

REFINING WEAKLY SUPERVISED ANOMALY DETECTION FOR NEW PHYSICS SEARCHES

Von der Fakultät für Mathematik und Naturwissenschaften der RWTH
Aachen University zur Erlangung des akademischen Grades einer Doktorin
der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Marie Louise Hein, M.Sc.

aus

Frankfurt am Main

Berichter: Prof. Dr. Michael Krämer
Prof. Dr. Martin Erdmann

Tag der mündlichen Prüfung: 17.07.2026

Diese Dissertation ist auf den Internetseiten der Universitätsbibliothek verfügbar.

Refining Weakly Supervised Anomaly Detection for New Physics Searches

© Marie Louise Hein, 2026

GUTACHTER DER DISSERTATION:

Prof. Dr. Michael Krämer
Prof. Dr. Martin Erdmann

ZUSAMMENSETZUNG DER PRÜFUNGSKOMMISSION:

Prof. Dr. Michael Krämer
Prof. Dr. Martin Erdmann
Prof. Dr. Robert Harlander
Prof. Dr. Achim Stahl

VORSITZENDER DER PRÜFUNGSKOMMISSION:

Prof. Dr. Achim Stahl

DATUM DER DOKTORPRÜFUNG:

17.07.2026

DEKAN DER FAKULTÄT FÜR MATHEMATIK UND NATURWISSENSCHAFTEN:

Prof. Dr. Carsten Honerkamp

No CWoLas were harmed in the making of this thesis.

ABSTRACT

Since the discovery of the Higgs boson, numerous searches for physics beyond the Standard Model have been conducted at the Large Hadron Collider, so far without success. This has motivated the development of novel, model-agnostic search strategies. In particular, machine learning has allowed for the development of highly sensitive anomaly detection methods. Among these, weakly supervised anomaly detection has shown great promise as it provides a well-defined, controllable anomaly score by training a classifier on two mixed data sets: a Signal Region (SR), which may contain background and signal, and a background-only Background Template (BT). In this thesis, we focus on improving both the performance and our understanding of weakly supervised anomaly detection. We explore two parts of a weakly supervised search in detail, namely the classifier and how significances are obtained.

First, we replace the Multi-Layer Perceptron (MLP), which was used in early proof-of-concept studies, with a Gradient-Boosted Decision Tree (GBDT). The GBDT performs well for the high-level feature sets typically used in weakly supervised anomaly detection and limits the classifier’s susceptibility to irrelevant features in the classification feature set. Second, we show that optimizing an anomaly detection analysis in a data-driven manner is possible. For this, we explore a number of optimization tasks, namely the selection of the best epochs, hyperparameter optimization, architecture choice, and feature selection. Such data-driven choices are made possible by the introduction of the new ARGOS (Above Random Gain of SIC) metric, which is strongly correlated with the significance improvement of a classifier.

We also introduce a new method to obtain the significance of a potential discovery in a weakly supervised analysis. Previous weakly supervised resonant analyses have relied on fits to the background mass spectrum. This has made them vulnerable to the sculpting of the resonant mass spectrum. Our “direct background estimation” instead simply compares the selected event numbers in the BT and SR after a cut on the anomaly detector, thereby avoiding a fit. We account for systematic biases from imperfect BTs using both simulation and data-driven methods. Lastly, we explore the statistical behavior of anomaly detectors to ensure that significances are calibrated correctly. In particular, we investigate whether a look-elsewhere effect is incurred by anomaly detectors trained to select events from a large phase space. By analyzing miscalibrations of p-values and their effect on sensitivity, we provide practical guidance for future analyses.

ZUSAMMENFASSUNG

Seit der Entdeckung des Higgs-Bosons wurden am Large Hadron Collider zahlreiche Suchen nach Physik jenseits des Standardmodells durchgeführt, bislang jedoch ohne Erfolg. Dies hat die Entwicklung neuartiger, modellunabhängiger Suchstrategien vorangetrieben. Insbesondere durch die Anwendung von maschinellem Lernen wurde die Entwicklung hochsensibler Methoden zur Anomalieerkennung ermöglicht. Besonders die schwach überwachte Anomalieerkennung hat sich als vielversprechend erwiesen, da hierbei ein klar definierter, kontrollierbarer Anomalie-Score ermittelt wird, indem ein Klassifikator auf zwei gemischten Datensätzen trainiert wird: einer Signalregion (SR), die Untergrund und Signal enthalten kann, und einem reinen Untergrunddatensatz, der sogenannten „Background Template“ (BT). In dieser Arbeit konzentrieren wir uns darauf, sowohl die Leistungsfähigkeit als auch das Verständnis der schwach überwachten Anomalieerkennung zu verbessern. Dabei untersuchen wir zwei Aspekte einer solchen Suche im Detail: den Klassifikator selbst und die Bestimmung der Signifikanzen.

Zunächst ersetzen wir das in frühen Proof-of-Concept-Studien verwendete Multi-Layer Perceptron (MLP) durch einen Gradient-Boosted Decision Tree (GBDT). Der GBDT erreicht eine hohe Trennung von Untergrund und Signal auf für die schwach überwachte Anomalieerkennung üblichen High-Level-Feature-Sets und reduziert gleichzeitig die Anfälligkeit des Klassifikators gegenüber irrelevanten Merkmalen im Feature-Set. Darüber hinaus zeigen wir, dass eine datengetriebene Optimierung von Anomalieerkennungsanalysen möglich ist. Dazu untersuchen wir eine Reihe von Optimierungsaufgaben, darunter die Auswahl der besten Epochen, die Hyperparameteroptimierung, die Architekturauswahl und die Feature-Selektion. Solche datengesteuerten Entscheidungen werden durch die Einführung der neuen ARGOS-Metrik (Above Random Gain of SIC) ermöglicht, die stark mit der durch einen Klassifikator erreichten Signifikanzverbesserung korreliert.

Weiterhin stellen wir eine neue Methode zur Bestimmung der Signifikanz einer potenziellen Entdeckung in schwach überwachten Analysen vor. Bisher stützten sich Resonanzsuchen mit Anomalieerkennung auf Fits des Untergrundmassenspektrums, was sie anfällig für Verzerrungen dieses Spektrums macht. Unsere „Direct Background Estimation“ vergleicht stattdessen einfach die Anzahl der ausgewählten Ereignisse in BT und SR nach einem Cut auf den Anomalie-Score, wodurch ein Fit vermieden wird. Systematische Verschiebungen infolge nicht perfekter BTs werden sowohl mittels Simulation als auch mittels datengesteuerter Methoden berücksichtigt. Abschließend untersuchen wir das statistische Verhalten von Anomaliedetektoren, um sicherzustellen, dass die Signifikanzen korrekt kalibriert sind. Insbesondere untersuchen wir, ob bei Anomaliedetektoren, die darauf trainiert sind, Ereignisse aus einem großen Phasenraum auszuwählen, ein Look-elsewhere-Effekt auftritt. Durch die Untersuchung von Fehlkalibrierungen der p-Werte und deren Auswirkungen auf die Sensitivität geben wir Handlungsempfehlungen für zukünftige Analysen.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank Prof. Michael Krämer for giving me the opportunity to pursue my PhD in his group. Over the last three years, I have especially valued the supportive freedom he gave me to explore research topics, collaborations, and outreach activities I was interested in. Throughout, I could always rely on his guidance and support and knew that he would make time whenever I needed advice.

I am also grateful to Prof. Martin Erdmann for agreeing to be my second supervisor. Whenever we met to discuss my progress, I appreciated his genuine interest in my research, outreach work, and personal development.

My PhD would not have been as successful without Dr. Alexander Mück. His guidance throughout both my master's thesis and my PhD has been invaluable. Whether directly involved in a project or not, he was always available for detailed discussions at a moment's notice. These exchanges have had a lasting impact on both my research and this thesis.

I also want to thank Dr. Thorben Finke for his guidance during my master's thesis and at the beginning of my PhD. I still remember how he calmed my nerves before my first progress presentation. Working with him was always fun and insightful. I also want to thank Lukas Lang, whose master's thesis was the first I co-supervised, and he has since become my office mate and collaborator – so it can't have been too terrible an experience for him.

I have been fortunate to work with many engaged collaborators beyond my own institute. In particular, I want to thank Prof. David Shih and Prof. Gregor Kasieczka. Most of my projects were joint efforts with them and their groups, and I greatly benefited from their insights and willingness to engage in detailed discussions. Their interest in my master's thesis research was one of the reasons I chose to pursue a PhD in this field. I am also grateful for their support beyond research – Gregor for his advice on my professional future, and David for many opportunities, including inviting me to spend time at Rutgers. My visit there was a valuable experience, both scientifically and personally.

I would also like to thank my other collaborators. From Hamburg, I thank Dr. Manuel Sommerhalder, Dr. Tobias Quadfasel, and Dr. Louis Moureaux. The collaboration for my first paper originated from a fateful discussion with Manuel at the DPG spring conference in Dresden, to which Tobias was added via Zoom at some point. I also truly appreciate Louis' practical knowledge of what does or doesn't work in experiment, which he was happy to share with me. From Rutgers/Heidelberg, I also want to thank Dr. Ranit Das for making the density estimation work so well that I don't need to worry about it. Many thanks also to Dr. Radha Mastandrea for suggesting we research EFPs together. I am also grateful to Prof. Ben Nachman for working with David and me on figuring out look-everywhere effects. Lastly, for the project that is still ongoing, I want to thank Dr. Philipp Soldin, Vincent Benne, Prof. Christopher Wiebusch and Dr. Humberto Reyes-Gonzalez for their collaboration.

Beyond my collaborators, I have made many friends and “conference buddies” throughout my PhD. I would particularly like to thank Prof. Ramon Winterhalder and Dr. Claudius Krause. Conferences with them involved not only research discussions but also sightseeing,

soccer in various Irish pubs across Europe, and significant complaints about bad slide design. I especially thank Ramon for inviting me to give my first seminar, for providing me with 90% of the gossip I hear about our community, and for putting in a good word for me with my future boss. I also thank the Heidelberg group for adopting me so thoroughly at conferences that I ended up regularly having to correct people that I am not in fact part of the Heidelberg group.

Beyond research, I greatly enjoyed my work on the Physik-Projekt-Tage (PPT). I am especially grateful to Dr. Anna Benecke, whose presentation first introduced me to the project and who was always willing to offer advice. I would also like to thank Prof. Heidrun Heinke for her support and advice. She organized for me to give a presentation of the PPT to the physics department, provided her own know-how, and connected me to her group members, such as Dr. Sebastian Nell and Maria Hinkelmann, who provided theirs. Prof. Malgorzata Worek organized much of the money, worked in the organization, and lent her authority to making the wheels of bureaucracy turn. I am grateful to the entire PPT organization team, both past and present – Manal Alsairafi, Mareike Berkner, Svenja Diekmann, Inke Erichsen, Christine Falter, Benedikt Gurdon, Clara Haustedt, Dr. Lidia Kibkalo, Yannick Klein, Leon Mans, Andreas Nöll, Dr. Magnus Schaaf, Lea Schlickmann, and Prof. Malgorzata Worek. While the work was sometimes demanding, it was almost always enjoyable, even when dealing with administrative challenges. I also thank all project supervisors and especially Florian Mausolf, who supervised the Machine Learning project with me.

I would also like to thank my office mates. The office began as the “High Energy Girls” with Dr. Kathrin Nippel, Dr. Lena Rathmann, and myself. More recently, Lukas Lang and Lars Bündgen have joined the office. I have greatly enjoyed our time together, including the negotiations about heating and ventilation. More broadly, I appreciated the friendly atmosphere in our group and institute. Group meetings and coffee breaks often led to discussions that were not always productive but consistently fun and informative. I am grateful to everyone who contributed to this atmosphere, as well as for the support I received during my science slam. I also thank my friends across the physics department, both longstanding and newly made during my PhD. In particular, I thank the “Exis”, Markus Bachlechner, Dr. Thilo Birkenfeld, Elisabeth Neuerburg, and Dr. Philipp Soldin, for always welcoming me when I needed a break and for their help with machine learning questions. I would also like to thank Svenja Diekmann for very fun “Look, I made a cool looking graphic” sessions.

I am grateful to everyone who helped proofread this thesis, especially Alexander, who read it in full, as well as Philipp, Lukas, Magnus, Kira Laumann, and Mark Fanselow for their careful reading of individual parts.

Last but very definitely not least, I want to deeply thank my friends and family. Throughout my PhD, they listened to many complaints, especially during the writing phase. Nevertheless, their support has never wavered. I am particularly grateful to my parents, who have been a constant pillar throughout my studies and my PhD – their support means the world to me. I look forward to celebrating with them once this is all done.

Computations were performed with computing resources granted by RWTH Aachen University under project rwth0934. Throughout this thesis, ChatGPT was used to help debug and write Python and TikZ code. It was used as a conversation partner to find and understand sources as well as to reflect on and discuss small parts of the text in this thesis. ChatGPT was also used to obtain an initial translation of the abstract. Grammarly was used to find grammar and spelling mistakes at the end of the writing process.

PUBLICATIONS

During my PhD studies, I co-authored the following publications, the first four of which are the basis of this thesis:

- [1] T. Finke et al., “Back To The Roots: Tree-Based Algorithms for Weakly Supervised Anomaly Detection,” *Phys. Rev. D* **109**, 034033 (2023), [arXiv:2309.13111 \[hep-ph\]](#)
Contributions: Produced all results in the body of the paper, some contribution to writing.
- [2] R. Das et al., “Accurate and robust methods for direct background estimation in resonant anomaly detection,” *Phys. Rev. D* **111**, 094041 (2025), [arXiv:2411.00085 \[hep-ph\]](#)
Contributions: Produced all results in the paper, contribution to writing.
- [3] M. Hein et al., “How to pick the best anomaly detector?” (2025), [arXiv:2511.14832 \[hep-ph\]](#)
Contributions: Produced all results in the paper, significant contribution to writing.
- [4] M. Hein, B. Nachman, and D. Shih, “Look everywhere effects in anomaly detection,” (2025), [arXiv:2512.13787 \[hep-ph\]](#)
Contributions: Produced all results in the paper, significant contribution to writing.
- [5] R. Das et al., “Kitchen Sink Anomaly Detection,” (2026), [arXiv:2604.20965 \[hep-ph\]](#)
Contributions: Co-supervised master’s thesis, in which this project started, checked reproducibility of results.

CONTENTS

1	Introduction	1
2	Jets as a Window to New Physics	5
2.1	From Fundamental Interactions to Measurements	5
2.2	Defining and Understanding Jets	11
2.2.1	Jet Clustering	11
2.2.2	Jet Observables	13
2.3	Jets and the Search for New Physics	15
3	Fundamentals of Machine Learning	17
3.1	Machine Learning as a Functional Fit	17
3.2	The Neyman-Pearson Lemma and Classification Tasks	19
3.3	Classical Machine Learning and Decision Trees	22
3.3.1	Adaptive Boosting	25
3.3.2	Gradient Boosting	26
3.4	Deep Learning and Neural Networks	27
3.5	Density Estimation and Generative Models	35
3.5.1	Normalizing Flows	37
3.5.2	Continuous Normalizing Flows and Flow Matching	39
3.5.3	Conditional Density Estimation	41
4	Weak Supervision for New Physics Searches	43
4.1	From Standard Resonance Searches to Anomaly Detection	43
4.1.1	The Fundamental Principle of Resonance Searches	43
4.1.2	The Range of Anomaly Detection Methods	45
4.2	Weakly Supervised Anomaly Detection	46
4.2.1	Background Template Generation Methods	47
4.3	Design Choices for Weakly Supervised Searches	49
4.3.1	Data Set Selection	50
4.3.2	Feature Choice	51
4.3.3	Background Template Generation	53
4.3.4	Classifier Training	53
4.3.5	Performance and Significance Evaluation	54
5	Improving the Classifier	57
5.1	Tabular Data and Boosted Decision Trees	57
5.1.1	Setup	58
5.1.2	Uninformative Features, Rotational Invariance and Ensembling	61
5.1.3	Extended Feature Sets and Model Agnosticism	64
5.1.4	Varying Signal Amounts and Data Set Size	66
5.1.5	Realistic Background Templates for Extended Feature Sets	69
5.1.6	Conclusion and Outlook	70
5.2	Selecting the Optimal Anomaly Detector	72
5.2.1	Metrics for Optimization	72

5.2.2	Setup	75
5.2.3	NN Epoch Selection	77
5.2.4	Hyperparameter Optimization	78
5.2.5	Architecture Selection	83
5.2.6	Feature Selection	86
5.2.7	False Discoveries and Background Sculpting	89
5.2.8	Conclusion and Outlook	91
6	Obtaining and Calibrating the Significance	93
6.1	Significance Extraction by Direct Background Estimation	93
6.1.1	Setup	94
6.1.2	Direct Background Estimation	96
6.1.3	IAD	98
6.1.4	CWoLa Hunting	100
6.1.5	CATHODE	103
6.1.6	The Impact of Signal Randomization	107
6.1.7	CWoLa Hunting for Extended Feature Sets	107
6.1.8	Studies of δ_{sys} Using Independent Simulations	108
6.1.9	Conclusion and Outlook	111
6.2	The Look-Everywhere Effect	112
6.2.1	Setup	113
6.2.2	Comparing Analysis Strategies	116
6.2.3	Look-Everywhere Effects and Overfitting	120
6.2.4	The Impact of the Look-Everywhere Effect on Signal Sensitivity . . .	122
6.2.5	Conclusion and Outlook	127
7	Conclusion	129
A	Setup Specifications	133
A.1	Density Estimation Setup	133
A.2	χ^2 Metric for Background Sculpting	134
B	Additional Results for Direct Background Estimation	135
B.1	δ_{sys} Values Used in the Analysis	135
B.2	Window-Wise Estimation of δ_{sys}	137
B.3	Significances When Using $\sigma_{\text{sys}} = 0$	137
B.4	Estimate of δ_{sys} With Signal for CATHODE	139
B.5	Significances for CATHODE When Using the MC Estimate	139
	Acronyms	141
	Bibliography	145

1

INTRODUCTION

In the course of the last fourteen years since the Higgs discovery [6, 7], our understanding of the Standard Model (SM) of particle physics has been strengthened by precision measurements and tests. However, while the SM has been confirmed with unprecedented accuracy [8], there are a number of open questions concerning both its structure and completeness, which remain unanswered. For example, we have thus been unable to pin down both the nature of Dark Matter, which makes up 85% of the matter in the universe, and the nature of neutrino masses [9]. For both of these and many other open questions, a number of different extensions to the Standard Model have been proposed, which could resolve them. However, any searches for corresponding signatures have not found any significant deviations from the null hypothesis. For once favored models such as Supersymmetry (SUSY), this has led to the exclusion of much of the viable parameter space [9].

The lack of success in our searches for new physics can have different explanations. Firstly, it is possible that physics Beyond the Standard Model (BSM) is realized in such a way that no signatures measurable at our current particle colliders are produced. This may include BSM physics occurring at higher energy scales or, as is possible for dark matter, not coupling to Standard Model particles. While nothing can be done if BSM physics does not couple to the SM, future colliders can provide higher energies, giving us access to new energy scales in our searches. Secondly, we may not have sufficient statistics to detect new physics if it is very rare, i.e., is produced at very low cross sections. Thirdly, with the large possible model space, we may thus far not have searched for new physics in the correct regions of model space. Both of the last two problems may be addressed with the help of new search approaches designed to complement our standard approaches, for which we typically observe a clear trade-off between sensitivity and coverage. Additionally, statistics are improved as more data is taken, and with the coming HL-LHC, a 10-fold increase in the data taken with the pre-upgrade LHC is expected [10].

Fundamentally, all new physics searches live on the spectrum between model-specific and model-agnostic. As mentioned above, with standard approaches, this spectrum also represents a trade-off between the statistics required to detect a model and the size of the scanned region of model space: A model-specific search aims to detect a particular new physics model and is therefore only sensitive to a small region of model space. However, in this region, its optimized sensitivity is typically the best possible for this model. A model-agnostic search, on the other hand, can detect a broad range of new physics models but requires more statistics to detect a particular model than a model-specific search would. We consider a bump hunt as an example: A bump hunt searches for the resonant production of a new heavy particle – a typical signature of many new physics models, which produces a bump on top of a smooth spectrum of background events. This is often done by fitting the smooth spectrum away from the resonance and then characterizing the overabundance of events caused by the resonance. If no additional cuts are applied – this is also called an inclusive search –, this search is sensitive to a broad range of models, which produce resonances. Such a search can be made more sensitive and simultaneously more model-specific by applying cuts to target particular features

of the specific final state. In addition to reducing the range of models to which the search is sensitive, this more model-specific approach has an additional disadvantage. Typically, model-specific cuts are derived with the help of simulation and, as such, are vulnerable to data-MC mismodeling. This mismodeling could help hide new physics (see, e.g., how PDF fits may hide new physics in Ref. [11]), particularly as simulations are typically tuned to reproduce the data [12].

Machine Learning (ML) has provided a new route to make relatively model-agnostic search methods such as bump hunts more sensitive without adding many additional signal model assumptions. In general, ML has revolutionized many processes in particle physics in the last decade. Deep Learning in particular performs well when data sets are large and high-dimensional, as is typically the case at the LHC. In particular, for jets, which are not only one of the most abundant but also one of the most complex signatures at the LHC, recent developments using ML have allowed us to make much more efficient use of the information they provide. Improvements in jet tagging (see, e.g., Ref. [13]), for example, have led to analyses being able to much more efficiently select data corresponding to a particular final state. Anomaly Detection (AD) relies on ML to automatically identify “anomalous” events in typically high-dimensional feature space, allowing us to enhance the sensitivity of analyses without explicitly specifying any particular BSM signature. Defining what is regarded as anomalous is a key part of designing an anomaly detector and crucially affects the coverage and sensitivity of the anomaly detector. For example, outlier anomaly detection typically defines anomalous events as particularly rare (see, e.g., Ref. [14]), while semi-supervised methods typically use simulated signal templates to more explicitly define signatures expected of new physics models (see, e.g., Ref. [15]). In this thesis, we use weakly supervised methods following the Classification Without Labels (CWoLa) [16] approach, which makes use of physics assumptions to formulate the anomaly detection task as a classification task between mixed samples. For resonance searches, weakly supervised anomaly detection typically relies on the division into the Signal Region (SR), where the signal is localized, and the background-dominated Sidebands (SBs) around it. Using the SBs, a model of the Signal Region (SR) background, a so-called Background Template (BT), is derived. A classification is then performed between the BT and the SR data. For a high-quality background template, the only difference between the two data sets is the signal. As such, if signal is present, a cut on the classification score enhances the signal and, therefore, the sensitivity of the resonance search. There are several distinct advantages to weak supervision when compared with other anomaly detection methods. Firstly, compared with outlier anomaly detection, weak supervision produces a more well-controlled anomaly score, with the background-only case producing a random classification as long as the BT is accurate. Secondly, compared with semi-supervised anomaly detection, weak supervision places fewer assumptions on the signal and, as such, typically provides a better coverage of new physics models.

To conduct a weakly supervised analysis, one must first select a final state and classification features in which to search. Then a background template must be generated for this feature set and a classifier trained. Lastly, the significance of the analysis is evaluated. Over the last few years, there have been many developments in each of these steps, which we quickly summarize here and describe in more detail in Sec. 4.3. Most of the initial studies of weak supervision were conducted on data sets from the LHC Olympics 2020 [17], a community challenge for anomaly detection. Particularly the R&D data set [18], which includes a BSM dijet resonance in QCD dijet background, has provided an active testing ground, which is also used in this thesis. Recent work has aimed to push to other final states [19] and other signatures, for example with tail-based searches [20], mono-jet searches [21] or 2D resonances [22]. Additionally, Ref. [5]

has extended the R&D data set by additional signal models to better test how model-agnostic an anomaly detector is. The model-agnosticism of a weakly supervised search also strongly depends on the classification features used. Early proof-of-concept studies largely limited themselves to small high-level feature sets using only four features [17]. Moving to larger high-level feature sets initially failed because the MLP classifiers used in early studies were highly sensitive to the addition of any uninformative features. In this thesis, we resolve this issue with the use of Boosted Decision Tree (BDT) classifiers, which perform particularly well on high-level, tabular feature sets and are largely able to ignore uninformative features [1] (also studied in Ref. [23]). BDTs have enabled the use of much larger high-level feature sets, both with subjettnesses [24, 25], which we study in this thesis, and other feature types studied in, e.g., Refs. [5, 26]. In recent years, there have also been repeated attempts to move to low-level jet constituent information for weak supervision to improve model-agnosticism, which have thus far typically fallen short of the sensitivity achieved with high-level features [27–30]. Most studies of classifier improvements are conducted on a limited number of simulated benchmark signals – often the LHC R&D set alone. To reduce the dependence of classifier setup decisions on simulation, we study data-driven optimization methods in this thesis. By introducing a metric highly correlated with the significance improvement of a classifier but calculated in a fully data-driven manner, we can select the optimal setup for the given data [3]. The classifier’s signal sensitivity also relies crucially on the background template, and many methods have been designed to obtain accurate background templates over the last decade [31–43]. In recent years, developments have particularly focused on reducing computational cost [41–43] and improving the fidelity of background templates [27, 28, 41]. Lastly, to assess the significance of a search, a fit is typically performed, as done in analyses by ATLAS [44, 45], CMS [46], and on CMS open data [47]. Here, a fundamental issue has been background sculpting. Background sculpting of the resonant invariant mass spectrum is caused by the AD event selection due to correlations between the anomaly score and the invariant mass. A method to address this has been introduced in Ref. [38], which creates a background template in such a way that no correlations between classification features and invariant mass remain. In this thesis, we propose “direct background estimation” to eliminate background sculpting entirely. For this, we compare the event numbers selected by a cut on the anomaly score in SR and BT using a cut and count analysis [2]. Classical analyses searching for a signal across multiple possible regions of phase space (e.g., through binning) incur miscalibrations of p-values – called the look-elsewhere effect – due to multiple testing. We investigate whether similar statistical miscalibrations are incurred by anomaly detectors, which also optimize the phase space selections [4].

In general, this thesis is structured as follows: We start with the physics foundation of this thesis in Chapter 2, where we explain how jets are produced and measured in colliders, how they are defined, and how new physics may be hiding in them. In Chapter 3, we turn to the ML foundation, on which much of this thesis and particularly the classification improvements rely. Afterward, we introduce resonance searches, weak supervision, the design of anomaly detection-based analyses, as well as our data and our performance metrics in Chapter 4. After laying the foundation in this way, we turn to the results on improvements of the classifier in Chapter 5 and results regarding the statistical analysis of results in Chapter 6. Lastly, we summarize and conclude in Chapter 7.

With the current state-of-the-art particle accelerator, the LHC, being a hadron collider, jets are the most abundant signature by which we observe the Standard Model (SM). Therefore, they are also abundant in our searches for physics Beyond the Standard Model (BSM). In this chapter, we explore the fundamentals of collider physics, both theoretical and experimental, which underlie the production and measurement of jets. We define and characterize jets and discuss how physics beyond the Standard Model may be realized in jets.

2.1 FROM FUNDAMENTAL INTERACTIONS TO MEASUREMENTS

This section is based on Refs. [9, 48, 49]. Additional sources are included where appropriate.

The Standard Model of particle physics provides a predictive description of physics at the particle level. It describes all seventeen elementary particles discovered thus far and their interactions. Its predictions have been confirmed with up to parts-per-trillion precision [8]. The SM relies on the gauge symmetries realized by the group structure

$$U(1) \times SU(2) \times SU(3). \quad (2.1)$$

Fundamental interactions are described by the SM Lagrangian $\mathcal{L}_{\text{SM}}$, based on which the kinematics and rates of different particle interactions can be calculated. Taking a look at the particle content of the Standard Model, shown in Fig. 2.1, we see two groups of particles, fermions and bosons, with two subgroups each. For the bosons, we have the gauge vector bosons – gluon, photon, W and Z boson – which mediate the gauge interactions, and the scalar Higgs boson, which gives the other SM particles their masses. On the fermionic side, there are the quarks and the leptons. Particularly key to us are the quarks and gluons, as collisions at the LHC are proton collisions¹ and the proton is made up of quarks and gluons, in this context also referred to as *partons*². With quarks and gluons interacting via the strong interaction, Quantum Chromodynamics (QCD) is particularly relevant for understanding a majority of collisions at the LHC. QCD is based on an $SU(3)$ gauge symmetry and mediates interactions of the *color* charge, which is only carried by quarks and gluons in the SM. However, as quarks interact with all of the bosons within the Standard Model and all bosons except for the gluon also interact with leptons, these proton collisions indeed give us access to and therefore let us test the entire Standard Model.

Particle colliders have been one of our key tools to first build and later test the Standard Model since the first cyclotrons in the 1930s. While other experiments, particularly neutrino experiments, test specific parts of the Standard Model, particle colliders offer a high degree of control, a large range of measurements, high energies (at the LHC up to 13 TeV), and

¹ Additionally, lead ion collisions are also performed at the LHC. The ALICE detector was designed specifically to measure these heavy ion collisions.

² Technically, the photon can also be a parton.

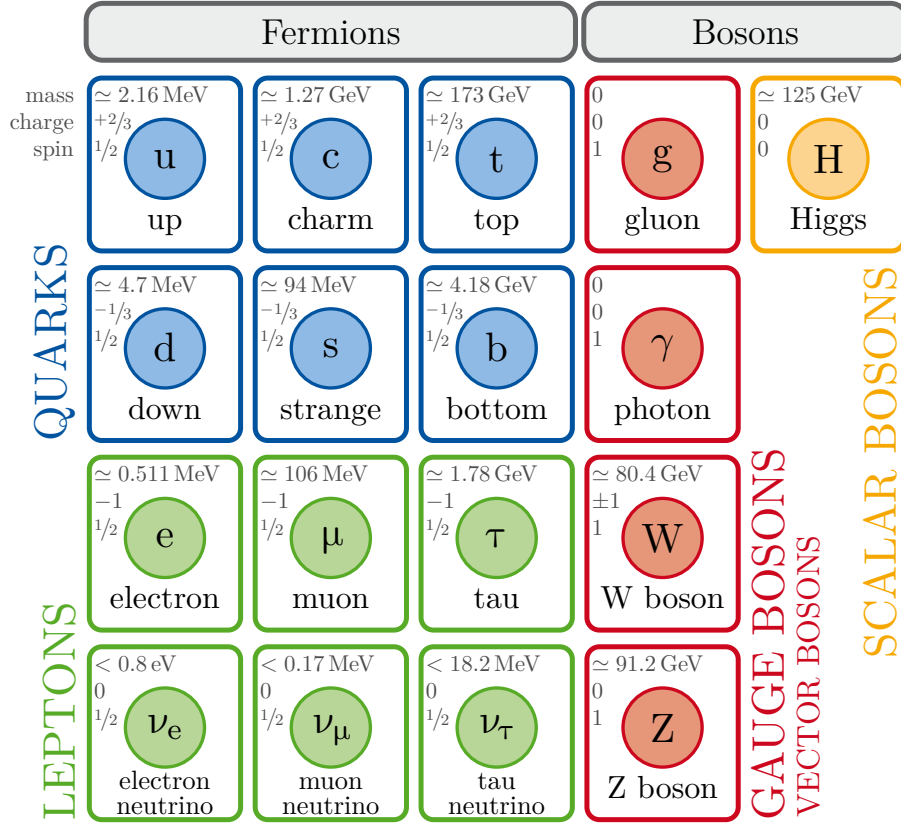


Figure 2.1: Particles of the Standard Model. Figure adapted from Ref. [50].

high event rates. Using colliders, we can test the SM in a few different ways: First of all, we can make predictions based on the SM and test whether they are consistent with our measurements. For this, we must measure the 19 – more if neutrino mass terms are included – parameters of the SM in some processes to test consistency with others. Additionally, we can search for processes, such as the production of new particles, which are not predicted by the Standard Model. Even if new physics is not found in such a search, it will set limits on parameters of different BSM models. Fundamentally, both of these types of SM tests rely crucially on our knowledge of each part of the collision and data taking: By understanding the entire process, namely the acceleration, collision, parton showering and hadronization in the case of quarks and gluons, how particles are measured in the detector and lastly how we select and calibrate events, we can extract knowledge about the underlying theory from the data we measure. We walk through this process, shown in Fig. 2.2, step by step here, to understand the fundamental considerations – both theoretical and experimental – behind our measurements.

The first step is the injection of particles into the collider and their acceleration to the beam energy E_{beam} . There are a few criteria for which particles to inject. They should be electrically charged to allow for acceleration and manipulation via electric and magnetic fields. They should generally be readily and abundantly available, and ideally, be stable or at least have a long lifetime. There are only a few particles, which meet these criteria, namely electrons and their antiparticles positrons on the leptonic side³ and (anti-)protons on the hadronic

³ Muon colliders have also been actively discussed, particularly in the US American particle physics community in recent years. They allow for higher center of mass energies and higher Higgs coupling cross sections than electrons can provide [51].

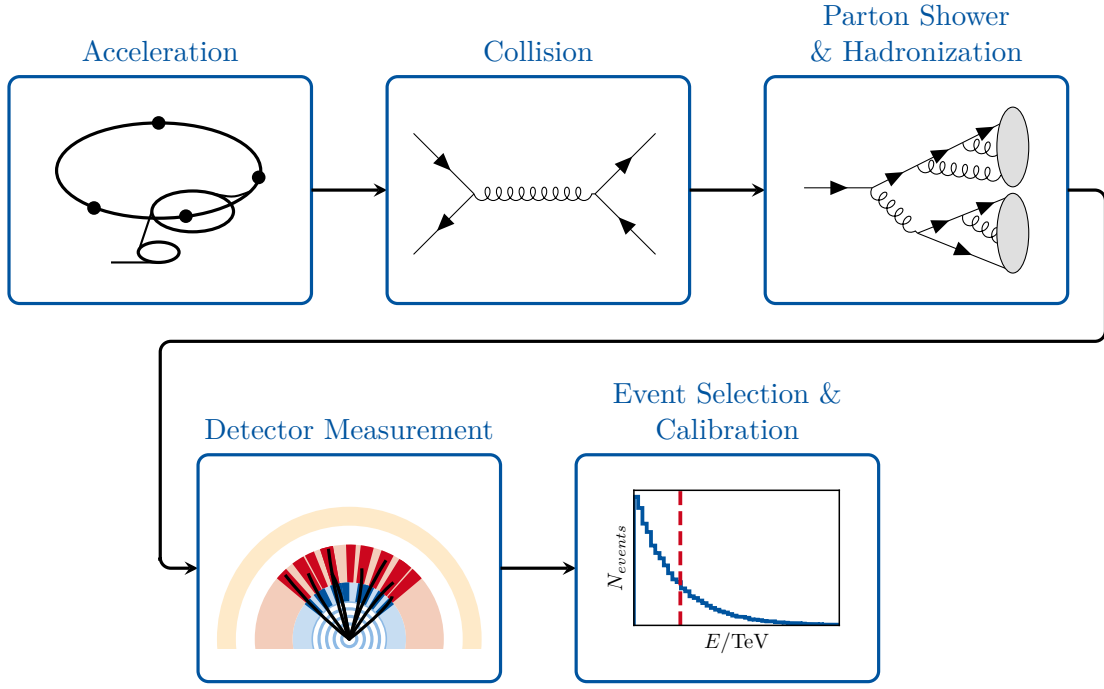


Figure 2.2: Schematic diagram of particle collision processes

side. Injecting isolated quarks is not possible as free quarks do not appear in nature (see confinement, below). Both electrons and protons have particular advantages: Electrons provide cleaner collisions as they themselves are elementary particles, while protons can probe higher energies due to a smaller synchrotron radiation loss. At CERN, this complementarity has been exploited by reusing the LEP collider tunnel for the LHC to reach higher energies. Future projects such as FCC-ee and FCC-hh explicitly propose such a staged approach [52]. For the high energies reached in the LHC, the acceleration must be done in multiple steps (as seen in the “Acceleration” box of Fig. 2.2), which partly utilize the accelerators of older CERN experiments. After multiple such steps through a series of linear and circular colliders, beams are finally injected into the LHC beam pipe, where energies reach up to 6.8 TeV. Particles are injected into the LHC in *bunches* of $n_{\text{bunch}} \approx 10^{11}$ particles with a bunch separation of 25 ns.

Once two beams moving in opposite directions are in the beam pipe, they can be brought to collision at specific collision points at the center of each detector by focusing the beam using quadrupole magnetic fields. Collision rates are proportional to the luminosity

$$\mathcal{L} = f \frac{n_1 \cdot n_2}{4\pi\sigma_x\sigma_y}, \quad (2.2)$$

with the frequency of bunch crossings $f = 1/(25 \text{ ns}) = 40 \text{ MHz}$ and the numbers of particles per bunch per beam $n_{1/2}$ as well as the horizontal and vertical beam size $\sigma_{x/y}$, which are influenced for example by the focusing magnets. Collisions then occur at center of mass energies E_{CM} determined by the beam energy as

$$s = E_{\text{CM}}^2 = (p_1 + p_2)^2 = 4E_{\text{beam}}^2. \quad (2.3)$$

As the luminosity increases, the likelihood of multiple collisions occurring simultaneously also increases. This is referred to as *pile-up*. For the LHC, the average number of interactions per bunch crossing is 60. With the coming HL-LHC, an increase to 200 is expected. Most of

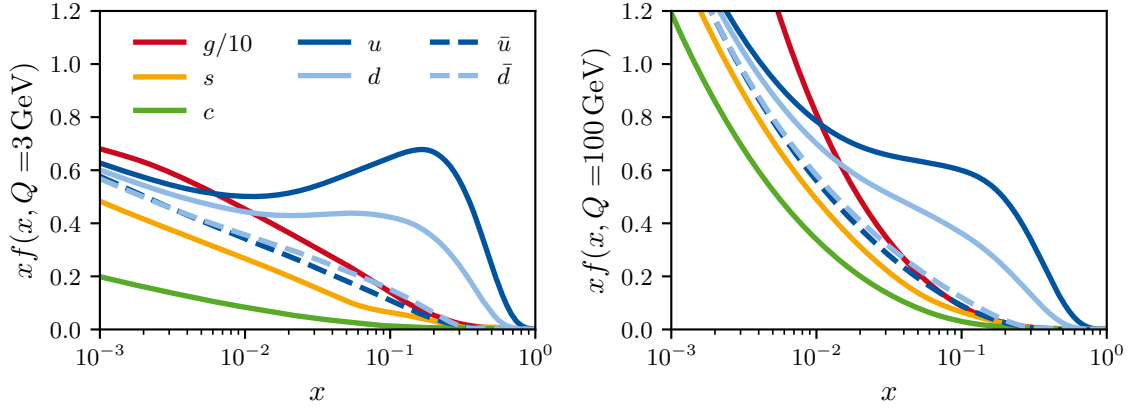


Figure 2.3: Proton Parton Distribution Functions at two different factorization scales Q . Fits are from NNPDF3.1 [53] with data obtained from Ref. [54].

these simultaneous collisions are relatively low-energy scatterings (so-called soft processes), which are not particularly interesting.

We can consider any collision on several different length scales. On large length scales, we collide protons and measure colorless final states, either produced directly in the hard interaction (e.g., leptons and photons) or via hadronization (hadrons). However, if we consider the same process at very small length scales, the actual interaction occurs at the parton level in the hard scattering. Typically, to make predictions, we assume factorization, separating long-distance physics from short-distance interactions, justified by the hierarchy of energy scales. As such, we separate the proton’s substructure in the initial state, the hard scattering, and the subsequent parton showering and hadronization of the final state.

We start considering these factorized components with the proton’s substructure as described by Parton Distribution Functions (PDFs). Generally, the PDF $f_i(x, \mu_F)$ describes the likelihood of a parton of flavor i interacting as a function of the momentum fraction of the parton $x = p_{\text{parton}}/p_{\text{proton}}$ and the factorization scale μ_F , which is often set to the energy of the hard process Q . As PDFs describe non-perturbative QCD at low energy scales, they cannot be calculated with perturbation theory. Calculations from lattice QCD have thus far not reached the required accuracy for current experimental and theoretical precision. Therefore, PDF fits performed on past collider data are used instead. An example of such a fit for two different energy scales Q can be seen in Fig. 2.3. The up and down *valence* quarks dominate at moderate momentum fractions x , reflecting that they carry a significant portion of the proton’s momentum. However, the distributions are broad, and gluons as well as *sea* quarks contribute significantly, particularly at low x . Heavy quark (bottom and top) contributions are suppressed by their larger masses.

The partons interact with one another in the *hard scattering*, which can be described in fixed-order perturbation theory. The most common process is a scattering of two quarks or gluons (as shown in the “Collision” box in Fig. 2.2), but more complex processes occur as well, though at lower rates, such as gluon fusion Higgs production. The process rates are determined by the matrix element $\mathcal{M}_{fi}$ (i referring to the initial, f to the final state)

of a process, which itself can be calculated using the Feynman rules derived from the SM Lagrangian. Using this matrix element and the parton distribution functions, the cross section

$$\sigma = \sum_{a,b} \int_0^1 dx_a dx_b \int d\Phi_f f_a(x_a, \mu_F) f_b(x_b, \mu_F) \frac{1}{2x_a x_b s} |\mathcal{M}_{ab \rightarrow f}|^2(\Phi_f; \mu_F, \mu_R) \quad (2.4)$$

can be determined, where we integrate over the final state phase space Φ_f and both partons' momentum fractions $x_{a/b}$. We also consider all possible initial states by summing over all partons a and b , which can interact to produce the final state f . μ_R is the renormalization scale, which we do not discuss here. The number of events

$$N = \sigma \int \mathcal{L}(t) dt, \quad (2.5)$$

which we expect to measure for this process, is proportional to the cross section and the integrated luminosity.

Depending on the particles in the final state, what we measure in the detector may not be the final state of the hard process. For example, a particle may have a short lifetime, like the τ lepton, which decays hadronically or leptonically. The relative rate of different decays is given by the branching fraction $\mathcal{B}$. In addition to particles decaying before we can measure them, we observe quarks and gluons not as individual particles but as jets because of parton showering and hadronization. A final state parton with a high energy Q_0 first undergoes successive probabilistic QCD-governed splittings, i.e., $q \rightarrow qg$, $g \rightarrow q\bar{q}$, and $g \rightarrow gg$. In all likelihood, such splittings are low energy (soft) and collinear. A collimated spray of partons is observed. Successively, these splittings reduce the energy per parton until energy scales are reached, at which perturbation theory is no longer applicable (approximately 1 GeV). The non-perturbative processes below these energy scales are described by hadronization: Fundamentally, we do not observe any free colored particles. This property of QCD is called *confinement* and believed (no analytic proof exists) to be caused by the gluon-gluon self-interaction. Because of this self-interaction, two quarks being separated experience an approximately constant force drawing them together (in contrast to the force decreasing with $1/r$ for QED). In practice, this means that any colored particles are observed only as parts of colorless hadrons. There are a number of different models that describe the hadronization process. We discuss a string model [55], as shown in Fig. 2.4 for a $q\bar{q}$ final state: (i) The two quarks separate at high velocities. (ii) The energy in the color field between them increases linearly with their distance, and a color field tube with an energy density of approximately 1 GeV/fm forms. (iii) As separation and energy increase, the energy in the color field becomes sufficient for the production of an additional $q\bar{q}$ pair to be energetically favorable. (iv) This process repeats until parton energies reach scales of approximately 1 GeV, at which point (v) hadronization occurs, where quarks and gluons become bound into colorless mesons and baryons. As the initial particles are boosted, there are now two groups of hadrons boosted in the respective quark's original direction, forming a jet of particles, as can also be seen in the “Parton Showering & Hadronization” box of Fig. 2.2.

Since we now understand how the final particles originate from a collision, the next step is the detector measurement. For this, we consider a simplified and generalized detector model with a magnetic field comprising a tracking detector, Electromagnetic Calorimeter (ECAL), Hadronic Calorimeter (HCAL), and a muon chamber. This structure describes both the CMS and ATLAS detectors relatively well. The tracking detector can measure the flight path of charged particles. Both calorimeters measure the deposited energy of particles. Their energy measurement is only accurate if the full particle energy is deposited within the detector.

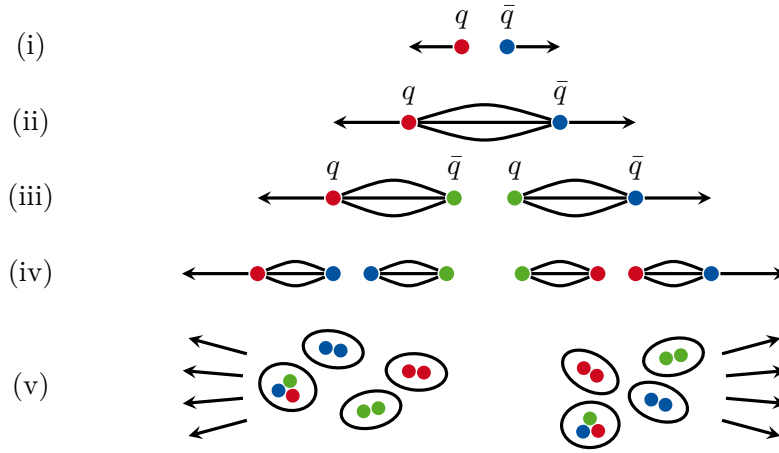


Figure 2.4: Visualization of hadronization according to the string model. Anti-quarks carry anti-color, which here is represented by the same color (i.e., anti-quarks colored green are anti-green). Recreated from Ref. [48].

Therefore, it is important for the calorimeter to be large enough. The mechanism by which energy is deposited in the calorimeters may be different, the electromagnetic calorimeter measures the energy of particles, which interact electromagnetically, i.e., photons and charged particles, while energy depositions in the hadronic calorimeter occur due to the strong interaction. The muon chambers are designed specifically to identify and measure the energy of muons. Because of their higher masses compared to electrons, they are typically not fully stopped in the ECAL. Since they do not interact much with the HCAL, they can then be relatively cleanly identified as they are typically the only particle type reaching the muon chamber. We can see in the left panel of Fig. 2.5 how the different detector components help us identify the type, charge, and energy of a particle. Neutrinos have low cross sections and escape the detector. In collider detectors, they are only measured as missing energy when considering momentum conservation. The right panel of Fig. 2.5 shows how a jet made up of numerous charged and neutral hadrons shows up in a detector. Importantly, we cannot separate all individual hadrons in each detector part as energy deposits from different hadrons may occur in neighboring or even the same detector cells.

With the high frequency of bunch crossings and the large volume of data measured in the detector, it is not possible to save every collision. Therefore, multiple triggering steps, which, for example, may place simple cuts on the minimum energy recorded in an event (see “Event Selection & Calibration” box of Fig. 2.2), reduce the data amount to a level that can be saved on disk. Before analysis, measured events also need to be calibrated. While the detector is well known fundamentally, dead or radiation-worn detector cells may cause miscalibrated measurements. By considering data from well-known processes, calibration curves can be fitted to obtain more accurate and less biased measurements.

By understanding and considering all of these steps, we can make predictions for our measurement data. Often these predictions are produced by Monte Carlo (MC) *simulations*, which follow the same general steps: Assuming the known collision energies and luminosity, we can simulate the collisions based on matrix element calculations and PDF fits using, e.g., **MadGraph** [56], simulate parton showering and hadronization with, e.g., **Pythia**⁴ [57] or

⁴ Note that simple hard processes, which are commonly used, may also be programmed into **Pythia** directly, allowing us to skip **MadGraph** entirely for some simulations.

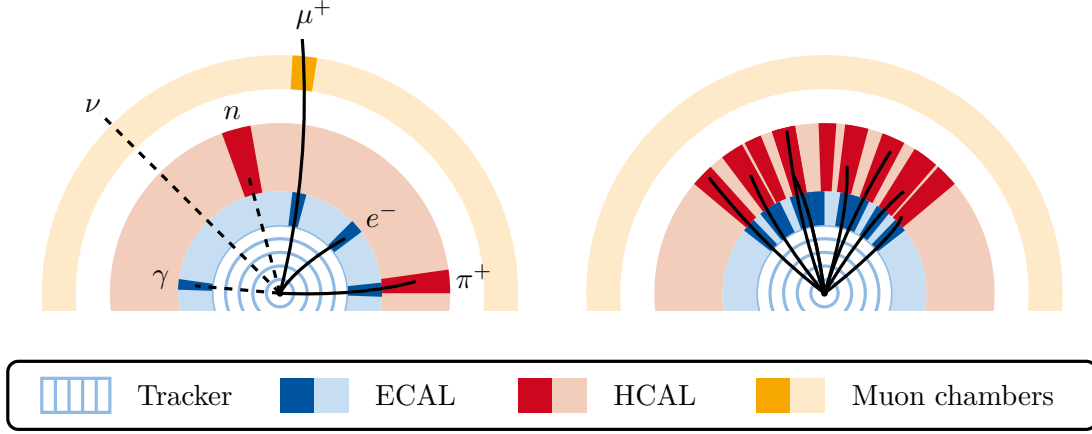


Figure 2.5: Schematic diagram of how different particles are measured in typical modern detectors. Deposited energy in the calorimeter is shown as a darker color in the detector. Left: Typical interactions of particles in a detector. Charged particles, which can be measured by the tracker, are shown as solid lines with curves caused by the magnetic field denoting charge. Neutral particles are shown as dashed lines and are not measured in the tracking detector. Right: Measurement of a jet in a typical detector. Only charged particle tracks are shown. Neutral particles are only displayed as deposited energy. Figure recreated from Ref. [48].

Herwig [58, 59] and simulate detector measurements either via the general effects of the detector in Delphes [60] or via single particle tracking with higher fidelity but slow computation speeds in GEANT4 [61–63]. Such pipelines allow us not only to make more nuanced predictions but also to test methodologies, as is done in this thesis, on realistic but fully understood and labeled data.

2.2 DEFINING AND UNDERSTANDING JETS

This section is largely based on Refs. [49, 64, 65]. Refs. [49, 64] consider jets from an experimental and Ref. [65] from a theoretical viewpoint. The original literature and additional sources are also cited where appropriate.

As we have seen in the previous section, jets are not only very common but also highly complex objects, for which signatures of multiple particles may overlap strongly in the detector measurement, e.g., for decays of highly boosted objects into quarks. It is therefore key for our understanding of LHC physics to be able to define what we consider a jet in our data through jet clustering (see Sec. 2.2.1) as well as to consider how we may characterize the resulting jets via jet observables (see Sec. 2.2.2).

2.2.1 Jet Clustering

We first discuss the criteria for a good clustering algorithm. Fundamentally, these criteria are all based on the need for the jet algorithm to be well defined and easily applicable to both theory predictions and experimental measurements. For theory predictions, we require that the perturbative cross section be well defined and finite, which means that it should be

insensitive to the effects of hadronization and that it should not be affected by collinear or infrared (low-energy) emissions, i.e., that it should be collinear and infrared safe. From the experimental side, it is key that detector effects and effects from pile-up should not overly affect the resulting jet.

In order to facilitate ease of use across different detectors as well as in both theory and experiment, the most common jet clustering algorithms rely solely on particle four-momenta, which are defined in theory and can be reconstructed from experimental measurements. Based on only these, clustering is performed, commonly with **FastJet** [66, 67], an open source framework, which implements many of the most common jet algorithms.

We can broadly categorize jet clustering algorithms by a few different characteristics. Firstly, jet algorithms are typically either a *cone algorithm* or a *sequential recombination algorithm*, two broad categories describing common traits of many algorithms: Cone algorithms cluster particles together that are within a fixed angular distance from the jet direction. The jet direction is initialized with a trial axis, such as the momentum direction of a high-energy object. It is recomputed from the momenta of clustered particles and iterated until a stable configuration is found. Historically, these algorithms have had problems with infrared safety. Sequential recombination algorithms, on the other hand, sequentially recombine pairs of objects to iteratively produce jets. They are based on the idea that jets are themselves based on sequential splittings and therefore aim to reverse this process. When recombining objects, there is, of course, the question of how one object is defined from a pair of objects. For example, the four-momentum of the highest energy object may be used (Winner Take All-scheme), or a sum of four-momenta may be used (E -scheme). For the rest of this section, we focus on a class of sequential recombination algorithms, namely the *generalized k_T algorithm*, which typically uses an E -recombination scheme.

Starting with a list of particle four-momenta, the generalized k_T algorithm works as follows:

1. For all objects in the list, the distance between pairs of particles d_{ij} and beam direction d_{iB} are calculated as

$$d_{ij} = \min(p_{Ti}^{2p}, p_{Tj}^{2p}) \cdot \frac{R_{ij}^2}{R^2} \quad (2.6)$$

$$d_{iB} = p_{Ti}^{2p} \quad (2.7)$$

where $R_{ij}^2 = (y_i - y_j)^2 + (\varphi_i - \varphi_j)^2$ is an angular distance between objects using the rapidity y and azimuthal angle φ of an object, both of which can be calculated from the four-momentum. R is the jet radius and p a weighting parameter. They are discussed in more detail below.

2. The minimum of $\{d_{ij}, d_{iB}\}$ across all i and j is chosen.
 - a) If the minimum is a distance between objects d_{ij} , objects i and j are combined into a new object by summation of four-vectors.
 - b) If the minimum is d_{iB} , object i is labeled as a jet and removed from the list.
3. Steps one and two are repeated until a stopping criterion is met.

There are three key choices for these clustering algorithms: Firstly, the stopping criterion needs to be set. Based on the stopping criterion, a clustering is classified into either *inclusive* or *exclusive* clustering. In inclusive clustering, jets are formed until all particles are contained

in a jet. In exclusive clustering, a required number of jets N is set, and particles are combined until all particles are contained in N jets. For jet clustering applied to a full event, typically inclusive clustering is used with the N jets with the largest p_T being chosen after the clustering is complete. This may, for example, remove low-energy pile-up jets from the event. Exclusive clustering may be used, for example, when calculating jet observables, as is discussed below in Sec. 2.2.2. Secondly, the parameter p needs to be chosen to determine the shape of the jets. Specific cases of p have their own names, namely the k_T algorithm [68, 69] with $p = 1$, the Cambridge/Aachen algorithm [70] with $p = 0$ and the anti- k_T algorithm [71] with $p = -1$. The anti- k_T algorithm is most commonly used because it results in stable and conical jets. The jet radius R is also crucial to determine the types of jets found by the clustering algorithm. In the case of highly boosted decays, it may be wise to choose a large jet radius such as $R = 1$, which catches all subjects. This is also referred to as a *fatjet*. A clustering with a smaller radius, such as $R = 0.4$, may be able to differentiate the individual decay products' jets in some cases. Both can also be used jointly to obtain a more complete view of the jet structure of an event.

Considering the generalized k_T algorithm in light of our clustering algorithm criteria, we can see that they are fulfilled: Firstly, collinear splittings approach $R_{ij} = 0$ and will thus, with a high likelihood (always for $R_{ij} = 0$), be clustered into the same jet. Secondly, infrared safety is fulfilled by the anti- k_T algorithm, which clusters high-energy particles first, meaning that the low-energy particles added later are unlikely to significantly impact the jet axis. Similarly, pile-up tends to result in additional soft particles. With the invariance of anti- k_T to additional soft particles, jets remain relatively stable as pile-up is added. The same holds for slight modifications of jet energies or directions due to detector uncertainties.

2.2.2 Jet Observables

Not all jets are identical. Instead, their substructure encodes information about the underlying physics of the initiating parton or particle. Firstly, a jet may contain the decay products of a strongly boosted particle. Take for example a W boson decaying into two quarks. Because they are highly boosted, the two quarks are typically reconstructed as a single large-radius jet. Nevertheless, the internal structure of the jet still reflects the multi-prong structure of the underlying decay topology, such as the two-prong decay of the W boson or the three-prong structure of top quark jets, which results from $t \rightarrow bW \rightarrow bq\bar{q}$. Secondly and independently of boosted heavy-particle decays, even pure quark or gluon jets exhibit characteristic differences depending on the initiating particle. For example, gluons have a larger color factor than quarks, resulting in a larger particle multiplicity. The heavier quark jets typically contain displaced vertices due to the production and decay of long-lived hadrons.

To characterize these differences between jets, we define jet observables based on the clustered jet and its constituents. Such observables can be broadly categorized into global jet observables and substructure observables. Starting with global jet observables, a key observable is the jet mass. This can be particularly useful to distinguish between boosted particle production and single parton jets: A characteristic peak is observed in the mass spectrum at the boosted particle's mass. Considering the jet momentum, it is common to use the transverse momentum p_T (invariant under boosts along the beam axis), the pseudorapidity η (approximately uniform distribution in the number of events), and the azimuthal angle φ instead of the Cartesian momentum vector. The pseudorapidity η is also often used for event selection in analyses. The jet multiplicity is the number of constituents of a jet. This observable is not infrared or

collinear safe but may indeed be characteristic between different jet types (e.g., quark gluon tagging [72] as also discussed above).

Jet substructure is the much more complex part of the picture. Describing it with as few observables as possible is a very active area of research. One of the most well-established substructure observables both in simple cut-based searches and in anomaly detection, is the *N-subjettiness* [24, 25] τ_N . It is designed to identify whether a jet has N or fewer subjets. This is done by calculating

$$\tau_N^\beta = \frac{1}{d_0} \sum_i p_{T,i} \min \left[(\Delta R_{1,i})^\beta, (\Delta R_{2,i})^\beta, \dots, (\Delta R_{N,i})^\beta \right], \quad (2.8)$$

where $\Delta R_{J,i} = \sqrt{(\Delta y_{J,i})^2 + (\Delta \varphi_{J,i})^2}$ is the angular distance between subjet axis J and jet constituent i , R_0 is the jet radius, β is an angular weighting parameter, the default value of which is $\beta = 1$, and d_0 is a normalization factor, usually defined as $d_0 = \sum_i p_{T,i} (R_0)^\beta$. Theoretically, the subjet axes are defined such that τ_N^β is minimal. This means that $0 \leq \tau_N^\beta \leq \tau_{N-1}^\beta$. In practice, a complete optimization is not possible, but often an exclusive clustering to obtain N subjets is used to obtain a first guess, to which a few optimization steps are applied. For the exclusive clustering, anti- k_T clustering is not ideal as the clustering first recombines hard particles, which may be required to define additional jets. k_T clustering, on the other hand, tends to first recombine close soft particles in close geometric proximity. The N -subjettiness is collinear safe as long as $\beta \geq 0$. Infrared safety is given as τ_N^β is linear in particle transverse momenta, limiting the impact of low-energy constituents. N -subjettinesses are implemented in **fj-contrib**, making them easy to use alongside **fastjet**. In this thesis, we use N -subjettiness observables calculated using an exclusive k_T -clustering with a single optimization step. For the normalization d_0 , we use the **UnnormalizedMeasure**, which sets $d_0 = 1$.

Often, not the subjettinesses themselves are used but the subjettiness ratios. For example, the 21-subjettiness ratio $\tau_{21} = \tau_2/\tau_1$ is said to be the best single-feature discriminator between a jet being 1- or 2-pronged [25]. Searches for 2-pronged signatures in a simple QCD background, therefore, often cut on τ_{21} . Similarly, such searches for 3-pronged signatures use τ_{32} [46]. While N -subjettinesses are very useful for condensing information into a single number, they are not designed and therefore also unable to fully represent a jet's structure. Many other jet observables focusing on different kinds of structures and representation goals exist (see, for example, the discussions in Ref. [49]). For example, Energy Flow Polynomials (EFPs) [73] were designed to provide a full substructure basis and can therefore better represent jets fully. Modern machine learning algorithms, particularly for jet tagging, often also directly work with jet constituent four-momenta, making use of the full jet information (see, e.g., Ref. [74–78]).

In addition to substructure observables, a jet may also be characterized by observables targeting the identification of specific jet components. For example, flavor tags, most commonly b -tags, can be crucial to identify particular decays. Lepton subjet fractions may also serve to identify leptonic jet components. Beyond single jets, multi-jet observables can also be key to identifying event types. Very commonly, a multi-jet invariant mass, often the dijet invariant mass $m_{JJ} = (p_{J1} + p_{J2})^2$, is used as resonant production of heavy particles can be identified as characteristic bumps in the m_{JJ} distribution and is the basis for the standard bump hunt. The distribution of jets may also be useful in analysis, reflected for example in the angular distance $\Delta R = \sqrt{\Delta \eta^2 + \Delta \varphi^2}$ between two jets.

2.3 JETS AND THE SEARCH FOR NEW PHYSICS

This section is based on Refs. [9, 48, 49, 65]. Additional sources are included where appropriate.

While jets are a common signature of many different Standard Model processes, they are also common predictions of BSM physics models. In particular, many models predict new heavy particles that couple to SM fields and can be produced resonantly at colliders, leading to observable resonances. While some open questions about the SM, such as the nature of dark matter, could in principle be explained by sectors that interact with the SM only through gravity [79], others, for example, the strong CP problem, are typically resolved by extensions of the SM introducing new particles that interact with SM fields. If such particles are to be detected at the LHC, they must be produced in proton–proton collisions, typically via quark or gluon initial states. Consequently, decays into quarks and gluons — and thus final states containing jets — are generally expected.

It is also crucial in this regard that the limits on cross sections for decays into jets are often weaker than those for, e.g., leptonic decays [80]. This is caused by a few different overlapping effects: For one, the SM background at the LHC is dominated by QCD processes and therefore jets (see Sec. 2.1). As such, a new resonance decaying into jets must typically be produced with a higher cross section to be detected at the same significance above the SM background. Furthermore, the complex, multi-particle structure of jets, shaped by parton showering, hadronization, and detector effects, can obscure exotic decay patterns that would be easier to identify in leptonic final states. This also affects the strength of limits for BSM models decaying hadronically versus leptonically. As a result, viable parameter space for new physics may remain in hadronic channels even when analogous leptonic searches impose stronger constraints.

While the complex structure of jets presents a challenge, it also presents an opportunity for improvement. With the large amount and the high dimensionality of jet data, machine learning has already been able to achieve significant gains in the last decade, for example, in the realm of jet tagging [13]. By developing ML methods to detect anomalous signatures in a wealth of SM data, the sensitivity of analyses to new physics in jets could be greatly improved, reducing the gap to leptonic limits. In particular, approaches that minimize reliance on specific signal models are well suited to exploring the large space of possible BSM scenarios [17]. The machine learning techniques that enable such searches are introduced in the following chapter.

3

FUNDAMENTALS OF MACHINE LEARNING

Machine Learning (ML) has been used in High Energy Physics since its infancy. However, over the last decade, it has become invaluable across a range of different applications, be that calibration, tagging, discrimination, or fast simulation. In this chapter, we introduce the fundamentals of ML and how it works, explain classical machine learning, in particular decision trees, before turning to Deep Learning. Lastly, we explain how probability distributions can be learned with Density Estimation, which is necessary for many of the anomaly detection methods used in this thesis, which are discussed below in Sec. 4. This chapter does not aim to be a complete description of Machine Learning, but rather to introduce the general concepts needed to follow the research presented in this thesis.

3.1 MACHINE LEARNING AS A FUNCTIONAL FIT

This section provides a relatively general and fundamental view of Machine Learning and is not explicitly based on any textbook. However, the fundamental concepts can be found, e.g., in Ref. [81]. Additional sources are cited, where more specific concepts or applications are considered.

There are numerous definitions of Machine Learning, which is in itself a broad term encompassing many different techniques. Which definition is used typically depends on the particular goal and viewpoint of the author or speaker. For example, a common computer science definition calls Machine Learning “learning without explicit programming” [82]. However, for statisticians it is often more useful to view machine learning as a probabilistic process (see, e.g., Ref. [83]). For physics applications and particularly for the applications in this thesis, we largely define ML as a strategy for performing functional fits without requiring an explicit parametrization. This description is useful because it explicitly highlights what Machine Learning can replace in physics and what its advantages are.

Functional fitting is an ever-present tool across physics: Given data $\{x, y\}$ and a fit function $f_{\vartheta}(x)$ with parameters ϑ , which are adjusted to optimize the χ^2 , physical parameters can be estimated. As an example, by fitting a line to the measured current as a function of the applied voltage in a circuit, the resistance of the circuit can be estimated. In ML, the χ^2 as a goodness of fit metric is generalized to any *loss function*, which is chosen according to the specific task and minimized. The fit function is replaced by the ML model, which can represent a wide range of functions. This more general functional shape is achieved through *overparametrization*, i.e., the model has many more parameters than would be needed to accurately fit the data. The precise structure of the model and the fit are determined by the model’s *hyperparameters*, a term which refers to any parameters which are not learned but set before fitting. When the model’s parameters are optimized according to the loss function, the solution space is usually degenerate with multiple parameter configurations leading to similar and equally accurate (same loss value) fits to the data. Using ML instead

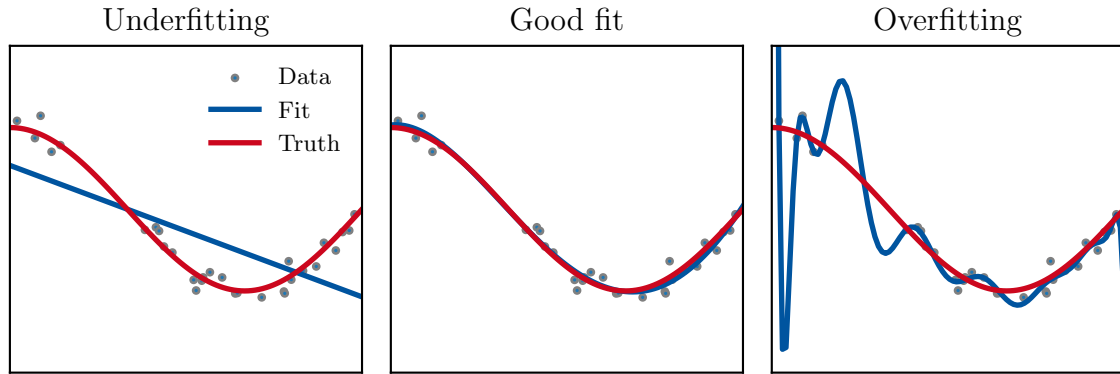


Figure 3.1: Representation of underfitting (left), a good fit (middle), and overfitting (right) to the same data. The data is drawn from a cosine function with noise, while the fits are polynomials of degrees 1 (left), 4 (middle), and 15 (right). Figure recreated from Ref. [86].

of a standard functional fit is therefore especially useful when the functional form is unknown, and no specific physical parameter (such as the resistance) needs to be extracted from the data.

A good example for an application that meets these criteria is the fitting of parton distribution functions (see Ch. 2), which is required to match theory predictions to experimental measurements. For PDFs, no full, theory-driven parametrization is known. There are a number of different collaborations working on this problem with different methodologies, often including orthogonal polynomial expansions such as Bernstein or Chebyshev polynomials [84]. NNPDF [85] provides a fit with the use of Multi-Layer Perceptron Neural Networks (see Sec. 3.4).

An important aspect of a good fit is the generalization it provides, namely that accurate predictions can be made for previously unseen data. When the true functional form is known, this generalization (at least in close proximity to the measurement data) is a given, but with the overparametrization of ML models, it is not. Instead, ML models may *overfit*, i.e., learn the precise distribution of the training data (the data on which the fit was performed) instead of the underlying functional shape, as can be seen in Fig. 3.1. Here, overfitting is shown for polynomials of different degrees d fitting the same non-polynomial, noisy data. As d increases, first, the fit becomes more accurate to the function underlying the distribution. However, for large d , the fit has the capacity to follow the points from the training set relatively closely, reducing the loss further than the underlying function would. This, we call overfitting. For new unseen data, it would lead to a large loss value. We say the fit does not generalize. Fundamentally, this problem is one of limited data: Infinite training data points, even if noisy, would not lead to overfitting even for large d . However, realistic applications do not have infinite data, and so usually the problem needs to be controlled in other ways, as is discussed in more detail in Sec. 3.4. While controlling overfitting may be difficult, diagnosing it is relatively easy: A part of the available data is kept separate from the training set to evaluate the trained model on an independent set. Should the performance of the model differ greatly between the training set and this independent test set, then overfitting has occurred. However, it should be noted that small differences are expected. With model types, where fits are improved iteratively, e.g., MLPs and Gradient Boosted Decision Trees, it can also be useful to track the performance on an independent set throughout the training (validation set), to, e.g., stop the training before the overfitting of the model becomes too severe.

The interplay between under- and overfitting is also referred to as the *bias-variance tradeoff*, a term which we return to throughout this chapter. Bias is reflected in the training loss of the model, which is high for an underfitting model, low for a good fit, and very low for an overfitting model. Variance here refers to fluctuations between different estimator trainings, particularly between those using different sets of training data. The variance is particularly low for underfitting models, low for good fits, and high for overfitted models. A good fit, therefore, sits at the optimal point between bias and variance and represents the tradeoff between both.

When describing ML as a functional fit, there is one aspect that is undercut slightly, namely the broad range of possible ML applications: Functional fitting, as introduced so far, mostly describes a regression task, which is just one of these possible applications. Regression – together with classification – falls into the category of supervised ML, which is defined by requiring truth labels to fit the model. Classification is the division of data points into two (binary classification) or more (multi-class classification) pre-defined categories and is central to this thesis. It is discussed in more detail in Sec. 3.2. In contrast to supervised ML, unsupervised ML does not use labels. There is a number of different unsupervised ML tasks, such as clustering or density estimation (see Sec. 3.5). While supervised and unsupervised ML are the extremes, the degree of supervision inherent to a task is a spectrum, with techniques such as semi-supervised, self-supervised, or weakly supervised learning, which we focus on in this thesis, bridging the gap. Across this spectrum, there is a number of different applications in particle physics, both in experiment and in theory: Regression tasks are not only used in PDF fits but also to calibrate detector measurements. Classification is used to separate signal from background events for an analysis and to perform jet tagging (see, e.g., Ref. [13]) or particle tagging more generally, identifying particle types using as much of the detector information as possible. Different types of unsupervised machine learning are used as anomaly detectors (see, e.g., Ref. [17, 87]), flagging rare events, and density estimation is used throughout the entirety of simulation pipelines to speed up event generation, from the hard process level with MadNIS [88, 89] to detector simulations (see, e.g., Ref. [90])¹.

3.2 THE NEYMAN-PEARSON LEMMA AND CLASSIFICATION TASKS

The general information on classification in this section can be found in Refs. [82, 83]. Additional sources for more specific concepts are cited where appropriate.

Classification is the division of data into two or more pre-defined classes. The classification function or *classifier* generally outputs a classification/classifier score for each class, which in a binary classification can be accounted for by a single classifier score. In this case, low scores mean an event is likely to belong to one class, high scores to the other. Machine learning classifiers are usually defined in such a way that their classifier scores can be interpreted as class probabilities. This is achieved by using the crossentropy loss function (based on the Shannon entropy [91]), here specifically the Binary Crossentropy (BCE) loss

$$\text{BCE}(y_{\text{pred}}, y_{\text{true}}) = y_{\text{true}} \log y_{\text{pred}} + (1 - y_{\text{true}}) \log(1 - y_{\text{pred}}), \quad (3.1)$$

where $y_{\text{true}} \in \{0, 1\}$. While any classification would be solved perfectly with $y_{\text{pred}} = y_{\text{true}}$, for almost all real-world and particularly physics applications, this is not possible. See, for

¹ In addition to supervised and unsupervised ML, there is also reinforcement learning, which trains models via occasional rewards (such as after winning a computer game by taking the correct actions for the given input).

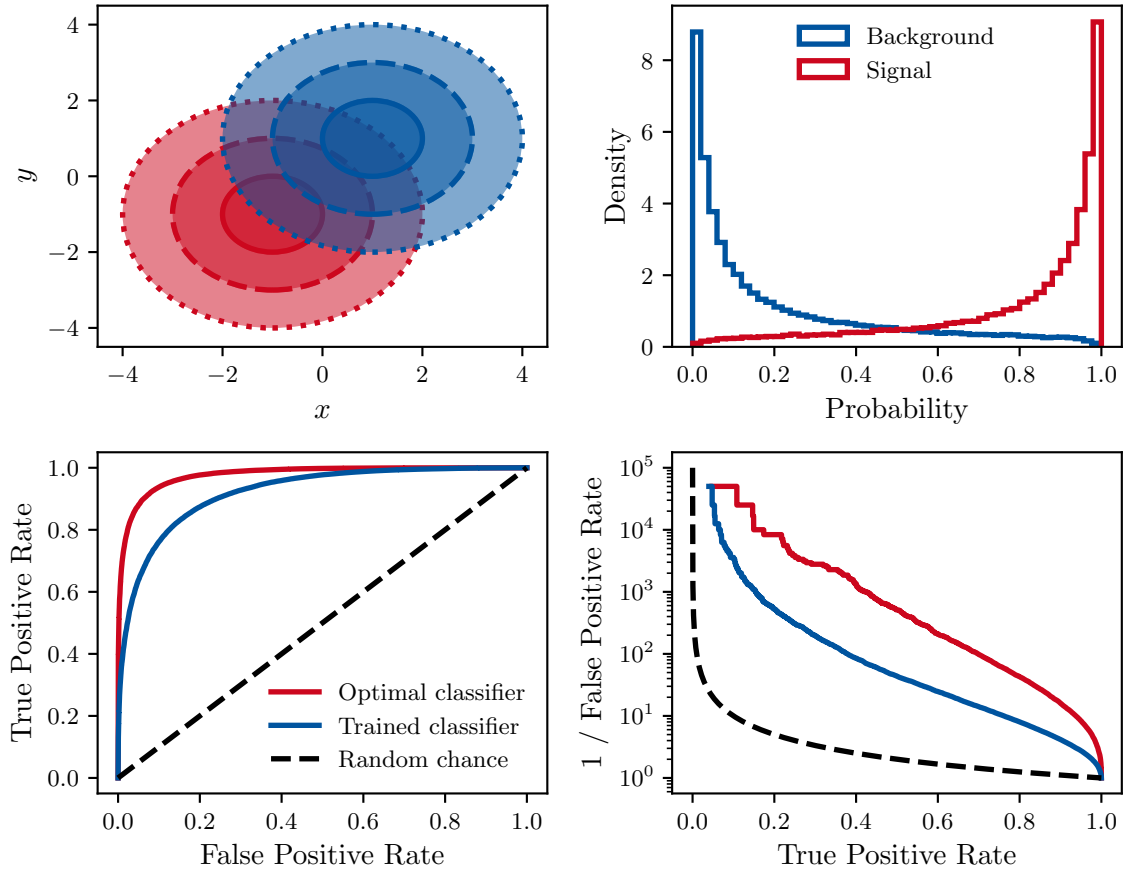


Figure 3.2: Visualization of a classification task between two 2D Gaussians with means $\mu = (\pm 1, \pm 1)$ and standard deviations $\sigma = 1$. Top left: distribution of both classes showing the 60%, 95%, and 99% coverage regions of signal and background. Top right: Score distribution of a classifier trained on the task. For visualization purposes, the trained classifier is trained on only one of the features. It therefore does not use the full information. As the problem is so simple, most trained classifiers indeed match the optimal classifier perfectly. Bottom left: Standard ROC curve of the Neyman-Pearson optimal classifier, the trained classifier, and a random chance classifier. Bottom right: ROC curve of the Neyman-Pearson optimal classifier, the trained classifier, and a random chance classifier showing logarithmic background rejection on the y-axis.

example, the upper left panel of Fig. 3.2, which shows two 2D Gaussians at means $(\pm 1, \pm 1)$ and with standard deviations $\sigma = 1$. The overlap between the two distributions makes a perfect classification impossible. Indeed, the BCE is minimized by

$$f_{\text{opt}} = \frac{p_1(\vec{x})}{p_0(\vec{x}) + p_1(\vec{x})}, \quad (3.2)$$

for data points from classes 0 and 1 drawn from probability distributions $p_{0/1}(x)$, respectively. The interpretation of classifier scores as probabilities comes from this optimal solution, which is essentially the probability of an event at position x being drawn from class 1. For our 2D Gaussians, classifier scores may, for example, look like the ones shown in the upper right panel of Fig. 3.2.

While useful to understand a classifier's training, score distributions alone make it difficult to compare different classifiers' performances. This is particularly true if not all classifiers are

BCE trained and, as such, may be normalized differently. These normalizations are irrelevant for the classification performance, as classification cares about the ordering of events and not the precise values of the classifier score. Therefore, we define cut-based metrics to compare different classifiers: Consider a cut R_c on the classifier score $R(x)$, for which we keep all events above this cut, i.e., with $R(x) \geq R_c$. An ideal classifier should have as many events as possible from class 1 pass this cut (true positives) and as few events as possible from class 0 pass this cut (false positives). We can consider this for different cuts and compare classifiers based on their True Positive Rates (TPRs) at any given False Positive Rate (FPR). The best classifier is the one which across all FPRs has the highest TPRs. We can plot these rates against each other in a Receiver Operating Characteristic (ROC) curve as seen in the lower left panel of Fig. 3.2. Here, we show the trained classifier from the top right panel and a random chance classifier, which assigns score values randomly and therefore always returns $\text{TPR} = \text{FPR}$ for any cut. In physics applications, we often display ROC curves as shown in the lower right panel of Fig. 3.2, which has a better resolution for very accurate classifications. These may be difficult to distinguish in the top left corner in the standard ROC curve format. The y-axis ($1/\text{FPR}$) is then often referred to as the *Background Rejection*. In particle physics, it is more common to refer to FPR and TPR as efficiencies ε (often ε_B and ε_S for background and signal), reflecting the fact that classifiers are treated as selection cuts, much like energy or transverse-momentum thresholds.

In addition to the trained and random chance classifiers, both ROC curves in Fig. 3.2 contain a line labeled as the “optimal classifier”. This optimal classifier is based on the Neyman-Pearson-Lemma [92], which states that the optimal classifier is given by the likelihood ratio

$$R_{\text{opt}} = \frac{p_1(x)}{p_0(x)}. \quad (3.3)$$

We prove the Neyman-Pearson-Lemma by following Ref. [93]: As we wish to compare classifiers at fixed values of FPR, we fix the cut value $R_{c,\text{NP}}$ for the Neyman-Pearson classifier as well as the cut value $R_{c,A}$ for another classifier A such that we obtain the same fixed FPR. As the probability distributions and therefore their ratios are positive definite, $R_{c,\text{NP}} \geq 0$ must also be true. We define the decision function of a classifier as

$$\Phi(x) = \begin{cases} 1 & \text{if } R(x) \geq R_c \\ 0 & \text{else.} \end{cases} \quad (3.4)$$

First, we show that

$$\int_x \underbrace{(\Phi_{\text{NP}}(x) - \Phi_A(x))}_{U_1} \cdot \underbrace{(p_1(x) - R_{c,\text{NP}} \cdot p_0(x))}_{U_2} dx \geq 0, \quad (3.5)$$

where we integrate over the entire phase space in x . We use that

$$\frac{p_1(x)}{p_0(x)} \geq R_{c,\text{NP}} \quad (3.6)$$

if and only if $\Phi_{\text{NP}}(x) = 1$ and consider the integrand for the three possible configurations of $\Phi_{\text{NP}}(x)$ and $\Phi_A(x)$:

1. $\Phi_{\text{NP}}(x) = \Phi_A(x)$: $U_1 = 0$, meaning that the integrand on the LHS is 0.
2. $\Phi_{\text{NP}}(x) = 1$ and $\Phi_A(x) = 0$: $U_1 \geq 0$ and $U_2 \geq 0$ because the likelihood-ratio criterion in Eq. 3.6 is met.

3. $\Phi_{\text{NP}}(x) = 0$ and $\Phi_{\text{A}}(x) = 1$: $U_1 \leq 0$ and $U_2 \leq 0$ because the likelihood-ratio criterion in Eq. 3.6 is not met.

Since the integrand $U_1 \cdot U_2 \geq 0$ for all x , this is also true for the integral. We can then rearrange Eq. 3.5 such that we obtain

$$\begin{aligned} & \int_x \Phi_{\text{NP}}(x)p_1(x)dx - \int_x \Phi_{\text{A}}(x)p_1(x)dx \\ & \geq R_{c,\text{NP}} \left(\int_x \Phi_{\text{NP}}(x)p_0(x)dx - \int_x \Phi_{\text{A}}(x)p_0(x)dx \right). \end{aligned} \quad (3.7)$$

The integration of $\Phi(x)p_{0/1}(x)$ over the full feature space gives us the respective classifier's FPR and TPR. As such

$$\text{TPR}_{\text{NP}} - \text{TPR}_{\text{A}} \geq R_{c,\text{NP}} (\text{FPR}_{\text{NP}} - \text{FPR}_{\text{A}}) = 0, \quad (3.8)$$

for $\text{FPR}_{\text{NP}} = \text{FPR}_{\text{A}}$. As such, the TPR of the likelihood ratio classifier is at least as good as that of any other classifier A for a fixed FPR, proving the Neyman-Pearson Lemma.

In real-world applications, the true likelihood functions are usually not known, and it can therefore be difficult to obtain a good estimation of the optimal solution to a classification task². However, the optimality of the likelihood ratio also has a direct connection to BCE-based classification: It is in fact monotonically related to the optimal solution of the BCE (see Eq. 3.2). As only the ordering of events by a classifier and not their precise scores are relevant for our classification metrics, any monotonic function of a classifier score is equivalent to that classifier score. Thus, the optimal solution of the BCE loss is also the optimal classifier.

While theoretical optimality is a good foundation for choosing how to train ML models, actually obtaining optimal results does not tend to be practically achievable. Data statistics, model sizes, computation budgets, as well as trainability limit realistic models. How robust, close-to-optimal classifiers can be obtained in practice and what needs to be taken into account to train them is discussed in the next two sections.

3.3 CLASSICAL MACHINE LEARNING AND DECISION TREES

While the original papers as well as additional sources are cited where appropriate, the explanations in this Section are largely based on Refs. [96, 97], which provide well-referenced and easy-to-follow explanations of these concepts.

Classical Machine Learning is a phrase used to distinguish older Machine Learning techniques from Deep Learning and is, as such, an umbrella term for many vastly different techniques such as genetic algorithms, support vector machines, kernel methods, Gaussian processes, and Decision Trees (DTs). While most of these methods are often seen as obsolete, they do still have their specific use cases. For example, some problems may be relatively easy to solve, meaning that a simpler model still suffices, but has other advantages such as shorter run times and lower computational demands. In other cases, specific properties allow them to outperform deep learning models. This is the case for decision tree-based models trained on tabular data, particularly with limited statistics. In fact, gradient-boosted decision trees, which train multiple trees to iteratively improve the model (see Sec. 3.3.2), remain state of the

² Although attempts to do this even when no likelihood functions are known have been made in Refs. [94, 95].

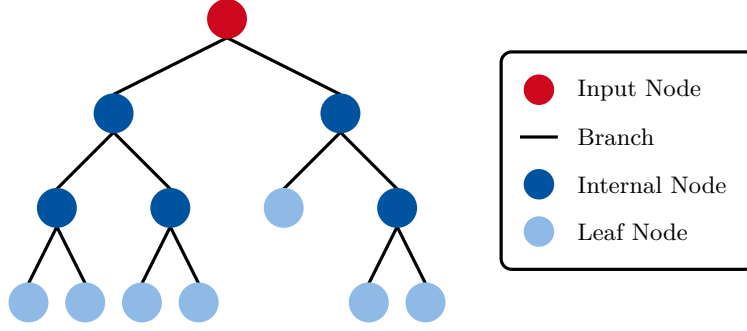


Figure 3.3: Schematic representation of a decision tree.

art for these problems [98], which is particularly relevant to anomaly detection (see Sec. 5.1). We therefore explain the working principles of decision trees and boosted decision trees in this section.

In order to solve, for example, a classification problem, decision trees split data sequentially into smaller and smaller groups to isolate the classes, essentially defining a custom, irregular binning. This sequential splitting structure can be visualized as shown in Fig. 3.3: At several *nodes*, both in the first input (or root) node and the later internal nodes, a split is performed along one feature x_i , dividing the data into one group with $x_i < t_i$ and one with $x_i \geq t_i$. Finally, in the final output or leaf nodes, class labels or probabilities are assigned to the subgroups. This general structure is common across many different decision tree algorithms. How the splits are chosen in the training procedure depends strongly on the particular algorithm. While the first decision tree algorithm was published in the 1960s [99], modern implementations largely use more recent algorithms. Many modern implementations of DTs are based on the CART (Classification And Regression Trees) algorithm [100]. We therefore focus on this algorithm below.

To achieve the globally optimal decision tree, it would be necessary to optimize the entire tree structure including all splits simultaneously. However, such an optimization would neither be practical nor even doable in realistic problems. Therefore, CART performs a greedy optimization, adding one node at a time and optimizing that split only. For a single feature, the number of possible split points is $k - 1$ for k unique values the feature takes in the given data. This finite number of splits can be compared for all features in terms of the lowest loss value. This loss should optimize the purity of both resulting nodes. While the binary crossentropy (Eq. 3.1) can be used for this (and is used in the examples in this thesis), a common choice is also the *Gini impurity*, which is defined as

$$\text{Gini}(p, q) = 1 - p^2 - q^2 \quad (3.9)$$

with the fraction of events per class in the node p and q . This measure was first introduced with CART [100] but is based on the Gini coefficient [101]. The quadratic form of this function encodes a preference for a single pure node rather than two equally impure nodes where possible. It is somewhat faster to compute as there are no logarithms in the formula, but it does not have the optimality argument of the BCE (see Sec. 3.2). For binary classification, both lead to similar results [102]. For regression tasks, often the variance of the regression target y within each node is minimized.

To select the next node, CART uses a depth-first approach: After a split at a node has been defined, a split is added to one of the two new nodes resulting from it. This is repeated until a stopping criterion is met. Then, splits are added to other nodes. The

tree becomes deeper before it becomes wider. This approach was followed in **XGBoost** (*Extreme Gradient Boosting*) [103], which was the state-of-the-art boosted decision tree implementation for a considerable time and still is a very commonly used implementation [98, 104]. However, in the current state-of-the-art [98] implementation **LightGBM** (Light Gradient Boosting Machines) [105], a best-first strategy based on Ref. [106] is employed: In this method, the split optimization is performed for all current leaf nodes, and the best possible split is performed next. In this case, too, the optimization is stopped once one of the stopping criteria is met. Stopping criteria are very important when using decision trees, as without them, trees can overfit perfectly, isolating each data point in its own leaf node. Possible stopping criteria are, for example, maximum depth, maximum number of leaf nodes, minimum number of samples per leaf node, and a perfect purity of a leaf node. In addition to stopping criteria, overfitting can also be reduced by pruning after the fitting is completed, i.e., by removing branches. It is important to note that, in general, fitting a decision tree is a fully deterministic process. However, randomness can be introduced, for example, by optimizing on subsets of the data or allowing each split to only optimize across a subset of features.

After the training is complete, each leaf is assigned a class label according to the majority class from the training set in that leaf node. For class probabilities, the fraction of events from each class in each leaf node is considered. A previously unseen data point is assigned a class label or probability based on the leaf node it reaches, determined by its feature values and the learned splits. Where classes are imbalanced, weights can be applied to samples to account for this, both when fitting the tree and when calculating the class labels or probabilities.

While decision trees consider features one-by-one, their recursive nature does allow them to learn correlations: A class boundary, which is, for example, diagonal in a 2D feature space, can be approximated by staircase-like consecutive splits. As the number of splits increases, this boundary approaches the diagonal. However, making splits across one feature at a time also allows them to deal particularly well with *Tabular Data*. This term describes data with features, which are relatively independent of one another and would, as such, make most sense written in a table. Take, for example, the age, height, and shoe size of a person: Mixing these values tends to destroy the inherent meaning of the numbers and may not make sense unit-wise. On the other hand, take a momentum vector's components p_x , p_y , and p_z : If we mix two features, e.g., $p_x + p_y$, the resulting feature still contains physical meaning, and indeed a simple redefinition of the coordinate system already mixes the features. The first are tabular features, while the second are not. The boundary, especially for physics features, is hard to draw precisely, but a test for whether features are tabular is discussed in Sec. 5.1. The property of decision trees to deal well with tabular data is a focus there.

Despite the ability to learn correlations and their particular advantageous properties when working with tabular data, single decision trees are not particularly powerful models. In fact, they are often referred to as *weak learners*, particularly if relatively shallow trees with few leaf nodes are used. In order to obtain a *strong learner*, multiple trees are often combined into a single model, also known as *ensemble learning*. The ensemble model keeps many of the advantages of decision trees, while also allowing the model to build more complex decision boundaries.

The simplest form of ensembling is to train multiple models independently (ensuring some randomization) and combine the results in some way. This combination can take different forms, such as a mean of class probabilities or a majority vote, where the fraction of weak learners voting for each class determines the ensemble vote. As long as each weak learner is better than random, a majority vote will always improve the likelihood for making a correct

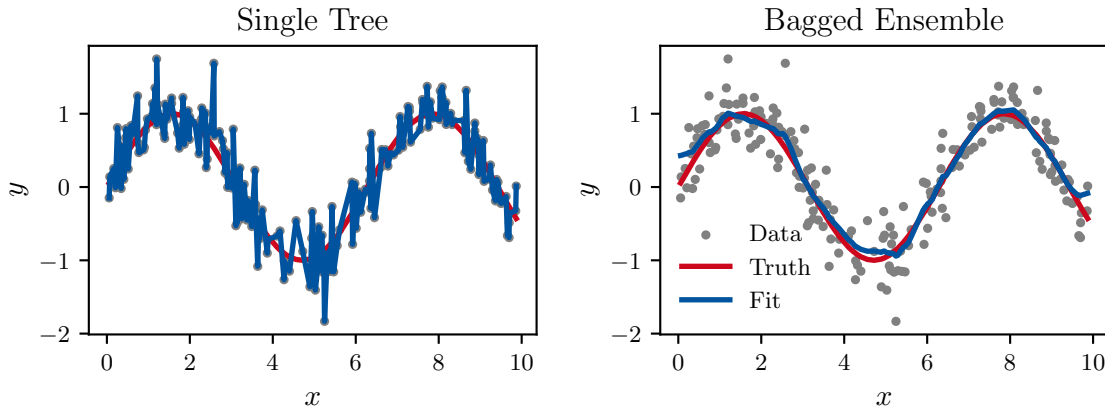


Figure 3.4: Visualization of Bagging using 200 data points drawn from a noisy sine wave showing data, the true sine wave function, and the fit. Left: A single decision tree with as many nodes as data points. Right: A bagging ensemble of 1000 decision trees with the same hyperparameters, subsampling 10% of the data. Figure recreated from [97].

decision [107, 108]³, just as the mean of multiple possible predictions will allow for a more stable and precise prediction. This type of simple ensembling is also called *Bagging* [110], which stands for “Bootstrap Aggregating”, with bootstrapping referring to the drawing of data points (usually with replacement), which introduces the required randomization. A special case of bagging is a *Random Forest* [111], which, in addition to the random data subsets, also randomly selects a subset of features each tree is trained on.

Bagging works particularly well if each weak learner tends to overtrain and is relatively unstable, i.e., if small changes of the training data lead to large model changes. The goal is less to improve the bias (see Sec. 3.1) but to reduce the variance. This can be seen in Fig. 3.4, where a single tree and a large bagged ensemble with 10% subsampling of data points per tree are trained on the same data and compared. While the single tree has a very large variance, overfitting perfectly to the data, the bagged ensemble follows the true distribution very closely. Each individual tree in the ensemble still overfits to its subsample of the data but this overfitting averages out when combining their predictions. While this works very well to reduce variance, it does not decrease a model’s bias if it is fundamentally not complex enough. For this, Bagging is not the right method, and *Boosting* should be used instead. These ensembling methods train the decision trees sequentially, with each subsequent tree’s setup and task depending on previous results, essentially subsequently adding correction terms to the fit function. The most commonly used boosting methods are AdaBoost and Gradient Boosting, which are introduced below.

3.3.1 Adaptive Boosting

We start with Adaptive Boosting (AdaBoost) [112], the earlier of the two methods, which aims to focus on previously badly classified events in subsequently trained learners. This is achieved by assigning each data point i a weight $w_{t,i}$, which encodes how difficult it was to classify a data point previously and determines what data points learner t puts particular

³ That political theory justifications of democracy are equivalent to the justifications of ensemble learning is discussed in detail in Ref. [109].

focus on. These weights are adjusted after each learner and are initialized with either an equal weighting of all data points or class/sample weights, for example, to correct for unequal class sizes. The sum of all weights is normalized to 1. Once the training of learner t is complete, a tree weight γ_t is calculated

$$\gamma_t = \frac{1}{2} \log \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right), \quad (3.10)$$

where ε_t describes the learner's total error and is defined as $\varepsilon_t = \sum_i w_{t,i} |y_{\text{pred},t,i} - y_{\text{true},i}|$, where $y_{\text{pred},t,i}$ is the class prediction for point i of learner t and $y_{\text{true},i}$ is the class label for point i . Using this, sample weights are then adjusted for the next learner $t + 1$ according to

$$w_{t+1,i} = \frac{1}{c_N} w_{t,i} \begin{cases} e^{\gamma_t} & \text{if } y_{\text{pred},t,i} \neq y_{\text{true},i} \\ e^{-\gamma_t} & \text{if } y_{\text{pred},t,i} = y_{\text{true},i} \end{cases} \quad (3.11)$$

where c_N is chosen such that the weights are normalized to one. Hence, weights are increased if events were incorrectly classified by tree t and decreased if they were correctly classified. How large this increase or decrease is depends on the accuracy of the last tree, which is encoded in γ_t . This process is repeated until a maximum number of iterations T is reached or the total error is zero. The final model output is then computed according to a weighted majority vote with weights γ_t .

In Ref. [113], it was later shown that AdaBoost fundamentally performs a *Gradient Descent*-based optimization (strategy explained more in Sec. 3.3.2) on a specific loss function, namely the exponential loss

$$L_{\text{exp}}(y_{\text{pred},i}, y_{\text{true},i}) = \exp(-y_{\text{pred},i} \cdot y_{\text{true},i}), \quad (3.12)$$

for which class labels are defined as -1 and 1 . An easy-to-follow version of this proof can be found in Ref. [114]. The exponential loss is generally not ideal for classification because the large weights of misclassified examples make it relatively unstable. This limitation motivated the development of *Gradient Boosting*, which extends the underlying gradient descent-based optimization strategy of boosting to more general loss functions.

3.3.2 Gradient Boosting

Building on the interpretation of boosting as functional gradient descent established in Ref. [113], Ref. [115] introduced Gradient Boosting, which generalizes the approach to arbitrary differentiable loss functions. The fundamental idea is the following: To improve upon a current prediction $y_{\text{pred},t}$ and minimize the loss function $L(y_{\text{true}}, y_{\text{pred}})$, a standard method is to take a step proportional to the negative gradient of the loss function at the current prediction. In Gradient Boosting, this method is simply applied to the current ensemble prediction $y_{\text{pred},t} = F_t(x)$ by taking a next ensemble step

$$F_{t+1}(x) = F_t(x) - \alpha \frac{\partial L(y_{\text{true}}, F_t(x))}{\partial F_t(x)}, \quad (3.13)$$

where α is referred to as the learning rate and is a tunable hyperparameter. The goal of each tree is therefore to approximate the current ensemble state's gradients for the training data, allowing for them to be estimated for unlabeled test data. To perform this iterative optimization in practice, a constant initial prediction $F_0(x) = f_0$ is made for all data points, with the value of f_0 being chosen to minimize the loss function. Then, regression trees f_t

are trained to approximate the negative gradient of the loss with respect to the ensemble prediction $F_{t-1}(x)$ at each data point. This iterative procedure is repeated until a certain number of iterations is reached or another stopping criterion, such as an early stopping on a validation loss, is met. The full ensemble prediction of an ensemble with T trees is then calculated as

$$F(x) = f_0 + \sum_{t=1}^T \alpha f_t(x). \quad (3.14)$$

As already stated above, frameworks using gradient boosted decision trees have been state of the art for tabular data for a considerable time now, particularly since their efficient implementation in **XGBoost** and later **LightGBM**⁴. Both implementations make targeted changes to the original proposal. In **XGBoost**, the most significant difference is that it takes into account the second derivatives in addition to the gradient. In **LightGBM**, the first-order approximation is kept, but data is binned before training, leading to a large computational speed-up due to the limited number of split locations and required memory. There are other, more minor changes, for example, the CART trees being fit best-first in **LightGBM** as stated above and the inclusion of additional features such as sub-sampling and regularization (more on regularization in Sec. 3.4). In this thesis, a somewhat simplified version of the **LightGBM** implementation from **scikit-learn** [117] is used, namely the **HistGradientBoostingClassifier**.

It should be noted that the general procedure of Gradient Boosting is not limited to decision tree models and can, in principle, be extended to many other learners, although the application to decision trees is most common.

3.4 DEEP LEARNING AND NEURAL NETWORKS

This section is based on Refs. [81–83, 118]. Additional citations, as well as the original papers introducing certain concepts, are included where appropriate. If one particular textbook was used for a specific paragraph, a citation to this is added at the end of the paragraph for readers seeking to read up on certain concepts in more depth.

In contrast to classical machine learning, which is a broad term encompassing many vastly different methods, deep learning describes methods built on the same fundamental building blocks, concerning both training methods and network structures. The simple Multi-Layer Perceptron (MLP), for example, is an integral part of many more complex architectures (such as the density estimation methods discussed in Sec. 3.5) as it can easily fit a wide range of functions while at the same time being computationally efficient and simple. MLPs are used throughout the different projects discussed in this thesis, and therefore, we describe this architecture in detail in this section and use it as an example to discuss the key concepts in deep learning.

We begin by discussing the structure of MLPs, an example of which is shown in Fig. 3.5. MLPs, as the name says, are built in *layers*. In each layer, there are *nodes* (also called *neurons* as the structure of MLPs was inspired by the structure of the brain), which are usually specified by whether the layer they are located in is the input layer, output layer, or one of the internal or *hidden* layers. The number of layers in a network is referred to as its *depth* and the

⁴ **CatBoost** [116] is another popular implementation [98], but as it was neither first nor the basis for implementations used in this thesis, we do not discuss it in more detail here.

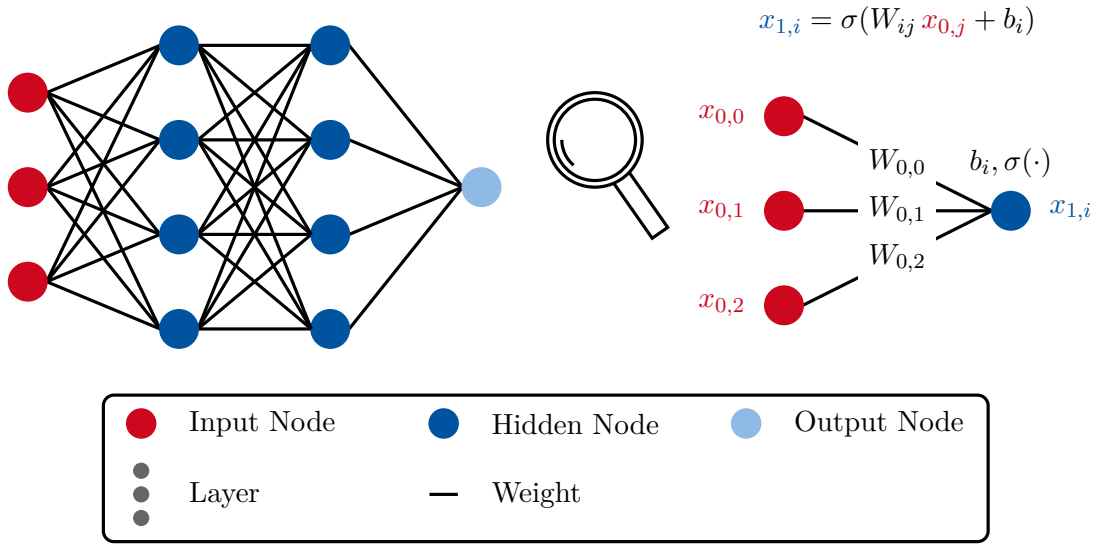


Figure 3.5: Schematic representation of a multi-layer perceptron showing an MLP with two hidden layers (left) as well as an example for the calculation performed in each node (right).

number of nodes per layer as the *width*. Each node is connected to all nodes of the previous and next layer, which is the reason MLPs are also commonly referred to as *Fully-Connected Neural Networks*. Every connection represents a *weight* W_{ij} , with which the *activation* of the previous layer's node $x_{n,j}$ contributes to the *activation* $x_{n+1,i}$ of node i in the next layer. The calculation of this activation is given by

$$x_{n+1,i} = \sigma \left(\sum_j W_{ij} x_{n,j} + b_i \right). \quad (3.15)$$

In addition to the weights W_{ij} , there is also the *bias* b_i of node i , which is a simple offset. Together, all weights and biases in a network make up the learnable parameters ϑ of the network. Fundamentally, the calculation at each node is linear, and it is crucial that this linearity is broken by applying a non-linear activation function σ . Otherwise, a multi-layer network would collapse into an effective single-layer network.

There are, of course, many different functions, which may be used as activation functions. A very popular example is the Rectified Linear Unit (ReLU) defined as

$$\text{ReLU}(x) = \max(0, x), \quad (3.16)$$

which is shown in the left panel of Fig. 3.6. It is fast and easy to compute but has the disadvantage of gradients vanishing for negative values, which can become problematic for training (see below). There are a number of alternatives to ReLU, which seek to address this problem while keeping its general structure, such as Leaky ReLU [119] or GeLU [120]. However, often ReLU suffices, and its simplicity leads it to be commonly used. Often, all hidden layers use the same activation function, but the output layer activation function commonly differs, as the output may need to be brought into a specific form. Classification is a good example for this as the binary crossentropy (see Eq. 3.1) requires outputs to be bounded between 0 and 1. As such, it is sensible to use an activation function bounded between 0 and 1. This is the case for the sigmoid function

$$\text{sigmoid}(x) = \frac{e^x}{1 + e^x}, \quad (3.17)$$

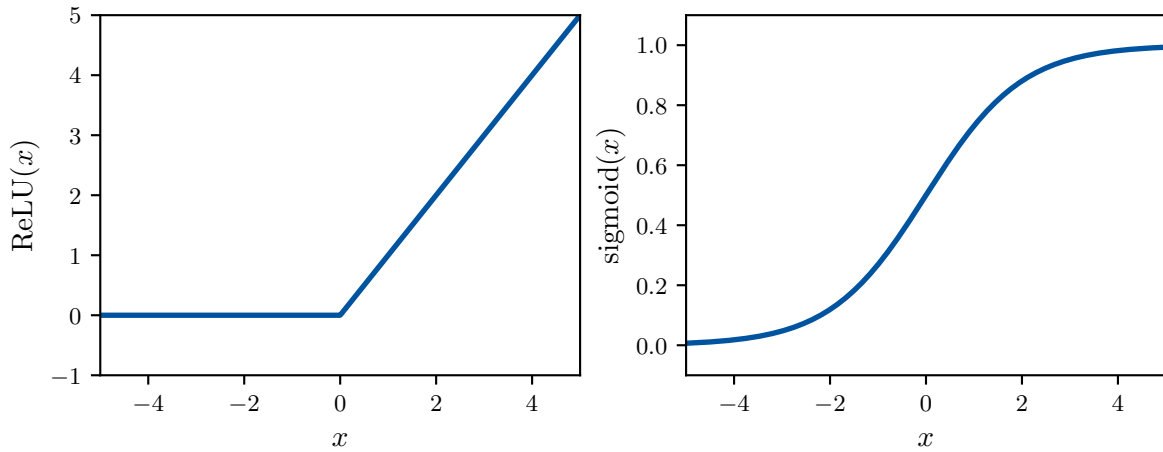


Figure 3.6: Common activation functions ReLU (left) and sigmoid (right).

which is therefore used by default in a binary classification⁵. In many cases, a network might be able to theoretically learn the normalization, but including known normalizations in activation functions may lead to improved performance.

As a whole, an MLP is then just a series of subsequent transformations applied to the data forming a function $f_{\vartheta}(x)$, which depends on its learnable parameters. While the operations within each node are relatively simple, MLPs can represent general functional shapes. Indeed, MLPs can be built in such a way as to be *Universal Functional Approximators* under the *Universal approximation theorem*:

Universal approximation theorem. *A multilayer perceptron with one hidden layer with any “squashing” activation function⁶ can approximate any Borel Measurable function⁷ from one finite-dimensional input space to another with arbitrary precision provided it has enough hidden layer nodes [81, 121–123].*

While often referred to in the singular as the universal approximation theorem, there are in fact many different versions of this theorem, which consider different classes of functions to be approximated, different activation functions, such as ReLU [124], and different network structures. However, the single very wide – sometimes infinitely wide – hidden layer version is used most commonly, standing in stark contrast to commonly used network structures, which tend to be rather narrow and deep. This is due to the disconnect between the general ability of a Neural Network (NN) to approximate a function and how easy it is to train: In most real-world examples, it is most useful for networks to be able to learn useful features hidden in the input data in their first layers and make decisions based on these features in later layers. For example, a circular decision boundary can be approximated by a shallow network only through many piecewise-linear regions. A deeper network can first learn a radial feature (distance from the origin) and subsequently apply a simple threshold, resulting in a more parameter-efficient solution. This effect has been studied in Refs. [125–128], of which Ref. [126], for example, found that the number of nodes required in one-layer networks may

⁵ In a multi-class classification, softmax is used instead, which normalizes the sum of all outputs to one. For binary classification, a network with two output nodes and softmax activation is equivalent to a network with one output node and sigmoid activation.

⁶ A function, which maps a its input onto a limited range, e.g., the sigmoid function.

⁷ Approximately, any continuous function on a closed and bounded subset of $\mathbb{R}^n$.

in some cases scale exponentially with the required nodes in deeper networks. The concept of learning generalized features for the task at hand is called *feature engineering* and is one of the fundamental strengths of neural networks [81].

While we have talked about trainability in general terms now, we have not yet touched on how the training of neural networks works in practice: Fundamentally, the training of neural networks relies on minimizing the loss function from a randomly initialized starting point (discussed in more detail below) via gradient descent using

$$\vartheta \rightarrow \vartheta - \alpha \frac{\partial L}{\partial \vartheta}, \quad (3.18)$$

where α is called the learning rate. Fundamentally, this relies on the same idea of linearly approximating the loss function to iteratively minimize it as the Gradient Boosting discussed in Sec. 3.3. However, it should be noted that the actual optimization procedure still differs significantly between the two, as Gradient Boosting adds corrections to the function while NN gradient descent changes the function's parameters themselves. Additionally, Gradient Boosting adjusts the functional value for each data point, while NN trainings attempt to find the parameter combination that best improves the loss averaged over all data points. To speed up this iterative process, this average is usually not performed for the entire data set at each weight update but instead over a random selection of data points called a *batch*. This is called *batched gradient descent*. If each batch contains only one data point, it is called *stochastic gradient descent* [129]. The size of these batches is a hyperparameter of the training. Typical batch sizes may lie around 256 data points for a standard MLP, though specific applications or other model types may require larger or smaller batch sizes, and variations are always possible. The batch size should strike a balance between lowering computation time, as fewer data points need to be considered, and providing adequate statistics to perform the optimization. Each iteration over the full data set is called an *epoch*. The number of epochs is a hyperparameter. How many epochs are necessary may depend on both the data set size and the complexity of the model and problem.

In addition to batch size and number of epochs, the learning rate is also a crucial hyperparameter of the training and can strongly influence convergence, as can be seen in Fig. 3.7: While a well-chosen learning rate results in quick convergence to the minimum, a very low learning rate leads to very slow convergence and is therefore highly inefficient. A learning rate that is too high, on the other hand, can also hinder convergence as it may result in oscillations around the minimum where the minimum cannot be reached. Therefore, initial learning rates need to be tuned well, and in some cases, additional learning rate schedulers may be necessary, which, for example, reduce the learning rate once the loss plateaus.

In order to perform a gradient descent optimization, the gradients first need to be determined. This is done through *backpropagation* [130], which provides an efficient method of obtaining gradients for all weights and biases simultaneously: For a data point x_i , a forward pass through the network is performed to determine the network output $y_{\text{pred},i}$ as well as all intermediate results in all hidden layer nodes. The differentiable loss function can be evaluated for this prediction and its derivative

$$\frac{\partial L(y_{\text{true},i}, y_{\text{pred},i})}{\partial y_{\text{pred},i}} \quad (3.19)$$

can be determined. Using the chain rule and the intermediate results of the forward pass, the derivatives $\partial_{\vartheta} L$ can then be calculated. As the operation in each node is linear except for the activation function, all derivatives of computation steps within the network can be easily computed so long as those activation functions are differentiable. Resulting derivatives

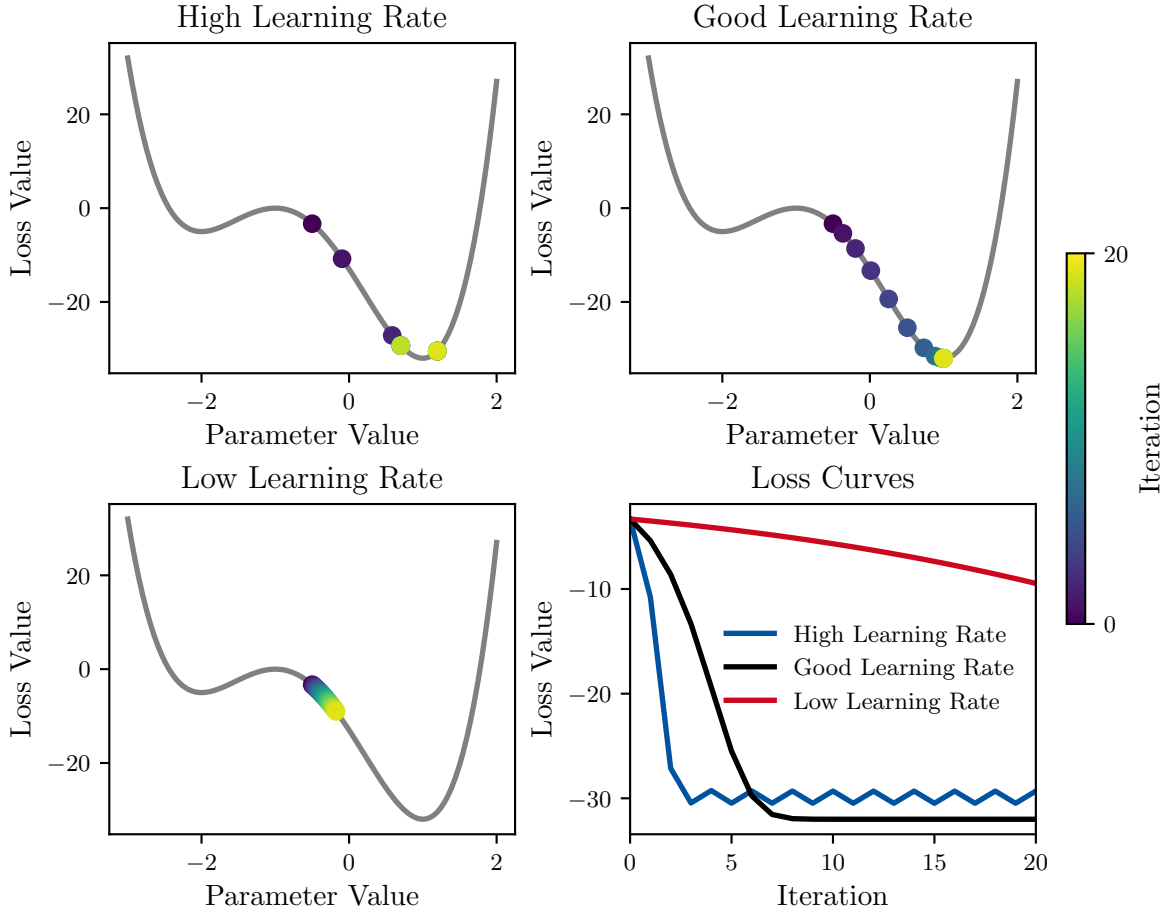


Figure 3.7: Visualization of gradient descent-based parameter optimization on a simple 1D example with three different learning rates. Shown are the iterations of the optimization using a color scale for a high learning rate (top left), good learning rate (top right), and low learning rate (bottom left), as well as the respective loss curves, i.e., loss values over iterations (bottom right).

can then be averaged over multiple data points to obtain batch-wise derivatives. For large numbers of weights, this computation requires $\mathcal{O}(N_W)$ operations, where N_W is the number of weights within the network. This efficiency is crucial for gradient descent training to be viable [118].

While gradient descent is a useful approach, realistic NN trainings are very complex: Not only is the parameter space very high-dimensional, but the loss landscape is also often very irregular. For example, the loss landscape may contain many local minima, in which a simple gradient descent optimization may become stuck. To deal with this complexity, different modifications of the stochastic gradient descent procedure are possible, which may reach faster and/or better convergence. The precise procedure used is called the *optimizer*, of which the most commonly used is Adaptive Moment Estimation (ADAM) [131],

$$\vartheta_t = \vartheta_{t-1} - \alpha_t \frac{1}{\sqrt{s_t + \varepsilon}} m_t, \quad (3.20)$$

where t is the update step and

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \cdot g_t \quad \text{and} \quad s_t = \beta_2 s_{t-1} + (1 - \beta_2) \cdot g_t^2 \quad (3.21)$$

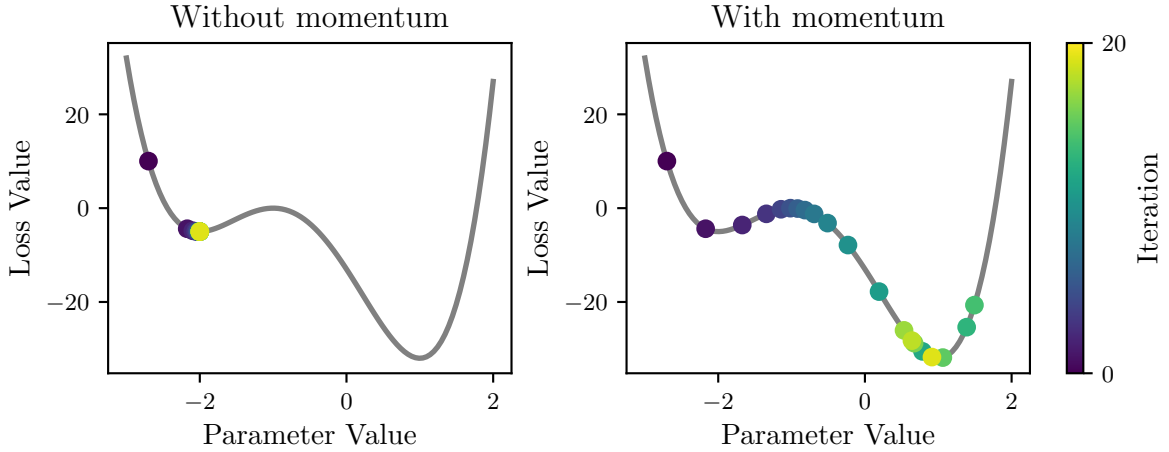


Figure 3.8: Visualization of parameter optimization on a simple 1D example with gradient descent optimization without (left) and with momentum (right), with a local minimum between the initialization value and the global minimum.

are calculated from past and current gradients g_t for each parameter. By default, the constants are set to $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-6}$. This procedure includes two modifications to stochastic gradient descent:

Adaptive learning rates seek to gradually decrease the learning rates depending on the size of previous gradients, individually for each parameter. In Adam, which follows Root Mean Squared Propagation (RMSProp) [132] for this modification, this is done by dividing the learning rate by $\sqrt{s_t + \varepsilon}$, where ε stabilizes the calculation numerically, and $\tilde{s}_t$ is an exponentially weighted moving average over the square of past gradients. The earlier Adaptive Gradient (AdaGrad) [133] used a simple squared sum of past gradients in the same place, which could lead to learning rates decreasing too quickly, hindering the convergence. The influence of past gradients is determined by the hyperparameter β_2 .

Secondly, in addition to the adaptive learning rates, Adam uses momentum to enable the network to take the previous directions of gradients into account. This momentum may be able to push an optimization out of a local minimum, as can be seen in Fig. 3.8. While pure gradient descent becomes stuck in the local minimum, the optimization with momentum sees large gradients before the local minimum is reached and pushes beyond the local maximum into the global minimum. While this cannot fully eliminate the possibility of an optimization becoming stuck, it is able to mitigate the problem significantly. In Adam, the momentum is encoded in Eqs. 3.20 and 3.21 in the first term of $\vec{m}_t$. The influence of past updates can be turned off, and RMSProp recovered by setting $\beta_1 = 0$ [83].

Whether a network becomes stuck can also strongly depend on the optimization's starting point, which itself depends on the initialization strategy. While there is no clearly understood optimal strategy for choosing the *initialization*, there are a few general considerations, which should be taken into account: From an architecture standpoint, each hidden node is equivalent, and the result is fully symmetric under an exchange of two hidden nodes within the same layer. Therefore, for nodes not to become redundant, a random initialization is clearly preferable to initializing all weights and biases to the same values. Usually, weights and biases are sampled from either a uniform distribution in $[-s, s]$ or a Gaussian distribution with mean $\mu = 0$ and standard deviation σ . Centering the distribution around 0 makes sense, as most common activation functions exhibit their largest changes around 0. The scale of the initialization, i.e.,

the choice of s and σ , is key, as a small scale will fail to break the symmetry between nodes, but a very large scale and therefore large weights can lead to exploding values and gradients. A common choice for initialization is the *Glorot normal* [134] with

$$\sigma^2 = \frac{2}{n_{in} + n_{out}}, \quad (3.22)$$

where n_{in} is the number of inputs to the node and n_{out} is the number of outputs. This initialization leads to outputs with a variance of order 1 if the inputs are of order 1. For uniform distributions, choosing $s = 3\sigma$, where σ is the standard deviation chosen for a Gaussian initialization, is sensible to obtain the same standard deviation. Glorot initialization is not ideal for very large networks, for which each weight becomes very small. This can be addressed by using a sparse initialization, where only a limited number of weights connected with a node are initialized to non-zero values. It is also possible to encode information on the task in the initialization more explicitly by, for example, initializing the bias of output nodes such that the output is equal to the mean of the data⁸. In this thesis, initialization is not discussed in detail, but particularly for complex networks and difficult tasks, treating initialization as the hyperparameter it is and tuning it may be helpful [81, 82].

Similarly, *preprocessing* of data is also very relevant to improving the convergence of NN trainings: In nature, observables and, as thus, input features may live on very different scales, both in their orders of magnitudes and in what orders they may change. However, adjusting initialization and the impact on the loss when input features differ wildly may be difficult and a process of trial and error. It is much easier instead to rescale the data features themselves. For example, should energies change logarithmically, a rescaling to $\log(E/E_{\max})$ may be useful. A common rescaling is the *standardization*, meaning

$$x \rightarrow \frac{x - \mu}{\sigma}, \quad (3.23)$$

where μ is the expectation value and σ the standard deviation of feature x . After this, all features have $\mu' = 0$ and $\sigma' = 1$, which not only results in features living on similar scales but also brings the data into those regions where most activation functions have their largest gradients. The relevance of the latter can be easily seen when considering ReLU: Should all features be positive, any node randomly initialized with negative weights will only receive vanishing gradients and be unable to learn. It is effectively removed from the model. Some tasks additionally profit from specific preprocessing. Density estimation, for example, often attempts to reshape distributions onto Gaussian distributions. Therefore, it is useful to rescale features to fall into $[0, 1]$ and then apply the logit function

$$\text{logit}(x) = \ln \frac{x}{1 - x}, \quad (3.24)$$

particularly to features with sharp edges, to smooth the input distributions. This function is bijective with the inverse being the sigmoid (see Eq. 3.17). It should be noted that decision tree-based models are generally invariant under monotonic transformations of the data and do not require any particular preprocessing, as cuts are considered only in terms relative to the data.

The characteristics and working principles of MLPs and other deep learning models touched on so far, such as their efficient computation (particularly on modern GPUs), simple structures, at least theoretical universal approximation, and trainability, make them the powerful tool

⁸ This is a very similar concept to the initial guess for gradient boosting (see Sec. 3.3).

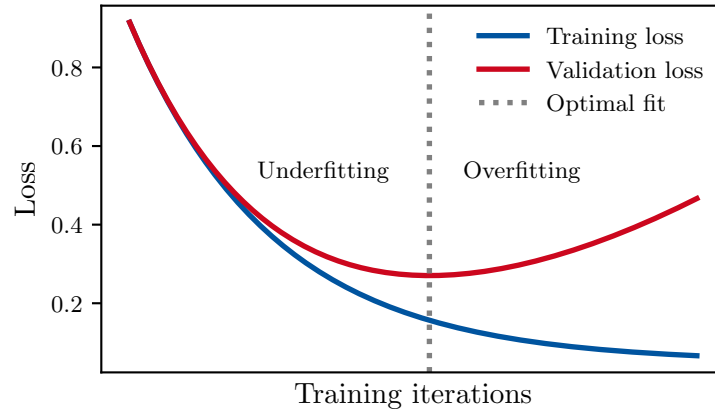


Figure 3.9: Visualization of training and validation loss throughout a training showing both the early underfitting and late overfitting regimes.

they are today. However, as touched on in Secs. 3.1 and 3.3, generalization is key for such tools to be truly useful. Depending on hyperparameters and the task at hand, deep neural networks can be very prone to overfitting. For neural networks, this shows up gradually as can be seen in Fig. 3.9, where we consider both the loss calculated on the training set (training loss) and the loss calculated on an independent validation set (validation loss): At the beginning of the training, a randomly initialized network has both a high training and validation loss. It underfits. As the training progresses, both losses decrease as the network learns the general structure of the data. If the training is continued past the point where the network has learned the easy-to-learn structure in the data, it may begin to learn the training data by heart and start to overfit. The training loss still decreases, but this is not reflected in the validation loss. For this training, an optimal fit is achieved at the lowest *generalization error* (approximated by the validation error) before the overfitting begins. It would be possible to stop the training here when the validation loss begins to rise (*early stopping*) or to revert to the best model according to the validation loss after the training is complete. However, there may be an additional, more general structure to learn in the data, even if a model overfits. For this same reason, it is also usually not recommended to make a network smaller to fight overfitting: Although decreasing the model’s capacity to learn data by heart does reduce its ability to overfit, it also impedes its ability to learn the true structure of the data. Instead, if overfitting is a significant problem, it should be addressed via regularization, which tends to restrict learning to improve generalization.

One method of regularization is *parameter norm penalties*, which are additional terms added to the loss

$$L' = L + \lambda L_{\text{reg}}, \quad (3.25)$$

and punish different types of parameter configurations. In general, this can push a network out of a local minimum and therefore help it generalize. The influence of these penalty terms is modulated by the hyperparameter λ , which must be tuned for any task. *L1 regularization* [135] introduces the penalty

$$L_1 = \sum_{i,j} |W_{i,j}|, \quad (3.26)$$

which tends to push networks towards sparse parametrizations. *L2 regularization* [136]

$$L_2 = \sum_{i,j} W_{i,j}^2, \quad (3.27)$$

punishes large weights particularly strongly, therefore leading networks tend to learn to utilize all nodes relatively equally. Parameter norm penalties can also be applied to gradient boosted decision trees, where they affect leaf values. For L1 regularization, this means that leaves with only a small impact are discounted, while with L2 regularization, the impact of individual leaf nodes is limited.

While working by a different mechanism, dropout, similarly to parameter norm penalties, places additional constraints on network training. Nodes are randomly deactivated with a dropout probability p at each update during training. This forces the network to build redundant pathways, which tend to generalize better [137]. Conceptually, this is based on the idea that neurons may co-adapt, i.e., rely too much on each other's presence [138].

In addition to regularization, there are other methods which can help reduce overfitting. For example, increasing training data statistics gives a better representation of the truth, reducing overfitting. Usually, the available data is limited, and this is not an option, but *data augmentation* may be: If data has known symmetries, such as images being translationally invariant, the same data point can be used to produce multiple data points, which will look different to the network while fundamentally still being valid data. An advantage of this is that the network is also enticed to learn known symmetries of the task⁹. For some tasks, adding noise, particularly when related to measurement uncertainties, to data points may also be a sensible data augmentation. As was already discussed in Sec. 3.3, ensembling several high-variance networks can also decrease the variance, leading to a more stable prediction. For neural networks, this is used far less commonly than for decision trees, as single networks take longer to train, and often choosing a more complex architecture or optimizing the regularization can provide a larger gain in performance. However, especially an ensemble of different network types, which may find complementary information and not overfit in the same way, can outperform even well-performing single networks, as can be seen in Ref. [13].

In general, any of these methods may be used both separately and in tandem. The optimal regularization may be a combination of different measures.

3.5 DENSITY ESTIMATION AND GENERATIVE MODELS

This section is based on Refs. [141–143]. Additional citations, as well as the original papers introducing certain concepts, are included where appropriate.

In this section, we discuss two specific tasks from the unsupervised learning regime, which are crucial for some of the anomaly detection methods discussed in this thesis, namely, Density Estimation (DE) and generative modeling. While many modern architectures – including the ones used in this thesis – are able to do both, the two tasks are fundamentally different: Density estimation is the task of learning the probability distribution $p(x)$ of a given data set. Generative modeling, on the other hand, concerns itself with generating additional data points that match the underlying data distribution. Technically, any density estimator can be sampled from – using Monte Carlo techniques in the worst case – and most modern architectures natively allow for sampling, though computation speed may in some cases be an issue. Generative models, on the other hand, may learn to reproduce data without producing

⁹ Invariances can also be built directly into network structures, which has become increasingly popular, particularly in particle physics, where there are many assumptions on which symmetries should be observed (see, e.g., Refs. [75, 77, 78, 139, 140]).

tractable likelihood functions. Generative Adversarial Networks (GANs) [144], for example, are trained to produce data, which an additional adversarial classifier network cannot distinguish from the original training data. There is by now a whole range of density estimation and generative modeling architectures, and while some have become largely obsolete in recent years, as with classical machine learning, each may still have niche advantages. In particular, older architectures with faster sampling speeds may be advantageous for fast simulations even if more modern architectures are more accurate [90]. Although we require the density estimators used in this thesis for their generative properties, the models we use are fundamentally density estimators, and we discuss them in this context. In particular, the models we use are Flow Matching [145] models, a type of continuous normalizing flows.

Density estimation is a technically complex task because of the requirement that probability distributions have to be normalized to one when integrated over, which is difficult for a general parametrized function (such as an MLP) to learn precisely. Therefore, most density estimation techniques enforce normalization in the model design, which allows the training to focus on learning the actual distribution. Enforcing the normalization can be very simple in classical machine learning. For example, taking the fraction of events per bin in a histogram gives an easy, correctly normalized, although coarse-grained density estimate. Another classical ML method is Kernel Density Estimation (KDE), which places a known distribution, such as a Gaussian or uniform distribution, the so-called kernel, at the position of each data point. The probability distribution is then given by

$$p(x) = \frac{1}{N} \sum_{i=1}^N p_K(x - x_i), \quad (3.28)$$

where N is the number of data points x_i in the data set and the kernel distribution $p_K(x)$ is centered at 0. While KDE works relatively well in low dimensions, it scales badly in higher dimensions. This problem is also called the *Curse of Dimensionality* [146], which goes far beyond both density estimation and machine learning: It describes the fundamental problem that as dimensionality increases, the number of data points must scale exponentially with the number of dimensions to retain the same distance between points and therefore the same statistical accuracy. This affects any algorithms and models working in high-dimensional phase spaces. However, while both are affected, modern deep learning-based density estimation techniques can, by virtue of being parametric models, interpolate well between data points as they seek a shared representation between the full data set.

To set up parametric density estimators, normalization needs to be enforced differently than it is for KDEs. However, the key in common architectures is often still to reuse known probability distributions to produce correctly normalized results. The most recent advances in density estimation, namely normalizing flows [142], diffusion models [148, 149], and flow matching models, all do this by mapping data onto a latent space. In this latent space, the data follows such a known distribution, often a multivariate Gaussian, as can be seen in Fig. 3.10. The model only learns the mapping $f_\theta(x)$ between data space $\mathcal{X}$ and latent space $\mathcal{Z}$. If this mapping $f_\theta(x) = f(x)$, where $f(x)$ correctly maps between $p_{\mathcal{X}}$ and $p_{\mathcal{Z}}$ ¹⁰, the probability distribution of x is given by

$$p_{\mathcal{X}}(x) = p_{\mathcal{Z}}(f(x)) \left| \det \frac{\partial f}{\partial x} \right| \quad (3.29)$$

according to the change of variables formula, where $p_{\mathcal{Z}}(z)$ is the chosen distribution in the latent space and $\det \frac{\partial f}{\partial x}$ is the Jacobian determinant of $f(x)$. This provides a conceptually

10 Note that this transformation is not unique.

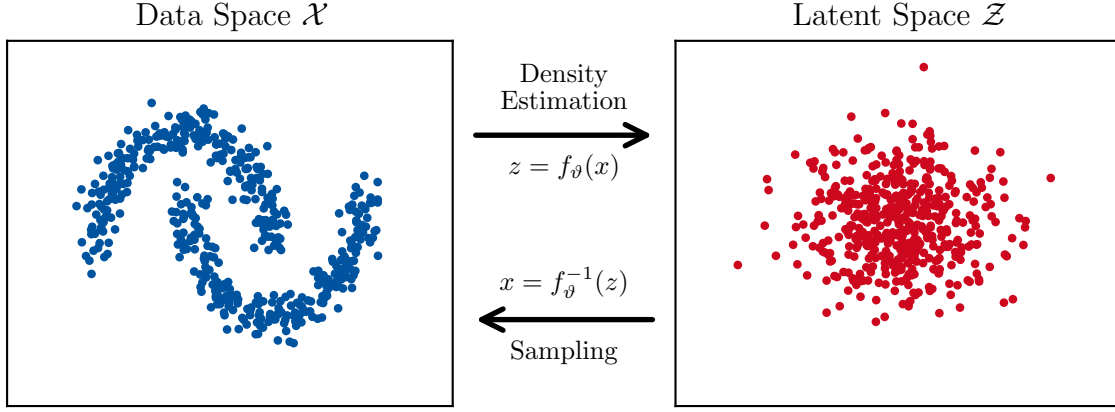


Figure 3.10: General working principle of latent space-based density estimation and generative modeling, where the trained network is the invertible mapping between data and latent space $f_\vartheta(x)$. Figure based on Ref. [147].

simple way of obtaining a correctly normalized density estimate. However, constructing bijective transformations $f_\vartheta(x)$, for which the Jacobian determinant can be calculated, and training them are highly nontrivial tasks. Between the model types based on this latent space density estimate construction, this construction is handled differently. We use only flow matching models in this thesis, so our focus in this section is on them. However, as flow matching is a type of *continuous normalizing flow* and was developed from these earlier architectures, we begin our explanations with a short description of normalizing flows.

3.5.1 Normalizing Flows

Normalizing flows aim to construct the transformation $f_\vartheta(x)$ in a way that allows for a direct computation of the likelihood function under the change of variables formula (see Eq. 3.29). This requires them to provide easy computation of the Jacobian determinant, but it also allows them to directly train to maximize the likelihood – or minimize the negative log likelihood

$$-\log p_\vartheta(x) = -\log p_Z(f_\vartheta(x)) - \log \left| \det \frac{\partial f}{\partial x} \right|, \quad (3.30)$$

as is done in practice. We can show that this minimization is optimal to achieve our actual goal of obtaining ϑ such that $p_\vartheta(x)$ and $p_X(x)$ are equivalent by considering the Kullback-Leibler divergence (KL divergence) [150]. The KL divergence is a similarity measure between distributions, which we apply to the model and target distribution. We write its definition

$$D_{\text{KL}}(p_X || p_\vartheta) = \int_x p_X(x) \log \frac{p_X(x)}{p_\vartheta(x)} dx. \quad (3.31)$$

and then separate its components

$$D_{\text{KL}}(p_X || p_\vartheta) = \int_x p_X(x) \log p_X(x) dx - \int_x p_X(x) \log p_\vartheta(x) dx \quad (3.32)$$

$$\simeq \mathbb{E}_{x \sim p_X} [\log p_X(x)] - \mathbb{E}_{x \sim p_X} [\log p_\vartheta(x)], \quad (3.33)$$

where $\mathbb{E}_{x \sim p} [\log A(x)]$ is the expectation value of $A(x)$ given x is drawn from distribution p . The first term in this equation is a constant and therefore irrelevant in an optimization of the

parameters ϑ . The second term, on the other hand, is the expectation value of the negative log likelihood used as a loss for normalizing flow trainings. Therefore, minimizing $-\log p_\vartheta(x)$ directly minimizes the KL divergence up to the limited statistics available in realistic trainings.

This training procedure, of course, only works if we can quickly and easily evaluate $\log p_\vartheta(x)$, and if we wish to sample, we also need to ensure that $f_\vartheta(x)$ is invertible. While there is a variety of different approaches for this, we discuss this based on Real-Valued Non-Volume Preserving (RealNVP) flows [147], which is a type of *coupling flow* [151]: Coupling flows work by splitting the dimensions into two group and leaving the first group as is, while the second group of features is transformed based on the first. For RealNVP, these transformations take the form

$$z_{1:d} = x_{1:d} \quad z_{d+1:D} = x_{d+1:D} \odot \exp(s(x_{1:d})) + t(x_{1:d}), \quad (3.34)$$

where D is the number of total dimensions, d is the split point between the groups, x_i and z_i are the features before and after the transformation respectively and $\odot$ denotes an element-wise multiplication. The so-called translation and scale functions $t(x_i)$ and $s(x_i)$ are typically represented by deep neural networks, e.g. MLPs. $x_{n:m}$ denotes that features with indices $n \leq i \leq m$ are used. The coupling layer structure here is truly key, as it ensures several advantageous properties at once: Firstly, we see that to invert this transformation, $t(x_i)$ and $s(x_i)$ do not need to be invertible, as they only depend on unchanged features. For the invertibility, it is also key that the multiplication with the scale function uses an exponential function, as this ensures positive scaling factors. Secondly, the coupling layer structure ensures differentiability as the Jacobian becomes a triangular matrix, and as such, only diagonal elements affect the Jacobian determinant. The affine transformation $(Ax + b)$ also means that no complicated derivatives must be calculated. Indeed, for the Jacobian and its log determinant, we obtain

$$\frac{\partial z}{\partial x} = \begin{pmatrix} \mathbb{I}_d & 0 \\ \frac{\partial z_{d+1:D}}{\partial x_{1:d}^T} & \text{diag} \exp(s(x_{1:d})) \end{pmatrix} \Rightarrow \ln \left| \det \frac{\partial z}{\partial x} \right| = \sum_j s(x_{1:d})_j, \quad (3.35)$$

which can be quickly evaluated for MLPs and other deep neural networks. It is possible to retain these advantageous properties of coupling layer flows by moving to autoregressive transformations: Masked Autoregressive Flows (MAFs) [152], for example, keep the transformation structure of RealNVP while using autoregressive layer structures. These autoregressive layers still have a triangular Jacobian, while being more expressive. However, even with clever implementation of the autoregressive structure, they tend to be significantly slower than coupling flows in either the sampling or the density estimation direction¹¹. The scale and translation structure of RealNVP can also be exchanged for other structures, so long as they retain the analytic invertibility and differentiability along the diagonal. This can, for example, be done by using spline flows [154].

In order to express arbitrary distributions, a single coupling or autoregressive layer is clearly not sufficient. However, flow layers can be stacked to perform multiple transformations subsequently with permutations of the features in between, such that correlations in the full D -dimensional feature space can be learned. The Jacobian determinant is then simply given by the sum of each individual transformation's Jacobian determinant. Similarly, inverting the transformation can be done one layer at a time.

¹¹ Masked Autoencoder for Density Estimation (MADE) [153] provides a structure, which can evaluate autoregressive functions in one call in one direction. However, for the inverse direction, D calls are required for D features.

3.5.2 Continuous Normalizing Flows and Flow Matching

While normalizing flows have been very successful, particularly because of the optimality of their loss and their clever, fast-to-train implementations of transformations, they are still limited in their expressivity because of their relatively fixed functional structure. This becomes particularly problematic as data dimensionality increases or distributions become more complex. To address this, Ref. [155, 156] proposed continuous normalizing flows parametrized by a time coordinate t such that

$$\frac{dz_t}{dt} = u_t^\vartheta(z_t) \quad (3.36)$$

with $z_1 = x$ and $z_0 = z$, which can move continuously between data and latent space. Density estimation and sampling can then be performed via

$$z = x - \int_0^1 u_\vartheta(t, z_t) dt \quad x = z + \int_0^1 u_\vartheta(t, z_t) dt, \quad (3.37)$$

which requires numerical integration via ODE solvers, which is possible and accurate but tends to be slow. The log likelihood at x is then given by

$$\log p_1(z_1) = \log p_0(z_0) - \int_0^1 \text{div}(u_t)(z_t) dt \quad (3.38)$$

where z_0 and z_t are calculated from z_1 using Eq. 3.36. Having to solve ODEs to obtain the log likelihood makes training with a log likelihood loss highly impractical due to computational constraints. This is resolved in flow matching, which learns to match the predicted velocity (see Eq. 3.36) to a target velocity.

This works by following the steps illustrated in Fig. 3.11: First, the problem is set up by defining the data distribution (here, $q(z_1)$) and the target distribution (here, $p(z_0)$). The goal is to find a flow, which maps samples z_0 from the latent space distribution p to samples z_1 from the data space distribution q (see top left panel). In order to achieve this, a continuous path between the two distributions p_t needs to be designed, which can interpolate between the distributions (top right panel). This is typically chosen as an optimal transport path such that

$$z_t = tz_1 + (1 - t)z_0 \quad (3.39)$$

With the paths defined, it is possible to train using a regression loss matching predicted velocity fields u_t^ϑ to the target velocity u_t , which can generate path p_t (bottom left panel). This can be done with *Conditional Flow Matching* by taking expectation values over random samples of t , z_0 and z_1 to obtain the loss

$$L_{\text{CFM}}(\vartheta) = \mathbb{E}_{t, z_t, z_0} \|u_t^\vartheta(z_t) - u_t(z_t|z_0)\|^2. \quad (3.40)$$

For optimal transport velocities

$$u_t(z_t|z_0) = \frac{z_0 - z_t}{t} \quad (3.41)$$

and a Gaussian latent space, this becomes

$$L_{\text{CFM}}^{\text{OT, Gauss}}(\vartheta) = \mathbb{E}_{t, z_t, z_0} \|u_t^\vartheta(z_t) - (z_0 - z_1)\|^2, \text{ where } t \sim U[0, 1], z_1 \sim \mathcal{N}(0, I), z_0 \sim q. \quad (3.42)$$

z_t can for any given set of samples $\{t, z_1, z_0\}$ be calculated using Eq. 3.41. By taking the expectation value over many different samples, the trivial optimal transport ansatz for the target velocities results in a non-trivial learned velocity field with parametric interpolation.

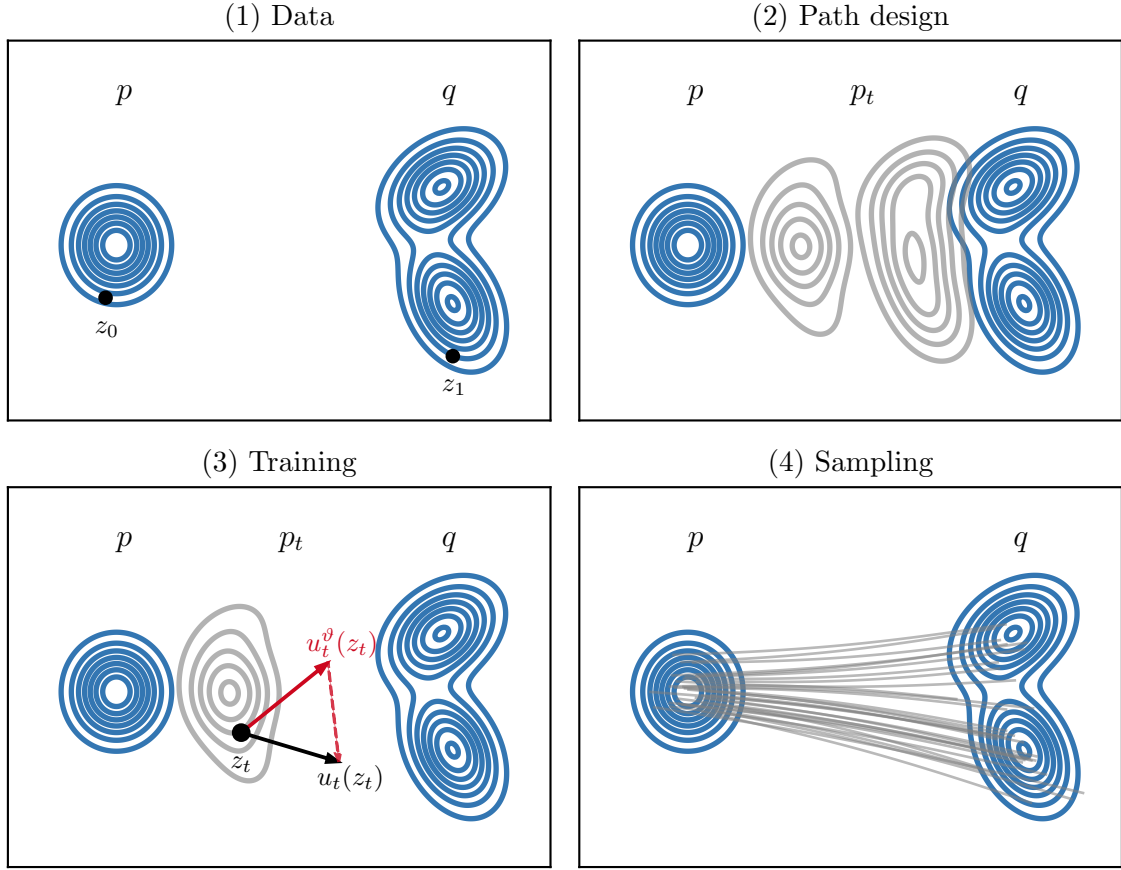


Figure 3.11: Visualization of the Flow Matching training procedure starting with: (1) The data and latent space distribution q and p with samples to match z_1 and z_0 . (2) The design of paths smoothly interpolating distributions by positing p_t between $t = 0$ and $t = 1$. (3) Training performed by matching velocities at all times t to design velocities. (4) Sampling along learned velocity paths. Figure recreated from Ref. [143]

This averaging of linear target velocities is visualized in Fig. 3.12: Here, we consider a single point z_t and the variety of sample pairs $\{z_0, z_1\}$ that contribute to the learned velocity. A point z_0 can only contribute to the velocity if a corresponding point z_1 can be drawn, for which the optimal transport line would cross z_t . Therefore, large regions of the distributions do not contribute to the learned velocity in z_t . This precludes the distribution from learning a trivial solution, where all velocities point to the distribution's mean. After training is complete, new data points can be sampled by sampling the latent space and following the learned paths (bottom right panel of Fig. 3.11).

By training on the velocities, flow matching sacrifices the optimality of training directly on the model's log likelihood. However, the gain in expressivity makes up for this loss of theoretical optimality in practice.

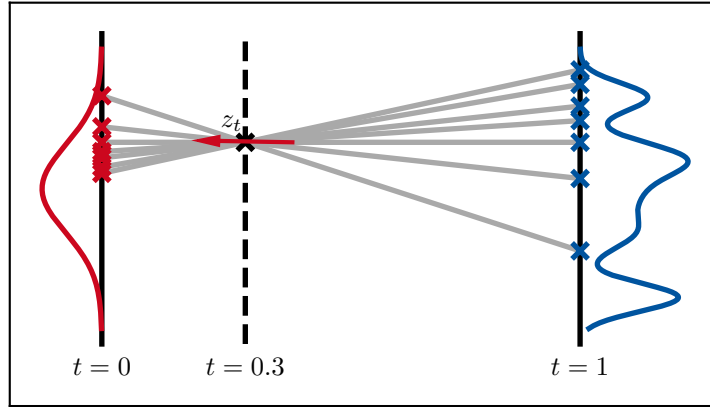


Figure 3.12: Visualization of FM training showing how linear velocities from sample sets are averaged at each point z_t to obtain the final velocity field.

3.5.3 Conditional Density Estimation

When using density estimation in the projects in this thesis, we are interested not just in probability distributions $p(x)$ but in probability distributions $p(x|m)$, which are conditional on a conditioning variable m . This allows for decorrelation and interpolation and, therefore, sampling data points at specific values of m . In practice, this conditional density estimation is straightforward to implement in deep learning models: One simply provides m as an additional input to the networks, which learn transformations such as $s(x_i)$ and $t(x_i)$ for RealNVP (see Eq. 3.34) or the velocities in Flow Matching models (see Eq. 3.36). The training objectives and transformation formulae remain unchanged.

4

WEAK SUPERVISION FOR NEW PHYSICS SEARCHES

Over the last two chapters, the physics and machine learning foundations for this thesis have been built. Based on these, we can now discuss resonance searches and anomaly detection more concretely, starting with more general considerations of resonance searches and how anomaly detection may enhance their sensitivity. We then explain weakly supervised anomaly detection in more detail. We also discuss the design choices that must be made for such an analysis, considering common choices and their impact, as well as recent advances.

4.1 FROM STANDARD RESONANCE SEARCHES TO ANOMALY DETECTION

4.1.1 The Fundamental Principle of Resonance Searches

The fundamentals of resonance searches are based on Refs. [49, 157]. Additional sources are cited where appropriate.

The resonance search is one of the most well-established tools for discovery in particle physics. It has been a staple of new physics searches since its first successful use in the J/Ψ discovery [158, 159] and was the foundation of the Higgs discovery in 2012 [6, 7]. A new particle is produced resonantly if the center of mass energy E_{CM} matches its mass m_R . This resonant production occurs not in an infinitely narrow peak at $E_{\text{CM}} = m_R$ but instead in a peak of finite width. This peak follows the Breit-Wigner [160] distribution

$$\sigma \propto \left| \frac{1}{p^2 - m^2 + im\Gamma} \right|^2, \quad (4.1)$$

with the particle's decay width Γ . When the particle decays, we measure its decay products and can reconstruct the particle's mass by calculating the invariant mass m of the decay products. In this invariant mass, the resonance peak of width Γ is smeared further by the detector resolution. In the case of jet resonances, the jet energy resolution, which is a combination of detector resolutions and the reconstruction process, is dominant. Such resonances do not occur in isolation but instead are observed as bump-like overdensities of events in smooth background distributions, as can be seen in Fig. 4.1.

To detect such resonances, bump hunts model the smooth background and search for any localized overdensities. This is done by dividing the data into a Signal Region (SR) around the mass bump and the Sidebands (SBs) elsewhere. As shown in Fig. 4.1, the SBs contain little signal and can therefore be used to estimate the background via a smoothly falling fit function, which may then be interpolated into and evaluated in the SR. This estimation can then be compared to the data in the SR either by fitting a signal model with the signal strength as a free parameter or by counting the event number in the SR and comparing it to the expectation from a continuation of the smoothly falling spectrum. Practical applications

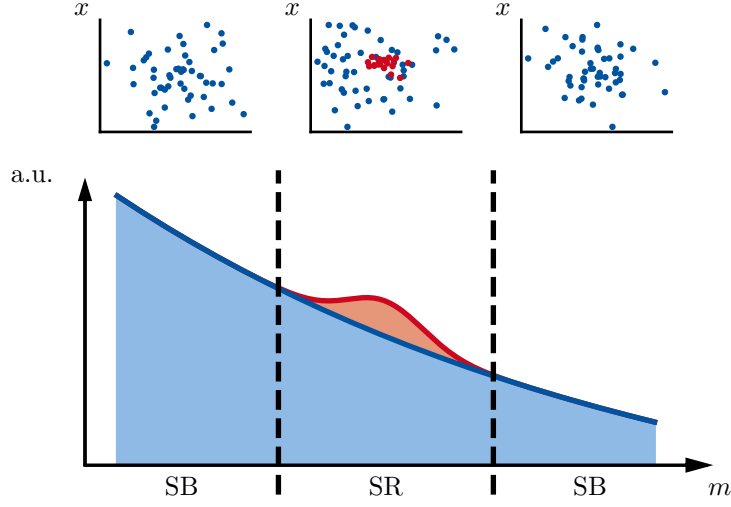


Figure 4.1: Visualization of a bump hunt showing the distribution in a resonant feature m as well as in additional observables x . Figure recreated from Ref. [36].

vary in their precise statistical treatment of resonance searches, and methods that estimate the significance based on likelihood ratios are also used. Because the mass of a new particle is not known beforehand, the location of the SR is varied to scan across the resonant feature.

There are two key choices going into any bump hunt analysis: Firstly, we choose data to perform the analysis in, and as such also choose a particular invariant mass as the resonant feature of the analysis. For example, if we require the data to contain two jets, the dijet mass m_{JJ} is a sensible choice. As was discussed in Sec. 2.3, the high number of jets measured at the LHC and their complex structure make them a common target for new physics searches. Any resonance, which does not decay into the chosen final state, cannot be detected by the search. Additionally, we also choose a signal region width for our analysis. As we assume our signal is localized to the SR, any signal significantly wider than the SR cannot be detected in our analysis. A relatively model-agnostic choice assumes a narrow resonance, in which case the SR width is determined by the known detector resolution.

Such simple, and non-specific searches (also called *inclusive* searches) are commonly conducted to ensure that no obvious new particle is missed (see, e.g., Refs. [161, 162] for ATLAS and CMS 13 TeV inclusive dijet searches). However, their sensitivity is limited, and high statistics are required to observe low cross sections. Therefore, more model-specific searches are also conducted. As we discussed in Sec. 2.3, jets in particular are very common and highly complex objects, for which a lot of auxiliary information exists. This auxiliary information (denoted as x in the top panels of Fig. 4.1) can be used to enhance the sensitivity of a jet-based resonance search. This may come in the form of placing cuts on a single observable, such as the subjeettiness ratio τ_{21} (see Sec. 2.2.2). Such a cut enhances the sensitivity of a search for a resonant particle decaying into two boosted particles, which themselves decay into two quarks ($pp \rightarrow M \rightarrow X(\rightarrow qq)Y(\rightarrow qq)$), in QCD dijet backgrounds ($pp \rightarrow qq/gg$). Both are dijet processes but with strongly differing jet substructures. Such simple cuts are still relatively unspecific as they may target a broad range of models within these decay categories. The sensitivity to a particular model can be further enhanced by specifying additional cuts, for example, based on decay product masses, or by applying neural networks or BDTs trained on

classifying a particular model on simulated data. After cuts have been applied, the general procedure of detecting a possible resonance can remain as discussed above.

Anomaly detection-based cuts can take the same role in the analysis procedure as model-specific cuts. At the same time, anomaly detection introduces fewer additional assumptions on either signal or background structure. In the next section, we discuss the range of anomaly detection methods available for this purpose.

4.1.2 The Range of Anomaly Detection Methods

Anomaly detection improves resonance searches by automatically identifying anomalous patterns in the data, rather than relying on model-specific cuts. The goal is to obtain an anomaly score, a cut on which enhances a signal if present or retains the smoothly falling background spectrum if not. To design such an anomaly detector, the first step is to define what is counted as anomalous.

One of the most model-agnostic types of approaches relies on outlier detection, i.e., defining rare events as anomalies. Generally, outlier detection methods are based on unsupervised learning techniques. For example, a simple outlier detection method is based on isolation forests [163], a decision tree-based technique that isolates data points and computes an anomaly score based on the number of splits required to isolate them. Another relatively general method in this space is autoencoder-based outlier detection (see e.g. Ref. [14] for HEP autoencoder AD). Autoencoders [164] are networks with a *bottleneck*, which compress data into a low-dimensional latent space to then decompress it back to the original size. For anomaly detection, they are trained to reconstruct their input. The autoencoder learns to reconstruct events less accurately if they are rare in the training data. The events with the worst reconstruction (assessed, e.g., by the mean squared error between input and output) are then considered the most anomalous. One issue with the autoencoder-based approach is that the reconstruction performance is biased with respect to the complexity of a data point: More complex events (e.g., top jets compared to QCD jets) may have a worse reconstruction even if the model is trained on them [165].

On the other end of the spectrum lie semi-supervised approaches, which rely on some signal simulations to learn potential anomalous signatures. This is, for example, the case with Quasi Anomalous Knowledge (QUAK) [15], which trains classifiers on a background-versus-signal task for several distinct signal samples. The anomaly score is highly controlled in this scenario as there are clear templates for possible anomalies. On the other hand, such semi-supervised approaches may struggle to identify truly unexpected signal signatures and can be vulnerable to data-MC disagreement.

On the spectrum of model-specificity/-agnosticism, many different anomaly detection methods lie between these two extremes. A key class of methods explicitly uses the physics assumptions on which the analysis as a whole is built. For example, in the case of resonance searches, this may mean designing anomaly detectors based on the bump hunt assumption that the signal is highly localized. Weakly supervised [16] methods, for example, use noisy labels to perform a classification, and for resonance searches typically use the bump hunt assumption to assign the labels (more detail below in Sec. 4.2). Other methods such as Anomaly Detection with Density Estimation (ANODE) [33] and Residual ANODE (R-ANODE) [166] rely on the same assumption but do not use classifiers. Building clear physics assumptions into the anomaly

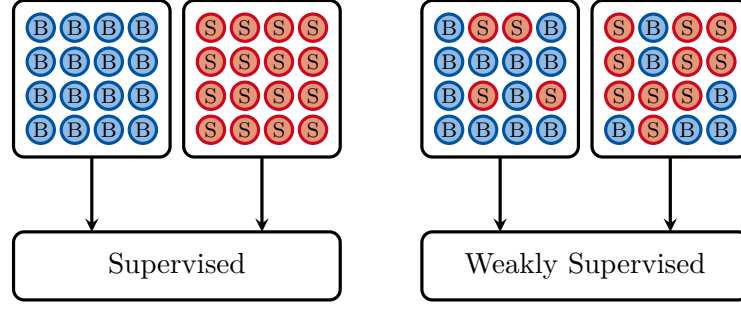


Figure 4.2: Visualization of both supervised (left) and weakly supervised (right) class labels used for a classification training.

detector’s design can help obtain well-controlled and understandable results while avoiding strong additional signal assumptions as well as reliance on simulation.

The focus of this thesis lies in the improvement of weakly supervised methods. Therefore, we discuss this class of methods in more detail in the next section. A more complete list of anomaly detection methods and applications in particle physics can be found in the HEP ML Living Review [167].

4.2 WEAKLY SUPERVISED ANOMALY DETECTION

Fundamentally, anomaly detection, at least in the new physics search sense, is a classification task: the goal is to separate background events following $p_B(x)$ and signal events following $p_S(x)$. Therefore, as discussed in Sec. 3.2, the ideal solution could be obtained by a supervised classifier as seen on the left of Fig. 4.2, which approximates the likelihood ratio

$$R_{\text{opt}}(x) = \frac{p_S(x)}{p_B(x)} \quad (4.2)$$

in the classification features x . In the anomaly detection scenario, training a supervised classifier is not possible, as no labeled data is available. However, it is possible to instead perform a classification between two mixed data sets drawn from distributions

$$p_i(x) = f_i p_S(x) + (1 - f_i) p_B(x), \quad (4.3)$$

where $i = 1, 2$ and f_i is the respective data set’s signal fraction, as shown in the right panel of Fig. 4.2. For a classifier trained on these mixed data sets, the optimal solution is given by

$$R_{\text{mixed}}(x) = \frac{p_1(x)}{p_2(x)} = \frac{f_1 R_{\text{opt}}(x) + (1 - f_1)}{f_2 R_{\text{opt}}(x) + (1 - f_2)}. \quad (4.4)$$

So long as $f_1 > f_2$, $R_{\text{mixed}}(x)$ is a monotonically increasing function of $R_{\text{opt}}(x)$ and as such makes the same decisions. In this case, data set 1 is called a signal-enriched data set as it has the larger signal fraction, while data set 2 is called a background-enriched data set. This method of mixed data set classification is called Classification Without Labels (CWoLa)¹ and was first introduced for HEP application in Ref. [16].

¹ Pronounced “koala”.

In order to make use of the optimality of mixed data set classifications for anomaly detection, methods for deriving mixed data sets from data need to be obtained. For resonance searches, the division of data into signal region and sidebands – as described in Sec. 4.1.1 and shown in Fig. 4.1 – can be the basis of the mixed data sets. Under the bump hunt assumption, the sidebands contain no signal, while the signal region is a mixed data set containing both signal and background. The signal region data can be used directly as a signal-enriched data set, while the sidebands are used to derive a background template, for which $p_{BT}(x) \approx p_B(x|m \in SR)$. In the background-only scenario, a classification between two equivalent data sets is performed, and a random chance classifier is obtained, at least if the background template is perfect. In practice, deriving an accurate background template is a highly non-trivial task, and we discuss a few different methods for this below in Sec. 4.2.1. It should be noted that weakly supervised anomaly detection methods for resonance searches, which construct their mixed data sets differently, also exist, for example, in the form of Tag ‘N’ Train (TNT) [168].

4.2.1 Background Template Generation Methods

The first proposal made to obtain a background template, CWoLa Hunting [31, 32], is relatively simple: Assuming that correlations between the classification features x and the resonant feature m are small enough to be neglected, $p_B(x|m \in SB) = p_B(x|m \in SR)$ is true. Therefore, the SB data, for which $p_{data}(x|m \in SB) = p_B(x|m \in SB)$ may be used directly as the BT. To limit the impact correlations with the dijet mass have on the CWoLa Hunting performance, one typically uses Short Sidebands (SSBs), i.e., only narrow regions around the SR instead of the full SBs. Nevertheless, the performance can be limited by even relatively weak correlations, as well as break down fully if more strongly correlated features are used at all.

Beyond CWoLa Hunting, there is a wide range of different approaches which aim to improve the quality of the background template. In particular, such methods typically seek to account for correlations between classification features and the resonant feature. This enables the use of features which are correlated with the dijet mass in the classification. To understand the range of available methods and their respective advantages and disadvantages, it is useful to consider a few different ways such methods may be categorized:

Firstly, methods may make use of SM background simulations or be fully data-driven. Simulation-assisted methods such as Simulation Assisted Likelihood-Free Anomaly Detection (SALAD) [34] or Flow-Enhanced Transportation for Anomaly Detection (FETA) [39] aim to learn a mapping between simulation and data in the SBs and apply this in the SR to obtain SM background events. This results in SM background events at the correct SR mass values. To learn the mapping SALAD uses a classifier-based reweighting, while FETA is based on deep learning density estimation with normalizing flows. With the use of simulation, one obtains a starting point that is already relatively close to data distributions. However, biases may be introduced at the same time.

Secondly, there is a notable distinction between methods, which are density estimation-based, and those which are not. Beyond FETA, there are many other methods which use density estimation to obtain background templates. Generally, such methods tend to be highly accurate – at least for a small number of features – as they are able to learn complex correlations between features. However, they also suffer significantly from the curse of dimensionality (see Sec. 3.5) and therefore tend to be limited to relatively low-dimensional feature spaces.

Method Name	Possible on data	Uses simulation	Uses DE	Uses morphing	Description
IAD [36]	✗	✓	✗	✗	Use extra simulation
CWoLa Hunting [31]	✓	✗	✗	✗	Use SSBs as BT
CATHODE [36]	✓	✗	✓	✗	Train DE for $p(x m)$ on SBs and interpolate into SR
SALAD [34]	✓	✓	✗	✗	Train classifier to reweigh simulated events
FETA [39]	✓	✓	✓	✓	Train DE to transform between simulation and data
CURTAINS [37, 169]	✓	✗	✓	✓	Learn to transport events between different m values, transport from SBs into SR
LaCATHODE [38]	✓	✗	✓	✗	Train DE as with CATHODE, train classifier in latent space
RAD-OT [42]	✓	✗	✗	✓	Use optimal transport to move events between SBs and stop halfway
TRANSIT [43]	✓	✗	✗	✓	Train network adversarially to transport events between different m values, move to SR mass values

Table 4.1: Comparison of different background template generation methods for CWoLa resonant anomaly detection, including whether they can be performed on data, use simulation (could introduce simulation-dependence), use DE (typically slower), whether they use feature morphing, and a short description. Note that this list is not complete. Template generation methods used in this thesis are marked in red.

Thirdly, methods may be divided into those that use feature morphing or those that learn distributions from scratch. FETA, for example, learns to morph simulation into data. Similarly, Constructing Unobserved Regions by Transforming Adjacent Intervals (CURTAINS) [37, 169] learns to morph events from the sidebands from one mass value to another. To obtain the background template, events can then be morphed from the sidebands to mass values in the SR. Classifying Anomalies Through Outer Density Estimation (CATHODE) [36], on the other hand, learns the conditional probability distribution $p(x|m)$ from scratch based on the sideband data. This distribution is then interpolated into the SR to sample at mass values there. Technically, feature morphing requires the network to learn only relatively small changes, which should be easier than learning from scratch. However, with current state-of-the-art density estimators, no fundamental performance difference has been found. Additionally, feature morphing typically requires slightly more involved setups. However, feature morphing can more easily be performed by methods not based on deep learning, such as the optimal transport method in Resonant Anomaly Detection with Optimal Transport (RAD-OT) [42], allowing for shorter computation times.

All of these methods have in common that they aim to provide a data set that approximates the background distribution in the SR. When performing proof-of-concept studies on simulation, it is possible to obtain an idealized version of this approximation by using additional background

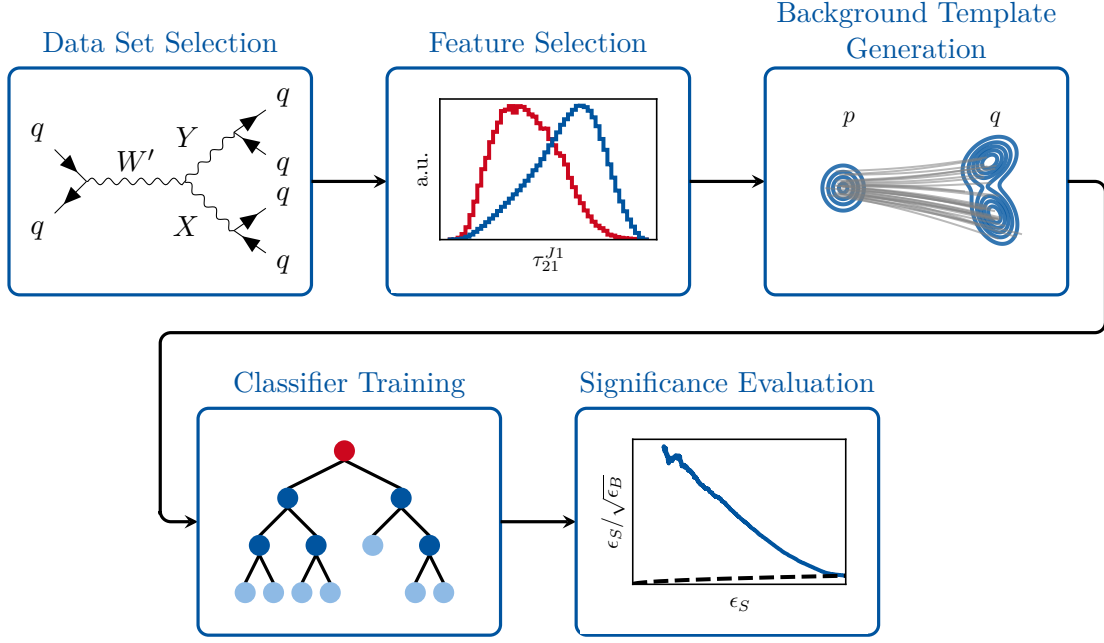


Figure 4.3: Schematic representation of the pipeline of weakly supervised anomaly detection.

simulation in the SR. This Idealized Anomaly Detector (IAD) was introduced in Ref. [36] to provide a realistic performance optimum. The IAD can be useful to study and improve specific components of a weakly supervised analysis, which are independent of the background template quality, in isolation. In an analysis on real data, the IAD cannot be used.

Tab. 4.1 provides a quick reference summary of the background template generation methods mentioned here, as well as a few additions. A more complete list of different background template generation methods can be found in Ref. [167].

4.3 DESIGN CHOICES FOR WEAKLY SUPERVISED SEARCHES

When setting up a search for new physics based on weakly supervised anomaly detection, there are a number of design choices. Such choices may be limited by the available technology – for example, if using a DE-based background template, the feature number may be limited by the state-of-the-art DE performance –, statistical considerations, or similar. We discuss the required design choices based on the workflow of weakly supervised searches, as shown in Fig. 4.3: First, the data set must be selected, then classification features must be calculated. Next, a background template must be obtained, a classifier trained, and lastly, the significance of the results must be evaluated. In the course of this discussion, we seek to highlight standard choices in the literature as well as recent advancements. However, this overview does not aim to be complete, and additional research directions can be found in Ref. [167].

4.3.1 Data Set Selection

The data set selection is intricately connected to the scope of the analysis as we choose a final state, resonant feature, and signal region width. For early proof-of-concept studies and the CMS [46] and ATLAS [44, 45] weakly supervised searches, a dijet resonance with a dijet mass m_{JJ} as the resonant feature was chosen. The signal region width was chosen according to the detector’s mass resolution. As such, these analyses were limited to narrow resonances with decays to this final state. Many of the proof-of-concept studies have relied on the LHC Olympics (LHCO) 2020 R&D data set, which continues to be used to develop new methods and research the behavior of existing methods. This data set is also the basis of the studies summarized in this thesis.

The LHC Olympics 2020 were an anomaly detection challenge created to provide clear benchmarks for anomaly detection. It was made up of a fully labeled data set for research and development (the LHCO R&D data set) and three black boxes to evaluate the anomaly detection performance of the developed methods. The R&D set contains 1 million QCD dijet background events and 100 thousand BSM signal events. The BSM signal is the resonant production of a W' boson decaying into new X and Y particles, which themselves decay into two quarks each. The tree-level Feynman diagram for this process can be seen in the “Data Set” box of Fig. 4.3. The new particles have masses of $m_{W'} = 3.5$ TeV, $m_X = 100$ GeV and $m_Y = 500$ GeV. The data was produced using `Pythia` 8.219 [57] and `Delphes` 3.4.1 [60], and both production cards and events are available in Ref. [18]. The same signal structure with a decay into three quarks of the X and Y particles, i.e., $X/Y \rightarrow qqq$, was added to Ref. [18] later. Under the hood, the W' decay products X and Y are Z and W bosons, respectively, with scaled masses. Events are selected using a single anti- k_T jet trigger with jet radius $R = 1$ and a p_T threshold of 1.2 TeV. Ref. [18] provides the selected events as both low-level information characterizing the reconstructed particles and as high-level information calculated from the jet constituent information. The left panel of Fig. 4.4 shows the invariant mass histogram for background and signal, where we can see the p_T trigger turn-on and the resonant structure of the signal. The low-level features assume massless particles in the event and provide (p_T, η, φ) for 700 particles per event with zero-padding. For the jet-level information, an anti- k_T jet clustering with $R = 1$ is performed, selecting the top two jets by p_T and providing four-momenta as well as the subjettnesses (see Sec. 2.2.2) with $N = 1, 2, 3$ for both jets. For this, the default $\beta = 1$ is chosen, clustering is performed via k_T clustering with one optimization step, and the `UnnormalizedMeasure` as provided via the `Nsubjettness` package in `fj-contrib` is selected as described in Sec. 2.2.2.

Typically, the signal region for this signal is chosen to have a width of 0.4 TeV and is centered on $m_{W'}$. The location of this signal region can be seen in the right panel of Fig. 4.4. Most signal events (approximately 75%) fall into the defined SR. Of the 1 million QCD background events, approximately 120 thousand fall into this region. In this signal region, Ref. [170] provides an additional 612 858 events with the same information available, which was generated for Ref. [36]. For proof-of-concept studies, a default setting uses 1000 signal events and the full 1 million background events from the R&D data set. In the SR, this results in an initial significance of approximately $S/\sqrt{B} = 2.2\sigma$. Therefore, a simple inclusive resonance search would not be able to detect this signal with a discovery-level significance. The sensitivity of different methods and setups is often also tested by varying the injected signal amount.

The three black boxes are designed as follows: The first black box contains a signal which only differs from the R&D data set in the particle masses and the detector card. The second

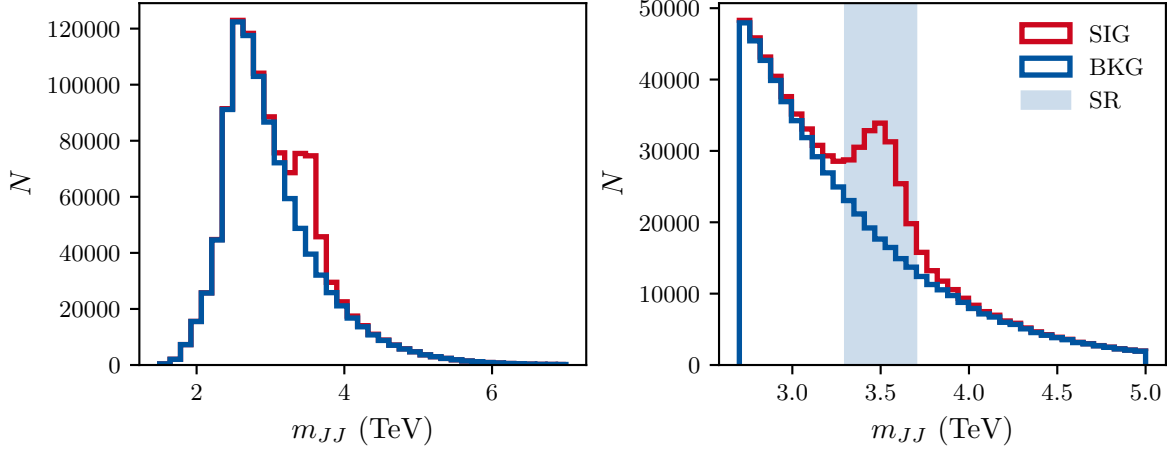


Figure 4.4: Stacked histogram showing the signal (red) and background (blue) distribution in the resonant dijet invariant mass of the LHC R&D data set as the full spectrum including trigger turn-on and tail (left) and focused on the resonance (right). Note that this figure shows all 100 thousand signal events and is not a realistic anomaly detection scenario.

black box contains only background, generated with the same detector card in `Delphes` but with the parton shower and hadronization simulated with `Herwig` [58, 59]. The third black box contains a much more challenging signal with a mixture of di- and trijet decays. This signal was not detected successfully by any of the methods at the time. All black box data is available in Ref. [171], which contains events released for the competition, as well as labels and run cards used to produce the events, which were made public after the challenge was closed.

In recent years, efforts have been made to move beyond the LHC R&D data set, as well as beyond resonances in general. The conducted ATLAS [44, 45] and CMS [46] searches, of course, move beyond proof-of-concept and onto real data. Neither search was able to find new physics. However, both searches set limits on particular BSM signal models, with the ATLAS searches focusing on a rather R&D data set-like signal, while the CMS search considers a broader range of dijet resonant signals. A number of additional benchmark signal models based on the CMS search signals were generated for use with the LHC R&D data set in Ref. [5]. The common final states of narrow dijet resonances used in these studies and searches limits their breadth. Efforts to broaden the horizon of weak supervision have explored a variety of directions, such as anomalous signatures in tails [20], mono-jet searches [21], 2D resonances [22], or other final states [19]. Based on CMS open data, a rediscovery of the Υ has also been conducted [47]. Several searches for stellar streams have shown the applicability of weak supervision beyond particle physics [172–174].

4.3.2 Feature Choice

In order for the significance of the resonance to be enhanced, the weakly supervised classifier must have access to features in which the signal and background are distinct. To obtain sensitivity across a broad range of models, this must be the case for a variety of different signal signatures. In general, this means that a model-agnostic anomaly detection analysis aims to use a broad set of features and let the anomaly detector decide which are important. Ideally, an anomaly detector would indeed use low-level event information and learn to construct

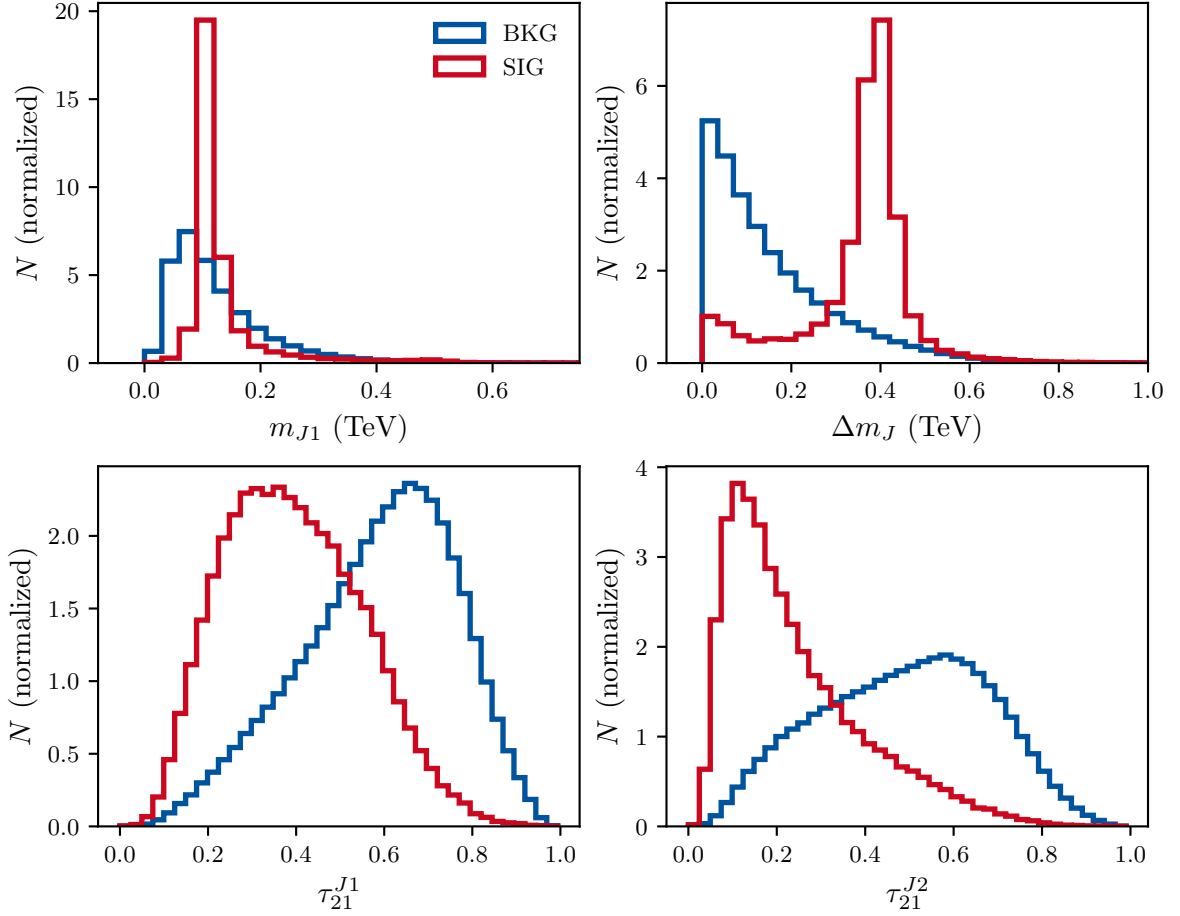


Figure 4.5: Baseline classification features for the LHC R&D data set. The two highest p_T jets are sorted by mass and the mass of the lighter jet m_{J1} (top left), the difference of jet masses Δm_J (top right) as well as the 21-subjettiness ratios τ_{21} for jet 1 (bottom left) and for jet 2 (bottom right) are shown as normalized histograms for signal and background events.

its own features from that. However, as of now, such setups typically are unable to detect relatively low signal amounts [27, 28, 30], which a setup using high-level features may still be able to discover. This is so far also the case when models are pretrained on larger SM data sets [29] but may change with future work, particularly with larger data sets.

With a large set of high-level features, reaching a broad sensitivity is also possible. However, early proof-of-concept studies on weak supervision focused on a limited feature set of only four high-level features (used for several studies in the LHC challenge [17]). This feature set sorts the two jets with the highest p_T by mass and uses the mass of the lighter jet m_{J1} , the difference of the jet masses Δm_J , as well as both jets' 21-subjettiness ratio $\tau_{21}^{J1/J2}$ for classification. Fig. 4.5 shows the distribution of signal and background events in these four features. One can clearly see the X and Y resonances in the mass features as well as the clear distinction between the one-pronged QCD background and the two-pronged signal events in the subjettiness features. While this feature set is very well equipped to detect the LHC R&D data set signal, it does not provide a broad sensitivity. This is acceptable for simple proof-of-concept studies, but not ideal for an analysis. For this reason, the choice of feature sets has been an active research direction in recent years, with different papers adding more subjettiness features [1] (discussed in Sec. 5.1), using energy flow polynomials [5], deriving

new high-level features via contrastive learning [26], or, as is mentioned above, working with low-level features [27–29]. Studies working with different signal topologies typically also consider different classification features to remain sensitive to the different signal (see, e.g., Ref. [19, 20, 22, 47]). This is also the case for the CMS analysis [46]. However, in many of these scenarios, the feature set size is limited significantly by architecture limitations, both for the background template generation and for the classifier training. Additionally, if using CWoLa Hunting, correlations of the classification features with the resonant mass need to be carefully considered.

4.3.3 Background Template Generation

The background template is a crucial component of a weakly supervised analysis and impacts the overall performance. We already discussed above how the method chosen to generate the background template may impact the feature choice. In addition, the quality of the background template significantly impacts both the classifier training – if the classifier is more easily able to detect the difference between BT and SR background than between background and signal, the anomaly detection performance decreases – and the significance evaluation by increasing systematic biases. Beyond this, for DE-based methods, computation time can also be a relevant factor as limit setting and the scan over SR locations may require training many DEs.

Several of these limitations have been addressed in recent studies: Computational limitations may be reduced by using faster generation methods (e.g., RAD-OT [42] or TRANSIT [43]) or by training only one DE for the full scan, as is possible with Single Interpolated Model for Anomalies (SIGMA) [41]. The quality of DE-based BT generation can also be improved by using state-of-the-art density estimators such as diffusion models [27, 28] or flow matching models [27, 41]. This quality improvement can enable the inclusion of more features while retaining a similar BT quality, including a possible use of low-level features [27].

In this thesis, we work with the IAD for method testing as well as CWoLa Hunting and CATHODE for realistic template generation. Generally, it can be expected that the CATHODE performances discussed in this work should also be achievable with other DE-based methods.

4.3.4 Classifier Training

The classifier’s sensitivity is key to the sensitivity of the analysis as a whole. However, with the number of signal events in data being low and studies on simulation being relatively model-specific, optimizing the classifier setup can be difficult. Indeed, the ATLAS and CMS analyses used hyperparameters from LHCO R&D data set studies. In general, these setups tend to be relatively uniform using a simple three-hidden-layer MLP trained using ADAM (see, e.g., Ref. [36]).

Studies concerning the classifier architecture are a central part of this thesis, discussed in Ch. 5. This includes both the study of BDTs for anomaly detection in Sec. 5.1 based on Ref. [1] (also studied in Ref. [23]) and how classifier optimization may be done on data in Sec. 5.2 based on Ref. [3]. In addition to this work, recent years have in particular seen studies of classifiers for low-level features, both training from scratch [27, 28, 30] and with pretraining [175].

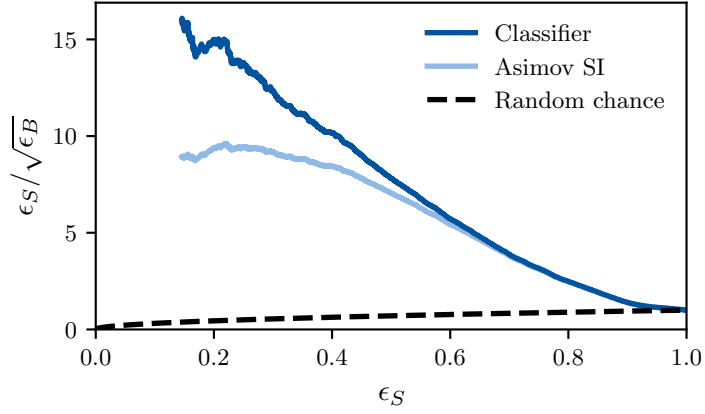


Figure 4.6: SIC curve for a BDT classifier trained in an IAD setting on baseline features with a signal injection of $N_{sig} = 1000$ signal events. In addition, both a random chance classifier and the Asimov estimate for the significance improvement are shown. A statistics cutoff at a 20% relative statistical error on the number of background events is applied.

4.3.5 Performance and Significance Evaluation

The final step of a weakly supervised analysis is evaluating the significance. As this step is highly non-trivial – take, for example, the estimation of systematic errors – many anomaly detection papers simply compare methods by their classification power using ROC curves (see Sec. 3.2) or Significance Improvement Characteristic (SIC) curves. The SIC curve is a simple reformulation of the ROC curve, plotting

$$\text{SIC} = \frac{\varepsilon_S}{\sqrt{\varepsilon_B}}, \quad (4.5)$$

which is the significance improvement in the Gaussian limit, against ε_S . This Gaussian limit is only valid if $B \gg S$, where B and S are the number of background and signal events after the cut. For most anomaly detection classifiers on the LHC0 R&D data set, this is not true for low signal efficiencies ε_S , and the true significance improvement is significantly lower than the SIC suggests. This true significance improvement can be calculated by dividing the Asimov estimate [176] of the significance

$$\mathcal{S} = \sqrt{2[(S + B) \ln \left(1 + \frac{S}{B}\right) - S]} \quad (4.6)$$

after the cut by the initial significance. Both the SIC and this Asimov significance improvement are shown in Fig. 4.6. One can see that the curves match at high signal efficiencies and diverge as the cut becomes tighter. A statistics cutoff limiting the relative statistical error on the number of background events to 20% is applied to both curves to limit fluctuations in the low efficiency regime.

If a full analysis is to be conducted, a cut on the classifier score is chosen. Typically, this cut selects between 1% and 10% of the data (see Refs. [44–46]) to retain sufficient statistics for the background fit. However, it should be noted that weakly supervised anomaly detectors typically achieve their highest significance improvement for tighter cuts. Ref. [47] used a reweighting-based strategy to avoid making a cut and found this to improve the anomaly

detection performance. After the cut and reweighting, a significance can be obtained via fits, as is done in a regular bump hunt.

However, the fits themselves have proved somewhat problematic in typical weakly supervised bump hunt setups: As the classifier is largely unconstrained in the score it learns, that score may be correlated with the dijet mass. In particular, the classifier is typically not trained on any SB data and may as such not extrapolate well to this region. A cut on the classifier score can then lead to *background sculpting*, i.e., a change of the background shape. This sculpting can make the fit more difficult, requiring a more flexible fit function. The more flexible fit function can more easily fit the signal and therefore lower the detection significance. This problem was addressed with Latent CATHODE (LaCATHODE) [38], which trains the weakly supervised classifier in the latent space of a CATHODE-style density estimator. Because the DE is conditional in the resonant feature, the latent space is decorrelated from the resonant feature and, therefore, so is the classifier score.

In Ch. 6, we discuss two projects concerning the significance evaluation: Firstly, to avoid the issue of background sculpting, we introduce a fully cut and count-based approach for significance extraction in Sec. 6.1, which is based on Ref. [2]. Secondly, we investigate whether anomaly detection searches incur a look-elsewhere effect in Sec. 6.2 based on Ref. [4].

The classifier is *the* crucial component of a weakly supervised anomaly detection method. While different methods exist to obtain a background template, every weakly supervised analysis requires a classifier. As such, its sensitivity, particularly to small amounts of signal, strongly impacts the discovery potential of a weakly supervised analysis. In this chapter, we discuss two projects aiming to improve the classifier’s performance: Firstly, in Sec. 5.1, we discuss how boosted decision trees can provide a more robust anomaly detection than the standard MLPs used previously and therefore allow the use of larger, high-level feature sets. Secondly, in Sec. 5.2, we discuss how the classifier can be selected and optimized in a data-driven way, reducing the reliance on simulation and benchmark signals for setup tuning.

5.1 TABULAR DATA AND BOOSTED DECISION TREES

This section is based on Ref. [1]. The code for this project can be found in Ref. [177]. Some additional research, which was not included in the original publication, is added here. Any additional research is marked as such in figure captions.

Early proof-of-concept studies for weakly supervised anomaly detection largely worked with small, model-specific feature sets to show that they were able to detect potential BSM signal models. Such feature sets cannot provide the model-agnostic sensitivity to broad regions of BSM model space that anomaly detection seeks to provide. However, including more features proved difficult as the MLP classifier’s performance dropped significantly when adding relatively uninformative features – i.e., features which do not provide a large distinction between background and signal. This is a well-known property of MLPs and other deep learning models and is strongly connected to the concept of tabular data discussed in Sec. 3.3: If features are tabular, i.e., if they are defined to be informative on their own, then decision boundaries are likely to not be highly correlated between multiple features. The structure of MLPs, which relies on matrix multiplications with randomly initialized matrices, is largely rotationally invariant. As such, they are unable to use the information inherent in the orientation of the feature space. Instead, the MLP must learn the optimal orientation of features from scratch. MLPs can do this well if sufficient data is available. Decision tree-based models, such as gradient boosted decision trees, on the other hand, directly make use of the feature structure in the data set and do not perform feature engineering. While this can be disadvantageous for some data sets, it sets them up for success if data is tabular and the data set is small [178].

For weakly supervised anomaly detection, both points required for decision tree-based models to be optimal are typically true: Firstly, high-level features are fundamentally tabular data. They are often designed to enhance signal sensitivity with single feature cuts (see, for example, 2-pronged analyses performing cuts on the subjettiness ratio τ_{21} as described in Sec. 4.1.1). Secondly, while the full data set in particle physics anomaly detection is not small (typically

at least 200 000 data points in this thesis), the number of signal events is small (less than a thousand). For anomaly detection to be useful, this must necessarily be the case as larger signal fractions could also be detected by an inclusive search.

For these reasons, BDTs are a strong contender to solve the problems with the inclusion of additional features observed for MLPs (shown below). We perform a number of tests on this using the Idealized Anomaly Detector (IAD), which uses a simulation background sample as its background template (see Sec. 4.2.1). This allows us to focus on the classifier performance without being affected by imperfect background templates. For these tests, we begin by considering truly uninformative features drawn from the same Gaussian distributions for signal and background in Sec. 5.1.2. Here, we also highlight the crucial aspects of our setup that lead to the strong performance we observe, namely the tabular nature of our features and the use of ensembling. To show that this behavior is relevant in realistic anomaly detection scenarios, we move on to larger physics-motivated feature sets in Sec. 5.1.3, for which we also see a gain in the model-agnostic sensitivity when testing an additional decay channel of the same BSM signal. Additionally, we show how these larger feature sets can improve the sensitivity to smaller signal amounts in Sec. 5.1.4, where we also investigate the dependence of the performance on the overall data set size. Lastly, to show how larger feature sets impact classifier performance for realistic background templates, we test CWoLa Hunting and CATHODE for our extended feature sets in Sec. 5.1.5. Before any such tests are discussed, however, we describe the setup that underlies them.

5.1.1 Setup

Data Sets

Our tests rely on the LHC0 R&D data set [18], which was introduced in Sec. 4.3.1 and contains a signal of the structure $W' \rightarrow X(\rightarrow qq)Y(\rightarrow qq)$. In addition to this 2-pronged jet substructure, we also use a version with a 3-pronged substructure, for which $X/Y(\rightarrow qqq)$. If not otherwise specified, we use the 2-pronged signal. We prepare our data set by injecting N_{sig} signal events into the 1 000 000 background events of the LHC0 data set. As a default, we use a fixed set of $N_{\text{sig}} = 1\,000$ signal events. The signal selection is randomized in our two-dimensional scans over signal injections and background set size in Sec. 5.1.4. Otherwise, it is kept constant. Within our signal region, which is centered around the W' mass $m_{W'} = 3.5\text{ TeV}$ and has a width of 0.4 TeV , we retain approximately 120 000 background events and approximately 75% of the signal events. This data is shuffled, and the first 120 000 events are used as the training and validation set, as was done in Ref. [36].

For the majority of the tests we perform, the IAD is used. For the IAD, we use 272 000 of the additional SR background events from Ref. [170] as the BT, which are trained against the SR data in the classification. As a test set, we use 340 000 SR background events and 20 000 SR signal events. For both the BDT and the NN, we use a 50-50 training-validation split, which is randomized between trainings.

Name	Feature No.	Features
Baseline	4	$\{m_{J1}, \Delta m_J, \tau_{21}^{\beta=1,J1}, \tau_{21}^{\beta=1,J2}\}$
Baseline + NG	$4 + N$	$\{m_{J1}, \Delta m_J, \tau_{21}^{\beta=1,J1}, \tau_{21}^{\beta=1,J2}\} \cup \{x_i\}_{i=1}^N, x_i \sim \mathcal{N}(0, 1)$
Extended set 1	10	$\{m_{J1}, \Delta m_J, \tau_{N,N-1}^{\beta=1,J1}, \tau_{N,N-1}^{\beta=1,J2}\}$ for $2 \leq N \leq 5$
Extended set 2	12	$\{m_{J1}, \Delta m_J, \tau_N^{\beta=1,J1}, \tau_N^{\beta=1,J2}\}$ for $N \leq 5$
Extended set 3	56	$\{m_{J1}, \Delta m_J, \tau_N^{\beta,J1}, \tau_N^{\beta,J2}\}$ for $N \leq 9$ and $\beta \in \{0.5, 1, 2\}$

Table 5.1: Summary of the feature sets considered in this work. Details of the observables are given in the text. $x_i \sim \mathcal{N}(0, 1)$ denotes features drawn from univariate Gaussians with mean $\mu = 0$ and standard deviation $\sigma = 1$.

Feature Sets

As we seek to explore the robustness of classifiers to additional features, feature sets are central to the studies in this section. Our baseline for comparison is the standard set of features used for weakly supervised anomaly detection, which was introduced in Sec. 4.3.2. This feature set sorts the two highest p_T jets by mass and uses the mass of the lighter jet m_{J1} , the difference of jet masses Δm_J , and the 21-subjettiness ratios for both jets $\tau_{21}^{J1/J2}$ for the classification. We call this feature set the “Baseline” feature set.

In addition to this Baseline set, we use a set which contains both the Baseline features and in addition fully uninformative features. For each data point, we draw N additional features from univariate Gaussians with mean $\mu = 0$ and standard deviation $\sigma = 1$. These features are appended to the 4 baseline features to make a new feature set with $4 + N$ features. This is done equivalently for both signal and background, meaning that the additional features are distributed the same for both signal and background. As such, they are indeed not informative for the classification, while at the same time introducing noise. We call these feature sets “Baseline + NG” and vary N to test the impact of variable numbers of uninformative features.

In addition to this artificially enlarged feature set, we also explore larger subjettiness-based feature sets. For these feature sets, additional subjettiness features were calculated from the LHC0 R&D low-level features and published in Ref. [177]. This data includes the jet four-momenta as well as a total of 27 N-subjettinesses per jet including $1 \leq N \leq 9$ for $\beta \in \{0.5, 1, 2\}$, calculated with the same settings used for the LHC0 R&D data set high-level features (see Sec. 4.3.2). From these features, 3 extended feature sets are constructed using the two mass features used in the baseline feature set, and adding subjettiness-based features. Firstly, Extended set 1 includes additional subjettiness ratios $\tau_{N,N-1}^{\beta=1}$ up to $N = 5$ for both jets, resulting in 10 features total. Extended set 2 uses the N-subjettinesses themselves instead of their ratios, going up to $N = 5$, resulting in 12 features. Extended set 3 uses the full set of subjettinesses calculated for this study, resulting in a total of 56 features. All feature sets are summarized in Tab. 5.1.

For the classification, we standardize all features such that their distributions have mean $\mu = 0$ and standardization $\sigma = 1$.

Classifiers

We compare two classifiers throughout this work, one BDT- and one MLP-based. Neither setup was optimized systematically. However, the hyperparameters for the MLP are based on previous weakly supervised studies such as Ref. [36], while for the BDT, default hyperparameters are largely used.

BOOSTED DECISION TREES As our boosted decision tree classifier, we use the `HistGradientBoostingClassifier` from `scikit-learn` [117]. This architecture is based on the state-of-the-art [98] implementation of Gradient Boosting `LightGBM` [105] (see also Sec. 3.3.2). Fundamentally, both provide a GBDT classifier which histograms its input before the classification. The `scikit-learn` implementation provides an easy-to-use setup, which was found to perform well out of the box, making it ideal for anomaly detection, where setup optimization can be difficult.

We largely use the default hyperparameters. This includes early stopping on the validation loss with a patience of ten iterations, where each iteration is one decision tree. A validation fraction of 50% is implemented as mentioned above, and class weights are used to weight both classes equally. To ensure all optimizations are stopped by early stopping, the maximum number of iterations is set to 200. The training of the individual decision trees within the BDT is typically stopped by the maximum number of leaf nodes, which is set to 31, as is the default. A crucial component of our classification setup is an ensemble of N BDTs trained with randomized training-validation splits. The ensemble prediction is the mean of predictions from all N individual BDTs.

NEURAL NETWORKS Our neural network architecture is an MLP implemented using `TensorFlow` [179]. The MLP has three hidden layers with 64 nodes each with ReLU activation¹. The training is conducted on the BCE loss with balanced class weights. We use the ADAM optimizer with a learning rate of 10^{-3} and a batch size of 128. Early stopping is applied based on the validation loss with a patience of 10 epochs, and we train for a maximum of 100 epochs. As with the BDT, we ensemble $N = 50$ trainings with randomized training and validation splits. This ensembling strategy is where we differ from previous implementations: Ref. [36] ensembled the best ten epochs from a single training selected according to the validation loss. To allow for this, no early stopping was used in Ref. [36].

Performance Evaluation and Validation

We evaluate the performance of our classifiers and feature sets based on SIC curves, which we show using a statistics cutoff allowing at most a 20% relative error on the background (see Sec. 4.3.5). Additionally, for scans across multiple signal injections, we use the maximum SIC across all cuts passing the statistics cutoff. We typically show these curves for a single classifier for the NN and ten classifiers for the BDT as the NN is much more computationally expensive. The statistical fluctuations of this highly ensembled NN setup were found to be minimal. The ten BDT classifiers are shown as the median performance with an error band

¹ In the original code, one of the MLP layers was missing this ReLU activation, functionally collapsing the three hidden layers to two. Tests showed that fixing this error did not result in any significant performance change. We show the original paper results, which were produced with this bug.

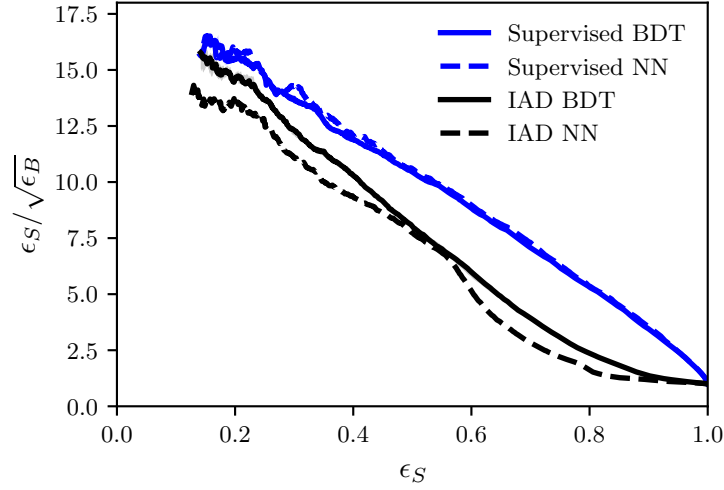


Figure 5.1: Setup validation. SIC curves for a supervised and weakly supervised IAD classification using both the BDT and NN setups for the Baseline feature set.

equivalent to a 1σ fluctuation around the median, i.e., containing 68% of classifiers.

To validate our setup, we compare the performance of our BDT and NN classifiers on the Baseline feature set for both a supervised and IAD classification in Fig. 5.1. In both cases, performances largely match between the BDT and NN. The error bands for the BDT are very narrow as the performance is stabilized by the ensemble. Overall, we are able to achieve the performance that was also seen in Ref. [36] with both of our classifiers, validating our setup.

5.1.2 Uninformative Features, Rotational Invariance and Ensembling

We begin our studies by considering the uninformative Gaussian noise feature sets introduced above to test our robustness to additional uninformative features. We generate different sets using different numbers N of additional noisy features ranging from $N = 1$ to $N = 50$. The informative features are the same in all cases, namely the Baseline features, meaning that a perfect classifier, which can ignore uninformative features, should retain the same performance. In practice, uninformative features also introduce noise, which makes achieving the same performance difficult if the signal fraction is not increased.

The results for the NN and BDT classifiers for different feature set sizes can be seen in Fig. 5.2. A stark difference is visible between the two classifiers. Already with two features, the NN’s performance (left panel) drops by a factor of three, from 15 to 5. With five Gaussian features, the performance drops to be almost random. The BDT (right panel), on the other hand, retains almost the same performance up to an addition of 10 Gaussian features, with still significantly better than random performance for 30 and 50 Gaussian features. Essentially, we observe that the BDT is able to ignore uninformative features, whereas the NN is not.

The reason for this difference lies fundamentally in the different network structures and their treatment of input features, which we discussed shortly at the start of this section: BDTs make use of the existing feature division, while the NN’s structure attempts to engineer the optimal feature space itself. We can show this, and at the same time show that our high-level

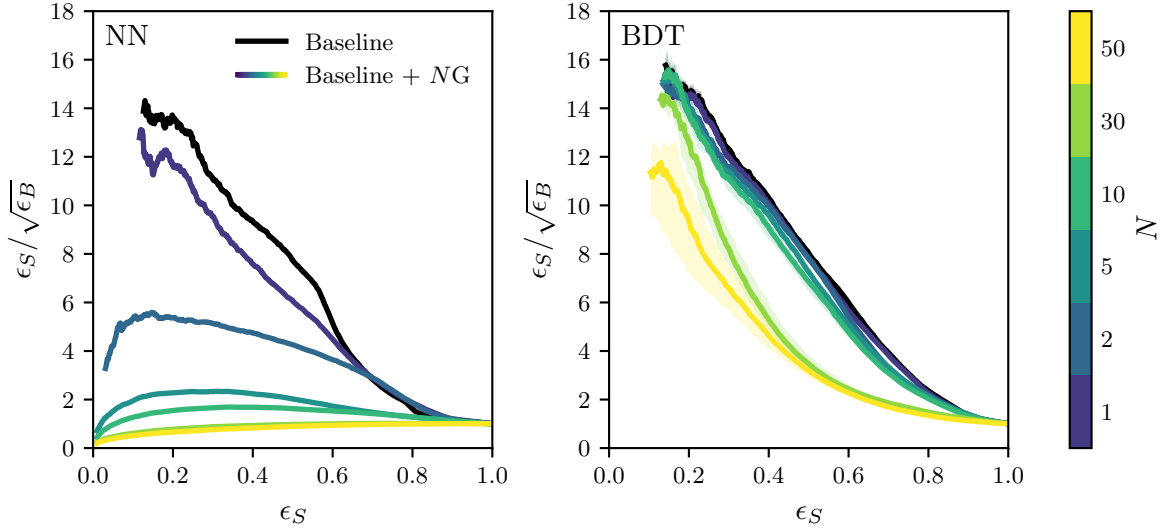


Figure 5.2: The impact of uninformative features on classifier performances. SIC curves for NN (left) and BDT (right) classifiers trained on the Baseline feature set as well as the Baseline + NG feature set, where N Gaussian-distributed uninformative features are additionally included. $N \in \{1, 2, 5, 10, 30, 50\}$ are chosen. For $N = 30$ and $N = 50$, the ensemble size is increased to 100 for the BDT classifier.

features are indeed tabular in nature by recreating a test from Ref. [178]: We apply random rotations to the feature sets and then compare the performance of classifiers trained on both the standard and the rotated feature sets. As NNs are rotationally invariant, we expect their performance to be largely invariant under this change. The performance of a BDT classifier, however, should drop if the feature set contains tabular data. This is true of tabular features as they are designed to contain important information on their own. As such, the standard version of this feature space should be close to an optimally oriented feature space. If the original features are not tabular, we would expect to see that the performance of the BDT remains unchanged under a rotation since the original features were already a close to random orientation of the feature space.

We show the results of this test in Fig. 5.3 for the Baseline feature set, Baseline + 2G, and Baseline + 5G². We use a fixed, randomly generated rotation matrix here to keep the single NN classifier and the ten BDT classifiers comparable. To ensure that no trivial rotations are used (e.g., a unit matrix), the rotation matrices are checked by eye. For the NN, the performance of classifiers trained with or without the application of a random rotation matches up for each of the three feature sets. This shows the expected rotational invariance of the NN. For the BDT, however, a clear performance drop is observed if feature sets are rotated. It is already present for the Baseline feature set, showing that the features are tabular. If Gaussian features are added and a rotation is applied, the performance drops further. In our tests, the performance of the NN and BDT with Gaussian features approximately match when the BDT's input features are rotated. This equivalence may change for larger data set sizes as the NN is better able to learn optimal feature orientations with more data. The BDT's structure may also allow it to achieve some performance gain with more data by learning

² While this test was performed in Ref. [1], it was reimplemented for this thesis. The original test performed the rotation before the standardization of features. This resulted in feature normalizations being lost and the NN being unable to completely reverse the rotation. In the tests shown in Fig. 5.3, the rotation is applied after feature standardization.

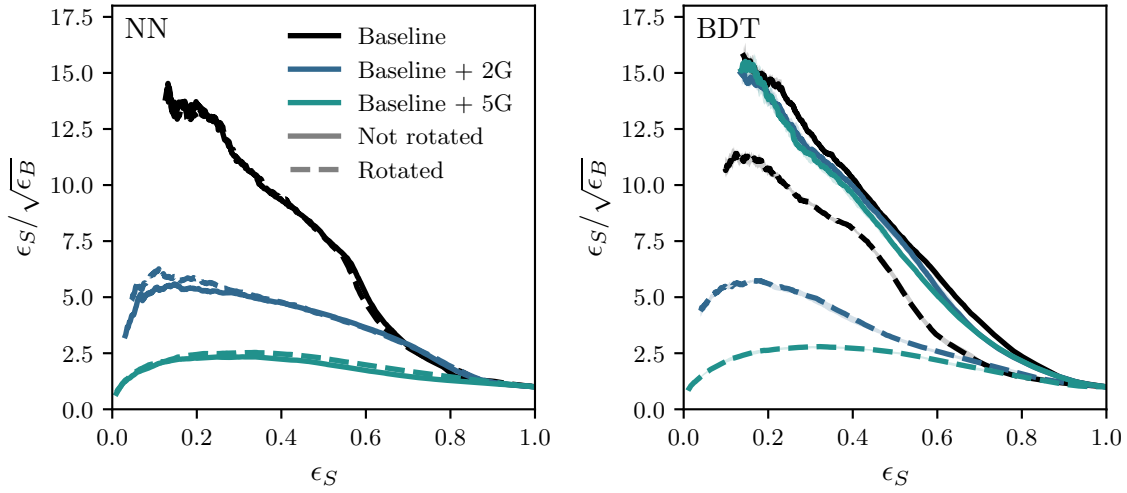


Figure 5.3: Behavior of NNs (left) and BDTs (right) under rotations of the feature space. SIC curves for classifiers trained on Baseline and Baseline + NG feature sets for $N = 2, 5$, where trainings are conducted on standard feature sets (solid lines) and feature sets with a fixed randomly generated rotation applied to them (dashed lines). Test reimplemented from Ref. [1] and based on studies from Ref. [178].

correlated cuts, but it cannot learn an optimal orientation.

In addition to the treatment of input features, we also observe that the ensembling we apply has a large impact on the performance. We can see this when comparing the performance of single trainings of BDTs or NNs with their median performance and their ensemble performance in Fig. 5.4. We consider the Baseline + 10G feature set here. For the NN (left panel), the variance of the individual trainings is low, with the performance being relatively stable around the median. While the variance of individual NN trainings is somewhat larger if, for example, Baseline + 2G is considered, the variance of individual BDT trainings is consistently significantly larger. For the example shown in the right panel of Fig. 5.4, we see trainings with performances close to random and single trainings with maximum SICs of around 12.5. However, none of the trainings for the BDT achieves the performance of the ensemble. This points to the different trainings learning somewhat complementary information from their different training-validation splits.

The ensembling procedure we follow here is a type of Bagging, i.e., ensembling multiple classifiers trained on subsamples of the same data. As was discussed in Sec. 3.3, bagging works best to stabilize the performance of high-variance models and can extract good fits to the data from highly overfitted individual models. All individual models fit the full distribution of their respective training data, including both the underlying distribution and statistical fluctuations. If enough models with sufficient subsampling are ensembled, the statistical fluctuations average out, and a fit of only the underlying distribution is retained (see also Fig. 3.4 in Sec. 3.3). While this has not been shown in our case, it seems particularly applicable to anomaly detection, as the low signal amounts may cause the signal to look much like a statistical fluctuation to each individual model. Further studies may be able to provide a more thorough understanding of the behavior of individual classifiers and ensembles in weakly supervised anomaly detection.

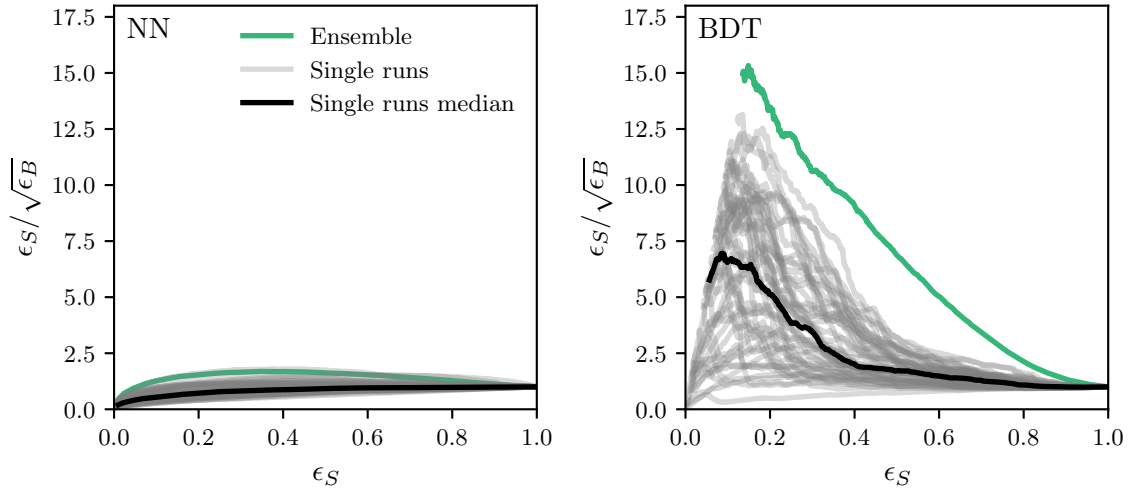


Figure 5.4: The impact of ensembling on anomaly detection performance. SIC curves for single trainings of the NN (left) and BDT (right) architectures (light grey), their median performance (black), and the ensemble performance (green) for the Baseline + 10G feature set.

5.1.3 Extended Feature Sets and Model Agnosticism

Having shown that BDTs provide a gain in robustness to uninformative features, we seek to now show the physics advantage of this property. For this purpose, we use the physics-informed extended feature sets introduced in Sec. 5.1.1. These feature sets use additional subjettinesses and subjettiness ratios. As such, they provide sensitivity to decays with more prongs than the Baseline set. While each successive feature set contains additional information compared to the last – i.e., Extended set 1 contains more information than Baseline, and Extended set 2 more than Extended set 1, etc. – information is represented differently in some of the feature sets. The Baseline set and Extended set 1 represent the features as subjettiness ratios. Extended set 2 and Extended set 3 contain the subjettinesses themselves. This is particularly relevant as for a 2-prong signal subjettiness ratios $\tau_{N,N-1}$ contain little information for $N > 2$. The subjettinesses themselves, however, remain slightly informative. Both of this can be seen in Fig. 5.5, where we show the 54-subjettiness ratio τ_{54}^{J2} and the 5-subjettiness τ_5^{J2} for jet 2 for both signal and background. While both distributions largely overlap in the ratio τ_{54}^{J2} , they still differ somewhat in the subjettiness τ_5^{J2} . For both signal and background, the N -subjettiness no longer changes significantly as N is increased if N is larger than the number of subjets, and the overall distribution remains similar. As such, no significant separation between background and signal is observed in the subjettiness ratios in this regime. Considering this, Extended set 2 and 3 contain only informative but very correlated features, while Extended set 1 also contains uninformative features.

The performance of the different classifiers can be seen in Fig. 5.6, where SIC curves are shown for both the NN and the BDT. The BDT performance (right panel) consistently increases as additional features are added, showcasing its robustness to different feature set structures. The NN (left panel), on the other hand, experiences a significant performance drop for Extended set 1, which contains the uninformative features. For an anomaly detection scenario, where it may be unclear which features are informative beforehand, this susceptibility to the structure of feature sets is a disadvantageous characteristic of the classifier. Nevertheless, it should be noted that for the larger fully informative Extended sets 2 and 3, both classifiers are

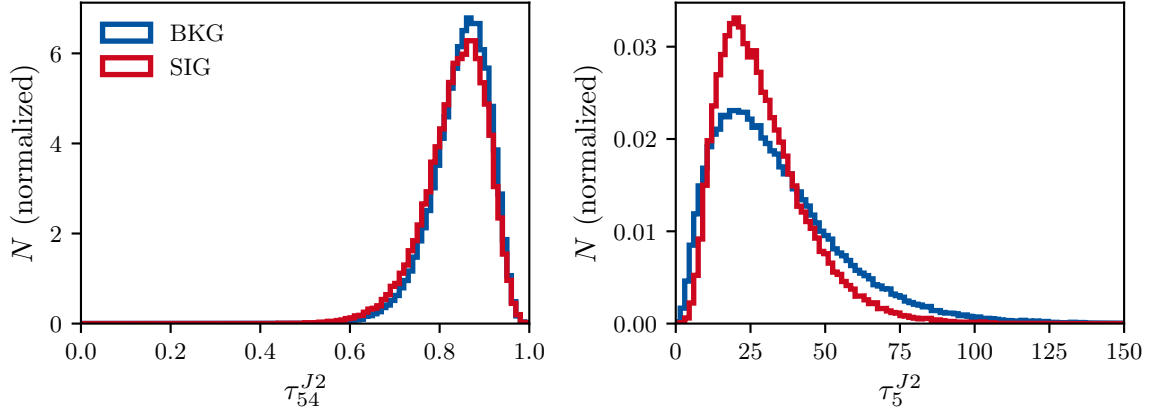


Figure 5.5: The difference between subjettiness ratios and subjettinesses. Histogram of LHC signal (red) and background (blue) distributions for 54-subjettiness ratio τ_{54}^{J2} (left) and 5-subjettiness τ_5^{J2} (right) for jet 2.

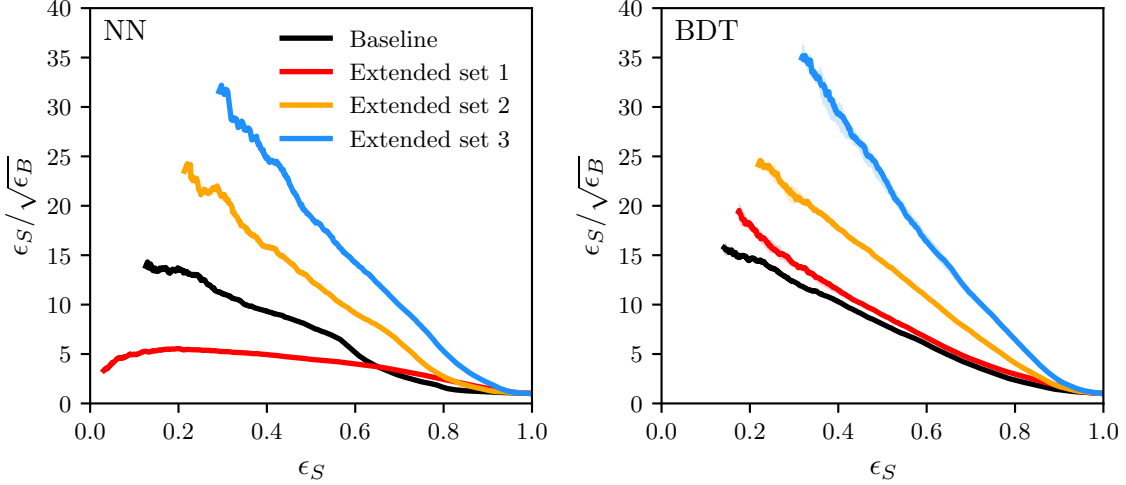


Figure 5.6: The impact of physics features on the NN and BDT classifiers. SIC curves for NN (left) and BDT (right) classifiers trained on the Baseline features as well as the three extended sets summarized in Tab. 5.1.

able to achieve a high significance improvement. Absolute numbers should not be compared too closely between the BDT and NN classifiers here, as no dedicated optimization of the architectures or trainings were conducted. Slight improvements, though not a change of general trends, could be expected with such an optimization.

With the introduction of feature sets sensitive to additional substructures, an improvement in the model-agnostic sensitivity of the classifier is to be expected. We can test this by considering the LHC 3-prong signal. The Baseline feature set is not expected to provide any significant classification power for this signal, and indeed this is observed in Fig. 5.7. While the BDT classifier is able to reach a maximum SIC of around 5 with these features, the NN classifier does not perform significantly better than random with the Baseline features. In this case, both classifiers consistently improve as additional information is added with the extended feature sets. However, the performance of the NN falls short of the BDT performance for all feature

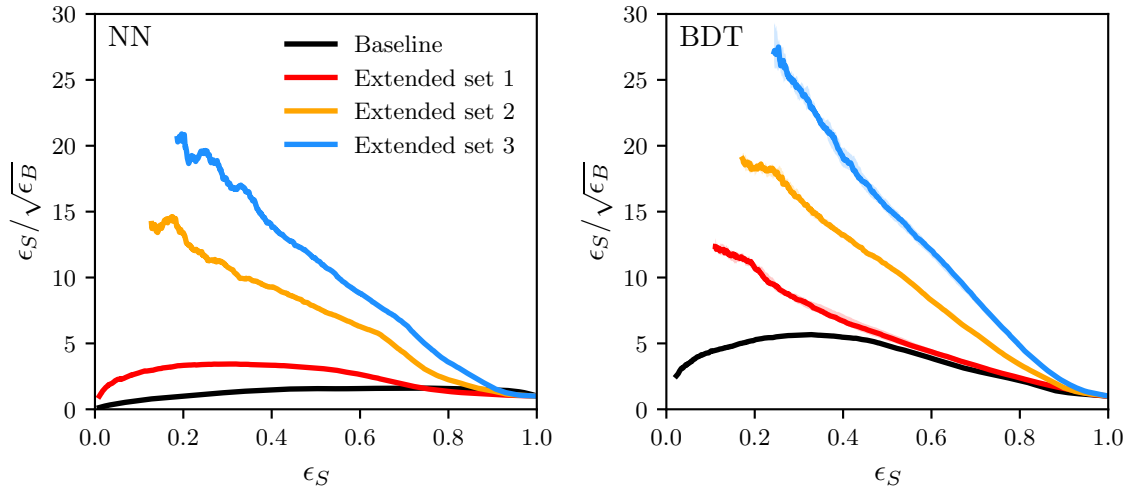


Figure 5.7: Same as Fig. 5.6 but for the 3-prong LHCO signal.

sets, particularly for Extended set 1. This is likely at least partially due to the uninformative features also present in Extended set 1 for the 3-prong signal. Although it can be expected that some performance improvement may be achieved through dedicated hyperparameter tuning, the susceptibility to uninformative features of the NN remains. Additionally, the high performance without the necessity of tuning is a clear benefit of the BDT. Tuning hyperparameters model-agnostically is a challenge, and as such, setups which do not require tuning to perform well for a variety of signals and feature sets are preferable.

5.1.4 Varying Signal Amounts and Data Set Size

So far, all tests we have conducted have used a standard signal injection corresponding to $N_{\text{sig}} = 1000$ signal events. Using such a standardized working point, we can study variations of setups and feature sets in detail. However, for this working point, which has an initial significance of approximately 2.2σ in the signal region, the significance improvement of the Baseline feature set would already be sufficient to detect the LHCO signal in an analysis – depending, of course, on the magnitude of systematic uncertainties. As such, we are most interested in the improvement at lower signal injections. Improvements here allow us to reach lower cross sections for the same data set size, detecting a possible BSM signal earlier, and as such, with less measurement time and costs. We study the behavior of the classifiers for lower signal injections and vary the injection amount N_{sig} .

We compare the performance of the BDT and NN for the Baseline and extended feature sets in Fig. 5.8. In general, we observe the following shape for all feature sets and classifiers, which is common for weakly supervised anomaly detection (see, e.g., Ref. [36]): For high signal injections – here, typically $N_{\text{sig}} \geq 1000$ – the maximum SIC tends to be relatively stable. The classifier has extracted all available information from the feature set. As the signal injection is lowered, a drop-off point is reached, where performance drops quickly to almost random levels. Below this, no signal detection is achieved. This shape still varies somewhat, particularly with the drop-off being less sharp for larger feature sets, because the classifier may be able to use more auxiliary information with additional signal events. For smaller feature sets, such

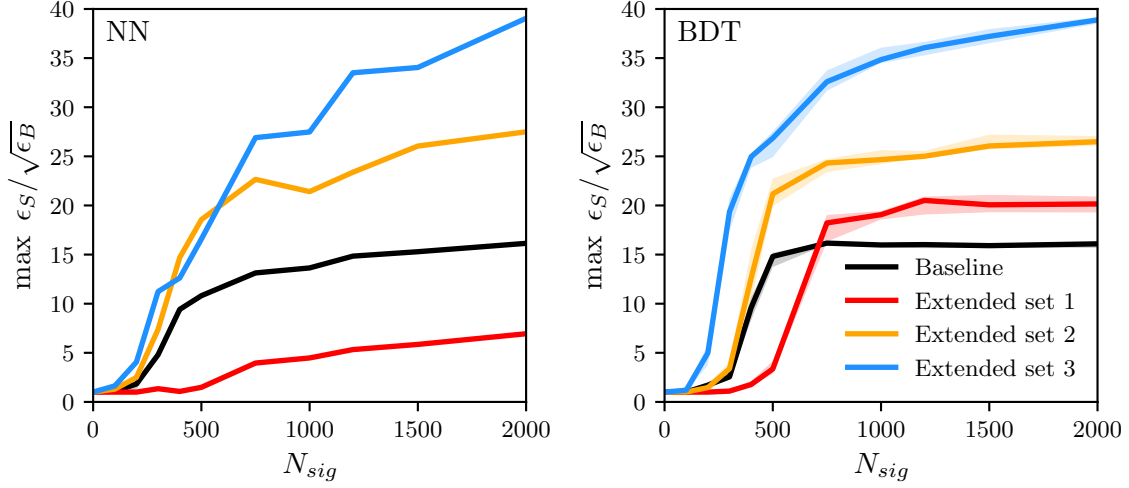


Figure 5.8: The impact of physics features on the sensitivity improvement. Maximum SIC with statistics cutoff shown as a function of the number of signal events N_{sig} in the data set for the NN (left) and BDT classifier (right). The number of background events is held fixed. Curves shown for the Baseline and extended sets. Ref. [1] included only results for Baseline features and Extended set 3 for the BDT and Baseline features for the NN. Additional runs were produced using the existing code.

auxiliary information is rather limited.

Considering the subplots in more detail, we can see that results for the NN (left) look as expected, with the ordering of performances from Fig. 5.6 retained across different signal injections. Curves are not fully smooth here, as only one run is used. In particular, around the drop-off point, fluctuations tend to increase and may not be completely stabilized by our ensemble. For the BDT (right), which shows the median performance of ten runs, the curves are much smoother. While results for high signal injections with $N_{\text{sig}} \geq 750$ show the trends already observed in Fig. 5.6, at lower cross sections, an earlier performance drop-off is observed for Extended set 1 compared with the Baseline set. This is despite Extended set 1 containing some additional information. We can test how the BDT drop-off point is influenced by uninformative features in general by again considering the Baseline + NG feature sets, as shown in Fig. 5.9. As we increase N from 1 to 10, we see a consistent shift of the drop-off point to slightly higher values of N_{sig} . The magnitude of the shift is consistent with the shift observed for Extended set 1, which contains 6 largely uninformative features. While the drop-off point of the classifier is shifted, it is able to extract the full information from the feature set for larger signal injections.

In addition to the sensitivity to lower cross sections, the sensitivity change for larger data sets is also highly relevant to anomaly detection analyses at the LHC. Not only is data continually collected, with Run 3 data already providing much larger data sets than the LHC Olympics, but the coming HL-LHC promises to enhance current data amounts significantly. For this purpose, we perform tests scanning over different signal and background amounts simultaneously. We set the background amount in the SR to N_B and the signal amount in the SR to N_S ³. We vary the signal amount in such a way as to keep the initial signal significance $N_S/\sqrt{N_B}$ in the SR constant per row. The background template also uses N_B

³ Note that $N_{\text{sig}} \neq N_S$. N_{sig} denotes the number of signal events in the full data set, while N_S is the signal amount in the SR only. For a fixed data set, $N_S \approx 0.75 \cdot N_{\text{sig}}$.

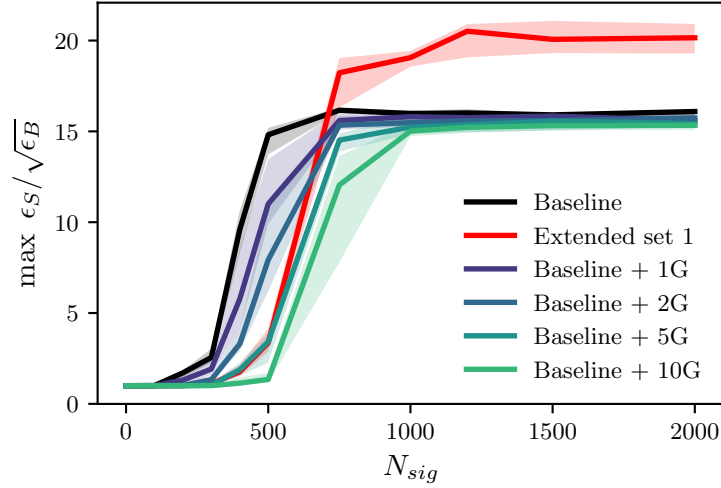


Figure 5.9: The impact of uninformative features on the BDT sensitivity for varying signal amounts. Maximum SIC with statistics cutoff shown as a function of the number of signal events N_{sig} in the data set for the BDT classifier. Results for Baseline, Extended set 1 and Baseline + NG feature sets with $N \in \{1, 2, 5, 10\}$ are shown. The number of background events is held fixed. Results are not included in Ref. [1].

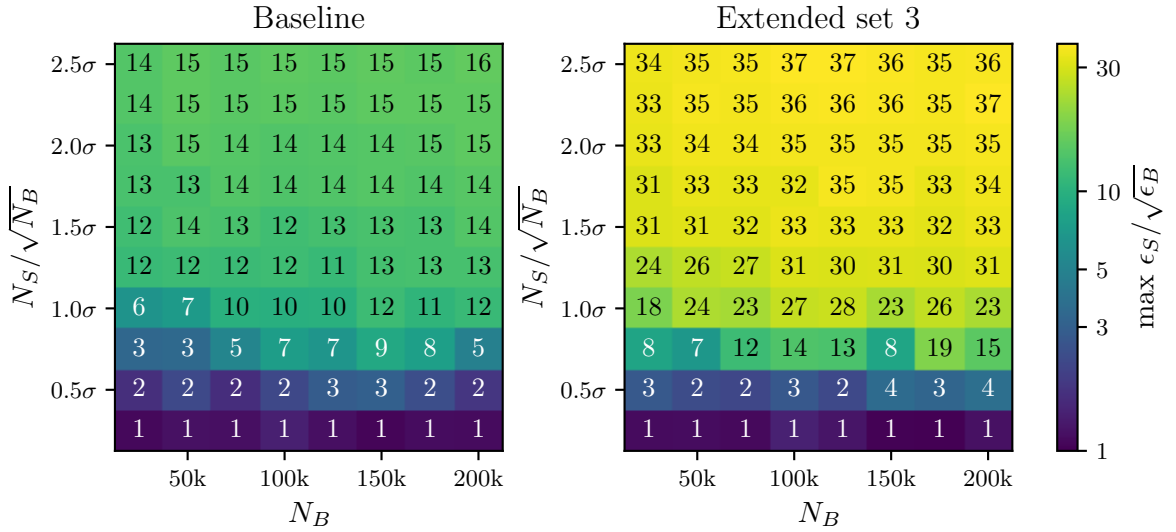


Figure 5.10: The impact of changing data set sizes on the sensitivity of weak supervision. Maximum SIC values with statistics cutoff are shown for varied background N_B , and signal amounts N_S in the SR data set. The BT is chosen with $N_{\text{BT}} = N_B$. Signal amounts are varied such that the initial significance $N_S / \sqrt{N_B}$ is constant. Results are shown for the Baseline features (left), and Extended set 3 (right).

SR background events. To limit computation time, we perform these tests with the BDT classifier only. Results of these scans for the Baseline features and Extended set 3 are shown in Fig. 5.10. We see a consistent performance across rows, with a fixed initial significance providing a performance that is largely invariant under changes of the overall data set size. One caveat is the performance for very small data set sizes, where some improvement is still observed if the background amount is increased. We observe a similar sharp drop-off as for the 1D scans in Figs. 5.8 and 5.9 at approximately 0.5σ and 0.75σ . Here, we also observe large

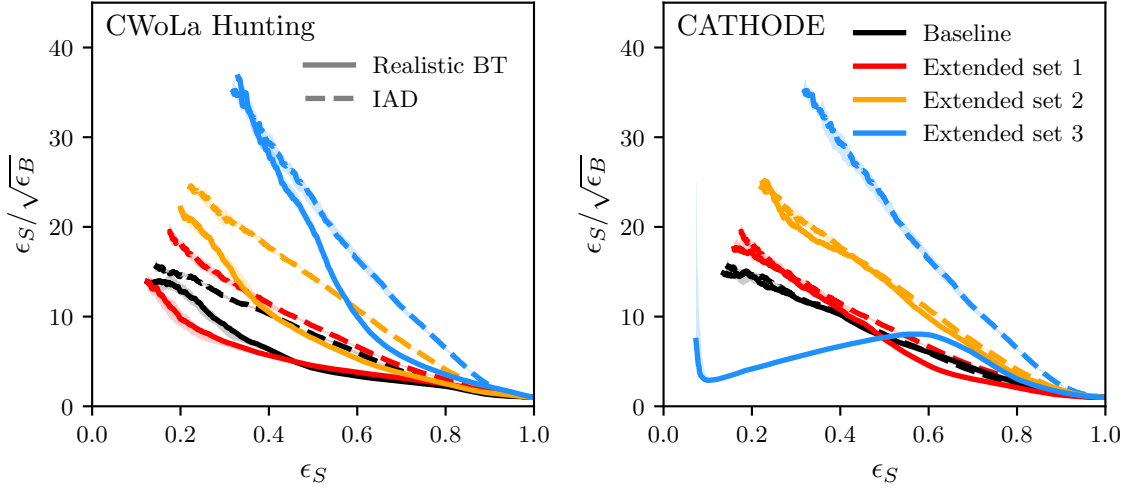


Figure 5.11: The impact of realistic background templates on the performance of extended feature sets. SIC curves for CWoLa Hunting (left) and CATHODE (right) for the Baseline and extended sets are shown. IAD curves are shown as dashed lines to better observe the impact of the BT. Results are not included in Ref. [1].

fluctuations between runs. Signals at an initial significance of approximately 1σ may still be detected at discovery level sensitivity – depending, of course, on systematic considerations.

The invariance of the performance under data set size for fixed initial significances means that as larger data sets are collected, lower cross sections could be detected by the same method. If the performance were constant for a fixed signal fraction N_S/N_B , this would not be the case. As such, the relation between initial significance and performance is crucial considering that it promises potential for discovery even if analyses at lower cross sections are unable to detect significant deviations from the Standard Model. This performance trend was seen to hold up for larger data set sizes in tests conducted in Ref. [180].

5.1.5 Realistic Background Templates for Extended Feature Sets

While the observed performance and particularly the improved robustness of the BDT compared with the NN classifier are promising, for an analysis with larger feature sets to be viable, a high-quality background template needs to be derived from data. Therefore, we test CWoLa Hunting and CATHODE background templates here. For CWoLa Hunting, we use the data from SSBs with a width of 0.2 TeV around the SR. For CATHODE, we use a Flow Matching density estimator as described in App. A.1. We sample $3 \cdot N_{\text{SR}}$ events from this DE, i.e., we oversample by a factor of 3. Other than exchanging the BT, all setup specifications remain as described in Sec. 5.1.1. We test only the BDT classifier.

Results for both methods can be seen in Fig. 5.11. For CWoLa Hunting (left), we observe a consistent difference in performance between CWoLa Hunting and IAD background templates. The slight mismodeling of the CWoLa Hunting background template, which is caused by correlations between the classification features and the dijet mass, seems to be relatively consistent between all feature sets. However, we can see that a larger feature set also contains additional mismodeling. In particular, the SIC curve for Extended set 1 lies slightly below

the one for the Baseline set. A significant performance gain can still be achieved by using a larger and more informative feature set, such as Extended set 3. For CATHODE (right), the performance for the smaller feature sets (Baseline, Extended set 1, and Extended set 2) largely matches the performance of the IAD. This speaks to a high-quality background template having been generated by the Flow Matching density estimator. For Extended set 3, the performance difference between IAD and CATHODE increases drastically. The CATHODE performance not only falls short of the CWoLa Hunting performance for Extended set 3 but also of the performance of CATHODE with Extended set 2. With Extended set 3 being significantly larger – 56 compared to 12 classification features – the DE is no longer able to learn the features and their correlations to the required accuracy. Therefore, while CATHODE can achieve a higher quality background template and, as such, performance for small feature sets, CWoLa Hunting retains its background template quality better for larger feature sets. The limit of CATHODE performance is likely to shift to larger and larger feature sets as density estimation is improved. Dedicated optimization may also be able to achieve viable background templates for Extended set 3 with currently available DE technology.

5.1.6 Conclusion and Outlook

In this section, we have shown how using BDT classifiers instead of neural networks can enhance the sensitivity of weakly supervised searches for BSM physics. In the large data regime, which LHC physics typically deals in, deep learning can usually provide high sensitivity by extracting information from the full high-dimensional phase space. With anomaly detection and especially weakly supervised anomaly detection, we by design do not work in this regime. While total data set sizes may increase as more data is collected, the effective data set size is governed by the small number of signal events we aim to detect. As such, we remain in the small data set regime, where BDTs using high-level, tabular features are likely to remain more effective than deep learning and low-level information – bar the invention of new methods which fundamentally revolutionize our understanding of machine learning.

In our tests, we have shown this to be the case by exploring the addition of uninformative features. We have understood the relation to rotational invariance and the ability of deep learning to perform feature engineering. While typically the reason for deep learning’s success, feature engineering only outperforms decision tree-based learning with sufficient data set statistics. This can also be seen in recent studies using low-level features for anomaly detection, such as Refs. [27, 28, 30, 175], which do not reach the low signal injections detected using high-level features. In addition to the use of BDTs, we have also proposed an ensembling strategy which allows our setup to remain particularly robust to uninformative features.

This setup also allows for the use of larger feature sets as the performance remains relatively robust whether the added features are informative or not. Since the publication of Ref. [1], this has allowed the use of larger high-level feature sets (see e.g., Refs. [5, 26]). Such larger feature sets, in particular, improve the range of BSM signal models a weakly supervised search can be sensitive to, as we have shown on the LHCO 3-prong signal. Ref. [5] showed the possible significance improvement across a wider range of BSM signal models based on the model-agnostic CMS analysis [46]. In addition to improvements in the model agnosticism of analyses, larger feature sets can also improve both the maximum significance improvement and the discovery potential at low signal injections. As such, a signal may be detected at discovery level significance with less data.

While most of our tests were performed with an idealized background template, we have also shown that both CWoLa Hunting and CATHODE achieve an improved performance when using our extended feature sets. This is crucial for the applicability of our improvements to actual analyses.

5.2 SELECTING THE OPTIMAL ANOMALY DETECTOR

This section is based on Ref. [3]. The code of this project can be found in Ref. [181]. Research not included in the original publication is specified as such in the respective figure caption.

For the sensitivity of weakly supervised searches, setup choices like the choice of classifier architecture can be crucial, as we have seen in the previous section. However, while the choice of using a BDT versus a NN for high-level features can be based on general, well-founded considerations, this is not true for most setup specifications. For example, hyperparameter choices can typically not be derived from first principles. Reasonable choices according to typical model behavior may be possible, but there is no guarantee for optimality. Additionally, if optimal model and hyperparameter choices differ between different BSM signals, a single optimal choice may not exist.

So far, no systematic setup optimization is reported by any of the experimental analyses using weak supervision [44–47]. Limited optimization on benchmark signals was performed in some cases [182]. Beyond this, only a selection of the best epoch or classifier runs based on the validation loss or the highest number of selected signal region events was performed in Refs. [46, 182]. Here, we investigate the viability of performing a variety of different setup optimization tasks directly on data to further improve the model agnosticism of our results. However, data-driven optimization is difficult in anomaly detection scenarios because of the small signal amounts available in practice. This makes all optimization metrics relatively noisy, which we will in particular see for the standard BCE loss. As an alternative, we introduce the Above Random Gain of SIC (ARGOS) metric. It is highly correlated with the supervised SIC, which cannot be calculated on unlabeled data. The definition and theoretical foundation for the ARGOS metric are discussed in Sec. 5.2.1.

To evaluate the viability of data-driven optimization of weakly supervised anomaly detection, we test a wide range of different optimization tasks. We start with a selection of the best epochs, as was done in Refs. [46, 182] before turning to tasks not studied in the existing literature, such as hyperparameter optimization in Sec. 5.2.4, architecture selection in Sec. 5.2.5, and lastly feature selection in Sec. 5.2.6. We also discuss how an optimization on data may affect the rate of false discoveries in Sec. 5.2.7 before concluding in Sec. 5.2.8. However, first, we must introduce our metrics (see Sec. 5.2.1) and discuss our setup (see Sec. 5.2.2).

5.2.1 Metrics for Optimization

A good estimation of the anomaly detection performance of a classifier for a specific benchmark signal model can be made by the Significance Improvement Characteristic (SIC) given by

$$\text{SIC} = \frac{\varepsilon_S}{\sqrt{\varepsilon_B}}, \quad (5.1)$$

which, as was explained in Sec. 4.3.5, is the significance improvement of the classifier in the Gaussian limit. This metric is useful to compare different anomaly detectors on simulation, but requires labeled data to be calculated.

Theoretically, classification between mixed data sets as performed in weak supervision is equivalent to supervised classifications (see Sec. 4.2). Therefore, any metric which is optimal to compare classifiers for a standard classification, such as a ROC curve or the BCE loss,

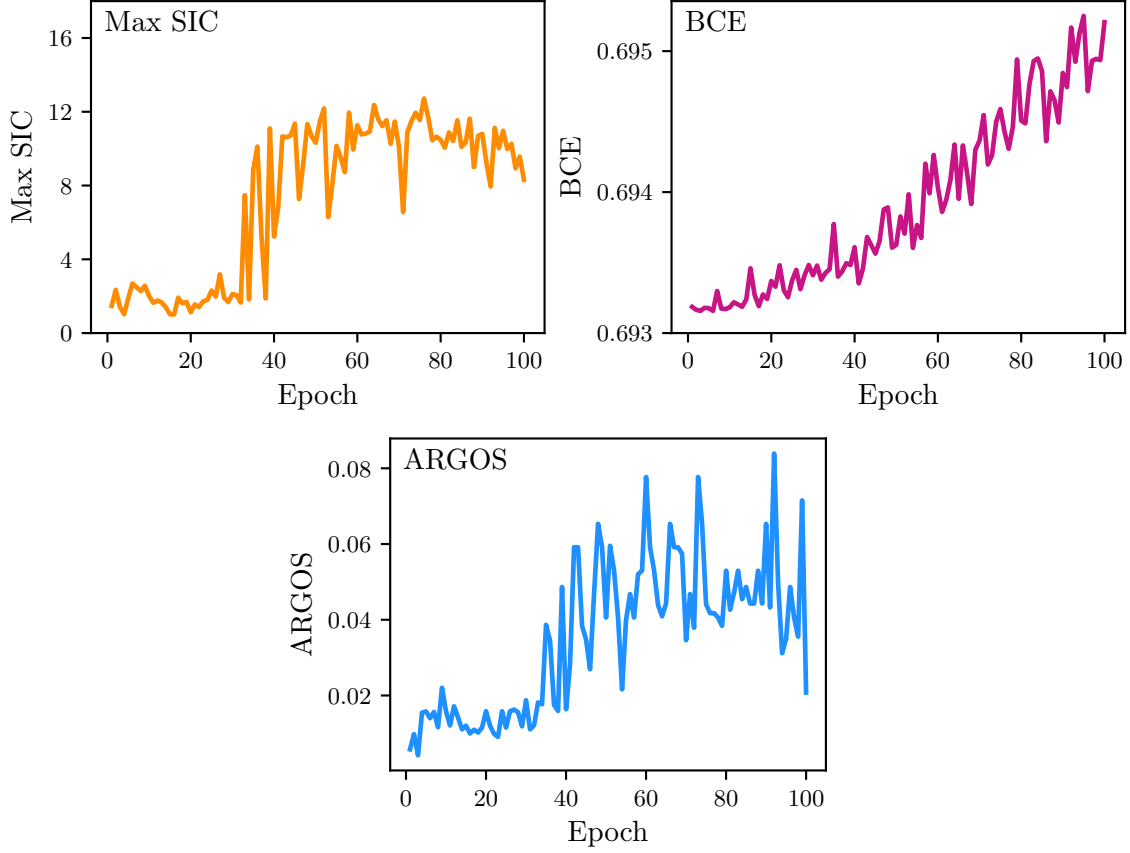


Figure 5.12: Example of metrics tracked throughout a NN training with $N_{sig} = 400$ signal events using the default hyperparameters as introduced in Sec. 5.2.2. We show the supervised max SIC metric (top left) evaluated on the test set as well as BCE (top right) and max ARGOS (bottom), both evaluated on the validation set.

should also be optimal for a weakly supervised classification. However, in practice, we are limited by the finite statistics of our data sets. In particular, with $\mathcal{O}(1000)$ signal events or fewer, such metrics can be dominated by the background. We can see this in Fig. 5.12, where different metrics were tracked throughout a single NN training with $N_{sig} = 400$ signal events in the data set (setup specifications can be found below in Sec. 5.2.2). The top left panel shows the maximum SIC as defined in Sec. 4.3.5 calculated on the test set, i.e., with the background and signal labels. We compare this to the BCE validation loss with class weights chosen to obtain a balanced class weighting in the top right panel, which uses weakly supervised, i.e., noisy labels. Overall, we see that the BCE does not correlate well with the maximum SIC. Indeed, while the BCE shows overtraining almost immediately, we can see from the maximum SIC that the signal versus background task is learned only around epoch 40. This learning is not reflected in the BCE because the BCE is dominated by the background. To obtain the BCE loss of a data set, we take the average of the BCE values for each individual data point as is standard practice. However, as the data set is dominated by background, the loss is then also dominated by the background. To eliminate this issue, we must therefore choose a metric which is less affected by the large amounts of background in our data set.

This can be achieved by building a metric based on cuts on the classifier score, as is done for the SIC. We define the Above Random Gain of SIC (ARGOS) metric as

$$\text{ARGOS} = \frac{\varepsilon_{\text{SR}}}{\sqrt{\varepsilon_{\text{BT}}}} - \sqrt{\varepsilon_{\text{BT}}}, \quad (5.2)$$

using the SR and BT efficiencies $\varepsilon_{\text{SR/BT}}$. We can rewrite the ARGOS metric under the assumptions of our weakly supervised anomaly detection (see Sec. 4.2). Firstly, we assume that the probability distribution in the SR is given by

$$p_{\text{SR}}(x) = f_S \cdot p_S(x) + (1 - f_S) \cdot p_B(x), \quad (5.3)$$

where $p_{S/B}$ are the signal and background probability distributions, respectively, and f_S is the signal fraction in the SR. Secondly, we assume a perfect background template with

$$p_{\text{BT}}^{\text{ideal}}(x) = p_B(x). \quad (5.4)$$

Even for imperfect background templates, $p_{\text{BT}}(x) \approx p_B(x)$ should be given. Using these assumptions, we can rewrite ε_{SR} and ε_{BT} such that

$$\text{ARGOS} = \frac{f_S \cdot \varepsilon_S + (1 - f_S) \cdot \varepsilon_B}{\sqrt{\varepsilon_B}} - \sqrt{\varepsilon_B} = f_S \cdot (\text{SIC} - \sqrt{\varepsilon_B}). \quad (5.5)$$

As for any given data set, the signal fraction f_S is fixed, the ARGOS metric is monotonically related to the SIC for a fixed background efficiency. ARGOS also has a direct statistical interpretation. By definition and under the assumptions of 5.3 and 5.4, ARGOS is related to the commonly used test statistic of a cut and count experiment (see also Sec. 6.1)

$$\mathcal{S} = \frac{N'_{\text{SR}} - \langle N'_B \rangle}{\sqrt{\langle N'_B \rangle}} = \frac{\varepsilon_{\text{SR}} N_{\text{SR}} - \varepsilon_B \langle N_B \rangle}{\sqrt{\varepsilon_B \langle N_B \rangle}}, \quad (5.6)$$

where $N_{\text{SR}}^{(i)}$ is the number of SR events (after the cut), $\langle N_B^{(i)} \rangle$ is the expected number of background events in the SR (after the cut) and $\varepsilon_{\text{SR/B}}$ are the SR and background efficiencies. $\mathcal{S}$ is the significance of the excess of events in the SR compared with the background expectation after a cut in the Gaussian limit. In ARGOS, $\langle N_B \rangle$ is approximated via the BT. If we use ARGOS to select models based on a fixed background efficiency, the fixed signal fraction f_S results in a monotonic relation to the SIC. In this case, many other metrics, including the number of signal region events as used in Refs. [46, 182], are equivalent to ARGOS. However, through its statistical interpretability, we can also use the maximum ARGOS to scan over different cut and select the optimal one. As, typically, the highest SIC is found for very low background efficiencies (see, e.g., Fig. 5.1), where $\text{SIC} \gg \sqrt{\varepsilon_B}$, max ARGOS is also dominated by max SIC.

These mathematical arguments for optimality, of course, do not yet guarantee that ARGOS as a metric calculated on data works well for very few signal events. However, as ε_B tends to be low – both for typically selected fixed values and if max ARGOS is used – ARGOS should be less affected by background overfitting than the BCE loss. We can see this to be true in the bottom panel of Fig. 5.12, where max ARGOS is tracked on the validation set of a NN training. Here, max ARGOS, while more noisy, largely follows the trends observed in the supervised max SIC (top left panel). In particular, we see the same sharp rise as the signal is found around epoch 40, as we do in the max SIC. Indeed, we do not observe the large impact of background overfitting seen in the BCE loss (top right panel).

5.2.2 Setup

Having introduced the optimization metrics we wish to use, we now turn to the setup for the experiments we conduct with them. Here, we focus on the overarching setup choices for this section. The precise setup for each individual test, for example, how hyperparameters are optimized, is described in each respective experiment subsection.

Data Set

We perform our studies using the LHC0 R&D data set [18], which was described in detail in Sec. 4.3.1. It provides one million QCD dijet background events as well as 100 000 signal events originating from a $W' \rightarrow X(\rightarrow qq)Y(\rightarrow qq)$ signal. We inject a set of N_{sig} signal events into the full set of background events. The value of N_{sig} is varied in our tests. We then define the sidebands and the signal region, which is centered on the W' mass at $m_{W'} = 3.5 \text{ TeV}$ and has a width of 0.4 TeV . The full SR data makes up our signal-enriched data set for training and validation, with the split depending on the classifier. The BT set depends on the weakly supervised method and is described below. Our test set to calculate the maximum SIC uses 20 000 SR signal events as well as 340 000 SR background events from Ref. [170].

For the classification, we use the Baseline features as described in Secs. 4.3.2 and 5.1.1. This feature set is made up of the mass of the lighter jet m_{J1} , the difference of the jet masses Δm_J , as well as the 21-subjettiness ratios for both jets $\tau_{21}^{J1/J2}$.

Weakly Supervised Methods

We compare the performance of three different weakly supervised methods. Firstly, we use the idealized anomaly detector, for which Eq. 5.4 is true. In this case, the optimality of the ARGOS metric is given up to data set statistics. We use this to show that our methodology works as intended and to understand the maximum performance gain we can expect when optimizing on data. Then, we work with CWoLa Hunting and CATHODE to show that our methodology still works for realistic background templates, for which Eq. 5.4 only holds approximately.

Our background templates are chosen as follows:

IDEALIZED ANOMALY DETECTOR For the IAD (see Sec. 4.2.1), we use approximately 272 000 extra SR background events from Ref. [170] as the background template.

CWOLA HUNTING For CWoLa Hunting (see Sec. 4.2.1), we use the data from short sidebands to both sides of the SR with a width of 0.2 TeV each. This results in approximately equal statistics of SR data and BT.

CATHODE For CATHODE (see Sec. 4.2.1), we use a Flow Matching density estimator based on Ref. [41] (code published in Ref. [183]). This density estimator is trained on the full sideband data. Samples are obtained by randomly sampling dijet mass values from the SR data and using these as conditions. We ensemble ten trainings of the density estimator by

combining their samples. This limits the impact of slight mismodeling of the distributions, which may be present in an individual density estimator training. We train one such density estimator for each signal injection N_{sig} . For the classifier training, a fixed oversampling factor of four is used for the BT compared with the SR data set size. The specifics of the DE architecture and training can be found in App. A.1.

Classifiers

We compare three different classifier setups with variations in both ensembling strategies and training and validation set usage.

NEURAL NETWORK Firstly, we use an MLP neural network setup based on Ref. [36]. We build our implementation on Ref. [184] using `PyTorch`. This setup uses an MLP with three hidden layers of 64 nodes each with ReLU activation. Training is conducted using the Adam optimizer on a BCE loss for 100 epochs with class weights balancing the classes. Other hyperparameters are optimized later on. As a default, a learning rate of 10^{-3} , a batch size of 128, an Adam momentum of 0.9, and no dropout or weight decay are used.

For the training, we use a 50-50 training and validation split and calculate BCE and max ARGOS on the validation set as well as max SIC on the test set after each epoch. We use these metrics for our ensembling strategy: To obtain final predictions, predictions for the ten best epochs are averaged. Each metric may be chosen to select these epochs. This ensembling strategy is based on Ref. [36], where the validation loss was used to select epochs. The validation loss was also used for CATHODE and CATHODE-b (CATHODE with an additional b-tagging classification feature) in the CMS analysis [46]. By default, we select epochs based on the validation max ARGOS.

GRADIENT BOOSTING We also use a gradient boosted decision tree setup based on Ref. [1] as described in Sec. 5.1. This setup relies on the `HistGradientBoostingClassifier` (HGB) from `scikit-learn` [117]. We perform 50 trainings with randomized 50-50 training-validation splits and ensemble them by averaging their predictions. Each training is conducted using the BCE loss with balanced class weights and early stopping on the validation loss with a patience of ten iterations. Other hyperparameters are optimized later on. As a default, we largely use the defaults of the `scikit-learn` implementation, changing only the maximum number of iterations to 200. The set of optimized parameters additionally contains the learning rate (default 0.1), the maximum number of leaf nodes (default 31), the maximum depth (default no maximum depth), the maximum number of bins (default 255), and the L2 regularization weighting factor (default 0).

ADABOOST In addition to the commonly used MLP and HGB setups, we also use an AdaBoost setup based on the `AdaBoostingClassifier` from `scikit-learn`. This setup was studied in Ref. [1], where the performance was generally found to be slightly worse than the HGB when trained on the four Baseline features as used here and significantly worse under the inclusion of uninformative features, even with optimized hyperparameters. While this AdaBoost implementation does not use a validation set, we still perform a randomized 50% subsampling of the data for each training to obtain variations between different trainings. Predictions are obtained by ensembling ten such trainings. We use a set of hyperparameters

based on Ref. [21], which we optimize later on: This set includes the maximum number of iterations (default 50), the learning rate (default 1) for the AdaBoost algorithm. Each tree within the Boosting process is chosen as the `DecisionTreeClassifier` from `scikit-learn` with the optimized hyperparameters being the maximum number of leaf nodes (default 31), the maximum depth (default no maximum depth), and the minimum number of samples per split (default 20). The class weights for the individual decision trees are set to “balanced”.

Performance Metrics

For the three optimization metrics (max SIC, BCE loss, and max ARGOS), the following setup is used: Firstly, the maximum SIC with a statistics cutoff (see Sec. 4.3.5) is always calculated using the same held-out test set. As mentioned above, it is a supervised metric, which we use to benchmark the performance of our two data-driven metrics. The data-driven metrics, BCE and maximum ARGOS, are always calculated on a subset of the data. The definition of this set is specified for each of the tests below. The BCE is always calculated using balanced class weights.

The performance of different setups is compared by considering the max SIC at different signal injections N_{sig} . We typically use ten classifiers for this, showing their median performance as well as a 68% error band. For each test, we specify what effects the error bands contain. For example, the hyperparameter optimization is performed ten times at each signal injection to obtain the fluctuations of this optimization. In all tests, we keep the injected signal events fixed to isolate other causes of fluctuations. This also allows us to show how varying only the classification setup can impact the performance for a fixed data set, as would be present in a realistic analysis.

5.2.3 NN Epoch Selection

We start with the simplest model selection task, namely the selection of the ten best epochs used for ensembling in the NN classifier. We show the impact of using different metrics to select these epochs in Fig. 5.13. All three metrics select epochs for the same set of ten trainings. As such, all metrics could theoretically select the exact same epochs and, as such obtain the exact same performance. Between the ten classifier runs, from which the median and error band are calculated, only the classifier’s random seed is varied.

Focusing first on the IAD (left panel), we see that the max SIC results in the best performance. This is, of course, expected as we select on our target metric here. Selecting on max ARGOS, we are able to largely match the performance obtained from the max SIC selection. In particular, we obtain the same drop-off point, allowing us to reach the same low cross sections with both selection metrics. For the BCE-based selection, we reach a similar maximum performance at high signal injections but observe a slightly earlier drop-off. We can easily understand the worse performance of the BCE by taking into account the observations from Fig. 5.12: The classifier only found the signal at approximately epoch 40, while overtraining on the background started immediately. In this particular run, the BCE would not select a single run with a max SIC ≥ 5 . While Fig. 5.12 shows a handpicked example in order to better visualize the difference between BCE and ARGOS, this general behavior is not uncommon. Particularly for runs at low signal injections, the signal is not always found in the first few epochs, but overfitting typically begins immediately. Therefore, well-performing epochs for

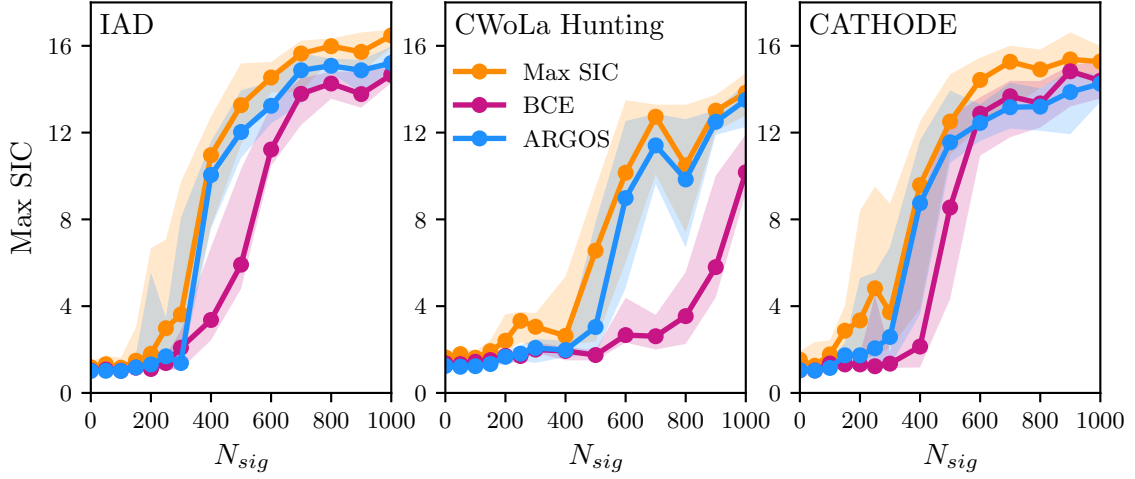


Figure 5.13: Anomaly detection performance (max SIC) as a function of the number of signal events N_{sig} after epoch selection. The epoch selection is performed using the supervised benchmark metric max SIC and the two data-driven metrics (max ARGOS and BCE), shown for IAD (left), CWoLa Hunting (middle), and CATHODE (right).

these runs are missed consistently by the BCE epoch selection, leading to the earlier drop-off observed in Fig. 5.13. Moving beyond the IAD, we find very similar behavior for CATHODE (right panel). For CWoLa Hunting (middle panel), we find a much more severe early drop-off when selecting based on BCE rather than max SIC and ARGOS. Max SIC and ARGOS selection performance still largely match.

One interesting feature seen in the epoch selection using the max SIC across all three BT generation methods is the slightly better than random performance for $N_{sig} = 0$. While we generally expect the classifier not to have any signal sensitivity if no signal is included in the training, the max SIC selects epochs based on their signal sensitivity. As classifier weights fluctuate throughout the training, some epochs are likely to have slightly better, others slightly worse than random signal sensitivity. By selecting on the max SIC, we therefore retain only those epochs with slightly better than random signal sensitivity. The result is a classifier somewhat able to detect signal without having seen any in training. Crucially, this effect is not observed for either BCE or max ARGOS. If no signal is injected, the data sets they are calculated on also do not contain any signal, making them unable to select advantageous fluctuations of NN parameters.

We select epochs using the max ARGOS in the following, as this provides the better performance of our two data-driven metrics.

5.2.4 Hyperparameter Optimization

An optimization of hyperparameters has so far not been performed systematically for weakly supervised anomaly detection. However, the performance of any machine learning algorithm depends crucially on the choice of hyperparameters. In our tests, we follow a relatively simple random sampling strategy for optimization. We perform ten separate optimizations for each signal injection, obtaining ten different hyperparameter sets for each metric at each signal

Hyperparameter	Default	Choices
Learning rate	10^{-3}	[0.01, 0.005, 0.001, 0.0005, 0.0001]
Batchsize	128	[64, 128, 256, 512, 1024, 2048, 5096]
Dropout	0	[0, 0.1, 0.2, 0.3, 0.4, 0.5]
Weight Decay	0	[0, 10^{-5} , 10^{-4} , 10^{-3} , 10^{-2}]
Momentum	0.9	[0.8, 0.9, 0.95, 0.99]

Table 5.2: Default NN hyperparameter settings and discrete choices used for hyperparameter optimization.

Hyperparameter	Default	Sampling	Min	Max
Learning rate	0.1	Log10Uniform	10^{-4}	$10^{-0.1}$
Max. Iterations	200	UniformInt	2	200
Max. Leaf Nodes	31	UniformInt	2	100
Max. Depth	None	UniformInt	2	15
Max. Bins	255	UniformInt	31	255
L2 Regularization	0	Uniform	0	10

Table 5.3: Default HGB hyperparameter settings and distributions used for hyperparameter optimization.

Hyperparameter	Default	Sampling	Min	Max
Learning rate	1	Log10Uniform	10^{-4}	$10^{-0.1}$
Number Estimators	50	UniformInt	2	100
Max. Leaf Nodes	31	UniformInt	2	100
Max. Depth	None	UniformInt	2	15
Min. Samples per Leaf	20	UniformInt	1	100

Table 5.4: Default AdaBoost hyperparameter settings and distributions used for hyperparameter optimization.

injection N_{sig} . We run each of these ten optimized classifiers once to obtain ten runs, of which we show the median and error band in the results. As such, the error band includes the fluctuation of the hyperparameter optimization.

Below, we begin with the description of the precise optimization strategy before discussing the results of the optimization. For readers not interested in implementation details, the short explanation above should suffice to understand the results.

Optimization Strategy

Our hyperparameter optimization follows a random sampling approach. We train classifiers with a total of $N_{\text{hp}} = 100$ randomly sampled sets of hyperparameters and select the best set based on each metric. Hyperparameter values are sampled according to Tab. 5.2 for the NN, Tab. 5.3 for the HGB, and Tab. 5.4 for the AdaBoost classifier, where the default values mentioned in Sec. 5.2.2 are also specified. We choose a wide range of possible hyperparameter values, including some likely to result in weak performance. This is, for example, the case when

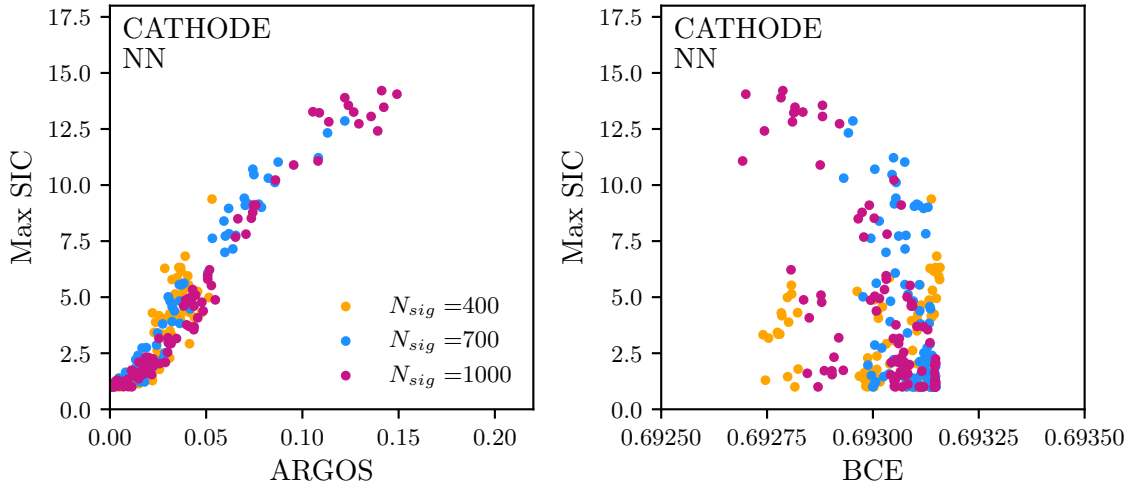


Figure 5.14: Correlation between median anomaly detection performance max SIC, and two data-driven metrics, max ARGOS (left) and BCE (right), for all 100 hyperparameter sets for CATHODE using the NN classifier at three example signal injections. It should be noted that for ARGOS, the maximum is optimal, while for BCE, the minimum is optimal.

using only two iterations for HGB and AdaBoost. Allowing for such suboptimal performance means that the difference between selections of a good and bad metric is likely to be more severe. For all metrics, we use the same classifier runs, which both reduces computation time and makes the different metrics' results more directly comparable. Additionally, we reduce computation time by adjusting classifier trainings for optimization runs compared to what is used for a full anomaly detection training. For the NN, the number of epochs is reduced to 20 for optimization runs. All metrics are calculated after every epoch. For each metric, the best value across all epochs is used to select the hyperparameter sets. For the BDT classifiers, no ensembling is used in the optimization. Instead, metrics for ten trainings with different training-validation splits are averaged. As our standard ensembling strategy randomizes training and validation sets for the BDT classifiers, no fixed held-out set of the data is given. Averaging metrics across individual BDT classifiers, therefore, ensures that metrics can always be calculated on a held-out validation set.

This procedure is conducted ten times for all $N_{hp} = 100$ hyperparameter sets and for each injected signal amount N_{sig} . As such, we obtain ten sets of optimized hyperparameters and train one classifier using each set. These ten classifiers are then used to estimate the median and error band of the performance. For a realistic analysis, one could also use these ten runs to either obtain an ensemble of well-performing but slightly different classifiers or to obtain a further stabilized hyperparameter selection. Alternatively, saving on this computation time could allow for larger or more analyses.

Results

Turning to our results, we first consider the correlation between the supervised max SIC and our two data-driven metrics across hyperparameter sets. In Fig. 5.14, we consider the median value of each metric for all 100 hyperparameter sets of the NN classifier for CATHODE for three different signal injections. We use the values from the optimization runs here, i.e.,

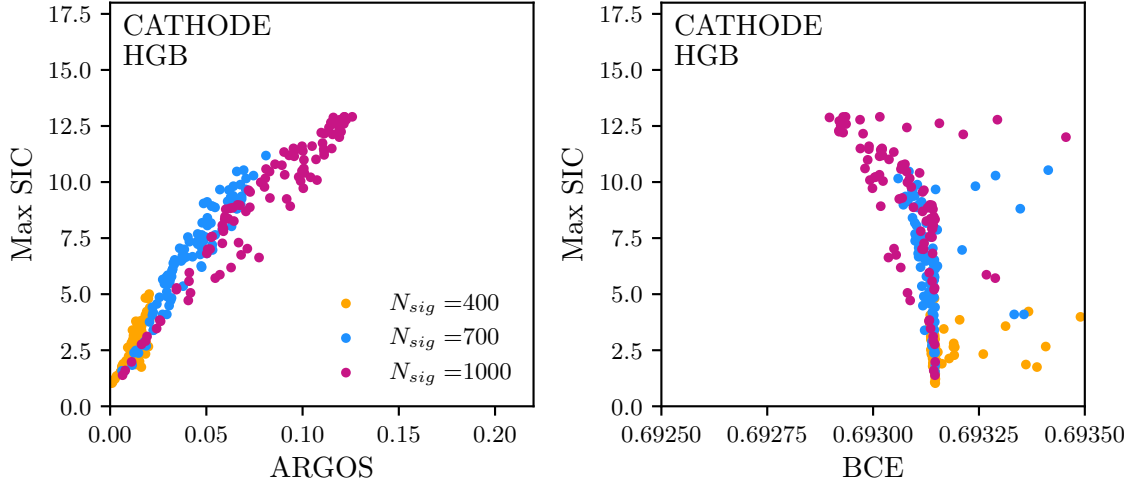


Figure 5.15: Same as Fig. 5.14 but for the HGB. Figure not included in Ref. [3].

the optimum of 20 epoch trainings. We see a very strong correlation between max SIC and ARGOS across all three signal injections (left panel). Typically, selecting the highest value of ARGOS (maximum is best) results in either the highest or one of the highest values of max SIC (maximum is best). For the correlation of BCE and max SIC (right panel), on the other hand, we only see a strong correlation for higher signal injections such as $N_{sig} = 700$ or $N_{sig} = 1000$. For the relatively low signal injection $N_{sig} = 400$, we see that the largest max SIC value is found at almost the highest BCE value. Selecting the lowest BCE (minimum is best for BCE) hyperparameter set here results in a performance loss from a max SIC of almost 10 to approximately 3. Judging from this figure, we expect to once more observe a significantly earlier max SIC drop-off for BCE-based optimization than for either max SIC-based or max ARGOS-based optimization. In Fig. 5.15, we also show correlations between data-driven metrics and max SIC for the HGB. Generally, we also observe a strong correlation between max ARGOS and max SIC here. However, we no longer observe that low BCE loss values can be obtained with poor anomaly detection performance if the signal amount is reduced. A key difference between the architectures is the binning conducted by the HGB. It is possible that this reduces background overfitting in the presence of signal, but further study would be required to definitively identify a reason for the improved correlation.

Results for the full hyperparameter optimization are shown in Fig. 5.16, with rows showing different BT generation methods and columns different classifier types. Starting with the IAD (top row), we see the expected trend very clearly for the NN (left) and AdaBoost (right). In particular, the BCE-optimized NN performance drops off very sharply between a signal injection of 600 and 700 events. The max ARGOS-optimized performance relatively closely matches the max SIC performance across all three IAD setups. For the HGB, we do not see a very large difference between max SIC, BCE, and max ARGOS. This is likely related to the improved correlation between max SIC and BCE seen in Fig. 5.15.

Overall, the trends observed for the IAD are retained as we consider CWoLa Hunting and CATHODE. However, we also see additional complications arise from these realistic background templates. In particular, we observe a widening of error bands for the NN hyperparameter optimization. Generally, a larger difference between max SIC and max ARGOS optimization is visible, although the BCE-based optimization still tends to fall short of either. Additionally,

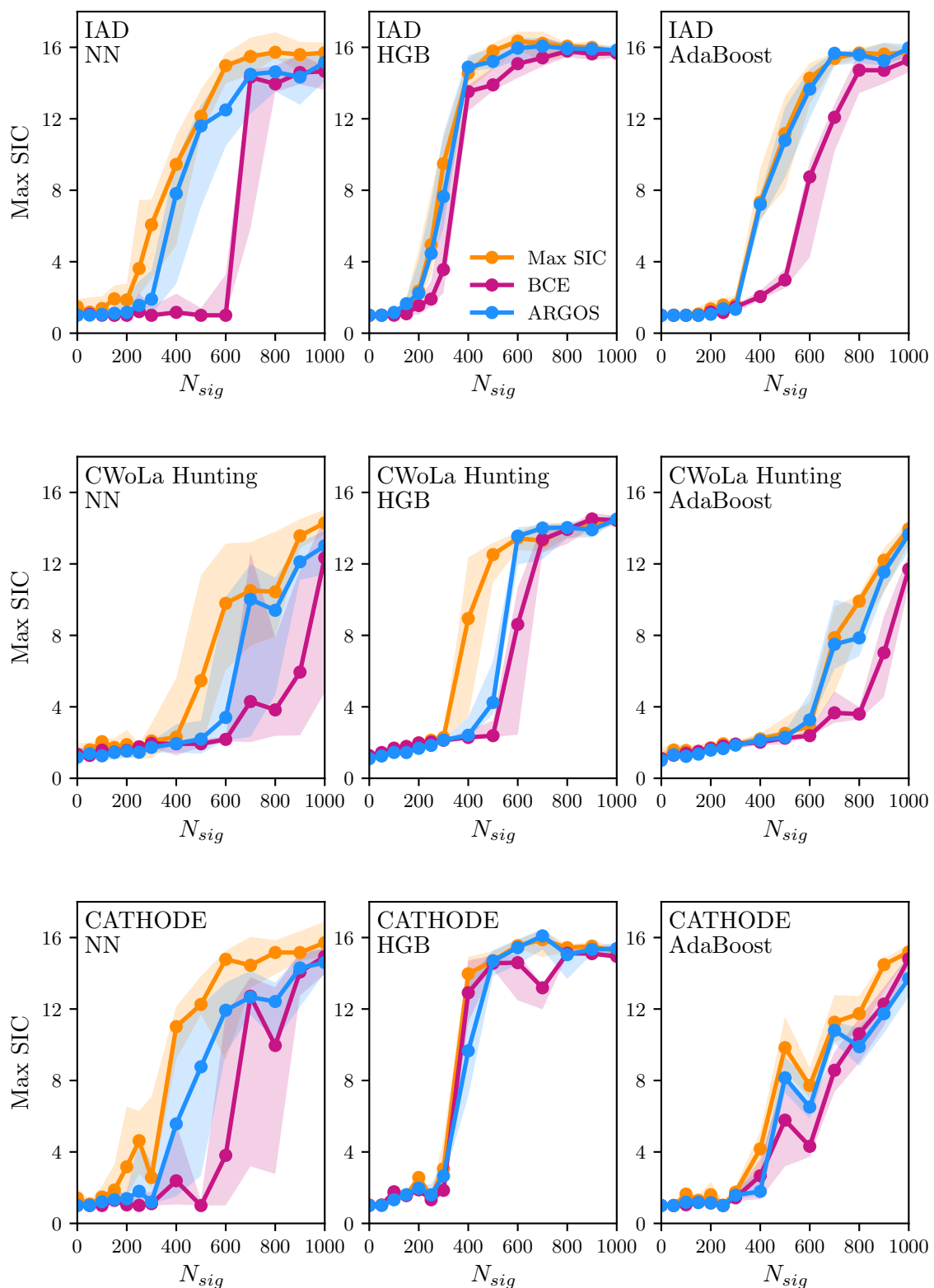


Figure 5.16: Anomaly detection performance (max SIC) after hyperparameter optimization for the NN (left), HGB (middle) and AdaBoost (right) classifiers. The optimization is performed using the supervised benchmark metric max SIC and the two data-driven metrics (max ARGOS and BCE), shown for IAD (top), CWoLa Hunting (middle), and CATHODE (bottom).

some of the curves do not show a smooth median performance. In particular, we observe a dip in the anomaly detection performance of CATHODE with the AdaBoost classifier at a signal injection of $N_{\text{sig}} = 600$. As this is not consistently observed across all tests with CATHODE, and we use the same DEs throughout our tests, it is unlikely to be caused by a badly performing DE. However, it is possible that the ten runs we perform are not sufficient statistics to obtain the true median or error band, and the observed dip is a simple statistical fluctuation.

As the primary goal of this test is to show that data-driven hyperparameter optimization is viable, no systematic analysis of the best-performing hyperparameters was conducted. However, by considering the correlations between hyperparameter values and the median max SIC of ten runs for a medium signal injection of $N_{\text{sig}} = 700$, the primary drivers of performance for the different architectures are determined. For the NN, the hyperparameters with the largest impact are the weight decay, for which smaller values are preferred, and the learning rate, for which a medium value of 10^{-3} performs best. For the HGB, the maximum depth of the tree is the primary driver of performance, with deeper, more expressive trees being preferred. A smaller correlation with anomaly detection performance is also observed for the learning rate, for which larger values lead to better performance. For AdaBoost, only the maximum depth shows a considerable contribution. It is likely that the optimal performance is obtained from the simultaneous variation of different hyperparameters, which cannot be observed in such simple 1D projections of performance versus hyperparameter values. It is also possible that the optimal performing hyperparameter setup depends on the signal injection. Additionally, there are many hyperparameters, which were not varied in this optimization, for example, the structure of the NN or its optimizer, which could also significantly impact the classifiers' performance.

As ARGOS is generally found to be the best performing data-driven hyperparameter optimization in our tests, we use the results from the ARGOS-based optimization in the following.

5.2.5 Architecture Selection

Beyond optimizing a particular chosen architecture, it is also possible to select the best performing of multiple different architectures. As we have seen with the work of Ref. [1], which was discussed in Sec. 5.1, choosing the correct architecture may not be a trivial choice and may, for example, depend on the type of features used. We first discuss our architecture selection strategy in more detail before turning to the results.

Architecture Selection Strategy

To compare different architectures, which may use the validation set differently, we ideally need a fully independent set to evaluate our data-driven metrics. At the same time, the size of our data set limits how low a cross section we may be able to detect (see also Sec. 5.1.4). For this reason, we propose to use a 50-50 split of the data set into training data and evaluation data for architecture selection and to then retrain the selected model on the full data for anomaly detection. While for statistical independence, a fully independent data set for selection and anomaly detection would be ideal, our method preserves our available statistics. Additionally, the full retraining of the model means that any overfitting of the half-statistics model should only have a limited effect on the final anomaly detection performance.

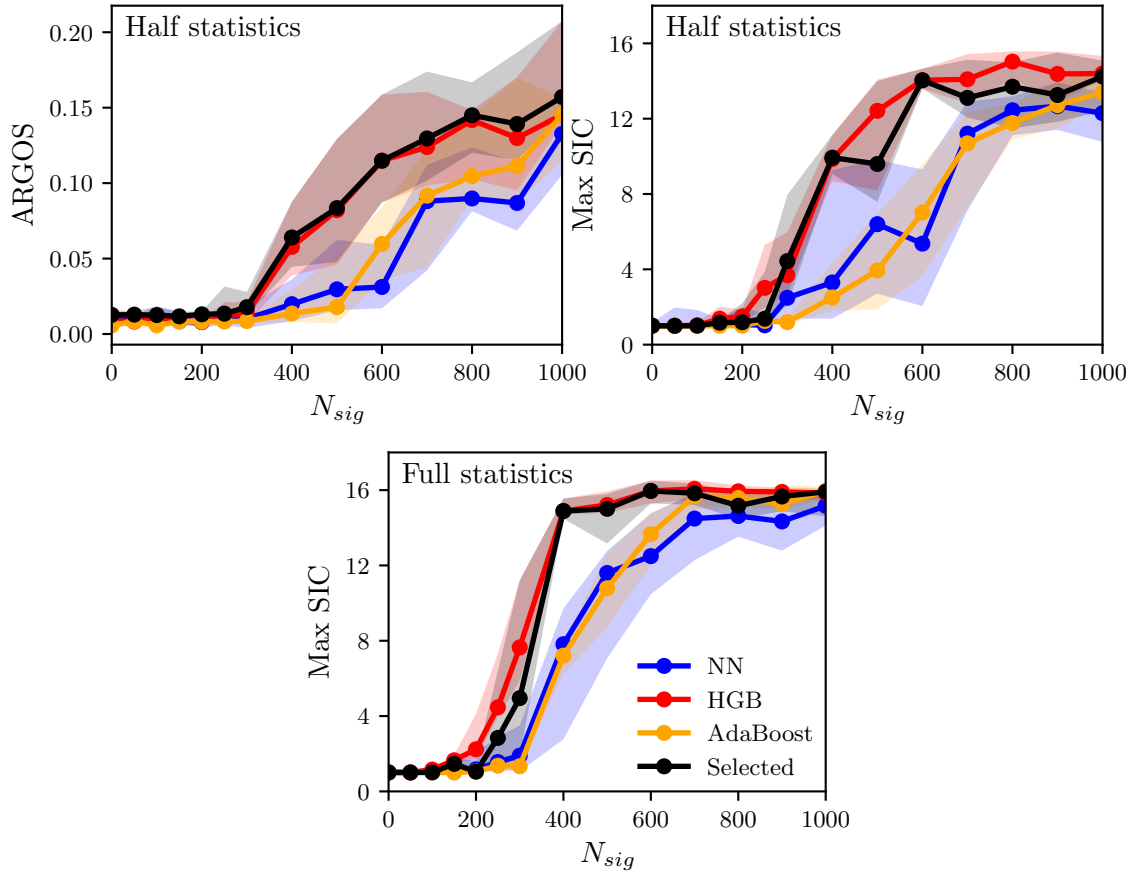


Figure 5.17: Architecture selection for IAD using max ARGOS. Top left: Max ARGOS calculated on half the data when trained on the other half. Top right: Max SIC of half data trainings. Bottom: Max SIC of full statistics training. In each case, the selected runs are shown in black. Note that the x-axis shows the full statistics signal injection values even for the half statistics runs.

We show the different stages of our architecture selection strategy in Fig. 5.17 for the IAD using the max ARGOS metric to explain the procedure in detail. First of all, we require architecture candidates. For these, we use the three different architectures with hyperparameters chosen as optimized in Sec. 5.2.4. Optimizing the hyperparameters for the selection is not strictly necessary. Instead, one could also compare hyperparameters chosen according to common defaults. The architecture proposals may also include differences in how validation data is used, as is the case for our architectures, where the NN only uses validation data to select epochs, whereas the HGB and AdaBoost do not have a fixed validation set at all. All of the architecture proposals are then trained on the first half of the data, which is designated as the training data. We use the other half of that data to evaluate our data-driven max ARGOS and BCE metrics using the full ensembling procedure. This 50-50 split is needed to avoid biased results caused by the different treatment of the validation data. The top left panel of Fig. 5.17 shows the max ARGOS evaluated for the different architectures. Error bands include the ten different hyperparameter sets per architecture and signal injection. We always compare one set of hyperparameters per architecture to obtain ten independent architecture selection runs. The result of the selection is shown as the black line with error band. We see that the result lies slightly above the best individual architecture (HGB) at some signal

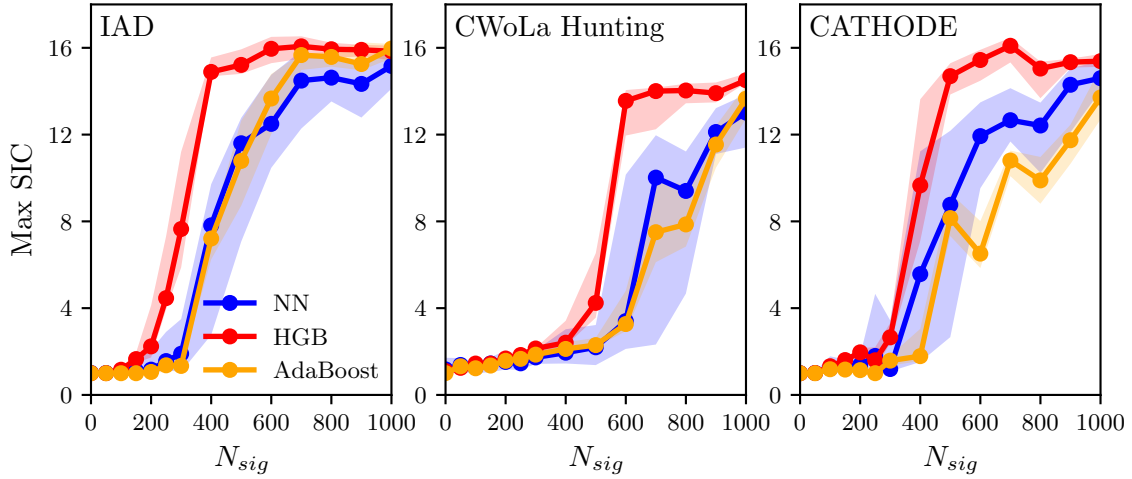


Figure 5.18: Comparison of different architectures' performance with hyperparameters optimized using max ARGOS. Anomaly detection performance (max SIC) shown for IAD (left), CWoLa Hunting (middle), and CATHODE (right). Figure not contained in Ref. [3].

injections, as a well-performing NN or AdaBoost run is chosen where it performs better than the respective HGB run. The top right panel of Fig. 5.17 shows the max SIC for the same runs trained on half of the data statistics. We see that the performance of the selected runs largely matches that of the best-performing individual model (HGB), although some runs were selected, which are suboptimal according to the max SIC.

Having selected an architecture based on the 50-50 split of the data, the final anomaly detector is then trained on the full data set statistics. We show the corresponding result in the bottom panel of Fig. 5.17, comparing with the individual architectures' performance. Generally, we see that the ordering of anomaly detection performance between the different architectures is preserved when moving from half to full statistics. The HGB still performs best, and the NN and AdaBoost performances largely match. The selected runs largely match the anomaly detection performance of the best-performing architecture. Some slight fluctuations to suboptimal selections are observed.

Results

To fully understand the results of the architecture selection, we first take a look at the possible choices. For this purpose, we compare the performance of each architecture after HP optimization using max ARGOS in Fig. 5.18. Across all BT generation methods and signal injections, the HGB generally performs best. The performance difference is most significant around the drop-off point for the classifiers, where the HGB is typically able to still detect slightly lower signal injections than the other two architectures. These results are also compatible with the observations from Ref. [1] discussed in Sec. 5.1.

Having understood these general trends, we now compare the performance of the different metrics for architecture selection in Fig. 5.19. Starting again with the IAD (left panel), we observe the same performance trends seen previously for the epoch selection and hyperparameter observation. In particular, the performance when selecting based on max SIC and max ARGOS largely matches, while BCE-based selection leads to a slightly earlier performance

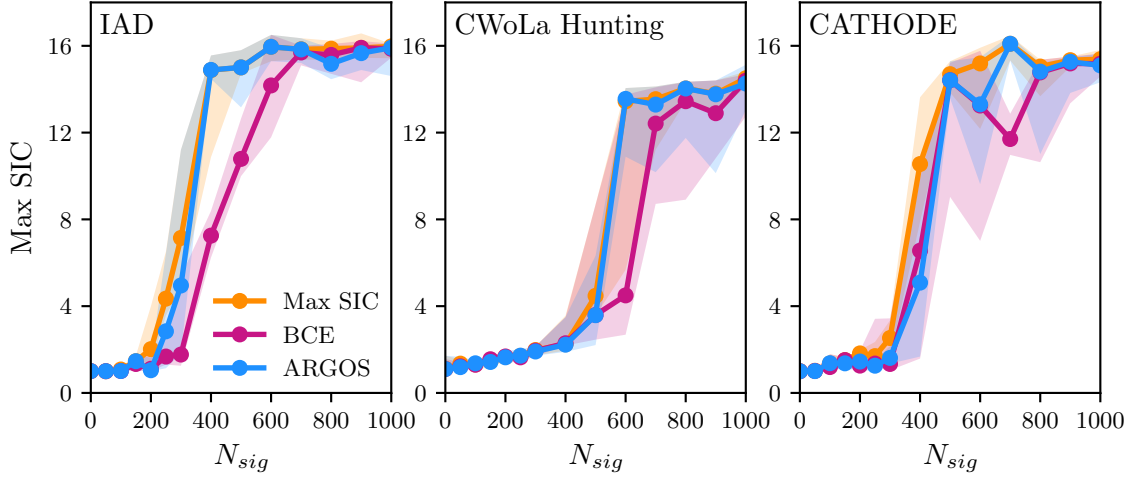


Figure 5.19: Anomaly detection performance (max SIC) after architecture selection based on the supervised metric max SIC as a benchmark and the two data-driven metrics max ARGOS and BCE for IAD (left), CWoLa Hunting (middle), and CATHODE (right).

drop-off. Comparing this with the different architectures’ performance in Fig. 5.18, we see that while max SIC and max ARGOS relatively consistently select the best performing architecture, namely HGB, in the region around the drop-off, BCE does not, matching better with the shape of the NN and AdaBoost performances.

We see this trend preserved for CWoLa Hunting (middle panel), for which max SIC and max ARGOS still match very closely. For the BCE-based selection, a lower median performance with a large error band is observed. The BCE-based selection selects both the HGB and the other architectures approximately half the time. Similar results are seen for CATHODE (right panel), where max SIC and max ARGOS-based selection still perform very similarly, while in particular the error band of the BCE-based selection is much larger, as the worse-performing architectures are sometimes selected.

5.2.6 Feature Selection

Beyond selecting the model itself, it may also be interesting to select the feature set for anomaly detection. For this purpose, we reconsider the extended feature sets used in Sec. 5.1. As a quick reminder, these feature sets include additional information in the form of more subjettness ratios and subjettnesses, subsequently adding information. In addition to these extended feature sets, we also consider a feature set we call “Baseline + ΔR ”, where the dijet angular distance ΔR is added to the baseline feature set, on which our tests have been conducted thus far. We add this feature set as ΔR is highly correlated with the dijet mass, which makes the SSBs an inaccurate representation of the SR background, impacting the BT quality for CWoLa Hunting. The feature sets are summarized in Tab. 5.5.

With these feature sets, we once again use the procedure from the architecture selection, i.e., selecting based on half-statistics runs and retraining on the full data set statistics to obtain the anomaly detection performance. We use only the HGB classifier with default hyperparameters here, as we have seen in Sec. 5.1 that these perform well for a variety of

Name	Feature No.	Features
Baseline	4	$\{m_{J1}, \Delta m_J, \tau_{21}^{\beta=1,J1}, \tau_{21}^{\beta=1,J2}\}$
Baseline + ΔR	5	$\{m_{J1}, \Delta m_J, \tau_{21}^{\beta=1,J1}, \tau_{21}^{\beta=1,J2}, \Delta R\}$
Extended set 1	10	$\{m_{J1}, \Delta m_J, \tau_{N,N-1}^{\beta=1,J1}, \tau_{N,N-1}^{\beta=1,J2}\}$ for $2 \leq N \leq 5$
Extended set 2	12	$\{m_{J1}, \Delta m_J, \tau_N^{\beta=1,J1}, \tau_N^{\beta=1,J2}\}$ for $N \leq 5$
Extended set 3	56	$\{m_{J1}, \Delta m_J, \tau_N^{\beta,J1}, \tau_N^{\beta,J2}\}$ for $N \leq 9$ and $\beta \in \{0.5, 1, 2\}$

Table 5.5: Summary of the feature sets considered for the feature selection. Details of the observables are given in the text.

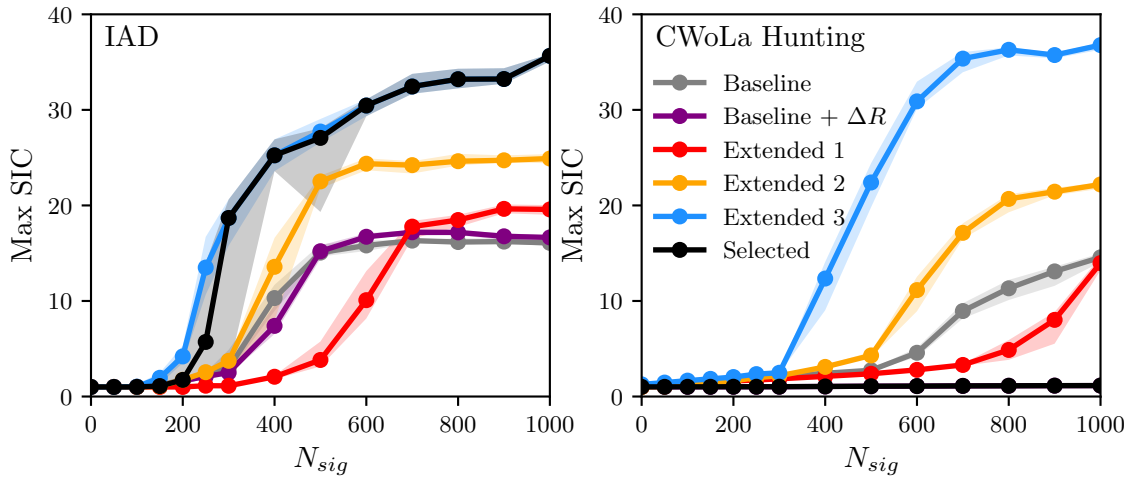


Figure 5.20: Anomaly detection performance max SIC for different feature sets and for the feature set selection using max ARGOS (black). We use the HGB classifier with default hyperparameters for both the IAD (left) and CWoLa Hunting (right). The feature sets are defined in Tab. 5.5. For CWoLa Hunting, “Baseline + ΔR ” feature set is always selected and as such fully hidden behind the black “Selected” line. Only results for the IAD without the “Baseline + ΔR ” feature set were included in Ref. [3].

feature sets. Additionally, we only perform a max ARGOS-based selection. Fig. 5.20 shows the performance of the IAD and CWoLa Hunting for all 5 feature sets as well as the selected runs. The ten runs for the specific feature sets vary only in the classifier training. The selection is performed for each of these ten runs separately to obtain a realistic uncertainty. Selecting based on the median or average metric of multiple runs per feature set would likely be able to further stabilize the selection.

Starting with the IAD (left panel), we see that the best feature set is Extended set 3, as this contains the most information, with most other feature sets being subsets of this feature set. ΔR does not appear to add significant information, as the performance of the Baseline and Baseline + ΔR feature sets largely match. Largely, Extended set 3, i.e., the best performing feature set, is selected by max ARGOS. Some fluctuations are observed, with a few selections of other feature sets occurring particularly at lower signal injections. For CWoLa Hunting (right panel), the picture differs starkly. Here, we can see that the Baseline + ΔR feature set is consistently selected despite not providing the best anomaly detection performance.

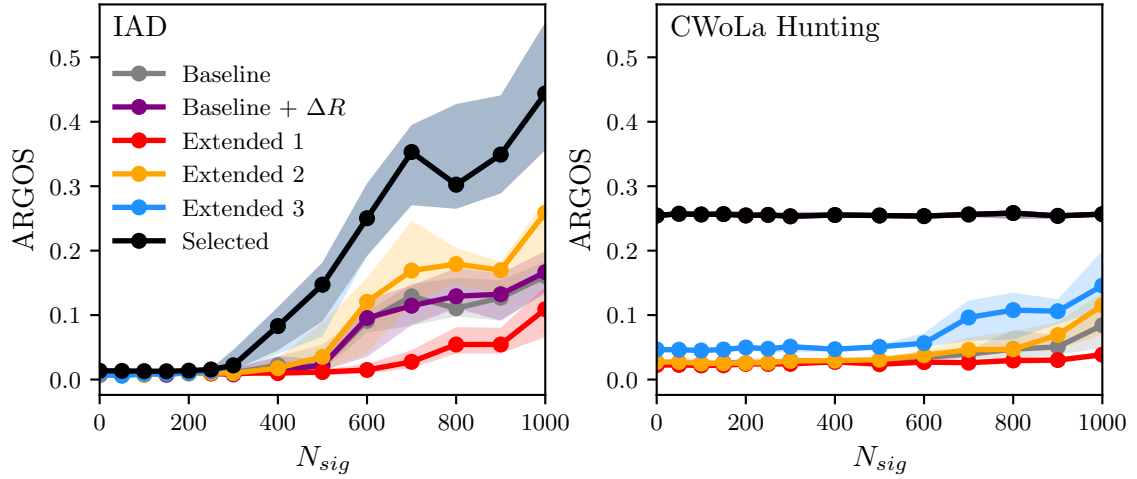


Figure 5.21: Max ARGOS calculated on half the data for a classifier trained on the other half for different feature sets. We use the HGB classifier with default hyperparameters for both the IAD (left) and CWoLa Hunting (right). The runs selected based on max ARGOS are shown in black. The feature sets are defined in Tab. 5.5. For CWoLa Hunting, “Baseline + ΔR ” feature set is always selected and as such fully hidden behind the black “Selected” line. Results were not shown in Ref. [3].

The worse BT quality when adding a feature correlated with the dijet mass affects the max ARGOS more than the high anomaly detection performance of Extended set 3. We investigate this further in Fig. 5.21, where we show the max ARGOS of each feature set as well as of the selected runs. For the IAD (left panel), the ordering is equivalent to the ordering of the anomaly detection performance as shown in Fig. 5.20. This is not the case for CWoLa Hunting (right panel). Here, we observe an offset of the max ARGOS for the Baseline + ΔR feature set, which is largely independent of the signal injection. Any such deviation from zero at $N_{sig} = 0$ can be seen as a direct impact of the background template quality. We can see such a deviation not just for the Baseline + ΔR set but also to a lesser extent for Extended set 3. For low signal injections up to $N_{sig} = 600$, Extended set 3 would be selected for its inferior background template rather than its anomaly detection performance. In this case, this inferior background template is caused by the increase in the number of features compared to the other feature sets and not by strong correlations with the dijet mass. However, it should be noted that for higher signal injections going beyond the values tested here, we expect to select Extended set 3 even with the inclusion of Baseline + ΔR for its superior anomaly detection performance.

Overall, these results show that with perfect background templates, a feature selection on data is viable. However, for feature selection to be useful in a realistic analysis, it would require more work. For realistic feature sets, which may contain different numbers and types of features as was shown here, CWoLa Hunting will always have different background template qualities. This also goes for DE-based background template generation methods such as CATHODE, which are particularly affected by the number of features, as we have also seen in Sec. 5.1.5. For any such realistic background template generation methods, a two criterion selection taking into account both background template quality and anomaly detection performance must therefore be conducted. This could, for example, be implemented by performing a no-signal calibration of the data-driven metric. For this, a sideband versus sideband classification for

CWoLa Hunting or a sideband BT versus sideband data classification for CATHODE may be used. Whether this suffices to eliminate the current issues inherent to feature selection requires further study.

5.2.7 False Discoveries and Background Sculpting

When optimizing on data, there may be some concern about increasing the rate of false discoveries. We discuss and investigate whether and to what extent this may be the case in this section. For this purpose, we start by considering the possible reasons for an increased rate of false discoveries, which are two-fold:

Firstly, it is possible that one may experience a look-elsewhere-like effect, i.e., that by choosing the best of multiple options, the likelihood of observing a larger deviation is increased. As an example, we can use epoch selection: Even if a neural network sees only background events in both classes and no overfitting occurs, weights fluctuate throughout training. Any metric determining differences in the classifier scores between SR and BT is sensitive to the data statistics and will select one of the better fluctuations of weights⁴. If the same data were used to evaluate the significance, we expect to find slightly larger differences in the selection of events from SR and BT than for a random selection. While we theoretically enhance such effects by selecting models and epochs in a data-driven manner, this effect can be eliminated fully through the use of an independent test set, as we also discuss below in Sec. 6.2.

Secondly, it is possible that selecting the best model results in being better able to detect inaccuracies of the background template. As these inaccuracies are due to an imperfect background template quality and not statistical fluctuations, they would still exist on an independent test set. In general, a classifier, which is better able to detect differences between background template and SR data, is the better classifier even if these differences are due to an imperfect background template. However, selecting this classifier may still lead to false discoveries by enhancing the significance in the background-only scenario. What this looks like depends crucially on the method used to obtain the significance. In the purely cut and count-based method, which is discussed in detail below in Sec. 6.1, a systematic bias is determined, which quantifies the typical overselection in the SR compared to the BT in the background-only case. If determined correctly, i.e., for the final classifier setup, this would absorb any enhanced overselection into a larger value of the systematic bias. On the other hand, for a fit-based determination of significances, the crucial question is whether background sculpting – i.e., changes in the shape of the background spectrum in the resonant feature caused by a cut on the classifier score (see also Sec. 4.3.5) – is observed to a larger extent. We investigate this below.

To quantify the extent of background sculpting, we perform NN classifier trainings with default hyperparameters and without injecting signal, i.e., in the background-only case. We then evaluate the degree of sculpting observed when selecting epochs for ensembling using validation BCE and max ARGOS, as was investigated in Sec. 5.2.3, and compare this to the sculpting observed when using the predictions of the last epoch. As the selection using the validation BCE loss was used in the model-agnostic CMS analysis [46], we know that this selection does not lead to sculpting prohibitive to the analysis as a whole. To quantify the background sculpting, we use the metrics proposed in Ref. [38], where a bin-based χ^2 metric

⁴ We have seen this for the anomaly detection performance when using max SIC to select epochs for the IAD at $N_{\text{sig}} = 0$ in Fig. 5.13.

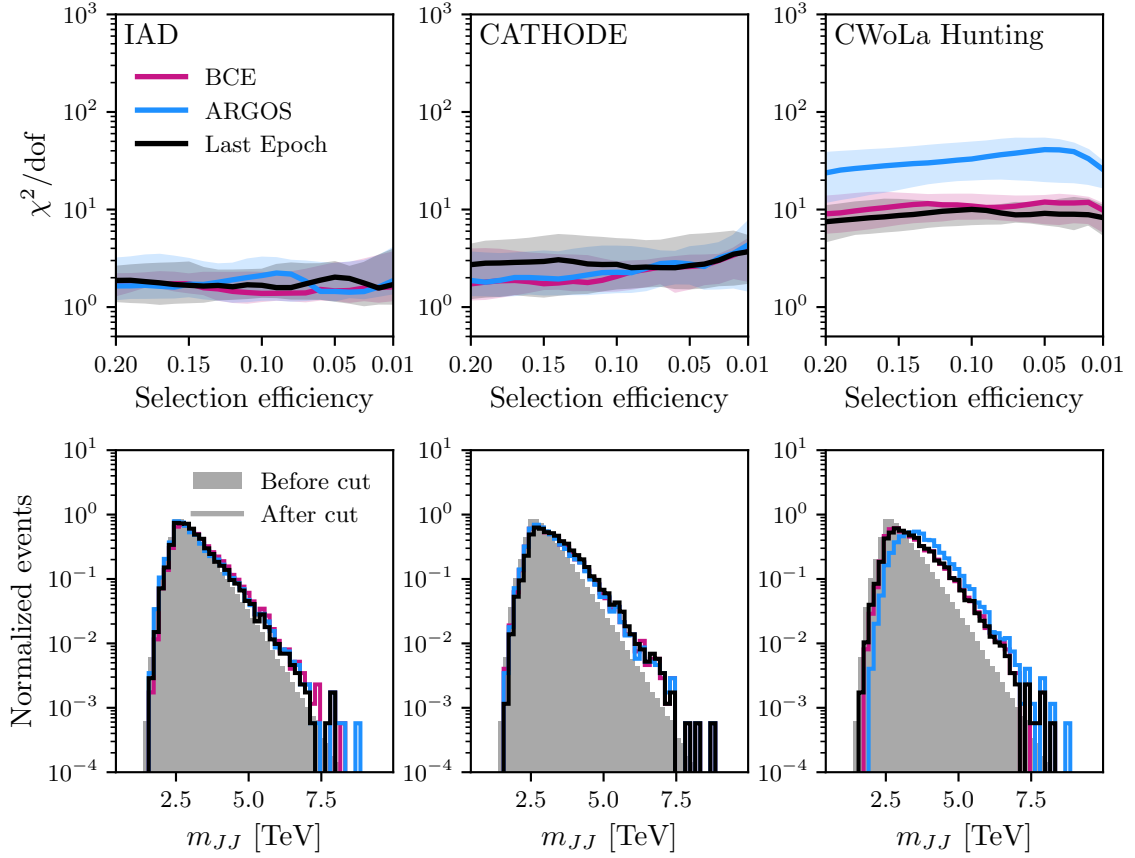


Figure 5.22: Background sculpting for IAD (left), CATHODE (middle), and CWoLa Hunting (right) when selecting epochs based on validation BCE loss, max ARGOS, or selecting the last epoch characterized by the χ^2 metric from [38] (top) and example spectra obtained when selecting 1% most anomalous events (bottom).

was chosen, which compares the spectrum before and after the cut assuming Poisson statistics. We calculate the χ^2 for a variety of different cuts corresponding to specific selection efficiencies in the SR. This metric is explained in more detail in App. A.2.

The results of our background sculpting tests are shown in Fig. 5.22. The top row shows the χ^2 -metric per degree of freedom for ten classifier trainings per background template generation method as median and error band. Only the random initializations differ between the runs. The bottom row shows normalized example spectra comparing the full background spectrum before the cut (shaded grey area) and the spectrum after the cut for each of the epoch selection methods (colored lines). Starting with the IAD (left panels), we can see that the χ^2/dof is approximately 2 across all selection efficiencies. No difference is visible between the epoch selection methods. Here, no background template mismodeling affects the classifier score. However, as the features are slightly correlated with the dijet mass, some correlation of the classifier score with the dijet mass is expected. This small correlation leads to some sculpting, resulting in the slightly enhanced value of $\chi^2/\text{dof} = 2$ compared to the expected value for a random selection of events, which would be 1. This slight sculpting can also be seen in the spectrum (bottom). It should be noted that we see only a slight change in the slope and no artificial bumps. A standard fit of a smoothly falling function should not be

affected by this change.

Moving to the realistic background templates with CATHODE (middle panels), we see a slightly increased value of χ^2/dof . None of the epoch selection methods differ significantly here either, and the spectrum still shows a smoothly falling shape. More interesting are the results for CWoLa Hunting (right panels). Here, we already see a much larger degree of sculpting than for IAD and CATHODE when using the last epoch or selecting using the BCE with $\chi^2/\text{dof} \approx 10$. This is also visible in the spectra. However, the sculpting is further exacerbated when selecting epochs using the max ARGOS, for which we see both an increase in the χ^2/dof to approximately 30 and a clear change of the spectrum shape resulting in a maximum at approximately the position of the SR ($m_{JJ} \approx 3.5 \text{ TeV}$). Indeed, this increase is to be expected for CWoLa Hunting, where the data-driven metrics aim to increase the difference in selection between SR and SB data. The ARGOS metric effectively picks up on this difference, which is caused by correlations of classification features with the dijet mass, and, as such, enhances the sculpting.

Overall, these results show that there is no general issue with switching to a better metric for the optimization task. However, the results for CWoLa Hunting show that the effect of setup changes on other elements of the analysis, such as the significance evaluation, should be carefully considered. For CWoLa Hunting, maximizing ARGOS is directly linked with increasing sculpting. As such, an analysis using a fit-based significance evaluation may need to use other metrics to select epochs or account for the increase in sculpting in the analysis. Beyond epoch selection, we expect results to be less extreme as both hyperparameter optimization and architecture selection retrain networks from scratch after the optimization. However, further studies are required to make definitive statements.

5.2.8 Conclusion and Outlook

In the above studies, we have shown how a variety of different optimizations of weakly supervised anomaly detection setups can be conducted in a data-driven manner. For these tasks, it is crucial to use a data-driven metric highly correlated with the actual anomaly detection performance. We introduced such a metric with ARGOS, which relies solely on the existence of a high-quality background template. Fundamentally, this metric places a cut on the classifier score and calculates the significance improvement using the mixed dataset assumption underlying weakly supervised anomaly detection. Crucially, if chosen appropriately, the cut on the classifier score allows ARGOS to ignore much of the background, which dominates anomaly detection data sets. Therefore, ARGOS focuses on events labeled as signal-like by the classifier. The statistical foundation of this metric at the same time ensures a strong correlation of the metric to the signal enhancement we are interested in.

We have tested the ARGOS metric by comparing it to the supervised optimum and the BCE loss as an often-used benchmark in a variety of different applications. In this context, we have discussed both applications used in previous analyses, such as the selection of the best epochs, and those which have not been systematically conducted in the past, such as hyperparameter optimization and architecture selection. In all cases, we observed the same general trend, namely that an optimization based on the BCE loss leads to suboptimal results, while the ARGOS metric typically performs relatively close to the supervised optimum. In particular, classifiers optimized using ARGOS were typically able to still detect the signal at lower signal injections than those optimized on the BCE loss. Hence, the signal can be detected with lower

statistics from experiment or at lower cross sections.

While the performance gain is already significant, ARGOS has several additional distinct advantages compared to the BCE loss. Firstly, one can optimize the selected cut for an analysis using ARGOS. In our work, we have used the max ARGOS, which selects the optimal ε_{BT} as a working point. This working point can then be used for an analysis. Additionally, if the working point is chosen in advance, an optimization of classifiers specifically for this working point is also possible by simply fixing ε_{BT} . Secondly, ARGOS does not require the underlying training task to be a classification task. Crucially, the BCE interprets inputs as class probabilities, and differently normalized classifiers lead to different values of the BCE loss even if the classifiers have the same relative ordering of events. With a cut-based metric like ARGOS, this is not the case. Therefore, an application to other anomaly detectors is also possible so long as a background template is derived to evaluate them. Hence, ARGOS can also serve to compare different types of anomaly detectors and select the optimal one.

Beyond optimizations of the classifier itself, we also investigated a data-driven selection of the best feature set. In an idealized scenario, feature selection showed promise. However, an application to realistic background templates requires further work to remove a bias toward feature sets with worse background templates. Further work is also required to investigate the effect of data-driven optimization on the rate of false discoveries. We have discussed a number of possible concerns and performed first studies to investigate how background sculpting may change with different epoch selection methods. While the results of these tests do not show any insurmountable problems for an analysis, only an investigation of the full analysis pipeline will provide conclusive results.

6

OBTAINING AND CALIBRATING THE SIGNIFICANCE

The outcome of an analysis is ultimately expressed in terms of its statistical significance. Obtaining a significance, which makes use of the full potential of an analysis and is still properly calibrated, i.e., it does not return a higher rate of false discoveries than is statistically expected, is therefore crucial. We approach this problem from two sides in this chapter: First, in Sec. 6.1, we discuss a simple cut and count-based approach to significance extraction, which is easy to control and allows us to use small background efficiencies, which typically yield the largest significance improvements. Second, in Sec. 6.2, we investigate whether anomaly detectors incur a look-elsewhere-like effect to understand the statistical behavior more fundamentally.

6.1 SIGNIFICANCE EXTRACTION BY DIRECT BACKGROUND ESTIMATION

This section is largely based on Ref. [2]. The code for this project can be found in Refs. [183, 185]. Preliminary studies of this method were submitted as part of my master's thesis [186]. Some follow-up studies of this method, included in Sec. 6.1.8, were performed by Mark Fanselow and submitted as part of his master's thesis [187], which I co-supervised.

In a standard resonance search, determining the significance of the excess of events in the signal region almost always involves fitting the background spectrum. This fit is used to obtain a background expectation in the signal region. However, in weakly supervised anomaly detection, the classifier typically does not see the full invariant mass spectrum during training, leading to uncontrolled extrapolation behavior in the sidebands. This background sculpting can make a fit to the sidebands difficult to perform (see also Secs. 4.3.5 and 5.2.7). To avoid having to perform a fit at all, we seek to perform a fully cut and count-based significance extraction, which we call “direct background estimation”. Fundamentally, this approach works as follows:

In weakly supervised anomaly detection, we have two separate data sets on which the classifier is trained, namely the background template and the signal region data. The background template is typically designed in such a way as to approximate the SR background, while the SR data contains background and signal – although it is possible that the signal fraction f_S is zero. The statistical analysis must correctly identify when this is the case. Otherwise, a false discovery would be obtained. Assuming that the background template is accurate, a cut on the classifier should retain the same fraction of background events in both data sets. Thus, we can use the BT to obtain a background expectation and compare the selection efficiency of the SR data to this value. This simple assumption underlies the methodology of our entire approach. However, as we will see, realistic background templates contain some mismodeling, leading to a systematic increase of selected SR events, which needs to be taken into account to avoid false discoveries.

Beyond avoiding sculpting, direct background estimation has a distinct advantage over a fit-based bump hunt. To perform a fit to the background, background statistics must be sufficient. This limits the background efficiency for an analysis. In the past, CMS and ATLAS analyses [44–46], no cut retaining less than 1% of the SR data was chosen. However, the significance improvement of weakly supervised classifiers is typically largest at larger background rejections. In a cut and count-based approach, a significance can still be obtained for infinite background rejections, allowing us to work with larger significance improvements.

In this work, we test both a background-only case to ensure that no false discoveries occur and a benchmark case with signal to ensure that the analysis retains its sensitivity. We start by discussing the setup for these tests in Sec. 6.1.1 before describing the methodology of direct background estimation in detail in Sec. 6.1.2. We then turn to our tests, which we conduct for the Idealized Anomaly Detector (IAD) in Sec. 6.1.3, for CWoLa Hunting in Sec. 6.1.4 and for CATHODE in Sec. 6.1.5. We discuss a number of follow-up studies concerning a randomization of signal events in Sec. 6.1.6, an extension to larger feature sets in Sec. 6.1.7, and a more accurate estimation of the systematic error for CWoLa Hunting in Sec. 6.1.8. We conclude in Sec. 6.1.9.

6.1.1 Setup

Data set

We perform studies for direct background estimation using the LHC Olympics R&D data set (see also Sec. 4.3.1). This data contains 1 million QCD dijet background events and 100 000 signal events. The signal is a resonant production of a new W' boson with a mass $m_{W'} = 3.5$ TeV, which decays into an X and Y boson simulated using *Pythia* and *DELPHES*. Both X and Y decay into two quarks each. To characterize this signal, we use the dijet invariant mass m_{JJ} of the two highest p_T jets as the resonant feature. For the classification, we largely focus on the “Baseline” features (see also Sec. 4.3.2), which contain the mass of the lighter jet m_{J1} , the difference of the two jet masses Δm_J , and the 21-subjettiness ratios $\tau_{21}^{J1/J2}$ for both jets. Additionally, we also use the “Baseline + ΔR ” feature set, which uses the dijet angular distance ΔR for the classification in addition to the four Baseline features. We include this set as ΔR is correlated with the dijet mass. This correlation leads to a low background template quality for CWoLa Hunting, allowing us to test the robustness of direct background estimation.

In addition to the LHC Olympics R&D data set, we also use the LHC Olympics Black Box 2 data. This data consists of 1 million QCD dijet events generated using *Herwig* with the same *DELPHES* detector card as was used for the LHC Olympics R&D data set. In our studies, we use this set to represent an imperfect Monte Carlo simulation of the background, as would be available in experiment. Further, we also use a larger set of 2 million QCD dijet events and 100 000 signal events generated in Ref. [188] using the settings of the LHC Olympics R&D data set for tests of the Idealized Anomaly Detector.

For the analysis, we perform a scan over the signal region position. Here, we are limited by the generation cut for the leading jet ($p_T > 1.2$ TeV). This cut results in a trigger turn-on region up to approximately $m_{JJ} = 2.7$ TeV, which can be seen in Fig. 4.4 in Sec. 4.3.1. For higher mass values, low data set statistics become a limiting factor. For this reason, we choose to use only nine strongly overlapping windows of a width of 400 GeV and with centers at

$m_n = (3.5 - 0.1 \cdot (5 - n))$ TeV. The signal is centered at window 5. The large overlap leads to most of the signal still being contained in windows 4 and 6, as well. For our tests, we perform both background-only studies, for which we inject no signal, and studies using a benchmark signal injection of 1000 events. We use a fixed set of signal events unless otherwise specified. This benchmark is approximately equivalent to an initial significance of 2.2σ in window 5.

All studies use k -fold cross validation with $k = 5$. k -fold cross validation splits the full data set into k separate folds and uses part of the folds for training (in our case, $k - 1$ folds) and the rest as an independent test set. By repeating this k times, using each fold as a test set once, predictions are obtained for the full data set. This retains the full statistics of the data set for evaluation while ensuring that each classifier is applied only to an independent test set. In the studies discussed in this section, we assume that results for all k folds are fully statistically independent, an assumption we test in Sec. 6.2.

Weakly Supervised Methods

We perform studies of direct background estimation using three different background template generation methods. First, we use an ideal background template with the IAD to show that the fundamental assumptions on which our method is built are valid. Then, we use CWoLa Hunting and CATHODE to show that a realistic analysis using direct background estimation does not report false discoveries, but at the same time remains sensitive to an injected signal. We define the respective background templates as follows:

IDEALIZED ANOMALY DETECTOR For the IAD, we use the larger Pythia data set containing 2 million QCD background events and 100 000 signal events described above. We split this data set into an LHC R&D data-like set using half of the background and all of the signal events, and a background-only set containing 1 million background events only. The background-only set is then used as a background template, while the LHC-like set is used to prepare the data set as normal.

CWOLA HUNTING For CWoLa Hunting, we use short sidebands of a width of 0.2 TeV each as the background template for each signal region.

CATHODE For CATHODE, we train flow matching density estimators based on Ref. [41] (code published in Ref. [183]). For training, we use the full sideband data corresponding to each signal region. For sampling, the m_{JJ} distribution in the SR is estimated using a kernel density estimate. This distribution is then sampled to obtain mass values, for which samples of the CFM density estimator are obtained. We oversample the SR data by a factor of four in the background template. We train ten density estimators for each signal region position to obtain an estimate of the fluctuations between density estimators. A more detailed description of the DE can be found in App. A.1

Classifier

The classifier setup is based on the boosted decision tree classifier as discussed in Sec. 5.1. We use the `HistGradientBoostingClassifier` from `scikit-learn`, largely using default

hyperparameters, changing only the maximum number of iterations to 200 and the validation fraction to 0.5. Early stopping on the validation loss with a patience of ten iterations is applied. Our full classifier is an ensemble of 50 BDTs with randomized training and validation splits.

6.1.2 Direct Background Estimation

To estimate the significance of an observed excess of events using direct background estimation, we require three components: an anomaly detector $R(x)$ ¹, a set of SR data and a set of the BT. We obtain a significance by making direct use of the assumptions underlying weakly supervised anomaly detection (see Sec. 4.2), namely that the SR data is made up of

$$p_{\text{SR}}(x) = f_S \cdot p_S(x) + (1 - f_S) \cdot p_B(x), \quad (6.1)$$

where the signal fraction f_S is zero in the background-only case. Additionally, our background template approximates the SR background, and as such, an ideal background template is given by

$$p_{\text{BT,ideal}}(x) = p_B(x|m \in \text{SR}). \quad (6.2)$$

If we now place a cut using the anomaly detector such that only events with $R(x) > R_c$ are selected, we can compare the selections on our SR data and BT. For both sets, we know the event numbers before the cut $N_{\text{SR/BT}}$ and the event numbers after the cut² $N'_{\text{SR/BT}}$. We call the number of events in the SR after the cut our observation $N_{\text{obs}} = N'_{\text{SR}}$ and want to quantify whether this observation is compatible with the background-only hypothesis. As such, we need to use the other three counts to derive an expectation N_{exp} to compare the observation to. Generally, we expect to retain

$$N_{\text{exp}} = \varepsilon_B N_{\text{SR}} \quad (6.3)$$

events in the SR for any background efficiency ε_B . Under the assumption that the background template is ideal, we can estimate

$$\varepsilon_B = \varepsilon_{\text{BT,ideal}} = N'_{\text{BT}}/N_{\text{BT}}. \quad (6.4)$$

on the background template. However, for any realistic analysis, it is not possible to obtain an ideal background template. Instead, we have to use a method to generate a background template from data, in our studies using CWoLa Hunting and CATHODE. For these realistic background templates, slight mismodeling is unavoidable and will be learned by the classifier. This leads to a shift between ε_B and ε_{BT} , which needs to be quantified to obtain an accurate expectation. We define the relative systematic shift δ_{sys} by

$$N_{\text{exp}} = \varepsilon_{\text{BT}} N_{\text{SR}} (1 + \delta_{\text{sys}}(\varepsilon_{\text{BT}})). \quad (6.5)$$

For an ideal background template, we obtain $\delta_{\text{sys}} = 0$ by definition. A reliable estimate of the systematic shift δ_{sys} and its error σ_{sys} is crucial. We use two methods for this estimation, one based on simulation and the other data-driven.

¹ Below, the anomaly detector is also sometimes called the classifier, as this is what our tests use. However, in theory the procedure is not limited to classifier-based anomaly detection.

² In our studies using k -fold cross validation, a method of combining the folds must be chosen. Here, we choose a classifier cut for each fold to obtain a fixed BT efficiency ε_{BT} . The event numbers for each fold are then summed up to obtain $N_{\text{SR/BT}}^{(l)}$.

MC-Based Estimation of δ_{sys}

Our first method to derive an estimate of δ_{sys} uses a Monte Carlo simulation of the expected SM backgrounds, in our case, a QCD dijet background. We can run our anomaly detection procedure on this Monte Carlo, including a sliding window scan of the SR position and calculate δ_{sys} . For each signal region window n , we obtain an estimate of the systematic shift

$$\delta_{\text{sys},n} = \frac{N_{\text{obs},n} - \varepsilon_B N_{\text{SR},n}}{\varepsilon_B N_{\text{SR},n}}, \quad (6.6)$$

which we average over 10 classifier runs and for CATHODE also over 10 density estimator trainings. Under the assumption that $\delta_{\text{sys},n}$ is constant in n – the validity of this assumption is checked below –, this estimate is then also averaged over all N signal region window positions n such that

$$\delta_{\text{sys}} = \frac{1}{N} \sum_{n=1}^N \delta_{\text{sys},n}. \quad (6.7)$$

In our fully simulation-based studies, we can also perform this derivation of δ_{sys} on signal-free “data” to obtain the truth value $\delta_{\text{sys}}^{\text{data}}$. This allows us to evaluate the accuracy of our estimates.

In our proof-of-concept studies, we choose the systematic error as $\sigma_{\text{sys}} = \delta_{\text{sys}}$. As we see below in our tests, the observed differences between the truth values and estimates are largely already covered by the statistical error of the analysis. For the values obtained in our studies, the choice of $\sigma_{\text{sys}} = \delta_{\text{sys}}$ is conservative. As we show below, even with such a conservative choice, our analysis remains sensitive to the benchmark signal injections we use.

Data-Driven Estimation of δ_{sys}

In addition to our simulation-based estimate of δ_{sys} , we also use a method of deriving a data-driven estimate for CATHODE. By the bump-hunt assumption, the sideband data should largely be background-only. This can be used to construct a SB-based data-driven estimate of δ_{sys} as follows: First, we sample our DE in the full sidebands to obtain a SB background template. We build a training set from these SB samples and the SB data by randomly sampling events to obtain the same data set sizes as used in the SR analysis. On this set, a classifier is trained, and the procedure described above (classifier, density estimator, and signal region window averaging) is followed to obtain $\delta_{\text{sys}}^{\text{SB}}$. If signal is present in the sideband, we expect the effect of it on δ_{sys} to be small as the signal fraction is diluted by a factor of approximately 10 due to our subsampling of events. Similar factors are expected in a realistic analysis. The impact of signal on the estimate of $\delta_{\text{sys}}^{\text{SB}}$ is further discussed in App. B.4.

While the data-driven estimate of $\delta_{\text{sys}}^{\text{SB}}$ can mitigate the effect of simulation mismodeling on the analysis, it is missing one crucial component of the background template quality. For the anomaly detector trained on the SR, the density estimator is interpolated into the SR, which it was not trained on. However, the SB-estimate is performed in the training region of the density estimator. To cover this possible underestimation of δ_{sys} , we use a combination of MC and SB estimates in our analysis. In particular, we use

$$\delta_{\text{sys}}^{\text{MC} \oplus \text{SB}} = \sqrt{(\delta_{\text{sys}}^{\text{MC}})^2 + (\delta_{\text{sys}}^{\text{SB}})^2}, \quad (6.8)$$

adding both estimates in quadrature. The validity of this combination is discussed in detail in Sec. 6.1.5.

Statistical Treatment of Direct Background Estimation

Having estimated δ_{sys} , the next step is to calculate the significance of the observed number of events. Using the Asimov estimate [176] of the significance given N_{exp} and N_{obs} , we have

$$\mathcal{S} = \left[2 \left(N_{\text{obs}} \ln \left[\frac{N_{\text{obs}}(N_{\text{exp}}^{-1} + \sigma_{\text{exp}}^2)}{1 + N_{\text{obs}}\sigma_{\text{exp}}^2} \right] - \frac{1}{\sigma_{\text{exp}}^2} \ln \left[\frac{1 + N_{\text{obs}}\sigma_{\text{exp}}^2}{1 + N_{\text{exp}}\sigma_{\text{exp}}^2} \right] \right) \right]^{1/2}, \quad (6.9)$$

where σ_{exp} is the relative error on the background estimate N_{exp} . As described above in Sec. 4.3.5, for $N_{\text{obs}} - N_{\text{exp}} \ll N_{\text{exp}}$, which we expect to be the case for larger background efficiencies ε_B , this formula reduces to the well-known Gaussian limit

$$\mathcal{S}_G = \left[\frac{(N_{\text{obs}} - N_{\text{exp}})^2}{(N_{\text{exp}} + N_{\text{exp}}^2 \sigma_{\text{exp}}^2)} \right]^{1/2}. \quad (6.10)$$

Here, we also obtain the standard Poisson error $\sqrt{N_{\text{exp}}}$. The error on the background estimate σ_{exp} is a combination of the error on the systematic shift σ_{sys} and the statistical error $\sigma_{\text{exp,stat}}$ caused by the limited statistics of the BT, resulting in fluctuations of ε_{BT} . Both are treated as relative errors and added in quadrature to obtain the error on the expectation

$$\sigma_{\text{exp}} = \sqrt{\sigma_{\text{exp,stat}}^2 + \sigma_{\text{sys}}^2}, \quad (6.11)$$

with $\sigma_{\text{exp,stat}} = 1/\sqrt{\varepsilon_{\text{BT}} N_{\text{BT}}}$. With the background templates chosen as specified in Sec. 6.1.1, $\sigma_{\text{exp,stat}}$ is approximately as large as the statistical error on the SR estimate for CWoLa Hunting, where both sets are of approximately equal size. For CATHODE, where we oversample the BT by a factor of four, $\sigma_{\text{exp,stat}}$ is significantly reduced such that it becomes almost negligible.

To better understand the systematic errors, we use the total statistical error for comparison. We can estimate this using the Gaussian limit of the Poisson error as

$$\sigma_{\text{stat}} = \sqrt{(N_{\text{exp}})^{-1} + (\varepsilon_{\text{BT}} N_{\text{BT}})^{-1}}. \quad (6.12)$$

6.1.3 IAD

We begin our studies with tests using the Idealized Anomaly Detector. Here, if we inject no signal, we should obtain results compatible with the null hypothesis for $\delta_{\text{sys}} = 0$ as the background template is ideal. We test this across all windows for both Baseline and Baseline + ΔR features. The results can be seen in the top row of Fig. 6.1. Crucially, our significance formula results in positive significances for both excesses and deficits. As such, we should obtain a Gaussian distribution with $\mu = 0$ and $\sigma = 1$ truncated to $x \geq 0$. The expectation value of this distribution is $\langle \mathcal{S} \rangle = \sigma \cdot \sqrt{2/\pi} \approx 0.80$. As we show the mean of ten runs for each of the data points in Fig. 6.1, we expect these to approximately fluctuate around this value. Indeed, we see this for all three cuts and across all windows. This shows the correct behavior of our analysis for the background-only assumption and an ideal background template. Tab. B.1 in App. B.1 also gives the δ_{sys} values on data for the IAD, which are all found to be compatible with zero within the statistical error.

Next, we move to results with our benchmark signal amount in the bottom row of Fig. 6.1. Here, we once more see results fluctuating around the null hypothesis expectation value for

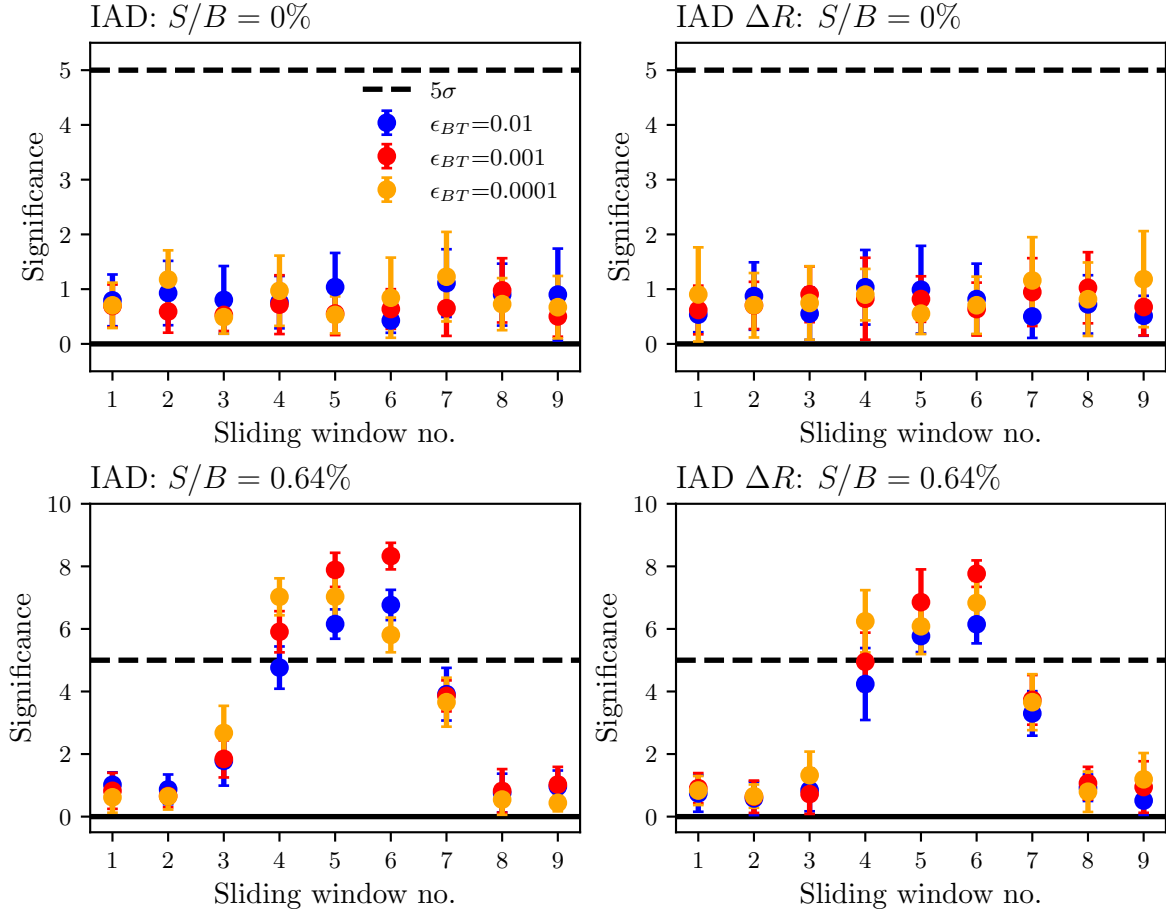


Figure 6.1: Significance $\mathcal{S}$, Eq. 6.9, for the different signal regions for the IAD without (top) and with signal injection (bottom) using the Baseline (left) and Baseline + ΔR (right) features with $\delta_{\text{sys}} = \sigma_{\text{sys}} = 0$. The error bars indicate the variance of the significance across 10 classifier runs.

the signal regions located further from the signal, i.e., windows number 1, 2, 8, and 9. In window 5, which is centered on the signal, we see a clear discovery-level significance, i.e., 5σ for all three investigated thresholds ϵ_{BT} . For the two neighboring windows, this is still the case. Between the center and edges of our tests, we observe a slow drop-off of the significance.

We can further investigate the significance by comparing it with our expectation based on SIC curves, as shown, e.g., in Sec. 5.1. For the Baseline features, the SIC, which gives an estimate of the significance improvement in the Gaussian limit, is approximately $\text{SIC} = \epsilon_S / \sqrt{\epsilon_B} \approx 11$ at $\epsilon_B = 10^{-3}$. Taking into account the initial significance of 2.2, this results in a naive expectation that the discovery in window 5 should occur at a significance of about 24, which is significantly higher than our observed significance of 8. However, there are a number of statistical inaccuracies and additional factors, which are not considered in the naive expectation: Firstly, taking into account the statistical error from the cut on the background template, we obtain $24/\sqrt{2} \approx 17$ as BT and SR are of approximately equal size. Secondly, using proper Poisson statistics for an expectation of $N_{\text{exp}} \approx 130$ at this working point instead of the Gaussian limit, we obtain a significance of 12. The difference between the observed $\mathcal{S} = 8$ and this expectation $\mathcal{S} = 12$ is then due to the k -fold cross validation and the lack of oversampling in the background template. The lack of oversampling for the observed significances leads to

a slight performance loss. The k -fold cross validation results in the training statistics being further reduced to approximately 80% of the statistics for our expectation. In addition to these reduced statistics for training, we are also vulnerable to a trade-off between training and test folds due to the limited test set statistics, as explained in the following. Fundamentally, not all signal events are equally easy to identify, as some are less and others more similar to the background. With the low number of signal events in the data, the signal in each fold is likely to not fully represent the signal distribution. As such, a particularly easy-to-classify fold may exist. If this fold is in the training data, the classifier is able to identify the signal very well. However, in this case, the evaluation set does not contain these easy-to-identify signal events, and the observed excess on this set is likely to fall below the expectation for this particular classifier. If the fold is in the evaluation data, the training is worse, and the easy-to-identify signal events in the evaluation data are not identified as well. For the naively expected significance, the test data is always made up of a large set of signal events, and this effect is not accounted for. Taking all of these factors into account, we obtain the observed significance.

6.1.4 CWoLa Hunting

Having understood the significances we observe in both the background-only case and for the signal benchmark, we move on to the realistic background template generation methods, starting with CWoLa Hunting. We begin our studies of CWoLa Hunting by estimating δ_{sys} , which we do both on our (Pythia) data for benchmarking and on our Herwig MC. The MC estimate $\delta_{\text{sys}}^{\text{MC}}$ is what we want to use in the analysis. Both estimates are shown for the Baseline and Baseline + ΔR features in the top panels of Fig. 6.2. Starting with the Baseline results (left), we see approximately $\delta_{\text{sys}}^{\text{data}} = 0.2$. On MC, a similar value is observed, though a slight decrease for very tight cuts (low ε_{BT}) can be seen. Comparing the size of the deviation between these two estimates with the statistical error (lower panel), we can see that the deviation is more than covered by the statistical error. Even if $\sigma_{\text{sys}} = 0$ were used, a systematic bias of less than 1σ would be expected. With our choice of $\sigma_{\text{sys}} = \delta_{\text{sys}}$, any deviation should therefore be well covered. With the inclusion of ΔR in the Baseline + ΔR features, we break the assumptions of CWoLa Hunting as ΔR is strongly correlated with the dijet mass. As such, the difference between the background template and the SR data events becomes larger. This is seen in the high value of δ_{sys} observed in this case (right side of Fig. 6.2). Again, the data and MC estimate of δ_{sys} are very similar and match within the statistical error (see lower panel).

Our estimates of δ_{sys} use an averaging over several windows. We test whether the underlying assumption – that the relative systematic shift is constant – is accurate for our tests by showing the window-wise shift $\delta_{\text{sys},n}$ in Fig. 6.3 for our Baseline set estimates. We see no clear trend in any of our three thresholds. This also holds for the ΔR estimates, which are shown in Fig. B.1 in App. B.2.

Having obtained estimates for δ_{sys} , we next consider the significances obtained both without and with signal in Fig. 6.4 using $\delta_{\text{sys}}^{\text{MC}}$. The precise values of $\delta_{\text{sys}}^{\text{MC}}$ at the chosen thresholds can be found in Tab. B.1 in App. B.1. Starting with the Baseline set (left column), we see no significant deviation from the null hypothesis if no signal is injected (top panel). As was observed for the IAD, significances approximately fluctuate around the expectation value of $\langle S \rangle = 0.8$ for the two tighter cuts at $\varepsilon_{\text{BT}} = 10^{-3}$ and 10^{-4} . For the looser cut at $\varepsilon_{\text{BT}} = 10^{-2}$, the observed significances are below this expectation. This is caused by the conservative choice

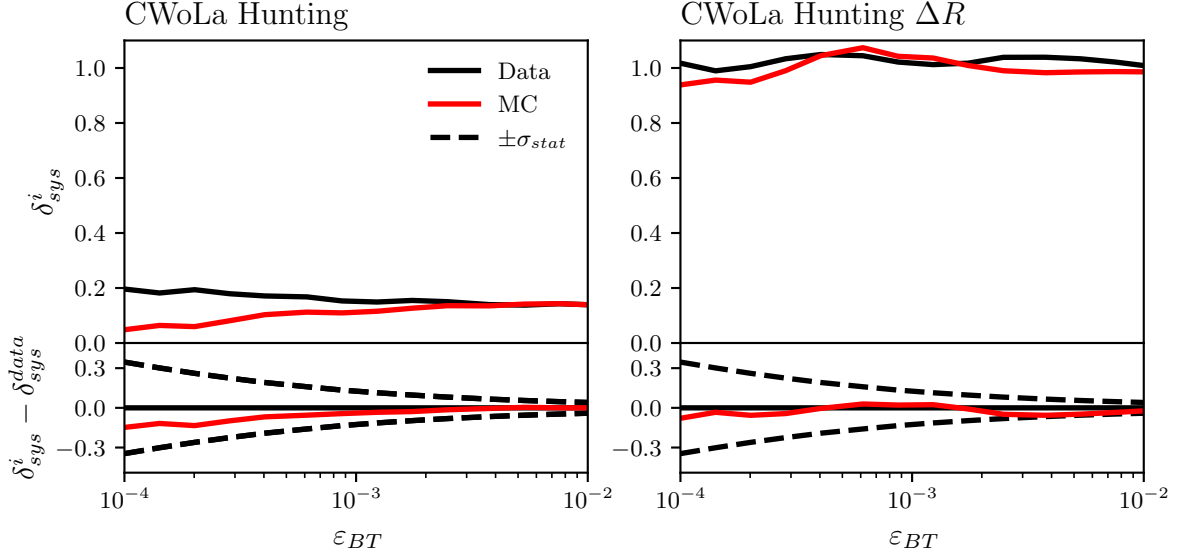


Figure 6.2: Relative systematic shift δ_{sys} for CWoLa Hunting, as a function of ε_B for the Baseline (left) and Baseline + ΔR (right) feature sets. δ_{sys} has been estimated from an analysis without signal on Pythia data (Data) and Herwig MC (MC). Results show the absolute value (top) and the deviation from the data value (bottom). In the lower panel, we also show the total statistical error σ_{stat} of the analysis as defined in Eq. 6.12 for reference. The results are across 10 classifier runs in each signal region.

of σ_{sys} we use. For this cut, the MC and Data estimates of δ_{sys} match almost perfectly (see again Fig. 6.2). Additionally, the relative statistical error is very low for $\varepsilon_{\text{BT}} = 10^{-2}$ compared to its value at $\varepsilon_{\text{BT}} = 10^{-4}$. The relative systematic error, on the other hand, decreases for smaller cuts as $\delta_{\text{sys}}^{\text{MC}}$ decreases since $\sigma_{\text{sys}} = \delta_{\text{sys}}$ is chosen. As such, the total error budget is dominated by the systematic error at $\varepsilon_{\text{BT}} = 10^{-2}$, and the conservative choice of σ_{sys} has a larger impact on the significance than for the other thresholds. This can also be seen in the studies with signal (bottom panel), for which discovery-level significances are reached in the central windows at the two tighter cuts. For $\varepsilon_{\text{BT}} = 10^{-2}$, no significances above 2σ are observed. However, a signal bump can still be observed in the central windows.

Comparing these results directly with the IAD results (see Fig. 6.1), we can see a few small differences: First of all, we see the impact of δ_{sys} suppressing significances for the looser cuts as discussed above. More interesting is the difference in behavior for windows 3 and 7. Here, we still see significances around 4σ for the IAD. The results for CWoLa Hunting show significances compatible with the null hypothesis. This difference is directly caused by the background templates used for the analysis: For the IAD, where the background template is pure background simulation, the BT contains no signal. For CWoLa Hunting, for which the background template is made up of the short sidebands, the BT may contain signal. For both window 3 and 7, the SR's edge is located at the center of the signal. Therefore, the sidebands and signal region both contain approximately half of the signal events. The signal fractions in SR and BT are approximately equal. Therefore, the classification is performed between equivalent data sets, and the signal is not detected.

Looking at the results for CWoLa Hunting with ΔR (right column of Fig. 6.4), we see that all significances are compatible with the null hypothesis. This result is crucial as it shows that the analysis method is robust even if the background template quality degrades. To ensure that this robustness is not solely due to the large systematic error that was chosen, the results

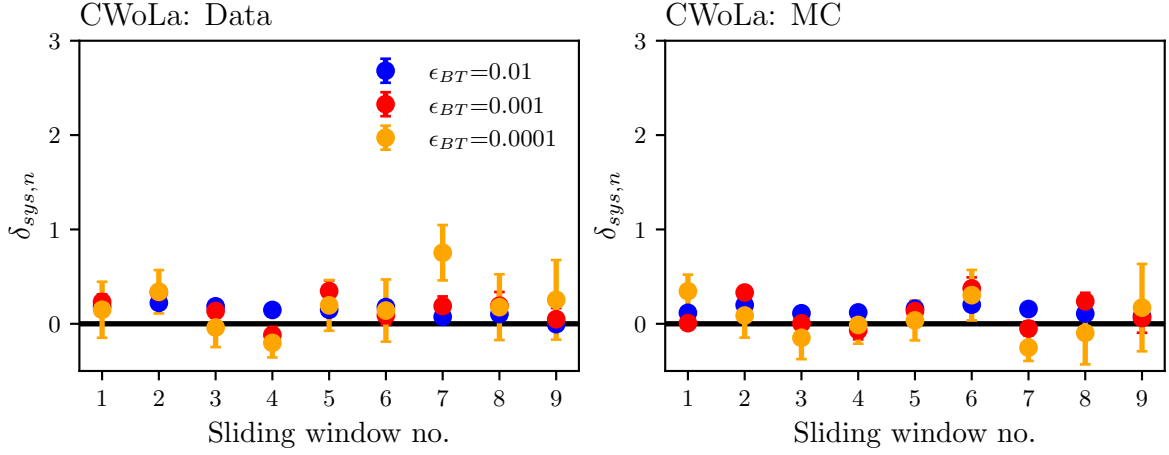


Figure 6.3: Relative systematic shift $\delta_{sys,n}$ (see Equation 6.6) per window for CWoLa Hunting determined on Pythia data (left) and Herwig MC (right) for the Baseline features. The error bars indicate the variance across 10 classifier runs.

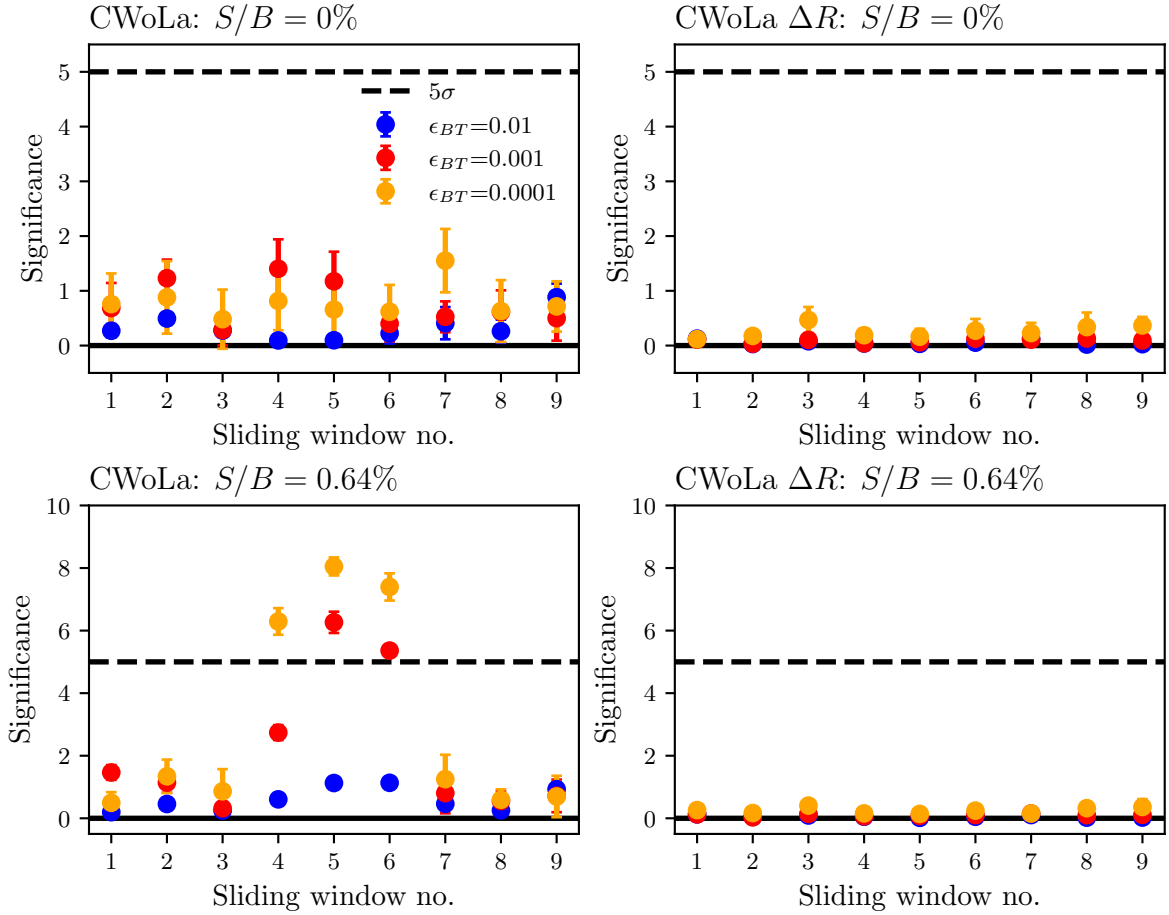


Figure 6.4: Significance $\mathcal{S}$, Eq. 6.9, for the different signal regions for CWoLa Hunting without (top) and with signal injection (bottom) using the Baseline (left) and Baseline + ΔR (right) features using δ_{sys}^{MC} . The error bars indicate the variance of the significance across 10 classifier runs.

for $\sigma_{\text{sys}} = 0$ are also considered in Fig. B.3 in App. B.3. This is the most aggressive choice possible and not advisable for an analysis, but can be used to understand the behavior of our method. Using $\sigma_{\text{sys}} = 0$, no excess above 2σ was observed for $\varepsilon_{\text{BT}} = 10^{-3}$ or 10^{-4} . For the looser cut at $\varepsilon_{\text{BT}} = 10^{-2}$, we observe a significance of approximately 4.8σ and two other windows with significances between 2 and 3σ . For this loose cut, where the relative statistical error is quite low, estimating σ_{sys} accurately is much more crucial than for the tighter cuts, for which σ_{stat} increases. The large deviation in window 1 may additionally be affected by the trigger turn-on, which may appear differently on real data. However, such trigger turn-on effects may in fact be a more general problem of weakly supervised analyses, as in the recent ATLAS analysis [45], a low m_{JJ} region also failed a validation test and was excluded from the analysis. Identifying the precise reason for these failures, both in our own studies and the ATLAS analysis, requires further study but could reveal nuances in the behavior of weak supervision. Overall, the results for $\sigma_{\text{sys}} = 0$ show that the advantages of tighter cuts go beyond an improved signal separation but are indeed also found in the reduced impact of σ_{sys} . For the studies with signal (lower panel), no excess is observed as the classifier is unable to identify the signal with the decrease of background template quality caused by the inclusion of ΔR .

6.1.5 CATHODE

Moving on to CATHODE, we begin once more with the estimation of δ_{sys} , which is shown in Fig. 6.5. Focusing first on the data estimate in black (largely covered by the yellow line), we observe the following trend: At looser cuts $\varepsilon_{\text{BT}} = 10^{-2}$, we observe a very low value of $\delta_{\text{sys}} \approx 0.1$ for both the Baseline and Baseline + ΔR features. Towards very tight cuts such as $\varepsilon_{\text{BT}} = 10^{-4}$, δ_{sys} increases significantly, rising to $\delta_{\text{sys}} \approx 0.7$ for both data sets. Comparing these values with the ones obtained for the Baseline set for CWoLa Hunting, where δ_{sys} was approximately constant across ε_{BT} , we see a higher background template quality for loose cuts and a lower one for tight cuts. In general, this behavior makes sense for DE-based background template generation. While DE can be highly accurate, correctly estimating the tails of distributions, where few events exist in the training data, becomes very difficult. Comparing these data estimates with the MC estimates (red), we see that the estimates match very well for looser cuts but increasingly diverge for tighter cuts, where $\delta_{\text{sys}}^{\text{MC}} \approx 0.2$.

The data-driven SB estimates, on the other hand, estimate the truth value well for tight cuts but underestimate it for looser cuts. As discussed in Sec. 6.1.2, the SB estimate does not include an interpolation of the DE, which is required to obtain the BT in CATHODE and is, as such, included in $\delta_{\text{sys}}^{\text{data}}$. Based on these results, we conclude that the interpolation error is dominant for looser cuts (e.g., $\varepsilon_{\text{BT}} = 10^{-2}$). This makes sense as the interpolation error may include more general shifts of distributions rather than being focused particularly on the tails. It also seems to be largely equivalent between MC and data. The difficulty of correctly estimating the tails, however, seems to be inherent to the data³ and is, as such, estimated correctly on the SBs but not on MC. We do not know, however, whether an average over multiple independent MC simulations may account for this, i.e., whether this is due to statistical fluctuations between data sets or systematic differences between the Pythia and Herwig simulations. This could be tested by using multiple data sets per simulation to estimate δ_{sys} , which is discussed for CWoLa Hunting in Sec. 6.1.8.

³ As multiple DE trainings are used for the estimates, it cannot be simply inherent to the DE.

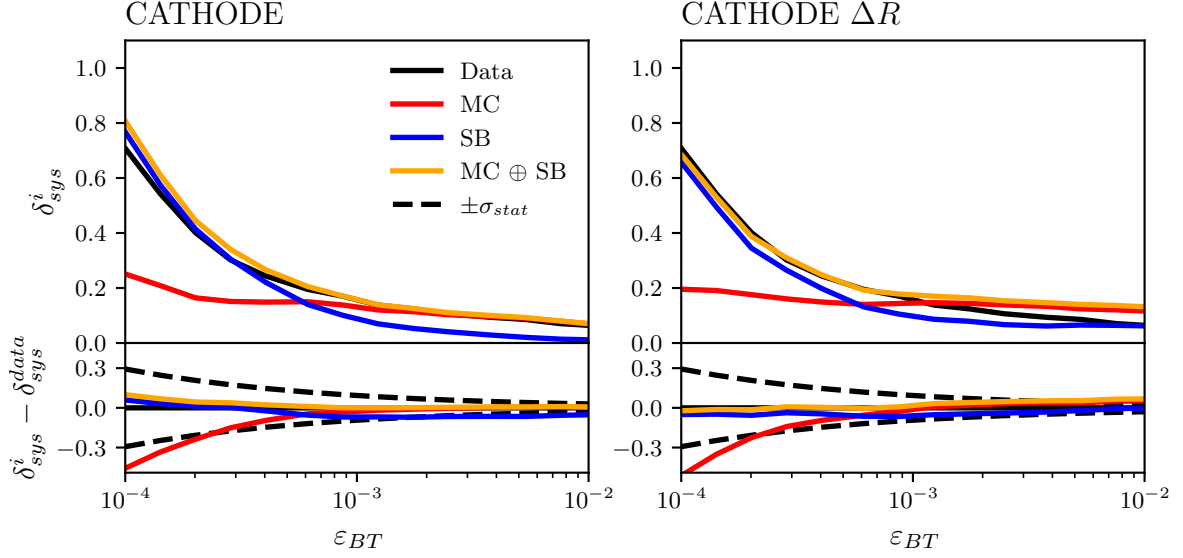


Figure 6.5: Relative systematic shift δ_{sys} for CATHODE, as a function of ε_B for the Baseline (left) and Baseline + ΔR (right) feature sets. δ_{sys} has been estimated from an analysis without signal on Pythia data (Data), Herwig MC (MC) as well as on the SB of Pythia data ($\delta_{\text{sys}}^{\text{SB}}$). We also show a quadratic addition of the Herwig MC and SB estimates ($\delta_{\text{sys}}^{\text{MC} \oplus \text{SB}}$) as described in the text. Results show absolute value (top) and deviation from the data value (bottom). In the lower panel, we also show the total statistical error σ_{stat} of the analysis as defined in Eq. 6.12 for reference. The results are based on 10 classifier runs in each signal region.

To account for both the interpolation error and the error caused by the difficulty of DE in the tails, we combine the MC and SB estimates using quadratic addition (yellow). This estimation closely matches the truth value $\delta_{\text{sys}}^{\text{data}}$ across all thresholds for both the Baseline and Baseline + ΔR set. That this is the case is partly due to luck: If the roles of the (Pythia) data and (Herwig) MC were reversed, we would strongly overestimate δ_{sys} for tight cuts such as $\varepsilon_{\text{BT}} = 10^{-4}$. It is possible that a different combination of estimates, weighting MC and SB based on the thresholds, would fix this issue. However, further studies are required to properly test different strategies, ideally using a larger variety of simulations.

As $\delta_{\text{sys}}^{\text{SB}}$ is estimated on data, it is impacted by the presence of signal in the data. We show this estimation in Fig. B.5 in App. B.4, where we see that the impact is generally minimal. Only a slight increase in $\delta_{\text{sys}}^{\text{SB}}$ and as such also in $\delta_{\text{sys}}^{\text{MC} \oplus \text{SB}}$ is observed.

As for CWoLa Hunting, we test our assumption that the relative systematic shift δ_{sys} is independent of the window number. For this purpose, we show the window-wise systematic shift $\delta_{\text{sys},n}$ on the Baseline features for each estimation method in Fig. 6.6. For the SB estimate, both the results with and without signal are included. For the looser cuts, namely $\varepsilon_{\text{BT}} = 10^{-2}$ and 10^{-3} , no trend is observed. However, for the tight cut at $\varepsilon_{\text{BT}} = 10^{-4}$, large fluctuations between different windows are observed for all of our (Pythia) data-based estimates (top left). We see a particularly large δ_{sys} in the first three windows with a maximum in window 2. This shape is also seen for the results for the Baseline + ΔR features in Fig. B.2 in App. B.2. Any conclusive statement on the cause of this would require further study. However, this may be caused by the trigger turn-on. For the SB estimates (lower panels), this shape is not visible. However, it should be noted that the data used here largely overlaps as only the relatively narrow signal regions are excluded. Overall, we conclude that the change of δ_{sys} between windows is small enough that it does not need to be considered. Additionally, the

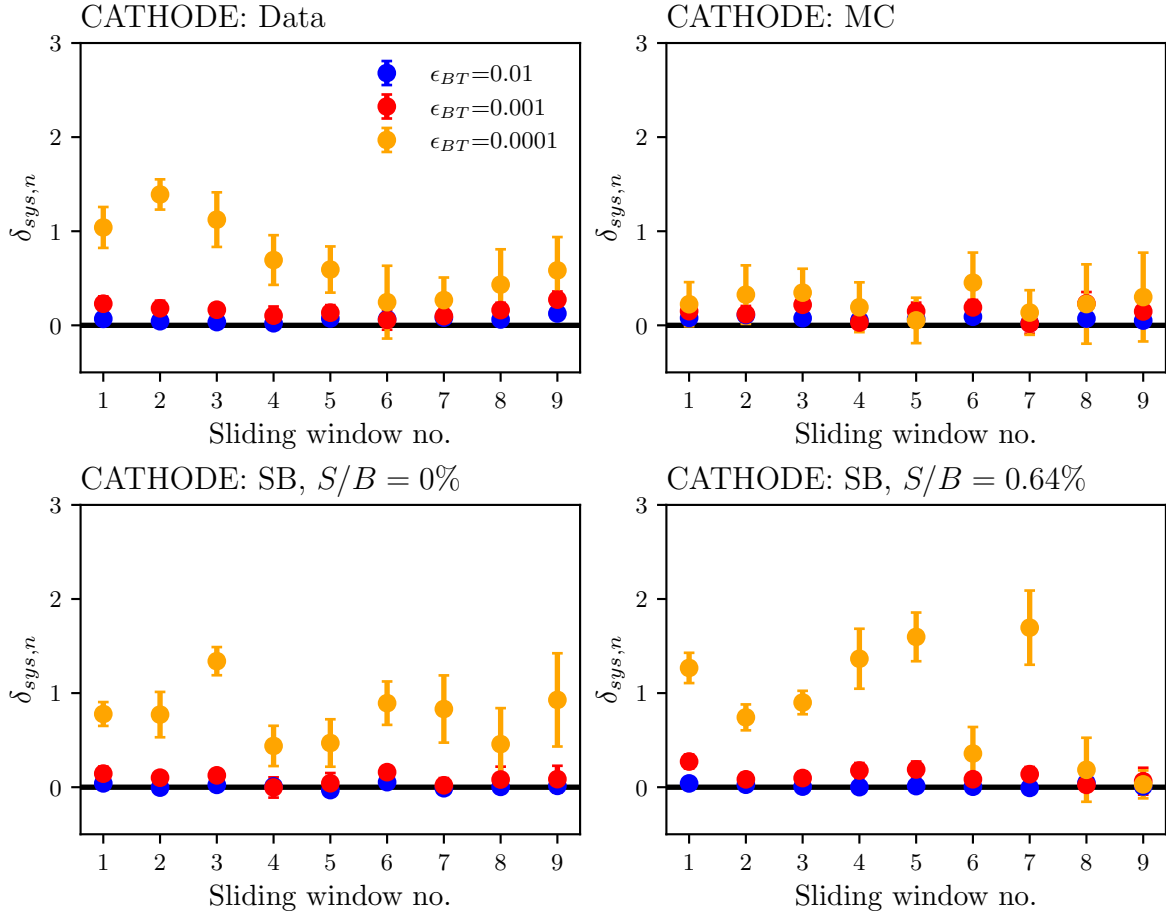


Figure 6.6: Relative systematic shift $\delta_{\text{sys},n}$ (see Equation 6.6) per window for CATHODE determined on Pythia data (top left), Herwig MC (top right), and on Pythia data using the SB (bottom) without (left) and with (right) signal for the Baseline features. The error bars indicate the variance across 10 DE and classifier runs.

small differences between $\delta_{\text{sys}}^{\text{data}}$ for different window numbers that are visible are not reflected in our estimates. This is also observed for the Baseline + ΔR features in Fig. B.2 in App. B.2. The values of all estimates of δ_{sys} at the three thresholds used in the full analysis below can be found in Tab. B.1 in App. B.1.

We now turn to the discussion of the significances we observe for CATHODE when using $\delta_{\text{sys}}^{\text{MC} \oplus \text{SB}}$ in Fig. 6.7. For the background-only case (top panels), results are compatible with the null hypothesis across all thresholds and for both feature sets. Significances lie below the background-only expectation of $\langle S \rangle = 0.8$ due to the conservative choice of the systematic error σ_{sys} . For the studies with signal (bottom panels), we see a clear bump in the central windows across all thresholds. In contrast to CWoLa Hunting, this bump is still visible in windows 3 and 7. A perfect density estimator would learn the visible half of the signal bump and complete it into the SR. However, these signal events are strongly diluted by the large event numbers in the sidebands and as such are difficult to learn. Discovery-level significances are only observed for the medium cut at $\epsilon_{\text{BT}} = 10^{-3}$. For the looser cut at $\epsilon_{\text{BT}} = 10^{-2}$, the conservative choice of σ_{sys} has a larger impact than for 10^{-3} as the relative statistical error is smaller. For the tighter cut at $\epsilon_{\text{BT}} = 10^{-4}$, the strong increase of δ_{sys} suppresses significances below the 5σ threshold. Especially here, the analysis sensitivity would greatly benefit from

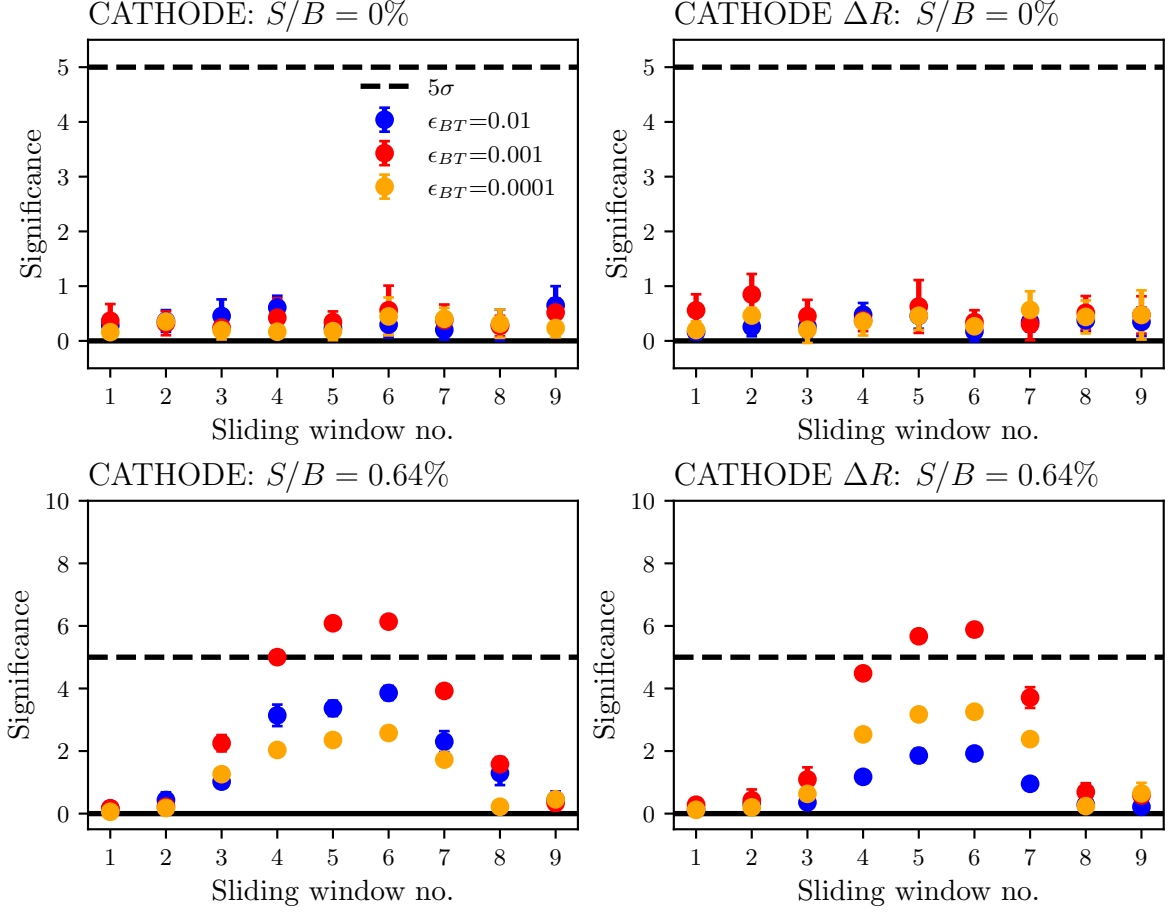


Figure 6.7: Significance $\mathcal{S}$, Eq. 6.9, for the different signal regions for CATHODE without (top) and with signal injection (bottom) using the Baseline (left) and Baseline + ΔR (right) features with $\delta_{\text{sys}}^{\text{MC} \oplus \text{SB}}$. The error bars indicate the variance of the significance across 10 DE and classifier runs.

a less conservative choice of σ_{sys} . We show this in Fig. B.4 in App. B.3, where we study the very liberal and inadvisable choice of $\sigma_{\text{sys}} = 0$. This leads to larger deviations in the background-only case, though those still lie below 3σ , but also results in a large significance gain with all thresholds reaching discovery-level significances and $\epsilon_{\text{BT}} = 10^{-3}$ significances of up to 18σ .

Additionally, tests using $\delta_{\text{sys}}^{\text{MC}}$ were performed to understand the impact of an estimate of δ_{sys} , which is misaligned with the truth value. In particular, such an imperfect estimation because of simulation mismodeling is a realistic scenario. As discussed above, we indeed underestimate δ_{sys} for tight cuts by using the MC estimate. In our tests, which are shown in detail in Fig. B.6 in App. B.5, we observe larger fluctuations in the background-only case. However, no significances above 3σ are found. Additionally, we obtain larger significances in our studies with signal, in particular for $\epsilon_{\text{BT}} = 10^{-4}$, for which we strongly underestimate δ_{sys} .

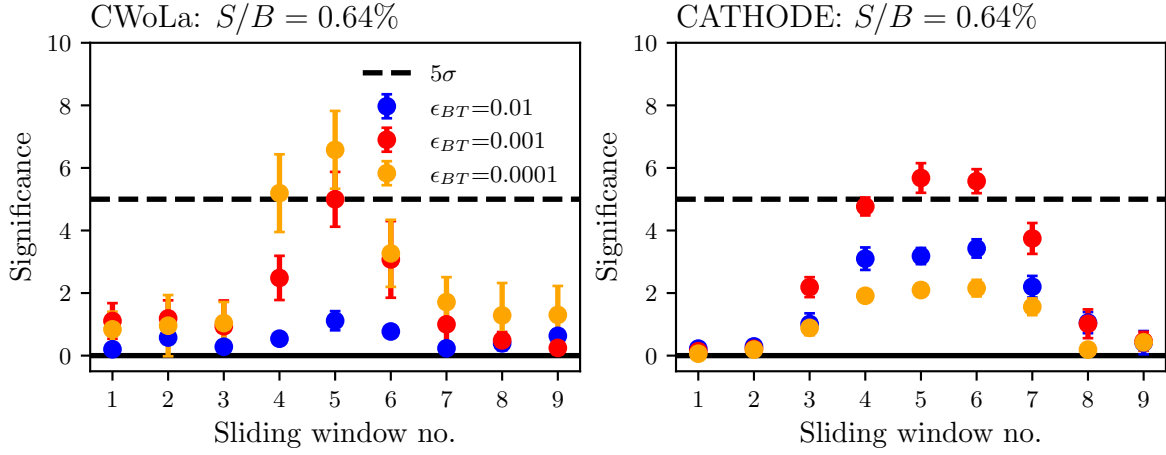


Figure 6.8: Significance $\mathcal{S}$, Eq. 6.9, for the different signal regions for CWoLa Hunting (left) and CATHODE (right) using the Baseline features when randomly selecting injected signal events. The error bars indicate the variance of the significance across 10 random signal injections, classifier runs, and DE runs for CATHODE. Results not included in Ref. [2].

6.1.6 The Impact of Signal Randomization

So far, we have studied a fixed set of signal events. In particular for CATHODE, where the SB estimate of the systematic shift may include signal events, this allows us to average over multiple runs as would be possible in a realistic analysis. However, we now test a randomized selection of signal events to obtain a more accurate picture of the expected sensitivity and fluctuations. We use the same δ_{sys} estimates as before, namely $\delta_{\text{sys}}^{\text{MC}}$ for CWoLa Hunting and $\delta_{\text{sys}}^{\text{MC} \oplus \text{SB}}$ for CATHODE. We do not perform the estimation of $\delta_{\text{sys}}^{\text{SB}}$ for the randomized set of signal events for CATHODE. Results for the Baseline features for CWoLa Hunting and CATHODE are shown in Fig. 6.8. Compared with the fixed signal injections (bottom left in Fig. 6.4 for CWoLa Hunting and Fig. 6.7 for CATHODE), we see slightly lower significances in the central windows for both CWoLa Hunting and CATHODE. However, the signal is still detected at discovery-level significances in windows 5 and 6 for the best threshold in each case. As expected, we also see increased fluctuations compared with the results in Figs. 6.4 and 6.7 when the signal events are randomized. For the tighter cuts ($\epsilon_{\text{BT}} = 10^{-3}$ and 10^{-4}), these fluctuations are much larger for CWoLa Hunting than for CATHODE. For CATHODE, with its oversampling, the total error budget is much more dominated by the systematic error, while for CWoLa Hunting, the statistical error is dominant. As such, the observed trends in the variance fit this expectation.

6.1.7 CWoLa Hunting for Extended Feature Sets

The two feature sets in our proof-of-concept studies above are also relatively limited. As such, we investigate the extended feature sets introduced in Sec. 5.1 with CWoLa Hunting. These feature sets – Extended sets 1, 2 and 3 – use additional subjettinesses and subjettiness ratios to increase the sensitivity to wider varieties of signal models. The feature sets contain 10, 12, and 56 features, respectively, with the smaller feature sets, including the Baseline set, always containing a subset of the information of the larger sets (details in Tab. 5.1). In Sec. 5.1.5, Extended sets 2 and 3 were found to improve the signal classification for CWoLa Hunting. For

ε_B	σ_{stat}	Baseline		Extended set 1		Extended set 2		Extended set 3	
		$\delta_{\text{sys}}^{\text{data}}$	$\delta_{\text{sys}}^{\text{MC}}$	$\delta_{\text{sys}}^{\text{data}}$	$\delta_{\text{sys}}^{\text{MC}}$	$\delta_{\text{sys}}^{\text{data}}$	$\delta_{\text{sys}}^{\text{MC}}$	$\delta_{\text{sys}}^{\text{data}}$	$\delta_{\text{sys}}^{\text{MC}}$
0.01	0.04	0.14	0.14	0.12	0.12	0.14	0.13	0.22	0.19
0.001	0.13	0.16	0.11	0.17	0.10	0.14	0.12	0.14	0.19
0.0001	0.35	0.20	0.05	0.18	0.14	0.14	0.23	0.02	0.27

Table 6.1: Relative systematic shift δ_{sys} estimated from an analysis without signal on Pythia data ($\delta_{\text{sys}}^{\text{data}}$) and Herwig MC ($\delta_{\text{sys}}^{\text{MC}}$). The results are based on 10 classifier runs in each signal region. For reference, the statistical error σ_{stat} is also given. Only results for the Baseline features are included in Ref. [2].

Extended set 1, the anomaly detection performance of CWoLa Hunting was found to slightly decrease from that obtained using the Baseline features. For CATHODE, the significantly larger feature sets make obtaining a BT of a high enough quality with density estimators difficult, which is why we do not consider CATHODE here.

To apply direct background estimation using these feature sets, we must first estimate δ_{sys} . We do this once more for both our data and MC and compare the obtained values at all three of our tested thresholds in Tab. 6.1. The total statistical error, as defined in Eq. 6.12, and the values for the baseline features are also given for comparison. Across all feature sets and thresholds, we see no deviations between data and MC estimates larger than the statistical error. Additionally, we see that even as the feature sets become larger, δ_{sys} does not increase dramatically.

Next, we consider the significances obtained for the extended sets in Fig. 6.9. Starting with the background-only tests (left column), we see no deviation larger than 3σ , with all but one below 2σ . Indeed, most significances again fluctuate around the null hypothesis expectation $\langle \mathcal{S} \rangle = 0.8$. When signal is injected, we observe it at discovery-level significance in at least one threshold and one window across all feature sets. Overall, we see a slight decrease of performance compared with the Baseline features in Extended set 1, which is compatible with results in Sec. 5.1.5. Both Extended set 2 and 3 are able to achieve a similar or slightly improved performance as the Baseline features. These results show that the use of larger feature sets with direct background estimation is possible without producing false discoveries or losing sensitivity due to systematic effects. Particularly for extended set 3, which through its many features can provide sensitivity to a wide variety of signal models at once [5], this can allow for more model-agnostic weakly supervised analyses in the future.

6.1.8 Studies of δ_{sys} Using Independent Simulations

In order for analyses using direct background estimation to improve sensitivity, a key area for improvement is a well-founded determination of the systematic error σ_{sys} on δ_{sys} . In our proof-of-concept studies above, we have used a simple, conservative choice with $\sigma_{\text{sys}} = \delta_{\text{sys}}$. For this reason, we now investigate the determination of δ_{sys} for CWoLa Hunting in more detail. By simulating 10 million (Pythia) data and (Herwig) MC events and bootstrap sampling data sets of 1 million events, the general distributions of $\delta_{\text{sys}}^{\text{data}}$ and $\delta_{\text{sys}}^{\text{MC}}$ can be determined. In Fig. 6.10, we see results at our three thresholds for both the IAD and CWoLa Hunting as mean and standard deviation of 100 such data sets. For the IAD (left panels), we obtain a mean compatible with $\delta_{\text{sys}} = 0$ across all thresholds and windows. Statistical fluctuations are

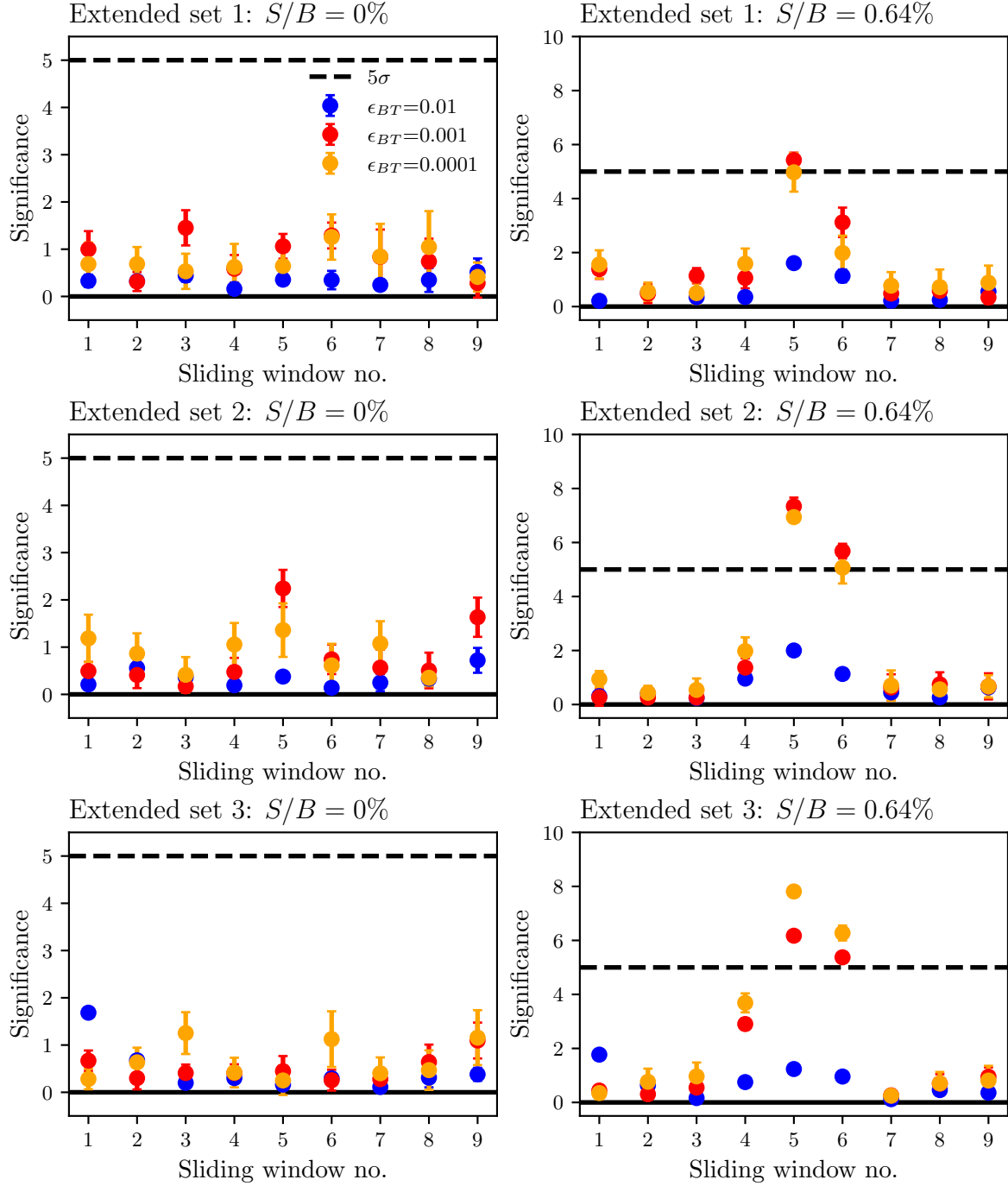


Figure 6.9: Significance $\mathcal{S}$, Eq. 6.9, for the different signal regions for CWoLa Hunting without (left) and with signal injection (right) using Extended sets 1 (top), 2 (middle), and 3 (bottom) with $\delta_{\text{sys}}^{\text{MC}}$. The error bars indicate the variance of the significance across 10 classifier runs. Results are not contained in Ref. [2].

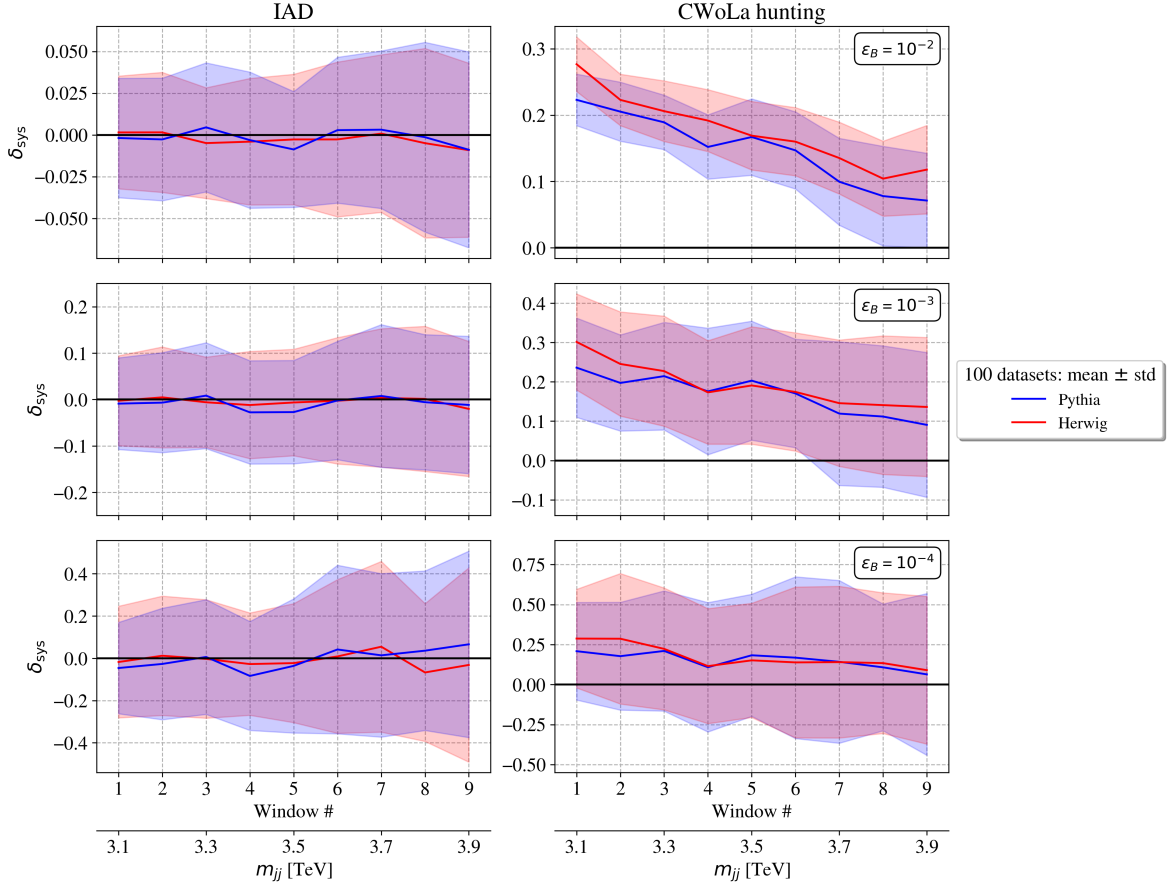


Figure 6.10: Relative systematic shift $\delta_{\text{sys},n}$ (see Equation 6.6) per window for IAD (left) and CWoLa Hunting (right) determined on Pythia data and Herwig MC for the Baseline features. The error bars indicate the variance across 100 classifier runs on different data sets. Results are not contained in Ref. [2] and are taken with permission from Ref. [187].

observed, which increase as ε_{BT} is decreased. For CWoLa Hunting (right panels), we see that at all three thresholds, both the mean and standard deviation of the $\delta_{\text{sys}}^{\text{data}}$ and $\delta_{\text{sys}}^{\text{MC}}$ match well overall. In contrast to the studies on our single (Pythia) data and Herwig MC sets (see Figs. 6.3 and 6.6 and App. B.2), we see a window-dependent structure in δ_{sys} , particularly for looser cuts. This structure is made visible by the additional effort resulting in a more precise determination δ_{sys} . This window-dependent structure is obtained for both Pythia and Herwig samples.

Considering the close match, it is possible to obtain a highly accurate estimation of δ_{sys} and σ_{sys} by using multiple MC samples. The mean and standard deviation of the per window systematic shift $\delta_{\text{sys},n}$ can then be used as $\delta_{\text{sys},n}$ and its error $\sigma_{\text{sys},n}$. In further studies conducted in Ref. [187], this was found to provide well-calibrated significances. While computationally expensive, such a robust and well-calibrated choice of the systematic shift and its error may be able to provide an enhanced sensitivity of weakly supervised analyses.

6.1.9 Conclusion and Outlook

In this section, we have shown how a weakly supervised analysis can be conducted using a fully cut and count-based significance extraction called direct background estimation. By using the background template to obtain a background expectation, we can avoid fitting the background spectrum and, as such, the issue of background sculpting. Without the fit, we are also able to use larger background rejection rates, for which weakly supervised methods typically provide the largest discovery potential.

To use direct background estimation, it is necessary to estimate the systematic difference between cuts on the classifier score for the SR background and the background template. We have formalized this in the relative systematic shift δ_{sys} between the two sets with $\varepsilon_B = \varepsilon_{\text{BT}}(1 + \delta_{\text{sys}})$. We have proposed two methods of estimating this shift, namely, based on MC simulation, as is possible in general, and in a data-driven manner on the sidebands for CATHODE. The data-driven estimation is applicable to many other methods beyond CATHODE, including but not limited to SALAD [34], FETA [39], and CURTAINS [37].

We have shown that direct background estimation and our estimation of δ_{sys} work in proof-of-concept studies using the IAD, CWoLa Hunting, and CATHODE. In the background-only case, no false discoveries were observed. This includes tests with CWoLa Hunting using a feature set including the dijet angular distance ΔR , for which the background template quality is poor. In this case, the systematic shift δ_{sys} increased on both data and Monte Carlo. Studies with injected signal were able to detect a signal with an initial significance of 2.2σ beyond 5σ for both CWoLa Hunting and CATHODE for our Baseline features. This was the case both for a fixed set of signal events and randomized signal injections.

Beyond these studies, we were able to show that both the sensitivity and the absence of false discoveries are retained for larger feature sets with up to 56 features. These larger feature sets could enable weak supervision to be much more model-agnostic (see also Sec. 5.1) by providing sensitivity to a wider variety of different signal models.

Particularly an improved estimation of the systematic shift δ_{sys} and its error σ_{sys} could enhance the sensitivity of direct background estimation. One particularly robust method for this, as shown above, is the estimation on multiple MC sets using the mean and standard deviation. However, this method is computationally very expensive. Additionally, if the mismatch between real data and MC is larger than the one between our two different MC simulations, an additional component of σ_{sys} may be required to account for this. For a wider invariant mass spectrum, it may also be possible to estimate δ_{sys} directly on data: Under the bump hunt assumption, signal regions far away from a signal do not contain any signal. For these, a background-only analysis is essentially conducted. In our studies, with the limited width of the available spectrum, this was only the case for few windows. However, in a wider invariant mass spectrum, a mean over windows away from the current signal region can be used to estimate δ_{sys} . This method does not take into account any window-dependent changes in δ_{sys} . However, it provides a computationally inexpensive method of estimating the systematic shift, which is not sensitive to any data-MC disagreement.

Beyond fixed cuts, it could also be interesting to consider a reweighting-based strategy instead of a cut on the classifier to obtain significances as was done in Ref. [47]. There, a fit was used, and a performance gain compared to a fixed cut was observed. The effect of this on direct background estimation and particularly the determination of δ_{sys} provides an interesting avenue for further study.

6.2 THE LOOK-EVERYWHERE EFFECT

This section is based on Ref. [4]. The code for this project can be found in Ref. [189]. Any results not contained in Ref. [4] are marked as such in the captions.

In any new physics search where multiple possible signal parameterizations are tested, the multiple testing must be accounted for when reporting the results of the search. Each separate test has a *local* p-value, which describes how likely it is to obtain a deviation from the null-hypothesis at least as large as the one observed in the data. When combining multiple tests, we are typically interested in the largest deviation, i.e., the lowest p-value. However, performing multiple tests means that any low p-value becomes more likely as we take the minimum of multiple tests. Reporting the lowest local p-value as the *global* p-value of the analysis leads to a miscalibration of the analysis and, as such, an overestimation of significances. This is called the *look-elsewhere effect*. One can calibrate the global p-value by applying a trials factor to account for the multiple testing. For follow-up analyses using independent data, which only consider the particular test with the lowest deviation, we do not incur a look-elsewhere effect. As such no trials factor is required and the analysis is more sensitive. When performing anomaly detection using machine learning, the anomaly detector also considers many possible signal configurations and picks the optimal one. For this, it does not look at only specific signal positions but the full phase space of the given features. In this section, we explore to what extent anomaly detectors are affected by a *look-everywhere effect* because of this implicit multiple testing.

Above, we consider new physics searches using statistics language, but we can also translate the problem at hand into machine learning language: When a local p-value is reported as global, this means that the analysis was evaluated on its training set. In machine learning, evaluating any network on its training data is typically avoided to keep overfitting from biasing the results. Instead, best practice always uses an independent test set. This is equivalent to a follow-up analysis on independent data as described above. In addition to these two options for evaluation, *k*-fold cross validation is often used [44–46] (see also Sec. 6.1). Here, data is split into *k* folds: training is conducted on *k* − 1, and 1 is used for evaluation, iterating over all folds and combining results. Each individual network is evaluated on independent test data. However, the methods used to combine the results across folds assume full statistical independence, which, as we discuss below, is not the case.

We investigate the nature of look-everywhere effects and their effect on sensitivity on both 2D Gaussian toy data and more realistic data based on the LHC Olympics R&D data set. We study the statistical behavior of our anomaly detector for an evaluation on training data, independent test data, and using *k*-folding. To understand the relation to the look-elsewhere effect, we perform our tests with both a binned anomaly detection scenario and ML anomaly detectors, namely a NN- and a BDT-based classifier. The binned anomaly detector (as introduced in Sec. 6.2.1) not only allows us to more intuitively understand the statistical effects at play but also represents many early anomaly detection analyses (see e.g. [190, 191] and references therein). Our investigations are structured as follows: First, in Sec. 6.2.2, we aim to understand the look-elsewhere and look-everywhere effects and their relation by comparing our binned anomaly detector with a NN-based anomaly detector. Next, in Sec. 6.2.3, we draw a parallel between the look-everywhere effect and overfitting by strategically applying early stopping, thus limiting overfitting and investigating the effect on p-value (mis-)calibration. Here, we study both the NN- and the BDT-based anomaly detector. Then, in Sec. 6.2.4, we investigate what analysis strategy provides the best signal sensitivity. For this, we perform

tests with a signal based on an LHCO-like data set using both the NN- and the BDT-based anomaly detector. Lastly, we conclude in Sec. 6.2.5. However, before any of our tests can be discussed, we must first understand the underlying setup and statistical procedures in Sec. 6.2.1.

6.2.1 Setup

Statistical Procedures

We build our statistical procedure on the weakly supervised comparison (see also Sec. 4.2) of a background-only background template and a mixed signal region with distributions

$$p_{\text{BT}}(x) = p_B(x) \quad p_{\text{SR}}(x) = f_S \cdot p_S(x) + (1 - f_S) \cdot p_B(x), \quad (6.13)$$

where $p_{B/S}$ are the background and signal distributions, respectively. Throughout our studies, we assume an idealized scenario, where the background template distribution $p_{\text{BT}}(x)$ perfectly matches the SR background distribution $p_B(x)$. Data sets then only differ by statistical fluctuations. The difference between the two sets of events is quantified via the likelihood ratio

$$R(x) = \frac{p_{\text{SR}}(x)}{p_{\text{BT}}(x)}, \quad (6.14)$$

which we estimate either through a binned or a classifier-based procedure. By selecting events from both sets based on $R(x)$ and comparing the number of selected events – as was also done in Sec. 6.1 – we can derive p-values. We only look for excesses of events and, as such, calculate one-sided p-values. The exact statistical treatment differs depending on the procedure and is described below. In each case, we learn $R(x)$ on a training set and then evaluate p-values on an evaluation set. We use different setups for the training and evaluation set, which are described below.

BINNED PROCEDURE We perform our binned procedure only for the 2D Gaussian toy data. We bin the SR training data and approximate $p_{\text{SR}}(x)$ based on the number of SR events N_{SR}^i in each bin i . We use the known underlying distribution of the BT $p_{\text{BT}}(x)$ to obtain the expected fraction of events per bin p_{exp}^i . We calculate the binned likelihood ratio

$$R^i = \frac{N_{\text{SR}}^i}{N_{\text{SR}} \cdot p_{\text{exp}}^i}, \quad (6.15)$$

where N_{SR} is the total number of events in the SR. We then select the bin with the maximum R^i . On the evaluation set, we then calculate p-values. For a background-only scenario, N_{SR}^i follows a binomial distribution⁴ with

$$N_{\text{SR}}^i \sim B(N_{\text{SR}}, p_{\text{exp}}^i). \quad (6.16)$$

As such, we can use the cumulative distribution function (CDF) of the binomial distribution

$$F(N_{\text{SR}}^i; N_{\text{SR}}, p_{\text{exp}}^i) = \sum_{X=0}^{N_{\text{SR}}^i} B(X|N_{\text{SR}}, p_{\text{exp}}^i), \quad (6.17)$$

⁴ We obtain binomial statistics as we sample a fixed number of events for the full data set. For a standard counting experiment, the total rate would fluctuate, and a Poisson distribution would be obtained. For small selection efficiencies, which are not necessarily given by our binned procedure, the binomial distribution is well-represented by a Poisson distribution.

to calculate the one-sided p-value

$$p = \sum_{X=N_{\text{SR}}^i}^{N_{\text{SR}}} B(X|N_{\text{SR}}, p_{\text{exp}}^i) = 1 - F(N_{\text{SR}}^i - 1; N_{\text{SR}}, p_{\text{exp}}^i), \quad (6.18)$$

which is the probability of obtaining an excess at least as large as N_{SR}^i .

CLASSIFIER-BASED PROCEDURE For our classifier-based procedure, a machine learning classifier is trained on the training set to distinguish BT and SR. For an optimally trained classifier, this gives us $R(x)$ [92] (see also Sec. 3.2). P-values are obtained on the evaluation set by applying a cut on the anomaly score to both the BT and the SR. We select this cut to retain a fixed fraction ε_{BT} of events on the BT. With the cut, we retain N'_{BT} events in the background template and N'_{SR} events in the signal region. In the background-only case, both of these counts follow a Poisson distribution with the same expectation value λ , i.e., that

$$N'_{\text{SR}}, N'_{\text{BT}} \sim \text{Poisson}(\lambda). \quad (6.19)$$

In Sec. 6.1, we used the Asimov estimate [176] (see also Sec. 4.3.5) to calculate the significance of the obtained results. We directly used $\lambda \approx N'_{\text{BT}}$ in the IAD case and included the Gaussian limit of the Poisson error on N'_{BT} , namely $\sqrt{N'_{\text{BT}}}$, in the Asimov estimate. While approximately correct, this procedure does contain some statistical approximations, for example, treating the uncertainty in the estimation of λ in the Gaussian limit. These approximations are necessary as they allow us to also include a systematic error on the background estimation. Here, we exclusively follow an idealized AD approach, which eliminates the systematic biases considered in Sec. 6.1. Therefore, we can follow a more rigorous approach to the statistical treatment of our cut and count analysis. As such, we ensure that no statistical approximation leads to additional miscalibrations of p-values beyond the look-everywhere effects we wish to investigate.

The calculation of a correct p-value can be achieved with Bayesian statistics (see Ref. [192] for a more detailed explanation) to obtain a distribution of N'_{SR} given N'_{BT} . We measure N'_{BT} events and ask what the probability of any value of λ is given this measurement. According to Bayesian statistics, a sensible distribution to describe this probability is the Poisson distribution's conjugate prior⁵, namely a Gamma distribution with

$$\lambda \sim \Gamma(\alpha = N'_{\text{BT}}, \beta = 1) \quad (6.20)$$

where the rate β is 1 by definition, as our BT and SR have the same size. Using this distribution of λ , we can then integrate over all possible choices of λ to obtain

$$\text{Pr}(N'_{\text{SR}}|N'_{\text{BT}}) = \int_0^\infty \text{Poisson}(N'_{\text{SR}}|\lambda) \Gamma(\lambda|N'_{\text{BT}}, 1) d\lambda. \quad (6.21)$$

The result of this integral is a known distribution, namely the negative binomial distribution with $\text{Pr}(N'_{\text{SR}}|N'_{\text{BT}}) = \text{NB}(k = N'_{\text{SR}}, r = N'_{\text{BT}}, p = \frac{1}{2})$. The negative binomial distribution models the number of failures k in a sequence of Bernoulli trials before a fixed number of successes r occurs, when p is the success probability of each trial.

As for the binned case above, we can use the CDF of $\text{Pr}(N'_{\text{SR}}|N'_{\text{BT}})$ to obtain the p-value [192].

⁵ The conjugate prior is a distribution, for which $p(x|y) = \int p(x|\vartheta)p(\vartheta|y)d\vartheta$ is a known distribution, see also Ref. [192].

Evaluation strategies

We use three different strategies to obtain a training and evaluation set for the statistical procedures described above. For each, we describe how the training and evaluation sets are obtained and used, as well as the expected statistical behavior, which we consider in detail in our studies below. Firstly, we evaluate on the training set. This means that we use the full data for both training and evaluation. While this provides the best statistics for both, it also incurs the largest look-elsewhere and look-everywhere effects. Secondly, we evaluate on an independent test set. In this case, we use a 50-50 split of the data to obtain fully independent training and evaluation sets. This eliminates look-elsewhere and look-everywhere effects but significantly reduces training and evaluation statistics. Lastly, we evaluate using k -fold cross validation as was done in Sec. 6.1 with $k = 5$ by using $k - 1$ folds as the training and 1 fold as the evaluation set. We iterate such that all k folds are used once as the evaluation set. The results for all folds are then combined. There are a variety of ways of combining results from multiple tests by assuming statistical independence. We use the strategy of summing over all counts and calculating a common p-value, i.e., $N'_{\text{SR}} = \sum_{i=1}^k N'_{\text{SR},i}$. Compared with an evaluation on the training set, using k -folding results in only a slight reduction of training statistics (80% for $k = 5$), no reduction in evaluation statistics, and aims to reduce the severity of look-elsewhere and look-everywhere effects.

Data Sets

TOY DATA We first perform calibration testing of p-values on 2D Gaussian toy data with mean $\mu = 0$ and standard deviation $\sigma = 1$. We sample 25 000 events in the SR to serve as data. When evaluating on the training set, we directly apply our statistical procedures to this data. When evaluating on the test set, we reuse these trainings and evaluate on independently sampled sets of 25 000 events. This allows us to reduce computation time slightly and does not impact results, as we expect and obtain calibrated p-values when training and evaluating on fully independent sets. For the k -fold cross validation, the 25 000 events are split evenly into $k = 5$ folds. For the binned procedure, we assume infinite statistics for the BT. We use a fixed binning with 5 bins per dimension between -2 and 2 , giving us a total of 25 bins. For the classifier-based procedure, all BT sets are treated as described for the SR data above.

LHCO DATA SET We use a data set based on the LHC Olympics R&D data set (see also Sec. 4.3.1) to both test our statistical procedures on a more realistic data set and to evaluate our signal sensitivity for the different classifiers and evaluation procedures. The R&D set contains 1 000 000 QCD dijet background and 100 000 signal events for a W' resonance with mass $m_{W'} = 3.5$ TeV. The W' decays into $X(\rightarrow qq)$ and $Y(\rightarrow qq)$ with masses $m_X = 100$ GeV and $m_Y = 500$ GeV, respectively. We use the dijet mass of the two highest p_T jets as the resonant feature and center our signal region with a width of 0.4 TeV on $m_{W'}$. For the classification, we use the Baseline features as described in Sec. 4.3.2, namely the mass of the lighter jet m_{J1} , the mass differences of the two jets $\Delta m_J = m_{J2} - m_{J11}$ and both jets' 21-subjettiness ratios $\tau_{21}^{J1/J2}$.

To calibrate p-values, we need to obtain statistically independent, background-only data sets. To approximately achieve this, we use a bootstrapping method based on a set of 10 000 000 background events produced for Ref. [187], of which we retain only the approximately 1 200 000 SR events. For each data set, we randomly sample 120 000 events each for the BT and SR,

which is approximately equivalent to the statistics of the standard R&D set SR.

When testing with signal, we again use the background events from Ref. [187] and combine these with signal events taken from Ref. [18]. For the BT, we again sample 120 000 background events. For the SR set, we sample $N_{\text{sig,SR}} = f_{\text{SR}} \cdot N_{\text{sig}}$ signal events, where f_{SR} is the fraction of signal events in the SR (approximately 75.3%), and N_{sig} is the number of SR events one would inject into the whole data set. This definition is made to remain as compatible as possible with studies using the standard R&D set. We sample background events from our large set such that the full SR set contains 120 000 events. For each signal injection, we train and evaluate ten times with random selections of signal events and data shuffled before splitting into training and test sets or folds. For these trainings, we not only calculate p-values but also evaluate on a labeled test set to obtain the maximum SIC with a statistics cutoff equivalent to a maximum relative error on the background of 20%. This test set contains 20 000 signal events from the R&D data set and 340 000 events of the additional SR background events from Ref. [170]. When using this test set for the k -fold cross validation, we must combine results of multiple classifiers. We choose to base the SIC calculation on the concatenation of the test set classifier scores on the test set for each of the k folds, meaning that each event in the test set is used k times.

Classifiers

NEURAL NETWORK CLASSIFIER For our NN classifier, we use a simple MLP with three hidden layers with 64 nodes each and ReLU activation. We train for a maximum of 50 epochs using the Adam optimizer with a learning rate of 10^{-3} using a batch size of 128. We perform trainings both with and without early stopping. If used, the early stopping is applied to the validation loss with a patience of 10 epochs, and the network restores the best weights after stopping. The network is implemented using TensorFlow [179].

BOOSTED DECISION TREE For our BDT classifier, we use a setup based on Ref. [1] as described in Sec. 5.1. This means that we use the `HistGradientBoostingClassifier` from `scikit-learn` [117] with largely default hyperparameters. We ensemble ten BDTs with randomized 50-50 training and validation splits. If using early stopping, we stop based on the validation loss with a patience of ten iterations, reverting to the best model.

6.2.2 Comparing Analysis Strategies

We start our experiments with the Gaussian toy data by comparing the behavior of p-values in the background-only case of the binned procedure with that of the classifier-based procedure for the NN without early stopping. We investigate the behavior of p-values by performing N_{tests} tests with independently generated data. For each test, we calculate a p-value as explained in Sec. 6.2.1. These calculated p-values are then sorted in ascending order. The empirical p-value of the i^{th} smallest value is then given by i/N_{tests} , corresponding to the fraction of tests yielding a p-value at least as small. If p-values are calibrated correctly, the empirical and calculated p-values should be equivalent up to statistical fluctuations and lie on a line with gradient one (dashed grey line in the figures). If calculated p-values are larger than the empirical p-values, the statistical procedure is conservative. If they are smaller, the statistical procedure is liberal. In our case, this means that a look-elsewhere/look-everywhere effect has

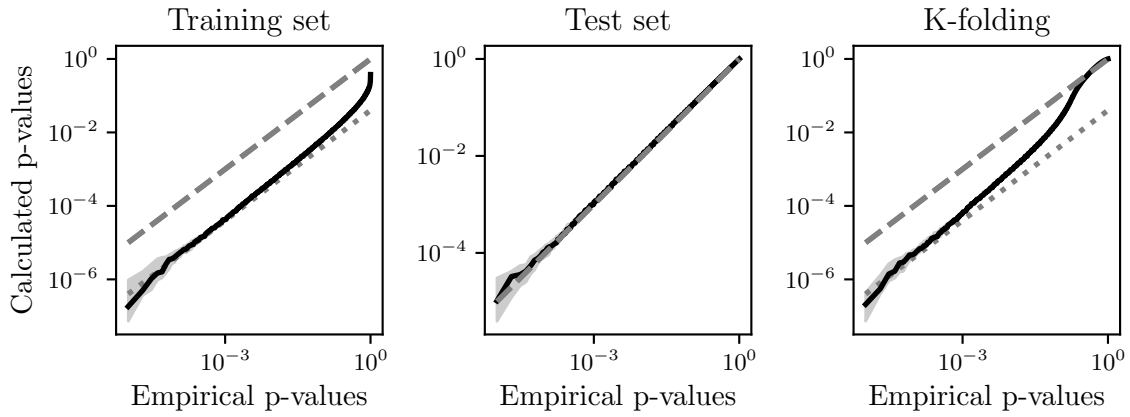


Figure 6.11: Calibration curves showing empirical cumulative p-values and calculated p-values observed for the binned procedure when evaluating on the training set (left), on an independent test set (middle), and using k -folding (right) on the 2D Gaussian toy data. The Bonferroni correction is shown as a grey dotted line.

occurred. To be less vulnerable to statistical fluctuations, we stabilize the performance by performing $10 \cdot N_{\text{tests}}$ tests, randomly subdividing into 10 sets. We then plot the median and 68% quantile at each empirical p-value.

We first focus only on the simpler binned procedure in Fig. 6.11. Here, we clearly see the expected trends for the first two panels: When evaluating on the training set (left panel), we obtain miscalibrated p-values. For a multiple testing scenario with N tests, the Bonferroni correction [193, 194] applies a trials factor of N . This choice of correction is conservative [195]. For our binned tests, where we use 25 bins, the trials factor according to the Bonferroni correction would be 25. We show the expected p-values according to the Bonferroni correction as a grey dotted line in Fig. 6.11. When evaluating on the training set, our observations for low p-values match the Bonferroni correction relatively well. For high p-values close to 1, the Bonferroni correction is no longer accurate and is indeed conservative for our tests. When evaluating on the test set (middle panel), we observe calibrated p-values as expected. However, as mentioned above, these calibrated p-values are obtained at the cost of both training and evaluation statistics. The impact of this on signal sensitivity is investigated in Sec. 6.2.4.

For the evaluation using k -fold cross-validation (right panel), we also observe miscalibrated p-values. When comparing to the evaluation on the training set, the miscalibration is less severe for relatively high p-values. However, for very low p-values, the miscalibration is not improved, and we once again match the Bonferroni correction. While each selection is evaluated on independent test sets, the combination of the k folds assumes an independence of all results. This independence is not given, as we can easily understand by considering a simplified example with 9 bins and $k = 2$ as shown in Fig. 6.12: If the largest excess is relatively small (top row), dividing the data into multiple folds dilutes the excess sufficiently and leads to relatively independent fluctuations in both folds. Different bins are chosen for each fold, and as such, the results from both are largely independent. This is not the case if the excess is large (bottom row). In this case, the excess is likely to still occur in multiple folds. Each of these selects the same bin. If this is the case for all folds, as shown in the example, combining the counts results in the same p-value as training and evaluating on the same set. For larger excesses, the likelihood of the excess occurring in all folds increases. This fits very well with the results observed in the right panel of Fig. 6.11, where p-values are

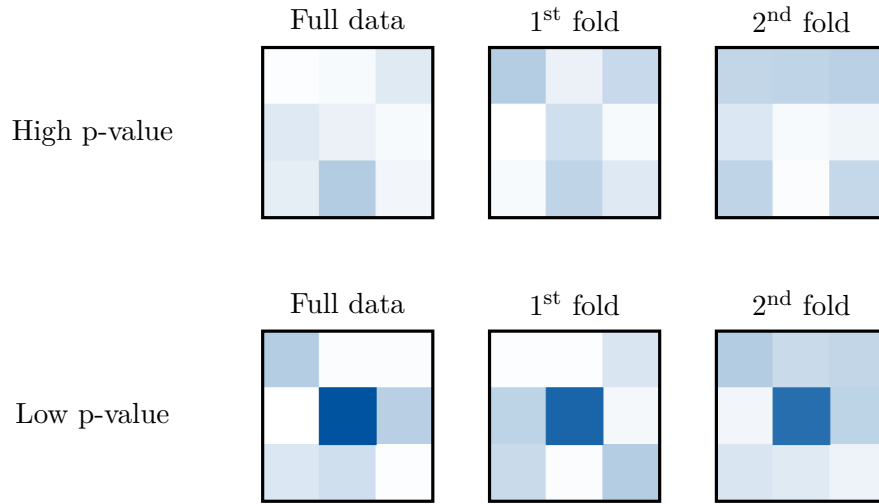


Figure 6.12: Illustration of the miscalibration seen in k -fold cross validation. The blue-scale represents the excess in each bin.

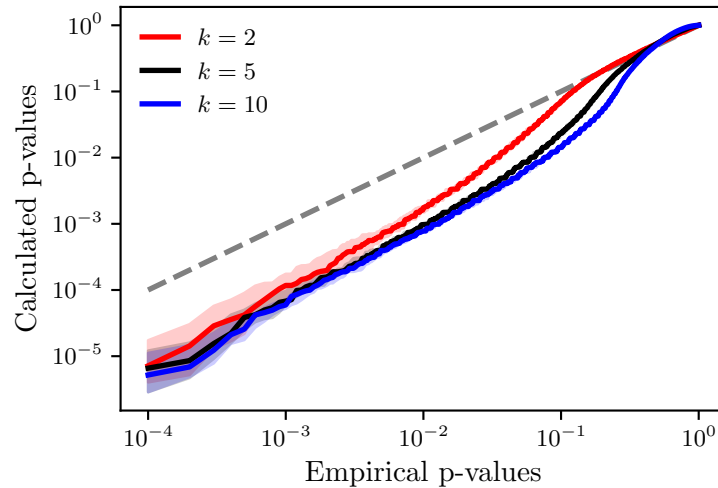


Figure 6.13: Calibration curves showing empirical cumulative p-values and calculated p-values observed for the binned procedure when evaluating using k -folding for different values of k on the 2D Gaussian toy data. Results not contained in Ref. [4].

approximately calibrated when they are large and miscalibrated when they are small, with a transition region between.

We generally expect the degree of miscalibration to vary with k . The effect explained above should become less severe as k increases. However, the overlap between training sets also increases with increasing k , introducing another source of statistical dependence between different folds. We investigate how varying k affects the miscalibration of p-values for our binned procedure in Fig. 6.13 for $k = 2$, our standard choice of $k = 5$, and $k = 10$. We observe a consistent shift of the transition point between calibrated and miscalibrated p-values. The transition occurs at higher p-values, i.e., miscalibration becomes more severe, as k increases. As explained above, this is likely caused by the increasing overlap between training sets.

Having understood the statistical considerations of the simpler binned procedure, we now

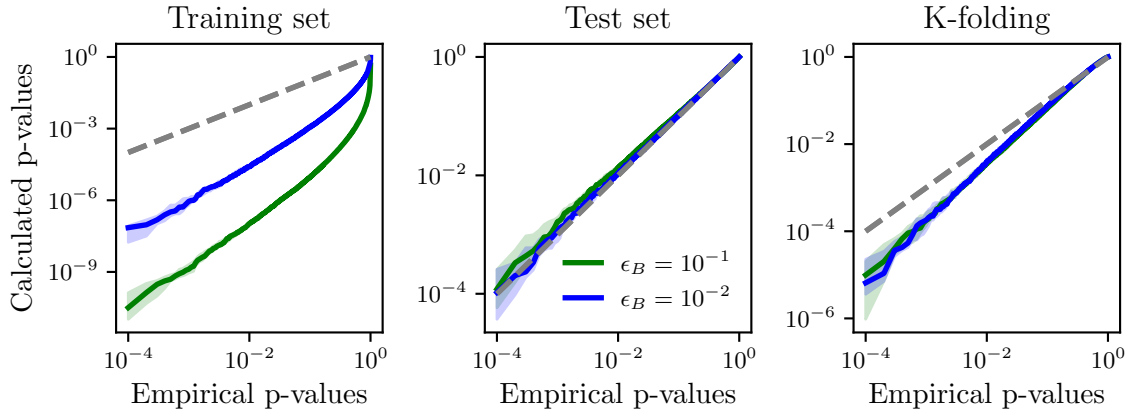


Figure 6.14: Calibration curves showing empirical cumulative p-values and calculated p-values observed for the NN classifier when evaluating on the training set (left), on an independent test set (middle), and using k -folding (right) on the 2D Gaussian toy data.

turn to the results of our classifier-based procedure with the NN without early stopping in Fig. 6.14. Overall, the trends are consistent with the results for the binned procedure in Fig. 6.11: We observe calibrated p-values when evaluating on an independent test set and miscalibrated p-values when evaluating on the training set or using k -fold cross validation. We can understand this similarity by considering what fundamentally occurs in both procedures: Both the binned and the classifier-based procedures select events based on their location in feature space. For the binned procedure, this selection depends on a fixed binning, while for the NN, the selection is defined by the network’s parameters. Any set of parameters results in a particular classifier score, which is a function of the feature space. A cut on this score selects a specific region of the feature space. This selected region can be viewed as the NN’s analog of a bin⁶. In the binned procedure, we pick the best of a fixed set of bins and can thus, based on the Bonferroni correction and the size of the set, approximate the size of the trials factor. If we wanted a fixed set of tests for the NN, we could train by picking the best of a fixed set of randomly generated parameter sets. For N parameter sets, we expect to observe a trials factor of N for low p-values according to the Bonferroni correction, though this would have to be confirmed by further studies. However, in practice, NN trainings are optimizations based on a single random initialization. As a result, the effective trials factor of a NN training is unknown. Nevertheless, this analogy highlights the similarity in the statistical treatment of data between the two procedures.

While the general trends are similar, there are a few notable differences between the binned and NN results. Firstly, for the NN, we choose a cut for our analysis. We show two examples here, namely $\epsilon_B = 10^{-1}$ and $\epsilon_B = 10^{-2}$. For the evaluation on the training set, we see that the tighter cut reduces the degree of miscalibration. We can understand this by considering the relative size of the statistical error, which increases for tighter cuts (see also Sec. 6.1). If the bias between SR and BT events is constant, this leads to a reduced impact on the p-value of this bias as the cut becomes tighter. Secondly, for the binned procedure, we saw that for low p-values, the miscalibration when evaluating on the training set or using k -folding was equivalent. This is not the case for the NN, for which the miscalibration when evaluating on the training set is consistently more severe. There are two key features of the NN, which can easily be pinpointed as possible reasons for this difference in behavior, namely the random

⁶ In contrast to bins, the phase space regions selected by the NN may also be irregular and disjointed.

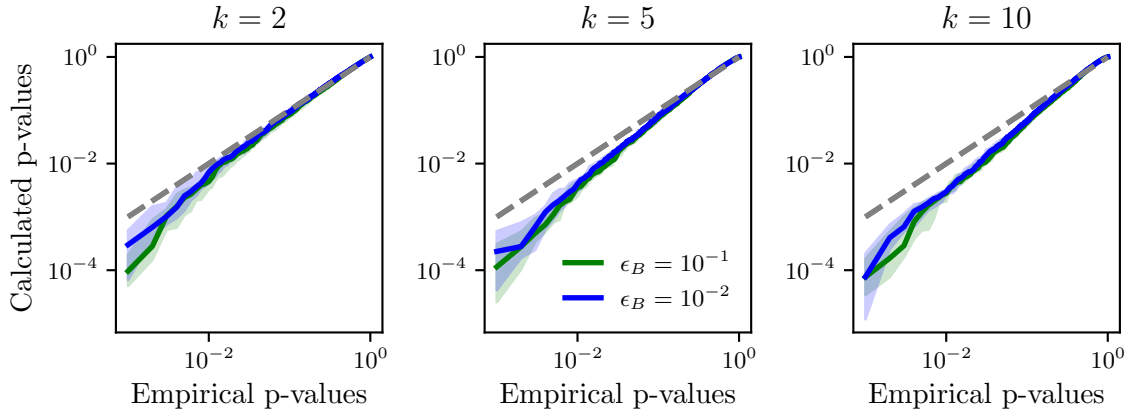


Figure 6.15: Calibration curves showing empirical cumulative p-values and calculated p-values observed for the NN classifier when evaluating using k -folding for $k = 2$ (left), $k = 5$ (middle), and $k = 10$ (right) on the 2D Gaussian toy data. Results not contained in Ref. [4].

initialization and the very large set of possible parameter configurations. Both mean that it is less likely that the exact same region of feature space is selected for different folds. For the binned procedure, the fixed set of bins limits choices and may, as such, cause a more significant overlap. Performing studies where a fixed set of parameter choices is picked for a NN could confirm or disprove this hypothesis. Additionally, to test the impact of the random initialization in isolation, a fixed initialization could be tested.

We also test the calibration when varying k for the NN in Fig. 6.15. As for the binned procedure, we observe that miscalibration becomes slightly more severe as k increases, likely as a result of the larger overlap between training sets.

6.2.3 Look-Everywhere Effects and Overfitting

In our tests with a NN without early stopping, we have seen that NNs incur a look-everywhere effect, much like a simple multiple testing scenario results in a look-elsewhere effect. We now aim to highlight how the look-everywhere effect is connected to overfitting. For this purpose, we consider what happens in the bin selection/NN training. For both, the optimal result of the optimization is $R(x)$ and hence independent of the given data. It relies exclusively on the underlying distributions. However, in practice, optimization on data is always limited by finite statistics and, as such, affected by fluctuations in the training set. In the background-only case, where the underlying distributions of both classes are equivalent, only these fluctuations can be learned, and the network overfits. Evaluating this network on the same data set then leads to the miscalibration of p-values observed in the previous section and as such a look-everywhere effect.

We can test whether this equivalence holds by strategically applying techniques to limit overfitting and evaluating whether the miscalibration of p-values becomes less severe. In our case, we choose to use early stopping. For the NN, we show a comparison between the results of trainings without (solid lines) and with early stopping (dashed lines) in Fig. 6.16. No change is observed for the evaluation on the test set (middle panel), for which p-values are already calibrated when early stopping is not applied. However, for the evaluation on

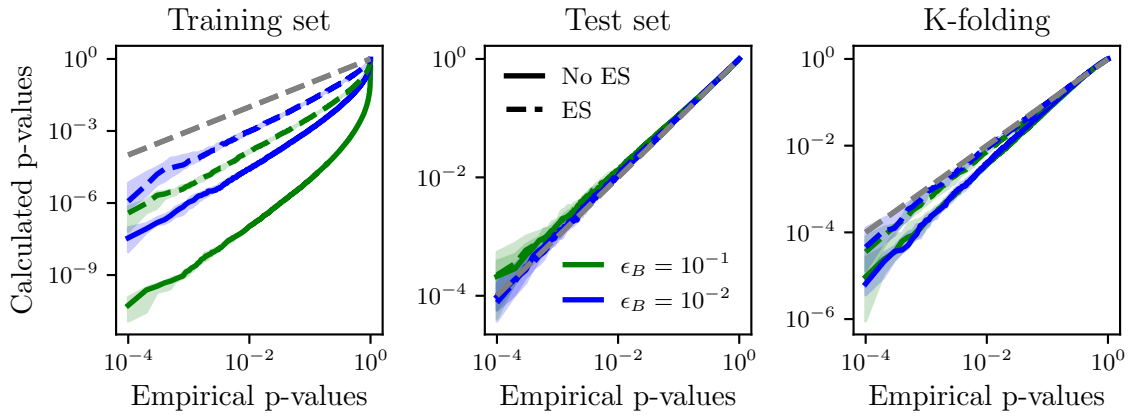


Figure 6.16: Calibration curves and fits showing empirical cumulative probability and calculated p-values observed for the NN without (“No ES”, solid) and with early stopping (“ES”, dashed) for an evaluation on the training set (left), on the test set (middle), and using k -folding (right) on 2D Gaussian toy data.

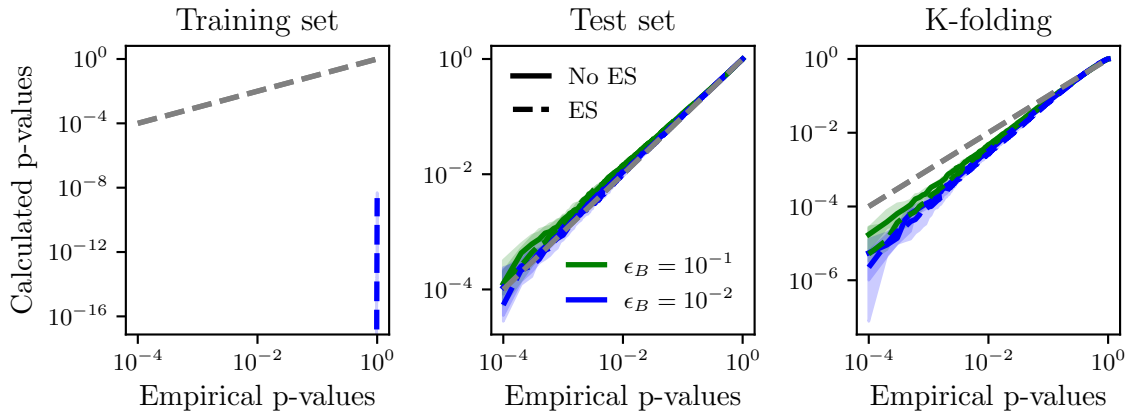


Figure 6.17: Calibration curves showing empirical cumulative probability and calculated p-values observed for the BDT without (“No ES”, solid) and with early stopping (“ES”, dashed) for an evaluation on the training set (left), on the test set (middle), and using k -folding (right) on 2D Gaussian toy data.

the training set (left panel), we observe a clear improvement in the miscalibration. When evaluating using k -fold cross validation (right panel), the previously observed miscalibration is almost completely removed by early stopping.

Besides the NN, we now also consider the BDT classifier in Fig. 6.17, both with (dashed lines) and without early stopping (solid lines). Once again, we observe calibrated p-values when evaluating on an independent test set (middle panel). However, when evaluating on the training set, we observe a much more severe miscalibration than for the NN, with p-values quickly falling below our numerical precision and, as such, not being plotted. In tests with less expressive BDTs (obtained by limiting each decision tree in the BDT to a maximum of 5 leaf nodes instead of 31), this could be reduced somewhat, as seen in Fig. 6.18. Even with this strong limitation of the BDT size, the miscalibration is still significantly more severe than for the NN. Nevertheless, early stopping is able to slightly reduce the miscalibration. This

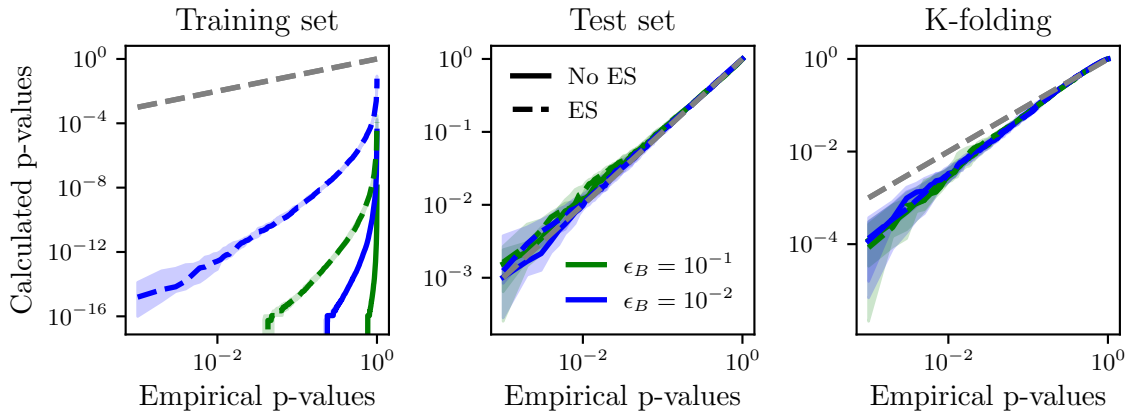


Figure 6.18: Same as Fig. 6.17 but for a less expressive BDT, where each decision tree has a maximum of 5 instead of 31 leaf nodes.

is not the case when evaluating with k -fold cross validation, where miscalibration with and without early stopping shows no difference.

Definitively identifying the reason for this deviation with respect to the NN would require further study. However, we can pinpoint a number of differences between the NN and BDT classifiers. Follow-up studies on these may provide further insights into the observed statistical behavior. Particularly for the k -folding, the lack of a random initialization in the BDT may cause a higher likelihood of the selections between different folds overlapping. The same is true for the binning, which is applied in our BDT implementation and limits the possible feature space selections. However, while these components may affect the results of the evaluation using k -folding, the impact this should have when evaluation occurs on the training set is limited. For this, we find a possible reason for the deviation in the difference between score calculations for NNs and decision tree-based models: As discussed in Sec. 3.3, a decision tree's classifier score is based on the events from the training set, which are sorted into a particular leaf node. A NN, on the other hand, is a function that calculates the classifier score based on its parameters and the values of the input features. As such, while an untrained NN never exhibits overfitting, a BDT with randomly generated splits may still overfit as long as scores are still calculated based on the training set. This overfitting cannot be eliminated by early stopping. We do, of course, still observe some improvement through early stopping as the BDT's splits are stopped from becoming overly focused on fluctuations.

For the binned procedure and the NN, we tested how the miscalibration of p-values when k -folding changes as k is varied in Figs. 6.13 and 6.15 in Sec. 6.2.2. There, the miscalibration was observed to become worse as k is increased and the overlap of the training sets increases. We perform this same test for the BDT classifier in Fig. 6.19, where the same trends are observed.

6.2.4 The Impact of the Look-Everywhere Effect on Signal Sensitivity

Based on our toy data, we have understood a variety of different factors which can impact the severity of look-everywhere effects. We now turn to our more realistic LHC0 R&D data to study how these considerations actually impact signal sensitivity. For this, we first perform

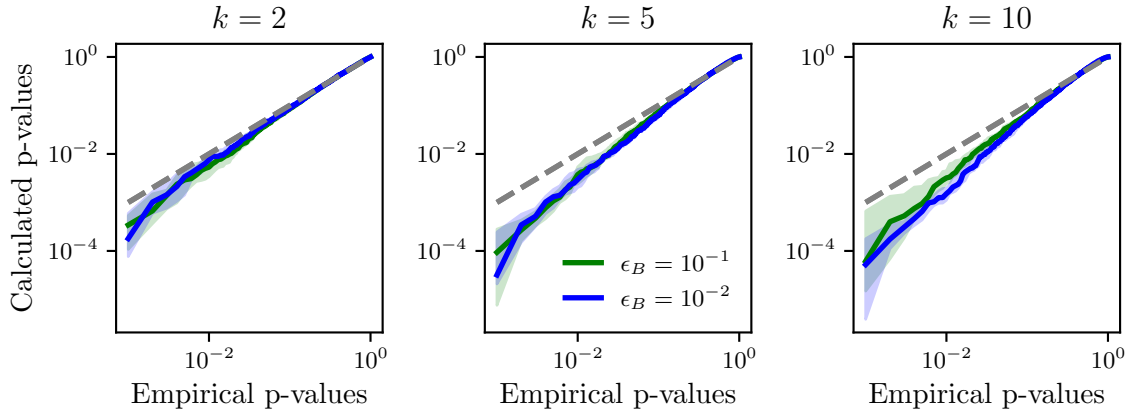


Figure 6.19: Same as Fig. 6.15 but for the BDT classifier.

background-only calibration tests as above, which we then use to obtain the trials factors for each scenario. Then, we turn to studies with signal. Here, we first consider the classification power of our classifiers using the max SIC on a labeled test set before we lastly study the p-values obtained in studies with signal. In all of these tests, we no longer consider the BDT without early stopping, as the difference to the BDT with early stopping in previous studies was minimal. We also add an additional, lower threshold of $\epsilon_B = 10^{-3}$ as this data set is larger.

Starting with the calibration tests in Fig. 6.20, we observe the same general trends seen above in Figs. 6.16 and 6.17. We do, however, also observe that, where miscalibration was already present previously, it is increased. We expect that this is largely due to the larger feature set – four instead of two classification features – but some impact from the larger data set size is also possible. To these results, we fit calibration curves, which allow us to calibrate the p-values obtained in the studies with signal later on. For the fits, we pursue the following strategy: We perform our fits on the medians of $\log_{10} p_{\text{empirical}}(\log_{10} p_{\text{calculated}})$, which ensures that especially the low p-values are fitted correctly. It also means that the linear relationship we observe in the double-logarithmic scale of Fig. 6.20 for very low p-values remains linear in the fit. We make use of this linear region and fit a linear function for $p_{\text{empirical}} < 10^{-2}$. In the more complex region above $p_{\text{empirical}} = 10^{-2}$, we fit a `UnivariateSpline` from `scikit-learn` [117]. As spline fits tend to diverge at the edge of their fit data, we fit this function to the entire spectrum, ensuring well-controlled behavior around the overlap at $p_{\text{empirical}} < 10^{-2}$. The very constrained linear fit allows us to extrapolate slightly beyond our fit region. We limit this extrapolation to $\min p_{\text{empirical}}/10$. No rigorous tests of our fits were performed, as we require only a stable interpolation between the p-values obtained in our tests. However, we plot them in Fig. 6.20 as dashed lines to test them by eye. Here, we observe no obvious issues. We do not perform calibration fits when evaluating on the training set for the BDT or the NN without early stopping as we obtain p-values below our numerical accuracy, i.e., the p-value calculation returns $p = 0$.

Now that we know how to calibrate our p-values, we turn to our tests with signal. We study the performance with signal by varying the number of injected signal events N_{sig} . For each signal injection, we train ten classifiers (when k -folding, we repeat the entire procedure ten times) with randomly selected signal events. We evaluate on the respective evaluation set to obtain p-values, which we discuss below, as well as on a labeled test set, based on which we

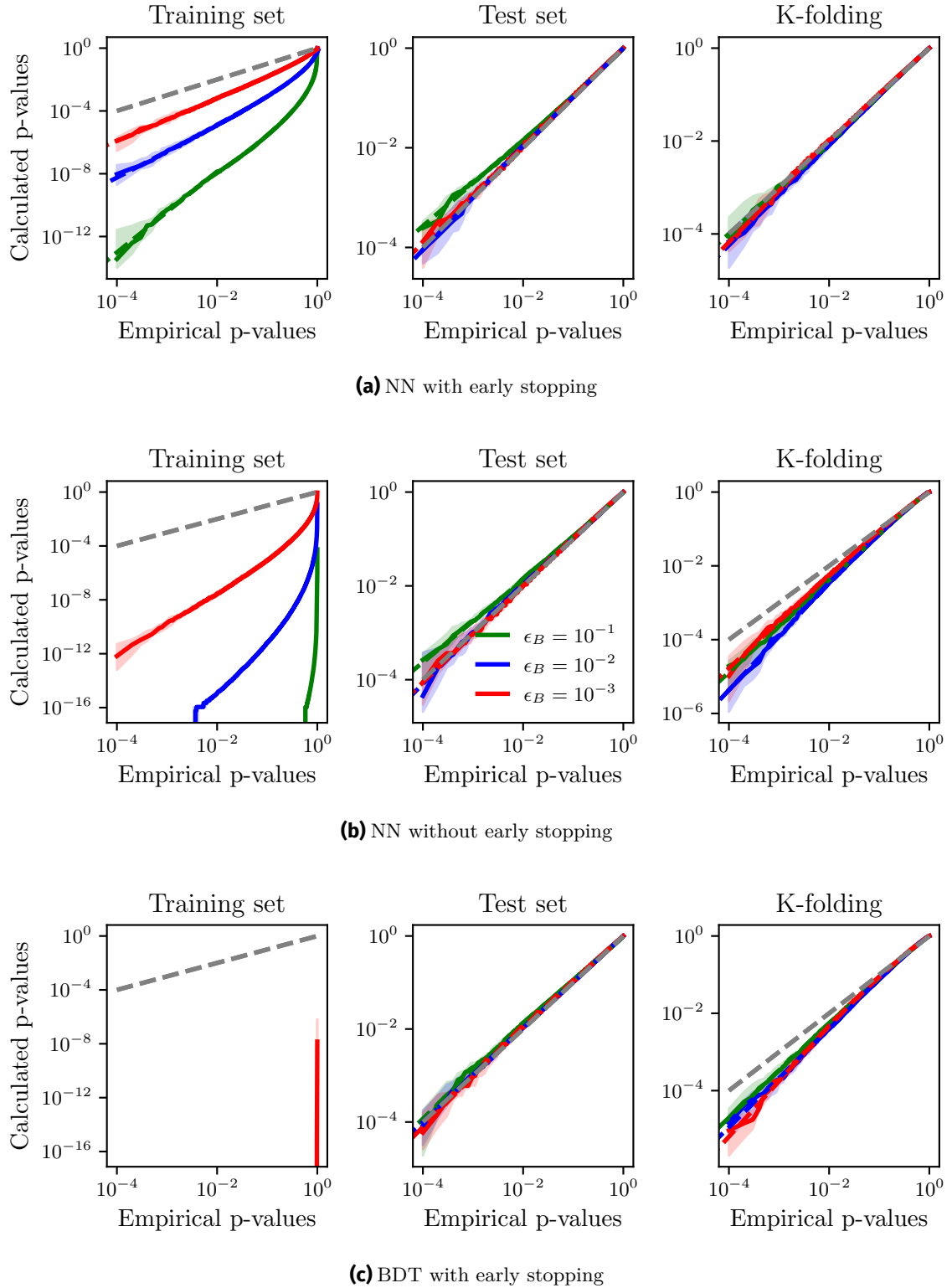


Figure 6.20: Calibration curves (solid) and fits (dashed) showing empirical cumulative probability and calculated p-values observed for the NN with early stopping (top), the NN without early stopping (middle) and the BDT without early stopping (bottom) for an evaluation on the training set (left), on the test set (middle) and using k -folding (right) on LHC0 data.

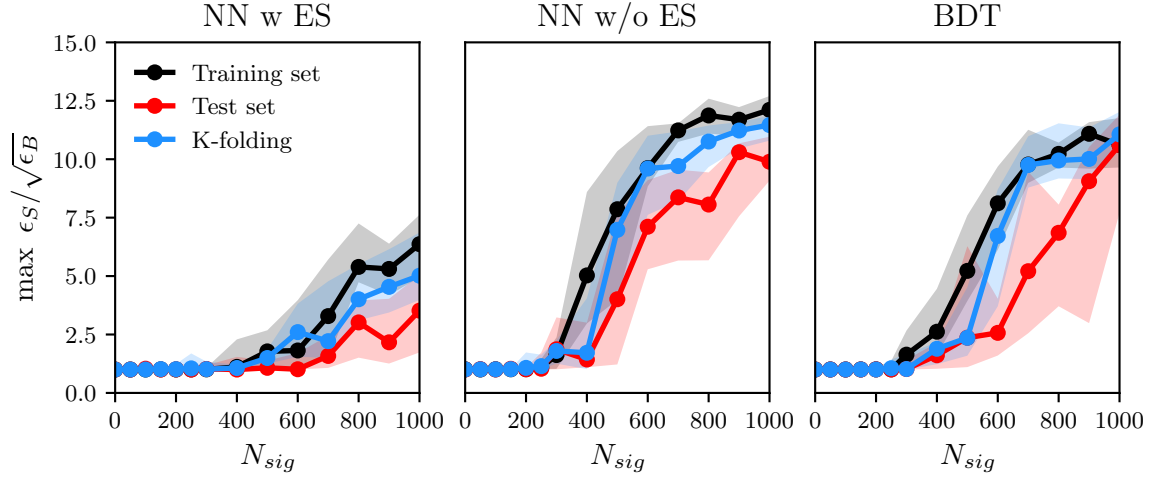


Figure 6.21: Maximum SIC as a function of the signal injection N_{sig} for classifiers trained on the statistics retained when evaluating on the training set (black), on the test set (red) and using k -folding (blue) for NN with early stopping (left), without early stopping (middle) and for the BDT with early stopping (right).

obtain the max SIC of each classifier. This allows us to study the signal sensitivity of the classifier itself without taking the look-everywhere effect into account. We show these results in Fig. 6.21. For all three classifier setups, we observe the same general trend, namely that the classifier sensitivity depends strongly on the evaluation method. This is caused by the different training statistics each evaluation method uses. For the evaluation on the training set, the training set uses 100% of the available data. For the evaluation with 5-fold cross validation this is reduced to 80% and for the evaluation on an independent test set to 50%. The typical performance drop-off for low signal injections of weakly supervised anomaly detection thus sets in earlier if the training set size is reduced. Beyond this dependence on the type of evaluation, we also observe that the NN with early stopping (left panel) performs significantly worse than the NN without early stopping (middle panel) or the BDT (right panel). We can understand this behavior by taking into account the epoch dependence of validation loss and max SIC tracked throughout a training in Fig. 5.12 in Sec. 5.2. There, we saw that the validation loss is not strongly correlated with the signal sensitivity of a classifier. In the shown example, the classifier found the signal at epoch 40, while the validation loss began to rise at approximately epoch 20. Early stopping applied to this training would therefore have stopped the training before the signal was discovered, leading to the suboptimal performance we see in Fig. 6.21.

We have now considered both the signal sensitivity of the different classifiers and whether and to what extent it incurs a look-everywhere effect. We now combine both components by considering the p-values obtained in an analysis with signal, which we show for all three classifiers in Fig. 6.22. We show both uncorrected p-values as dashed lines and p-values corrected using the calibration fits shown in Fig. 6.20. As we limit the extrapolation of our calibration fits, corrected p-values are cut off at 10^{-5} . Values affected by this limit are marked by a triangle instead of a circle marker. Where calibration fits were not possible (evaluation on the training set for the NN without early stopping and the BDT), only the uncorrected results are shown.

We can see the severe miscalibration of p-values incurred by evaluating on the training set.

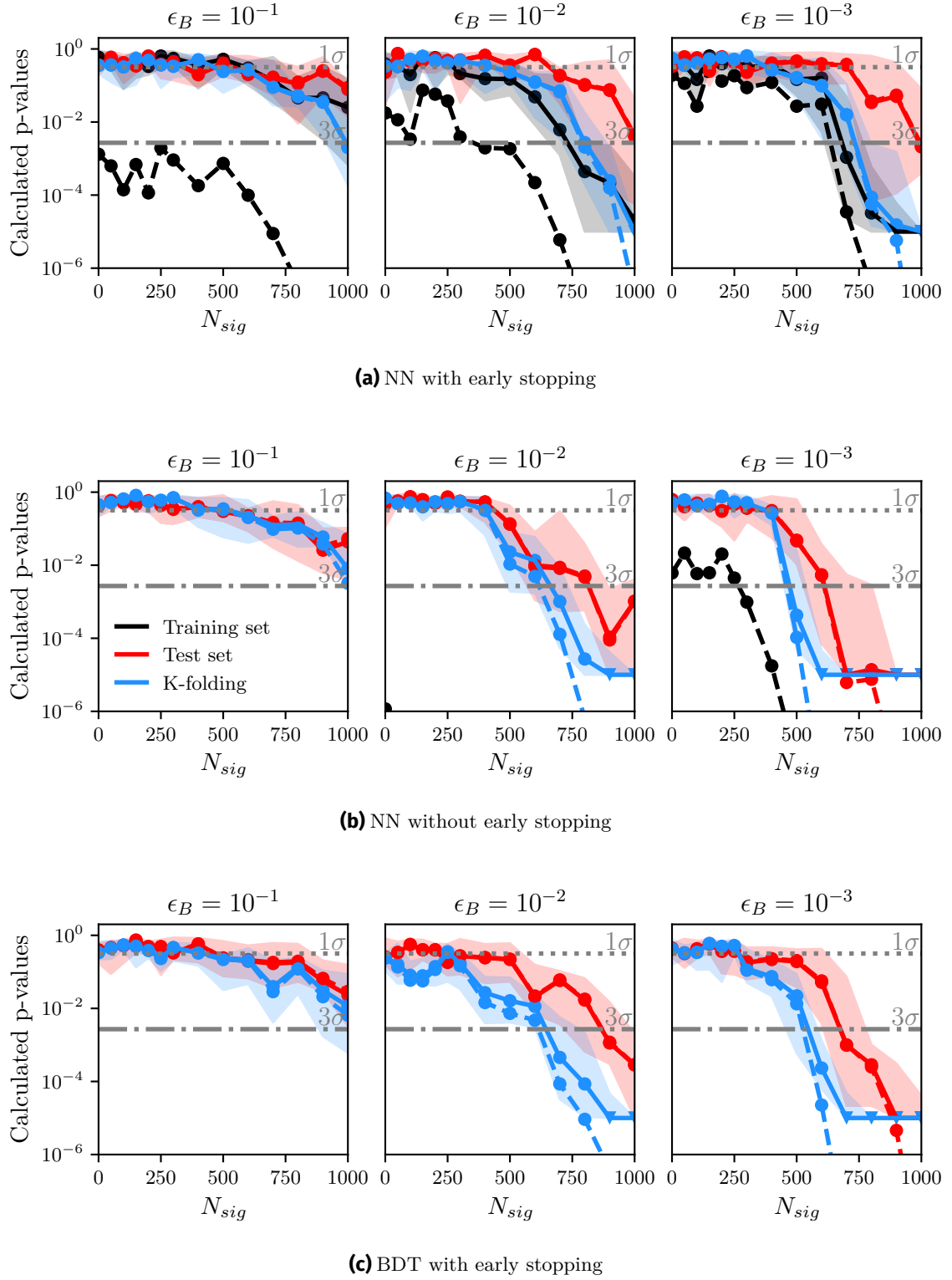


Figure 6.22: Uncorrected (dashed) and corrected (solid) p-values for different signal injections for NN with early stopping (top), without early stopping (middle), and for BDT with early stopping (bottom) evaluated on the training set, the test set, and using k -folding at three different working points (left, middle, right). The median of ten runs is shown. For the corrected p-values, we additionally show an error band equivalent to a 68% quantile. Corrected p-values are limited to $p \geq 10^{-5}$ by our extrapolation limit (triangle markers).

Indeed, many of the results for an evaluation on the training set fall outside the plotting boundaries. Where calibration is possible (NN with early stopping), we nevertheless observe the strongest signal sensitivity of all evaluation methods. As already seen in Fig. 6.21, the NN without early stopping and the BDT are able to provide a significantly improved sensitivity to lower signal injections. We also observe a clear advantage in sensitivity obtained when using k -fold cross validation compared to an evaluation on the test set. Here, in addition to the larger training set statistics, which already improved results in Fig. 6.21, we also make use of the improved evaluation statistics obtained from iterating over the entire data set. While the required calibration when k -folding is minimal, as seen in the small difference between corrected and uncorrected p-values, it is still necessary. This would also affect the results of our studies in Sec. 6.1, where we used k -fold cross validation without calibration, though the effect should be small. Evaluating on an independent test set causes a slight loss in sensitivity. However, it is always a safe option able to completely eliminate look-everywhere effects and may still be able to discover new physics.

6.2.5 Conclusion and Outlook

In the above studies, we have investigated how the statistical behavior of an analysis depends on both the classifier and the evaluation method. In our tests, we have seen a miscalibration of p-values for ML classifiers, which closely mirror classical look-elsewhere effects. As these classifiers do not have explicit and separate multiple tests but optimize over the full phase space, we call the observed miscalibration look-everywhere effects. By applying methods to avoid overfitting of the classifiers (early stopping), we were able to show that overfitting and look-everywhere effects are closely related. In general, we observe a strong dependence of the extent of miscalibration on the explicit classifier architecture, as seen both with the application of methods such as early stopping and with the switch from a NN to a BDT classifier. The BDT generally experienced more severe miscalibration. This highlights that any calibration test should always be conducted with the final architecture.

In our studies of different evaluation methods, we considered an evaluation on the training set, on an independent test set, and using k -fold cross validation. Generally, we observed strongly miscalibrated p-values when evaluating on the training set, slight miscalibrations when evaluating using k -folding, and calibrated p-values when using an independent test set. Each of these methods results in different training and evaluation statistics, e.g., evaluation on the training set uses the full available statistics for both, while evaluation on an independent test set uses 50% for each. This directly affects the signal sensitivity. While an evaluation on the training set leads to the best available classifier performance, the severity of the miscalibration can make a calibration of p-values prohibitive. In our studies, evaluating using k -folding was able to obtain the best trade-off between calibration and classifier performance, with a detection of the signal at relatively low signal injections. This supports the choice of weakly supervised analyses using k -folding [44–46] (studies for the calibration of Ref. [46] can be found in Ref. [157]). However, evaluating on an independent test set is always a safe option, leading to calibrated p-values and, as such, can reduce the computational cost of calibration testing.

Our studies have generally been aimed at pushing tests only as far as was required to observe underlying trends. To calibrate analysis results, p-values must be calibrated at least to 5σ to allow for a discovery, meaning p-values of $\mathcal{O}(10^{-7})$. In our studies, we allowed for a slight extrapolation of our linear fit in the double-logarithmic space. Extending this

extrapolation further may allow calibrating p-values to more than 5σ without significantly increased computation. However, the validity of this extrapolation should be tested before applying it in an analysis.

There are many interesting avenues for further study opened up by this research. In particular, the precise impact of NN and BDT training on the extent of miscalibration may deepen our understanding of their statistical behavior. For example, in a binned analysis, we have seen that the miscalibration when k-folding for low p-values matches that when evaluating on the training set. This is not observed for our ML classifiers, where k-folding leads to significantly reduced miscalibration. We have discussed possible reasons for this based on the NN, namely, its random initialization and that there is a fixed set of possible parameters. This needs to be studied further. Additionally, we observe more severe miscalibration for the BDT classifier than for the NN. Possible reasons for this increase include the method of score calculation of decision tree classifiers and the binning our BDT architecture uses. Studying whether and how these differences between our architectures contribute to their different statistical behavior could also help deepen our understanding of how they identify signal.

While we have studied these statistical trends in the context of a specific application, namely weakly supervised anomaly detection with a significance calculation based on direct background estimation, we expect many of our observations to hold beyond this. For example, fit-based bump hunts as performed in previous weakly supervised analyses [44–46] are expected to be similarly affected by false excesses in the SR. Beyond weak supervision, other anomaly detection methods and other ML-based analyses, such as parameter estimation, may also experience a connection between overfitting and statistical miscalibration, which should be accounted for.

In this thesis, we work to understand and systematically improve weakly supervised searches for new physics. We have focused on two components of weakly supervised searches, namely the classifier, which we study in Chapter 5, and the significance, which we study in Chapter 6.

In Sec. 5.1, we exchange the standard MLPs used in early weakly supervised studies for BDTs, in particular, bagged ensembles of gradient BDTs. This change of classifier aims to address the susceptibility of MLPs to uninformative features. We find that our BDT classifiers are highly robust to uninformative features, with strong performance still observed when 50 out of 54 features are uninformative. In contrast, a significant performance breakdown is already observed for the MLP with just 2 uninformative features in 6. This difference in performance is fundamentally caused by the different network structure, with the MLP’s rotational invariance forcing it to perform feature engineering even if the feature space is tabular, as is the case for high-level features in particle physics. BDTs, on the other hand, make direct use of the orientation of the feature space, which makes them ideal for use with tabular data, particularly if data sets are small. In addition to the network structure, we observe that the BDT benefits significantly from our ensembling strategy, enabled by the BDT’s fast runtimes. The increased robustness of BDTs to uninformative features also allows the use of larger and therefore more model-agnostic feature sets, as we showed with subjettiness-based feature sets. This has been further shown in Refs. [5, 26], which use BDTs for large, high-level feature sets. These larger feature sets also allow us to push our sensitivity to lower signal injections. Significance improvement factors of at least 10 are achieved for an initial significance of 0.6σ rather than 1.1σ with our larger subjettiness-based feature sets. In Ref. [5], this sensitivity could be further improved to significance improvements of at least ten being achieved at 0.3σ with even larger feature sets.

In Sec. 5.2, we explore how a variety of optimizations involved in weakly supervised anomaly detection can be made in a data-driven and model-agnostic manner. We introduce the ARGOS metric, which we show to be mathematically optimal for perfect background templates and, in practical applications, highly correlated with the significance improvement. We show how this metric achieves close-to-optimal results on a certain signal benchmark (compared with a supervised metric) in epoch selection, hyperparameter optimization, and architecture selection. Simply switching from a validation loss-based epoch selection to an ARGOS-based epoch selection improves sensitivity, yielding significance improvements of more than 10 for 400 signal events in one million background events instead of 600 for the IAD and 700 instead of 1000 for CWoLa Hunting. We also explore data-driven feature selection for the first time and investigate whether data-driven optimizations increase the rate of false discoveries, where we find no substantial issues.

In Sec. 6.1, we establish direct background estimation as an alternative for the standard fit-based bump hunt as performed in previous weakly supervised analyses [44–46]. These fit-based analyses were impacted by background sculpting, i.e., sculpting of the background resonant mass spectrum, caused by the AD event selection due to correlations between the anomaly score and the resonant mass. Direct background estimation avoids sculpting by

estimating the significance through a cut and count analysis, which does not depend on the shape of the spectrum. We simply compare the number of events retained from the BT and SR after a cut is applied to the anomaly score. As a realistic BT contains some mismodeling, the systematic shift δ_{sys} between BT and SR must be estimated. We develop methods to obtain δ_{sys} on simulation and in a data-driven manner, and test them using CWoLa Hunting and CATHODE. We obtain no false discoveries in the background-only case, even for a highly inaccurate BT, which we obtain by using CWoLa Hunting with the angular distance ΔR as a classification feature. For analyses with signal, we achieve discovery-level significances for both CWoLa Hunting and CATHODE despite using a relatively conservative estimate of the systematic error on δ_{sys} . Improving the estimate of this error would further increase the discovery potential of direct background estimation, as we show by testing a very liberal choice of $\sigma_{\text{sys}} = 0$, which results in an 18σ discovery of our benchmark signal with CATHODE. While this choice would be inadvisable for an analysis, it demonstrates the potential of direct background estimation to discover much lower signal amounts.

In Sec. 6.2, we investigate whether machine learning anomaly detectors incur miscalibrations of p-values similar to look-elsewhere effects. We find that the statistical behavior of binned analyses, where the best of several bins is chosen, and of classifier-based anomaly detectors show strong similarities. Both result in calibrated p-values when evaluated on an independent test set and miscalibrated p-values when evaluated on their training set or using k -fold cross validation. We call these miscalibrations of the classifiers look-everywhere effects since ML classifiers look everywhere in the phase space instead of only in specific, fixed positions. We draw a parallel between overfitting and look-everywhere effects as both refer to the failure of a network to generalize. We apply techniques to limit overfitting, namely early stopping, and show that this reduces the observed miscalibration of p-values. Lastly, we explore how look-everywhere effects affect signal sensitivity by obtaining and calibrating p-values in the presence of signal. Here, we observe a trade-off between training statistics of each evaluation method and the severity of miscalibrations. In our tests, the best performing setup uses k -fold cross validation, which incurs only slight miscalibrations and retains most of the training set and all of the evaluation set statistics. However, while evaluating on an independent test set was found to result in a slight loss of sensitivity, this procedure always results in calibrated p-values, which should not be discounted. Our investigations here provide clear, practical guidance for future ML analyses.

While our focus has been on weakly supervised anomaly detection, much of the research presented in this thesis should hold for other anomaly detection methods and beyond. For example, the use case of BDTs is general – and well known in the machine learning community – and one should keep BDTs in mind whenever data is limited, or feature sets are tabular. Our methods for model selection and to obtain significances with direct background estimation can be applied to any anomaly detector as long as a background-only background template can be derived. This background template does not necessarily need to be used in the anomaly detector’s training. The broadest applicability is found for our studies on look-everywhere effects, which are expected to hold relatively generally for machine learning models trained and applied on data.

Research on anomaly detection is just beginning, with a variety of open questions remaining, both directly related to the work in this thesis – a variety of these have been highlighted throughout the past two chapters – and beyond. While there have been attempts to control background sculpting, both in Ref. [38] and in the ATLAS and CMS searches conducted thus far [44–46], it remains a problem for fit-based searches using most anomaly detectors. Additionally, studies of weak supervision using low-level features have thus far not been able

to achieve the sensitivity to low signal injections that high-level features can provide [27–30]. Nevertheless, low-level features as input would further improve how model-agnostic weakly supervised analyses can be. While from-scratch training is unlikely to match the sensitivity of BDTs with high-level features anytime soon, pretrained foundation models already improve low-level feature performance [29]. However, it needs to be investigated whether such models are biased significantly by their pretraining to ensure that searches remain as model-agnostic as possible.

With the improvements suggested in this thesis and other developments in recent years – for example, with respect to improved background template quality [27, 28, 41] – the next round of weakly supervised anomaly detection searches is likely to be much more sensitive than the last. Additionally, the increase of data statistics – both with the addition of Run 3 data and the soon-to-come HL-LHC – continually increases the likelihood that new physics will soon be found. As such, there is a bright future for weakly supervised anomaly detection ahead.

In this appendix, we explain a number of different setup specifications in detail, which are only briefly mentioned in the main text.

A.1 DENSITY ESTIMATION SETUP

All density estimators used in this work are built on the setup of Ref. [41]. This setup was adapted for Ref. [2], for which the DE code was published in Ref. [183]. As a default, we also use the hyperparameters specified in Ref. [41]. Fundamentally, the density estimation in this setup is based on Conditional Flow Matching (see Sec. 3.5.2).

First, we prepare the data: A frequency embedding of the time parameter t is used, such that

$$\beta t = (\sin(\pi t), \cos(\pi t), \dots, \sin((L' + 1)\pi t), \cos((L' + 1)\pi t)), \quad (\text{A.1})$$

where $\beta : [0, 1] \rightarrow \mathbb{R}^{2(L'+1)}$ and the embedding dimension $L' = 3$ as a default. The resonant feature m is first normalized to $[-1, 1]$ and then also frequency embedded with $\alpha : [-1, 1] \rightarrow \mathbb{R}^{2L}$, where $L = 9$ is the default. A context and an input vector,

$$\gamma_c(m, t) = \phi_c(\alpha(m), \beta(t)) \quad \gamma_i(x, m, t) = \varphi(x, \alpha(m), \beta(t)), \quad (\text{A.2})$$

are prepared from the time parameter t , the resonant mass m and the features $x \in \mathbb{R}^d$ using two MLPs $\phi_c : \mathbb{R}^{2(L+L')} \rightarrow \mathbb{R}^n$ and $\phi_i : \mathbb{R}^{2(L+L')+d} \rightarrow \mathbb{R}^n$. As a default, these MLPs consist of a single layer with 256 nodes and ReLU activation.

To learn the velocity fields from these vectors, a Residual Network (ResNet) [196] architecture implemented using `nflows` [197] in `PyTorch` [198] is used. K ResNet blocks are used, with the k^{th} block $h^k : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^n$ taking h^{k-1} (for $k = 0$, $h^0 = \gamma_i$) and the context vector γ_c as input. It outputs

$$h^k = h^{k-1} + \phi^k(h^{k-1}) \cdot \gamma_c, \quad (\text{A.3})$$

where $\phi^k : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is the ResNet block's corresponding MLP. As a default, this MLP consists of 2 layers with 256 nodes each with a dropout of 0.2 and batch normalization between layers. The output of the final ResNet block is passed through a linear layer, which returns the output's dimensionality to the data dimensionality d .

By default, the training is conducted over a maximum of 2000 epochs with a batch-size of 4096 using the AdamW [199] optimizer¹ with a learning rate of $3 \cdot 10^{-4}$. After 100 epochs have been completed, the negative log-likelihood of the validation data is evaluated every 5 epochs. A learning rate scheduler uses this metric to multiply the learning rate by a factor of 0.3 once the negative log likelihood has not improved over 50 epochs. The training is stopped if it has not improved over 100 epochs. After training, the 10 best models are selected to sample from.

¹ AdamW largely works the same as Adam but implements weight decay differently.

Samples are generated by sampling from the base distribution and integrating the vector field from $t = 1$ to $t = 0$. This is done using `odeint` from Zuko [200] using the Dormand-Prince method [155, 201] to perform the integration.

A.2 χ^2 METRIC FOR BACKGROUND SCULPTING

In Sec. 5.2, we investigate the extent of background sculpting observed when applying our method for model selection by using a metric from Ref. [38]. This metric calculates a binned χ^2 assuming Poisson statistics to compare spectra before and after a cut has been applied.

The χ^2 metric is calculated as follows: First, quantile-based binning is applied to the m_{JJ} spectrum before the cut with $n_{\text{bins}} = 300$. The number of events per bin normalized to unit area is then given by

$$n^{\text{full}} = a \frac{N_{\text{tot}}}{n_{\text{bins}}} \quad (\text{A.4})$$

with the normalization factor a . The same binning is then applied to the spectrum after the cut to obtain the number of selected events in bin i normalized to unit area

$$n_i^{\text{sel}} = \frac{a}{\varepsilon_{\text{sel}}} N_i^{\text{sel}}, \quad (\text{A.5})$$

where ε_{sel} is the selection efficiency and N_i^{sel} is the number of selected events. Using these counts, the χ^2 is then defined as

$$\chi^2 = \sum_i \frac{(n^{\text{full}} - n_i^{\text{sel}})^2}{a n^{\text{full}}} \cdot \frac{\varepsilon_{\text{sel}}}{1 - \varepsilon_{\text{sel}}}. \quad (\text{A.6})$$

B

ADDITIONAL RESULTS FOR DIRECT BACKGROUND ESTIMATION

In this appendix, we present additional results complementing those in Sec. 6.1. Many of these overlap with results shown there and are included here for completeness. If not otherwise specified, results are taken from Ref. [2].

B.1 δ_{sys} VALUES USED IN THE ANALYSIS

In Sec. 6.1, we show different estimates of the relative systematic shift δ_{sys} in Fig. 6.2 for CWoLa Hunting and in Fig. 6.5 for CATHODE. In these figures, we can see the overall behavior across a variety of thresholds. However, for the later analysis δ_{sys} at particular thresholds, namely $\varepsilon_{\text{BT}} = 10^{-2}$, 10^{-3} and 10^{-4} , is used. We include these values in Tab. B.1. For comparison, the respective total statistical error σ_{stat} as defined in Eq. 6.12 is also included.

Data set		Baseline			ΔR		
ε_B		0.01	0.001	0.0001	0.01	0.001	0.0001
IAD	$\delta_{\text{sys}}^{\text{data}}$	-0.01	0.01	0.05	0.01	0.06	0.20
	σ_{stat}	0.04	0.13	0.34	0.04	0.12	0.34
CWoLa	$\delta_{\text{sys}}^{\text{data}}$	0.14	0.16	0.20	1.01	1.02	1.02
	$\delta_{\text{sys}}^{\text{MC}}$	0.14	0.11	0.05	0.98	1.04	0.94
	σ_{stat}	0.04	0.13	0.35	0.04	0.13	0.35
CATHODE	$\delta_{\text{sys}}^{\text{data}}$	0.06	0.16	0.71	0.13	0.22	0.50
	$\delta_{\text{sys}}^{\text{MC}}$	0.07	0.14	0.25	0.12	0.15	0.20
	$\delta_{\text{sys}}^{\text{SB}}(S/B = 0)$	0.01	0.08	0.77	0.06	0.10	0.66
	$\delta_{\text{sys}}^{\text{MC} \oplus \text{SB}}(S/B = 0)$	0.07	0.16	0.81	0.13	0.18	0.68
	$\delta_{\text{sys}}^{\text{SB}}(S/B = 0.64\%)$	0.02	0.13	0.90	0.07	0.16	0.79
	$\delta_{\text{sys}}^{\text{MC} \oplus \text{SB}}(S/B = 0.64\%)$	0.07	0.19	0.94	0.14	0.21	0.81
	σ_{stat}	0.03	0.09	0.30	0.03	0.09	0.30

Table B.1: Relative systematic shift δ_{sys} estimated from an analysis without signal on Pythia data ($\delta_{\text{sys}}^{\text{data}}$), Herwig MC ($\delta_{\text{sys}}^{\text{MC}}$) as well as for CATHODE in a data-driven manner on the SB of Pythia data with and without signal ($\delta_{\text{sys}}^{\text{SB}}(S/B = 0)$ and $\delta_{\text{sys}}^{\text{SB}}(S/B = 0.64\%)$ respectively). For the IAD, $\delta_{\text{sys}}^{\text{data}}$ is determined on the Pythia reproduction of the LHCO R&D data set. The results are based on 10 classifier runs in each signal region. For CATHODE, independent DEs are used for each classifier run. For reference, the statistical error σ_{stat} is also given.

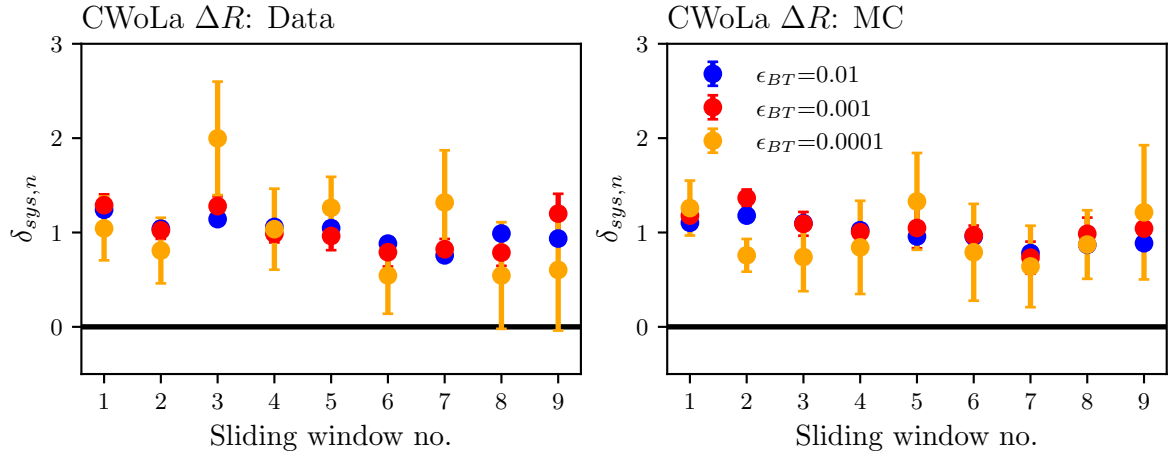


Figure B.1: Same as Fig. 6.3 but for the Baseline + ΔR features. Figure not included in Ref. [2].

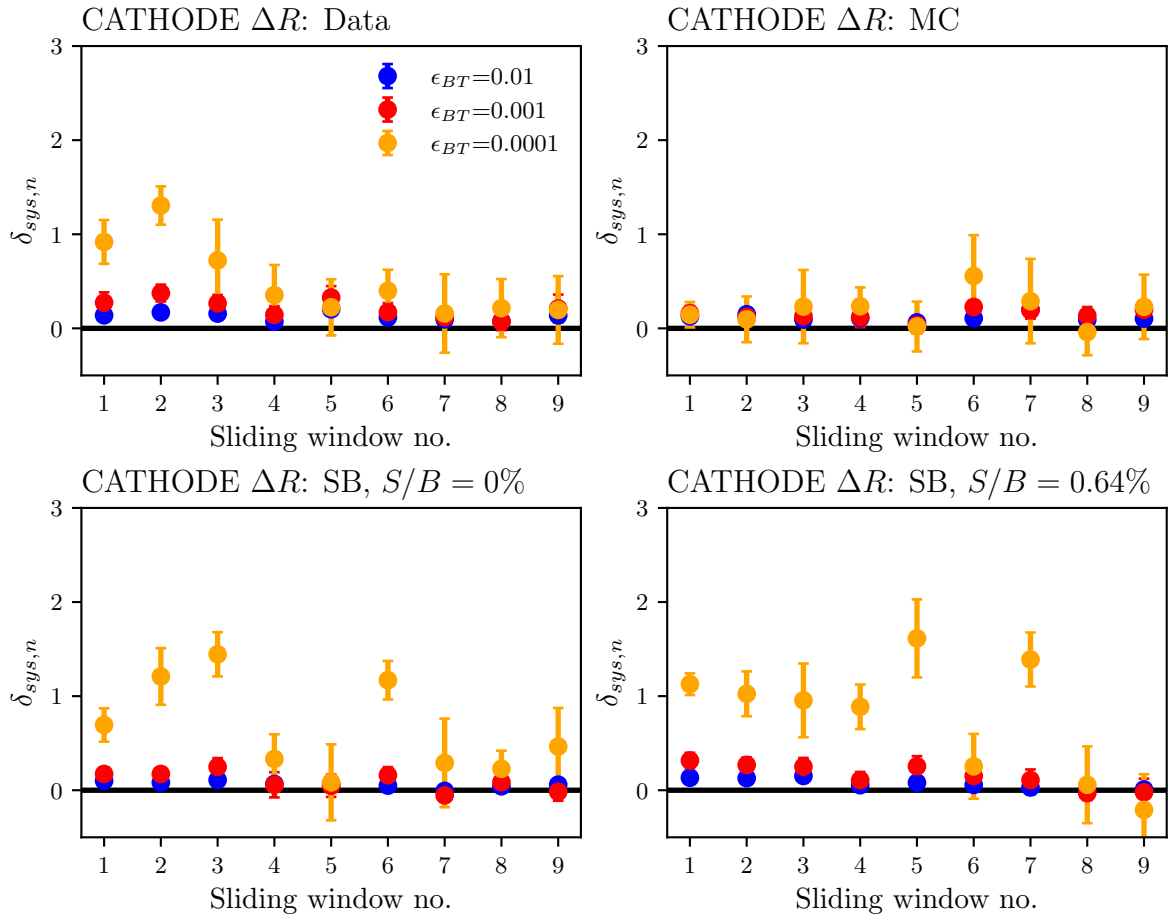


Figure B.2: Same as Fig. 6.6 but for the Baseline + ΔR features. Figure not included in Ref. [2].

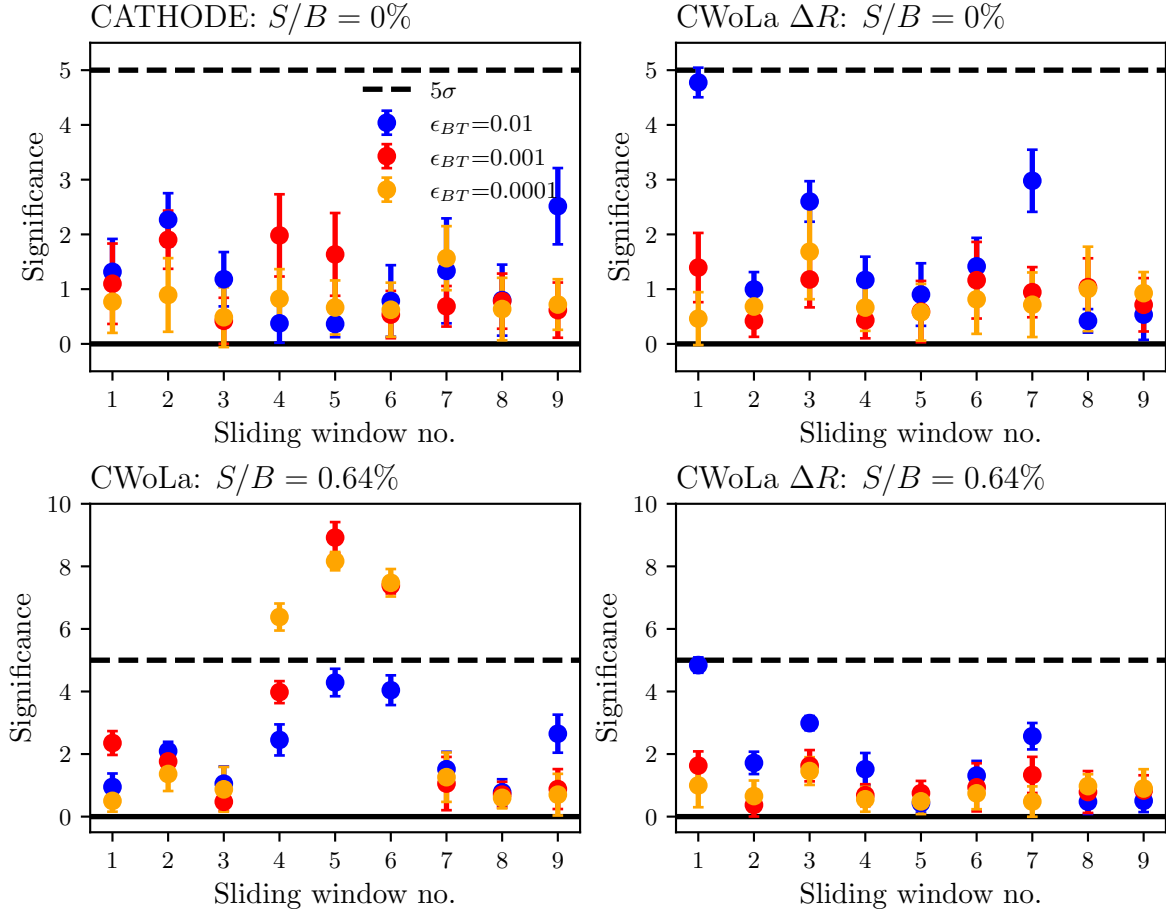


Figure B.3: Significance $\mathcal{S}$, Eq. 6.9, for the different signal regions for CWoLa Hunting without (top) and with signal injection (bottom) using the Baseline (left) and Baseline + ΔR (right) features using $\delta_{\text{sys}}^{\text{MC}}$ with $\sigma_{\text{sys}} = 0$. The error bars indicate the variance of the significance across 10 classifier runs.

B.2 WINDOW-WISE ESTIMATION OF δ_{sys}

For the Baseline features, Figs. 6.3 and 6.6 show the window-dependent structure of δ_{sys} for each of our methods of estimating δ_{sys} . In Figs. B.1 and B.2, we include the same studies for the Baseline + ΔR features. We generally observe the same trends, namely that no clear underlying structure is visible.

B.3 SIGNIFICANCES WHEN USING $\sigma_{\text{sys}} = 0$

In most of our studies, we elect to use a relatively conservative estimate for the error on δ_{sys} , namely $\sigma_{\text{sys}} = \delta_{\text{sys}}$. With this, we obtain no false discoveries in the background-only scenario. To ensure that this is not solely due to the conservative choice of σ_{sys} , we test a very liberal and inadvisable choice of $\sigma_{\text{sys}} = 0$ here. For CWoLa Hunting, we see in Fig. B.3 that larger variations in the significances of the background-only case (top) occur, largely below 3σ . As discussed in Sec. 6.1.4, we see a relatively large deviation of just below 5σ for window

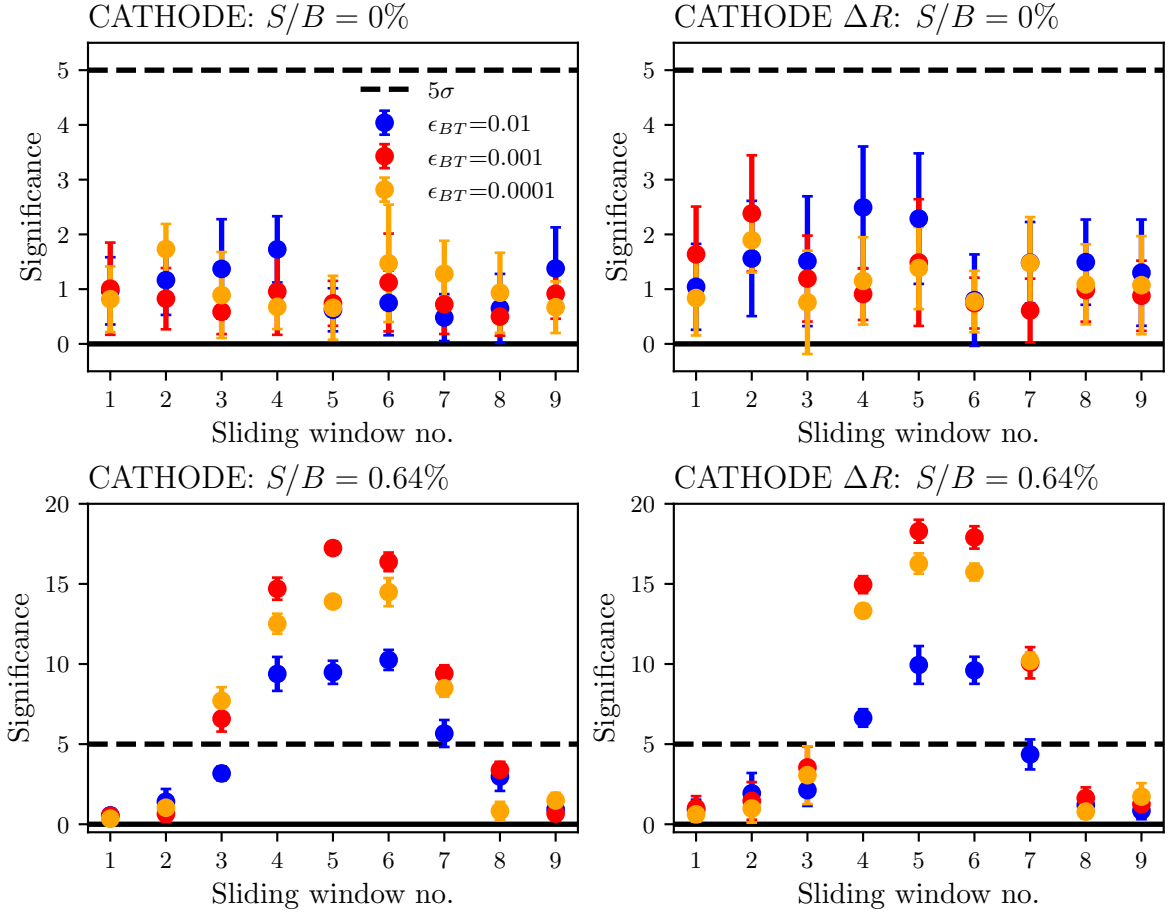


Figure B.4: Significance $\mathcal{S}$, Eq. 6.9, for the different signal regions for CATHODE without (top) and with signal injection (bottom) using the Baseline (left) and Baseline + ΔR (right) features using $\delta_{\text{sys}}^{\text{MC} \oplus \text{SB}}$ with $\sigma_{\text{sys}} = 0$. The error bars indicate the variance of the significance based on 10 DE and classifier runs.

1 for the Baseline + ΔR feature set. This may be caused by the trigger turn-on. Generally, we see no significant differences in the results without (top) and with signal (bottom) for the Baseline + ΔR features. The classifier is unable to detect the signal here due to the low-quality background template resulting from correlations between ΔR and the dijet mass. For the Baseline feature set with signal, we observe large significances of over 8σ in the central window for the thresholds $\epsilon_{BT} = 10^{-3}$ and 10^{-4} , representing a strong increase for $\epsilon_{BT} = 10^{-3}$. The change is minimal for 10^{-4} , for which $\delta_{\text{sys}}^{\text{MC}}$ is small and as such the statistical error dominates even with $\sigma_{\text{sys}} = \delta_{\text{sys}}$.

For CATHODE, we see in Fig. B.4 that significances for background-only studies (top) are increased, fluctuating between 1 and 3σ . Comparing with Fig. 6.7, significances with signal (bottom) are strongly increased, going up to 18σ in the central window for $\epsilon_{BT} = 10^{-3}$. The discovery threshold of 5σ is met in multiple windows across all thresholds. The stark increase from the values seen for CWoLa Hunting in Fig. B.3 can especially be found in the smaller statistical error $\sigma_{\text{exp,stat}}$ for CATHODE resulting from the oversampling. We can very clearly see the advantage of a more accurate estimation of σ_{sys} here.

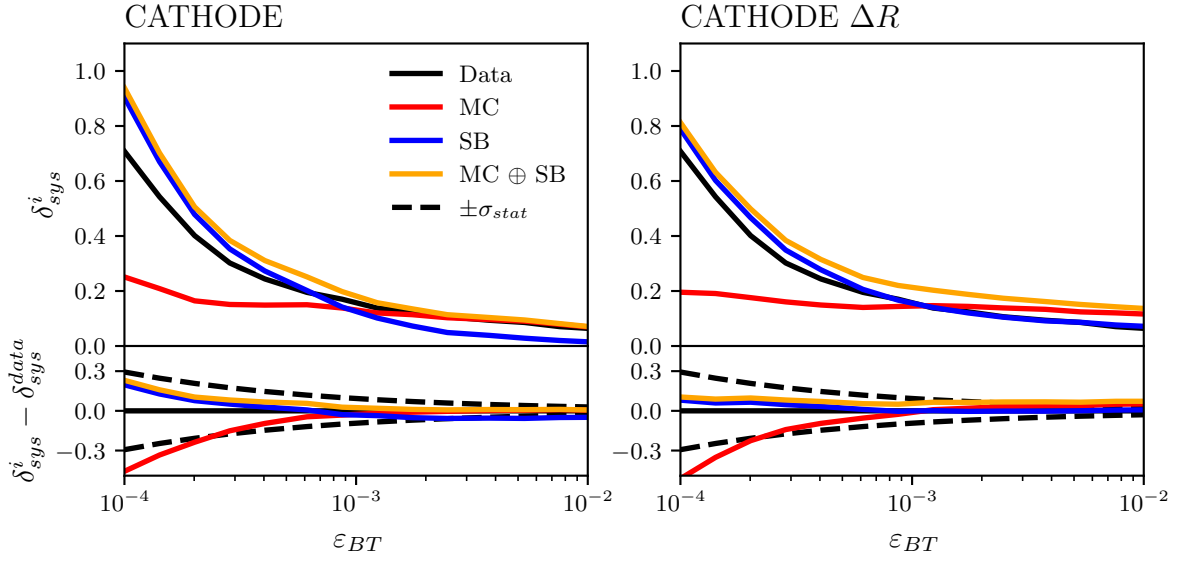


Figure B.5: Same as Fig. 6.5 but for the SB estimate of δ_{sys} , the data set contains signal ($S/B \approx 0.64\%$).

B.4 ESTIMATE OF δ_{sys} WITH SIGNAL FOR CATHODE

For CATHODE, we perform a data-driven estimate of δ_{sys} using the SB data. This estimate depends on whether signal is present in the data. In Sec. 6.1.5, we showed $\delta_{\text{sys}}^{\text{SB}}$ and $\delta_{\text{sys}}^{\text{MC} \oplus \text{SB}}$ for the background-only case. In Fig. B.5, we show how these estimates change when signal is included. For both feature sets, we can see that $\delta_{\text{sys}}^{\text{SB}}$ slightly overestimates $\delta_{\text{sys}}^{\text{data}}$ for small ϵ_{BT} . For the background-only case (see Fig. 6.5), both values match very well. This then also leads to a slight overestimation by $\delta_{\text{sys}}^{\text{MC} \oplus \text{SB}}$.

B.5 SIGNIFICANCES FOR CATHODE WHEN USING THE MC ESTIMATE

In Sec. 6.1, we use two different ways of estimating δ_{sys} for CWoLa Hunting and CATHODE, namely $\delta_{\text{sys}}^{\text{MC}}$ and $\delta_{\text{sys}}^{\text{MC} \oplus \text{SB}}$ respectively. We therefore take a look at results for CATHODE when using $\delta_{\text{sys}}^{\text{MC}}$. In this case, we underestimate δ_{sys} for small ϵ_{BT} (see Fig. 6.5). This can also be seen in the top panels of Fig. B.6, where we show the significances for both feature sets when no signal is injected. For $\epsilon_{\text{BT}} = 10^{-4}$, we see significances of over 2σ in window two. Here, the choice of $\sigma_{\text{sys}} = \delta_{\text{sys}}$ is crucial to cover data-MC mismodeling and avoid false discoveries. In the lower panels, we can see results with signal. For $\epsilon_{\text{BT}} = 10^{-2}$, the difference to the results of Fig. 6.7 are minimal as $\delta_{\text{sys}}^{\text{MC} \oplus \text{SB}}$ is dominated by $\delta_{\text{sys}}^{\text{MC}}$. For $\epsilon_{\text{BT}} = 10^{-3}$, we can see a slight increase in significance. For $\epsilon_{\text{BT}} = 10^{-4}$, the large underestimation of δ_{sys} leads to large significances up to approximately 13σ .

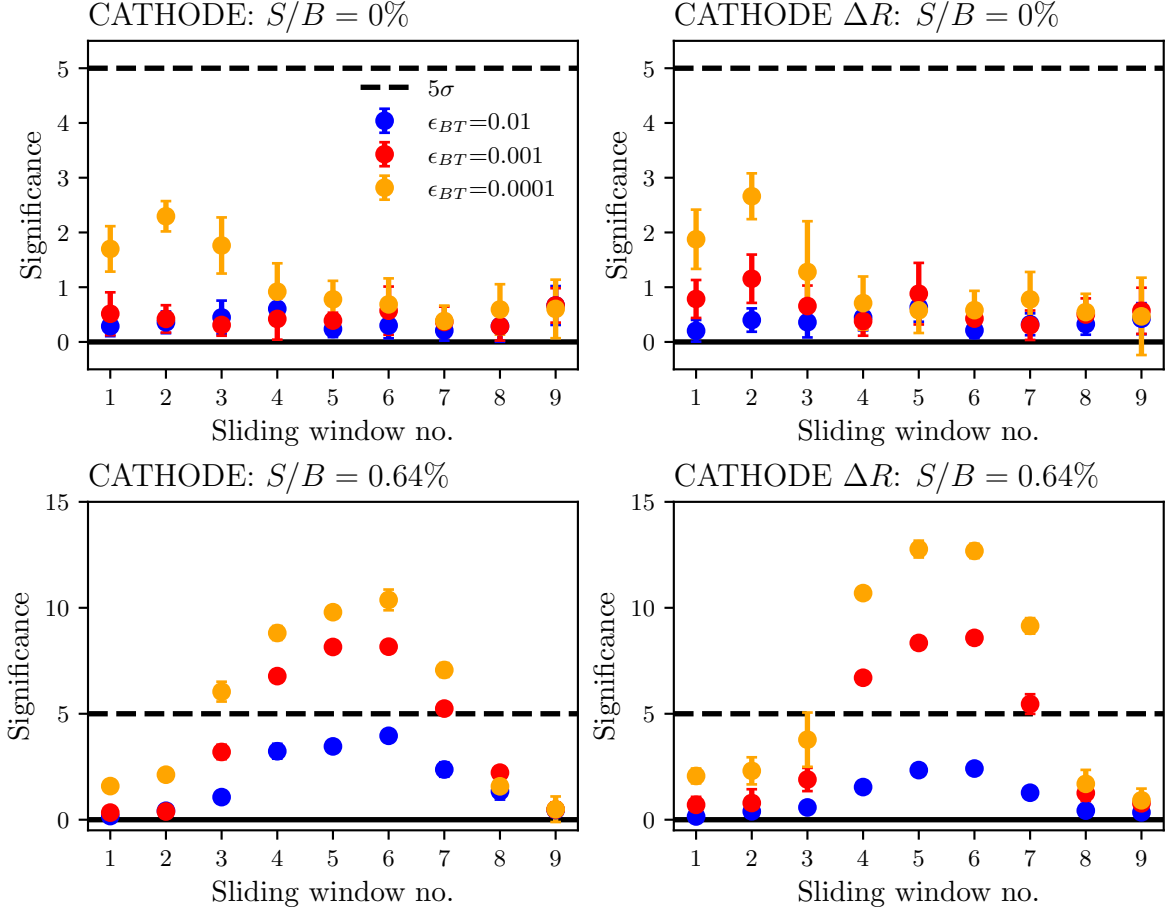


Figure B.6: Significance $\mathcal{S}$, Eq. 6.9, for the different signal regions for CATHODE without (top) and with signal injection (bottom) using the Baseline (left) and Baseline + ΔR (right) features using $\delta_{\text{sys}}^{\text{MC}}$. The error bars indicate the variance of the significance across 10 DE and classifier runs.

ACRONYMS

AD Anomaly Detection

ADABOOST Adaptive Boosting

ADAGRAD Adaptive Gradient

ADAM Adaptive Moment Estimation

ANODE Anomaly Detection with Density Estimation

ARGOS Above Random Gain of SIC

BCE Binary Crossentropy

BDT Boosted Decision Tree

BSM Beyond the Standard Model

BT Background Template

CATHODE Classifying Anomalies Through Outer Density Estimation

CDF cumulative distribution function

CURTAINS Constructing Unobserved Regions by Transforming Adjacent Intervals

CWoLA Classification Without Labels

DE Density Estimation

DL Deep Learning

DT Decision Tree

ECAL Electromagnetic Calorimeter

EFPP Energy Flow Polynomial

FETA Flow-Enhanced Transportation for Anomaly Detection

FPR False Positive Rate

GAN Generative Adversarial Network

HCAL Hadronic Calorimeter

HGB HistGradientBoostingClassifier

IAD Idealized Anomaly Detector

KDE Kernel Density Estimation

KL DIVERGENCE Kullback-Leibler divergence

LACATHODE Latent CATHODE

LHCO LHC Olympics

MADE Masked Autoencoder for Density Estimation

MAF Masked Autoregressive Flow

MC Monte Carlo

ML Machine Learning

MLP Multi-Layer Perceptron

NN Neural Network

PDF Parton Distribution Function

QCD Quantum Chromodynamics

QUAK Quasi Anomalous Knowledge

R-ANODE Residual ANODE

RAD-OT Resonant Anomaly Detection with Optimal Transport

REALNVP Real-Valued Non-Volume Preserving

RELU Rectified Linear Unit

RESNET Residual Network

RMSPROP Root Mean Squared Propagation

ROC Receiver Operating Characteristic

SALAD Simulation Assisted Likelihood-Free Anomaly Detection

SB Sideband

SIC Significance Improvement Characteristic

SIGMA Single Interpolated Model for Anomalies

SM Standard Model

SR Signal Region

SSB Short Sideband

SUSY Supersymmetry

TNT Tag ‘N’ Train

TPR True Positive Rate

TRANSIT Transport Adversarial Network for Smooth Interpolation

BIBLIOGRAPHY

- [1] T. Finke et al., “Back To The Roots: Tree-Based Algorithms for Weakly Supervised Anomaly Detection,” *Phys. Rev. D* **109**, 034033 (2023), [arXiv:2309.13111 \[hep-ph\]](#).
- [2] R. Das et al., “Accurate and robust methods for direct background estimation in resonant anomaly detection,” *Phys. Rev. D* **111**, 094041 (2025), [arXiv:2411.00085 \[hep-ph\]](#).
- [3] M. Hein et al., “How to pick the best anomaly detector?” (2025), [arXiv:2511.14832 \[hep-ph\]](#).
- [4] M. Hein, B. Nachman, and D. Shih, “Look everywhere effects in anomaly detection,” (2025), [arXiv:2512.13787 \[hep-ph\]](#).
- [5] R. Das et al., “Kitchen Sink Anomaly Detection,” (2026), [arXiv:2604.20965 \[hep-ph\]](#).
- [6] CMS Collaboration, “Observation of a New Boson at a Mass of 125 GeV with the CMS Experiment at the LHC,” *Phys. Lett. B* **716**, 30–61 (2012), [arXiv:1207.7235 \[hep-ex\]](#).
- [7] ATLAS Collaboration, “Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC,” *Phys. Lett. B* **716**, 1–29 (2012), [arXiv:1207.7214 \[hep-ex\]](#).
- [8] X. Fan et al., “Measurement of the Electron Magnetic Moment,” *Phys. Rev. Lett.* **130**, 071801 (2023), [arXiv:2209.13084 \[physics.atom-ph\]](#).
- [9] S. Navas et al. (Particle Data Group), “Review of particle physics,” *Phys. Rev. D* **110**, 030001 (2024).
- [10] ATLAS Collaboration, CMS Collaboration, “Highlights of the HL-LHC physics projections by ATLAS and CMS,” (2025), [arXiv:2504.00672 \[hep-ex\]](#).
- [11] E. Hammou et al., “Hide and seek: how PDFs can conceal new physics,” *JHEP* **11**, 090 (2023), [arXiv:2307.10370 \[hep-ph\]](#).
- [12] ATLAS Collaboration, “The ATLAS Simulation Infrastructure,” *Eur. Phys. J. C* **70**, 823–874 (2010), [arXiv:1005.4568 \[physics.ins-det\]](#).
- [13] A. Butter et al., “The Machine Learning Landscape of Top Taggers,” *SciPost Phys.* **7**, edited by G. Kasieczka and T. Plehn, 014 (2019), [arXiv:1902.09914 \[hep-ph\]](#).
- [14] T. Heimel et al., “QCD or What?” *SciPost Phys.* **6**, 030 (2019), [arXiv:1808.08979 \[hep-ph\]](#).
- [15] S. E. Park et al., “Quasi Anomalous Knowledge: Searching for new physics with embedded knowledge,” *JHEP* **06**, 030 (2020), [arXiv:2011.03550 \[hep-ph\]](#).
- [16] E. M. Metodiev, B. Nachman, and J. Thaler, “Classification without labels: Learning from mixed samples in high energy physics,” *JHEP* **10**, 174 (2017), [arXiv:1708.02949 \[hep-ph\]](#).
- [17] G. Kasieczka et al., “The LHC Olympics 2020: A Community Challenge for Anomaly Detection in High Energy Physics,” *Rept. Prog. Phys.* **84**, 124201 (2021), [arXiv:2101.08320 \[hep-ph\]](#).

- [18] G. Kasieczka, B. Nachman, and D. Shih, *R&D dataset for lhc olympics 2020 anomaly detection challenge*, <https://zenodo.org/record/6466204>, 2019.
- [19] L. Brennan et al., “Weakly supervised anomaly detection with event-level variables,” *Phys. Rev. D* **112**, 055040 (2025), [arXiv:2504.13249 \[hep-ph\]](#).
- [20] G. Bickendorf et al., “Combining Resonant and Tail-based Anomaly Detection,” *Phys. Rev. D* **109**, 096031 (2023), [arXiv:2309.12918 \[hep-ph\]](#).
- [21] T. Finke et al., “Boosting mono-jet searches with model-agnostic machine learning,” *JHEP* **08**, 015 (2022), [arXiv:2204.11889 \[hep-ph\]](#).
- [22] G. Kasieczka et al., “Complete Optimal Non-Resonant Anomaly Detection,” (2024), [arXiv:2404.07258 \[hep-ph\]](#).
- [23] M. Freytsis, M. Perelstein, and Y. C. San, “Anomaly Detection in Presence of Irrelevant Features,” *JHEP* **02**, 220 (2023), [arXiv:2310.13057 \[hep-ph\]](#).
- [24] J. Thaler and K. Van Tilburg, “Identifying Boosted Objects with N-subjettiness,” *JHEP* **03**, 015 (2011), [arXiv:1011.2268 \[hep-ph\]](#).
- [25] J. Thaler and K. Van Tilburg, “Maximizing Boosted Top Identification by Minimizing N-subjettiness,” *JHEP* **02**, 093 (2012), [arXiv:1108.2701 \[hep-ph\]](#).
- [26] A. Bhargava, T. Cai, and B. Nachman, “Optimal Transport Event Representation for Anomaly Detection,” (2025), [arXiv:2512.04839 \[hep-ph\]](#).
- [27] E. Buhmann et al., “Full Phase Space Resonant Anomaly Detection,” *Phys. Rev. D* **109**, 055015 (2023), [arXiv:2310.06897 \[hep-ph\]](#).
- [28] D. Sengupta et al., “Improving new physics searches with diffusion models for event observables and jet constituents,” *JHEP* **04**, 109 (2023), [arXiv:2312.10130 \[physics.data-an\]](#).
- [29] V. Mikuni and B. Nachman, “A Method to Simultaneously Facilitate All Jet Physics Tasks,” *Phys. Rev. D* **111**, 054015 (2025), [arXiv:2502.14652 \[hep-ph\]](#).
- [30] V. Bresò-Pla et al., “Explicit or Implicit? Encoding Physics at the Precision Frontier,” (2026), [arXiv:2603.08802 \[hep-ph\]](#).
- [31] J. H. Collins, K. Howe, and B. Nachman, “Anomaly Detection for Resonant New Physics with Machine Learning,” *Phys. Rev. Lett.* **121**, 241803 (2018), [arXiv:1805.02664 \[hep-ph\]](#).
- [32] J. H. Collins, K. Howe, and B. Nachman, “Extending the search for new resonances with machine learning,” *Phys. Rev. D* **99**, 014038 (2019), [arXiv:1902.02634 \[hep-ph\]](#).
- [33] B. Nachman and D. Shih, “Anomaly Detection with Density Estimation,” *Phys. Rev. D* **101**, 075042 (2020), [arXiv:2001.04990 \[hep-ph\]](#).
- [34] A. Andreassen, B. Nachman, and D. Shih, “Simulation Assisted Likelihood-free Anomaly Detection,” *Phys. Rev. D* **101**, 095004 (2020), [arXiv:2001.05001 \[hep-ph\]](#).
- [35] K. Benkendorfer, L. L. Pottier, and B. Nachman, “Simulation-Assisted Decorrelation for Resonant Anomaly Detection,” *Phys. Rev. D* **104**, 035003 (2020), [arXiv:2009.02205 \[hep-ph\]](#).
- [36] A. Hallin et al., “Classifying Anomalies THrough Outer Density Estimation (CATH-ODE),” *Phys. Rev. D* **106**, 055006 (2021), [arXiv:2109.00546 \[hep-ph\]](#).
- [37] J. A. Raine et al., “CURTAINS for your Sliding Window: Constructing Unobserved Regions by Transforming Adjacent Intervals,” *Front. Big Data* **6**, 899345 (2022), [arXiv:2203.09470 \[hep-ph\]](#).

- [38] A. Hallin et al., “Resonant anomaly detection without background sculpting,” *Phys. Rev. D* **107**, 114012 (2022), [arXiv:2210.14924 \[hep-ph\]](#).
- [39] T. Golling et al., “Flow-enhanced transportation for anomaly detection,” *Phys. Rev. D* **107**, 096025 (2023), [arXiv:2212.11285 \[hep-ph\]](#).
- [40] T. Golling et al., “The Interplay of Machine Learning-based Resonant Anomaly Detection Methods,” *Eur. Phys. J. C* **84**, 241 (2023), [arXiv:2307.11157 \[hep-ph\]](#).
- [41] R. Das and D. Shih, “SIGMA: Single Interpolated Generative Model for Anomalies,” *Phys. Rev. D* **112**, 074040 (2024), [arXiv:2410.20537 \[hep-ph\]](#).
- [42] M. Leigh et al., “Accelerating template generation in resonant anomaly detection searches with optimal transport,” *JHEP* **12**, 105 (2024), [arXiv:2407.19818 \[hep-ph\]](#).
- [43] I. Oleksiyuk, S. Voloshynovskiy, and T. Golling, “TRANSIT your events into a new mass: Fast background interpolation for weakly-supervised anomaly searches,” *JHEP* **07**, 177 (2025), [arXiv:2503.04342 \[hep-ph\]](#).
- [44] ATLAS Collaboration, “Dijet resonance search with weak supervision using $\sqrt{s} = 13$ TeV pp collisions in the ATLAS detector,” *Phys. Rev. Lett.* **125**, 131801 (2020), [arXiv:2005.02983 \[hep-ex\]](#).
- [45] ATLAS Collaboration, “Weakly supervised anomaly detection for resonant new physics in the dijet final state using proton-proton collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector,” *Phys. Rev. D* **112**, 072009 (2025), [arXiv:2502.09770 \[hep-ex\]](#).
- [46] CMS Collaboration, “Model-agnostic search for dijet resonances with anomalous jet substructure in proton-proton collisions at $\sqrt{s} = 13$ TeV,” *Rept. Prog. Phys.* **88**, 067802 (2024), [arXiv:2412.03747 \[hep-ex\]](#).
- [47] R. Gambhir et al., “Isolating Unisolated Upsilon’s with Anomaly Detection in CMS Open Data,” *Phys. Rev. Lett.* **135**, 021902 (2025), [arXiv:2502.14036 \[hep-ph\]](#).
- [48] M. Thomson, *Modern particle physics* (Cambridge University Press, New York, Oct. 2013).
- [49] D. Kar, *Experimental particle physics*, 2053-2563 (IOP Publishing, 2019).
- [50] I. Neutelings, *SM particles*, https://tikz.net/sm_particles/, Mar. 2024.
- [51] K. M. Black et al., “Muon Collider Forum report,” *JINST* **19**, T02015 (2024), [arXiv:2209.01318 \[hep-ex\]](#).
- [52] A. Abada et al. (FCC), “FCC-hh: The Hadron Collider: Future Circular Collider Conceptual Design Report Volume 3,” *Eur. Phys. J. ST* **228**, 755–1107 (2019).
- [53] R. D. Ball et al. (NNPDF), “Parton distributions from high-precision collider data,” *Eur. Phys. J. C* **77**, 663 (2017), [arXiv:1706.00428 \[hep-ph\]](#).
- [54] A. Buckley et al., “LHAPDF6: parton density access in the LHC precision era,” *Eur. Phys. J. C* **75**, 132 (2015), [arXiv:1412.7420 \[hep-ph\]](#).
- [55] B. Andersson et al., “Parton Fragmentation and String Dynamics,” *Phys. Rept.* **97**, 31–145 (1983).
- [56] J. Alwall et al., “The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations,” *JHEP* **07**, 079 (2014), [arXiv:1405.0301 \[hep-ph\]](#).
- [57] C. Bierlich et al., “A comprehensive guide to the physics and usage of PYTHIA 8.3,” *SciPost Phys. Codeb.* **2022**, 8 (2022), [arXiv:2203.11601 \[hep-ph\]](#).

- [58] M. Bahr et al., “Herwig++ Physics and Manual,” *Eur. Phys. J. C* **58**, 639–707 (2008), [arXiv:0803.0883 \[hep-ph\]](#).
- [59] J. Bellm et al., “Herwig 7.0/Herwig++ 3.0 release note,” *Eur. Phys. J. C* **76**, 196 (2016), [arXiv:1512.01178 \[hep-ph\]](#).
- [60] M. Selvaggi, “DELPHES 3: A modular framework for fast-simulation of generic collider experiments,” *J. Phys. Conf. Ser.* **523**, edited by J. Wang, 012033 (2014).
- [61] S. Agostinelli et al. (GEANT4), “GEANT4 - A Simulation Toolkit,” *Nucl. Instrum. Meth. A* **506**, 250–303 (2003).
- [62] J. Allison et al., “Recent developments in Geant4,” *Nucl. Instrum. Meth. A* **835**, 186–225 (2016).
- [63] F. Hariri et al. (Geant4), “Geant4 detector simulations for future HEP experiments,” *PoS ICHEP2018*, 268 (2019).
- [64] K. Rabbertz, *Jet Physics at the LHC: The Strong Force beyond the TeV Scale*, Vol. 268, Springer Tracts in Modern Physics (Springer, Berlin, 2017).
- [65] S. Marzani, G. Soyez, and M. Spannowsky, *Looking inside jets: an introduction to jet substructure and boosted-object phenomenology*, Vol. 958 (Springer, 2019), [arXiv:1901.10342 \[hep-ph\]](#).
- [66] M. Cacciari and G. P. Salam, “Dispelling the N^3 myth for the k_t jet-finder,” *Phys. Lett. B* **641**, 57–61 (2006), [arXiv:0512210 \[hep-ph\]](#).
- [67] M. Cacciari, G. P. Salam, and G. Soyez, “FastJet User Manual,” *Eur. Phys. J. C* **72**, 1896 (2012), [arXiv:1111.6097 \[hep-ph\]](#).
- [68] S. Catani et al., “Longitudinally invariant K_t clustering algorithms for hadron hadron collisions,” *Nucl. Phys. B* **406**, 187–224 (1993).
- [69] S. D. Ellis and D. E. Soper, “Successive combination jet algorithm for hadron collisions,” *Phys. Rev. D* **48**, 3160–3166 (1993), [arXiv:9305266 \[hep-ph\]](#).
- [70] Y. L. Dokshitzer et al., “Better jet clustering algorithms,” *JHEP* **08**, 001 (1997), [arXiv:9707323 \[hep-ph\]](#).
- [71] M. Cacciari, G. P. Salam, and G. Soyez, “The anti- k_t jet clustering algorithm,” *JHEP* **04**, 063 (2008), [arXiv:0802.1189 \[hep-ph\]](#).
- [72] S. Vent, R. Winterhalder, and T. Plehn, “The Physics Behind ML-based Quark-Gluon Taggers,” *SciPost Phys.* **20**, 084 (2026), [arXiv:2507.21214 \[hep-ph\]](#).
- [73] P. T. Komiske, E. M. Metodiev, and J. Thaler, “Energy flow polynomials: A complete linear basis for jet substructure,” *JHEP* **04**, 013 (2018), [arXiv:1712.07124 \[hep-ph\]](#).
- [74] H. Qu and L. Gouskos, “ParticleNet: Jet Tagging via Particle Clouds,” *Phys. Rev. D* **101**, 056019 (2020), [arXiv:1902.08570 \[hep-ph\]](#).
- [75] S. Gong et al., “An Efficient Lorentz Equivariant Graph Neural Network for Jet Tagging,” *JHEP* **07**, 030 (2022), [arXiv:2201.08187 \[hep-ph\]](#).
- [76] H. Qu, C. Li, and S. Qian, “Particle Transformer for Jet Tagging,” (2022), [arXiv:2202.03772 \[hep-ph\]](#).
- [77] J. Brehmer et al., “A Lorentz-Equivariant Transformer for All of the LHC,” *SciPost Phys.* **19**, 108 (2024), [arXiv:2411.00446 \[hep-ph\]](#).
- [78] J. Spinner et al., “Lorentz Local Canonicalization: How to Make Any Network Lorentz-Equivariant,” (2025), [arXiv:2505.20280 \[stat.ML\]](#).

- [79] N. Grimm, C. Bonvin, and I. Tutusaus, “Comparing the motion of dark matter and standard model particles on cosmological scales,” *Nature Commun.* **16**, 9399 (2025), [arXiv:2502.12843 \[astro-ph.CO\]](#).
- [80] S. Schramm, “Searching for New Physics in Hadronic Final States with Run 2 Proton–Proton Collision Data at the LHC,” *Symmetry* **14**, 1173 (2022), [arXiv:2311.16040 \[hep-ex\]](#).
- [81] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning / ian goodfellow, yoshua bengio and aaron courville*, eng, Adaptive computation and machine learning series (MIT Press, Cambridge, Massachusetts, 2016).
- [82] M. Erdmann et al., *Deep Learning for Physics Research* (Feb. 2021).
- [83] K. P. Murphy, *Probabilistic machine learning: an introduction* (MIT Press, 2022).
- [84] R. D. Ball et al. (PDF4LHC Working Group), “The PDF4LHC21 combination of global PDF fits for the LHC Run III,” *J. Phys. G* **49**, 080501 (2022), [arXiv:2203.05506 \[hep-ph\]](#).
- [85] R. D. Ball et al. (NNPDF), “An open-source machine learning framework for global analyses of parton distributions,” *Eur. Phys. J. C* **81**, 958 (2021), [arXiv:2109.02671 \[hep-ph\]](#).
- [86] scikit-learn developers, *Underfitting vs. overfitting*, https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html, Accessed: 2026-02-03.
- [87] T. Aarrestad et al., “The Dark Machines Anomaly Score Challenge: Benchmark Data and Model Independent Event Classification for the Large Hadron Collider,” *SciPost Phys.* **12**, 043 (2021), [arXiv:2105.14027 \[hep-ph\]](#).
- [88] T. Heime1 et al., “MadNIS – Neural Multi-Channel Importance Sampling,” *SciPost Phys.* **15**, 141 (2022), [arXiv:2212.06172 \[hep-ph\]](#).
- [89] T. Heime1 et al., “The MadNIS Reloaded,” *SciPost Phys.* **17**, 023 (2023), [arXiv:2311.01548 \[hep-ph\]](#).
- [90] O. Amram et al., “CaloChallenge 2022: a community challenge for fast calorimeter simulation,” *Rept. Prog. Phys.* **88**, edited by C. Krause et al., 116201 (2025), [arXiv:2410.21611 \[physics.ins-det\]](#).
- [91] C. E. Shannon, “A mathematical theory of communication,” *The Bell System Technical Journal* **27**, 379–423 (1948).
- [92] J. Neyman and E. S. Pearson, “On the Problem of the Most Efficient Tests of Statistical Hypotheses,” *Phil. Trans. Roy. Soc. Lond. A* **231**, 289–337 (1933).
- [93] L. Wasserman, *36-705 Intermediate Statistics: Lecture Notes*, Online lecture notes, Available at <https://www.stat.cmu.edu/~larry/=stat705/> (accessed February 10, 2026), Carnegie Mellon University, Department of Statistics & Data Science, 2025.
- [94] J. Geuskens et al., “The Fundamental Limit of Jet Tagging,” in , Vol. 112 (Nov. 2024), p. L091901, [arXiv:2411.02628 \[hep-ph\]](#).
- [95] I. Pang et al., “SURFing to the Fundamental Limit of Jet Tagging,” (2025), [arXiv:2511.15779 \[hep-ph\]](#).
- [96] I. Narsky and F. C. Porter, *Statistical analysis techniques in particle physics: Fits, density estimation and supervised learning* (Wiley-VCH, Weinheim, Germany, 2014).

- [97] B. M. Greenwell, *Tree-based methods: a practical introduction with applications in r*, eng, Chapman and Hall/CRC Data Science (CRC Press, Boca Raton, Florida, 2022).
- [98] H. Carlens, *State of machine learning competitions in 2024*, <https://mlcontests.com/state-of-machine-learning-competitions-2024>, 2025.
- [99] J. N. Morgan and J. A. Sonquist, “Problems in the analysis of survey data, and a proposal,” *Journal of the American Statistical Association* **58**, 415–434 (1963).
- [100] L. Breiman et al., “Classification and regression trees,” in (1984).
- [101] C. Gini, *Variabilità e mutabilità: contributo allo studio delle distribuzioni edelle relazioni statistiche* (Tipografia di Paolo Cuppini, Bologna, 1912).
- [102] L. Breiman, “Technical note: some properties of splitting criteria,” *Mach. Learn.* **24**, 41–47 (1996).
- [103] T. Chen and C. Guestrin, “XGBoost: a scalable tree boosting system,” in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, KDD ’16 (2016), pp. 785–794.
- [104] H. Carlens, *State of competitive machine learning in 2022*, <https://mlcontests.com/state-of-competitive-data-science-2022>.
- [105] G. Ke et al., “Lightgbm: a highly efficient gradient boosting decision tree,” *Advances in neural information processing systems* **30**, 3146–3154 (2017).
- [106] H. Shi, “Best-first decision tree learning,” MA thesis (The University of Waikato, Hamilton, New Zealand, 2007).
- [107] M. d. Condorcet, “Essay on the application of analysis to the probability of majority decisions,” Paris: Imprimerie Royale (1785).
- [108] R. E. Schapire, “The strength of weak learnability,” *Machine learning* **5**, 197–227 (1990).
- [109] H. Shteingart et al., “Majority voting and the condorcet’s jury theorem,” (2020), [arXiv:2002.03153 \[cs.LG\]](https://arxiv.org/abs/2002.03153).
- [110] L. Breiman, “Bagging predictors,” *Machine Learning* **24**, 123–140 (1996).
- [111] L. Breiman, “Random forests,” *Machine Learning* **45**, 5–32 (2001).
- [112] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of Computer and System Sciences* **55**, 119–139 (1997).
- [113] L. Mason et al., “Boosting algorithms as gradient descent,” *Advances in Neural Information Processing Systems* (1999).
- [114] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning*, Springer Series in Statistics (Springer New York Inc., New York, NY, USA, 2001).
- [115] J. H. Friedman, “Greedy function approximation: A gradient boosting machine.,” *The Annals of Statistics* **29**, 1189–1232 (2001).
- [116] L. Prokhorenkova et al., *Catboost: unbiased boosting with categorical features*, 2017, [arXiv:1706.09516 \[cs.LG\]](https://arxiv.org/abs/1706.09516).
- [117] F. Pedregosa et al., “Scikit-learn: machine learning in Python,” *Journal of Machine Learning Research* **12**, 2825–2830 (2011).
- [118] C. M. Bishop, *Pattern recognition and machine learning (information science and statistics)* (Springer-Verlag, Berlin, Heidelberg, 2006).

- [119] A. L. Maas, “Rectifier nonlinearities improve neural network acoustic models,” in (2013).
- [120] D. Hendrycks and K. Gimpel, *Gaussian error linear units (gelus)*, 2016, [arXiv:1606.08415 \[cs.LG\]](#).
- [121] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Networks* **2**, 359–366 (1989).
- [122] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Math. Control Signals Syst.* **2**, 303–314 (1989).
- [123] K. Hornik, “Approximation capabilities of multilayer feedforward networks,” *Neural Networks* **4**, 251–257 (1991).
- [124] M. Leshno et al., “Multilayer feedforward networks with a nonpolynomial activation function can approximate any function,” *Neural Networks* **6**, 861–867 (1993).
- [125] J. T. Håstad, *Computational limitations for small-depth circuits* (MIT Press, Cambridge, MA, USA, 1987).
- [126] G. Montúfar et al., “On the number of linear regions of deep neural networks,” (2014), [arXiv:1402.1869 \[stat.ML\]](#).
- [127] T. Poggio et al., “Why and when can deep – but not shallow – networks avoid the curse of dimensionality: a review,” (2016), [arXiv:1611.00740 \[cs.LG\]](#).
- [128] M. Raghu et al., “Transfusion: understanding transfer learning for medical imaging,” (2019), [arXiv:1902.07208 \[cs.CV\]](#).
- [129] H. Robbins and S. Monro, “A stochastic approximation method,” *Annals of Mathematical Statistics* **22**, 400–407 (1951).
- [130] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature* **323**, 533–536 (1986).
- [131] D. P. Kingma and J. Ba, “Adam: a method for stochastic optimization,” [arXiv:1412.6980 \[cs.LG\]](#).
- [132] G. Hinton, *Neural networks for machine learning, lecture 6: rmsprop*, Lecture slides, University of Toronto, http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf, 2012.
- [133] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” **12**, 2121–2159 (2011).
- [134] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, Vol. 9, edited by Y. W. Teh and M. Titterton, Proceedings of Machine Learning Research (May 2010), pp. 249–256.
- [135] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society (Series B)* **58**, 267–288 (1996).
- [136] A. E. Hoerl and R. W. Kennard, “Ridge regression: biased estimation for nonorthogonal problems,” *Technometrics* **12**, 55–67 (1970).
- [137] N. Srivastava et al., “Dropout: a simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research* **15**, 1929–1958 (2014).
- [138] G. E. Hinton et al., “Improving neural networks by preventing co-adaptation of feature detectors,” (2012), [arXiv:1207.0580 \[cs.NE\]](#).

- [139] A. Bogatskiy et al., “PELICAN: Permutation Equivariant and Lorentz Invariant or Covariant Aggregator Network for Particle Physics,” (2022), [arXiv:2211.00454 \[hep-ph\]](#).
- [140] J. Spinner et al., “Lorentz-Equivariant Geometric Algebra Transformers for High-Energy Physics,” (2024), [arXiv:2405.14806 \[physics.data-an\]](#).
- [141] K. P. Murphy and K. P. Murphy, *Probabilistic machine learning: advanced topics*, eng, Adaptive computation and machine learning (The MIT Press, Cambridge, Massachusetts, 2023).
- [142] D. J. Rezende and S. Mohamed, “Variational Inference with Normalizing Flows,” in (May 2015), [arXiv:1505.05770 \[stat.ML\]](#).
- [143] Y. Lipman et al., “Flow matching guide and code,” (2024), [arXiv:2412.06264 \[cs.LG\]](#).
- [144] I. J. Goodfellow et al., “Generative Adversarial Networks,” (2014), [arXiv:1406.2661 \[stat.ML\]](#).
- [145] Y. Lipman et al., “Flow matching for generative modeling,” (2022), [arXiv:2210.02747 \[cs.LG\]](#).
- [146] R. Bellman, *Dynamic Programming* (Dover Publications, 1957).
- [147] L. Dinh, J. Sohl-Dickstein, and S. Bengio, *Density estimation using real nvp*, 2016, [arXiv:1605.08803 \[cs.LG\]](#).
- [148] J. Sohl-Dickstein et al., “Deep unsupervised learning using nonequilibrium thermodynamics,” in Proceedings of the 32nd international conference on machine learning - volume 37, ICML’15 (2015), pp. 2256–2265.
- [149] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” in, NIPS ’20 (2020).
- [150] S. Kullback and R. A. Leibler, “On Information and Sufficiency,” *The Annals of Mathematical Statistics* **22**, 79–86 (1951).
- [151] L. Dinh, D. Krueger, and Y. Bengio, *Nice: non-linear independent components estimation*, 2014, [arXiv:1410.8516 \[cs.LG\]](#).
- [152] G. Papamakarios, T. Pavlakou, and I. Murray, “Masked Autoregressive Flow for Density Estimation,” (2017), [arXiv:1705.07057 \[stat.ML\]](#).
- [153] M. Germain et al., “MADE: Masked Autoencoder for Distribution Estimation,” (2015), [arXiv:1502.03509 \[cs.LG\]](#).
- [154] C. Durkan et al., “Neural Spline Flows,” in (June 2019), [arXiv:1906.04032 \[stat.ML\]](#).
- [155] R. T. Q. Chen et al., “Neural Ordinary Differential Equations,” (2018), [arXiv:1806.07366 \[cs.LG\]](#).
- [156] W. Grathwohl et al., “FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models,” (2018), [arXiv:1810.01367 \[cs.LG\]](#).
- [157] M. Sommerhalder, “Developing and applying machine learning techniques for model-agnostic searches for new physics at the lhc,” English, PhD thesis (University of Hamburg, Feb. 2025).
- [158] J. E. Augustin et al. (SLAC-SP-017), “Discovery of a Narrow Resonance in e^+e^- Annihilation,” *Phys. Rev. Lett.* **33**, 1406–1408 (1974).
- [159] J. J. Aubert et al. (E598), “Experimental Observation of a Heavy Particle J ,” *Phys. Rev. Lett.* **33**, 1404–1406 (1974).

- [160] G. Breit and E. Wigner, “Capture of Slow Neutrons,” *Phys. Rev.* **49**, 519–531 (1936).
- [161] ATLAS Collaboration, “Search for new phenomena in dijet mass and angular distributions from pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector,” *Phys. Lett. B* **754**, 302–322 (2016), [arXiv:1512.01530 \[hep-ex\]](#).
- [162] CMS Collaboration, “Search for high mass dijet resonances with a new background prediction method in proton-proton collisions at $\sqrt{s} = 13$ TeV,” *JHEP* **05**, 033 (2020), [arXiv:1911.03947 \[hep-ex\]](#).
- [163] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation forest,” in *2008 eighth ieee international conference on data mining* (2008), pp. 413–422.
- [164] G. E. Hinton and R. S. Zemel, “Autoencoders, minimum description length and helmholtz free energy,” in *Proceedings of the 7th international conference on neural information processing systems, NIPS’93* (1993), pp. 3–10.
- [165] T. Finke et al., “Autoencoders for unsupervised anomaly detection in high energy physics,” *JHEP* **06**, 161 (2021), [arXiv:2104.09051 \[physics.data-an\]](#).
- [166] R. Das, G. Kasieczka, and D. Shih, “Residual ANODE,” (2023), [arXiv:2312.11629 \[hep-ph\]](#).
- [167] HEP ML Community, *A Living Review of Machine Learning for Particle Physics*.
- [168] O. Amram and C. M. Suarez, “Tag N’ Train: A Technique to Train Improved Classifiers on Unlabeled Data,” *10.1007/JHEP01(2021)153* (2020), [arXiv:2002.12376 \[hep-ph\]](#).
- [169] D. Sengupta et al., “CURTAINS Flows For Flows: Constructing Unobserved Regions with Maximum Likelihood Estimation,” *SciPost Phys.* **17**, 046 (2023), [arXiv:2305.04646 \[hep-ph\]](#).
- [170] D. Shih, *Additional qcd background events for lhco2020 r&d (signal region only)*, <https://zenodo.org/record/5759086>, 2021.
- [171] G. Kasieczka, B. Nachman, and D. Shih, *Official datasets for lhc olympics 2020 anomaly detection challenge* (2019).
- [172] D. Shih et al., “Via Machinae: Searching for Stellar Streams using Unsupervised Machine Learning,” *Mon. Not. Roy. Astron. Soc.* **509**, 5992 (2021), [arXiv:2104.12789 \[astro-ph.GA\]](#).
- [173] D. Sengupta et al., “skycurtains: model-agnostic search for stellar streams with Gaia data,” *Mon. Not. Roy. Astron. Soc.* **536**, 1104–1114 (2024), [arXiv:2405.12131 \[astro-ph.GA\]](#).
- [174] A. Hallin et al., “Via Machinae 3.0: A search for stellar streams in Gaia with the CATHODE algorithm,” (2025), [arXiv:2509.08064 \[astro-ph.GA\]](#).
- [175] V. Mikuni and B. Nachman, “OmniLearn: A Method to Simultaneously Facilitate All Jet Physics Tasks,” *Phys. Rev. D* **111**, L051504 (2024), [arXiv:2404.16091 \[hep-ph\]](#).
- [176] G. Cowan et al., “Asymptotic formulae for likelihood-based tests of new physics,” *Eur. Phys. J. C* **71**, [Erratum: *Eur.Phys.J.C* 73, 2501 (2013)], 1554 (2011), [arXiv:1007.1727 \[physics.data-an\]](#).
- [177] M. Hein and T. Quadfasel, *Tree-Based Algorithms for Weakly Supervised Anomaly Detection*, https://github.com/uhh-pd-ml/treebased_anomaly_detection, 2023.
- [178] L. Grinsztajn, E. Oyallon, and G. Varoquaux, “Why do tree-based models still outperform deep learning on tabular data?” [arXiv:2207.08815 \[cs.LG\]](#).

- [179] Martín Abadi et al., *TensorFlow: large-scale machine learning on heterogeneous systems*, Software available from <https://www.tensorflow.org/>, 2015.
- [180] N. Deimann, “Weakly supervised anomaly detection in collider physics for large data sets,” MA thesis (RWTH Aachen University, 2026).
- [181] M. Hein, *ARGOS metric for anomaly detection*, <https://github.com/mariehein/ARGOS-metric-for-anomaly-detection>, 2025.
- [182] CMS Collaboration, “Machine-learning techniques for model-independent searches in dijet final states,” [10.5281/zenodo.16656501](https://arxiv.org/abs/2512.20395) (2025), [arXiv:2512.20395 \[hep-ex\]](https://arxiv.org/abs/2512.20395).
- [183] R. Das, *Density Estimation for “Accurate and robust methods for direct background estimation in resonant anomaly detection”*, https://github.com/rd804/cut_and_count_FM, 2024.
- [184] O. Amram, J. Birk, and M. Sommerhalder, *Sk_cathode*, https://github.com/uhh-pd-ml/sk_cathode, 2024.
- [185] M. Hein, *Code for “Accurate and robust methods for direct background estimation in resonant anomaly detection”*, <https://doi.org/10.5281/zenodo.15066482>, Mar. 2025.
- [186] M. Hein, “Using density estimation for resonance searches at the lhc,” MA thesis (RWTH Aachen University, 2022).
- [187] M. Faselow, “Resonant anomaly detection beyond the standard model: statistical analysis for direct background estimation to improve sensitivity,” MA thesis (RWTH Aachen University, 2025).
- [188] J. Geuskens, “Identifying anomalous events using low-level lhc data,” MA thesis (RWTH Aachen University, 2023).
- [189] M. Hein, *Look everywhere effects in anomaly detection*, https://github.com/mariehein/look_everywhere_in_AD, 2025.
- [190] CMS Collaboration, “MUSiC: a model-unspecific search for new physics in proton–proton collisions at $\sqrt{s} = 13$ TeV,” *Eur. Phys. J. C* **81**, 629 (2021), [arXiv:2010.02984 \[hep-ex\]](https://arxiv.org/abs/2010.02984).
- [191] ATLAS Collaboration, “A general search for new phenomena with the ATLAS detector in pp collisions at $\sqrt{s} = 8$ TeV,” *ATLAS-CONF-2014-006* (2014).
- [192] A. Gelman et al., *Bayesian data analysis*, Third, Chapman & Hall/CRC Texts in Statistical Science Series (CRC, Boca Raton, Florida, 2013).
- [193] C. Bonferroni, “Teoria sttistica delle classi e calcolo delle probabilità,” *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze* **8** (1936).
- [194] O. J. Dunn, “Multiple Comparisons among Means,” *J. Am. Statist. Assoc.* **56**, 52–64 (1961).
- [195] L. Wasserman, *All of statistics: a concise course in statistical inference* (Springer Publishing Company, Incorporated, 2010).
- [196] K. He et al., “Deep Residual Learning for Image Recognition,” [10.1109/CVPR.2016.90](https://arxiv.org/abs/1512.03385) (2015), [arXiv:1512.03385 \[cs.CV\]](https://arxiv.org/abs/1512.03385).
- [197] C. Durkan et al., *Nflows: normalizing flows in pytorch*, version v0.14, Nov. 2020.

- [198] J. Ansel et al., “PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation,” in *29th acm international conference on architectural support for programming languages and operating systems, volume 2 (asplos '24)* (Apr. 2024).
- [199] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” (2017), [arXiv:1711.05101 \[cs.LG\]](#).
- [200] F. Rozet et al., *Zuko: normalizing flows in pytorch*, 2022.
- [201] J. Zhuang et al., “Adaptive checkpoint adjoint method for gradient estimation in neural ODE,” in *Proceedings of the 37th international conference on machine learning*, Vol. 119, edited by H. D. III and A. Singh, Proceedings of Machine Learning Research (July 2020), pp. 11639–11649.