within the two highest-p_{T} jets which are excluded from the ρ_{Event} calculation (see Section 3.6).

Notable Second Order Coefficients

The coefficients for $p_{\text{T, Const. 1}}$, $p_{\text{T, Const. 2}}$, and $p_{\text{T, Const. 3}}$ are linear, indicating a quadratic relationship. These have decreasing but positive slopes with increasing $p_{\text{T, Jet}}^{\text{True}}$, suggesting that as the momentum of each particle increases, their contribution to reconstruction decreases. The total jet momentum $p_{\text{T, Jet}}^{\text{Raw}}$ likely includes a larger fraction of these particles at higher-p_{T} ranges, so this makes sense.

Notable Higher Order Coefficients

The $p_{\text{T, Jet}}^{\text{Raw}}$ coefficient is almost linear, but seems slightly asymptotic towards high- and low-$p_{\text{T, Jet}}$. This indicates that the $p_{\text{T, Jet}}^{\text{Raw}}$ to $p_{\text{T, Jet}}^{\text{True}}$ relationship is at least quadratic, but is probably more nuanced. We know the area-correction method includes A_{Jet} and ρ_{Event} to improve this relationship, and we can see that the tails of $p_{\text{T, Jet}}^{\text{Raw}}$ are the roughly inverse of the tails of A_{Jet}. Also note that both A_{Jet} and ρ_{Event} are negative for all $p_{\text{T, Jet}}^{\text{True}}$, which seems suggestive of the area-correction relationship, since the combination of A_{Jet} and ρ_{Event} are subtracted. Referring back to Fig. 16, we see that linear regression outperforms area-correction with these features (middle column, third row). It seems there may be more to

this, but this is likely due to the intercept term. The linear regression coefficients for training with 3 features is included in Table 3.

Training Features		Lin. Regression Coefficient
Raw Momentum	$p_{\text{T, Jet}}^{\text{Raw}}$	22.130
Background p_{T} Density	ρ_{Event}	-4.363
Jet Area	A_{Jet}	-4.038
(y-Intercept)		50.013

Table 3: Linear regression coefficients for training with a flat $p_{\text{T, Jet}}^{\text{True}}$ distribution and 3 features.

We also see curves associated with coefficients for jet mass, m_{Jet}, the number of constituent particles, $N_{\text{Const.}}$, and average constituent momentum, $p_{\text{T, Const.}}^{\text{Mean}}$.

The m_{Jet} curve seems almost parabolic, which suggests a third-order relationship between m_{Jet} and $p_{\text{T, Jet}}^{\text{True}}$. The strong negative values also indicate that higher values of m_{Jet} are inversely related to $p_{\text{T, Jet}}^{\text{True}}$.

$N_{\text{Const.}}$ is the only coefficient to cross 0, being negative for $p_{\text{T, Jet}}^{\text{True}} \lesssim 35$ and positive for $p_{\text{T, Jet}}^{\text{True}} \gtrsim 35$, and it starts to flatten at high-p_{T}. This suggests a larger-than-average $N_{\text{Const.}}$ can indicate either low-$p_{\text{T, Jet}}^{\text{True}}$, likely for lower $p_{\text{T, Jet}}^{\text{Raw}}$ jets, or a high-$p_{\text{T, Jet}}^{\text{True}}$ for higher $p_{\text{T, Jet}}^{\text{Raw}}$ jets.

Finally, $p_{\text{T, Const.}}^{\text{Mean}}$ has a similar shape to $N_{\text{Const.}}$ but is always positive. Therefore, $p_{\text{T, Const.}}^{\text{Mean}}$ is less relevant in low-$p_{\text{T, Jet}}^{\text{True}}$ jets and is a stronger predictor in high-$p_{\text{T, Jet}}^{\text{True}}$ jets.

The determination of these relationships is essentially what the machine learning is attempting to do, though it uses trial-and-error with some guidance on where to start based on the type of estimator. This methodology of identifying relationships from linear regression may be useful for improving $p_{\text{T, Jet}}$ reconstruction methods, or perhaps for determining an optimal estimator for this particular problem.

5 Discussion

5.1 Final Thoughts

While machine learning methods have promised the ability to find unique and subtle connections between features, their opacity is a risk. We do not know what the random forest or multilayer perceptron methods are actually doing, only that they are somehow improving the resolution over the area-correction method. The feature importance values from random forests offer a some insight, but at best these indicate features to investigate further and their potential to improve reconstruction.

The power of linear regression must not be overlooked. While it is more susceptible to the training distribution, it has nearly identical performance to the machine learning types for reasonably flat distributions (i.e. the p_{T}^4 bias and flat distributions). It can be trained in seconds on large volumes of data, and is thus conducive to iteration and experimentation. Furthermore, training with linear regression over smaller bins can reveals deeper relationships between $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ and each feature. These relationships can be directly studied and quickly tested with different feature sets.

5.2 Conclusion

This investigation demonstrates that the training distribution does affect the quality of the reconstruction. If a $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ distribution has reasonably high statistics over the entire range, as with the p_{T}^4 bias, the effects of the distribution are lessened. Still, we find that the flat distribution is more centered for all tested $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ bins, and has slightly narrower widths for all estimators.

It is clear that constituent features offer significant improvement to the reconstruction resolution. With three jet features, $p_{\mathrm{T,\,Jet}}^{\mathrm{Raw}}$, A_{Jet}, and ρ_{Event}, we are able to reconstruct about as well as the area-correction method. Once constituent particle information is included, we are able to improve our resolution substantially achieving roughly 2/3 of the area-corrected width. However, this is not as strong as the halved resolution shown in the reference paper [1].

We do not see that machine learning – such as the random forest or multilayer perceptron algorithms used in this study – provides a significant benefit for jet momentum reconstruction, particularly when compared to linear regression. Machine learning is still a relatively new tool in heavy ion physics, but that does not require us to embrace it for every application. The inability to see exactly how an estimator works, and which relationships it is building, makes its implementation risky, particularly in contexts where the physics are still

developing. Furthermore, if machine learning is used on experimental data, the opacity of these tools makes error propagation difficult.

In contrast, linear regression is valuable *because* of its transparency and simplicity. It performs as well as machine learning in many cases. We can see directly which relationships are being built and use those to improve existing reconstruction algorithms, if desired. If applied to experimental data, the determination of systematic errors for linear regression reconstruction should be straightforward. Additionally, the simplicity of linear regression makes it easy to use and computationally quick to iterate with. We can use techniques such as training over smaller bins to explore relationships between features without needing machine learning.

That being said, linear regression has limitations. Its linearity makes fitting to certain feature sets impossible, such as those with symmetric distributions. It is also unable to combine features into new relationships. Using linear regression *in combination with* an estimator that outputs feature-specific metrics – like feature importance scores from random forest regression – may prove more fruitful than using a single tool.

Ultimately, we are trying to find deep connections in our data that will improve our physics. Though machine learning is built to make those connections and improve the output, the real value for us as physicists is understanding the deeper relationships. We can use any of these tools to further our research, but we should be careful that the tools not distract us from what we are really trying to accomplish.

5.3 Looking Forward

This investigation has been valuable but it is not complete. We propose two additional steps: training with improved collision models, and evaluating the differences in jet p_{T} spectra after unfolding.

We suggest training linear regression and machine learning estimators with a collision model that includes jet quenching effects and underlying event fluctuations. This will demonstrate if linear regression is still a valid substitute to machine learning when using realistic data. This also presents an opportunity to test machine learning models that are better suited to this particular problem, as we suspect that random forest and multilayer perceptron estimators may not be the best for this data set. Since this investigation has already highlighted several nonlinear relationships, using a polynomial regression estimator may provide additional insight as an immediate next step.

To properly evaluate these methods, one must consider the performance of the total workflow, including unfolding. We suggest comparing jet spectra after unfolding on data reconstructed with area-correction, linear regression, and select machine learning methods. Area-correction provides a baseline for broad method comparisons, while linear regression and machine learning can be directly compared. Jet momentum spectra can also be compared, in addition to the minimum number of iterations required for effective unfolding.

Finally, while machine learning may not be necessary for large improvements to jet momentum reconstruction, there are other areas within heavy ion and jet physics in which machine learning could be valuable. We expect many regression problems to present similar challenges to those seen here, since many seek to reconstruct a "true" value from noisy observables. Classification problems, such as the identification of true jets or of jet types, may be better avenues to explore.

5.4 Acknowledgements

Special thanks to the University of Colorado Boulder Undergraduate Research Opportunities Program (UROP) for partially funding this research, as well as to the family of Herbert E. Bowman for their generous scholarship.

This work was supported in large part by Joseph Clement, University of Colorado Boulder, who explored parts of this investigation in parallel to confirm results, and spent *many* hours helping me debug my code and improve my workflow.

I also wish to acknowledge the people whose research projects and conversations about machine learning helped me develop this investigation: Tanner Mengel and Prof. Christine Nattrass at the University of Tennessee, as well as Hannah Bossi and Dr. Isaac Mooney at Yale University.

6 Appendix

6.1 Baseline Model: Additional Figures and Tables

Feature Importance Scores and Coefficients

Training Features		Feature Importance	Lin. Regression Coefficient
1 Feature			
Raw Momentum	$p_{\text{T, Jet}}^{\text{Raw}}$	1.0	20.137
(y-Intercept)*		–	50.013
3 Features			
Raw Momentum	$p_{\text{T, Jet}}^{\text{Raw}}$	0.872	22.130
Background p_{T} Density	ρ_{Event}	0.093	-4.363
Jet Area	A_{Jet}	0.035	-4.038
(y-Intercept)*		–	50.013
11 Features			
Raw Momentum	$p_{\text{T, Jet}}^{\text{Raw}}$	0.742	16.288
Background p_{T} Density	ρ_{Event}	0.008	-1.350
Jet Area	A_{Jet}	0.005	-1.428
Jet Mass	m_{Jet}	0.009	-2.354
Jet Rapidity	y_{Jet}	0.015	-0.007
Constituents in Jet	$N_{\text{Const.}}$	0.003	-1.801
Constituent Mean p_{T}	$p_{\text{T, Const.}}^{\text{Mean}}$	0.149	1.133
Constituent 1 p_{T}	$p_{\text{T, Const. 1}}$	0.015	3.561
Constituent 2 p_{T}	$p_{\text{T, Const. 2}}$	0.022	1.928
Constituent 3 p_{T}	$p_{\text{T, Const. 3}}$	0.017	1.341
Constituent 4 p_{T}	$p_{\text{T, Const. 4}}$	0.015	3.553
(y-Intercept)*		–	50.013

Table 4: Table of features in each feature set. Feature importance is determined while training random forest regression. Linear regression coefficients are determined from training linear regression, and includes a y-intercept term.

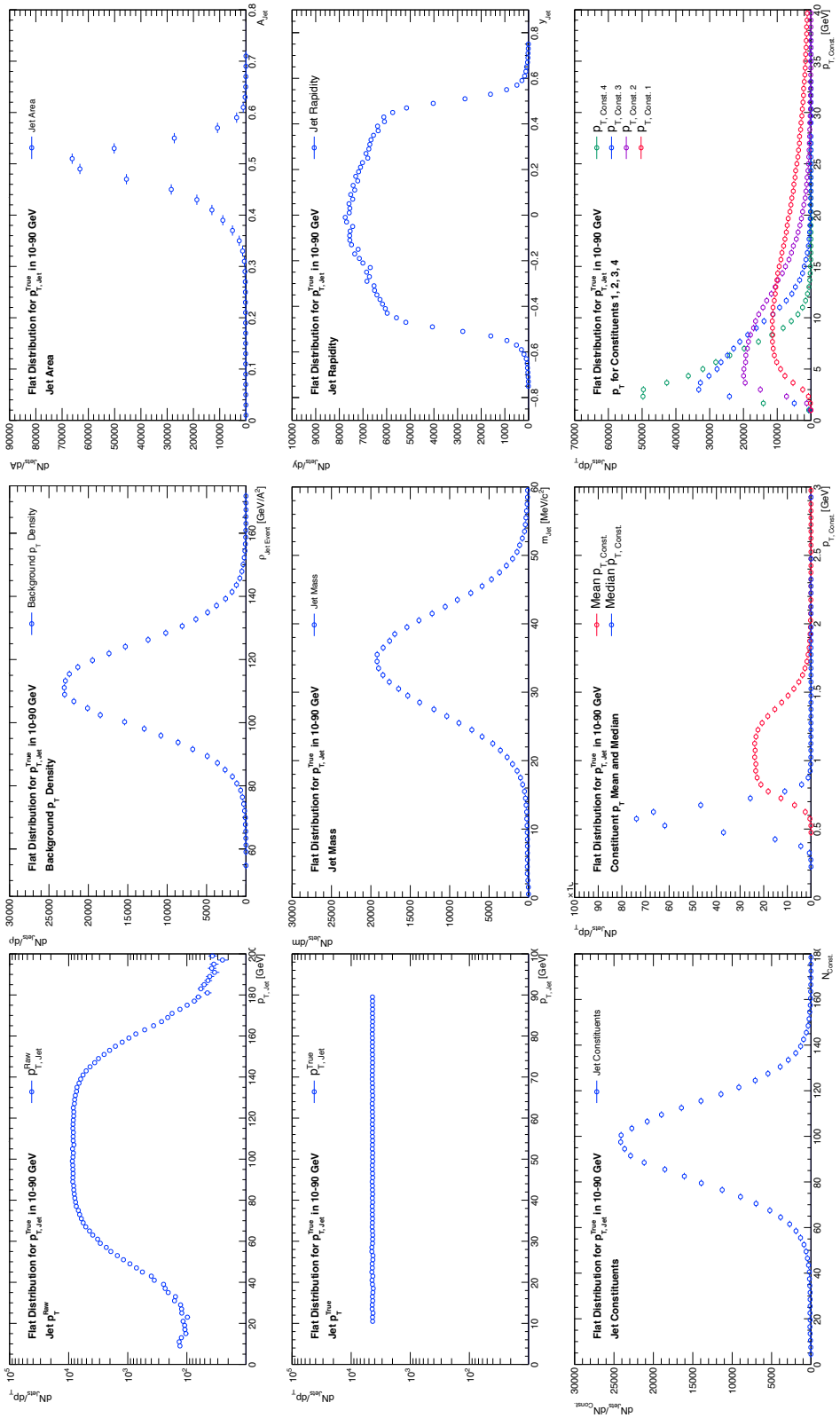

Figure 21: Distributions of input features for the baseline model (flat $p_{\mathrm{T,\,Jet}}^{\mathrm{True}}$ distribution). The constituent particle p_{T} values are included in one plot, and both $p_{\mathrm{T,\,Const.}}^{\mathrm{Mean}}$ and $p_{\mathrm{T,\,Const.}}^{Median}$ are included in one plot.

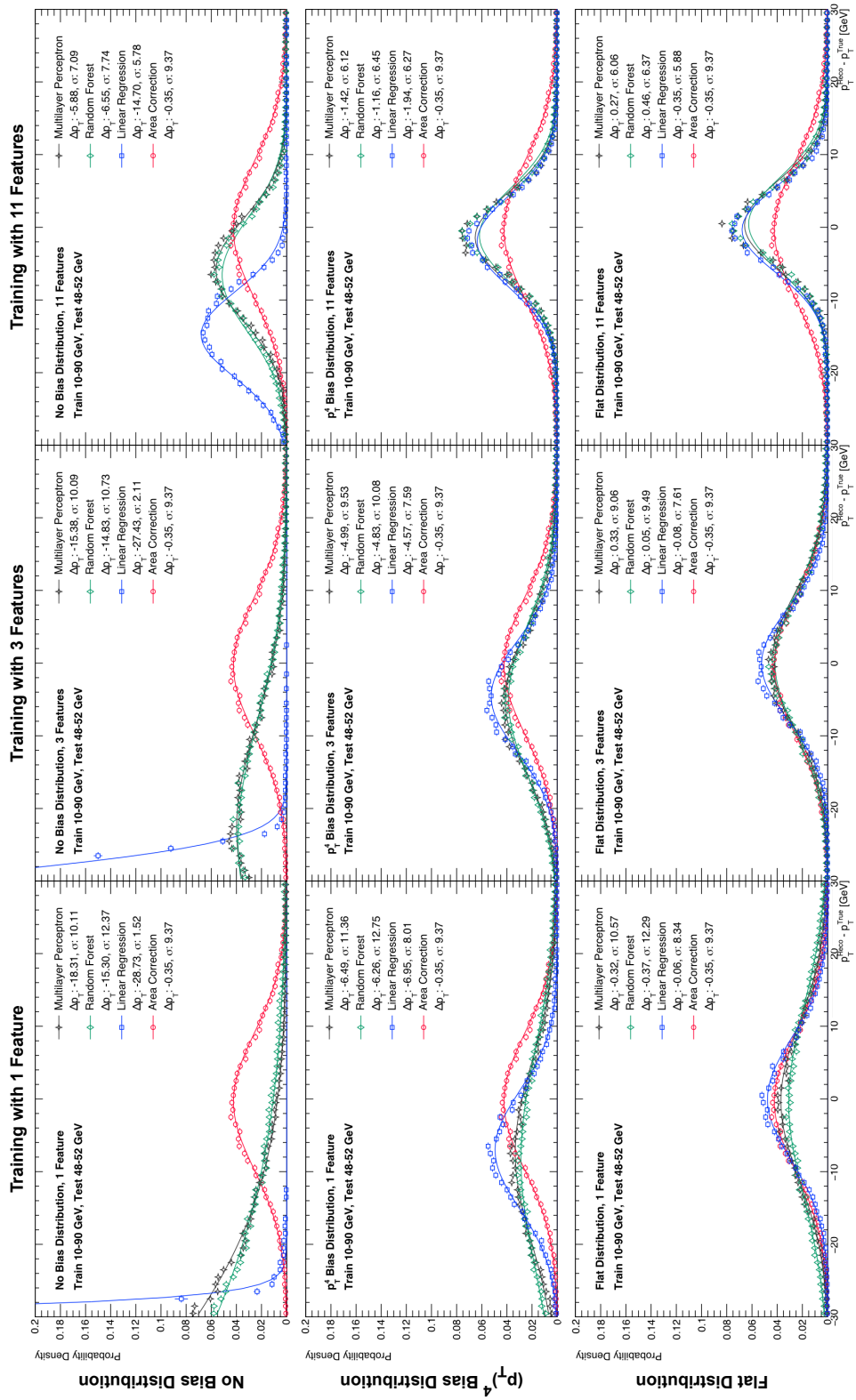

Figure 22: Training with a variety of distributions and features and testing with a flat distribution. Each row is trained on data with a different $p_{\mathrm{T}}^{\mathrm{True}}$ bias: Row 1 uses the no bias distribution. Row 2 uses the p_{T}^4 bias. Row 3 uses the flattened distribution. Each column is trained on a different number of features: Column 1 uses only 1 feature, Column 2 uses 3 features, and Column 3 uses 11 features.

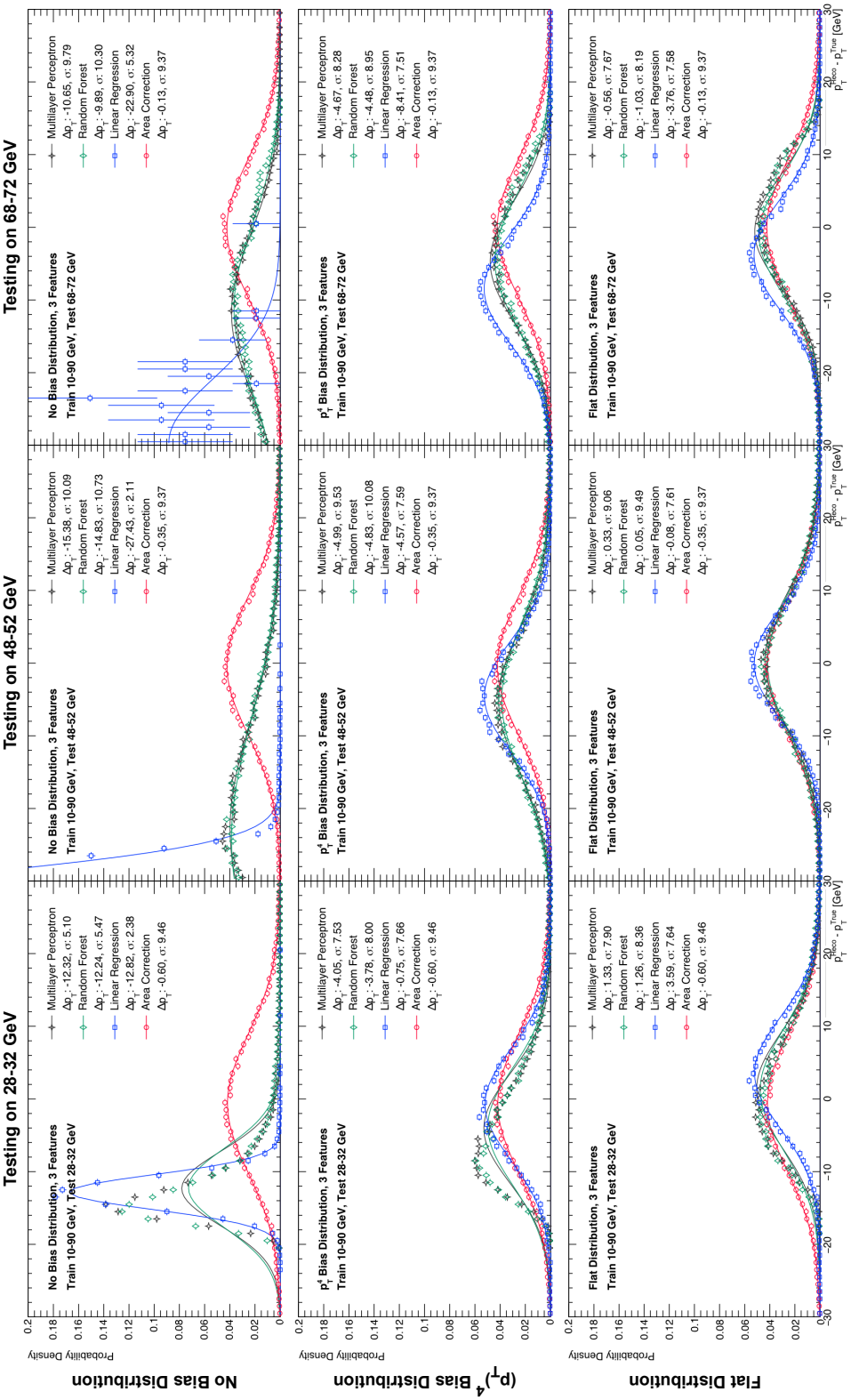

Figure 23: Reconstruction results using 3 features for all three distributions: $p_{\mathrm{T}}^{\mathrm{Raw}}$, jet area, and background p_{T} density (i.e. no constituent particle information). Each row is trained on data with a different $p_{\mathrm{T}}^{\mathrm{True}}$ bias: Row 1 uses the no bias distribution. Row 2 uses the p_{T}^4 bias. Row 3 uses the flattened distribution.

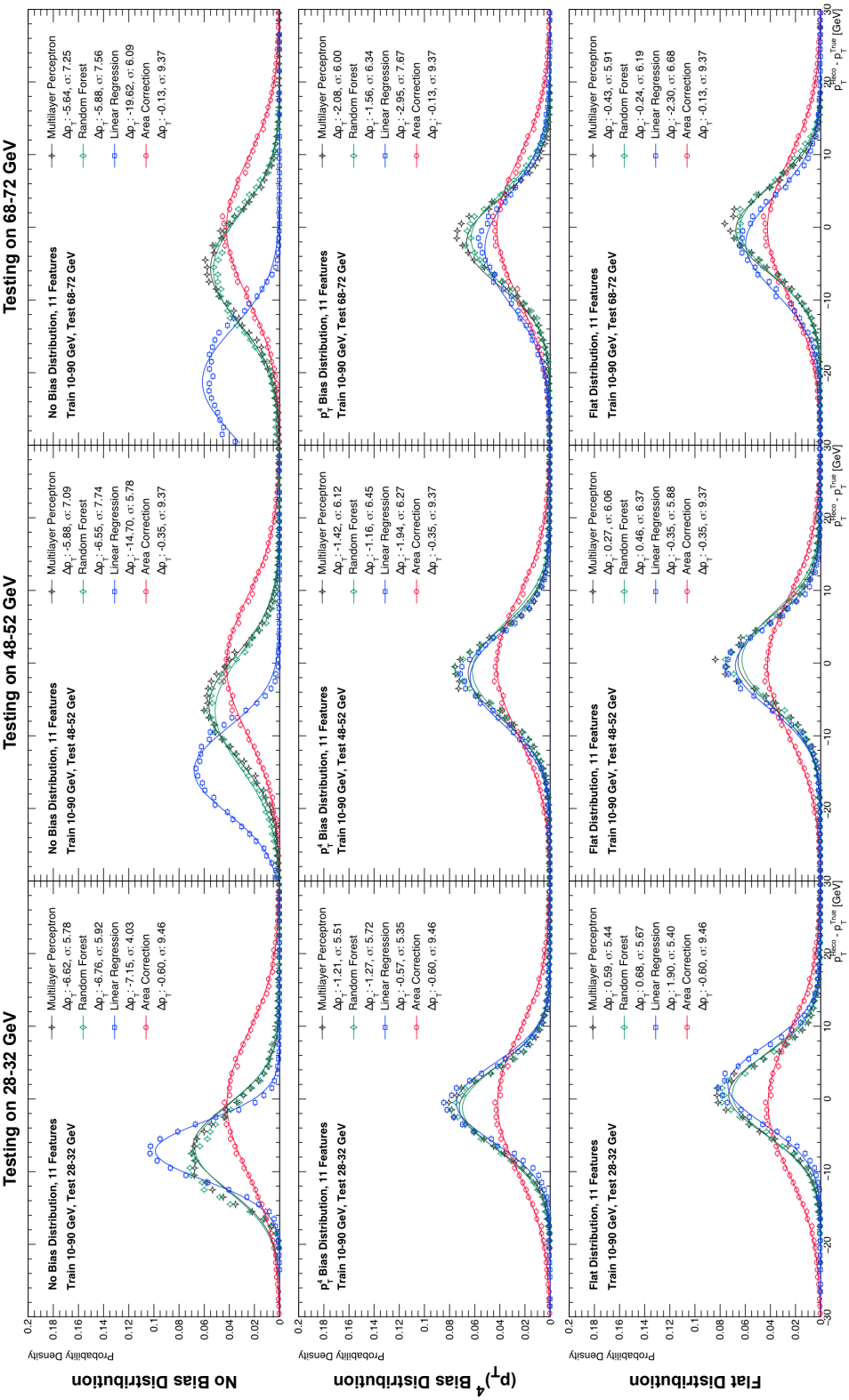

Figure 24: Reconstruction results using 11 features for all three distributions, with features containing constituent particle information. Each row is trained on data with a different $p_{\mathrm{T}}^{\mathrm{True}}$ bias: Row 1 uses the no bias distribution. Row 2 uses the p_{T}^4 bias. Row 3 uses the flattened distribution.

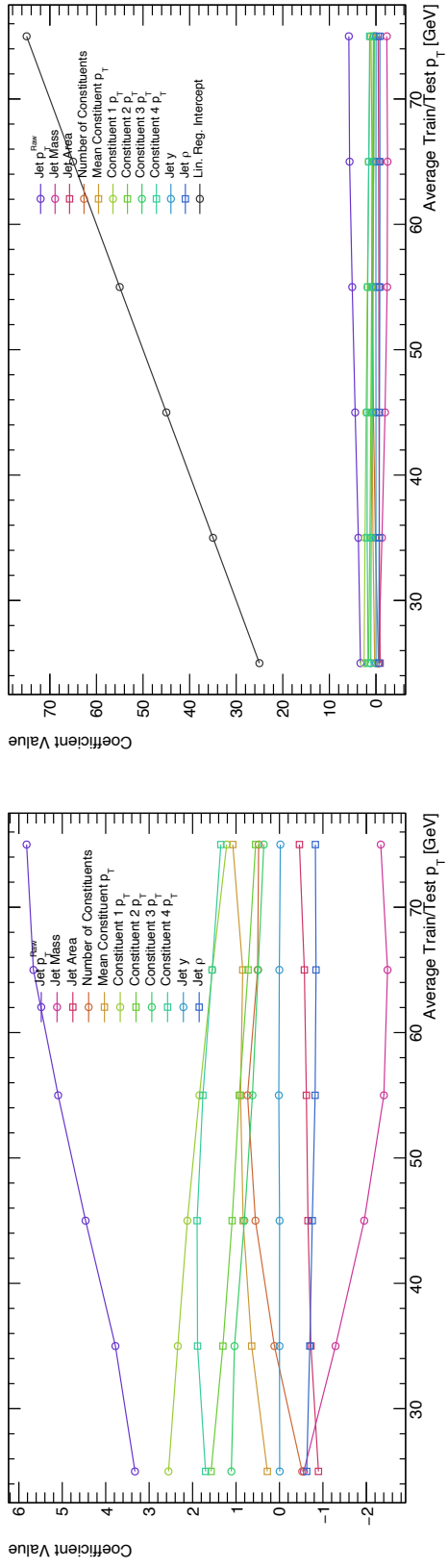

Figure 25: Linear regression coefficient values from training on 30 GeV-wide bins in $p_{T,\,\mathrm{Jet}}^{\mathrm{True}}$. The horizontal axis is the central $p_{T,\,\mathrm{Jet}}^{\mathrm{True}}$ value for each training bin, while the vertical axis is the coefficient value.

6.2 Project GitHub Link

The work for this thesis project is preserved on GitHub in the following repository:
https://github.com/jordandlang-phys/CU-Heavy-Ions-JLangUndergradThesis-Backup.git

This code was built in C++, with the ROOT [22], PYTHIA [25], and FASTJET [26] packages, and in Python with the scikit-learn library [19] for machine learning implementation. Additional details for installing and running code can be found in the GitHub repository above.

Note that raw data files (e.g. `.root` and `.csv` files) are not included in the repository due to the large files sizes, but can be generated locally using the tools above.

References

[1]R. Haake and C. Loizides, "Machine-learning-based jet momentum reconstruction in heavy-ion collisions", Physical Review C **99**, 10.1103/physrevc.99.064904 (2019).

[2]W. Busza, K. Rajagopal, and W. van der Schee, "Heavy Ion Collisions: The Big Picture and the Big Questions", Annual Review of Nuclear and Particle Science **68**, 339–376 (2018).

[3]D. J. Griffiths, *Introduction to elementary particles; 2nd rev. version*, Physics textbook (Wiley, New York, NY, 2008).

[4]Wikipedia contributors, *Pseudorapidity — Wikipedia, the free encyclopedia*, [Online; accessed 27-February-2023], (2023) https://en.wikipedia.org/wiki/Pseudorapidity.

[5]Wikipedia contributors, *Rapidity — Wikipedia, the free encyclopedia*, [Online; accessed 09-March-2023], (2023) https://en.wikipedia.org/wiki/Rapidity.

[6]ALICE Collaboration, *The ALICE experiment – A journey through QCD*, 2022.

[7]F. Wilczek, "QCD Made Simple", Physics Today **53**, 22–28 (2000).

[8]R. L. Workman et al. (Particle Data Group), "Review of Particle Physics", PTEP **2022**, 083C01 (2022).

[9]J. O. Koop, "On Kinetic Transport in Small System Collectivity and a Measurement of Separated Open Heavy Flavor Production in $p+p$ Collisions at $\sqrt{s_{NN}} = 200$ GeV", PhD thesis (University of Colorado Boulder, 2018).

[10]G. P. Salam, "Towards jetography", The European Physical Journal C **67**, 637–686 (2010).

[11]CDF Collaboration, "Charged jet evolution and the underlying event in proton-antiproton collisions at 1.8 TeV", Phys. Rev. D **65**, 092002 (2002).

[12]A. Lopes, *Recreating Big Bang Matter on Earth*, (Dec. 2020) https://phys.org/news/2020-12-recreating-big-earth.html (visited on 03/10/2023).

[13]P. Braun-Munzinger, K. Redlich, and J. Stachel, "PARTICLE PRODUCTION IN HEAVY ION COLLISIONS", in *Quark–gluon plasma 3* (WORLD SCIENTIFIC, Jan. 2004), pp. 491–599.

[14]Brookhaven National Lab, *The Physics of RHIC*, (2023) https://phys.org/news/2020-12-recreating-big-earth.html (visited on 03/10/2023).

[15]L. Cunqueiro and A. M. Sickles, "Studying the QGP with Jets at the LHC and RHIC", Progress in Particle and Nuclear Physics **124**, 103940 (2022).

[16]M. Cacciari, G. P. Salam, and G. Soyez, "The anti-k_t jet clustering algorithm", 10.1088/1126-6708/2008/04/063 (2008).

[17]E. Alpaydin, *Machine Learning* (The MIT Press, Cambridge, MA, 2021).

[18]A. Smola and S. Vishwanathan, *Introduction to Machine Learning* (Cambridge University Press, Cambridge, UK, 2008).

[19]F. P. et al., "Scikit-learn: Machine Learning in Python", Journal of Machine Learning Research **12**, 2825–2830 (2011).

[20]Wikipedia contributors, *Random forest — Wikipedia, the free encyclopedia*, [Online; accessed 18-March-2023], (2023) https://en.wikipedia.org/wiki/Random_forest.

[21]Wikipedia contributors, *Artificial neural network — Wikipedia, the free encyclopedia*, [Online; accessed 18-March-2023], (2023) https://en.wikipedia.org/wiki/Artificial_neural_network.

[22]R. Brun and F. Rademakers, "ROOT - An Object Oriented Data Analysis Framework", in Proceedings aihenp'96 workshop, lausanne, Vol. 389 (1996), pp. 81–86, See also "ROOT" [software], Release v6.26/04 - 2022-06-07.

[23]C. Bierlich, S. Chakraborty, N. Desai, L. Gellersen, I. Helenius, P. Ilten, L. Lönnblad, S. Mrenna, S. Prestel, C. T. Preuss, T. Sjöstrand, P. Skands, M. Utheim, and R. Verheyen, "A comprehensive guide to the physics and usage of PYTHIA 8.3", 10.48550/ARXIV.2203.11601 (2022).

[24]T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten, S. Mrenna, S. Prestel, C. O. Rasmussen, and P. Z. Skands, "An introduction to PYTHIA 8.2", Computer Physics Communications **191**, 159–177 (2015).

[25]T. Sjöstrand, S. Mrenna, and P. Skands, "PYTHIA 6.4 physics and manual", Journal of High Energy Physics **2006**, 026–026 (2006).

[26]M. Cacciari, G. P. Salam, and G. Soyez, "FastJet user manual", 10.1140/epjc/s10052-012-1896-2 (2021).

[27]Q. Wang, P.-P. Yang, and F.-H. Liu, "Comparing a few distributions of transverse momenta in high energy collisions", Results in Physics **12**, 259–267 (2019).

[28]ALICE Collaboration, "Suppression of charged particle production at large transverse momentum in central Pb-Pb collisions at $\sqrt{s_{\mathrm{NN}}} = 2.76$ TeV", Physics Letters B **696**, 30–39 (2011).

[29]ALICE Collaboration, *Suppression of charged particle production at large transverse momentum in central Pb-Pb collisions at* $\sqrt{s_{NN}} = 2.76$ *TeV* (HEPData).

[30]T. Kluyver et al., "Jupyter Notebooks – a publishing format for reproducible computational workflows", in Positioning and power in academic publishing: players, agents and agendas, edited by F. Loizides and B. Schmidt (IOS Press, 2016), pp. 87–90.