UNIVERSITY OF CALIFORNIA,
IRVINE

Jet Substructure in the Era of Machine Learning

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Physics

by

Alexis Romero

2023

DEDICATION

To Freja.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Daniel Whiteson, for always encouraging me to pursue my research interests. His commitment to fostering creativity and unique research ambitions in our group continues to inspire many of us. I would also like to thank colleagues who I have had the pleasure to work with, including Timothy Tait (UCI), Tilman Plehn (Heidelberg University), Benjamin Lillard (University of Oregon), Pierre Baldi (UCI), Julian Collado (UCI), Yadong Lu (UCI), and Kyle Cranmer (UW-Madison).

Additional thanks to my friends and office mates who have somehow made grad school life, even on occasion, fun. I am especially grateful to Freida Rivera, Arianna Braconi, Yvonne Ng, Michael Waterbury, Jacob Hollingsworth, Matthew Stanfield, Taylor Faucett, Alex Armstrong, Sunny Yu, and Jordan Smolinsky.

Lastly, I would like to thank Joakim and my cat Pippa for their unconditional support, regardless of how sassy (Pippa) it may be.

Chapter 3 of this dissertation is a reproduction of the material as it appears in the Journal of High Energy Physics, used with permission from Springer [5]. The co-authors listed in this publication are Yadong Lu, Michael James Fenton, Daniel Whiteson, and Pierre Baldi. Daniel Whiteson and Pierre Baldi directed and supervised research which forms the basis for the dissertation.

The work presented in this dissertation is in part supported by the Department of Education GAANN Grant No. P200A150121at UC Irvine, and the National Science Foundation under grant number 1633631.

VITA

Alexis Romero

EDUCATION

Doctor of Philosophy in Physics **2023**
University of California, Irvine *Irvine, CA*

Bachelor of Science in Physics **2016**
San Diego State University *San Diego, CA*

RESEARCH EXPERIENCE

Graduate Research Assistant **2016–2023**
University of California, Irvine *Irvine, California*

TEACHING EXPERIENCE

Teaching Assistant **2016–2021**
University of California, Irvine *Irvine, California*

REFEREED JOURNAL PUBLICATIONS

Resolving Extreme Jet Substructure **2022**
Journal of High Energy Physics

Multi-scale Mining of Kinematic Distributions with Wavelets **2020**
SciPost

ABSTRACT OF THE DISSERTATION

Jet Substructure in the Era of Machine Learning

By

Alexis Romero

Doctor of Philosophy in Physics

University of California, Irvine, 2023

Professor Daniel Whiteson, Chair

This work explores the application of machine learning for multi-prong jet classification at the LHC. We compare the performance of high-level networks trained on jet observables to low-level networks trained on the full event information. Our focus is on an extreme scenario, evaluating the performance of the classifiers on jets with a large number of sub-jets, larger than previously tested in the literature. The results indicate that traditional jet observables like N-subjettiness and Energy-Flow Polynomials are good discriminants when tested on jets with up to eight hard sub-jets, but that there is information in the events that is untapped by these observables. We introduce Jet Rotational Metrics, a new family of observables designed to capture features of the degree of discrete rotational symmetry of jets. These observables prove highly valuable for the classification task, bridging the gap between the low- and high-level networks.

This dissertation also introduces a technique for the accurate estimation of systematic uncertainties in physics studies using Gaussian process regression. A typical approach is to assume the factorization of the various sources of systematic uncertainties. This approach is often extended to assume that the impact of these individual sources of uncertainty on observables of the detector response also factories. Our technique uses Gaussian processes to model observables as functions of the nuisance parameters. We show that this technique is more accurate than the factorized approach and that it can learn from limited samples by including gradient information. Additionally, we present a Bayesian-based sampling strategy that efficiently explores the space of experimental response while reducing the predictive uncertainty of the Gaussian process.

Chapter 1

Introduction

The essence of particle physics lies in answering some of the most fundamental questions about the universe: How many elementary particles are there? What are their properties? And how do they interact with each other? An early answer to these questions was proposed by Empedocles (fl. 444–443 BC), who argued that matter is composed of four elements (fire, air, water, and earth). More recently, John Dalton formulated the first modern atomic theory in the early 19th century, which suggested that matter is made up of discrete, indivisible units called "atoms." While these theories are compellingly simple, they fail to explain experimental observations, which suggest that matter is composed of even smaller elementary particles of a probabilistic nature, interacting through fundamental forces.

Our current best answer to the composition of the universe is the Standard Model (SM), which was proposed in the 1970s and represents the culmination of decades of research in theoretical and experimental physics. The SM has successfully explained the properties of all known particles and their interactions with three of the four fundamental forces (electromagnetic, weak, and strong–excluding gravity). Predictions within the SM framework have been experimentally verified across a wide range of energy scales and with a precision up to the order of parts-per-trillion [6]. However, despite its unprecedented success, the SM comes with known limitations. For example, experimental observations from astronomy and cosmology suggest that all known matter accounts for less than 5% of the universe while the vast majority of it consists of dark matter and dark energy [7, 8], which do not fit into the SM framework. In addition, the matter-antimatter asymmetry of the universe requires the violation of charge and parity (CP) symmetry [9]. While the SM predicts some sources of CP violation, they are not enough to explain the large excess of matter over antimatter. Other limitations include neutrino oscillations, the hierarchy problem, and the integration of the SM framework with general relativity.

The aforementioned limitations hint at the possibility of particles and fields Beyond the SM (BSM). For decades, physicists have formulated theories proposing various extensions to the SM. Many such theories have been found to be self-consistent, but their predictions have yet to be tested. Experiments conducted at particle accelerators have the potential to test these theories by providing access to the rich and generally unexplored territory of high-energy particle collisions.

The state-of-the-art particle accelerator is the Large Hadron Collider (LHC) at CERN. At the LHC, protons are accelerated to energies close to the speed of light and smashed together with center-of-mass energies up to 14 TeV. This energy is converted into mass according to $E = mc^2$, thereby creating new particles. Experiments such as the ATLAS and CMS detectors study the particles resulting from these high-energy collisions to probe the predictions of the SM and to search for signals of new physics. To this day, hundreds of searches for BSM physics have been conducted without much success. This lack of evidence of new physics has partly been attributed to various experimental challenges. Most particles of interest are estimated to last only fractions of a second before decaying, leaving behind only signatures that they were ever created. These signatures are very rare and often buried in a sea of background. For example, the most famous discovery at the LHC is the Higgs boson discovery at the ATLAS and CMS experiments in July 2012. Approximately only one in every billion collisions results in a Higgs boson, from which only one in ten thousand has a signature that is easy to see. Signatures of new physics are believed to be even more elusive, and thus, finding them will demand our best efforts to extract and analyze all relevant information from the data collected at particle accelerators.

To this day, the LHC has only collected a small percentage of the data it is scheduled to collect during its lifetime. In an attempt to address the growing data volume, machine learning (ML) methods have been employed with the promise of handling larger data batches with better accuracy and faster execution times. These methods have been used in many aspects of data analyses, from the first-level hardware trigger [10–12] to event selection and object identification [13–15], and even to enhance detector simulations [16].

This thesis explores the power of machine learning for event classification. The work presented in this thesis addresses the challenges and advantages of the application of machine learning for identifying signals of interest in simulated events. Chapter 2 introduces key concepts of machine learning and their applications to high-energy physics. Chapter 3 compares the performance of various machine learning models for event classification of jets with multiple sub-jets. Chapter 4 expands on the work presented in Chapter 3 by introducing a novel way of calculating event observables designed to classify jets with different topologies. Chapter 5 presents a technique for the accurate estimation of the impact of systematic uncertainties in the calculation of detector response observables using Gaussian Processes. Lastly, Chapter 6 provides a discussion of the work presented in this thesis and its possible extensions.

Chapter 2

Deep Learning and High-Energy Physics

Machine learning (ML) is a branch of artificial intelligence dedicated to building statistical models that can *learn* from data without being explicitly programmed. Physicists have long utilized various traditional machine learning techniques to aid in physics searches. Some of the earliest applications include simple multivariate algorithms, such as support vector machines, random forests, and boosted decision trees [17, 18]. At the time, these tools provided an important boost in the data analysis tasks, but their abilities were limited, mainly due to their poor scaling with the size and dimensionality of the data.

It was not until the early 2010s that advances in graphics processing units (GPUs) allowed for neural networks with multiple layers to train several orders of magnitude faster [19]. These networks, also known as *deep* neural networks, greatly outperformed the previous state-of-the-art algorithms due to their ability to learn from data of higher complexity and dimensionality than previously feasible. Soon after, deep neural networks quickly became mainstream in many research fields, including particle physics.

The integration of deep learning into particle physics has required careful thought when interpreting the results due to the particular nature of the data. There are a few aspects that make data collected at particle detectors unique. The first is the quantum-mechanical nature of the collisions and subsequent interactions between particles and the detector. Quantum mechanics tells us that these interactions are fundamentally probabilistic, and thus, the output of the deep networks must be interpreted as such. For instance, traditional machine learning methods for classification consider $\mathrm{p}(x|\theta)$ to be a statistical model describing the probability of observing label x given the parameter θ. In particle physics, however, assigning labels to individual events is not straightforward. The labels more so refer to the probability of finding certain particles within some small region of phase space. This probability is calculated by various stochastic simulation tools that model the physics behind high-energy particle collisions.

The second aspect that makes particle physics unique is the access to high-fidelity simulations. For decades, experts have developed tools to simulate the progression of particle collisions, from the particles arising during the first instants after the collisions, which are dominated by perturbative quantum field theory (QCD) and can be modeled using first principles, to their subsequent decays into more "stable" particles, which are modeled by semi-classical Markov models [20, 21]. Interactions between these particles and the detector are also modeled, resulting in realistic and remarkably robust simulations of collision events as captured by the detector [22]. The fidelity of these simulations is at the core of most physics searches, where signals and backgrounds are simulated and used to train deep learning models.

2.1 Jets at the LHC

The most common phenomena observed at the LHC are jets. Jets are collimated sprays of hadrons resulting from the hadronization and fragmentation of quarks and gluons produced in hard scattering processes. This behavior is observed directly in experiments where hadronic final state particles appear collimated in a few directions in the detector.

Traditionally, jets have been used as proxies for the quarks and gluons produced during the first instants after high-energy collisions. This makes jets indispensable for studying the interactions between particles produced in such collisions. For example, jets have played a central role in the discovery and measurements of fundamental particles, such as the gluon [23–26] and the top-quark [27, 28]. In addition, because of their large production rates at the LHC, jets are often featured in searches of physics beyond the SM. However, integrating jets into searches of new physics often comes with additional, interesting challenges. Important information of the underlying processes is not only carried by the kinematics of the final state particles but also by the internal composition, known as the *jet substructure*. The study of jet substructure is a relatively new field, which requires an understanding of the phenomenological features of the underlying processes and of the unfolding of the particles as they travel through the detector.

At the detector level, jets are defined as the composite objects resulting from clustering lists of reconstructed particles according to a *jet algorithm* [29]. These particles could be reconstructed based on calorimeter cells (calorimeter jets) or on particle-flow candidates (particle-flow jets) [30, 31]. The latter employs a cell-based energy subtraction algorithm to remove overlaps between the momentum and energy measures made by the tracking detector and calorimeters, respectively. The tracking detector gauges the momentum of charged particles by measuring the bent on their trajectories when subjected to strong magnetic fields. The calorimeters are composed of cells with high-density materials designed to stop incoming particles, thus measuring their energy. The detectors employ two types of calorimeters: electromagnetic and hadronic. The electromagnetic calorimeter measures the energy of charged particles, predominately electrons and photons. The hadronic calorimeter measures the energies of hadronic particles, such as protons and neutrons. Figure 2.1 shows a schematic depiction of various particles traveling through and interacting with the ATLAS detector.

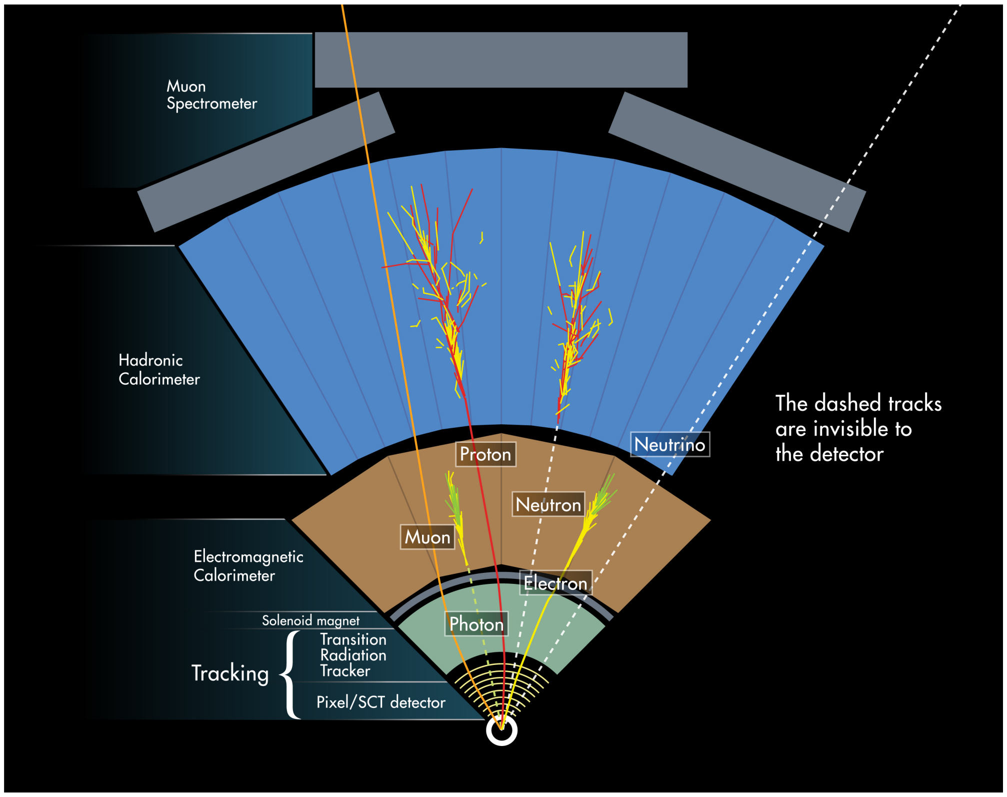


Figure 2.1: Illustration of the transverse plane of the ATLAS detector. The figure shows a schematic depiction of how various particles travel through and interact with the detector. ATLAS Experiment © CERN.

A popular class of jet algorithms is the *sequential recombination* class [32], which is based on the concept that jets are the result of successive QCD branchings. The algorithm thus tries to reverse the branching process by sequentially searching for the pairs of closest particles and “recombining” them into one by adding their four-momenta (E-scheme recombination), until the smallest distance is between a particle and the beam axis. The generalized-k_T algorithm is typically used when measuring the distances between the particles themselves and between a particle and the beam axis:

$$\text{Inter-particle distance:} \qquad d_{ij} = d_{ji} = \min(p_{\mathrm{T}i}^{2p}, p_{\mathrm{T}j}^{2p}) \frac{\Delta R_{ij}^2}{R}, \tag{2.1}$$

$$\text{Particle-beam distance:} \qquad d_{iB} = p_{\mathrm{T}i}^{2p}, \tag{2.2}$$

where $p_{\mathrm{T}i}$ is the transverse momentum of particle i and $\Delta R_{ij}^2 = (y_i - y_j)^2 + (\phi_i - \phi_j)^2$ is the geometric distance between particles i and j in the rapidity-azimuth plane (y, ϕ). The parameter R is the *jet radius*, which controls the angular reach of the jet. Lastly, the exponent p controls the relative weight of the energy terms. Common choices for p include $p = 1$ (k_t algorithm [33]), $p = 0$ (Cambridge-Aachen algorithm [34]), and $p = -1$ (anti-k_t algorithm [1]).

An example of an event clustered with the Cambridge-Aachen and anti-k_t algorithms is shown in Figure 2.2. The particle distance in the Cambridge-Aachen algorithm is only proportional to the geometric distance, resulting in irregular jet shapes. Similarly, the k_t algorithm also often results in irregular jet shapes. On the other hand, the particle distance in the anti-k_t algorithm is inversely proportional to the p_{T}, encouraging high-p_{T} particles to be clustered first. The aggregation of the particles in decreasing p_{T} order generally results in jets with circular shapes, which is why the anti-k_t algorithm is the preferred choice in most LHC jet analyses.

2.1.1 High-Level Variables

At the most fundamental level, a jet is a set of four-momentum vectors corresponding to the tracks and calorimeter towers selected by the jet clustering algorithm. Several approaches could be used to represent this information (many of which are discussed in the subsequent sections), but the simpler and most traditional one is to calculate jet observables, also referred to as *high-level* variables. The use of high-level variables has two main advantages. The first

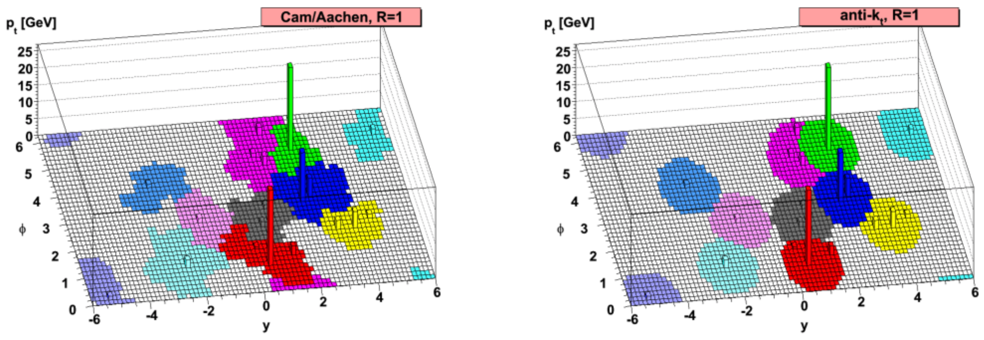


Figure 2.2: Sample jets obtained by clustering the same event with the Cambridge-Aachen (left) and anti-k_t (right) algorithms, with jet radius $R = 1$. Figures reproduced from [1].

is the dimensionality reduction of the datasets, distilling large sets of tracks and calorimeter towers into compact, 1D variables. The second is higher interpretability; since the analytical forms of the observables are known, physicists can use this information to gain insights into the substructure of the jets.

Examples of traditional high-level variables include the jet invariant mass (m_{jet}) and the multiplicity (M) of the jet constituents, which are given by

$$m_{\text{jet}} = \sqrt{\left(\sum_{i\in\text{jet}} E_i\right)^2 - \left(\sum_{i\in\text{jet}} \vec{p}_i\right)^2}, \tag{2.3}$$

$$M = \sum_{i\in\text{jet}} i, \tag{2.4}$$

where, with abuse of notation, i indices over all constituents in the jet. E_i and $\vec{p}_i$ are the energy and three-momentum of constituent i, respectively.

In the literature, physicists have engineered families of observables dedicated to characterizing jets according to specific properties. A well-known family is N-subjettiness (τ_N^β) [35], which measures the spread of the constituents along N candidate sub-jet axes. An observable in this family has the form

$$\tau_N^{\beta} = \sum_{i \in \text{jet}} p_{\text{T}i} \text{min}(\Delta R_{1i}, \Delta R_{2i}, \ldots, \Delta R_{Ni})^{\beta}, \tag{2.5}$$

where $\Delta R_{ni}^2 = (y_n - y_i)^2 + (\phi_n - \phi_i)^2$ is the distance between constituent i and candidate sub-jet n, $1 \leq n \leq N$. The angular weighting parameter β controls the relative importance of the spatial terms.

A more recent but commonly used family of observables is Energy-Flow Polynomials (EFPs) [4], which uses non-isomorphic, loopless multigraphs to build jet observables. In the EFP framework, a multigraph G with u vertices and $(k, l) \in G$ edges corresponds to an observable of the form

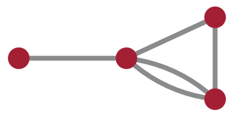
$$\text{EFP}_G = \sum_{i_1 \in \text{jet}} \cdots \sum_{i_u \in \text{jet}} p_{\text{T}i_1} \ldots p_{\text{T}i_u} \prod_{(k,l) \in G} \Delta R_{i_k i_l}. \tag{2.6}$$

Conceptually, an EFP summarizes the information of a jet by assigning each node on a graph a summation of all particles in a jet. The edges are assigned to pairwise distances between particles:

$$\text{For each node:} \qquad \bullet_j \Longleftrightarrow \sum_{i_j \in \text{jet}} p_{\text{T}i_j}, \tag{2.7}$$

$$\text{For each edge:} \qquad k \;\text{——}\; l \Longleftrightarrow \Delta R_{i_k i_l}. \tag{2.8}$$

As an example, the multi-graph below corresponds to an observable of the following form



$$\Longleftrightarrow \sum_{i_1 \in \text{jet}} \sum_{i_2 \in \text{jet}} \sum_{i_3 \in \text{jet}} \sum_{i_4 \in \text{jet}} p_{\text{T}i_1} p_{\text{T}i_2} p_{\text{T}i_3} p_{\text{T}i_4} \Delta R_{i_1 i_2} \Delta R_{i_2 i_3} \Delta R_{i_2 i_4}^2 \Delta R_{i_3 i_4}. \tag{2.9}$$

To avoid ambiguity when distinguishing between individual multigraphs, we accompany each EFP with a unique identifier (u, d, k), which specifies the number of nodes (u), edges (d), and index (k).

Lastly, we mention that Equation 2.6 is infra-red and collinear (IRC) safe–an important property discussed in Section 2.2–but EFPs can be generalized to IRC-unsafe forms by including energy (κ) and angular (β) weighting parameters:

$$\mathrm{EFP}_G^{\kappa,\beta} = \sum_{i_1 \in \mathrm{jet}} \cdots \sum_{i_n \in \mathrm{jet}} p_{\mathrm{T}i_1}^{\kappa} \cdots p_{\mathrm{T}i_n}^{\kappa} \prod_{(k,l) \in G} \Delta R_{i_k i_l} \beta. \tag{2.10}$$

2.1.2 Current High-Level Techniques: Dense Neural Networks (DNNs)

The current standard for jet classification using high-level variables is as input to dense neural networks (DNNs). Inspired by neurological networks (the brain), DNNs consist of a series of fully-connected layers, each of which is composed of nodes (neurons). An illustration of a typical DNN architecture is shown in Fig. 2.3. The input to the network comes all at once through the input layer and is processed by the input nodes, which are fully connected to the nodes in the intermediate layers, also known as the "hidden" layers. The information propagates through the network layer by layer until it reaches the nodes in the output layer. At each node, the corresponding inputs are weighted and summed together, then passed through an activation function. An illustration of these operations is shown in Fig. 2.4, where w corresponds to the weights multiplying the inputs, b an added quantity that is often referred to as the bias, and f the activation function. One of the most common activation functions used in the nodes in the hidden layers is the Rectified Linear Unit (ReLU) [36], which returns 0 if the value it receives is negative or the same value if it is positive, mimicking the threshold of excitation in a natural neuron. A common activation function used in the output nodes for binary classification tasks is the sigmoid function, which returns values between 0 and 1, and is interpreted as the probability assigned by the network of a sample belonging to each class.

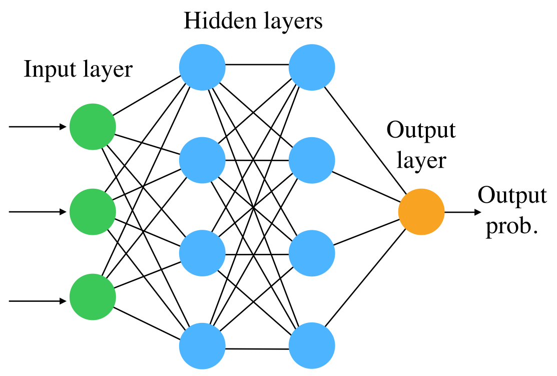


Figure 2.3: Sample architecture of a DNN.

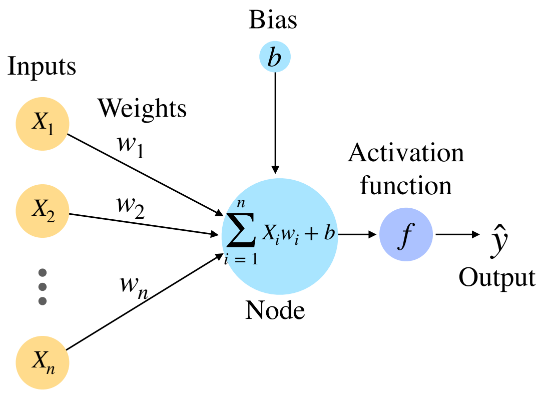


Figure 2.4: Diagram of the operations corresponding to a typical DNN node.

The initial magnitude of the weights w and biases b is often assigned randomly, resulting in a random initial prediction by the network. During learning, the training set is subdivided into batches, which are processed by the network, resulting in a predicted class probability $\hat{y} = \mathrm{p}(y)$. The quality of the predictions is evaluated by the loss function (L). A common loss function for classification is the categorical cross-entropy:

$$L = -\sum_{m=0}^{M-1} y_m \cdot \log(\hat{y}_m), \tag{2.11}$$

where M is the number of classes and $\hat{y}_m$ is the network's predicted probability of the samples belonging to class m. The expected class y_m is 1 when the samples belong to class m and 0 otherwise.

As an example, we consider the specific case of the binary cross-entropy loss (L_b):

$$L_\mathrm{b} = -\sum_{m=0,1} y_m \cdot \log(\hat{y}_m) = -y_0 \cdot \log(\hat{y}_0) - y_1 \cdot \log(\hat{y}_1). \tag{2.12}$$

In a binary setting, $\hat{y}_1 = 1 - \hat{y}_0$, and thus the loss can be simplified to

$$L_\mathrm{b} = -y \cdot \log(\hat{y}) - (1 - y) \cdot \log(1 - \hat{y}). \tag{2.13}$$

It is easy to see that $L_\mathrm{b} = -\log(\hat{y})$ when $y = 1$, and $L_\mathrm{b} = -\log(1 - \hat{y})$ when $y = 0$. Thus, accurate network predictions minimize the loss function.

After the loss is evaluated, a learning algorithm is called to update the weights and biases to values that are likely to minimize the loss. Typically, this is done by calling a stochastic gradient descent optimizer, which updates the weights along the direction of the negative gradient of the loss function. In practice, however, calculating the gradients is often computationally costly, particularly for deep multi-layer networks. This is resolved by using backpropagation, which computes the gradients of the final layer first and reuses this gradient in the calculation of the gradients in the prior layers. After this procedure is completed for all batches in the training set, a so-called epoch is completed, and the process is repeated until a maximum number of epochs is reached.

2.1.3 Low-Level Data

Recent advances in machine learning have made it feasible for networks to learn directly from the sets of particle tracks and calorimeter towers constituting jets, also known as low-level data. Appropriately, low-level data represents the jets at the lowest levels of abstraction and thus encodes all obtainable information.

Jet classification tasks have benefited immensely from using low-level data as input to deep neural networks. The main advantage has been an increase in performance when compared to networks operating on high-level variables [37–44], which makes low-level classifiers powerful probes of the information content of the jets.

On the other hand, a disadvantage of low-level classifiers is that they are often very expensive to train due to the high dimensionality of the data. In addition, training networks directly on the jet constituents leads to lower interpretability of the results since it is not possible to know which specific function of the input the networks are learning.

The following sections review some of the most common low-level neural networks for jet classification.

2.1.4 Convolutional Neural Networks (CNNs)

Inspired by computer vision research, Convolutional Neural Networks (CNNs) are one of the first deep neural networks trained on low-level detector data [37]. The main idea behind CNNs is to downsize an image into a form that is easier to process by extracting the critical features in the image structure. Particle physicists employ CNNs by treating the detector as a camera and a collision event as a snapshot. The event snapshots are mapped onto 2D images, reducing the jet classification task to one of image classification. Fig. 2.5 shows an example of a single jet calorimeter image where each pixel represents the energy deposited in a calorimeter cell. Similarly, Fig. 2.6 shows the average of 1,000 calorimeter images.

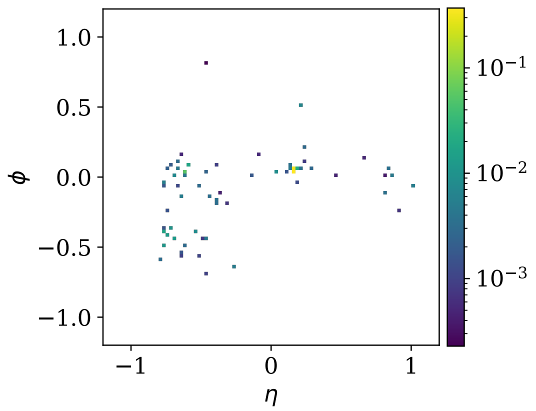


Figure 2.5: Sample jet image ($q\bar{q} \to q\bar{q}$).

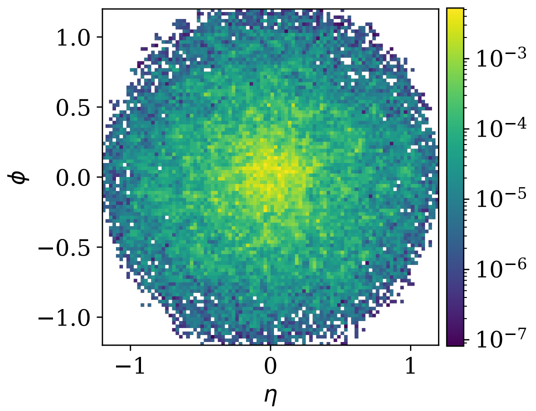


Figure 2.6: Average of 1,000 jet images ($q\bar{q} \to q\bar{q}$).

A typical CNN architecture for binary classification is shown in Fig. 2.7. The main building blocks of CNNs are convolutional layers. A convolution is a linear operation that involves the dot product between the input image and a 2D array of learnable weights called a filter. The filter is, by design, usually smaller than the input image. It slides across the height and width of the input, creating a two-dimensional output array called the feature map. The dimensions of the feature map are controlled by the stride and padding parameters. The stride controls the step size of the filter as it slides over the input, with larger strides corresponding to smaller output feature maps. Padding controls the number of zero-valued pixels added to the perimeter of the input. The feature map can be further reduced by applying a pooling layer. Two commonly used pooling layers are max-pooling and average-pooling, which create new feature maps containing the maximum/average pixel values of the original feature map. An example of a convolution layer is shown in Fig. 2.8, and of a max-pooling layer is shown in Fig. 2.9.

The convolution and pooling layers result in feature maps containing relevant features of the input images. These feature maps can be passed as inputs to additional convolutional and pooling layers or flattened and fed to a fully-connected layer. With each layer, the network increases in complexity, with earlier layers identifying simple features, such as colors and edges, and later layers focusing on the more complex elements of the image that are characteristic of each class.

2.1.5 Transformers

Inspired by natural language processing research, Transformer networks learn context by extracting relevant information in sequential data. First introduced in [45], Transformers currently provide state-of-the-art results in many machine learning tasks, including jet classification [46, 47]. However, a disadvantage of deep Transformer networks is that they are

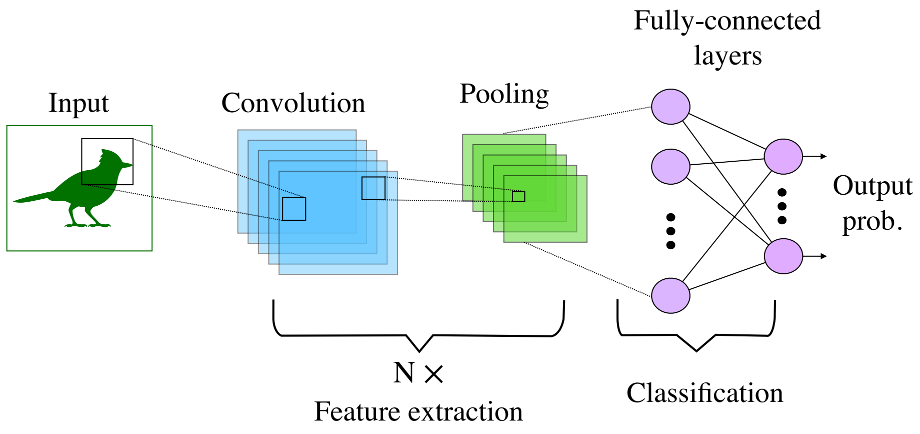


Figure 2.7: Sample architecture of a CNN.

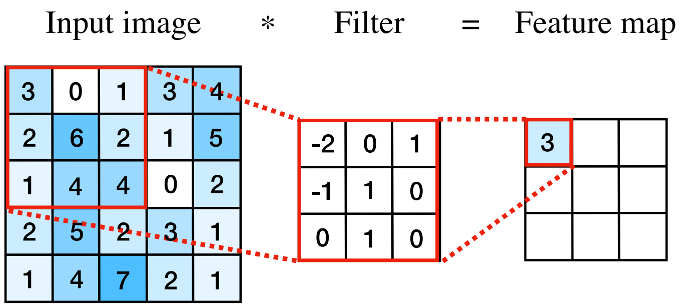


Figure 2.8: Illustration of a convolution operation.

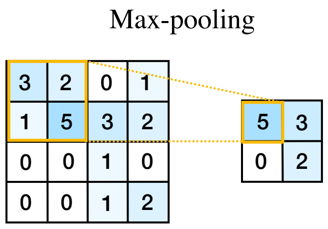


Figure 2.9: Illustration of a max-pooling operation.

often expensive to train, and like most low-level classifiers, their classification strategies are hard for physicists to interpret.

At the core of Transformer networks is the attention mechanism, which enables the model to learn long-range dependencies in the data, helping identify relevant features in the input sequence. The main components of the attention mechanism are the query (q), key (k), and value (v) vectors. The query vector encodes the information the model is inquiring about. The key vector is used as a reference, and it helps determine the similarity between elements in the query and elements in the input sequence. Lastly, the value vector encodes information about the input sequence. To compute the attention, the dot product between the query and key vectors is calculated. This dot product is scaled by the dimension of the vectors (d_k), and then passed through a softmax function which returns values between 0 and 1. The resulting values are used to weight the elements in the value vector, with weights larger in magnitude highlighting the elements that are deemed important by the network. These operations can be summarized as

$$\text{attention}(\mathrm{Q}, \mathrm{K}, \mathrm{V}) = \text{softmax}(\frac{\mathrm{QK}^{\mathrm{T}}}{\sqrt{d_k}})V, \tag{2.14}$$

where Q, K, and V are the matrices that aggregate all query, key, and value vectors, respectively. The scaled dot-product attention mechanism is illustrated in Fig. 2.10.

In practice, [45] finds that having multiple heads that compute the attention on different learned linear projections of the same query, key, and value vectors works better than a single attention computation. The results of each attention head are concatenated and fed to a linear network. This procedure is called the multi-head attention mechanism and is illustrated in Fig. 2.11.

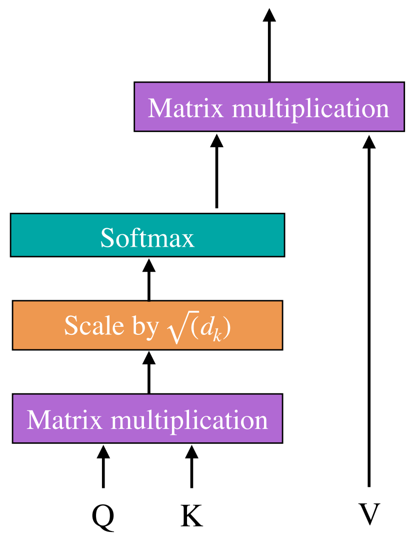


Figure 2.10: Diagram of the dot-attention mechanism.

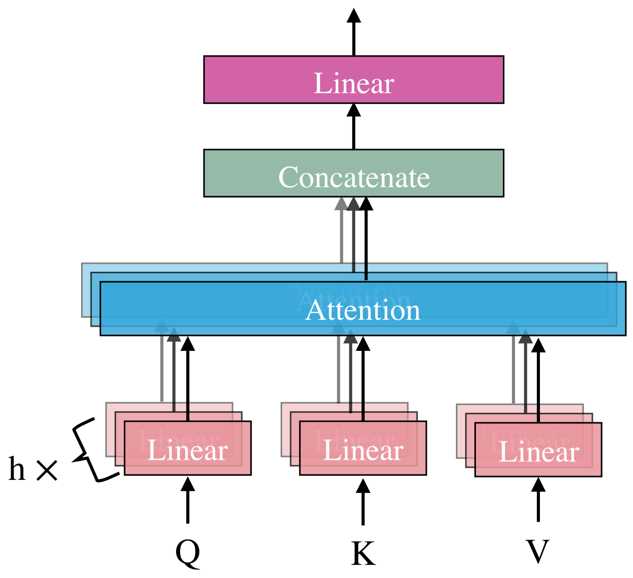


Figure 2.11: Diagram of the multi-head attention mechanism with h parallel attention heads.

A sample Transformer architecture is shown in Fig. 2.12. In brief, the Transformer follows an encoder-decoder structure. The role of the encoder, illustrated by the blue block, is to map an input sequence to a sequence of continuous representations. The encoder block consists of two sub-blocks: the first implements the multi-head attention mechanism and the second is a feed-forward network. Residual connections are employed around each sub-block, followed by a normalization layer. The role of the decoder, illustrated by the lilac block, is to receive the output of the encoder, together with the decoder output at a previous time step, and to generate an output sequence. The decoder block consists of three sub-blocks: the first receives the output of the previous decoder block in the stack, augments it with positional information, and implements a masked multi-head attention[1]. The second sub-block receives the queries from the first sub-block and the keys and values from the encoder output and implements multi-head attention. Lastly, the third sub-block implements a feed-forward network. Residual connections are employed around each sub-block, followed by a normalization layer. The positional encoding operations at the bottom of the encoder and decoder blocks are implemented to provide a sense of order to the input sequence, though particle physics applications of Transformer networks often omit positional encoding since particles in an event are permutation-invariant.

Typically, the architecture has multiple stacks of the encoder and decoder blocks, usually six stacks of each. The final output of the decoder stack is passed to a final sub-block specific to the learning task at hand. In classification tasks, this generally consists of a dense network followed by a softmax or sigmoid activation function, which outputs the probability of each class.

[1]The masking of the multi-head attention is to ensure that predictions depend only on known outputs.

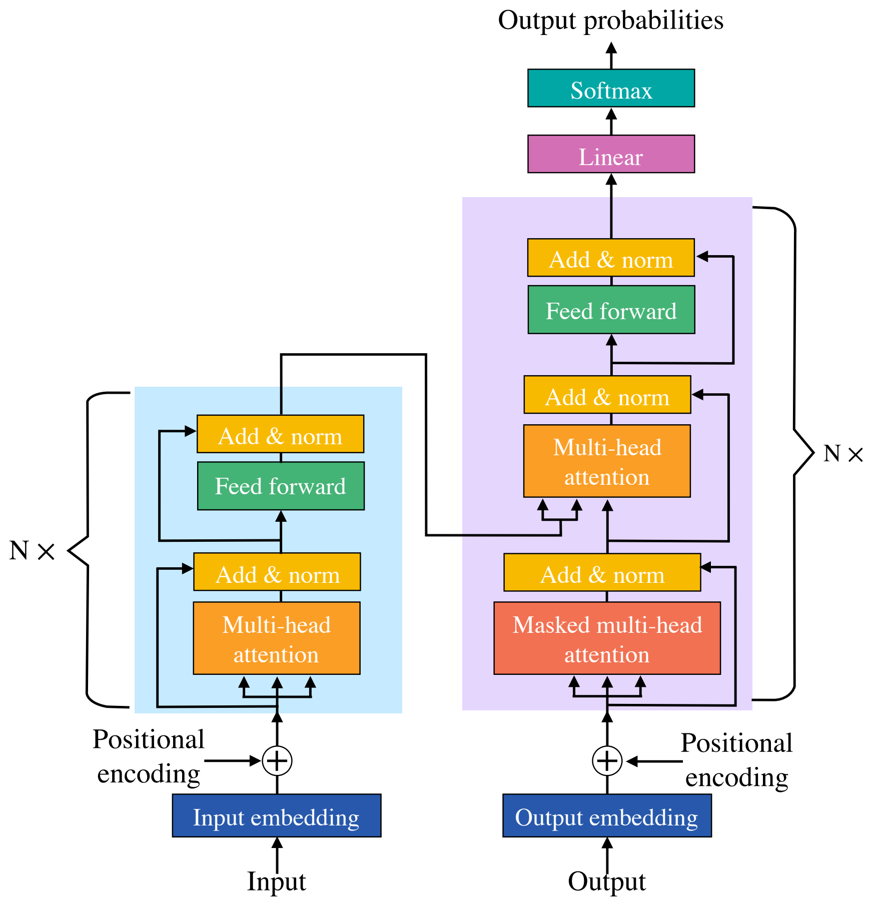


Figure 2.12: Sample Transformer architecture.

2.1.6 Physics-Inspired Networks

Many networks have been specifically designed for jet classification. As an example of such networks, we review the Particle-Flow (PFN) and Energy-Flow (EFN) networks [48]. Inspired by deep-set networks [49], which treat their inputs as permutation invariant point clouds, PFNs and EFNs provide a framework for learning permutation invariant representations of low-level particle data.

A sample architecture of a PFN is shown in Fig. 2.13. The network consists of the Φ-block and the F-block, represented by the blue and green blocks, respectively. The Φ-block takes as input the kinematic information of the constituents in a jet and processes them through a series of fully-connected layers, learning per-particle mappings (ϕ). The output of the last fully-connected layer is summed over all constituent mappings, generating event-level features that are permutation invariant ($\mathcal{O}$). These features are passed as the input to the F-block, which processes them through a series of fully-connected layers ending in an output node with softmax or sigmoid activation function that returns the probability of the jet belonging to each class.

The EFN is similar to the PFN but enforces IRC safety (discussed in the following section) in the per-particle mappings in the Φ block by learning only latent feature representations that are linear in energy. This is achieved by feeding only angular information of the constituents to the Φ block. The learned per-particle mappings are later weighted by the p_{T} of the constituents and summed over to generate the event-level features. A sample architecture of an EFN is shown in Fig. 2.14.

2.2 IRC-Safety

Infra-red and collinear (IRC) safety is a core guiding principle for constructing jet substructure observables that are calculable in perturbative quantum chromodynamics (pQCD) [50]. Compliance with IRC safety refers to whether the cross-section of an observable will be finite in any fixed order.

An observable O acting on M particles is IRC-safe if it is insensitive to soft emissions or collinear splittings [51]:

$$\text{Infrared safety:} \quad O(\{p_1^\mu \ldots, p_M^\mu\}) = \lim_{\epsilon\to 0} O(\{p_1^\mu \ldots, p_M^\mu, \epsilon p_{M+1}^\mu\}), \quad \forall p_{M+1}^\mu, \tag{2.15}$$

$$\text{Collinear safety:} \quad O(\{p_1^\mu \ldots, p_M^\mu\}) = O(\{p_1^\mu \ldots, (1-\lambda) p_M^\mu, \lambda p_M^\mu\}), \quad \forall \lambda \in [0,1], \tag{2.16}$$

where p_i^μ is the four-momentum vector of particle i.

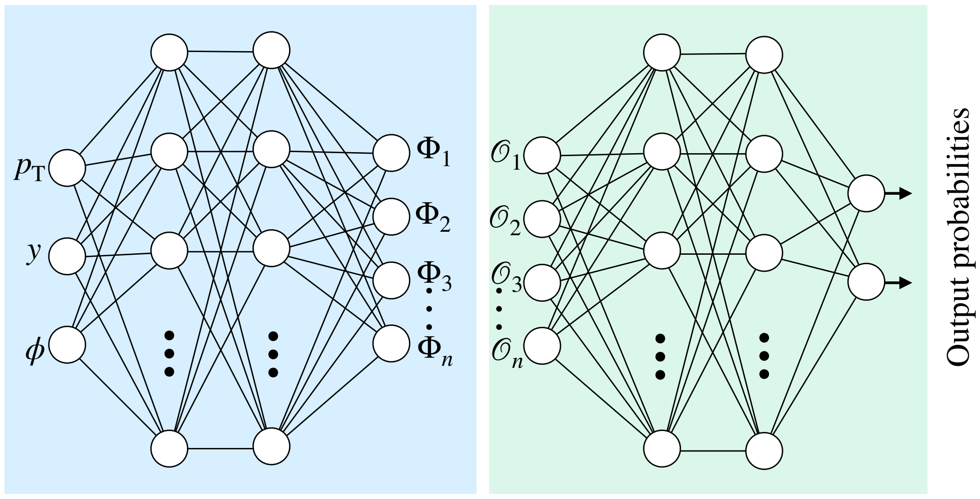


Figure 2.13: Sample PFN architecture.

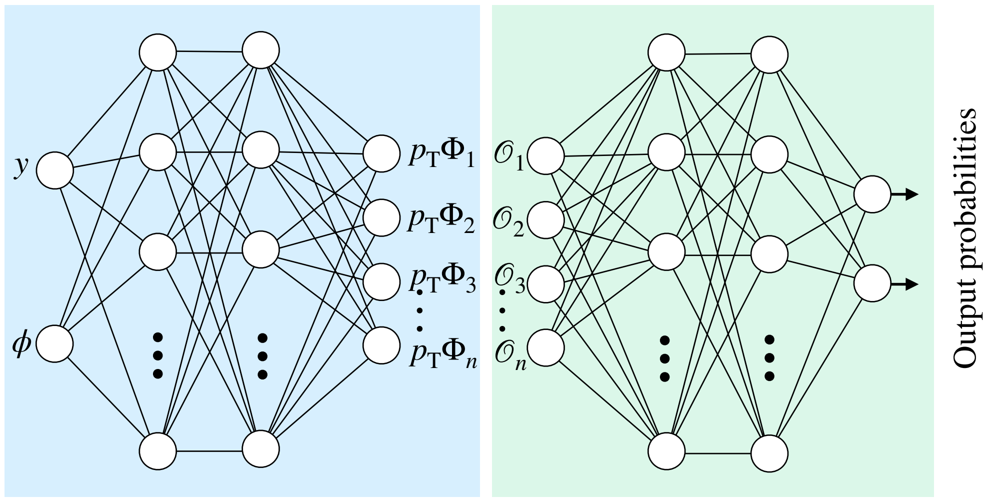


Figure 2.14: Sample EFN architecture.

By examining Equation 2.15, it is easy to see that observables must depend on positive powers energy to satisfy infrared safety. This way, the contributions of the particles whose energy tends to zero would also tend to zero. Similarly, by examining Equation 2.16, one can see that collinear safety has two requirements. The first is that observables must be linear in energy. The second is that observables must depend on positive powers of the angular terms. The latter is required to minimize the contributions of small-angle splittings.

Many attempts have been proposed to enforce IRC safety in jet observables [4, 35] or in network architectures [48, 52]. While the performance of the networks operating on IRC-safe information is formidable, studies have found a small but persistent gap in the performance between such networks and networks operating on the full event information [5, 48]. Explanations for the performance gap include:

- Networks operating on the full event information make use of soft and collinear radiation.
- The current IRC-safe observables do not capture *all* IRC-safe information that could be extracted from the constituents.

In the following chapters, we explore the nature of the information learned by deep networks to shed light on their learning strategies. With the classification of multi-prong jets as the case study, we compare the performance of various low- and high-level classifiers and study their potential for characterizing jet substructure.

Chapter 3

Multi-Prong Jet Classification

3.1 Introduction

We recall how jet classification is crucial to disentangling the various processes occurring at particle accelerators. Typical physics searches aim to sort through the overwhelming background of QCD jets to find jets of interest originating from processes beyond the standard model. Such jets can have complex energy patterns, including multiple, distinct sub-jets [53]. While jets with two or three sub-jets have been observed and extensively studied [54–60], jets with additional sub-jets are expected to become more important in future physics searches. As the LHC enters its high-luminosity era, physicists anticipate larger datasets in which high-p_{T} objects will appear in greater numbers [61, 62], leading to the creation of jets with multiple prongs.

Many theoretically motivated observables have been proposed and studied to identify jets with multiple prongs [4, 35, 63]. These observables have been used as inputs to neural networks, resulting in good classification performances. However, in practice, these networks are often outperformed by networks trained on low-level data [46, 64–67], suggesting that there is often additional information available in the jet constituents that is not captured by the observables.

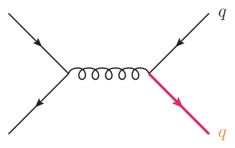
Specifically, studies with two hard sub-jets [41, 68] have found a small but statistically significant gap when comparing the performance of networks trained on low-level data vs. jet observables. This gap in performance has also been observed in several studies with three hard sub-jets in the context of top-quark tagging [69–72]. Studies with four hard sub-jets have utilized jet observables as input to classification taggers [73–75], but without a comparison to the performance of networks trained on low-level data.

In this chapter, we extend multi-prong jet classification studies to jets with up to eight hard sub-jets. The primary focus of this study is to answer the question of whether the low- vs. high-level performance gap found in jets with a low number of sub-jets is also found in jets with several sub-jets. To our knowledge, this is the first study of the performance gap for jets with more than three hard sub-jets. We employ calorimeter jets, but the strategy presented in this chapter could be applied to jets containing tracking information, which is left for future studies.

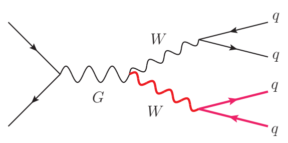
The rest of this chapter is organized as follows: Section 3.2 introduces the dataset. Section 3.3 specifies the jet observables that are used as input to dense networks. Similarly, Section 3.4 specifies the low-level networks. Section 3.5 shows the classification results of the networks. Section 3.6 presents a strategy for expanding the observable set with energy-flow polynomials (EFPs). Section 3.7 discusses the dependence of the classifiers on the topology of the jets. Section 3.8 presents the results of turning the multi-class classifier into a binary classifier. Concluding remarks and discussion are given in Section 3.9.

3.2 Dataset

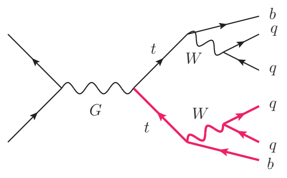
Simulated proton-proton collision events enriched in jets with many collimated quarks are generated using the processes shown in Figure 3.1. Most samples use the decay of a hypothetical heavy particle, such as a graviton (G), which subsequently decays via heavy Standard Model (SM) particles such as W bosons, Higgs bosons, or top quarks. Subsequent hadronic decays of these particles yield collimated pairs and triplets of quarks that contribute $N = 2$ hard sub-jets per boson or $N = 3$ hard sub-jets per top quark. For processes with $N \geq 4$ hard sub-jets, events are further required to contain high-energy photon radiation, which can, for example, boost a $G \to t\bar{t}$ decay into a single jet with $N = 6$ hard sub-jets.



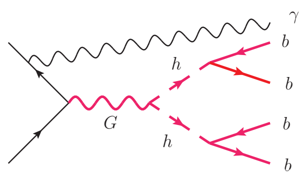
(a) $N = 1$



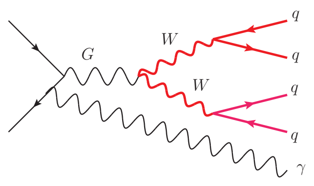
(b) $N = 2$



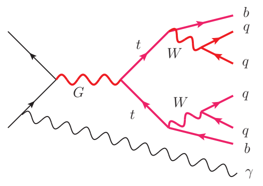
(c) $N = 3$



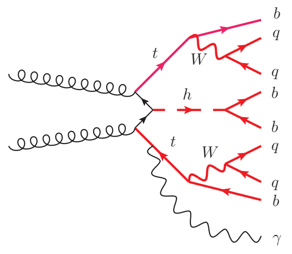
(d) $N = 4b$



(e) $N = 4q$



(f) $N = 6$



(g) $N = 8$

Figure 3.1: Feynman diagrams for processes that generate jets with $N = 1, 2, 3, 4b, 4q, 6, 8$ hard sub-jets. The lines in red indicate the components that are required to be truth-matched to the jet. See text and Table 3.1 for further generation details.

Samples are generated for $N = 1, 2, 3, 4, 6, 8$ hard sub-jet classes. To probe the dependence of the classification on the topology of the decay, the $N = 4$ class is subdivided into two: the $N = 4q$ class, in which two collimated W bosons produce a jet with four hard sub-jets from four light quarks, and the $N = 4b$ class, in which two collimated Higgs bosons result in a jet with four hard sub-jets from four b-quarks. A summary of the generated samples is given in Table 3.1.

Table 3.1: Processes used to generate jets with $N = 1, 2, 3, 4b, 4q, 6, 8$ hard sub-jets. Also shown are the particle masses and generator-level requirements used to more efficiently produce jets with $p_{\mathrm{T}} \in [1000, 1200]$ GeV and mass $\in [300, 700]$ GeV. All masses and momenta are in GeV. Selected W boson masses of (264.5,440.8,617.1) correspond to $M_Z = (300, 500, 700)$, respectively. See Figure 3.1 for the corresponding Feynman diagrams.

N hard sub-jets	Process	M_W	M_h	M_t	M_G	requirements
1	$q\bar{q} \to q\bar{q}$					$p_{\mathrm{T}}^{q} > 1000$
2	$q\bar{q} \to G \to W^+W^-$	80.4			2200	
		264.5			2200	
		440.8			2500	
		617.1			2800	
3	$q\bar{q} \to G \to t\bar{t}$			300	2200	
				500	2500	
				700	3000	
4b	$q\bar{q} \to \gamma G \to \gamma hh$				400	$p_{\mathrm{T}}^{\gamma} > 1000$
					600	$p_{\mathrm{T}}^{\gamma} > 1000$
					800	$p_{\mathrm{T}}^{\gamma} > 1000$
4q	$q\bar{q} \to \gamma G \to \gamma W^+W^-$				400	$p_{\mathrm{T}}^{\gamma} > 1000$
					600	$p_{\mathrm{T}}^{\gamma} > 1000$
					800	$p_{\mathrm{T}}^{\gamma} > 1000$
6	$q\bar{q} \to \gamma G \to \gamma t\bar{t}$				400	$p_{\mathrm{T}}^{\gamma} > 1000$
					600	$p_{\mathrm{T}}^{\gamma} > 1000$
					800	$p_{\mathrm{T}}^{\gamma} > 1000$
8	$q\bar{q} \to \gamma t\bar{t}h$		100	125		$p_{\mathrm{T}}^{\gamma} > 1000$
			125	175		$p_{\mathrm{T}}^{\gamma} > 1000$

Proton-proton collisions at a center-of-mass energy $\sqrt{s} = 13$ TeV are simulated with MADGRAPH5 v2.8.1 [21], showered and hadronized with PYTHIA 8.244 [20], and the detector response is simulated with DELPHES 3.4.2 [22] using an ATLAS-like card with calorimeter grids of uniform width 0.0125 in both η and ϕ. This fine granularity ensures that our studies include some of the effects of detector response while probing the limits of the algorithms rather than the resolution of the detector. Jets are clustered using the anti-k_{T} algorithm [1] with radius parameter $R = 1.2$ using FastJet 3.1.2 [76]. Only jets whose p_{T} lies in the range $[1000, 1200]$ GeV and jet mass in the range $[300, 700]$ GeV are kept. The quarks produced from each process are truth-matched to the large-radius jet by requiring $\Delta R = \sqrt{\Delta\phi^2 + \Delta\eta^2} < 1.2$; only jets with the full set of quarks passing this requirement are kept.

To generate jets with a variety of masses, several choices are made for the intermediate particle masses. The resulting spectrum of jet masses features clear artifacts due to these choices. Similarly, the distribution of jet p_{T} shows some dependence on the process used to generate the set of collimated quarks. To avoid learning artifacts in jet mass and p_{T}, we selectively reject events until we achieve uniform distributions of jet mass and p_{T} for all classes. This process yields a balanced sample of unweighted events at the expense of reduced generation efficiency. The balanced histograms are shown in Figure 3.2, demonstrating an approximately uniform distribution in jet mass and p_{T}.

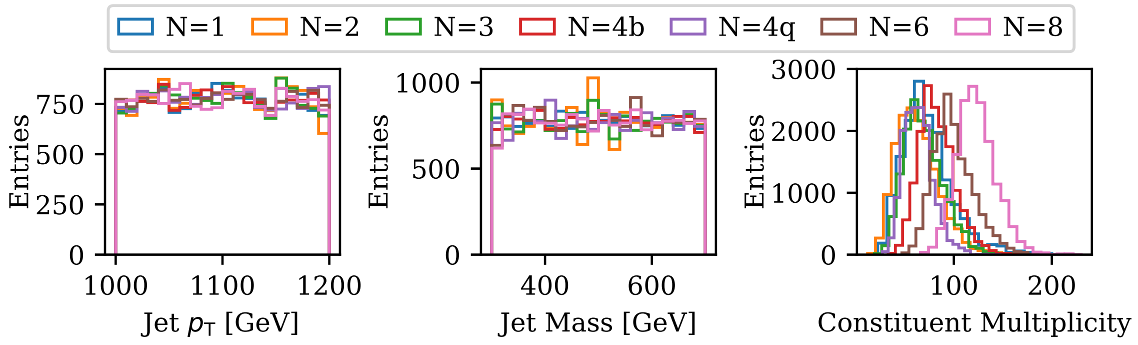


Figure 3.2: Distributions of jet p_{T}, mass, and constituent multiplicity for the simulated jets with $N = 1, 2, 3, 4b, 4q, 6, 8$ hard sub-jets.

The jets are preprocessed by normalizing the p_{T} of their constituents to sum to unity and centered based on the p_{T}-weighted arithmetic mean in η and circular mean in ϕ. In addition, we add zero-padding to the events to ensure all jets have a fixed-sized length of 230. We find that padding the events to this length is more than enough to capture all hard constituents in the events, as they have a mean constituent multiplicity of 82; see Figure 3.2.

In total, we have 108,359 simulated jets (around 15,480 of each class), which we divide into a training set, a validation set, and a test set with proportions 80 : 10 : 10, respectively. We use 10-fold cross-validation to ensure statistical robustness for all results.

3.3 High-Level Models

Identification of jets with N hard sub-jets is a well-explored topic experimentally for $N \leq 3$, for which many theoretically motivated observables have been constructed to summarize the information contained in the jet energy patterns [4, 35, 37, 63]. These observables have the advantage that they are compact and physically interpretable and can, in principle, be applied to jets with many more hard sub-jets. Here, we use N-subjettiness [35], together with the jet mass, as a well-known benchmark. Following Ref. [77], we calculate a total of 135 N-subjettiness observables ($\tau_{\mathrm{N}}^{\beta}$) along the k_{T} axis, with the sub-jet axis parameter $\mathrm{N} = 1, \ldots, 45$, and angular weighting exponent $\beta = \frac{1}{2}, 1, 2$, which together with the jet mass account for 136 jet observables. Distributions of some of the N-subjettiness observables are shown in Figure 3.3.

We train a DNN on the 136 high-level variables, which we refer to as DNN_{136}. A grid search of the hyperparameters of the dense network indicates that the best structure for the network has six hidden layers of size (800-800-800-800-800-64) and ReLu [36] activation function. To prevent overfitting and to facilitate training stability, dropout with rate 0.3 and batch normalization are applied respectively after every hidden layer. The output layer is a

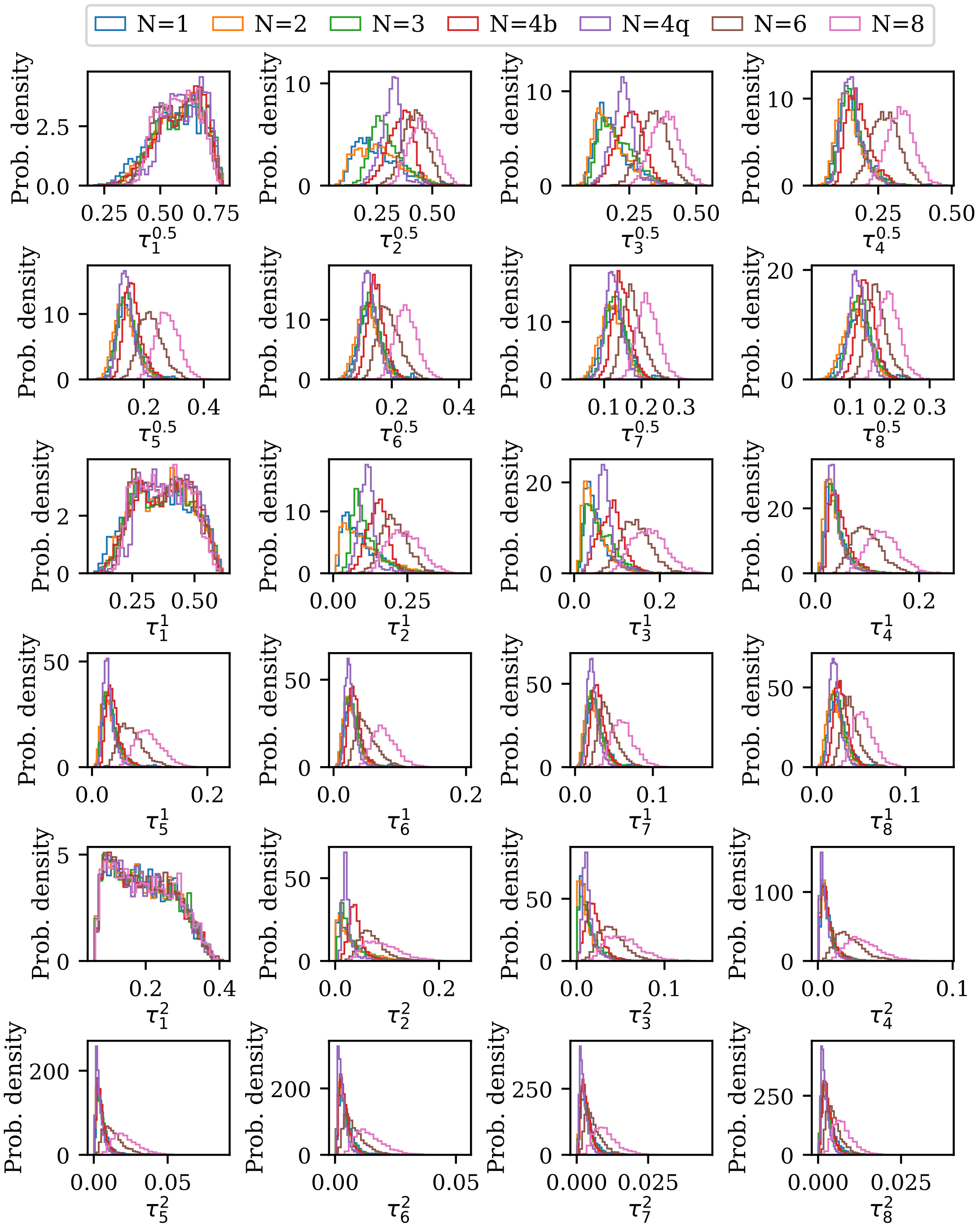


Figure 3.3: Distributions of N-subjettiness variables ($\tau_{\mathrm{N}}^{\beta}$) with $\mathrm{N} = 1, \ldots, 8$, and $\beta \in \{\frac{1}{2}, 1, 2\}$, in samples of simulated jets with $N = 1, 2, 3, 4b, 4q, 6, 8$ hard sub-jets.

7-dimensional softmax function, with one dimension for each category of N hard sub-jets. Refer to Table A.1 for the model summary of DNN_{136}.

3.4 Low-Level Models

We train low-level networks directly on the jet constituents, focusing on two such network architectures: Particle-Flow Networks (PFN) [48] and Transformers [45]. Both networks have matched or outperformed other low-level network architectures, such as convolutional networks, in a variety of classification tasks [48, 64, 68], but the choice of these particular networks lies in their specific learning strategies. PFNs learn an event-level latent representation of the jets; see Section 2.1.6. On the other hand, Transformers learn a contextualized embedding of the constituents and use attention mechanisms to determine which parts of the embedded input sequence to focus on when making predictions, see Section 2.1.5. Both networks are invariant to the input order, which makes them well-suited for jet classification tasks.

3.4.1 Particle-Flow Network

A grid search of the hyperparameters of the PFN finds that the best structure for the network has two layers in the Φ module of size $(128, 128)$ and two layers in the F module of size $(1024, 1024)$, with a dropout rate of 0.2. All hidden layers have ReLu [36] activation functions. The output layer is the same as for DNN_{136}, a 7-dimensional softmax with one element for each value of N hard sub-jets. Refer to Table A.1 for a summary of the PFN model.

3.4.2 Transformer

We employ the Encoder model from the Transformer in [78], which has a stack of multiple attention vectors computed in parallel to increase the expressiveness of the network. A grid search of the hyperparameters of the Transformer finds that the best structure for the network has four transformer layers, with hidden size of 256 and intermediate dimension of 128. For computational efficiency and to compare networks of similar complexity (see Table A.1), we focus our search on smaller architectures compared to the ones used in the original paper [45, 78]. Refer to Table A.1 for a summary of the Transformer model.

3.5 Performance

The predictive accuracy of a network is measured as the fraction of correctly identified samples. In this case, it refers to the fraction of jets whose predicted class matches the true class. The 10-fold accuracy of the networks is shown in Figure 3.4. The Transformer network achieves the highest overall classification performance with an accuracy of $91.27 \pm 0.31\%$, followed by the PFN with an accuracy of $89.19 \pm 0.23\%$.[1] The DNN_{136} is the least performant model, with an accuracy of $86.90 \pm 0.20\%$.

The accuracy of the networks is not uniform across all classes, with some classes having better classification performances than others. Despite this, the relative ranking of the three networks is mostly the same for each class, with the exception of N=4b, where the DNN_{136} outperforms the PFN. The classes with the lowest accuracy scores are $N = 2$, $N = 3$, and $N = 4b$. The confusion matrices showing the mean 10-fold classification predictions are shown in Figure 3.5. These matrices show that the networks often misclassify the $N =$

[1]We compared the performance of the PFN to an EFN [48], which has a strictly linear dependence on p_T, enforcing IRC safety. The performances of the networks were equivalent, indicating that the PFN was not learning IRC unsafe information.

$2, 3, 4b$ classes among each other (and with $N = 1$ to a lesser extent), which explains the lower performance. The confusion matrices also show that for a given N class, the largest classification mistype is often with the $N \pm 1$ classes. This suggests that the networks have learned to identify the number of hard sub-jets. Lastly, we note that the class with the highest accuracy score is the $N = 4q$ class and how infrequently it is misclassified with the $N = 4b$ class, suggesting that the networks are learning more information than simply the number of hard sub-jets.

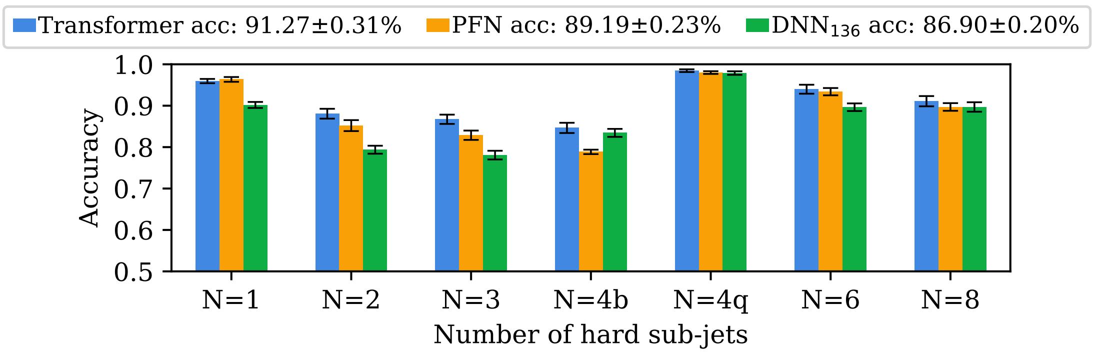


Figure 3.4: Mean 10-fold accuracy and statistical uncertainty of the network predictions for jets in each class.

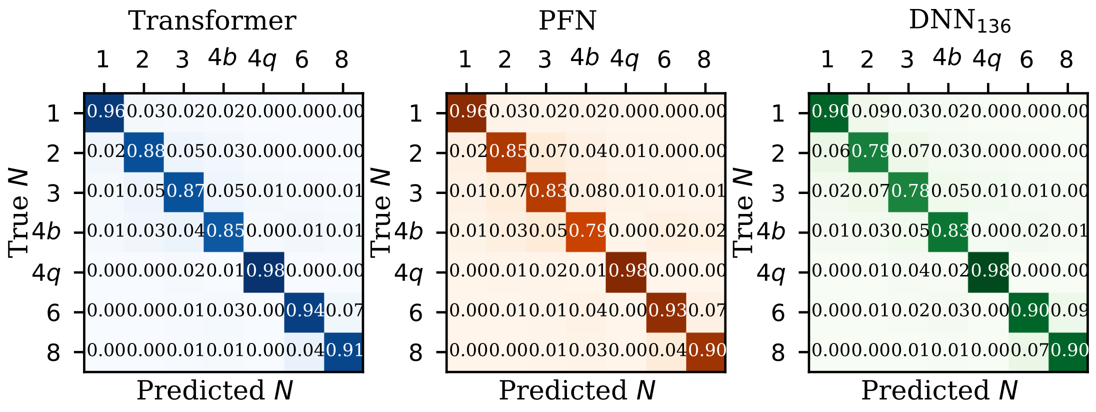


Figure 3.5: Confusion matrices of the networks. The entries show the frequency at which the networks predict a jet class for a given class.

To study whether the networks have a learned dependence on the p_{T} of the jets, we show the accuracy of their predictions as a function of jet p_{T} in Figure 3.6. The accuracy of the networks remains relatively constant across the spectrum, indicating that the networks do not have a strong p_{T} dependence. In all ranges, the Transformer slightly outperforms the PFN, both followed by the DNN_{136}.

We also study whether the networks have a learned dependence on the jet mass by assessing the accuracy of their predictions as a function of jet mass,[2] shown in Figure 3.7. The Transformer again outperforms the PFN and DNN_{136} in all ranges. However, unlike the p_{T} spectrum, the prediction accuracy of the networks increases with jet mass. This dependence is further studied in Section 3.6.2 by inspecting the correlations between the most important high-level variables and the jet mass.

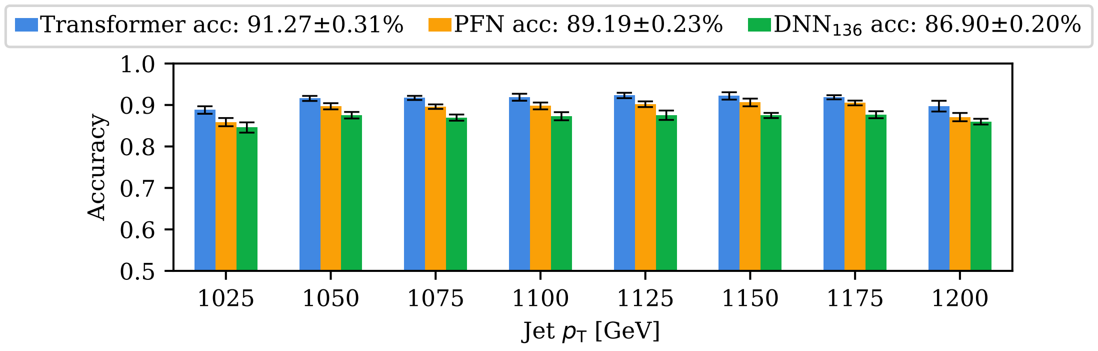


Figure 3.6: Mean 10-fold accuracy and statistical uncertainty of the network predictions for jets within the specified ranges of jet p_{T}. The jets are binned according to their p_{T} in intervals of 25 GeV. The x-axis labels correspond to the upper bound of the intervals.

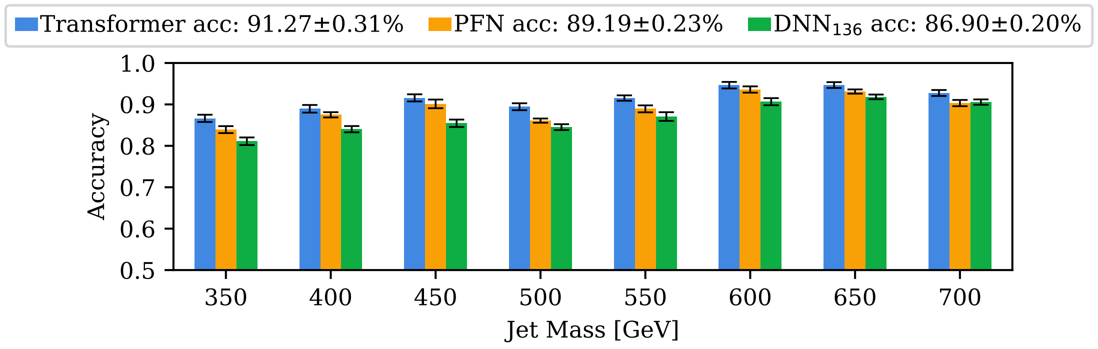


Figure 3.7: Mean 10-fold accuracy and statistical uncertainty of the network predictions for jets within the specified ranges of jet mass. The jets are binned according to their mass in intervals of 50 GeV. The x-axis labels correspond to the upper bound of the intervals.

[2] As a crosscheck, we tested the mass dependence of the networks across smaller mass ranges and found that their performance agrees with the results shown in Figure 3.7.

The results indicate that the observables are powerful discriminants for multi-prong jets but that they are often outperformed by networks trained directly on the jet constituents. This suggests that low-level networks may be identifying additional information not captured by the observables.

3.6 Closing the Performance Gap

In the following sections, we attempt to bridge the gap between high- and low-level information by expanding the observable set. A strategy for identifying jet observables that are able to bridge the performance gap is described in Ref. [68]. This strategy consists of searching among a pool of observables called energy flow polynomials (EFPs) [4] to identify those that yield similar classification decisions as networks trained on low-level detector data. This strategy, however, applies only to binary decision functions [64, 65]. We leave a generalization of that method to multi-class networks for future work and instead apply a simpler but commonly used technique for variable selection. First, we expand the observable set by including a large number of EFPs. Then, we systematically reduce the number of observables to find the minimum set that best approximates the performance of the low-level networks.

The reason for choosing EFPs as complimentary observables is that they form a basis of IRC-safe information [4]. In principle, an infinite number of EFPs should capture most, if not all, of IRC-safe information in the jet constituents.

3.6.1 Adding Energy-Flow Polynomials

For our studies, we select all connected (prime) EFP graphs with five or fewer edges and weighting factors $\kappa = 1$ and $\beta = \frac{1}{2}, 1, 2$, for a total of 162 observables, which are denoted by $\text{EFP}^{\beta}(u, d, k)$ in what follows; see Equation 2.10. We also include the constituent multiplicity, which is described by the one-node EFP with $\kappa = 0$, and has been shown to be a useful observable for jet discrimination [79]. Combined with the N-subjettiness variables and the jet mass, the new augmented dataset has 299 observables. We train a dense network on the 299 observables, labeled DNN_{299}. For consistency, we utilize the same hyperparameters as in DNN_{136}. The resulting overall accuracy is $89.23 \pm 0.26\%$, and the accuracy per class is shown in Figure 3.8. With the augmented set, we are able to match the overall performance of the PFN and to close the performance gap with the Transformer to within 2%.

The accuracies of DNN_{299} across the jet p_{T} and mass spectra are shown in Figure 3.9 and Figure 3.10, respectively. In most ranges, DNN_{299} matches or closely approximates the performance of the PFN. A small but persistent gap remains between the DNN_{299} and Transformer models, indicating that the Transformer model is still utilizing useful information not available in the augmented observable set. Additional EFP observables may be able to further narrow the gap. Such EFPs might have energy and angular measures not considered in this paper or a larger number of edges (degree), which allow for more complex polynomial forms. The costly computation of these variables makes this infeasible at the present time and so is left for future work.

3.6.2 Feature Selection

The next step in the selection process is to identify a subset of observables that can approximate the performance of the full set, which would facilitate the feature analysis and provide insights into the classifiers' strategies. We employ a simple feature selection method based on L_1 [80] regularization, also known as LASSO regularization. This type of regularization is commonly used in linear regression for feature selection. The core idea behind it is to penalize the loss function by adding a term proportional to the absolute values of the weights of the linear terms, which encourages irrelevant weights to be zeroed out. Here, we penalize

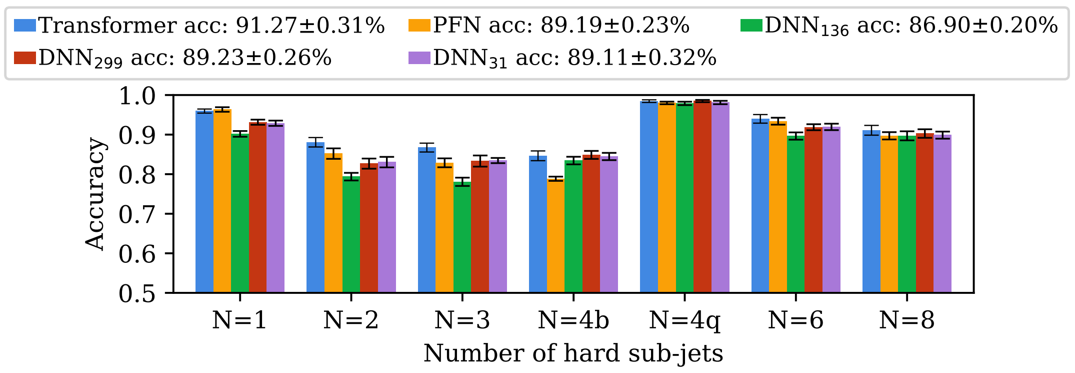


Figure 3.8: Mean 10-fold accuracy and statistical uncertainty of the network predictions jets in each class. The results for all networks in the study are shown.

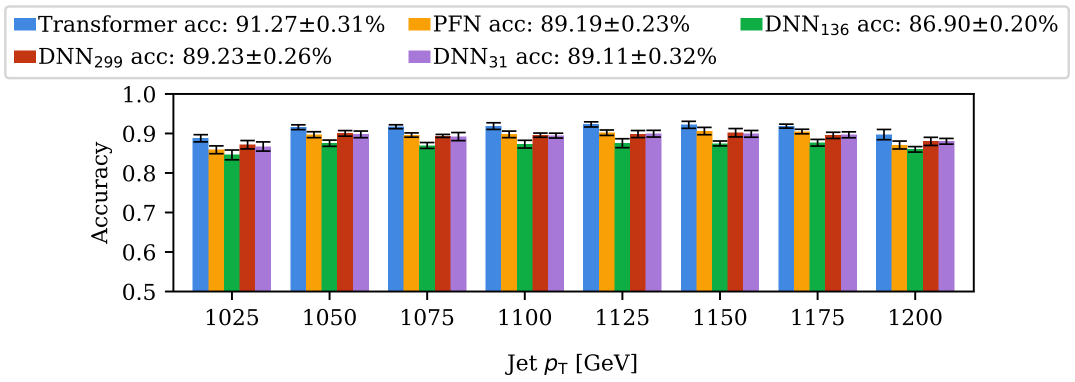


Figure 3.9: Mean 10-fold accuracy and statistical uncertainty of the network predictions for jets within the specified ranges of jet p_{T}. The jets are binned according to their p_{T} in intervals of 25 GeV. The x-axis labels correspond to the upper bound of the intervals. The results for all networks in the study are shown.

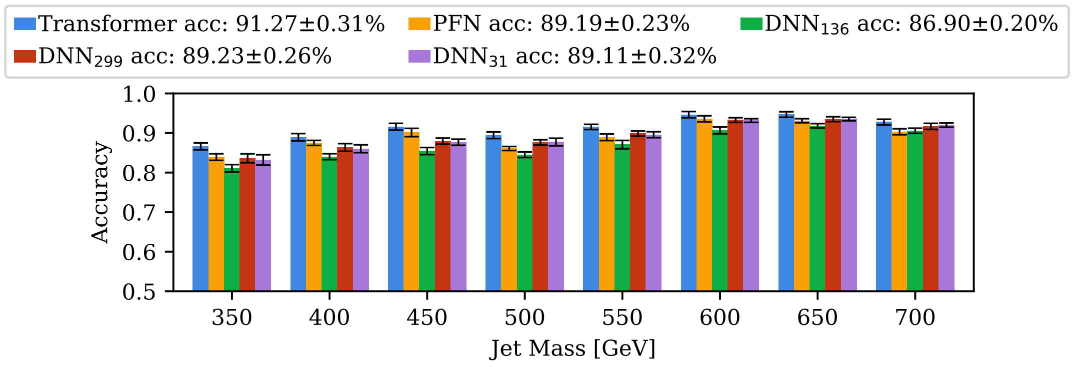


Figure 3.10: Mean 10-fold accuracy and statistical uncertainty of the network predictions for jets within the specified ranges of jet mass. The jets are binned according to their mass in intervals of 50 GeV. The x-axis labels correspond to the upper bound of the intervals. The results for all networks in the study are shown.

the loss function of the dense network by adding a term proportional to the weights of the input features. We recall that the input layer has dimension 299, and it is fully-connected to the first hidden layer of dimension 800, which results in a weight matrix of size 299×800. Thus, feature i has weights $\boldsymbol{w}_i = [w_1, \ldots, w_{800}]^T$ where $\boldsymbol{w}_i$ is a column vector of length 800. We implement a learnable gate parameter g_i that multiplies the weight vector of feature i; $\boldsymbol{w}_i \rightarrow g_i \boldsymbol{w}_i$. The LASSO regularization is applied to the gate parameters such that if g_i is zeroed out, so is the entire weight vector of feature i, and we can confidently exclude the ith observable during training. The loss function can then be written as

$$L = -\log f(y, \hat{y}) + \lambda \cdot \sum_{i=1}^{299} |g_i|, \tag{3.1}$$

where the first term, $-\log f(y, \hat{y})$, is the negative log-likelihood of the predicted label $\hat{y}$ and true label y, and the second term is the LASSO regularization term on the gate parameter g with regularization strength parameter λ.

We note that although the LASSO-inspired strategy can help us identify useful features, it does not guarantee that the selected features are the ideal ones that capture all relevant information for the classification task. Finding such features would require a more in-depth analysis of the features and their correlations, which we reserve for future studies.

A grid search of the regularization strengths ranging from 1 to 10 indicates that the best regularization strength parameter value for the classification task is $\lambda = 5$. To further limit the minimum observable set, only observables with a gate parameter of $|g_i| > 0.01$ are kept. With these settings, the selection strategy results in 31 selected observables whose distributions are shown in Figures 3.11 and 3.12.

We train a dense network operating on the 31 selected observables, labeled DNN_{31}. A hyperparameter search indicates that the best structure for the network has six hidden layers of size (800-800-800-800-800-32) and ReLu [36] activation function. Dropout with

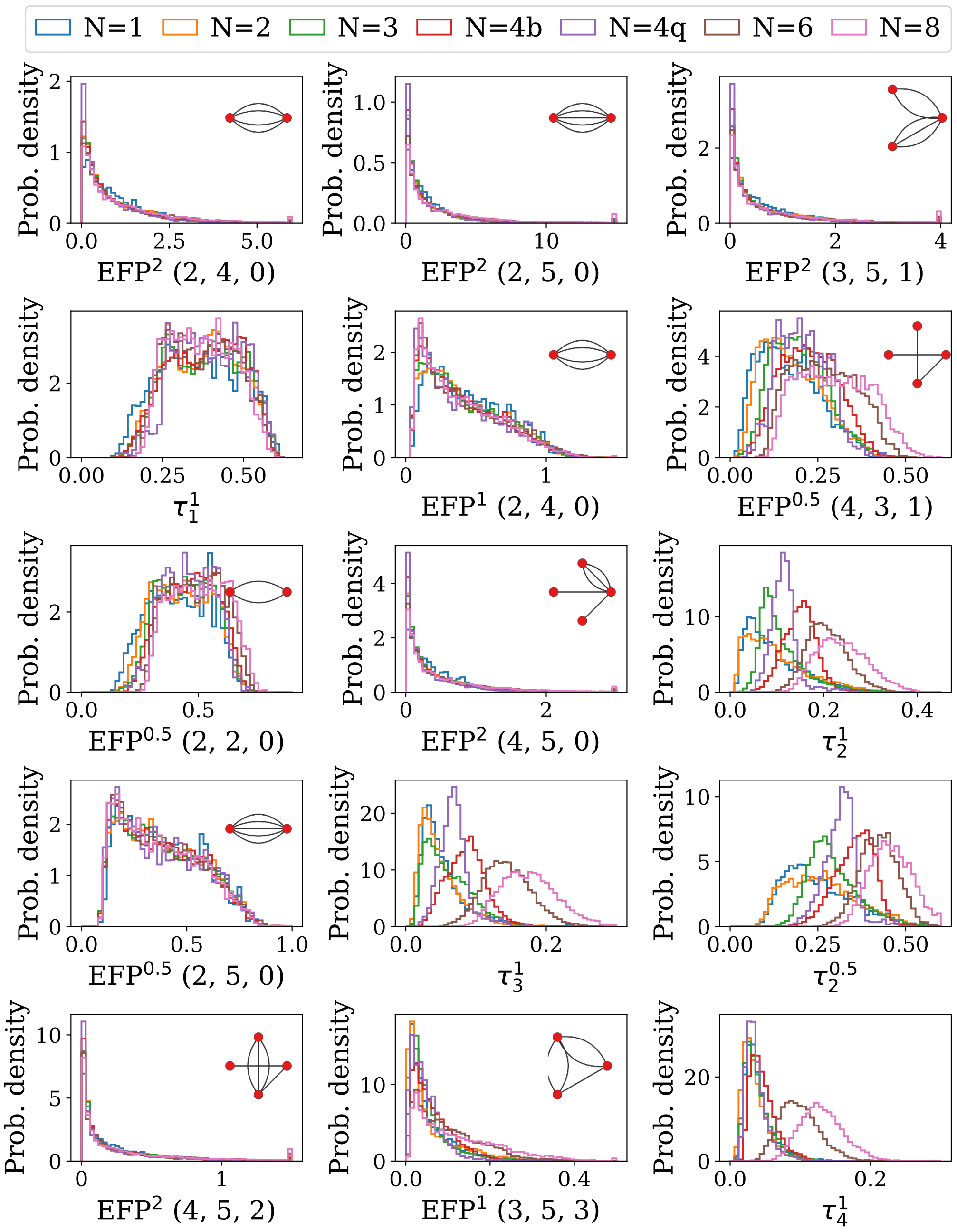


Figure 3.11: Distributions of the first half of the selected observables. The plots are ranked left-to-right top-to-bottom in order of importance according to the method described in Section 3.6.2. For each EFP observable, the corresponding graph is also displayed for visualization purposes. The jet mass, which ranks between τ_3^1 and $\tau_2^{0.5}$ as the 12th most important variable, is omitted for brevity as this distribution is approximately flat by design and is already shown in Figure 3.2. The distributions of the rest of the selected observables are shown in Figure 3.12.

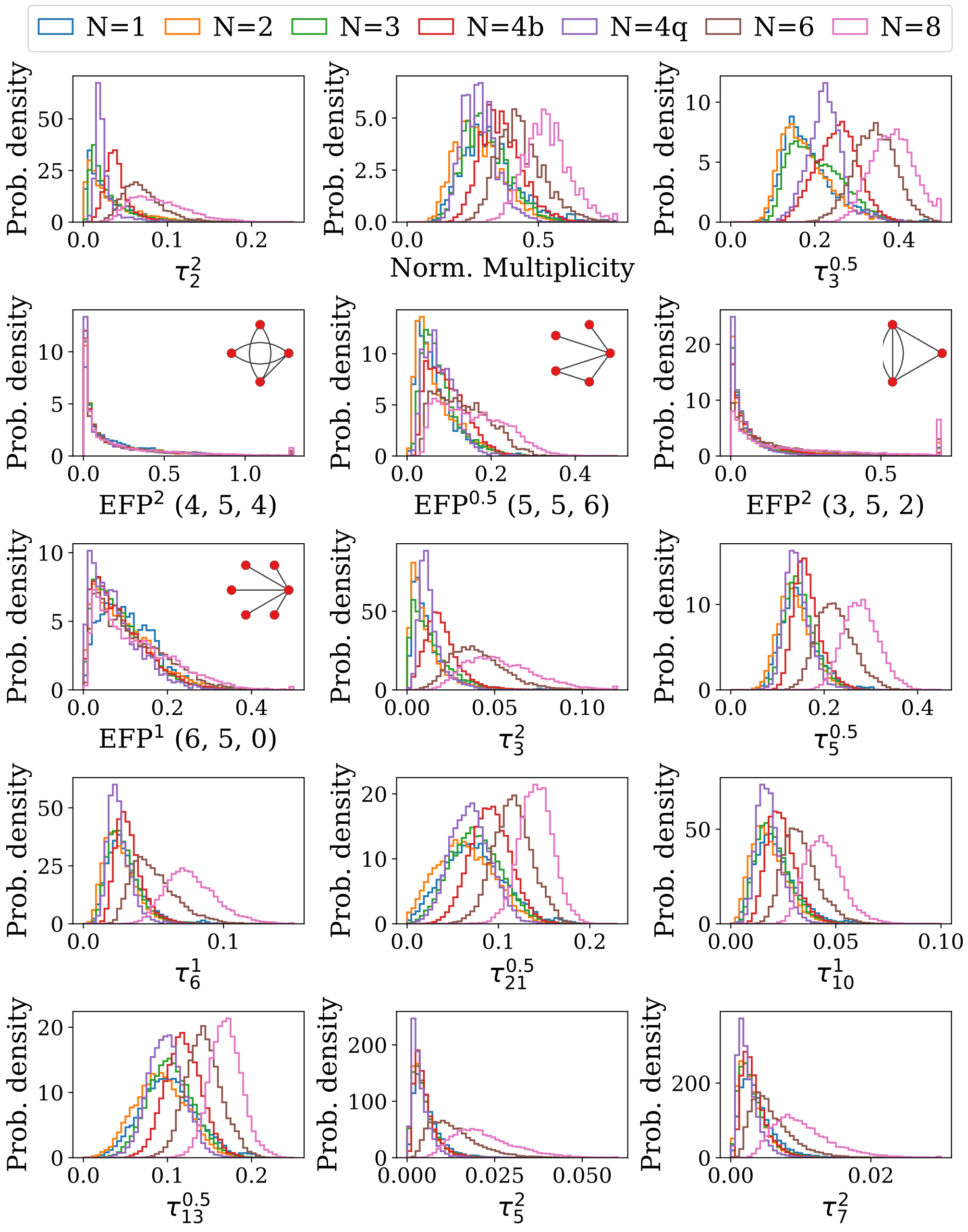


Figure 3.12: Distributions of the second half of the selected observables. The plots are ranked left-to-right top-to-bottom in order of importance according to the method described in Section 3.6.2. For each EFP observable, the corresponding graph is also displayed for visualization purposes. The distributions of the rest of the selected observables are shown in Figure 3.11.

rate 0.3 and batch normalization are applied respectively after every hidden layer. Refer to Table A.1 for the model summary of DNN_{31}.

The DNN_{31} network achieves an overall accuracy of $89.11 \pm 0.32\%$, and the accuracy per class is shown in Figure 3.8. We note that for all classes, the DNN_{31} matches or closely approximates the performance of the DNN_{299}, indicating that the LASSO selection strategy has successfully identified observables that capture most of the information in the full observable set.[3]

Figure 3.9 and Figure 3.10 show the accuracies across the jet p_{T} and mass spectra, respectively. Both figures show similar trends for DNN_{31} as for the other networks: uniform predictions across the jet p_{T} spectrum and generally rising accuracy with jet mass.

After narrowing down the set of observables to the 31 selected observables, we attempt to interpret the strategies of the low-level networks in terms of these observables by ranking them in order of importance.

We measure the reliance of the DNN_{31} model on each of the observables by randomly shuffling them during test time. In other words, the k-th observable ($k \in \{1, \ldots, 31\}$) in the test set is shuffled by randomly replacing it with a corresponding observable value from the training set on an event-by-event basis. This maintains the marginal distribution for the k-th observable but destroys any correlations with other observables. The performance of the network is then evaluated on the set of events that includes the shuffled observable. Assuming that the observables are weakly correlated, this method allows us to determine the importance of the k-th observable by measuring the drop in the performance of the network relative to the unshuffled set[4]. By this method, an observable is considered important if shuffling its values

[3]Although the selected observables do a good job at matching the performance of the full 299 observable set, the overlapping nature of the EFP observables makes it likely that this is not a unique solution and that different settings in the LASSO selection process may yield a somewhat different subset of observables.

[4]A full, proper ranking of the observables accounting for correlations would be too computationally costly. Instead, we rank their contributions as *individual* discriminators.

decreases the accuracy of the model. Likewise, an observable is considered unimportant if shuffling its values leaves the accuracy unchanged since it suggests that the model does not heavily rely on that singular observable when making predictions.

Table 3.2 shows the ranking of the 31 selected observables. These observables mainly consist of EFP variables with four or fewer nodes and N-subjettiness variables with N $<$ 8. The latter is not surprising given to the pronginess of the jets.

Rank	Observable	Accuracy drop
1	EFP^2 (2, 4, 0)	$50.60 \pm 1.95\%$
2	EFP^2 (2, 5, 0)	$44.81 \pm 2.28\%$
3	EFP^2 (3, 5, 1)	$41.68 \pm 2.35\%$
4	τ_1^1	$41.49 \pm 1.06\%$
5	EFP^1 (2, 4, 0)	$38.81 \pm 1.41\%$
6	$\text{EFP}^{0.5}$ (4, 3, 1)	$37.99 \pm 1.02\%$
7	$\text{EFP}^{0.5}$ (2, 2, 0)	$37.26 \pm 1.38\%$
8	EFP^2 (4, 5, 0)	$35.37 \pm 0.70\%$
9	τ_2^1	$34.97 \pm 0.63\%$
10	$\text{EFP}^{0.5}$ (2, 5, 0)	$33.66 \pm 2.02\%$
11	τ_3^1	$30.26 \pm 1.08\%$
12	Norm. Jet Mass	$29.44 \pm 0.92\%$
13	$\tau_2^{0.5}$	$29.38 \pm 0.90\%$
14	EFP^2 (4, 5, 2)	$27.66 \pm 1.66\%$
15	EFP^1 (3, 5, 3)	$27.58 \pm 0.88\%$
16	τ_4^1	$26.74 \pm 0.96\%$
17	τ_2^2	$26.17 \pm 0.92\%$
18	Norm. Multiplicity	$24.87 \pm 0.77\%$
19	$\tau_3^{0.5}$	$24.53 \pm 0.68\%$
20	EFP^2 (4, 5, 4)	$23.00 \pm 1.10\%$
21	$\text{EFP}^{0.5}$ (5, 5, 6)	$22.59 \pm 1.07\%$
22	EFP^2 (3, 5, 2)	$21.39 \pm 0.80\%$
23	EFP^1 (6, 5, 0)	$19.82 \pm 1.09\%$
24	τ_3^2	$16.35 \pm 0.77\%$
25	$\tau_5^{0.5}$	$15.00 \pm 0.85\%$
26	τ_6^1	$13.14 \pm 0.69\%$
27	$\tau_{21}^{0.5}$	$5.08 \pm 0.40\%$
28	τ_{10}^1	$4.94 \pm 0.37\%$
29	$\tau_{13}^{0.5}$	$4.15 \pm 0.44\%$
30	τ_5^2	$2.75 \pm 0.34\%$
31	τ_7^2	$0.63 \pm 0.25\%$

Table 3.2: Ranking of the 31 selected observables in order of importance, with the most important corresponding to the largest drop in accuracy of the model. The corresponding distributions and EFP graphs are displayed in Figures 3.11 and 3.12.

We recall how Figure 3.10 shows that the accuracy of the networks has a dependence on the jet mass. We investigate this dependence in Figures 3.13 and 3.14 by showing the contour plots of the selected observables versus the jet mass. The contour plots indicate that there is a correlation between the top-ranked observables and the jet mass. This correlation is particularly striking for the τ_1^1 variable, which is the fourth most important observable and can be interpreted as a measure of how collimated the jet constituents are, with lower τ_1^1 values corresponding to more collimated constituents. By inspection of this observable, we can explain the lower accuracy at lower jet mass values, as jets with lower jet masses may be more collimated and thus harder to classify.

Lastly, we note that an observable that may be important for the classification of a given class may be irrelevant for another. We measure the reliance of each class on each of the 31 selected observables by measuring the drop in class accuracy when the observables are shuffled. Figures 3.15 and 3.16 show the drop in class accuracy of the top 10 observables per class. For the $N = 1, 2, 3, 4b, 4q$ classes, the top observables mainly consist of EFP variables with four or fewer nodes and N-subjettiness variables with three or fewer sub-jet axes. We note that the constituent multiplicity and jet mass are highly important for the $N = 2$ class. It is also interesting how the $N = 4b, 4q$ classes share some top observables, but the ranking of the observables varies between the classes.

For most classes, the most relevant observables are two-node EFPs or N-subjettiness variables with relatively small N (N =1, 2, 3, or 6). Subsequent observables generally consist of EFPs with more complex shapes or N-subjettiness variables with larger N. This suggests that the network may be focusing on distinguishing between the different classes of jets by first utilizing observables that broadly capture the number of sub-jets and later utilizing more complex observables that specialize in capturing specific traits for each class, such as the presence of collimated jets or subtle differences in the topologies of the jets. The latter is further explored in Section 3.7.

3.7 Topology Dependence

The networks have learned to distinguish jets with various numbers of hard sub-jets. We do not, however, claim that they have learned to identify *any* jet with this number of hard sub-jets. On the contrary, it is likely that the patterns of energy depositions depend on the

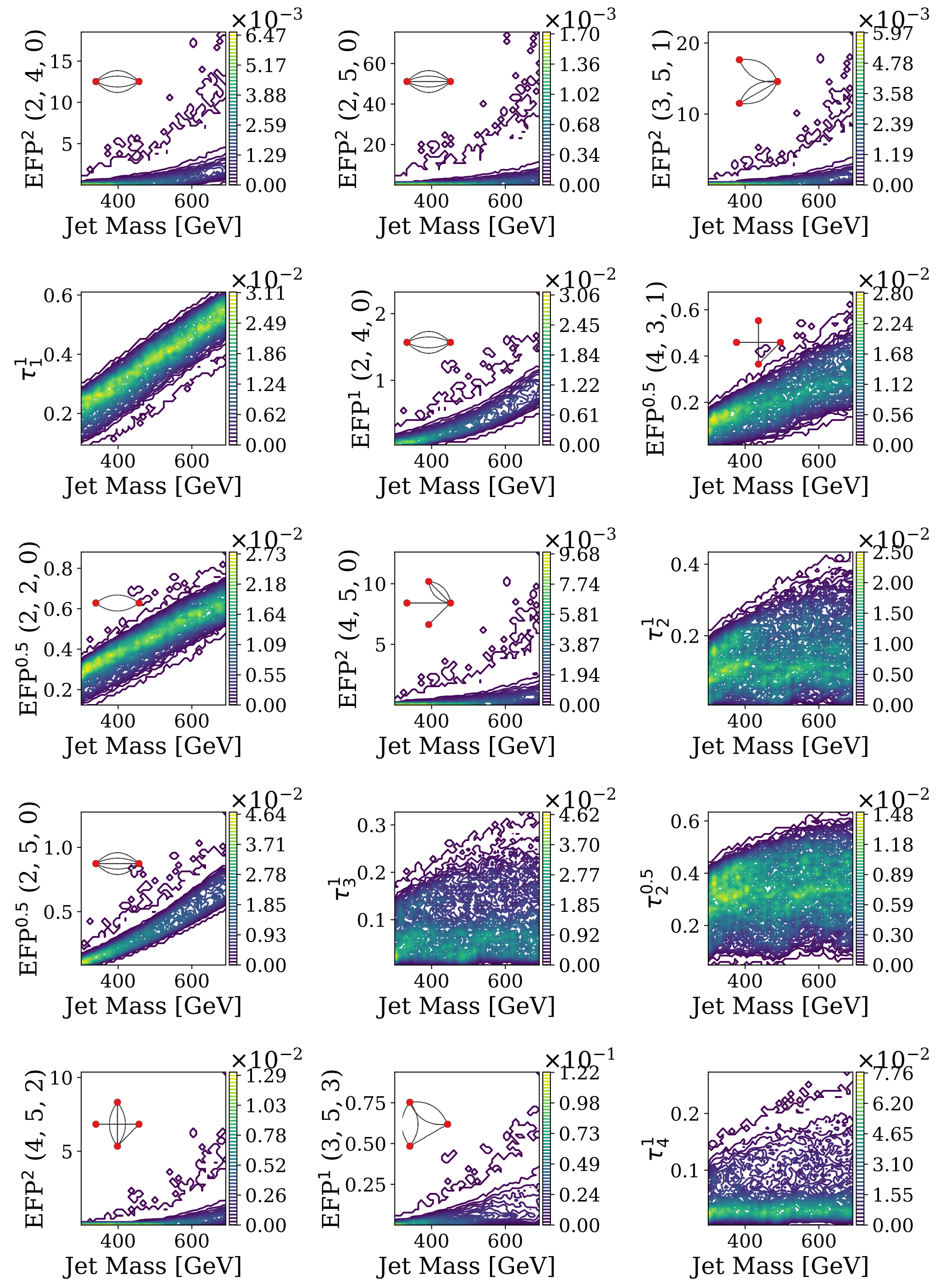


Figure 3.13: Contour plots showing the probability density of the first half of the selected observables versus the jet mass. The plots are ranked in order of importance, excluding the normalized jet mass, which ranks at number 12 and is omitted for brevity. The contour plots of the rest of the selected observables are shown in Figure 3.14.

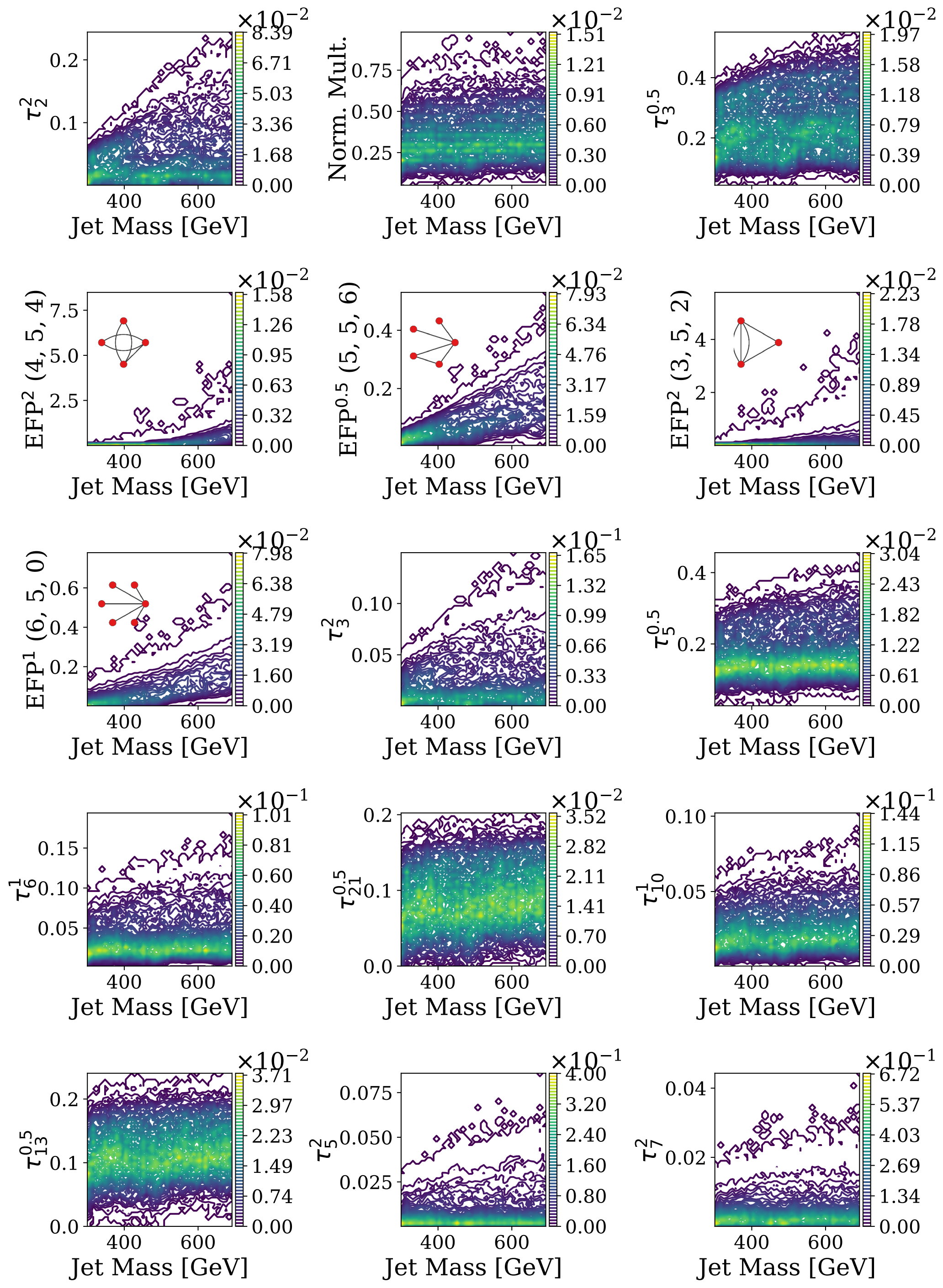


Figure 3.14: Contour plots showing the probability density of the second half of the selected observables versus the jet mass. The contour plots of the rest of the selected observables are shown in Figure 3.13.

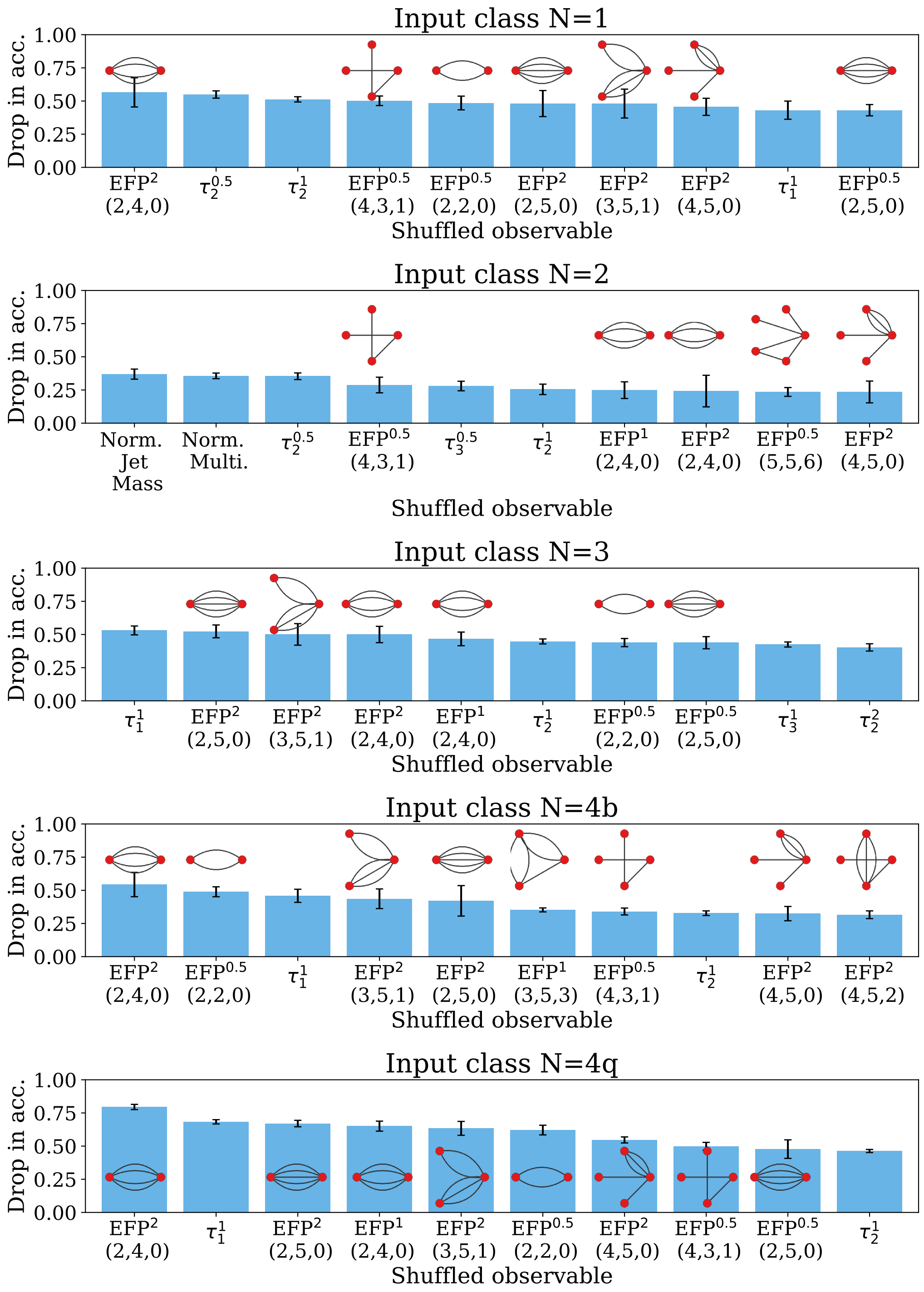


Figure 3.15: Accuracy drop and statistical uncertainty of the top 10 observables per input class $N = 1, 2, 3, 4b, 4q$. The observables are ranked in order of importance, where the most important are those resulting in the largest drops in the accuracy of the model when randomly replaced with observable values from the training set. The EFP graphs are included for visualization purposes.

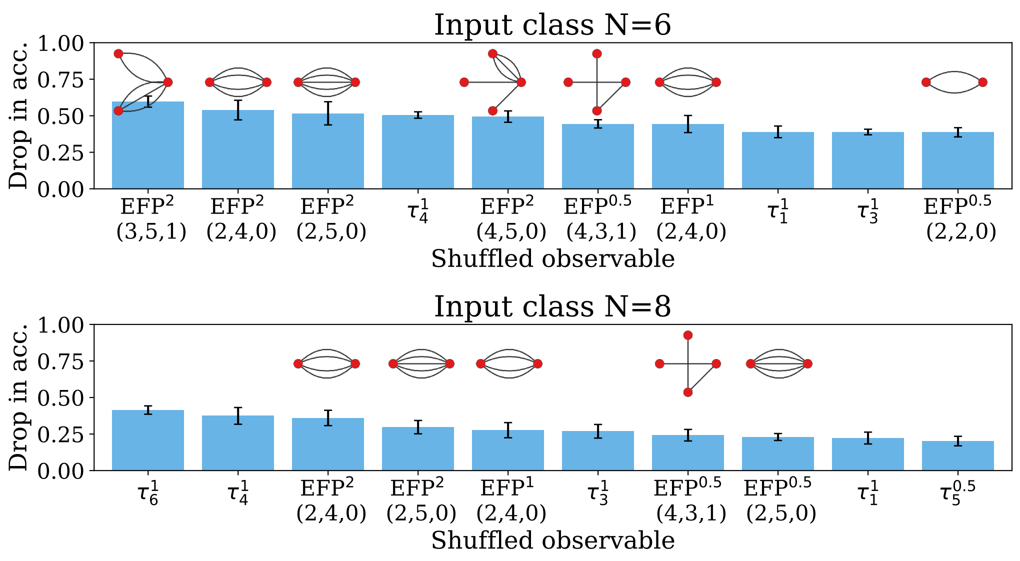


Figure 3.16: Accuracy drop and statistical uncertainty of the top 10 observables per input class $N = 6, 8$. The observables are ranked in order of importance, where the most important are those resulting in the largest drops in the accuracy of the model when randomly replaced with observable values from the training set. The EFP graphs are included for visualization purposes.

details of the topology, such as the invariant mass of intermediate resonances and the jet flavors. The networks' ability to distinguish jets from $G \to HH \to 4b$ and $G \to WW \to 4q$ is an example. In this section, we explore the dependence of the networks' classification strategies on these details of the sub-jet topology.

We generate two additional samples of jets with $N = 4$ hard sub-jets. In the first, labeled $4bM_W$, jets are produced with the $G \to HH \to 4b$ process, but the mass of the Higgs boson has been set to the mass of the W boson to more closely align with the $4q$ sample. In the second, labeled $4qM_H$, jets are produced with the $G \to WW \to 4q$ process, but the mass of the W boson mass has been set to the mass of the Higgs boson to more closely align with the $4b$ sample. Like with the other samples, the $4bM_W$ and $4qM_H$ samples are selected such that they have uniform distributions in jet mass and p_{T}.

We evaluate the networks' predictions on the $4bM_W$ and $4qM_H$ samples. The frequency of classification outputs of these two samples is shown in Figure 3.17. In the $4qM_H$ case, all three networks mainly classify the samples as $N = 4b$ jets and rarely classify them as $N = 4q$ jets. This suggests a strong correlation between the intermediate masses and the final state kinematics that the networks are learning. Conversely, in the $4bM_W$ case, the high- and low-level networks result in different classification predictions. The low-level networks mainly classify the $4bM_W$ samples as $N = 3$ jets, while the DNN_{31} mainly classifies these samples as $N = 4q$ or $N = 3$. The more frequent prediction of $N = 3$ by the low-level networks hints at the possibility that these networks are identifying complex details of the jet topology, which goes beyond simply identifying hard sub-jets, as the $N = 3$ sample includes an intermediate W boson as well as a b-jet.

The different predictions by the networks come as no surprise since they each have different learning strategies. The DNN_{31} learns functions of the 31 selected observables, the PFN learns an event-level latent representation of the jets by summing over the constituents, and the Transformer uses self-attention mechanisms to determine which parts of the constituent sequence it should focus on when making predictions.[5] It is interesting to note that despite their different learning strategies, all taggers seem to be dependent on the detailed topologies, which is a benefit rather than a flaw in studies aiming to classify specific sub-jet topologies.

3.8 Multi-Class to Binary Classification

Multi-class output networks like the ones used in this paper can efficiently distinguish among the different types of jets. A more realistic case, however, is the separation of jets with many hard sub-jets from the overwhelming background of $N = 1$ jets from QCD production. Here, we reinterpret the 7-dimensional network output vector $\boldsymbol{\alpha}$ to perform six binary classification tasks.

[5]We analyzed the redundancy of the networks' strategies by computing the boost in performance obtained by concatenating their outputs. A mild boost in performance was found when concatenating the outputs, suggesting that the networks are using unique information. Further investigation of the nature of this unique information was reserved for future work.

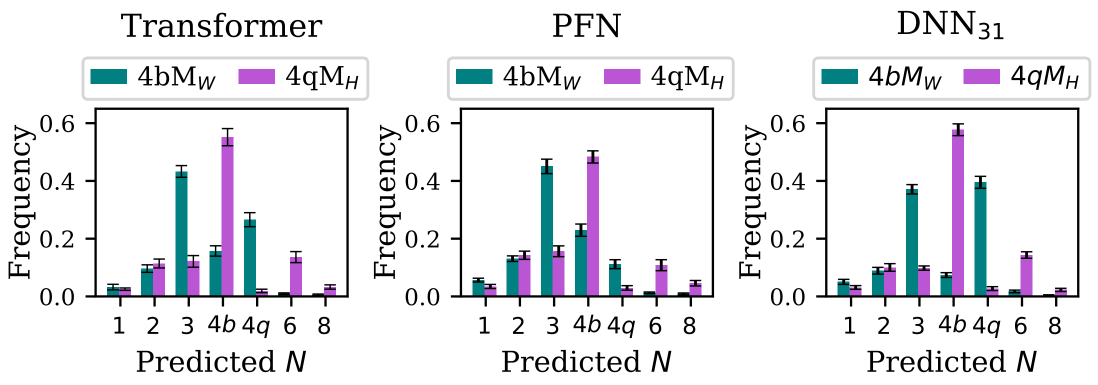


Figure 3.17: Frequency of class identification on samples with modified topology ($4bM_W$ and $4qM_H$) for networks trained with the standard topologies ($N = 1, 2, 3, 4b, 4q, 6, 8$).

For discrimination between $N = 1$ and $N = k$ classes, we define the *decision contrast score*:

$$C_k = \frac{\alpha_k}{\alpha_k + \alpha_1}, \tag{3.2}$$

where α_1 is the element in α corresponding to the probability of a jet having $N = 1$ sub-jets. Likewise, α_k corresponds to the probability of the jet having $N = k$ sub-jets. It is easy to see that C_k tends to 1 if the network predicts a higher probability of the $N = k$ class, and that it tends to zero if it predicts a higher probability of the $N = 1$ class. The binary classification performance can then be measured using the area under the receiver operating curve (ROC-AUC); see Table 3.3 for the ROC-AUC scores and Figure 3.18 for the ROC curves.

The binary classification results agree with the confusion matrices in Figure 3.5, which show a larger degree of confusion between the $N = 1$ and the $N = 2, 3, 4b$ classes and almost perfect classification between the $N = 1$ and the $N = 4q, 6, 8$ classes.

As a cross-check, the PFN and DNN$_{31}$ are retrained to perform the binary classification tasks. The results are included in Table 3.3 as PFN$_{binary}$ and DNN$_{31,binary}$. The multi-class taggers slightly outperform the binary taggers, particularly for lower N. This suggests that training on the different N-sub-jet classes may have resulted in a more robust decision boundary

Model	$N = 2$	$N = 3$	$N = 4b$
Transformer	99.06 ± 0.26	99.58 ± 0.11	99.66 ± 0.11
PFN	99.19 ± 0.13	99.31 ± 0.14	99.57 ± 0.12
PFN$_{binary}$	98.84 ± 0.2	99.01 ± 0.20	99.29 ± 0.17
DNN$_{136}$	97.04 ± 0.28	98.97 ± 0.16	99.39 ± 0.076
DNN$_{299}$	98.07 ± 0.23	99.36 ± 0.17	96.61 ± 0.063
DNN$_{31}$	98.02 ± 0.22	99.31 ± 0.16	99.57 ± 0.068
DNN$_{31,binary}$	97.74 ± 0.28	99.03 ± 0.13	99.33 ± 0.11

Model	$N = 4q$	$N = 6$	$N = 8$
Transformer	99.98 ± 0.021	99.95 ± 0.033	99.95 ± 0.047
PFN	99.93 ± 0.039	99.91 ± 0.048	99.93 ± 0.034
PFN$_{binary}$	99.79 ± 0.079	99.87 ± 0.052	99.84 ± 0.1
DNN$_{136}$	99.96 ± 0.016	99.92 ± 0.038	99.96 ± 0.021
DNN$_{299}$	99.98 ± 0.014	99.96 ± 0.026	99.98 ± 0.014
DNN$_{31}$	99.97 ± 0.018	99.95 ± 0.036	99.97 ± 0.021
DNN$_{31,binary}$	99.91 ± 0.044	99.95 ± 0.034	99.92 ± 0.034

Table 3.3: ROC-AUC percentage score for classifying jets with $N = k$ versus $N = 1$ hard sub-jets, averaged over results from 10-fold cross-validation.

for the $N = 1$ class. This hypothesis aligns with the results in [81, 82], where training on sub-classes helped improve multi-class classification performance.

3.9 Discussion

In this chapter, we have studied the task of classifying jets with a large number of hard sub-jets (up to $N = 8$). We have trained a network on a very typical set of high-level variables: jet mass and N-subjettiness observables. Our studies show that these observables are good sub-jet discriminators, particularly for jets with many hard sub-jets or large jet masses. However, we show that a dense network trained on these observables falls short in classification accuracy when compared to two low-level networks, a PFN and a Transformer, trained directly on the jet constituents. The results suggest that the low-level networks may be using information not captured by the observables.

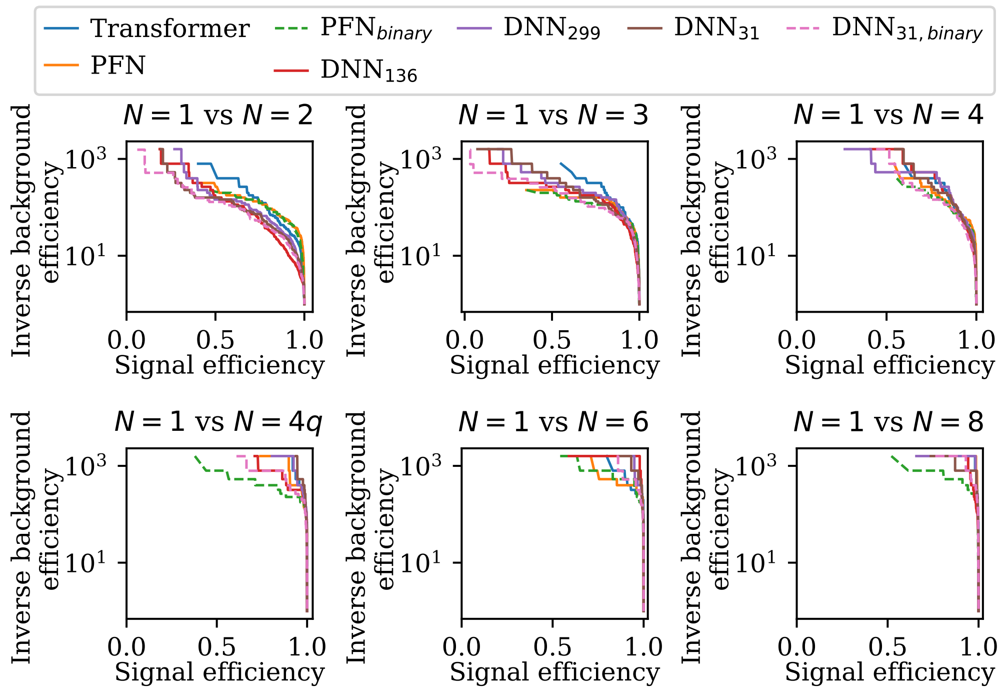


Figure 3.18: ROC curves of the various network studies classifying jets with $N = k$ versus $N = 1$ hard sub-jets. See Table 3.3 for the ROC-AUC percentage scores.

We attempt to bridge the gap between networks operating on low- and high-level information by supplementing the observable set with EFP observables. EFPs form a basis of IRC-safe information, which makes them good candidates for features implementing the observable set. By augmenting the high-level dataset with EFPs, we are able to match the performance of the PFN. However, a small but non-negligible gap in performance still remains with respect to the Transformer. This suggests that the Transformer may be utilizing information not captured even by the augmented high-level dataset. The nature of this missing information is studied in Chapter 4.

To gain insights into the classification strategies used by the classifiers, we find a subset of features that capture most of the information in the augmented observable dataset. This is done by employing a feature selection technique inspired by LASSO regularization, which narrows down the number of observables by zeroing out the weights of irrelevant observables during training. With this technique, we identify 31 observables that well-approximate the performance of the full observable dataset. The 31 features are ranked according to their individual contributions to the dense model by measuring the drop in accuracy when each feature is shuffled during test time. By this measure, we find that the most important observables are mainly EFPs with four or fewer nodes and N-subjettiness variables with N $<$ 8. We also find that the constituent multiplicity is one of the top observables, and it is particularly relevant for classifying $N = 2$ jets.

Our studies suggest that the dense classifier may focus first on simple observables that broadly capture the number of sub-jets, such as EFP polynomials with only a few nodes or N-subjettiness variables with small N values. Later, the classifier may utilize more complex observables to capture subtle traits of the topology of the jets. This is further confirmed in our results, which reveal the classifiers to have a strong topology dependence, as they appear to be sensitive to the sub-jet resonance masses and flavor rather than simply being sensitive to the sub-jet multiplicity. Future work may involve disentangling the nature of this information to probe more deeply the classification strategy for high-multiplicity sub-jets.

Our results bode well for future jet studies in high-energy collider settings, where jets with additional hard sub-jets will become more important as the high-luminosity LHC collects large datasets in which high-p_{T} objects appear in greater numbers [61, 62].

Chapter 4

Jet Rotational Metrics

4.1 Introduction

In this chapter, we extend our studies of multi-prong jet classification to include a new family of novel observables called Jet Rotational Metrics (JRMs). Embedding symmetries in the architectures of deep neural networks has generally resulted in improved performance and faster network convergence in the context of jet classification [69,83–87]. These results hint at the existence of symmetries in the jet energy depositions, such as rotational symmetry, arising from physical features of the underlying processes. Having compact, high-level features that capture the symmetries in the data could increase the performance of jet classification studies while helping physicists quantify the statistical uncertainties of the data and the models.

Several jet observables have been utilized as high-level features to distinguish between jets of different topologies, such as the jet thrust [88, 89], angularities [90, 91], energy-flow polynomials (EFPs) [4], and N-subjettiness [92], among others. These observables provide great insights into the interior of jets, but they do not provide a robust measure of their degree of rotational symmetry. For example, quark- and gluon-jets are expected to have isotropic-like patterns of energy depositions, thus having a high degree of rotational invariance. On the other hand, multi-prong jets could have patterns of energy depositions that are not isotropic but instead display multiple areas rich in harder constituents. While these jets are less likely to be rotationally invariant, they may show varying degrees of discrete, or n-fold, rotational symmetry (C_n).

A common approach is to treat symmetries in a binary fashion and determine whether data possesses a given symmetry or not. But in reality, symmetries exist along a spectrum, and data may possess a given symmetry in varying degrees [93]. Based on this notion, and on continuous symmetry measures [94], we introduce *Jet Rotational Metrics* (JRMs)[1], a new type of jet observables that provide insights into the degree of n-fold rotational symmetry of a jet. JRMs are defined by the similarity (S) between a jet (J) and a reference jet (J_n) with n constituents arranged to have exact C_n symmetry:

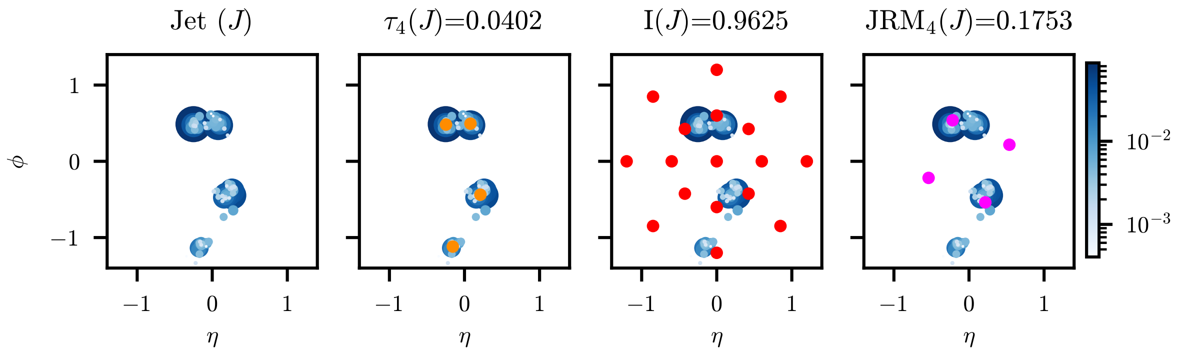
$$\mathrm{JRM}_n(J) \propto S(J, J_n). \tag{4.1}$$

The construction of J_n is discussed in Section 4.3. We emphasize that JRMs measure the deviation between J and the specific C_n symmetry element in the reference jet. We do not claim to measure a jet's absolute deviation from C_n symmetry.

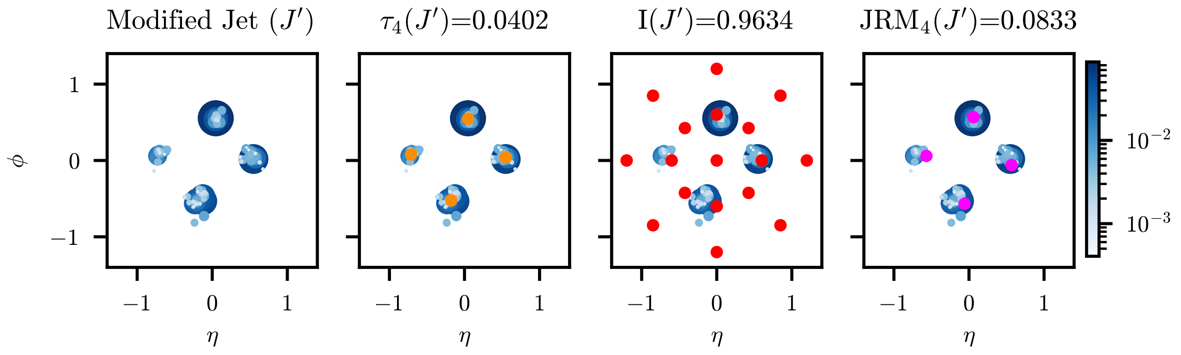
While observables like sphericity [95] and event isotropy [2] provide comparable measures by estimating the degree of isotropy in an event, and thus also its degree of rotational symmetry, we find that breaking down this symmetry into its discrete form results in better network performance when classifying jets with multiple sub-jets. As an example, in Fig. 4.1, we compare the differences between N-subjettiness (τ), a measure of isotropy (I) similar to those introduced in [2], and JRMs. Fig. 4.1a shows a four-prong jet with 4-axis N-subjettiness variable of $\tau_4 \approx 0.04$, isotropy measure of I ≈ 0.96, and JRM gauging C_4-symmetry of $\mathrm{JRM}_4 \approx 0.18$. In Fig. 4.1b, the same jet is modified such that it has a more symmetric

[1]We call them "Jerms."

constituent dispersion while maintaining the same τ_4 value. We find that the modified jet has a similar isotropy measure to the original jet, $\mathrm{I} \approx 0.96$. The most dramatic difference is in the JRM, which decreases to $\mathrm{JRM}_4 \approx 0.08$. In this example, the isotropy measure does not capture the change in the constituent dispersion as well as the JRM_4. This is likely because the discretization of the rotational symmetry helps capture more subtle changes in the jet's geometry.



(a) Comparison of the various observables calculated on a jet (J).



(b) Comparison of the various observables calculated on a jet (J'), which has been modified to be more C_4-symmetric.

Figure 4.1: Panel (a) shows a four-prong jet (J). The second image in this panel shows the four sub-jet axes (orange dots) used to calculate the N-subjettiness τ_4 measure. The third image shows the isotropic event, J_{iso} (red dots), used in the calculation of the isotropy measure; $\mathrm{I}(J) \equiv \mathrm{EMD}(J, J_{\mathrm{iso}})$ [2], where EMD is the Energy Mover's Distance [3]. Lastly, the fourth image shows the C_4-symmetric event, J_4 (pink dots), used in the calculation of the JDM_4; $\mathrm{JDM}_4 \equiv S(J, J_4)$. The images in panel (b) show similar events but with the jet modified to have a higher degree of C_4 symmetry (J'). The area and color intensity of the jets' constituents is proportional to their p_{T}.

The classification power of JRMs is tested on the dataset from Section 3. We show that JRMs achieve a good classification performance on their own. We also show that when combined with N-subjettiness and EFPs observables, JRMs increase the classifier's performance significantly; a dense network trained on these parameters surpasses the accuracy of the PFN and Transformer introduced in Section 3.4. The N-subjettiness and EFP families of observables form complete bases of IRC-safe information, and in principle, a combination of such observables could also gauge the degree of C_n symmetry of the jets. In practice, however, classifiers operating on these observable families tend to saturate quickly with the number of input features [4], which means that they may not capture this information as efficiently as JRMs. Evidence of this is in the boost in performance obtained by supplementing the N-subjettiness and EFPs with JRMs.

The rest of this chapter is organized as follows: Section 4.2 introduces the similarity measure used to compare the jets. Section 4.3 details the procedure to calculate the JRMs. Section 4.4 presents the results of using JRMs as high-level features for jet classification in the benchmark dataset. Concluding remarks and discussion are given in Section 4.5.

4.2 Similarity Measure

JRMs require a similarity measure between two jets. If we consider jets as point clouds, many distance metrics could be used, such as the Chamfer distance [96], the Hausdorff distance [97], the Earth Mover's distance [98] or its physics-inspired modification, the Energy Mover's distance (EMD) [3]. The last measures the work necessary to transform one event into another by solving a system of linear optimal transport equations. Any of these metrics could be used in the calculation of the JRMs, but many are computationally expensive[2]. A useful,

[2]We performed a test comparing the results of JRMs using EMD vs. the simple similarity measure used in this paper and found them to be comparable in performance, but the EMD is more computationally expensive.

cost-effective alternative is to sum the p_{T}-weighted distance between nearest neighbors:

$$\mathrm{S}(J, J_n)^{\beta} = \sum_{i \in J} p_{\mathrm{T}i} \min_{j \in J_n} \left(\frac{\Delta R_{i,j}}{R} \right)^{\beta}, \tag{4.2}$$

where i and j index over the constituents of J and J_n, respectively. Each constituent i in J queries its nearest neighbor in J_n. For massless constituents, the pairwise distance $\Delta R_{i,j}$ is their distance in the (η, ϕ) plane[3], which is normalized by the jet radius R. The distance between nearest neighbors is weighted by $p_{\mathrm{T}i}$, the transverse momentum of constituent i. The free parameter β controls the weight of the angular terms.

Eq. 4.2 resembles the clustering inertia used to measure how well a dataset is clustered by the centroids in k-means clustering, which is also used in N-subjettiness. For simplicity, we refer to this measure as the "distance" between J and J_n, though this is not a true distance function as the triangle inequality does not hold.

The distance measure defined in Eq. 4.2 is IRC-safe (for $\beta > 0$), but can be generalized to non-IRC-safe versions by varying the exponent of the energy term:

$$\mathrm{S}(J, J_n)^{\kappa,\beta} = \sum_{i \in J} p_{\mathrm{T}i}^{\kappa} \min_{j \in J_n} \left(\frac{\Delta R_{i,j}}{R} \right)^{\beta}. \tag{4.3}$$

Small κ values ($\kappa < 1$) enhance the contributions of softer constituents while large κ values ($\kappa > 1$) dampen their contributions. Varying κ can provide a sense of the relative dispersion of the softer/harder constituents. Similarly, varying β can provide a sense of the relative importance of the geometric terms.

[3]Or in the (y, ϕ) plane if constituents are not massless.

For consistency, we preprocess jets to have net p_{T} of unity and center them on their p_{T}-weighted mean in the (η, ϕ) plane. For each constituent $i \in J$:

$$p_{\mathrm{T}i} \to p_{\mathrm{T}i} \,/ \sum_{j \in J} p_{\mathrm{T}j}, \tag{4.4}$$

$$\eta_i \to \eta_i - \sum_{j \in J} p_{\mathrm{T}j}\, \eta_j / \sum_{j \in J} p_{\mathrm{T}j}, \tag{4.5}$$

$$\phi_i \to \phi_i - \sum_{j \in J} p_{\mathrm{T}j}\, \phi_j / \sum_{j \in J} p_{\mathrm{T}j}, \tag{4.6}$$

By normalizing the p_{T}, Eq. 4.2 is dimensionless and lies in the range $[0, 1]$. $\mathrm{S}(J, J_n) = 0$ indicates that the constituents in J are perfectly arranged like those in J_n while $\mathrm{S}(J, J_n) \approx 1$ indicates a dissimilar arrangement.

4.3 Gauging C_n Rotational Symmetry

We recall that JRMs measure the similarity between a jet J and a reference jet J_n with C_n symmetry. A potential obstacle is that there are infinite possible choices of J_n, each with perfect C_n symmetry but distinct elements. Ideally, one would choose the reference jet that is closest to J, minimizing the distance metric. In practice, an exhaustive search would be computationally prohibitive. To simplify the calculation, we develop a search recipe for the reference jet:

1. Consider n points in a circle centered at $(\eta, \phi) = (0, 0)$. The points are located at the angles $2\pi i/n$, $i = 0, \ldots, n-1$. These points represent the n constituents in the reference jet.

2. Let the points be located at a distance equal to the p_{T}-weighted mean constituent radius $(\bar{r})$.

3. Rotate the points by a common θ to minimize the distance between J and J_n, as measured by Eq. 4.3.

Following the steps specified above, J_n is characterized by three parameters: the number of points (constituents) n, the radius $\bar{r}$, and the rotation angle $\theta \in [0, 2\pi/n)$. With this in mind, JRMs can then be defined as

$$\mathrm{JRM}_n^{\kappa,\beta} \coloneqq \min_{\theta} \mathrm{S}(J, J_n(\bar{r}, \theta))^{\kappa,\beta}. \tag{4.7}$$

For clarity, the measure with $\kappa = 1$ and $\beta = 1$ is simply written as JRM_n. See Appendix B.1 for further discussion about the construction of J_n.

An illustration of JRMs with $n = 2, 3, 4, 8$ for a two-prong jet is shown in Fig. 4.2. Interestingly, $\mathrm{JRM}_3 > \mathrm{JRM}_2$, which makes sense considering how the constituents of the jet are mainly concentrated in two areas, and it would be hard to compare them to a reference jet with three equidistant constituents. By contrast, the same observables are illustrated in Fig. 4.3 for an eight-prong jet. The constituents in the jet are distributed more uniformly, which makes the JRM measure decrease with n; $\mathrm{JRM}_{n-1} > \mathrm{JRM}_n$. An analysis of JRM ratios is shown in Appendix B.3.

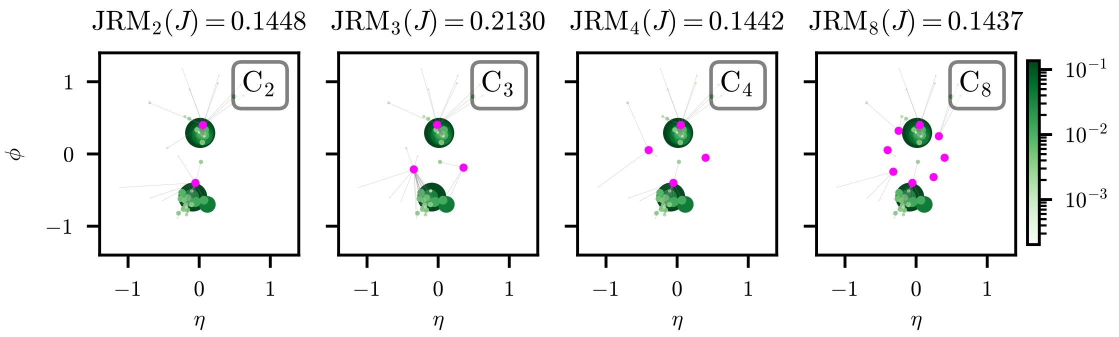


Figure 4.2: Illustration of the various JRM_n observables calculated on the same two-prong jet (J) in green. The area and color intensity of the constituents is proportional to their p_{T}. The pink dots represent the constituents of the C_n symmetric jet (J_n), which is compared against J. The gray lines depict the nearest neighboring constituents between J and J_n.

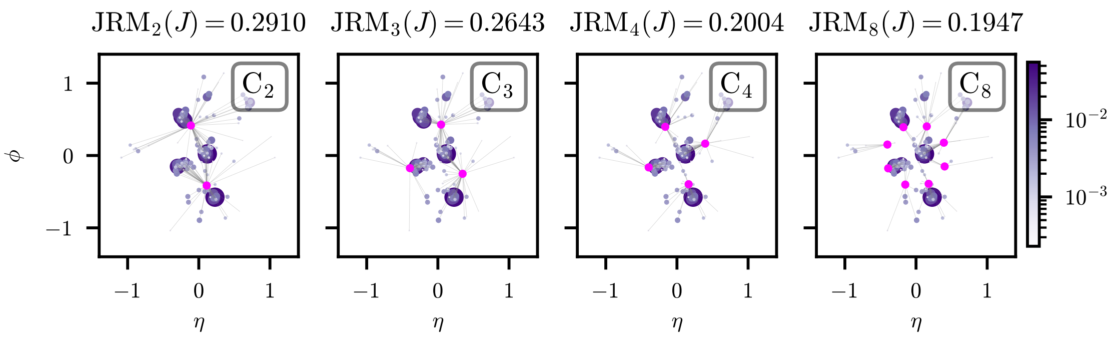


Figure 4.3: Illustration of the various JRM_n observables calculated on the same eight-prong jet (J) in purple. The area and color intensity of the constituents is proportional to their p_T. The pink dots represent the constituents of the C_n symmetric jet (J_n), which is compared against J. The gray lines depict the nearest neighboring constituents between J and J_n.

In Fig. 4.2, we note how similar JRM_2, JRM_4, and JRM_8 are for the two-prong jet. Because of the nearest neighbor operation, the majority of the constituents are assigned to only two reference constituents in the JRM_4 and JRM_8 observables. While this may seem like a weakness, we find that attempts to penalize the measure for an unbalanced constituent assignment result in slightly lower performances by the classifiers. This suggests that the networks may find useful information in JRM_ns having similar values for various n[4].

4.4 Multi-Prong Jet Classification

In this section, we evaluate the performance of JRMs in the classification of jets with multiple sub-jets using the dataset introduced in Section 3.2. We anticipate JRMs to perform well at this classification task since the various topologies that produce jets with different numbers of hard sub-jets are expected to have patterns of energy deposition with varying degrees of n-fold rotational symmetry. We recall that the dataset consists of jets with $N = 1, 2, 3, 4, 6, 8$ hard sub-jets. The $N = 4$ class is subdivided into $N = 4q$ and $N = 4b$, for a total of seven classes.

[4]Further evidence of the networks subtracting relevant information from JRMs using the simple distance measure from Eq. 4.3 is in how the results are comparable to JRMs using the EMD [3] as the distance measure.

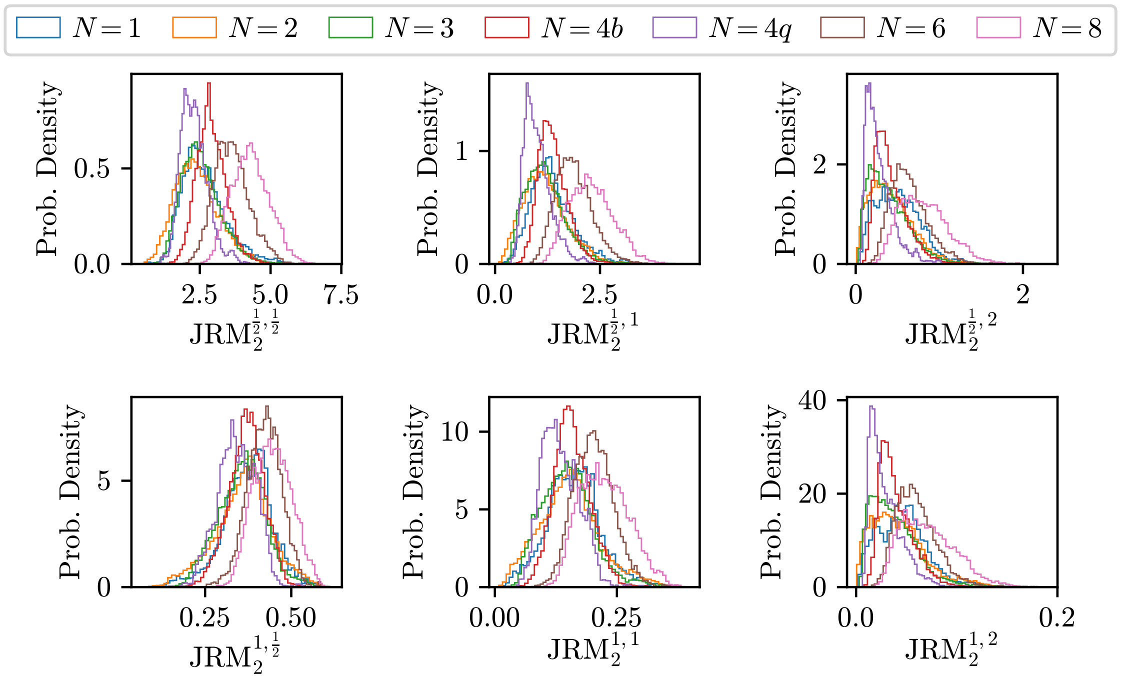


Figure 4.4: Distributions of the $\mathrm{JRM}_2^{\kappa,\beta}$ observables with the specified κ, and β parameters.

We calculate 30 $\mathrm{JRM}_n^{\kappa,\beta}$ observables with the following parameter combinations[5]: $n = 2, 3, 4, 6, 8$, $\kappa = \frac{1}{2}, 1$, and $\beta = \frac{1}{2}, 1, 2$. As an example of these observables, we show the distributions of the JRM_2 features in Fig. 4.4. It is apparent that the JRM_2 observables provide useful insights into the substructure of the jets. For example, the $N = 2$ and $N = 4q$ classes have the lowest mean JRM_2 values. Conversely, the $N = 6$ and $N = 8$ classes have the highest mean JRM_2 values, which suggests that the constituents in these classes are less likely to be collimated in such a way that they are concentrated in two symmetric areas.

To highlight the differences between JRM_2 and the 2-axis N-subjettiness features (τ_2^β), we show the latter in Fig. 4.5. The classes with the lowest τ_2 values are $N = 1$ and $N = 2$, while the $N = 4q$ class falls in the middle of all classes. Similarly to JRM_2, the $N = 6$ and $N = 8$ classes have the largest τ_2 values.

[5]We utilized the same n values as the number of sub-jets, which perform well for the classification task, though this may not always be the case. The optimal JRM feature for identifying jets with N sub-jets may have $n \neq N$.

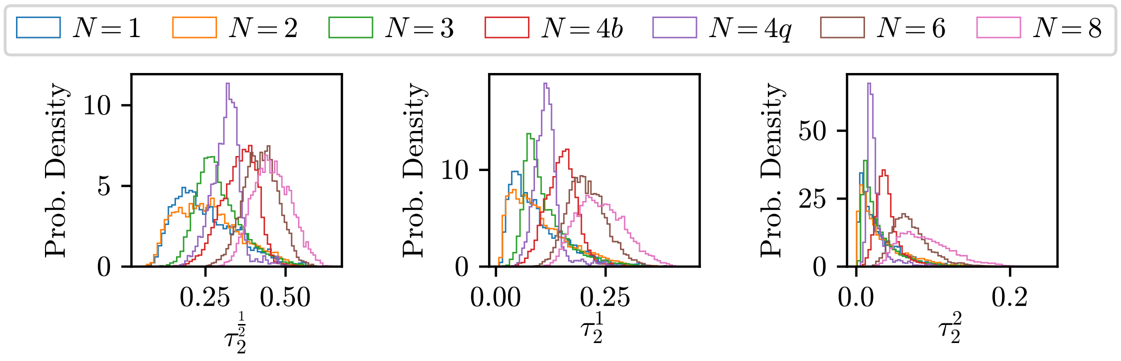


Figure 4.5: Distributions of the N-subjettiness features with N=2 (τ_2^β) observables with the specified β parameters.

The rest of the distributions are shown in Appendix B.4. In our experiments, the JRM$_n$ observables nearly saturate for $n = 8$. This is seen in the distributions in Appendix B.4, where the JRM$_6$ measures resemble JRM$_8$. In addition, including JRM$_n$ observables with $n > 8$ does not increase the classifier's performance.

To assess the performance of these observables and the uniqueness of the information they capture, we compare the performance of dense networks operating on JRMs to those operating on the N-subjettiness and EFP observables from Section 3.3. The set of N-subjettiness variables consists of 135 observables calculated along the k_T axis, with sub-jet axis parameter N $= 1, \ldots, 45$ and angular weighing parameter $\beta = \frac{1}{2}, 1, 2$. The EFP set consists of 162 observables representing prime multi-graphs with five or fewer edges, energy weighing parameter $\kappa = 1$, and angular weighing parameter $\beta = \frac{1}{2}, 1, 2$. We consider both sets to be extensive, as including more N-subjettiness or EFP observables is not likely to increase the performance of the networks, and if so, marginally.

For consistency and to test the boost in performance obtained by including JRMs, all dense networks share the same architecture of six hidden layers of size (800-800-800-800-800-64) and ReLu [36] activation function. Dropout with rate 0.3 and batch normalization are applied respectively after every hidden layer. The output layer has dimension seven and a softmax activation function.

To gain a better understanding of the nature of the information used by the models, we distinguish between IRC-safe and IRC-unsafe observables. Table 4.1 shows the results of the networks operating on IRC-safe observables. The DNN trained on IRC-safe JRMs achieves an overall accuracy of $85.49\pm0.35\%$. The DNN trained on the larger set 297 of N-subjettiness and EFPs exceeds that, with an overall accuracy of $88.94 \pm 0.37\%$. The highest accuracy of $90.72 \pm 0.23\%$ is achieved by combining all observables, which indicates that the JRMs capture unique information not captured by the N-subjettiness or EFP observables. This new information is highly valuable for the classification process since it not only increases the overall accuracy by more than 1.5% but also the accuracy of each class; see Appendix B.2 for the classification accuracies per class.

Table 4.2 shows the results of the models whose input is IRC-agnostic. The PFN and Transformer models are used as the benchmark low-level networks to compare against the high-level features. The DNN trained on the full set of 30 JRMs does very well, achieving an overall accuracy of $89.30 \pm 0.26\%$ and outperforming the overall accuracy of the PFN [48] ($89.19 \pm 0.23\%$), which operates on low-level information, constituent momenta. More significant is the boost in performance obtained when combining all the JRMs with the N-subjettiness and EFPs, which together achieve the top overall accuracy of $91.52 \pm 0.26\%$ and outperform the Transformer ($91.27 \pm 0.31\%$), which also operates on constituent momenta. By including the IRC-unsafe JRM observables, the gap between the Transformer and dense networks is bridged. This suggests that deep, low-level networks trained on the full event information may make use of soft and collinear radiation, which is not well-defined in the perturbative QCD regime. Moreover, these results provide insights into *how* the IRC-unsafe information is used, suggesting that low-level networks may look at the macroscopic structure of the radiation and learn functions of the global dispersion, such as the degree of rotational symmetry. We note that the contributions of the IRC-unsafe features are small when compared to the classification power of the IRC-safe ones, which is expected, but they cannot be discarded when accounting for statistical uncertainty of the low-level networks due to deviations in the particle simulations [99].

Table 4.1: Mean 10-fold prediction accuracy and statistical uncertainty of the DNNs operating on the specified IRC-safe observables. Only the 15 JRMs with $\kappa = 1$ are used as input to the DNNs.

Network	Input (IRC-safe)	Input Dim.	Overall Acc. (%)
DNN	JRM	15	85.49 ± 0.35
DNN	N-subs, EFPs	297	88.94 ± 0.37
DNN	N-subs, EFPs, JRMs	312	$\mathbf{90.72} \pm 0.23$

Table 4.2: Mean 10-fold prediction accuracy and statistical uncertainty of the various networks. The input to the PFN and Transformer is the constituents' three-momentum, which is zero-padded to have a uniform length of 230. See Section 3.4 for further details about the PFN and Transformer networks. The full set of 30 JRMs is used as input to the DNNs.

Network	Input	Input Dim.	Overall Acc. (%)
DNN	JRMs	30	89.30 ± 0.26
DNN	N-subs, EFPs, JRMs	327	$\mathbf{91.52} \pm 0.26$
PFN	Constituents	(230, 3)	89.19 ± 0.23
Transformer	Constituents	(230, 3)	91.27 ± 0.31

4.4.1 Feature Selection

We continue our exploration of JRMs for multi-prong classification by identifying their relevant importance in the classification task. We employ the LASSO-inspired feature selection method introduced in Section 3.6.2 and apply it to the 312 IRC-safe observables.

The feature selection method has identified 29 IRC-safe observables, which are shown in Table 4.3. A DNN trained on these observables reaches an accuracy of $89.97 \pm 0.28\%$, well approximating the accuracy of the full IRC-safe observable set and slightly surpassing the accuracy of the PFN. These results are impressive given the limited number of input features.

The choice of features provides insights into the classification strategies of the dense networks. The family of observables with the most selected features is N-subjettiness, and most of these

Table 4.3: IRC-safe observables chosen by the LASSO-inspired feature selection method, which achieve an overall accuracy of $89.97 \pm 0.28\%$ when used as input to a DNN. The selected N-subjettiness observables, $\tau_{\mathrm{N}}^{\beta}$, are shown in the leftmost column. The selected JRM observables are shown in the middle column. The selected EFP observables, $\mathrm{EFP}^{\beta}(u, d, k)$, are shown in the rightmost column and are accompanied by their unique identifiers indicating the number of nodes (u), edges (d), and index (k); see [4]. The observables are listed in no particular order.

N-subjettiness	JRMs	EFPs
τ_2^1	$\mathrm{JRM}(C2)^{1,0.5}$	$\mathrm{EFP}^{0.5}(2,1,0)$
τ_2^2	$\mathrm{JRM}(C2)^{1,1}$	$\mathrm{EFP}^{2}(2,2,0)$
$\tau_3^{0.5}$	$\mathrm{JRM}(C3)^{1,0.5}$	$\mathrm{EFP}^{2}(2,5,0)$
τ_3^1	$\mathrm{JRM}(C3)^{1,2}$	$\mathrm{EFP}^{0.5}(3,3,1)$
τ_4^1	$\mathrm{JRM}(C3)^{1,1}$	$\mathrm{EFP}^{2}(3,4,3)$
τ_5^1	$\mathrm{JRM}(C4)^{1,0.5}$	$\mathrm{EFP}^{2}(4,5,0)$
τ_7^1	$\mathrm{JRM}(C4)^{1,1}$	$\mathrm{EFP}^{2}(4,5,3)$
$\tau_{16}^{0.5}$	$\mathrm{JRM}(C6)^{1,0.5}$	
$\tau_{22}^{0.5}$	$\mathrm{JRM}(C6)^{1,2}$	
$\tau_{39}^{0.5}$	$\mathrm{JRM}(C8)^{1,1}$	
$\tau_{39}^{0.5}$		
$\tau_{45}^{0.5}$		

variables have N< 8, which makes sense given the pronginess of the jets in the dataset. The next family of observables is JRMs. For every C_n symmetry considered, at least one JRM observable is chosen. These results suggest that the network utilizes the measures of discrete rotational symmetry. The feature selection method also chooses a fair amount of EFP variables with 2, 3, and 4 nodes. Some of these observables correlate to other well-known jet observables, such as EFP(2, 1, 0) and the jet mass [4].

The 29 IRC-safe observables listed in Table 4.3 do a great job capturing information relevant to the classification task. However, we recall how including IRC-unsafe observables increased the performance of the dense networks, slightly surpassing that of the Transformer. We do not perform feature selection on the IRC-unsafe JRMs, but we note that including them in the training set still shows a boost in performance. For example, adding the five IRC-unsafe JRMs with $n = 2, 3, 4, 6, 8$, $\kappa = \frac{1}{2}$ and $\beta = 1$ to the 29 selected IRC-safe observables increases the overall accuracy to 90.78 ± 0.38.

4.5 Discussion

In this chapter, we have introduced a new family of jet observables, JRMs, that gauge the degree of discrete, or n-fold, rotational symmetry of a jet. The core idea behind JRMs is that the patterns of energy deposition of a jet are highly dependent on features of the underlying process, such as the number of sub-jets and the resonance masses, which could result in jets displaying varying degrees of n-fold rotational symmetry. For example, quark- and gluon-jets are expected to have uniform radiation patterns that yield isotropic constituent dispersions. On the other hand, multi-prong jets could have constituent dispersions that are not isotropic but instead have multiple areas rich in harder constituents.

We have tested the discriminative power of JRMs on a benchmark dataset for classifying jets with multiple sub-jets, more specifically with $N = 1, 2, 3, 4, 6, 8$ sub-jets. We find that the JRMs are very useful discriminants, achieving formidable classification accuracies when used as input to dense neural networks. Moreover, we find that when combining JRMs with more traditional observables, such as N-subjettiness and EFP variables, the dense neural networks manage to outperform two benchmark low-level networks. From these results, we draw two conclusions: first, the JRMs capture *unique* information not captured by the more traditional observables. Second, this unique information is *highly relevant* for the classification task, bridging the gap between high- and low-level networks and shedding light on the learning strategies used by these networks.

Recent advances in machine learning for jet classification have focused on deep, low-level networks. These networks learn functions directly from the jet constituents, so it is no surprise that they currently provide state-of-the-art results in many applications [46, 64–67]. However, the high performance of these networks comes with the downside of lower interpretability as it is not clear which specific functions of the complex, low-level input they have learned.

On the other hand, high-level observables, like JRMs, make it more feasible for physicists to understand the nature of the information learned by the networks. These observables are often used as inputs to dense neural networks, which are easier to interpret since we can control the functional forms of their inputs. An example of this is observables that are IRC-safe, an important property to test the substructure of an event while being insensitive to soft and collinear radiation. Although observable families like N-subjettiness and EFPs form bases of IRC-safe information, capturing all such relevant information in a small set of features has been proven difficult, often due to classifiers saturating quickly with respect to the number of features. We show that the IRC-safe JRMs are good complements to these observable families, likely because they capture information not efficiently captured by a small number of N-subjettiness or EFPs variables.

In addition, we find a small but non-negligible reliance of networks on observables with IRC-unsafe forms. This finding is important as it shows that low-level networks may rely on IRC-unsafe information, which is not well-defined in the perturbative QCD regime.

Lastly, we acknowledge that there could be many natural extensions to the JRMs, either by utilizing different forms of C_n-symmetric elements to measure the rotational symmetry or by employing different distance metrics. We leave further exploration of these choices for future work.

Chapter 5

Systematic Uncertainties in LHC data

5.1 Introduction

This dissertation has so far focused on aspects of multi-prong jet classification. In this chapter, we change gears and focus on another important aspect of data analyses at the LHC: the estimation of systematic uncertainties.

As the LHC continues to collect large experimental datasets, precise measurements of systematic uncertainties will become increasingly more vital. For instance, precision tests of the SM require the comparison between experimental measurements of particle properties and theoretical predictions. In many cases, potential discrepancies between measurements and predictions rely on the accurate estimation of systematic uncertainties. The recent measurement of the W boson mass [100] and the muon anomalous magnetic moment [101] are two examples where it has been claimed that the experimental measurements vary from the theoretically predicted values. These results are meaningful because they claim that the discrepancy is larger than the systematic uncertainties. However, a full, rigorous assessment of the systematic uncertainties is often computationally intractable since experiments generally have several sources of uncertainty, which create a high-dimensional space that must be characterized. To overcome this issue, most measurements rely on simplifying assumptions, such as the factorization of the various sources of uncertainty.

When dealing with multiple sources of uncertainty, several procedures have been established to simplify their estimations by factorizing their underlying correlations. Common procedures include orthogonalization [102–104] and treatment of residual correlations [105, 106]. The advantages and limitations of these procedures have been well-studied in the literature [102]. What has not been as well-studied is the impact that these simplifying assumptions have on the experimental response. For example, if the sources of uncertainty are assumed to be factorizable, then the space of the experimental response is generally also assumed to be factorizable in terms of the individual sources, such that each source corresponds to its own axis within the space. Then, the impact of multiple sources (off-axis) can be extrapolated from the impact of individual sources (on-axis). While this approach is simple and computationally inexpensive, it relies on the validity of the assumption that the off-axis values can be calculated from linear combinations of the on-axis values. Such an assumption may be appropriate for smaller, less complex datasets, but it will likely fail in the context of larger datasets with multiple sources of uncertainty.

In the following sections, we describe a strategy for characterizing the experimental response in terms of the systematic uncertainties *without* assuming factorization. Our strategy uses Gaussian processes (GPs) to regress observables of the experimental response given the various sources of uncertainty of a model. Gaussian processes are flexible, nonparametric machine learning models that have been used in high-energy physics studies involving background selection and signal extraction [107–109]. GP applications have been less widely used in estimating systematic uncertainties. We leverage the fact that the functional forms of experimental observables are often known to calculate their gradients and use them to enhance the GP regression. We show that GPs enhanced with derivative information are a powerful tool for estimating systematic uncertainties, even when dealing with non-differentiable observables.

In our example scenarios, the GP approach outperforms the factorized approach with significantly fewer samples. However, in cases where the impact of the sources of uncertainty is not straightforward to learn, few samples may not be enough for a GP model to accurately estimate the observables. In such cases, having a strategy to efficiently sample the space of the experimental response is essential. We use Bayesian experimental design (BED) as the sampling strategy, which we show is more efficient than random or grid sampling, particularly in higher dimensions.

The rest of this chapter is organized as follows: Section 5.2 describes the role of systematic uncertainties in statistical inference and discusses typical assumptions in the assessment of systematic uncertainties. Section 5.3 describes the elements of GP regression and section 5.4 the elements of the BED strategy. In Section 5.5, we demonstrate our method on a one-dimensional toy example, followed by Sections 5.6 and 5.7, where we respectively demonstrate our approach in two-dimensional and four-dimensional realistic high-energy physics scenarios. Section 5.8 contains our conclusions and future work.

5.2 Systematic Uncertainties

Experimental measurements in particle physics are typically based on the statistical inference of theoretical parameters of interest ($\boldsymbol{\theta}$) and nuisance parameters ($\boldsymbol{\nu}$) from experimental data ($\boldsymbol{x}$). The parameters of interest represent parameters of the theory, such as signal strength or resonance masses. In contrast, the nuisance parameters are treated as sources of systematic uncertainty. A major obstacle to inferring the values of these parameters is the lack of access to a tractable likelihood, $p(\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{\nu})$, which would make parameter estimation and limit-setting straightforward. Instead, the likelihood is often approximated as a product of Poisson terms for each bin of a histogram:

$$p(\boldsymbol{x}|\boldsymbol{\theta},\boldsymbol{\nu}) \approx \prod_{i\in\text{bins}} \text{Pois}(n_i|\eta_i(\boldsymbol{\theta},\boldsymbol{\nu})), i = 1,\dots,B, \tag{5.1}$$

where $\boldsymbol{x} = (n_1,\dots,n_B)$ are the observed bin counts and $\eta_i(\boldsymbol{\theta},\boldsymbol{\nu})$ the number of expected events in bin i estimated from simulation, which depends on $\boldsymbol{\theta}$ and $\boldsymbol{\nu}$. The expected number of events is often expressed as a function of the parameters, $\eta_i(\boldsymbol{\theta},\boldsymbol{\nu}) = s_i(\boldsymbol{\theta})\epsilon_{s,i}(\boldsymbol{\nu}) + b_i\epsilon_{b,i}(\boldsymbol{\nu})$, where $s_i(\boldsymbol{\theta})$ and b_i are the expected number of signal and background events, and $\epsilon_{s,i}(\boldsymbol{\nu})$, $\epsilon_{b,i}(\boldsymbol{\nu})$ the respective efficiencies.

In addition, some auxiliary measurements ($\boldsymbol{a}$) may be used to constrain the nuisance parameters through additional prior estimates, $p_\text{c}(\boldsymbol{a}|\boldsymbol{\nu})$ [110]. If there are multiple nuisance parameters, the sources of uncertainty are typically assumed to be independent, and thus their contributions to the model are assumed to be the product of the individual nuisance parameters. All in all, the likelihood can be approximated as

$$p(\boldsymbol{x},\boldsymbol{a}|\boldsymbol{\theta},\boldsymbol{\nu}) \approx \prod_{i\in\text{bins}} \text{Pois}(n_i|\eta_i(\boldsymbol{\theta},\boldsymbol{\nu})) \cdot \prod_{\nu_j\in\boldsymbol{\nu}} p_\text{c}(\boldsymbol{a}|\nu_j), \tag{5.2}$$

where

$$\eta_i(\boldsymbol{\theta},\boldsymbol{\nu}) = s_i(\boldsymbol{\theta})\epsilon_{s,i}(\boldsymbol{\nu}) + b_i\epsilon_{b,i}(\boldsymbol{\nu}). \tag{5.3}$$

However, even if the sources of uncertainty are independent such that the product of $p_\text{c}(\boldsymbol{a}|\nu_j)$ holds, that does not guarantee that the experimental response can be factorized in terms of the individual nuisance parameters. For example, the response of the efficiencies to simultaneous changes in the nuisance parameters will likely not factorize. In addition, this assumption does not capture the lack of pre-fit correlations in the sources of uncertainty given the chosen auxiliary measurements or any resulting post-fit correlations among the nuisance parameters induced by the efficiencies.

In the following sections, we review the procedure for GP regression with and without derivative observations. We also introduce the BED strategy, which uses the GPs as surrogate models and can efficiently sample the space of the experimental response.

5.3 Gaussian Process Regression

A GP [107] is a collection of random variables such that the joint distribution of any finite choice of variables forms a multivariate Gaussian. To illustrate this, consider a set $\mathcal{D}$ of N observation pairs, $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}$, $i = 1, \dots, N$, where $\boldsymbol{x}_i \in \mathbb{R}^D$ denotes an input vector and $y_i \in \mathbb{R}$ denotes a scalar output. To simplify the notation, the input vectors are aggregated in matrix X of size $N \times D$, and the targets are collected in the vector $\boldsymbol{y}$ of length N, so we can write $\mathcal{D} = (X, \boldsymbol{y})$. In the GP framework, the output values are seen as being drawn from a Gaussian distribution with mean function $\mu(\boldsymbol{x})$ and covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$:

$$f(\boldsymbol{x}) \sim \mathcal{N}(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')), \tag{5.4}$$

where

$$\mu(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})] \tag{5.5}$$

$$k(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}[(f(\boldsymbol{x}) - \mu(\boldsymbol{x}))(f(\boldsymbol{x}') - \mu(\boldsymbol{x}'))]. \tag{5.6}$$

For simplicity, we let the mean function be zero ($\mu(\boldsymbol{x}) = 0$), assuming no prior information contradicts this hypothesis. We also let the covariance function take the form of the *squared*

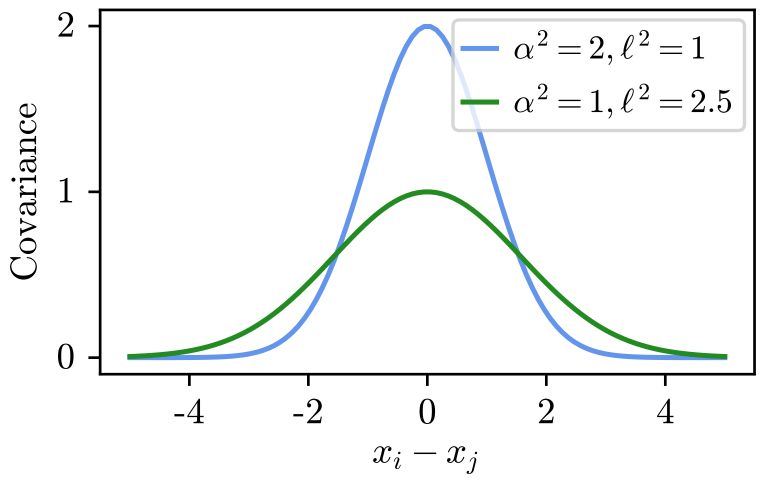


Figure 5.1: Covariance function as the distance between scalar inputs x_i and x_j is varied in the range [-5, 5]. The α^2 and ℓ^2 parameters represent the signal variance and length scale of the squared exponential in the covariance. See Equation 5.7.

exponential (SE) function, also known as the radial basis function:

$$\operatorname{cov}\left[y_i, y_j\right] = k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \alpha^2 \exp\left(-\frac{\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2}{2\boldsymbol{\ell}^2}\right). \tag{5.7}$$

The parameters $\alpha \in \mathbb{R}$ and $\boldsymbol{\ell} \in \mathbb{R}^D$ control the signal variance and length scale of the squared exponential, respectively. When the input values are the same, $\boldsymbol{x}_i = \boldsymbol{x}_j$, the covariance is equal to α^2. As the input values grow farther apart, the covariance approximates zero at a rate dictated by $\boldsymbol{\ell}^2$; smaller $\boldsymbol{\ell}^2$ values result in the covariance approaching zero at faster rates. Fig. 5.1 illustrates the covariance as a function of the distance between two scalar inputs for two sets of α^2 and ℓ^2 parameters.

Let $\boldsymbol{\omega} = \{\alpha, \ell\}$ be the set of free parameters of the model, which are commonly referred to as the *hyperparameters*. In principle, one could choose any combination of hyperparameters to describe the model, but in practice, their values are often chosen by maximizing the log marginal likelihood (lml), or equivalently, by minimizing its negative form:

$$\mathcal{L}(\boldsymbol{\omega}) = -\log p(\boldsymbol{y} \mid X) = -\frac{1}{2}\boldsymbol{y}^T K^{-1}\boldsymbol{y} - \frac{1}{2}\log \mid K \mid -\frac{N}{2}\log(2\pi), \tag{5.8}$$

where K is the $N \times N$ matrix of covariances evaluated at every input of the training data X, such that $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

A common approach to minimizing the negative lml is to perform gradient optimization, which requires the computation of the gradients with respect to each hyperparameter:

$$\frac{\partial \mathcal{L}(\boldsymbol{\omega})}{\partial \omega} = \frac{1}{2}\mathrm{Tr}\left[K^{-1}\frac{\partial K}{\partial \omega}\right] - \frac{1}{2}\boldsymbol{y}^T K^{-1}\frac{\partial K}{\partial \omega}K^{-1}\boldsymbol{y}. \tag{5.9}$$

Once the hyperparameters have been selected, we can define the joint probability distribution of the data as

$$p(y_1, \ldots, y_N | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) = \mathcal{N}(0, K). \tag{5.10}$$

But in general, we are not interested in sampling functions from the prior, but in incorporating the information that the training observations provide to make predictions about the function. Say we want to predict the outcome y_* at a new input location $\boldsymbol{x}_*$. Recalling that under the GP framework, the joint distribution of any finite set of variables must be Gaussian, the joint distribution between training outputs $\boldsymbol{y}$ and the new output y_* must follow

$$\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K & \boldsymbol{k}(X, \boldsymbol{x}_*) \\ \boldsymbol{k}(\boldsymbol{x}_*, X) & k(\boldsymbol{x}_*, \boldsymbol{x}_*), \end{bmatrix}\right) \tag{5.11}$$

where $\boldsymbol{k}(X, \boldsymbol{x}_*)$ is the $N \times 1$ vector of covariances between X and $\boldsymbol{x}_*$, and similarly for $\boldsymbol{k}(\boldsymbol{x}_*, X) = \boldsymbol{k}(X, \boldsymbol{x}_*)^T$, and $k(\boldsymbol{x}_*, \boldsymbol{x}_*)$.

Predictions are achieved by conditioning the joint probability distribution on the observed data $\mathcal{D} = (X, \boldsymbol{y})$ and $\boldsymbol{x}_*$; $p(y_* | \boldsymbol{x}_*, X, \boldsymbol{y})$. This conditional distribution is a Gaussian with mean μ_* and covariance Σ_* given by [111]

$$\mu_* = \boldsymbol{k}\left(\boldsymbol{x}_*, X\right) K^{-1}\boldsymbol{y}, \tag{5.12}$$

$$\Sigma_* = k\left(\boldsymbol{x}_*, \boldsymbol{x}_*\right) - \boldsymbol{k}\left(\boldsymbol{x}_*, X\right) K^{-1}\boldsymbol{k}\left(X, \boldsymbol{x}_*\right). \tag{5.13}$$

So far, we have only considered noise-free observations. But more realistic applications assume that we only have access to noisy function observations, such that $y = f(\boldsymbol{x} + \epsilon)$. Assuming the noise ϵ is an additive independent identically distributed Gaussian with variance σ^2, the covariance between noisy observations becomes

$$\operatorname{cov}\left[y_i, y_j\right] = k(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sigma^2\delta_{ij}. \tag{5.14}$$

The magnitude of σ^2 is treated as a hyperparameter of the model. The δ_{ij} term follows from the independence assumption about the noise, which, once we consider the full set of training observations, is equivalent to adding the noise term to the diagonal of the matrix of covariances. Thus, we can write the joint distribution between the training outputs $\boldsymbol{y}$ and a new output y_* as

$$\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K + \sigma^2 I & \boldsymbol{k}(X, \boldsymbol{x}_*) \\ \boldsymbol{k}(\boldsymbol{x}_*, X) & k(\boldsymbol{x}_*, \boldsymbol{x}_*) \end{bmatrix}\right). \tag{5.15}$$

The mean and covariance of the predictive distribution for noisy outputs then becomes

$$\mu_* = \boldsymbol{k}\left(\boldsymbol{x}_*, X\right) \left[K + \sigma^2 I\right]^{-1} \boldsymbol{y}, \tag{5.16}$$

$$\Sigma_* = k\left(\boldsymbol{x}_*, \boldsymbol{x}_*\right) - \boldsymbol{k}\left(\boldsymbol{x}_*, X\right) \left[K + \sigma^2 I\right]^{-1} \boldsymbol{k}\left(X, \boldsymbol{x}_*\right). \tag{5.17}$$

5.3.1 Including Derivative Information

If derivative observations are available, either by direct calculation or linearization, they can be incorporated into the GP framework. Suppose we are given new sets of input/output observation pairs $\mathcal{D}'_d = \{(\boldsymbol{x}_{d,i}, w_{d,i})\}$, $d = 1, \ldots, D$, $i = 1, \ldots, N_d$ corresponding to the N_d partial derivatives of $f(\boldsymbol{x})$ along dimension d:

$$w_{d,i} = \frac{\partial f(\boldsymbol{x}_{d,i})}{\partial x_d}. \tag{5.18}$$

The covariances between function and partial derivative observations must satisfy [112]

$$\operatorname{cov}\left[w_{d,i}, y_j\right] = \frac{\partial}{\partial x_d}\operatorname{cov}\left[y_i, y_j\right] = -\alpha^2 \frac{(x_{d,i} - x_{d,j})}{\ell_d^2} \exp\left(-\frac{\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2}{2\boldsymbol{\ell}^2}\right), \tag{5.19}$$

$$\operatorname{cov}\left[w_{d,i}, w_{e,j}\right] = \frac{\partial^2}{\partial x_d \partial x_e}\operatorname{cov}\left[y_i, y_j\right] = \alpha^2 \frac{(\delta_{d,e} - (x_{d,i} - x_{e,j})/\ell_e^2)}{\ell_d^2} \exp\left(-\frac{\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2}{2\boldsymbol{\ell}^2}\right). \tag{5.20}$$

Fig. 5.2 illustrates the covariances as a function of the distance between two scalar inputs with hyperparameters $\alpha^2 = 1$ and ℓ^2=1.

To make predictions, we must aggregate the function and derivative observations accordingly. For the training covariance matrix, this means aggregating the function (Equation 5.7), cross (Equation 5.19), and derivative (Equation 5.20) covariances into a single mixed block

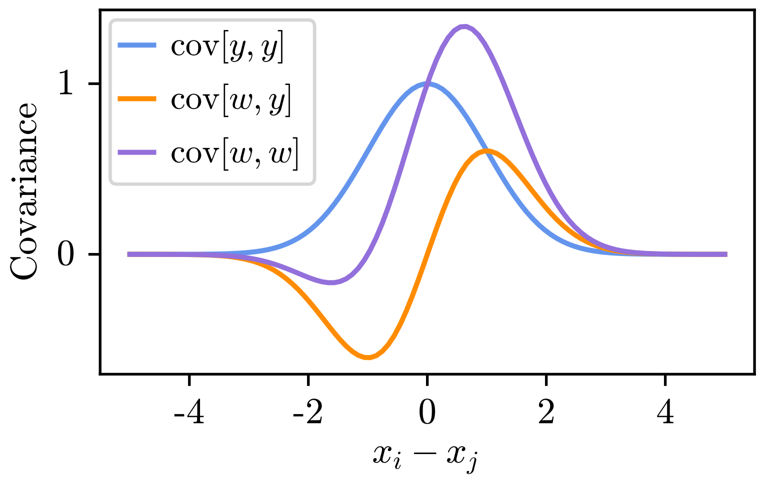


Figure 5.2: Covariance functions as the distance between scalar inputs x_i and x_j is varied in the range [-5, 5], with hyperparameters $\alpha^2 = 1$ and $\ell^2 = 1$. See Equations 5.7, 5.19, and 5.20.

covariance:

$$K_{\mathrm{mxd}} = \begin{bmatrix} K_{yy} & \cdots & K_{y,w_d} & \cdots & K_{y,w_D} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ K_{w_d,y} & \cdots & K_{w_d,w_d} & \cdots & K_{w_d,w_D} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ K_{w_D,y} & \cdots & K_{w_D,w_d} & \cdots & K_{w_D,w_D} \end{bmatrix}, \tag{5.21}$$

where K_{yy} is the $N \times N$ matrix of covariances between function observations, K_{y,w_d} is the $N \times N_d$ matrix of cross covariances between function and derivative observations along dimension d, and similarly K_{w_e,w_d} the $N_e \times N_d$ matrix of covariances derivative observations along dimensions e and d.

The outputs must also be aggregated, creating the mixed output vector

$$\boldsymbol{y}_{\mathrm{mxd}} = [y_1, \ldots, y_N, w_{1,1}, \ldots, w_{1,N_1}, \ldots, w_{d,1}, \ldots, w_{1,N_d}, \ldots, w_{D,1}, \ldots, w_{D,N_D}]^{\mathrm{T}}. \tag{5.22}$$

Similarly to the regular GP regression, predictions are achieved by conditioning the joint probability on the observed data. The predictive mean and covariance correspond to the forms as Eqs. 5.16 and 5.17, but the output vector and covariances are updated to represent the mixed nature of the data:

$$\mu_* = \boldsymbol{k}\left(\boldsymbol{x}_*, X\right)_{\text{mxd}} \left[K_{\text{mxd}} + \boldsymbol{\sigma}^2_{\text{mxd}}\boldsymbol{I}\right]^{-1} \boldsymbol{y}_{\text{mxd}}, \tag{5.23}$$

$$\Sigma_* = k\left(\boldsymbol{x}_*, \boldsymbol{x}_*\right) - \boldsymbol{k}\left(\boldsymbol{x}_*, X\right)_{\text{mxd}} \left[K_{\text{mxd}} + \boldsymbol{\sigma}^2_{\text{mxd}}\boldsymbol{I}\right]^{-1} \boldsymbol{k}\left(X, \boldsymbol{x}_*\right)_{\text{mxd}}. \tag{5.24}$$

Like in Equations 5.16 and 5.17, white noise is added to the training covariance to account for statistical fluctuations in the data or the model. In practice, these fluctuations may vary between function and derivative observations, and thus the $\boldsymbol{\sigma}^2_{\text{mxd}}$ term is a vector containing the variances added to the diagonal elements of the function (K_{yy}) and derivative (K_{w_d,w_d}) block covariances in K_{mxd}.

We refer to a GP model that incorporates derivative observations as a *derivative* GP. By contrast, we refer to a GP model without derivative observations as a *regular* GP.

5.3.2 Efficient Gaussian Process Regression with Approximated Gradients

The physics examples in this paper focus on estimating the efficiency (ϵ) as a function of the nuisance parameters ($\boldsymbol{\nu}$). For a dataset with many events, evaluating the efficiency is generally a computationally expensive task involving evaluating each event over a predetermined selection criteria.

The selection criteria are often based on statistical cuts, which are non-differentiable. To obtain the derivative observations of the efficiency, we replace the cuts with sigmoid functions (S), which are easily differentiable, allowing us to calculate an approximation of the gradients of the efficiency with respect to the nuisance parameters. We modify the sigmoid functions with the parameters a and c, which control the steepness and horizontal shift of the sigmoid curve, respectively:

$$S(x, a, c) = \frac{1}{1 + e^{-a(x-c)}} \tag{5.25}$$

The sigmoids are shifted to match threshold values of the statistical cuts by adjusting c, and the sign and magnitude of the steepness parameter a are tuned depending on the specifics of the selection criteria. Absolute values of a larger than unity result in steeper sigmoids, while absolute values smaller than unity result in less steep sigmoids.

Other methods may be used to calculate automated numerical derivatives of detector observables [113, 114], which can then be used to enhance the GP regression. In our studies, we find that replacing the statistical cuts with sigmoid functions results in more accurate GP regressions.

We note how derivative GPs have a higher computational cost than regular GPs ($\mathcal{O}(N^3D^3)$ vs. $\mathcal{O}(N^3)$) when conditioned on N samples in D dimensions. However, we expect the training and inference time of the GPs to be relatively negligible compared to the time it may take to directly sample the efficiency or other observables of the detector response. In addition, recent work has been done to scale derivative GPs in low- [115] and high-dimensions [116, 117].

5.4 Bayesian Experimental Design

We introduce a BED sampling strategy designed to efficiently select new training observations while reducing the predictive uncertainty of a GP model. At the core of the BED strategy is the utility function U. With every BED iteration, the utility function is maximized to find the next sampling point.

5.4.1 Utility

Suppose we have observed N input/output pairs $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}$, $i = 1, \ldots, N$, which have been used to train a GP model. We then wish to make a new observation, but due to high sampling costs, we wish to make an informative decision and observe the $(\boldsymbol{x}_{N+1}, y_{N+1})$ that is most likely to reduce the predictive uncertainty of the model.

Intuitively, the computation of the utility is as follows: we define a test *utility input* X_u that is believed to be representative of the input space, and it could be, for example, a set of grid points spanning the input domain. Then, we compute the posterior covariance Σ_u of the model evaluated at each point in X_u. For some arbitrary new input $\boldsymbol{x}$, we temporarily augment the model to include additional training data at $\boldsymbol{x}$, $\mathcal{D}^{\text{aug}} \leftarrow \mathcal{D} \cup \{(\boldsymbol{x}, \mu)\}$, where μ is the predictive mean calculated according to Equation 5.16. The augmented model is used to reevaluate the posterior covariance Σ_u^{aug} at each point in X_u. By comparing Σ_u and Σ_u^{aug}, we can estimate the relative loss in the predictive uncertainty of the model.

Many quantities could be used to characterize the predictive uncertainty, such as:

- **Determinant**: Considers the determinant of the posterior covariance matrix, which is correlated to the error ellipsoid of the model.
- **Trace**: Considers the trace of the posterior covariance matrix, which quantifies the net variance of the model.
- **Maximum variance**: Considers the maximum entry in the diagonal of the predictive covariance matrix, which quantifies the maximum variance of the model.

In our studies, we find that using the trace is a good and cost-efficient way to reduce the predictive variance of the model, thus reducing the overall uncertainty. In addition, we find the trace more numerically stable than the determinant, which can tend to zero when the input variables are strongly correlated.

Ideally, the trace of Σ_u^{aug} should be smaller than the trace of Σ_u. With this in mind, we define the utility as

$$\mathrm{U}(\boldsymbol{x}, \mathcal{D}, X_u) \equiv 1 - \mathrm{Tr}(\Sigma_u^{\text{aug}}) / \mathrm{Tr}(\Sigma_u). \tag{5.26}$$

The pseudocode of the utility function is shown in Algorithm 1. We note that we could alternatively define the utility as simply the negative of Σ_u^{aug} with the same effect, but we prefer the form above as it provides a better sense of the relative loss in the variance of the model.

Algorithm 1 Utility function (U)

```
Input: $(\boldsymbol{x}, \mathcal{D}, X_u)$
  Evaluate $\Sigma_u$ at every point in $X_u$ according to Equation 5.17.
  Evaluate $\mu$ at $\boldsymbol{x}$ according to Equation 5.16.
  Temporarily augment the training dataset $\mathcal{D}$:
    $\mathcal{D}^{\text{aug}} \leftarrow \mathcal{D} \cup \{(\boldsymbol{x}, \mu)\}$.
  Evaluate $\Sigma_u^{\text{aug}}$ at every point in $X_u$ according to Equation 5.17.
  return $1 - \mathrm{Tr}(\Sigma_u^{\text{aug}}) / \mathrm{Tr}(\Sigma_u)$.
```

5.4.2 Choice of Utility Input

A key choice in our BED strategy is the choice of utility input, X_u. Ideally, evaluating the posterior covariance at the utility input should provide a sense of the uncertainty of the model's predictions over the input space, and thus high-resolution uniform grids are good candidates for X_u. However, this choice has two disadvantages: first, evaluating the model at each grid point in X_u for every BED iteration can become expensive, particularly for models of high dimensionality. Second, the BED could overtrain on X_u by choosing samples that are most likely to reduce the uncertainty near the chosen grid instead of exploring the full input space.

To offset the issues mentioned above, we use uniform grids for X_u, but with added regularization. After every BED iteration, we perturb X_u by adding Gaussian noise with standard deviation of magnitude γ^2. Choosing an adequate γ^2 parameter may prevent the BED from overtraining on X_u. In addition, this strategy may result in cheaper BED evaluations since it allows us to choose grids with lower resolution while still making the BED strategy exploratory enough to span the input space over multiple iterations.

5.4.3 Updating the Beliefs About the Model

Once the utility function has been maximized, the location of the new observation is selected:

$$\boldsymbol{x}_{N+1} = \underset{\boldsymbol{x}\in\mathcal{X}}{\arg\max}\, \mathrm{U}(\boldsymbol{x}, \mathcal{D}, X_u). \tag{5.27}$$

We then sample y_{N+1} and recondition the GP model with the updated set of training observations, $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{x}_{N+1}, y_{N+1})\}$. This process is repeated until a stopping criterion is reached.

5.5 Simple 1D Toy Model

In this section, we test the GP regression and BED strategy on a simple 1D toy model that is described by the following sinusoidal dynamical system with scalar input x and output y:

$$y = x\cos(x), \tag{5.28}$$

$$\frac{dy}{dx} = \cos(x) - x\sin(x), \tag{5.29}$$

The function (y) and derivative ($\frac{dy}{dx}$) observations are shown in Fig. 5.3.

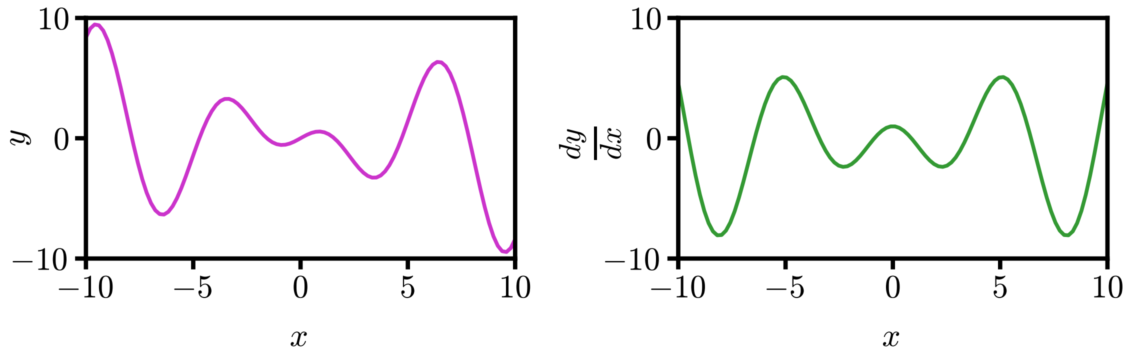


Figure 5.3: Function output (y) and and derivative ($\frac{dy}{dx}$) of the simple 1D toy model for $x \in [-10, 10]$.

5.5.1 Gaussian Process Regression

We sample the system at four random points within $x \in [-10, 10]$. The four observations are used to train regular and derivative noise-free GP models with hyperparameters $\alpha^2 = 1$ and $\ell^2 = 1$. The models are tested on 100 points uniformly distributed within the same range. The mean prediction (μ_{test}) and uncertainty bands are shown in Fig. 5.4.

The regular and derivative GP models reveal different predictions even when trained on the same observations. With the help of the gradients, the derivative GP is able to predict the

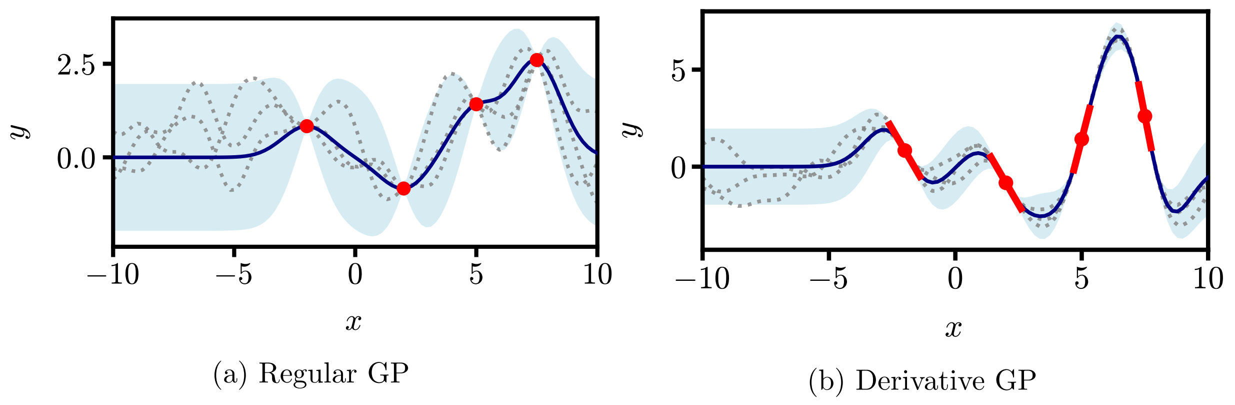


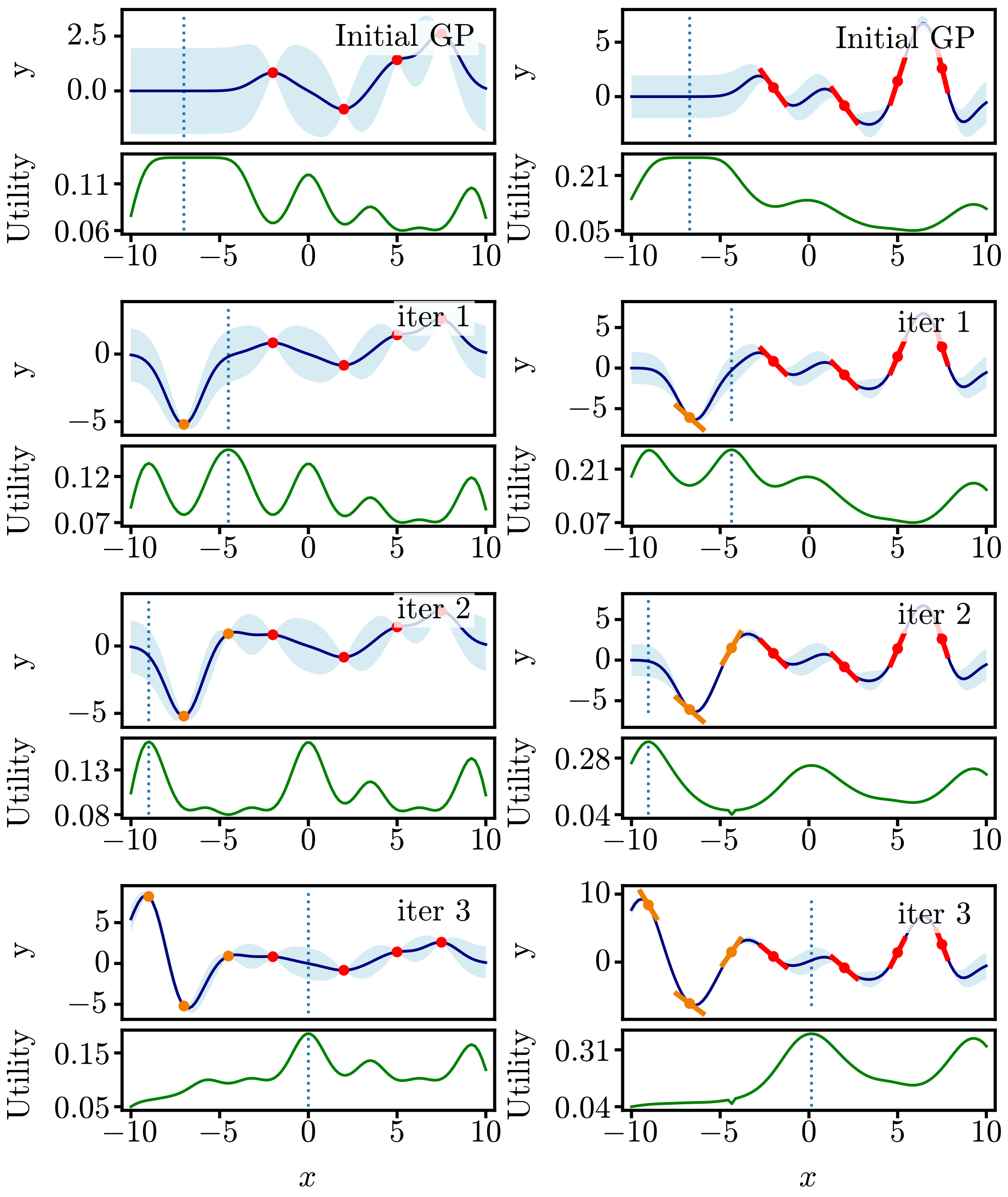
Figure 5.4: Panel (a) shows the four observations (red dots) used to train the regular GP model, μ_{test} (blue line), and confidence band (light blue area), together with three functions sampled from the posterior (gray dotted lines). Similarly, panel (b) shows the predictions of the derivative GP model using the same four observations but augmented by their gradients (red tangent segments).

turning points near the training observations, while the regular GP fails to capture most turning points, as well as the inflection point at $x = 0$.

5.5.2 Bayesian Experimental Design

We use the BED strategy to sample 20 additional observations. For the choice of utility input, we select the same input as the test set, 100 points uniformly distributed between $x \in [-10, 10]$. A visualization of the evolution of the GP models for the first three BED iterations is shown in Fig. 5.5. With every iteration, the utility is maximized and a new training observation is selected. As we condition the models to include the latest observation, their predictions resemble more and more the function shown in Fig. 5.3 while the uncertainty band decreases in area.

In total, 20 sequential BED iterations are performed. After every iteration, we update the predictive mean (μ_{test}) and the predictive covariance (Σ_{test}). The predictive mean is used to calculate the mean-squared-error (MSE) of the models, and the trace of the covariance ($\text{Tr}(\Sigma_{\text{test}})$) is used as a measure of the uncertainty of the predictions. Lower $\text{Tr}(\Sigma_{\text{test}})$ values



(a) Regular GP (b) Derivative GP

Figure 5.5: First three BED iterations of the regular (a) derivative (b) GP models. The panels show the initial observations (red dots), the observations selected by the BED (orange dots), μ_{test} (blue line), and the uncertainty band (light blue area). Also shown are the utility function (green line) and the location of the next observation (dotted vertical line), which is selected by maximizing the utility.

suggest lower predictive uncertainty. To test the sensitivity of the BED to initial random conditions, we run the BED ten times using different initial random seeds. The results are shown in Fig. 5.6.

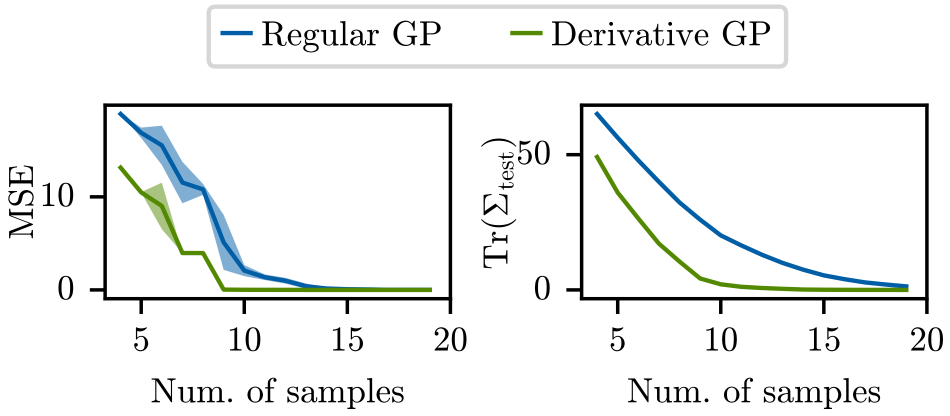


Figure 5.6: MSE and Tr(Σ_{test}) after each of the 20 BED iterations. The models are tested on 100 points uniformly distributed between $x \in [-10, 10]$. The solid lines represent the mean values of 10 runs of the BED strategy with different initial random seeds and the shaded areas represent the standard deviation.

The derivative GP has the edge over the regular GP, resulting in lower MSE and Tr(Σ_{test}) values, corroborating the power of including derivative information. The BED strategy does a good job at decreasing the overall uncertainty of the models, as seen by the monotonic decrease in Tr(Σ_{test}).

5.6 High-Energy Physics: 2D Efficiency Estimation

In this section, we apply the GP regression and BED strategy to experiments in high-energy physics. We focus on estimating the efficiency (ϵ) as a function of two jet energy scales. Efficiency often plays an important role in estimating systematic uncertainties (Equation 5.2) and having a strategy to accurately estimate it based on limited samples can be a powerful tool in high-energy physics experiments.

The dataset consists of 30K events with three jets each. The jets are arranged in descending p_{T} order, with j_1 corresponding to the hardest jet in an event, followed by j_2 and j_3. The p_{T} distribution of the two leading jets is shown in Fig. 5.7.

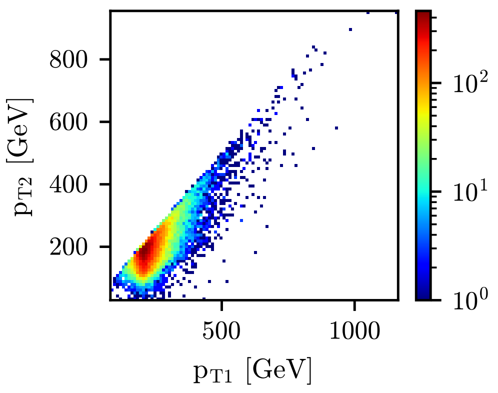


Figure 5.7: Distribution of the transverse momentum of j_1 (p_{T_1}) and j_2 (p_{T_2}).

We estimate the efficiency as a function of the following nuisance parameters:

- ν_1: The jet energy scale of the leading jet, j_1.
- $\nu_{2,3}$: The jet energy scale of the two softer jets, j_2 and j_3.

And the following p_{T}-based selection criteria:

$$\begin{aligned} &\frac{p_{\mathrm{T}1}}{\nu_1} > 200 \text{ GeV}, \\ &\frac{p_{\mathrm{T}2}}{\nu_{2,3}} < 200 \text{ GeV}, \end{aligned} \tag{5.30}$$

where p_{T_1} and p_{T_2} are the nominal transverse momentum of j_1 and j_2, respectively. Once the events have been selected according to the selection criteria, we calculate the efficiency (ϵ) of events with missing transverse energy ($\mathrm{E}_{\mathrm{T}}^{\mathrm{miss}}$) of less than 50 GeV. The calculations of

$\mathrm{E_T^{miss}}$ and ϵ can be summarized as

$$\mathrm{E_T^{miss}}(\nu_1, \nu_{2,3}) = \sqrt{\left(\sum_{i=1,2,3} \frac{p_{\mathrm{T}ix}}{\nu_i}\right)^2 + \left(\sum_{i=1,2,3} \frac{p_{\mathrm{T}iy}}{\nu_i}\right)^2}, \tag{5.31}$$

$$\epsilon(\nu_1, \nu_{2,3}) = \frac{\mathrm{N}_{\mathrm{E_T^{miss}}<50\mathrm{GeV}}}{\mathrm{N_{pass}}(\nu_1, \nu_{2,3})}, \tag{5.32}$$

where $\nu_{2,3} = \nu_2 = \nu_3$, $\mathrm{N_{pass}}(\nu_1, \nu_{2,3})$ is the number of events that satisfy the selection criteria in Equation 5.30, and $\mathrm{N}_{\mathrm{E_T^{miss}}<50\mathrm{GeV}}$ is the number of events with $\mathrm{E_T^{miss}}(\nu_1, \nu_{2,3}) < 50$ GeV. For reference, Fig. 5.8 shows the efficiency for $\nu_1, \nu_{2,3} \in [0.5, 1.5]$ in 25×25 intervals of size 0.4×0.4. The efficiency has been normalized by the central value $\epsilon(1, 1)$. The values shown in Fig. 5.8 are used as the ground truth to test the models employed throughout this section.

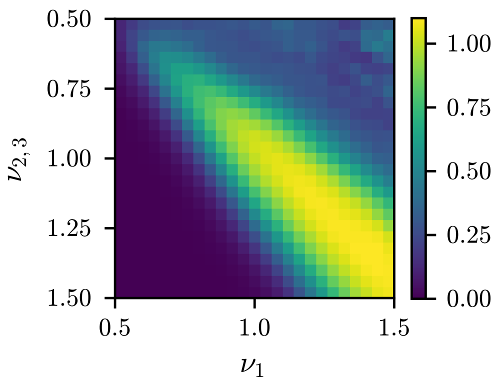


Figure 5.8: Efficiency as a function of the jet energy scales, normalized by $\epsilon(1, 1)$.

Directly evaluating the efficiency can be an expensive task, particularly when it involves datasets that are large in size and dimensionality. In such cases, simple sampling techniques are often employed to approximate the efficiency. These techniques often assume that efficiency can be factorized in terms of the nuisance parameters. As an example of such techniques, we consider the following regression, which we call the *on-axis regression* (OAR):

$$\epsilon(\nu_1, \nu_{2,3})_{\mathrm{OAR}} \cong \epsilon(\nu_1, 1) \times \epsilon(1, \nu_{2,3}). \tag{5.33}$$

The idea behind the on-axis regression is to sample the efficiency along the central axes of the input space, which in this case correspond to $\nu_1, \nu_{2,3} = 1$. These samples are then used to estimate the rest of the efficiency by assuming that the *off-axis* values can be approximated by the product of the corresponding *on-axis* values.

To implement the on-axis regression, we sample 49 on-axis observations (24 along each axis plus the central value), which are used to calculate the off-axis efficiency according to Equation 5.33. The results are shown in Fig. 5.9. The on-axis regression is significantly cheaper than evaluating the efficiency directly at every input location, but the performance is poor as it fails to capture correlations between the off-axis variables. For example, Fig. 5.8 shows that the highest efficiency values are concentrated near the diagonal starting from the top left corner. The on-axis regression fails to predict the high efficiency in the diagonal correctly. Instead, it predicts the center of the input space to be the area with the highest efficiency.

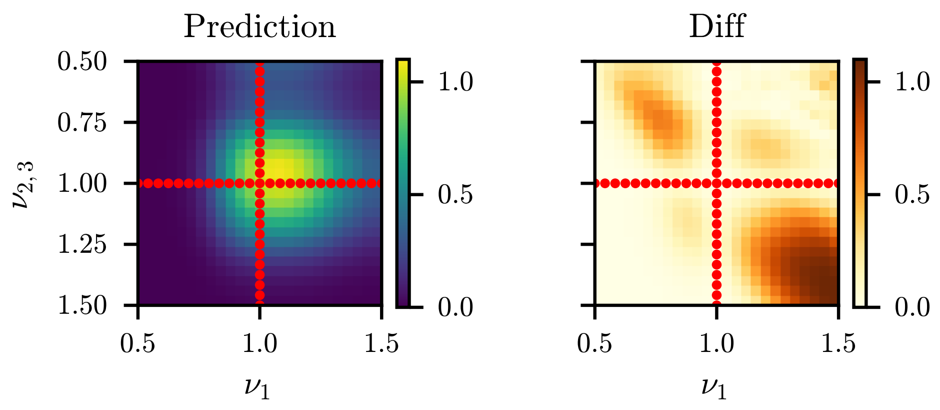


Figure 5.9: Prediction (left) of the normalized efficiency by on-axis regression, and the absolute difference (right) between ground truth and prediction. The red dots represent the samples used in the on-axis regression.

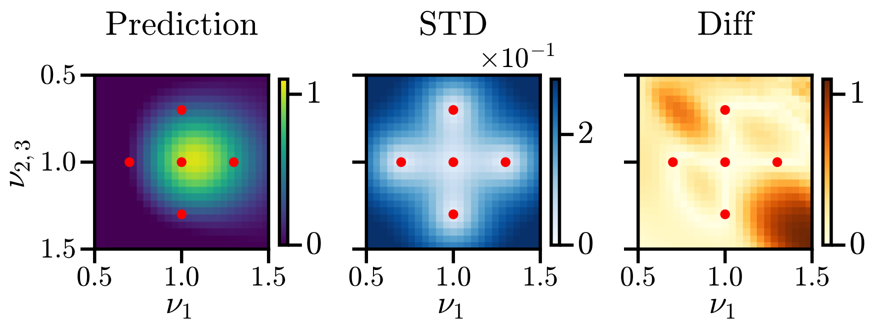
5.6.1 Gaussian Process Regression

As an alternative, we approximate the efficiency by GP regression. Like in the on-axis regression, we begin by sampling the efficiency along the $\nu_1, \nu_{2,3} = 1$ axes. To test the predictive power of the GPs, we only sample five observations (two along each axis plus the central value):

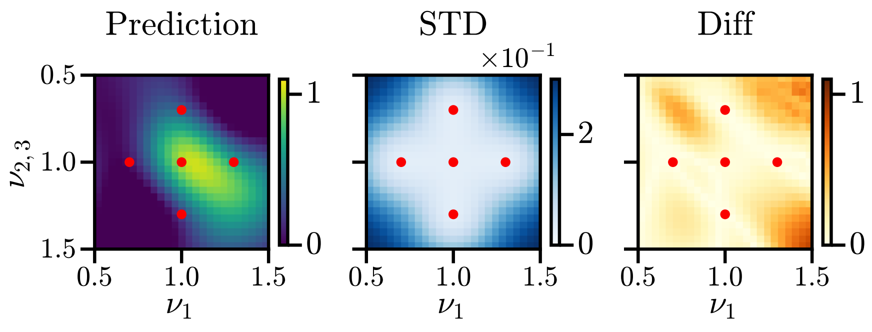
$$(\nu_1, \nu_{2,3}) = (0.7, 1.0), (1.0, 1.0), (1.3, 1.0), (1.0, 0.7), (1.0, 1.3).$$

These five samples are used to condition a regular GP model with hyperparameters $\alpha^2 = 0.1$, $\ell^2 = 0.25^2$, and $\sigma^2 = 1\mathrm{e}^{-3}$. The hyperparameters are selected by grid search. The results are shown in Fig. 5.10a. The prediction is similar to that of the on-axis regression. Both predict higher efficiencies near the center of the grid and have the highest error along the diagonal.

We also condition a derivative GP model using the same five training samples. For consistency, we use the same hyperparameters used in the regular GP model, but we add a white noise with variance $\sigma^2 = 1\mathrm{e}^{-1}$ to the derivative terms. To obtain the gradients of the efficiency with respect to ν_1 and $\nu_{2,3}$, we replaced the inequalities in Equations 5.30 and 5.32 with sigmoid functions, which are shifted to match the 200 GeV threshold ($c = 200$) and scaled by $\mid a \mid= 1/10$ to smooth out the gradients. The sign of a is determined by the sign of the statistical thresholds and its magnitude by a grid search. We find $\mid a \mid= 1/10$ to provide a good trade-off between the smoothness and fidelity of the sigmoids when approximating the cuts. The results are shown in Fig. 5.10b. The derivative GP does surprisingly well compared to the regular GP and the on-axis regression. With only five samples, it captures the general trend along the diagonal.



(a) Regular GP



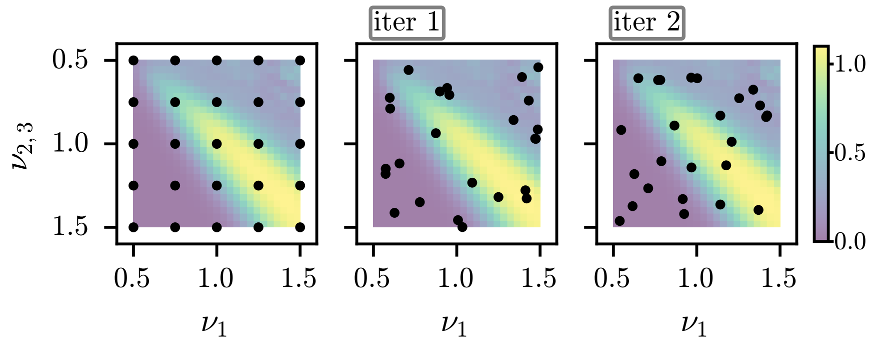
(b) Derivative GP

Figure 5.10: Prediction (left), standard deviation (middle), and the absolute difference (right) between ground truth and prediction by the specified GP model. The red dots represent the samples used to train the models.

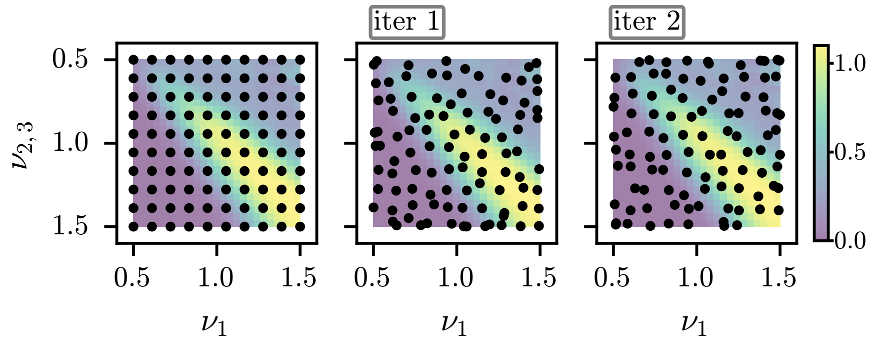
5.6.2 Bayesian Experimental Design

We use the BED strategy to obtain 44 additional samples for a total of 49, the same number sampled in the on-axis regression. We consider the following options for choice of utility input: a 25×25 with $\gamma^2 = 1/125$, a 10×10 grid with $\gamma^2 = 1/50$, and a 5×5 grid with $\gamma^2 = 1/10$. Larger γ^2 values are necessary for the lower-resolution grids to encourage more explorative grid perturbations. As an example, we illustrate the 5×5 and 10×10 grids after two sample BED iterations in Fig. 5.11.

After every iteration, we update the predictive mean (μ_{test}) and covariance (Σ_{test}). The MSE and $\text{Tr}(\Sigma_{\text{test}})$ results of the various choices of utility input are shown in Fig. 5.12. We observe a trade-off between the grid size and the efficacy of the BED strategy. This trade-off,



(a) $X_u : 5 \times 5$ grid with $\gamma^2 = 1/10$

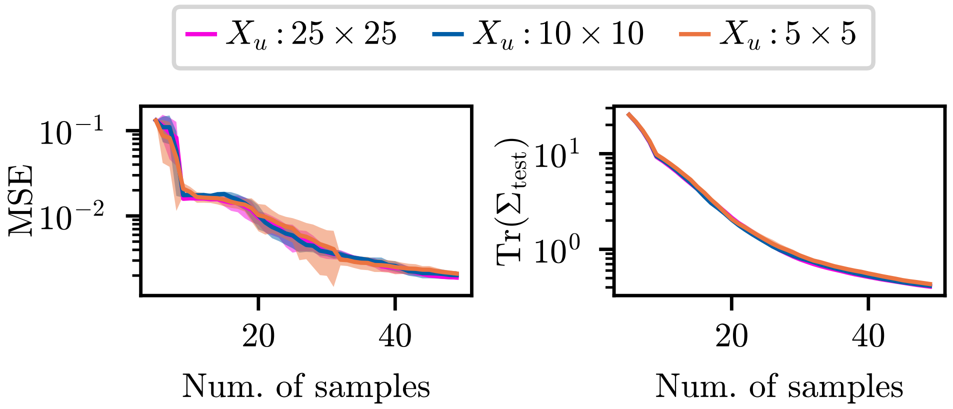


(b) $X_u : 10 \times 10$ grid with $\gamma^2 = 1/50$

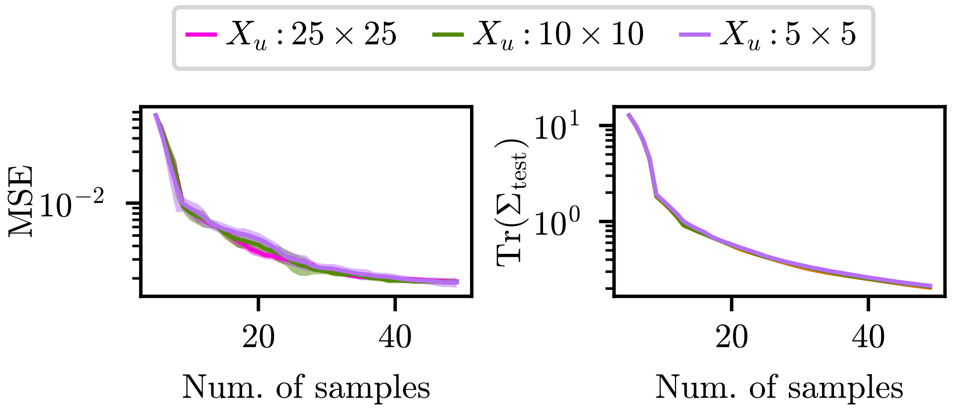
Figure 5.11: The panels illustrate two choices of utility input. The left images show the grids before regularization. The two right-most images show the grids after regularization by adding Gaussian noise $\mathcal{N}(0, \gamma^2)$. The grids are perturbed after every BED iteration. The color scale of the jet efficiency shown in the background is dimmed down for clarity.

however, is only marginal; the 5×5 and 10×10 grids are roughly equivalent to the 25×25 grid and significantly cheaper to evaluate.

We compare the results of the regular and derivative GP models to those of the on-axis regression in Fig. 5.13. For simplicity, we only show the results with the 10×10 grid choice of utility input, but they generalize to the other choices. Like in the toy 1D model, the derivative GP has the edge over the regular GP with lower MSE and $\mathrm{Tr}(\Sigma_{\text{test}})$ values. The error of the on-axis regression is initially comparable to that of the regular GP and almost twice as large as that of the derivative GP. As we add more training observations to the GP models, both the MSE and $\mathrm{Tr}(\Sigma_{\text{test}})$ values of the models quickly diminish. A visualization of the evolution of the GP models for the first three BED iterations is shown in Fig. 5.14.



(a) Regular GP



(b) Derivative GP

Figure 5.12: MSE and $\mathrm{Tr}(\Sigma_{\text{test}})$ of the regular and derivative GP model after each of the 44 BED iterations for the various choices of utility input. The models are tested on a 25×25 grid uniformly distributed between $\nu_1, \nu_{2,3} \in [0.5, 1.5]$. The solid lines represent the mean values of 10 BED runs with different initial random seeds and the shaded areas the standard deviation.

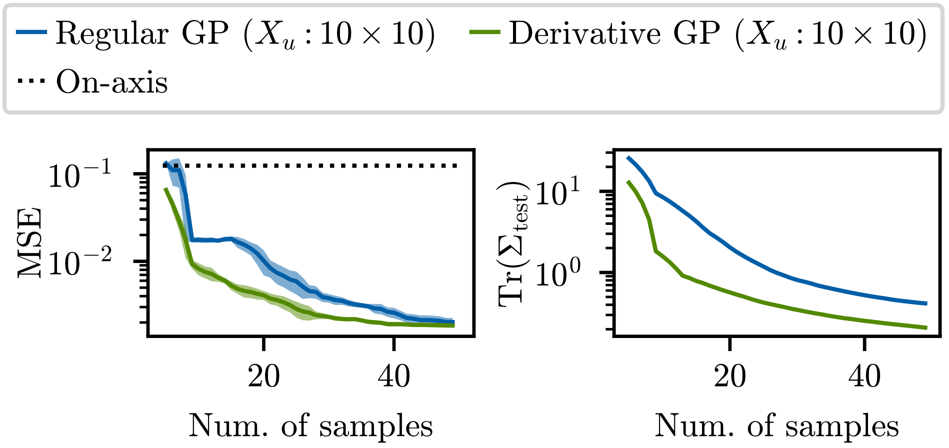
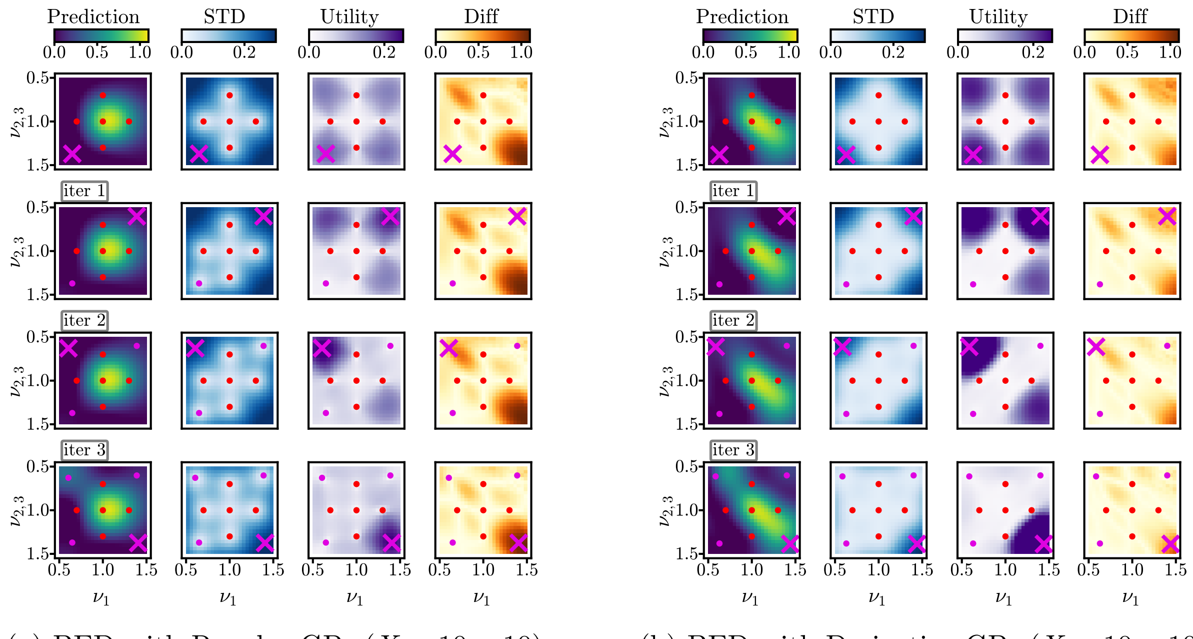


Figure 5.13: Comparison of the MSE and $\mathrm{Tr}(\Sigma_{\mathrm{test}})$ of the various models. The models are tested on a 25×25 grid uniformly distributed between $\nu_1, \nu_{2,3} \in [0.5, 1.5]$. The solid lines represent the mean values of 10 BED runs with different initial random seeds and the shaded areas the standard deviation.



(a) BED with Regular GPs ($X_u : 10 \times 10$) (b) BED with Derivative GPs ($X_u : 10 \times 10$)

Figure 5.14: Prediction (first column), standard deviation (second column), utility (third column), and the absolute difference (fourth column) between ground truth and prediction by the specified GP model. The panels show the initial training samples (red dots), the training samples selected by the BED (pink dots), and the location of the next sample selected by the BED by maximizing the utility (pink cross).

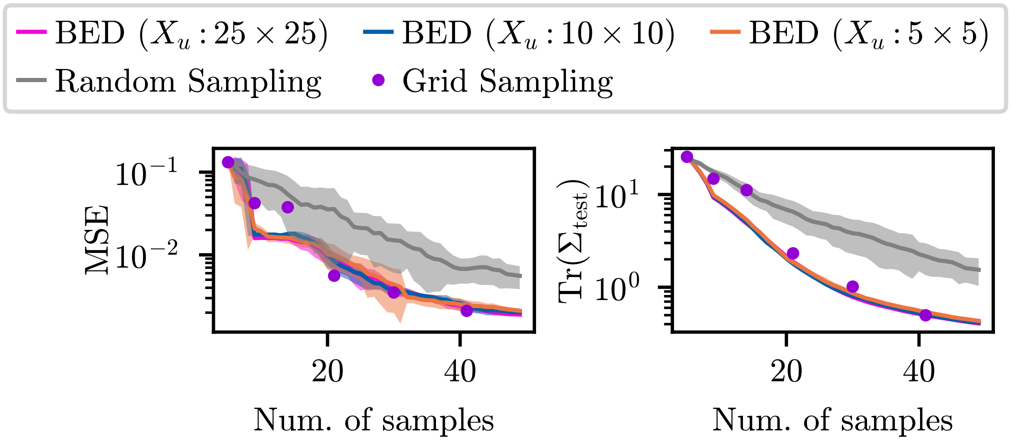
A critical question is whether the BED has successfully selected samples that efficiently reduce $\mathrm{Tr}(\Sigma_{\text{test}})$. To answer this question, we compare the BED to random and grid sampling. In the random sampling case, we sequentially add 44 uniformly distributed random observations to the training set. In the grid sampling case, we sample squared grids of size $n \times n$, $n = 2, \ldots, 6$, which are separately added to the initial five training samples. Unlike the BED and random sampling strategies, the grid sampling strategy is not sequential. See Appendix C for a sample visualization of the random and grid sampling strategies.

Figure 5.15 shows the MSE and $\mathrm{Tr}(\Sigma_{\text{test}})$ of the different sampling strategies. Random sampling performs the worst with the highest overall MSE and $\mathrm{Tr}(\Sigma_{\text{test}})$ values. Grid sampling performs better but falls behind the BED in the first couple of iterations. The BED has the lowest overall $\mathrm{Tr}(\Sigma_{\text{test}})$ values, regardless of the choice of utility input, suggesting that the BED sampling strategy has selected samples that efficiently reduced $\mathrm{Tr}(\Sigma_{\text{test}})$ in both the regular and derivative GP cases.

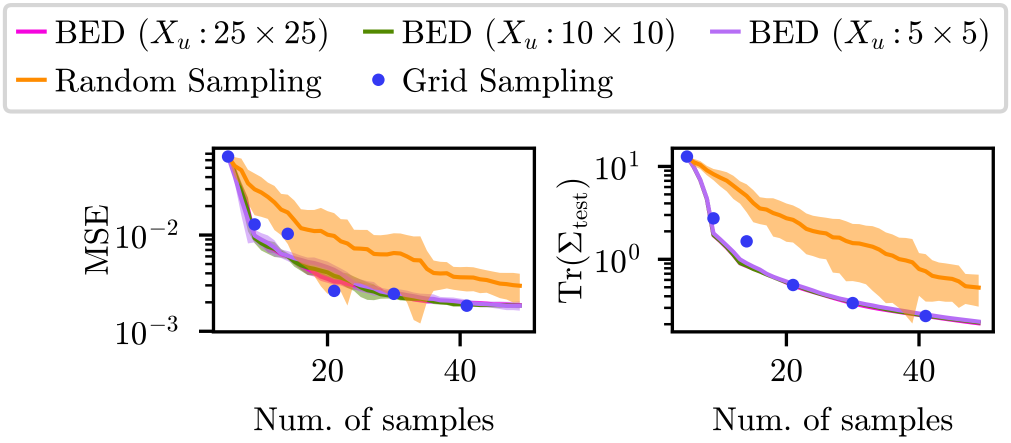
5.7 High-Energy Physics: 4D Efficiency Estimation

Lastly, we increase the dimensionality of the efficiency by considering four jet energy scales, which are dependent on the pseudorapidity of the jets, and roughly correspond to the central and outer calorimeter regions:

- ν_1^{central}: The jet energy scale of j_1 with $\mid \eta_1 \mid < 1$.
- ν_1^{outer}: The jet energy scale of j_1 with $\mid \eta_1 \mid \geq 1$.
- $\nu_{2,3}^{\text{central}}$: The jet energy scale of j_2 and j_3 with $\mid \sum_{i=2,3} p_{\mathrm{T}i}\, \eta_i / \sum_{i=2,3} p_{\mathrm{T}i} \mid < 1$.
- $\nu_{2,3}^{\text{outer}}$: The jet energy scale of j_2 and j_3 with $\mid \sum_{i=2,3} p_{\mathrm{T}i}\, \eta_i / \sum_{i=2,3} p_{\mathrm{T}i} \mid \geq 1$.



(a) Regular GP



(b) Derivative GP

Figure 5.15: MSE and $\mathrm{Tr}(\Sigma_{\mathrm{test}})$ after each iteration of the various sampling strategies. The models are tested on a 25×25 grid uniformly distributed between $\nu_1, \nu_{2,3} \in [0.5, 1.5]$. In the BED and random sampling cases, the solid lines represent the mean values of 10 sampling runs using different initial random seeds and the shaded areas the standard deviation.

Where η_i is the pseudorapidity of j_i. The distribution of the pseudorapidities is shown in Fig. 5.16. For every event, either ν_1^{central} or ν_1^{outer} is selected as the jet energy scale of j_1. Likewise, either $\nu_{2,3}^{\mathrm{central}}$ or $\nu_{2,3}^{\mathrm{outer}}$ is selected as the jet energy scale of j_2 and j_3. The efficiency is then calculated according to the procedure specified in Sec. 5.6 and normalized by the central value $\epsilon(1, 1, 1, 1)$.

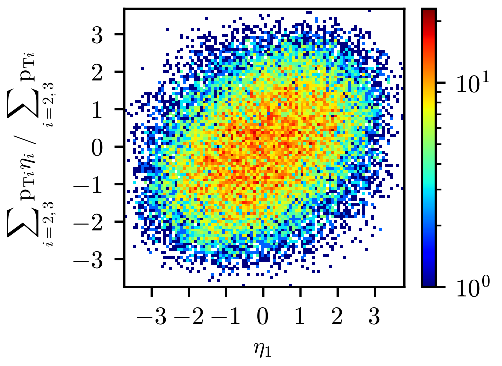


Figure 5.16: Distribution of the pseudorapidity of j_1, and the p_{T}-weighted pseudorapidity mean of j_2 and j_3.

Visualizing the efficiency as a function of the four jet energy scales is challenging due to the high dimensionality of the input space. For illustration purposes, we show only a slice of the efficiency space by plotting the efficiency as we vary ν_1^{central} and $\nu_{2,3}^{\text{central}}$ in the range $[0.5, 1.5]$ while we fix ν_1^{outer} and $\nu_{2,3}^{\text{outer}}$ to have a magnitude of 0.5. The selected slice is shown in Fig. 5.17.

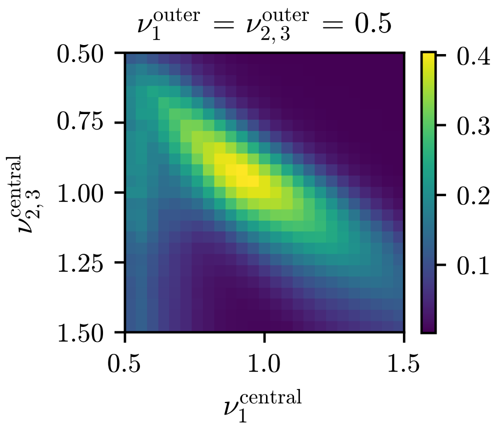


Figure 5.17: Efficiency as ν_1^{central} and $\nu_{2,3}^{\text{central}}$ are varied in the range $[0.5, 1.5]$ while ν_1^{outer}=$\nu_{2,3}^{\text{outer}} = 0.5$. The efficiency is normalized by $\epsilon(1, 1, 1, 1)$.

We first approximate the efficiency by on-axis regression. We sample a total of 97 on-axis observations (24 along each central axis plus the central value) and use them to estimate the off-axis values according to

$$\begin{aligned}\epsilon(\nu_1^{\text{central}}, \nu_1^{\text{outer}}, \nu_{2,3}^{\text{central}}, \nu_{2,3}^{\text{outer}})_{\text{OAR}} \cong & \epsilon(\nu_1^{\text{central}}, 1, 1, 1) \times \epsilon(1, \nu_1^{\text{outer}}, 1, 1) \times \\ & \epsilon(1, 1, \nu_{2,3}^{\text{central}}, 1) \times \epsilon(1, 1, 1, \nu_{2,3}^{\text{outer}}).\end{aligned} \tag{5.34}$$

Fig. 5.18 shows the results of the on-axis regression on the selected slice of the efficiency space. The on-axis regression is cheap but performs poorly as it again fails to predict the higher efficiency values along the diagonal.

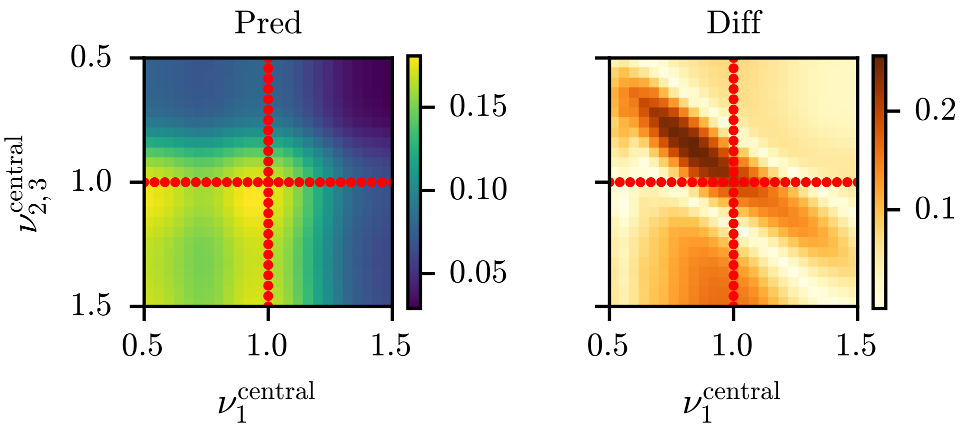
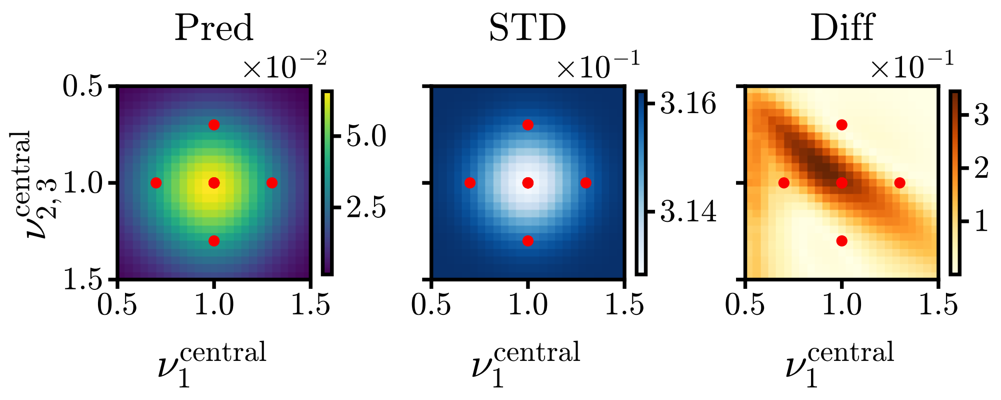


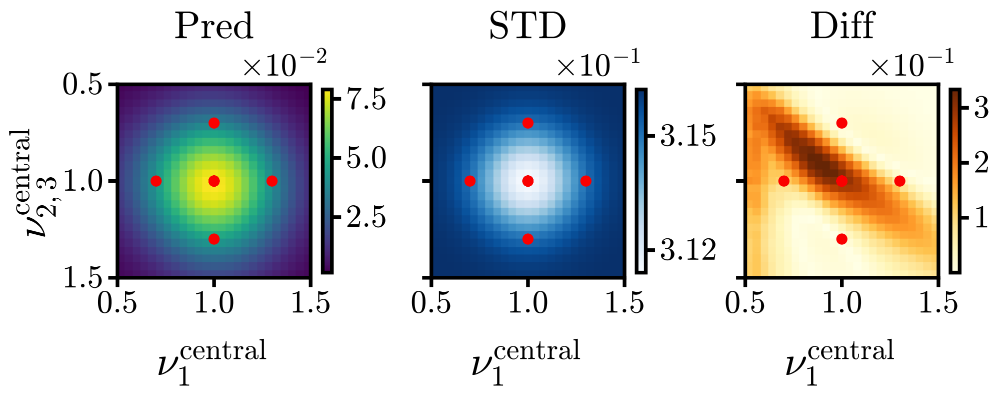
Figure 5.18: Prediction (left) of the selected slice of the efficiency space by on-axis regression and the absolute difference (right) between ground truth and prediction. The red dots represent the samples used in the on-axis regression.

5.7.1 Gaussian Process Regression

As an alternative, we approximate the efficiency by GP regression. We sample nine observations along the central axes (two along each axis plus the central value). Like in the 2D case, the initial observations are on-axis at the input values $0.7, 1, 1.3$. These nine observations are used to condition a regular and a derivative GP model with the same hyperparameters used in the 2D case. To obtain the gradients with respect to all four jet energy scales, we replace the pseudorapidity-based cuts with sigmoid functions, which are shifted to match the $\mid \eta \mid < 1$ threshold ($c = 1$) and scaled by $\mid a \mid = 1/100$. We also scale the sigmoids that replace the p_T-cut in eq. 5.30 by $\mid a \mid = 1/100$. The change the a scaling is necessary to compensate for the additional sigmoids. The variance of the Gaussian noise added to the derivative terms is increased to $\sigma^2 = 10$, which we have selected by grid search. The results are shown in Fig. 5.19.



(a) Regular GP



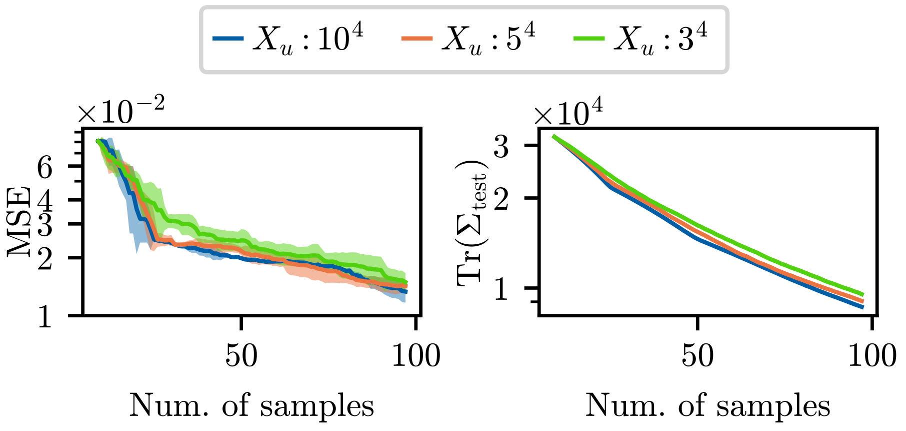
(b) Derivative GP

Figure 5.19: Prediction (left) of the selected slice of the efficiency space, standard deviation (middle), and the absolute difference (right) between ground truth and prediction by the specified GP model. The red dots represent the samples used to train the models.

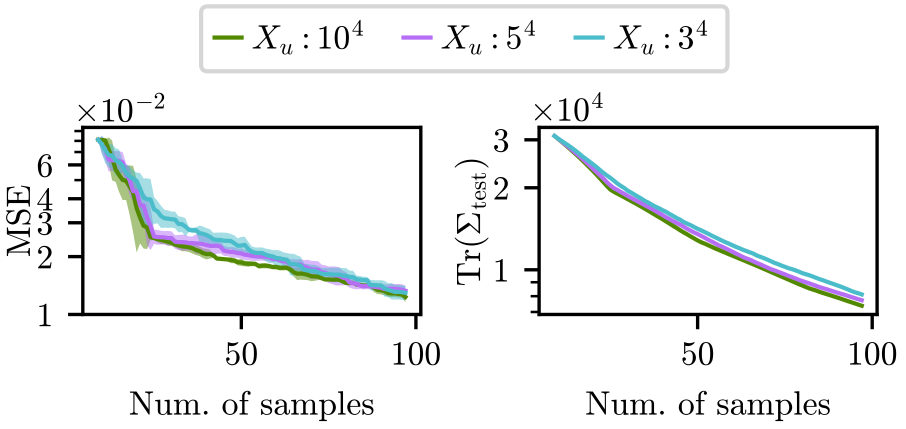
5.7.2 Bayesian Experimental Design

We use the BED strategy to sample 88 additional observations for a total of 97, the same number sampled in the on-axis regression. For the utility input X_u, we consider 4D uniform grids of size 10^4, 5^4, and 3^4 uniformly distributed within the input space with $\gamma^2 = 1/50, 1/10,$

and 1/5, respectively. Higher resolution grids are not considered due to high computational costs. The BED results with the various choices of utility input are shown in Fig. 5.20. The trade-off between grid size and the efficacy of the BED strategy is more noticeable than in the 2D case. Still, we note how even the cheapest, lowest-resolution 3^4 grid performs relatively well.



(a) Regular GP



(b) Derivative GP

Figure 5.20: MSE and $\mathrm{Tr}(\Sigma_{\text{test}})$ of the regular and derivative GP models after each of the 88 BED iterations for the various choices of utility input. The models are tested on a 25^4 grid uniformly distributed within the input space. The solid lines represent the mean values of five BED runs with different initial random seeds and the shaded areas the standard deviation.

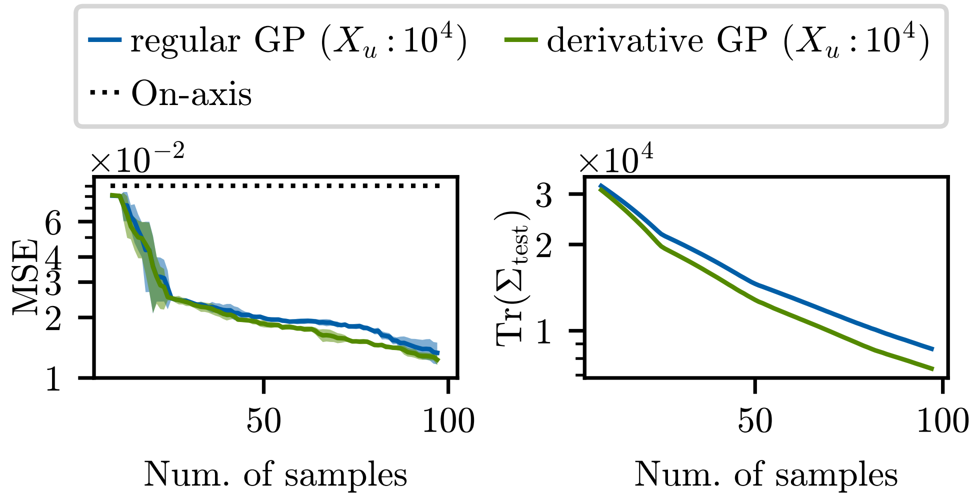
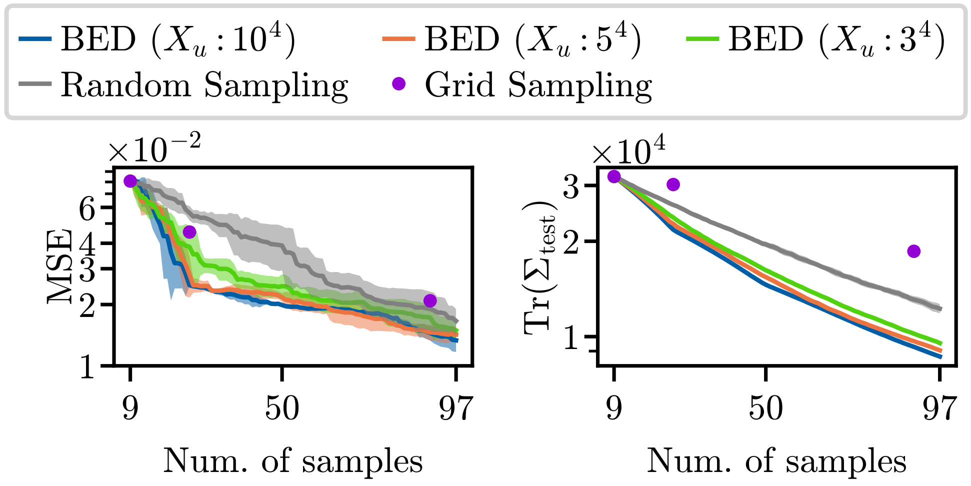


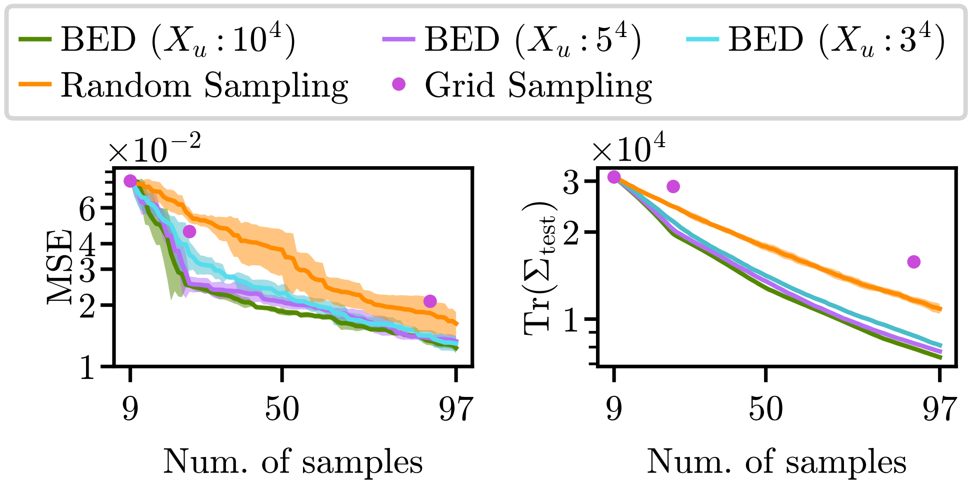
Figure 5.21: Comparison of the MSE and $\text{Tr}(\Sigma_{\text{test}})$ of the various models. The models are tested on a 25^4 grid uniformly distributed within the input space. The solid lines represent the mean values of five BED runs with different initial random seeds and the shaded areas the standard deviation

We compare the results of the regular and derivative GP models to those of the on-axis regression in Fig. 5.21. For simplicity, we only compare the BED results with the 5^4 grid choice of utility input, but they generalize to the other choices. The derivative GP once more outperforms the regular GP, resulting in lower MSE and $\text{Tr}(\Sigma_{\text{test}})$ values. Unlike the 1D toy and 2D high-energy physics cases, the edge that the derivative GP has over the regular GP is less pronounced. This is because the derivative information is less accurate as the gradients are approximated by multiple sigmoid functions. Both GP models greatly outperform the on-axis regression with lower MSE values. With only nine samples, the MSE of the GP models is lower than that of the on-axis regression. These results emphasize the potential of GP models for learning the dependence of experimental observables on multiple nuisance parameters.

To answer the important question of whether the BED has selected samples that efficiently reduce $\text{Tr}(\Sigma_{\text{test}})$, we again compare the BED results to random and grid sampling. In the random sampling case, we sequentially add 88 uniformly distributed random observations to the training set. In the grid sampling case, we sample 4D uniform grids of size 2^4 and 3^4, which are separately added to the training set. The results are shown in Fig. 5.22.



(a) Regular GP



(b) Derivative GP

Figure 5.22: MSE and Tr(Σ_{test}) after each iteration of the various sampling strategies. The models are tested on a 25^4 grid uniformly distributed within the input space. In the BED and random sampling cases, the solid lines represent the mean values of five sampling runs using different initial random seeds and the shaded areas the standard deviation.

The results show that the BED has a clear advantage over the other sampling methods. This advantage is more evident in higher dimensions, where an efficient exploration of the input space is key. The grid sampling strategy performs the worst. While uniform grids are a common sampling strategy, they are rarely optimal in higher dimensions due to their exponential growth with the dimensionality of the input space. Random sampling results in lower $\mathrm{Tr}(\Sigma_{\mathrm{test}})$ values than grid sampling, but it has large variations in the MSE. Overall, the BED strategy has the lowest MSE and $\mathrm{Tr}(\Sigma_{\mathrm{test}})$, regardless of the choice of utility input or GP model.

5.8 Discussion

Nuisance parameters, which can be viewed as sources of systematic uncertainty, play a crucial role in high-precision measurements at the LHC, where their accurate estimation can be a pivotal factor in a potential new discovery. However, rigorous assessments of the impact of nuisance parameters on the experimental response are generally intractable. Experiments are often marred by multiple sources of systematic uncertainty, which create a high-dimensional parameter space that must be characterized. Several techniques have been employed for assessing the impact of nuisance parameters on the experimental observables, but most of them rely on simplifying assumptions, such as the factorization of the underlying correlations of the parameters. This approach is typically extended to assume the impact of the parameters on the observables also factorizes. In this chapter, we argue that such assumptions are not always reliable.

We propose a more accurate method for assessing the impact of multiple nuisance parameters on observables of the detector response without assuming factorization. Our approach uses GP regression to learn a representation of observables as a function of the nuisance parameters. If the analytical forms of the observables are known, we take advantage of this and use their gradients to enhance the GP regression. We test our approach by estimating the efficiency of high-energy physics events as a function of multiple (two and four) nuisance parameters.

In our examples, the GP regression outperforms the factorized approach even when conditioned on significantly fewer samples. However, in cases where the impact of the nuisance parameters may be more complicated to learn, more samples may be needed to condition the GP models. In such cases, having an efficient sampling strategy is essential. We use a Bayesian sampling strategy to efficiently sample the observable space and select the samples that are likely to reduce the predictive uncertainty of the GP models. We show that the Bayesian sampling approach is more efficient than random or grid sampling, which are rarely optimal in high-dimensional spaces.

Chapter 6

Conclusion

Recent advances in machine learning for jet classification have focused on deep, low-level networks. These networks learn directly from the jet constituents and, thus, have access to information at the lowest level of abstraction. It comes as no surprise that they currently provide state-of-the-art results in various jet classification tasks [46, 64–67]. But a significant drawback of these networks is the lower interpretability of their results, as it is not clear which specific functions of their complex input they have learned.

On the other hand, high-level networks that learn from jet observables are generally easier to interpret since physicists can control the functional forms of their inputs. The higher interpretability generally comes at the cost of decreased performance. This is because it can be challenging to capture all relevant information in the jet constituents into a finite set of observables.

The work presented in this dissertation has explored the performance and learning strategies of low- and high-level neural networks in the context of multi-prong jet classification. While the classification of jets with up to four hard sub-jets has been widely studied in the literature [54–60, 73–75], we ask whether these results could be extended to jets with additional sub-jets. To our knowledge, this is the first study exploring the classification strategies of neural networks on jets with up to eight hard sub-jets.

In Chapter 3, we have compared the results of two mainstream low-level networks (a PFN [48] and a Transformer [45]) to networks trained on traditional jet observables (N-subjettiness [92] and EFPs [4]). We find a small but non-negligible gap in performance between the low- and high-level networks, suggesting that the traditional features fail to capture all information used by the low-level networks.

In Chapter 4, we have introduced a novel family of jet observables, Jet Rotational Metrics (JRMs), designed to capture features of the discrete rotational symmetry of the jets. By supplementing the traditional observables with IRC-safe JRMs, we are able to narrow the gap between low- and high-level networks significantly. These results are impressive, indicating that the JRMs are able to capture unique information not captured by the traditional observables, and that this information is highly relevant for the classification task.

We bridge the gap between low- and high-level networks by also including IRC-unsafe JRMs. IRC-unsafe information is not calculable in the perturbative QCD regime and, thus, warrants careful consideration when accounting for the statistical uncertainties in the data on which the models are trained. Our results suggest that the learning strategies of low-level networks may rely, in part, on IRC-unsafe information. Based on our results, we emphasize that physicists should be more careful at the prospect of end-to-end low-level classifiers, where the nature of the information learned by the networks is not always known.

In Chapter 5, we change gears and study another important aspect of physics studies at the LHC: the estimation of systematic uncertainties. Directly evaluating detector response observables, such as the event efficiency, is generally a computationally expensive task due to the large number of parameters that may impact the experimental response. Some parameters can be estimated from theoretical predictions, but others must be inferred from experimental data. The latter are referred to as nuisance parameters and are often treated as sources of systematic uncertainty. A typical approach is to assume that the nuisance parameters factorize. This approach is frequently extended to assume the impact of the individual nuisance parameters on the detector response also factorizes. While this approach generally decreases the computational costs significantly, we show that it is not always valid, particularly when the observables depend on multiple nuisance parameters, which create a high-dimensional space that must be characterized.

We introduce a technique to calculate the impact of the nuisance parameters on the detector response without assuming factorization. Our approach uses Gaussian process models to regress detector observables as functions of the nuisance parameters. When the functional forms of the observables are known, we calculate their gradients and use them to enhance the regressions. The results greatly outperform the factorized approach. In addition, we provide a technique to efficiently explore the space of the experimental response, thus reducing the predictive uncertainty of the Gaussian process models by using Bayesian Experimental Design. We show that this technique is more efficient than random and grid sampling, particularly when dealing with multiple nuisance parameters. Our approach shows great potential for the accurate and efficient analysis of systematic uncertainties in physics experiments, which deserve careful examination lest they mimic or obscure discoveries.

Bibliography

[1] M. Cacciari, G.P. Salam and G. Soyez, *The Anti-k_t jet clustering algorithm*, *JHEP* **04** (2008) 063 [0802.1189].

[2] C. Cesarotti and J. Thaler, *A robust measure of event isotropy at colliders*, 2020. 10.1007/JHEP08(2020)084.

[3] P.T. Komiske, E.M. Metodiev and J. Thaler, *Metric space of collider events*, *Physical Review Letters* **123** (2019) .

[4] P.T. Komiske, E.M. Metodiev and J. Thaler, *Energy flow polynomials: A complete linear basis for jet substructure*, *JHEP* **04** (2018) 013 [1712.07124].

[5] Y.L. *et al.*, *Resolving extreme jet substructure*, *Journal of High Energy Physics* **2022** (2022) .

[6] X. Fan, T.G. Myers, B.A.D. Sukra and G. Gabrielse, *Measurement of the electron magnetic moment*, *Phys. Rev. Lett.* **130** (2023) 071801.

[7] G. Bertone, D. Hooper and J. Silk, *Particle dark matter: evidence, candidates and constraints*, *Physics Reports* **405** (2005) 279.

[8] J.A. Frieman, M.S. Turner and D. Huterer, *Dark energy and the accelerating universe*, *Annual Review of Astronomy and Astrophysics* **46** (2008) 385.

[9] A.D. Sakharov, *Violation of CP Invariance, C asymmetry, and baryon asymmetry of the universe*, *Pisma Zh. Eksp. Teor. Fiz.* **5** (1967) 32.

[10] V.V. Gligorov and M. Williams, *Efficient, reliable and fast high-level triggering using a bonsai boosted decision tree*, *Journal of Instrumentation* **8** (2013) P02013.

[11] J.A. *et al.*, *Hep community white paper on software trigger and event reconstruction*, 2018.

[12] J.D. *et al.*, *FPGA-accelerated machine learning inference as a service for particle physics computing*, *Computing and Software for Big Science* **3** (2019) .

[13] T.A. collaboration, *A neural network clustering algorithm for the ATLAS silicon pixel detector*, *Journal of Instrumentation* **9** (2014) P09009.

[14] V.K. *et al.*, *Observation of the diphoton decay of the higgs boson and measurement of its properties*, *The European Physical Journal C* **74** (2014) .

[15] D.D. *et al.*, *Machine-learning-based global particle-identification algorithms at the LHCb experiment*, *Journal of Physics: Conference Series* **1085** (2018) 042038.

[16] S.t. Badger, *Machine learning and LHC event generation*, *SciPost Phys.* **14** (2023) 079 [2203.07460].

[17] TMVA collaboration, *TMVA - Toolkit for Multivariate Data Analysis*, physics/0703039.

[18] R. Brun, F. Rademakers and S. Panacek, *ROOT, an object oriented data analysis framework*, .

[19] Y. LeCun, Y. Bengio and G. Hinton, *Deep learning*, *Nature* **521** (2015) 436.

[20] C.t. Bierlich, *A comprehensive guide to the physics and usage of PYTHIA 8.3*, 2203.11601.

[21] J.t. Alwall, *The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations*, *JHEP* **07** (2014) 079 [1405.0301].

[22] DELPHES 3 collaboration, *DELPHES 3, A modular framework for fast simulation of a generic collider experiment*, *JHEP* **1402** (2014) 057 [1307.6346].

[23] W.B. *et al.*, *Observation of planar three-jet events in e^+e^- annihilation and evidence for gluon bremsstrahlung*, *Physics Letters B* **91** (1980) 142.

[24] C.B. *et al.*, *Evidence for gluon bremsstrahlung in e^+e^- annihilations at high energies*, *Physics Letters B* **86** (1979) 418.

[25] D.P.t. Barber, *Discovery of three-jet events and a test of quantum chromodynamics at petra*, *Phys. Rev. Lett.* **43** (1979) 830.

[26] R.B. *et al.*, *Evidence for planar events in e^+e^- annihilation at high energies*, *Physics Letters B* **86** (1979) 243.

[27] CDF COLLABORATION collaboration, *Observation of top quark production in $\overline{p}p$ collisions with the collider detector at fermilab*, *Phys. Rev. Lett.* **74** (1995) 2626.

[28] S.A. *et al.*, *Observation of the top quark*, *Physical Review Letters* **74** (1995) 2632.

[29] R. Atkin, *Review of jet reconstruction algorithms*, *J. Phys. Conf. Ser.* **645** (2015) 012008.

[30] A.S. *et al.*, *Particle-flow reconstruction and global event description with the CMS detector*, *Journal of Instrumentation* **12** (2017) P10003.

[31] M.A. *et al.*, *Jet reconstruction and performance using particle flow with the ATLAS detector*, *The European Physical Journal C* **77** (2017) .

[32] J. Tseng and H. Evans, *Sequential recombination algorithm for jet clustering and background subtraction*, *Physical Review D* **88** (2013) .

[33] S. Catani, Y.L. Dokshitzer, M.H. Seymour and B.R. Webber, *Longitudinally invariant K_t clustering algorithms for hadron hadron collisions*, *Nucl. Phys. B* **406** (1993) 187.

[34] Y. Dokshitzer, G. Leder, S. Moretti and B. Webber, *Better jet clustering algorithms*, *Journal of High Energy Physics* **1997** (1997) 001.

[35] J. Thaler and K. Van Tilburg, *Identifying Boosted Objects with N-subjettiness*, *JHEP* **03** (2011) 015 [1011.2268].

[36] V. Nair and G.E. Hinton, *Rectified linear units improve restricted boltzmann machines*, in *ICML*, 2010.

[37] P.T. Komiske, E.M. Metodiev and M.D. Schwartz, *Deep learning in color: towards automated quark/gluon jet discrimination*, *Journal of High Energy Physics* **2017** (2017) .

[38] P. Baldi, P. Sadowski and D. Whiteson, *Searching for Exotic Particles in High-Energy Physics with Deep Learning*, *Nature Commun.* **5** (2014) 4308 [1402.4735].

[39] L.G.t. Almeida, *Playing Tag with ANN: Boosted Top Identification with Pattern Recognition*, *JHEP* **07** (2015) 086 [1501.05968].

[40] L.t. de Oliveira, *Jet-images — deep learning edition*, *JHEP* **07** (2016) 069 [1511.05190].

[41] P.t. Baldi, *Jet Substructure Classification in High-Energy Physics with Deep Neural Networks*, *Phys. Rev.* **D93** (2016) 094034 [1603.09349].

[42] A.J. Larkoski, I. Moult and B. Nachman, *Jet Substructure at the Large Hadron Collider: A Review of Recent Advances in Theory and Machine Learning*, *Phys. Rept.* **841** (2020) 1 [1709.04464].

[43] G. Kasieczka, T. Plehn, M. Russell and T. Schell, *Deep-learning Top Taggers or The End of QCD?*, *JHEP* **05** (2017) 006 [1701.08784].

[44] P. Baldi, *Deep Learning in Science*, Cambridge University Press (2021), 10.1017/9781108955652.

[45] A.V. *et al.*, *Attention is all you need*, *CoRR* **abs/1706.03762** (2017) [1706.03762].

[46] H. Qu, C. Li and S. Qian, *Particle Transformer for jet tagging*, in *Proceedings of the 39th International Conference on Machine Learning*, pp. 18281–18292, 2022 [2202.03772].

[47] V. Mikuni and F. Canelli, *Point cloud transformers applied to collider physics*, *Machine Learning: Science and Technology* **2** (2021) 035027.

[48] P.T. Komiske, E.M. Metodiev and J. Thaler, *Energy Flow Networks: Deep Sets for Particle Jets*, *JHEP* **01** (2019) 121 [1810.05165].

[49] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R.R. Salakhutdinov and A.J. Smola, *Deep sets*, *Advances in neural information processing systems* **30** (2017) .

[50] G.t. Sterman, *Handbook of perturbative QCD*, *Rev. Mod. Phys.* **67** (1995) 157.

[51] G. Sterman and S. Weinberg, *Jets from quantum chromodynamics*, *Phys. Rev. Lett.* **39** (1977) 1436.

[52] P. Konar, V.S. Ngairangbam and M. Spannowsky, *Energy-weighted message passing: an infra-red and collinear safe graph neural network algorithm*, *Journal of High Energy Physics* **2022** (2022) .

[53] J.t. Butterworth, *Jet substructure as a new Higgs search channel at the LHC*, *Phys.Rev.Lett.* **100** (2008) 242001 [0802.2470].

[54] A.t. Hook, *High Multiplicity Searches at the LHC Using Jet Masses*, *Phys. Rev. D* **85** (2012) 055029 [1202.0558].

[55] ATLAS collaboration, *Performance of top-quark and W-boson tagging with ATLAS in Run 2 of the LHC*, *Eur. Phys. J. C* **79** (2019) 375 [1808.07858].

[56] CMS collaboration, *Identification techniques for highly boosted W bosons that decay into hadrons*, *JHEP* **12** (2014) 017 [1410.4227].

[57] ATLAS collaboration, *Measurement of jet-substructure observables in top quark, W boson and light jet production in proton-proton collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector*, *JHEP* **08** (2019) 033 [1903.02942].

[58] ATLAS collaboration, *Measurement of the Lund Jet Plane Using Charged Particles in 13 TeV Proton-Proton Collisions with the ATLAS Detector*, *Phys. Rev. Lett.* **124** (2020) 222002 [2004.03540].

[59] CMS collaboration, *Measurement of jet substructure observables in* $\mathrm{t\bar{t}}$ *events from proton-proton collisions at $\sqrt{s} =$ 13TeV*, *Phys. Rev. D* **98** (2018) 092014 [1808.07340].

[60] CMS collaboration, *Identification of heavy, energetic, hadronically decaying particles using machine-learning techniques*, *JINST* **15** (2020) P06005 [2004.08262].

[61] CMS collaboration, *Search for top squark production in fully-hadronic final states in proton-proton collisions at $\sqrt{s} =$ 13 TeV*, *Phys. Rev. D* **104** (2021) 052001 [2103.01290].

[62] ATLAS collaboration, *Search for heavy diboson resonances in semileptonic final states in pp collisions at $\sqrt{s}$=13 TeV with the ATLAS detector*, *Eur. Phys. J. C* **80** (2020) 1165 [2004.14636].

[63] A.J. Larkoski, G.P. Salam and J. Thaler, *Energy Correlation Functions for Jet Substructure*, *JHEP* **06** (2013) 108 [1305.0007].

[64] J. Collado, K. Bauer, E. Witkowski, T. Faucett, D. Whiteson and P. Baldi, *Learning to isolate muons*, *Journal of High Energy Physics* **2021** (2021) .

[65] J. Collado, J.N. Howard, T. Faucett, T. Tong, P. Baldi and D. Whiteson, *Learning to identify electrons*, *Physical Review D* **103** (2021) .

[66] E. Witkowski, B. Nachman and D. Whiteson, *Learning to isolate muons in data*, *Phys. Rev. D* **108** (2023) 092008.

[67] T. Faucett, S.-C. Hsu and D. Whiteson, *Learning to identify semi-visible jets*, *JHEP* **12** (2022) 132 [2208.10062].

[68] T. Faucett, J. Thaler and D. Whiteson, *Mapping Machine-Learned Physics into a Human-Readable Space*, *Phys. Rev. D* **103** (2021) 036020 [2010.11998].

[69] A.t. Butter, *The Machine Learning landscape of top taggers*, *SciPost Phys.* **7** (2019) 014 [1902.09914].

[70] H. Qu and L. Gouskos, *ParticleNet: Jet Tagging via Particle Clouds*, *Phys. Rev. D* **101** (2020) 056019 [1902.08570].

[71] V. Mikuni and F. Canelli, *Point cloud transformers applied to collider physics*, *Mach. Learn. Sci. Tech.* **2** (2021) 035027 [2102.05073].

[72] CMS collaboration, *Search for resonances decaying to three W bosons in the hadronic final state in proton-proton collisions at $\sqrt{s}$ = 13 TeV*, 2112.13090.

[73] J.A. Aguilar-Saavedra, F.R. Joaquim and J.F. Seabra, *Mass Unspecific Supervised Tagging (MUST) for boosted jets*, *JHEP* **03** (2021) 012 [2008.12792].

[74] J.A. Aguilar-Saavedra, J.H. Collins and R.K. Mishra, *A generic anti-QCD jet tagger*, *JHEP* **11** (2017) 163 [1709.01087].

[75] C. Chen, *Reconstruction and identification of $H \to WW^*$ with high transverse momentum in the full hadronic final state*, *Phys. Rev. D* **103** (2021) 033005 [2012.02884].

[76] M. Cacciari, G.P. Salam and G. Soyez, *FastJet User Manual*, *Eur.Phys.J.* **C72** (2012) 1896 [1111.6097].

[77] K. Datta and A. Larkoski, *How Much Information is in a Jet?*, *JHEP* **06** (2017) 073 [1704.08249].

[78] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019.

[79] J. Gallicchio and M.D. Schwartz, *Quark and Gluon Jet Substructure*, *JHEP* **04** (2013) 090 [`1211.7038`].

[80] R. Tibshirani, *Regression shrinkage and selection via the lasso*, *Journal of the Royal Statistical Society (Series B)* **58** (1996) 267.

[81] A. Hoffmann, R. Kwok and P. Compton, *Using subclasses to improve classification learning*, in *Machine Learning: ECML 2001*, L. De Raedt and P. Flach, eds., (Berlin, Heidelberg), pp. 203–213, Springer Berlin Heidelberg, 2001.

[82] Y. Luo, *Can subclasses help a multiclass learning problem?*, in *2008 IEEE Intelligent Vehicles Symposium*, pp. 214–219, 2008, DOI.

[83] A. Bogatskiy, B. Anderson, J.T. Offermann, M. Roussi, D.W. Miller and R. Kondor, *Lorentz group equivariant neural network for particle physics*, 2020.

[84] S. Gong, Q. Meng, J. Zhang, H. Qu, C. Li, S. Qian et al., *An efficient lorentz equivariant graph neural network for jet tagging*, *Journal of High Energy Physics* **2022** (2022) .

[85] A.t. Shmakov, *SPANet: Generalized Permutationless Set Assignment for Particle Physics using Symmetry Preserving Attention*, `2106.03898`.

[86] S. Qiu, S. Han, X. Ju, B. Nachman and H. Wang, *Holistic approach to predicting top quark kinematic properties with the covariant particle transformer*, *Physical Review D* **107** (2023) .

[87] C. Shimmin, *Particle convolution for high energy physics*, *arXiv preprint arXiv:2107.02908* (2021) .

[88] S. Brandt, C. Peyrou, R. Sosnowski and A. Wroblewski, *The principal axis of jets — an attempt to analyse high-energy collisions as two-body processes*, *Physics Letters* **12** (1964) 57.

[89] A. De Rújula, J. Ellis, E. Floratos and M. Gaillard, *QCD predictions for hadronic final states in e^+e^- annihilation*, *Nuclear Physics B* **138** (1978) 387.

[90] A.J. Larkoski, J. Thaler and W.J. Waalewijn, *Gaining (mutual) information about quark/gluon discrimination*, *Journal of High Energy Physics* **2014** (2014) .

[91] A.J. Larkoski, D. Neill and J. Thaler, *Jet shapes with the broadening axis*, *Journal of High Energy Physics* **2014** (2014) .

[92] J. Thaler and K.V. Tilburg, *Maximizing boosted top identification by minimizing N-subjettiness*, *Journal of High Energy Physics* **2012** (2012) .

[93] A.G. Wilson and P. Izmailov, *Bayesian deep learning and a probabilistic perspective of generalization*, *CoRR* **abs/2002.08791** (2020) [2002.08791].

[94] H. Zabrodsky, S. Peleg and D. Avnir, *Continuous symmetry measures*, *Journal of the American Chemical Society* **114** (1992) 7843.

[95] J.D. Bjorken and S.J. Brodsky, *Statistical model for electron-positron annihilation into hadrons*, *Phys. Rev. D* **1** (1970) 1416.

[96] T. Wu, L. Pan, J. Zhang, T. Wang, Z. Liu and D. Lin, *Density-aware chamfer distance as a comprehensive metric for point cloud completion*, 2021.

[97] A.A. Taha and A. Hanbury, *An efficient algorithm for calculating the exact hausdorff distance*, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **37** (2015) 2153.

[98] O. Pele and M. Werman, *Fast and robust earth mover's distances*, in *2009 IEEE 12th International Conference on Computer Vision*, pp. 460–467, 2009, DOI.

[99] P.G. *et al.*, *Systematics of quark/gluon tagging*, *Journal of High Energy Physics* **2017** (2017) .

[100] CDF collaboration, *High-precision measurement of the W boson mass with the CDF II detector*, *Science* **376** (2022) 170.

[101] Muon g-2 collaboration, *Measurement of the Positive Muon Anomalous Magnetic Moment to 0.46 ppm*, *Phys. Rev. Lett.* **126** (2021) 141801 [2104.03281].

[102] A.C. atlas. publications@ cern. ch, G. Aad, B. Abbott, D. Abbott, A.A. Abud, K. Abeling et al., *Jet energy scale and resolution measured in proton-proton collisions at $\sqrt{s}$= 13 TeV with the atlas detector*, *The European Physical Journal C* **81** (2021) 689.

[103] M. Vos, *Top-quark mass measurements at the LHC: alternative methods*, 2016.

[104] G. Aad et al., *Jet energy measurement and its systematic uncertainty in proton–proton collisions at $\sqrt{s} = 7$ TeV with the ATLAS detector*, *The European Physical Journal C* **75** (2015) .

[105] M.A. *et al.*, *Search for heavy charged long-lived particles in proton–proton collisions at $\sqrt{s}$=13 TeV using an ionisation measurement with the ATLAS detector*, *Physics Letters B* **788** (2019) 96.

[106] M.A. *et al.*, *Combination of searches for invisible higgs boson decays with the ATLAS experiment*, *Physical Review Letters* **122** (2019) .

[107] M.t. Frate, *Modeling Smooth Backgrounds and Generic Localized Signals with Gaussian Processes*, 1709.05681.

[108] A.G. *et al.*, *Model selection and signal extraction using gaussian process regression*, *Journal of High Energy Physics* **2023** (2023) .

[109] G. Bertone, M.P. Deisenroth, J.S. Kim, S. Liem, R.R. de Austri and M. Welling, *Accelerating the bsm interpretation of lhc data with machine learning*, *Physics of the Dark Universe* **24** (2019) 100293.

[110] K. Cranmer, *Practical Statistics for the LHC*, in *2011 European School of High-Energy Physics*, pp. 267–308, 2014, DOI [`1503.07622`].

[111] R. VON MISES, *Chapter vii - probability inference. bayes' method*, in *Mathematical Theory of Probability and Statistics*, R. VON MISES, ed., pp. 329–367, Academic Press (1964), DOI.

[112] C.E. Rasmussen and W.C.K. I., *Further issues and conclusions*, in *Gaussian processes for machine learning*, p. 191–192, MIT Press (2008).

[113] T. Pinder and D. Dodd, *Gpjax: A gaussian process framework in jax*, *Journal of Open Source Software* **7** (2022) 4455.

[114] N. Simpson and L. Heinrich, *neos: End-to-end-optimised summary statistics for high energy physics*, *Journal of Physics: Conference Series* **2438** (2023) 012105.

[115] D.E. *et al.*, *Scaling gaussian process regression with derivatives*, in *Neural Information Processing Systems*, 2018, https://api.semanticscholar.org/CorpusID:49739498.

[116] F. de Roos, A. Gessner and P. Hennig, *High-dimensional gaussian process inference with derivatives*, 2021.

[117] M.P. *et al.*, *Scaling gaussian processes with derivative information using variational inference*, 2021.

[118] D.P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, *CoRR* **abs/1412.6980** (2014) [`1412.6980`].

[119] L.H. *et al.*, *Sherpa: Robust hyperparameter optimization for machine learning*, *SoftwareX* (2020) .

[120] A.t. Paszke, *Pytorch: An imperative style, high-performance deep learning library*, in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett, eds., pp. 8024–8035, Curran Associates, Inc. (2019), http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[121] F.t. Chollet, "Keras." `https://keras.io`, 2015.

[122] M.A. *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015.

[123] D. Napoletano and G. Soyez, *Computing N-subjettiness for boosted jets*, *Journal of High Energy Physics* **2018** (2018) .

Appendix A

Multi-Prong Jet Classification

A.1 Technical Details of the Classifiers

In this appendix, we describe the details of the machine learning models and network architectures. The Transformer and PFN models are trained on the three-momenta of the simulated jet constituents, which are preprocessed normalizing the jet p_{T} to unity and subtracting the p_{T}-weighted angular means, as described in Section 3.2. The dense networks are trained on high-level variables, which are strict functions of the preprocessed constituent three-momenta. This ensures that the dense networks have access to only a subset of the information available in the low-level jet constituents.

The 10-fold average accuracy of the models and the number of parameters are summarized in Table A.1. Common properties across all networks include ReLu [36] activation functions for all hidden layers and 7-dimensional softmax output functions to classify between all seven sub-jet classes. All networks are trained using the Adam [118] optimizer for up to 1000 epochs and with a batch size of 256.

Table A.1: Summary of the machine learning models used in the classification task. The table shows a brief description of each of the models, as well as the number of trainable parameters and the accuracy measured using 10-fold cross-validation.

Model	Description	No. of Params.	Accuracy
Transformer	Transformer Network trained on the jet constituents.	1,388,807	91.27 ± 0.31 %
PFN	Particle-Flow Network trained on the jet constituents.	1,205,895	89.19 ± 0.23 %
DNN_{136}	Fully-connected neural network trained on the 135 N-subjettiness observables and the norm. jet mass.	2,732,519	86.90 ± 0.20 %
DNN_{299}	Fully-connected neural network trained on the 135 N-subjettiness, observables the normalized jet mass, and the full set of EFP observables.	2,862,919	89.23 ± 0.26 %
DNN_{31}	Fully-connected neural network trained on the 31 LASSO-selected observables.	2,622,663	89.11 ± 0.32 %

All networks were optimized by a hyperparameter search using the Sherpa [119] hyperparameter optimization library while ensuring that the range of trainable parameters in the optimization is roughly the same for all networks.[1] The hyperparameters of the dense networks were optimized in the ranges: intermediate dimension $[600, 800]$, dropout ratio $[0.3, 0.4]$, and learning rate $[10^{-3}, 10^{-4}]$. The hyperparameters of the PFN were optimized in the ranges: ϕ-module dimension $[128, 1024]$, F-module dimension $[128, 2014]$, F-module drop out rate $[0.1, 0.2]$, and learning rate $[10^{-3}, 10^{-4}]$. The hyperparameters of the Transformer were optimized in the ranges: hidden layer size $[256, 512]$, number of layers $[4, 8]$, and learning rate $[10^{-3}, 10^{-4}]$.

Training of the Transformer and all dense networks was implemented in PyTorch [120] using NVIDIA V100 GPUs. Training of the PFN was implemented in Keras [121] with Tensorflow [122] backend using an NVIDIA RTX 2080 Ti GPU. The accuracy of the networks was

[1]PFN, Transformers, and DNN_{136} models with a larger number of parameters were also considered, resulting in slightly better accuracy values, but these networks were excluded from the study as their increased performances were only marginal when compared to the large computational cost of training the larger models.

checked to be insensitive to the deep learning library and GPU. Using the NVIDIA RTX 2080 Ti GPU as a benchmark, the average inference time for 256 samples was calculated to be 2.46$\pm$0.12 ms for the Transformer; 2.90$\pm$0.36 ms for the PFN; 0.58$\pm$0.01 ms for the DNN$_{136}$; 0.59$\pm$0.02 ms for the DNN$_{299}$; and 0.60$\pm$0.03 ms for the DNN$_{31}$. We note that we did not perform any inference optimization. The inference times are provided merely to give a sense of the scale of the latency of each algorithm.

Appendix B

Jet Rotational Metrics

B.1 Further Discussion on the Choice of J_n

We mention in Sec. 4.3 how, ideally, the JRMs would measure the distance between a jet J and its closest C_n-symmetric element. In practice, however, finding this element would be computationally prohibitive due to the infinite possible ways to construct C_n-symmetric elements, all with different numbers of constituents, radii, and orientations. To simplify this task, we developed a recipe to construct the reference jet by (1) considering n equidistant points centered at $(\eta, \phi) = (0, 0)$, (2) placing them at a radius equal to the p_{T}-weighted constituent mean radius, and (3) rotating points to minimize their distance to J. The final arrangement of the points represents the constituents in the reference jet, which we call J_n.

In this appendix, we test whether the aforementioned recipe for J_n results in good discriminants or if different choices yield features with better classification performances. We test step (2) by comparing the standard JRMs to similar features where we set the radius equal to the jet radius. The results are shown in the second to last row of Table B.1. These new features achieve an overall accuracy of $87.83 \pm 0.47\%$, which is roughly 1.5% less than that of the standard JRMs.

We also test step (3) by comparing the standard JRMs to similar features where the polygon was not rotated to minimize its distance to the jet. The results are shown in the last row of Table B.1. These new features achieve an overall accuracy of $87.81 \pm 0.23\%$, which is also roughly 1.5% less than that of the standard JRMs.

The results suggest that the recipe used to construct J_n results in good jet substructure discriminants. We note, however, that while this recipe greatly simplifies the search for C_n-symmetric elements, it is not computationally trivial since it requires the calculation of the mean constituent radius and angular optimization. We show that when these steps are omitted, some information is lost, resulting in lower accuracies, but even then, the simplified features achieve good classification performances.

Table B.1: Mean 10-fold prediction accuracy and statistical uncertainty of DNNs operating on the specified input features.

Input	Overall Acc. (%)
Standard JRM	**89.30** ± 0.26
JRM modified so the radius of the polygon is at the jet radius	87.83 ± 0.47
JRM without the θ optimization	87.81 ± 0.23

B.2 Per-Class Accuracy

Here, we show the results of the various networks on each class of the multi-prong classification dataset. The results of the dense networks operating on IRC-safe observables are shown in Table B.2. The results of the networks operating on IRC-agnostic input are shown in Table B.3.

Table B.2: Mean 10-fold class prediction accuracy and statistical uncertainty of the DNNs operating on the specified IRC-safe observables. See Table 4.1 for the overall accuracy values.

Net.	Input	Acc. per Class (%)						
		$N = 1$	$N = 2$	$N = 3$	$N = 4b$	$N = 4q$	$N = 6$	$N = 8$
DNN	JRMs	92.3±0.8	79.3±1.3	78.6±1.0	76.6±1.1	97.1±0.5	91.1±0.6	83.2±1.0
DNN	N-subs, EFPs	93.0±0.6	83.0±0.9	82.9±1.3	83.7±0.5	98.0±0.3	92.0±0.7	90.1±1.2
DNN	N-subs, EFPs, JRMs	**94.6**±0.5	**85.5**±1.1	**85.2**±0.8	**85.1**±1.3	**98.5**±0.2	**94.7**±0.6	**91.4**±0.8

Table B.3: Mean 10-fold class prediction accuracy and statistical uncertainty of the various networks. The input to the PFN and Transformer is the constituents' three-momentum, which is zero-padded to have a uniform length of 230. See [5] for further details about the PFN and Transformer networks. The full set of 30 JRMs is used as input to the DNN. See Table 4.2 for the overall accuracy values.

Net.	Input	Acc. per Class (%)						
		$N = 1$	$N = 2$	$N = 3$	$N = 4b$	$N = 4q$	$N = 6$	$N = 8$
DNN	JRMs	96.0±0.6	86.6±1.1	84.2±0.5	80.1±0.9	98.2±0.3	93.4±0.7	88.4±1.0
DNN	N-subs, EFPs, JRMs	95.7±0.8	87.2±1.3	86.4±0.8	**86.2**±1.0	**98.6**±0.3	**94.9**±0.6	**91.5**±0.6
PFN	Consts.	**96.4**±0.6	85.2±1.3	82.9±1.1	78.8±0.5	98.0±0.3	93.4±0.9	89.7±0.9
Trans.	Consts.	96.0±0.5	**88.1**±1.2	**86.7**±1.1	84.6±1.2	98.5±0.3	94.0±1.1	91.1±1.2

B.3 JRM Ratios

In this dissertation, we have used the high-level observables as input to dense networks, but a common approach is to utilize their ratios in statistical-cut-based analyses. For example, ratios of $\tau_{\mathrm{N}}/\tau_{\mathrm{N}-1}$ N-subjettiness observables have been used to classify jets with different topologies [92, 123]. A jet that is very "N-subjetty" will show a relatively large difference between the τ_{N} and τ_{N-1} variables, which will be reflected in their ratio.

Because of the nature of JRMs, we do not expect ratios of $\mathrm{JRM}_n/\mathrm{JRM}_{n-1}$ to vary greatly for large n. Useful ratios could be $\mathrm{JRM}_3/\mathrm{JRM}_2$, and between JRM_3 and a larger n that captures features of the isotropy of the jets, such as $\mathrm{JRM}_8/\mathrm{JRM}_3$. Distributions of such ratios are shown in Fig. B.1. We do not perform an analysis using statistical cuts on the ratios, but their distributions show a good degree of class separation.

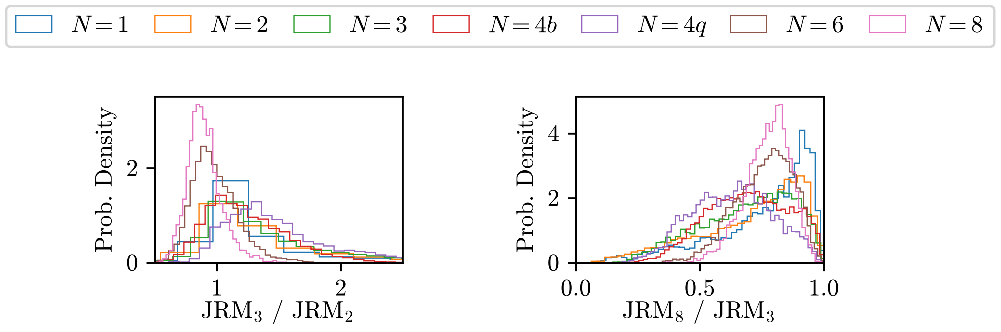


Figure B.1: Distributions of the selected JRM ratios.

B.4 JRM Distributions

Here, we show the distributions of the 30 JRM observables used in the classification task. The distributions are shown in Fig. B.2 and Fig. B.3.

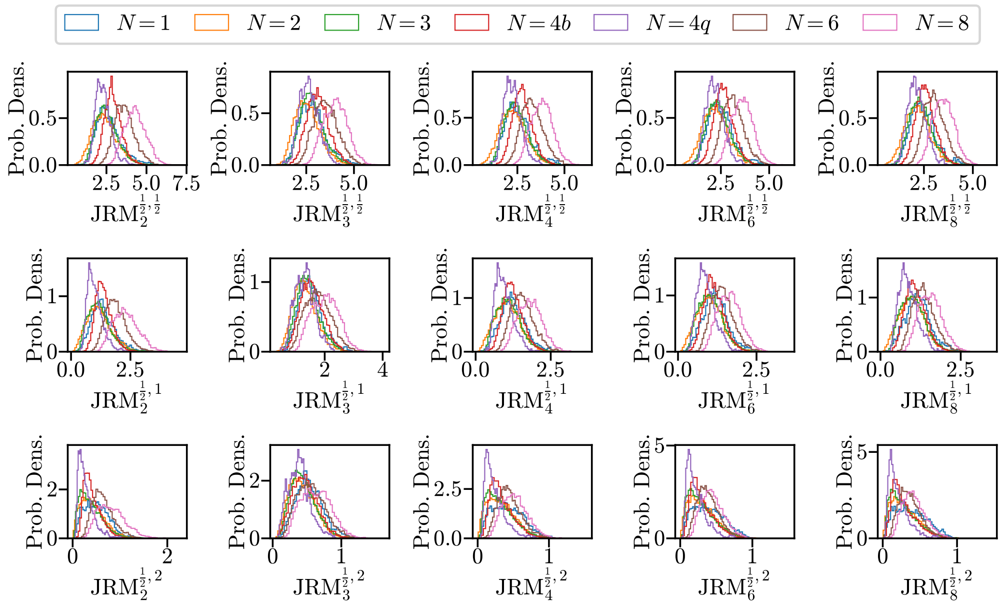


Figure B.2: Distributions of the $\mathrm{JRM}_n^{\kappa,\beta}$ features with $\kappa = \frac{1}{2}$, and the specified n and β parameters. The rest of the distributions are shown in Fig. B.3.

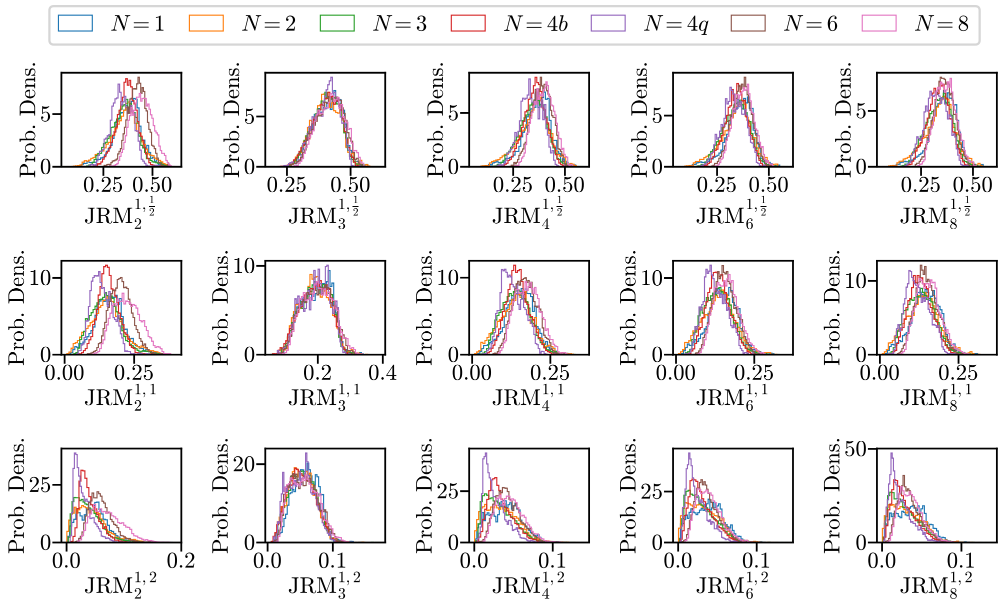


Figure B.3: Distributions of the $\mathrm{JRM}_n^{\kappa,\beta}$ features with $\kappa = 1$, and the specified n and β parameters. The rest of the distributions are shown in Fig. B.2.

Appendix C

Sampling Strategies

C.1 Random Sampling

Figure C.1 shows a sample visualization of the random sampling strategy mentioned in Section 5.6. After every BED iteration, a random observation is added to the training set of the GP models.

C.2 Grid Sampling

Figure C.2 shows a sample visualization of the grid sampling strategy mentioned in Section 5.6. For every BED iteration, observations forming a uniform grids are individually added to the training set of the GP models. The first iteration considers a grid of size 2×2, the second iteration considers a grid of size 3×3, and so on.

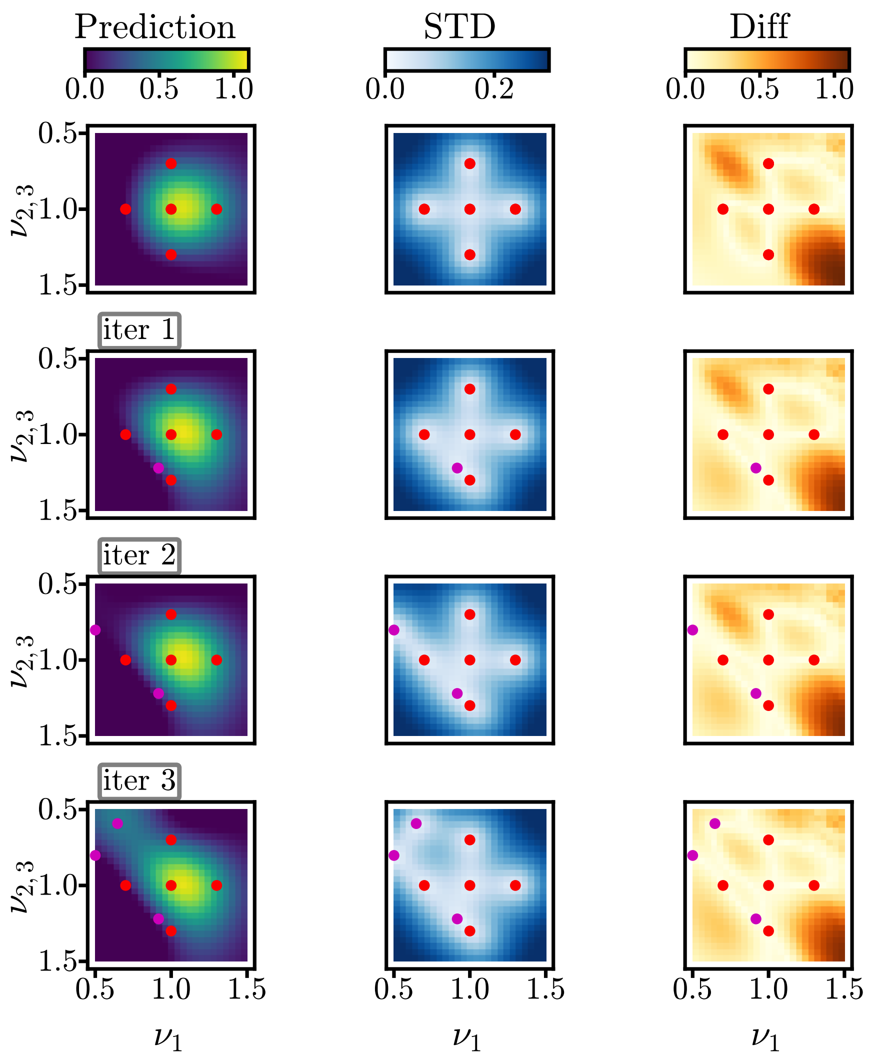


Figure C.1: Prediction (first column), standard deviation (second column), and the absolute difference (third column) between ground truth and prediction by the regular GP model as new observations (purple dots) are added to the training set according to the random sampling strategy.

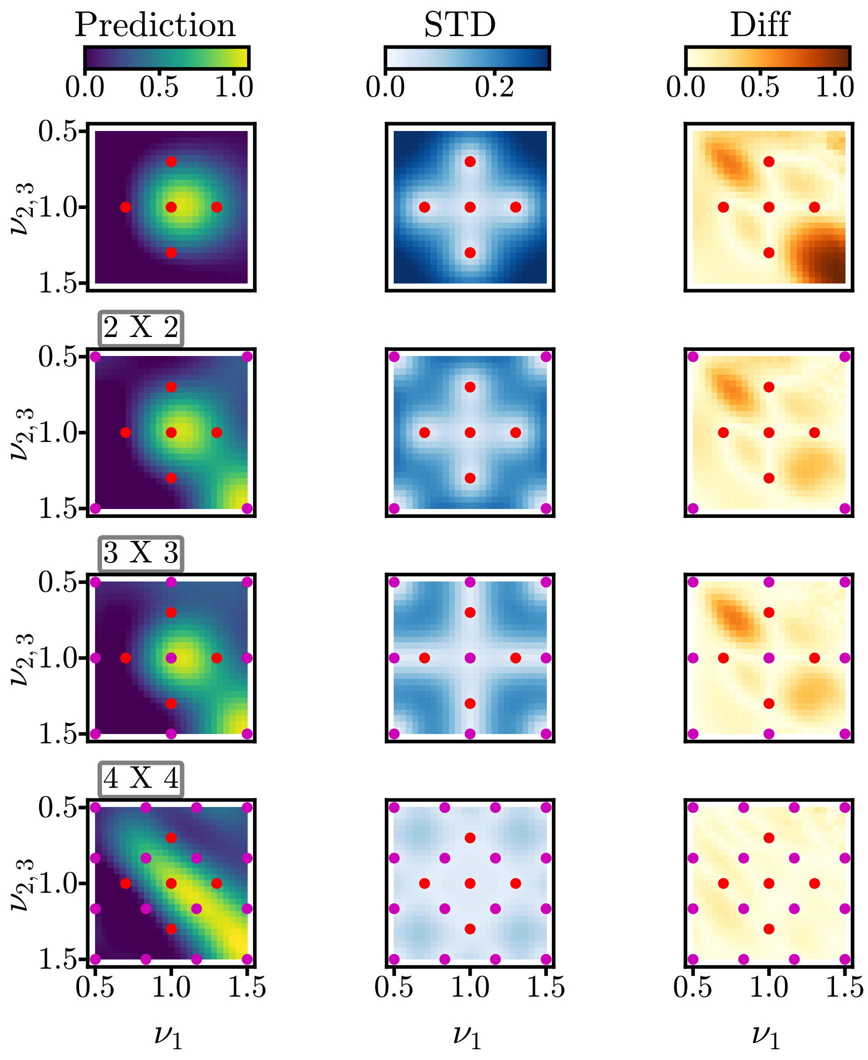


Figure C.2: Prediction (first column), standard deviation (second column), and the absolute difference (third column) between ground truth and prediction by the regular GP model as new observations (purple dots) are added to the training set according to the grid sampling strategy.