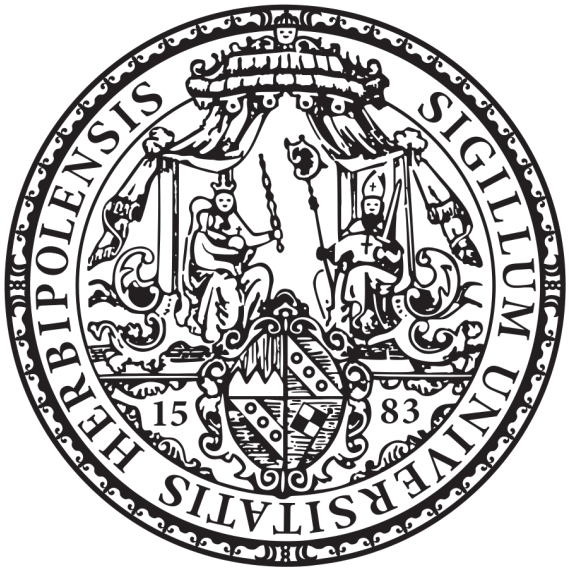
Machine learning methods for (semi-)hadronic final states at the LHC

Masterarbeit



Julius-Maximilians-Universität Würzburg
Fakultät für Physik und Astronomie
Studienfach: Physik
Betreuer: Prof. Dr. Werner Porod

vorgelegt von
Manuel Schmidt
Würzburg, 15.07.2024

Summary

Composite Higgs models are a theoretically well motivated extension of the standard model. The therein contained additional pseudo Nambu-Goldstone bosons are potentially accessible to searches at the high luminosity Large Hadron Collider. A characteristic process is the decay of doubly charged scalars to a $W^+t\bar{b}$ ($W^-\bar{t}b$) system. We focus on a final state containing many jets and two same-sign isolated leptons and propose various deep neural networks to discriminate this signal process from relevant standard model backgrounds.

For this, we simulate a dataset containing three representations of the collision events. Firstly, we calculate a set of high-level kinematic data and use a multilayer perceptron on it to classify signal events. Furthermore, we target a jet image representation of the events with a convolutional neural network based on ResNet. Finally, we construct graphs on a point cloud representation of the event. We then use a graph attention network for this representation. In each case, we beat our benchmarks, which are models that have proven to be successful on comparable tasks in established literature.

We also show how important it is to combine the different representations, as they highlight different physical aspects of the event. Adding high-level information to the jet images as new channels corresponding to jet-center positions improves the performance of the convolutional neural networks significantly. Moreover, by constructing classifiers that combine pretrained models that work on each representation into a single classifier that can access all representations simultaneously, we further enhance the discriminatory power of the neural networks. This combined classifier achieves the best performance and leads to an expected exclusion limit of 710 GeV and a discovery reach of 550 GeV for the doubly charged scalar production.

Zusammenfassung

Composite Higgs Modelle sind theoretisch fundierte Erweiterungen des Standardmodells. Die darin enthaltenen doppelt geladenen Skalarteilchen und deren Paarproduktion können voraussichtlich am high-luminosity Large Hadron Collider sondiert werden. Ein typischer Prozess ist der Zerfall der doppelt geladenen Skalare in ein $W^+t\bar{b}$ ($W^-\bar{t}b$) System. Dabei fokussieren wir uns auf einen Endzustand, der viele Jets und zwei isolierte Leptonen mit selber Ladung enthält. Um diesen Signalprozess von den relevanten Standardmodell-Hintergrundprozessen zu unterscheiden, schlagen wir vor, verschiedene neuronale Netzwerke zu verwenden.
Dazu simulieren wir einen Datensatz, der drei Repräsentationen der Kollisionsereignisse enthält. Wir verwenden ein Multi-Layer-Perceptron, das auf einem Satz an kinematischen Daten arbeitet. Des Weiteren implementieren wir Modelle, die auf ResNet basieren, und bezwecken damit eine Klassifizierung von Jet-Bildern. Letztlich konstruieren wir Graphen auf einer Teilchenwolken-Darstellung und klassifizieren diese mit Graph-Attention-Networks. In jedem Fall übertreffen wir unsere Vergleichsmodelle, die sich in etablierter Literatur als erfolgreich erwiesen haben.
Darüber hinaus zeigen wir, dass es wichtig ist, die unterschiedlichen Repräsentationen zu verbinden, da diese Informationen über unterschiedliche physikalische Aspekte des Kollisionsereignisses beinhalten. Die Leistung der Convolutional Neural Networks wird durch das Hinzufügen von Jet-Mittelpunktpositionen zu den Jet-Bildern deutlich verbessert. Des Weiteren kann ein Klassifikator konstruiert werden, der alle Repräsentationen des Kollisionsevents verbindet. Dazu werden vortrainierte Netzwerke, die auf den einzelnen Darstellungen arbeiten, verbunden und verbessern die Fähigkeit, Signal von Hintergrund zu trennen, weiter. Dieser kombinierte Klassifikator erreicht ein erwartetes Ausschlusslimit von 710 GeV und kann die Paarproduktion des doppelt geladenen Skalarteilchens bis zu 550 GeV entdecken.

Contents

1. Introduction

The standard model (SM) of particle physics has proven to be an exceptionally accurate description of nature, with many of its predictions tested to high precision by collider experiments over the past decades. The discovery of the Higgs boson by the ATLAS and CMS collaborations at the Large Hadron Collider (LHC) in 2012 [1, 2] has completed the discovery of all particles predicted by the standard model.
Despite its successes, not all aspects of particle physics phenomenology are explained by the standard model. It does not account for neutrino masses, dark matter or matter-antimatter asymmetry, nor does it give a full quantized description of gravity. Furthermore, the standard model has several theoretical challenges, such as the strong charge parity (CP) problem, the hierarchy among fermion masses, and the naturalness problem associated with the Higgs mass.
Given these limitations, the standard model is considered an effective field theory that describes nature only up to a certain energy scale Λ_{SM}, beyond which new physics could emerge. These unresolved issues and theoretical shortcomings motivate model building efforts that construct theories beyond the standard model.
One such model class are the composite Higgs models [3–6]. In composite Higgs models, the Higgs boson is not a fundamental scalar but a pseudo Nambu-Goldstone boson (pNGB), arising from the spontaneous breaking of a global symmetry in a new strongly coupled sector at the TeV scale. This is comparable to how the hadronic bound states in quantum chromodynamics (QCD) arise. This expanded sector involves new fermions, known as hyperquarks, interacting via a new gauge group. A key advantage of composite Higgs models is their potential to solve the naturalness problem. The relatively small mass of the Higgs compared to other states in these models is naturally explained by its pNGB nature.
In addition to the standard model Higgs boson, these models predict other scalar particles and fermionic resonances that can mix with standard model particles like the top quark. This mixing, termed partial compositeness [7], can explain the relatively high mass of the top quark. The search for ultraviolet completions of these composite sectors has identified several minimal models [8–10], including one based on the symmetry breaking pattern

$$\mathcal{G} = \text{SU}(5) \times \text{SU}(6) \times \text{U}(1) \to \text{SO}(5) \times \text{Sp}(6) = \mathcal{H}, \tag{1.1}$$

which was referred to as M5 in [10].
This model and the therein contained additional pNGBs motivate this thesis. It should be noted, that the doubly charged scalars are not only present in model M5,

but rather all models with the SU(5)/SO(5) electroweak (EW) coset. Making this analysis interesting for the phenomenology of many composite Higgs models.
Their LHC phenomenology has been studied, e.g. in [10–15].
A characteristic prediction of this model are doubly charged scalar pNGBs $S^{\pm\pm}$, which are common to all composite Higgs models, which contain the $\frac{\mathrm{SU}(5)}{\mathrm{SO}(5)}$ electroweak coset. The final states produced by their Drell-Yan production are rather busy, as they contain multiple jets. The three-body decay of those scalars via the decay chain $S^{++} \to W^+ t\bar{b}$ ($S^{--} \to W^- \bar{t}b$) is the main focus of this work. To reduce the standard model background, we focus on the di-leptonic decay of the W^+t ($W^-\bar{t}$) system, originating from one of the doubly charged scalars. This leads to a final state with two same-sign isolated leptons and high hadronic activity, resulting from many jets, some of which are b-tagged. We expand on the work done in [15] by using deep neural networks to target those final states. The neural networks function as classifiers of our signal process and the relevant standard model background processes.
We use the b-tagging information of the jets to further enrich the information contained in the jet images and apply advanced convolutional neural networks (CNNs) to the classification task. The application of computer vision inspired neural networks on jet images has proven to be a powerful tool in particle physics, e.g. in jet-tagging and jet-substructure analysis [16–24]. We compare the performance of different CNN architectures for the identification of our signal process.
Furthermore, we implement neural networks operating on graph representations of the LHC events. Networks based on the message-passing algorithm have also been successfully used in jet-physics [25, 26]. We use a point cloud representation of the LHC event inspired by [27] and compare different graph neural networks (GNNs).
The calculation of higher-level kinematic features of the events, allows us to use multilayer perceptrons and Transformers to target this third event representation.
Finally, we combine the best architectures for each representation to produce a model that has access to all three representations of the events. We show that this multi representation classifier provides significant increases in discriminatory power over the classifiers acting on a single representation.
We organize this thesis as follows: In section 2 we give a short review of composite Higgs models and explain the origin of the doubly charged scalars. Section 3 gives an introduction to deep learning and neural networks. To train the various networks, we have simulated a dataset containing signal and background LHC events. We describe the simulation process and data structure in section 4. Section 5 contains the

description of the neural networks that we compare in this thesis, and the results of this comparison are shown in section 6. Finally, we summarize and give an outlook on future research possibilities in section 7.

2. Composite Higgs models

Firstly, we give a brief review of the composite Higgs models. We start with the general motivation of composite Higgs models, followed by their general construction and classification. Finally, we introduce the concrete model that motivates this thesis.

2.1. The naturalness problem

The Higgs boson in the Standard Model is a scalar boson characterized by a mass of $m_H = 125$ GeV. It triggers the electroweak symmetry breaking and endows other elementary particles with mass through its non-vanishing vacuum expectation value (VEV) of $v = 246$ GeV. However, the incorporation of the Higgs boson into the standard model gives rise to the naturalness problem. Following the arguments in [28], the naturalness problem emerges due to the consideration of higher-order corrections to the bare mass of the Higgs boson $m_{h,0}$. For the Higgs boson, one would anticipate its mass to be on the order of the scale Λ^2_{SM}, where the Standard Model breaks down as a fundamental theory. This scale could be taken as the scale of a grand unified theory (GUT) $\Lambda_{\text{GUT}} \sim 10^{15}$ GeV or the Planck scale $\Lambda_{\text{Planck}} \sim 10^{19}$ GeV as in [28]. Explicitly, for a scalar Higgs boson, the effective squared Higgs mass m_h^2 can be expressed as

$$m_h^2 = m_{h,0}^2 + c^2 \Lambda^2_{\text{SM}}, \tag{2.1}$$

where c^2 represents a factor stemming from loop corrections. The measured Higgs boson mass being of the same order as the electroweak symmetry breaking scale poses a problem, since a cancellation of numerous orders of magnitude must occur to ensure the Higgs boson's relatively light mass. This situation gives rise to the naturalness problem, as the extensive fine-tuning required to maintain this alignment is considered inherently unnatural.

Other than the Higgs boson, there are no other elementary scalar particles. However, there are a plethora of non-elementary scalar particles, namely the QCD condensate mesons such as the pions. It is therefore reasonable to ask whether the Higgs boson could also be a non-elementary, i.e. composite particle. If this were the case the naturalness problem could be solved as any corrections to the effective mass of the Higgs boson would be cut off at the condensation scale of the composite Higgs and therefore could be orders of magnitudes closer to the electroweak symmetry breaking scale Λ_{EW}. One would expect this new scale to be at the TeV level, since one would

expect a plethora of new composite particles at this scale. Such particles have not been observed by current experiments, and therefore the condensation scale must lie outside of the grasp of contemporary experiments. Above this scale, one would have a new QCD like theory of a set of so called hyperquarks. Below this scale, the composite Higgs would function as the standard model Higgs boson. Therefore, preserving the standard model physics and solving the naturalness problem.

2.2. Composite Higgs scenario

The modern variants of the composite Higgs models [3, 4, 6, 29] rely on the breaking of a new global symmetry group $\mathcal{G}$ to a subgroup $\mathcal{H} \subset \mathcal{G}$ with the coset $\mathcal{G}/\mathcal{H}$ containing pseudo Nambu-Goldstone bosons (pNGBs), one of which must be the Higgs boson.

The spontaneous breaking of $\mathcal{G}$ generates a number of massless Goldstone bosons according to the Goldstone theorem [30, 31]. Those acquire a mass through the explicit breaking of $\mathcal{G}$ by an interaction $\mathcal{L}_{\text{int}}$ with the elementary standard model sector turning the Goldstone bosons into pNGBs and therefore allowing them to potentially contain the Higgs boson among them. Since the standard model Higgs boson is rather light with a mass of the order of $\Lambda_{\text{EW}} = v$ it is necessary for the interaction $\mathcal{L}_{\text{int}}$ to be a small perturbation of the so-called composite sector associated with the global group $\mathcal{G}$. In order to be compatible with the standard model, the subgroup $\mathcal{H}$ must be large enough to contain the electroweak part of the standard model symmetries $\mathcal{H} \supset \mathrm{SU}(2)_L \times \mathrm{U}(1)_Y$ and the coset $\mathcal{G}/\mathcal{H}$ must be able to contain the Higgs doublet. In fact, it is common for many explicit composite Higgs models that even the custodial $\mathrm{SU}(2)_{\mathrm{L}} \times \mathrm{SU}(2)_{\mathrm{R}}$ is part of the breaking pattern:

$$\mathcal{H} \supset \mathrm{SU}(2)_{\mathrm{L}} \times \mathrm{SU}(2)_{\mathrm{R}} \supset \mathrm{SU}(2)_L \times \mathrm{U}(1)_Y. \tag{2.2}$$

This allows for a convenient implementation of $\rho = m_W^2/(c_w m_Z)^2 \sim 1$, since otherwise dimension six operators would introduce tree level corrections to the value of ρ that depend on the details of the composite scenario.

This restricts the possible symmetry breaking patterns and leads to the most minimal pattern being SO(5)/SO(4), since $\mathrm{SO}(4) \simeq \mathrm{SU}(2) \times \mathrm{SU}(2)$. This so-called minimal composite Higgs model (MCHM) has, however, been severely constrained by experiments. Furthermore, it does not contain any possible top-partners that are an important part of modern composite Higgs models, as we will discuss later.

The spontaneous symmetry breaking is implemented by assuming that the vacuum

state $\mathbf{F}$ is only invariant under the subgroup $\mathcal{H}$

$$\hat{T}^{\hat{a}}\mathbf{F} \neq 0, \tag{2.3}$$

where $\hat{T}^{\hat{a}}$ are the broken generators of $\mathcal{G}$, whilst the unbroken generators T^a preserve the vacuum

$$T^a\mathbf{F} = 0. \tag{2.4}$$

It should be noted that this is merely a choice of a particular embedding of a reference system in the algebra of $\mathcal{G}$. All embeddings are equal, since in the context of the global group $\mathcal{G}$ there is no preferred direction. However, this choice is particularly convenient.

In accordance with the electroweak symmetry-breaking mechanism in the standard model and, in more general terms, the Callan-Coleman-Wess-Zumino (CCWZ) mechanism that will be discussed later, the Goldstone Fields can be written as

$$\mathbf{\Phi} = e^{i\theta^{\hat{a}}(x)\hat{T}^{\hat{a}}}\mathbf{F}. \tag{2.5}$$

In the case of the MCHM with the breaking pattern SO(5)/SO(4) this leads to exactly four Goldsotne bosons which can form the complex Higgs doublet as required by the standard model.

Due to the explicit symmetry breaking introduced by the interaction between the composite and standard model sectors, the Goldstone bosons θ acquire a vacuum expectation value $\langle\theta\rangle$.

This can be interpreted as a misalignment of the real vacuum to the reference vacuum $\mathbf{F}$ by the angle $\langle\theta\rangle$. This leads to the relation $v = f\sin(\langle\theta\rangle)$, where $f = |\mathbf{F}|$ is the scale where $\mathcal{G}$ spontaneously breaks to $\mathcal{H}$. This can be geometrically represented as in Figure 1. If there is no misalignment and thus $f = v$, the class of models are called technicolor models [33–35]. Those models have phenomenological problems and have been replaced by models with misalignment. If there is misalignment, which is indicated by the important ξ-parameter

$$\xi = \frac{v^2}{f^2} \ll 1, \tag{2.6}$$

there arise interesting differences to the technicolor models. This is the class of models which motivated this thesis.

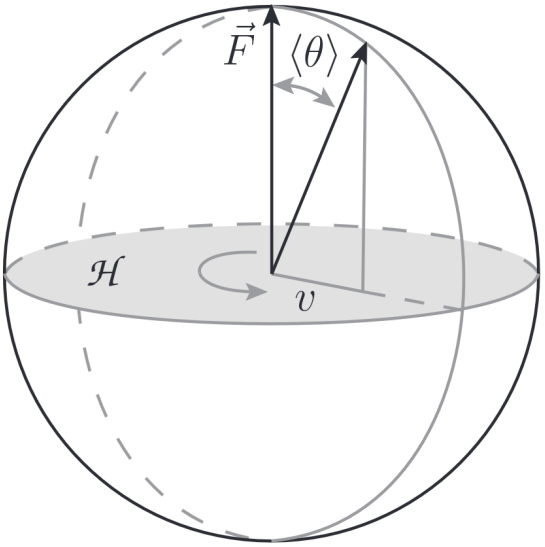


Figure 1: Depiction of vacuum misalignment on the SO(3) group manifold. The SO(3) symmetry is broken to a SO(2) symmetry by vacuum misalignment. The breaking is proportional to the projection of the reference vacuum state **F** to the SO(2)-plane. Taken from [32].

2.3. CCWZ construction

To construct an arbitrary composite Higgs theory there exists a construction formalism known as the Callan-Colemann-Wess-Zumino (CCWZ) formalism [36, 37]. The idea is to use the general transformation behavior under the global group $\mathcal{G}$ and the subgroup $\mathcal{H}$ respectively to construct minimal building blocks for the composite theory and finally arrive at a low-energy effective Lagrangian. However, this prescription does not include potential anomaly induced terms like the Adler-Bell-Jackiw anomaly [38, 39] that can also be present in composite Higgs theories. However, it is possible to construct those terms with the Wess-Zumino-Witten action [40, 41]. Such terms and their constructions will however not be discussed in detail in the scope of this thesis.

In the following we give a quick summary of the CCWZ formalism which follows the depictions in [32].

Any general group transformation g_α (with the whole set of generators T^A) can be decomposed into a transformation acting with the broken generators $\hat{T}^{\hat{a}}$ and the unbroken generators T^a:

$$g_\alpha = e^{i\alpha_A T^A} = e^{i\alpha_{\hat{a}}\hat{T}^{\hat{a}}} e^{i\alpha_a T^a}. \tag{2.7}$$

If one studies the transformation behavior of the so-called Goldstone matrix

$$U(\Pi) = e^{i\frac{\sqrt{2}}{f}\Pi_{\hat{a}}\hat{T}^{\hat{a}}}, \tag{2.8}$$

which is a fundamental object in composite Higgs theories and coincides with the Goldstone Fields in eq. (2.5) with a slight reparametrization, one finds

$$U(\Pi) \to U(\Pi^{(g)}) = gU(\Pi)h^{-1}(\Pi, g). \tag{2.9}$$

Here the decomposition of the general group element was used to define $\Pi^{(g)}$ implicitly and h is an element of the subgroup $\mathcal{H}$.
To arrive at the fundamental building blocks of the theory, namely the so called e- and d-symbols, it is useful to introduce the Maurer-Cartan Form:

$$iU(\Pi)^{-1}\partial_\mu U(\Pi) = d_{\mu,\hat{a}}(\Pi)\hat{T}^{\hat{a}} + e_{\mu,a}(\Pi)T^a = d_\mu + e_\mu, \tag{2.10}$$

which separates in broken and unbroken components. It follows from eq. (2.9), that the $d-$ and $e-$symbols transform as

$$d_\mu \to h(\Pi, g)d_\mu h^{-1}(\Pi, g) \tag{2.11}$$

$$e_\mu \to h(\Pi, g)\left(e_\mu + i\partial_\mu\right)h^{-1}(\Pi, g). \tag{2.12}$$

Since those transformation behaviors are only dependent on the subgroup element h, the CCWZ construction allows us to construct the Lagrangian using only $\mathcal{H}$-invariant terms and guarantees the global invariance under $\mathcal{G}$ automatically.
The perhaps simplest term is the contraction of two of the $d-$symbols

$$\mathcal{L} \supset \frac{f^2}{4}\,\mathrm{Tr}(d_\mu d^\mu) \tag{2.13}$$

$$= \frac{1}{2}\partial_\mu\Pi_{\hat{a}}\partial^\mu\Pi^{\hat{a}} + \sum_n \mathcal{O}\left((\partial\Pi)^2 \cdot \frac{\Pi^n}{f^n}\right), \tag{2.14}$$

which gives the kinematics of the Goldstone fields. Up to this point, we have only looked at global transformation of $\mathcal{G}$ and $\mathcal{H}$. To introduce the standard model to our composite Higgs theory, it is necessary to gauge at least a subgroup of $\mathcal{H}$. To do this, one can introduce a gauge field for all the $\mathcal{G}$ generators and set all the unphysical fields to zero at the end of the procedure. Hence, one finds the usual transformation behavior of the gauge fields with now local transformations $g(x)$:

$$T_A A_\mu^A \to (T_A A_\mu^A)^{(g)} = g(x)(T_A A_\mu^A + i\partial_\mu)g(x)^{-1}. \tag{2.15}$$

From this point on, for the sake of brevity, the contraction $T_A A_\mu^A$ is abbreviated as A_μ.
Following the example of the Maurer-Cartan Form, we can split the result of the action of the Goldstone Matrix $U(\Pi)$ on the gauge fields into broken and unbroken terms and introduce generalized $d-$ and $e-$symbols, that contain the gauge fields as follows:

$$\bar{A}_\mu = U(\Pi)^{-1}(A_\mu + i\partial_\mu)U(\Pi) = d_\mu(\Pi, A) + e_\mu(\Pi, A) \tag{2.16}$$

The generalized symbols $d_\mu(\Pi, A)$ and $e_\mu(\Pi, A)$ now transform in accordance with eq.(2.12) and can be used to construct a $\mathcal{H}$-invariant Lagrangian. Applying this procedure to the kinematic term (2.14), one obtains the kinematic term of the Goldstone fields as well as their gauge interactions via minimal substitution:

$$\mathcal{L} \supset \frac{f^2}{4} \operatorname{Tr}\left(d_\mu(\Pi, A) d^\mu(\Pi, A)\right) = \frac{1}{2}\left(D_\mu \Pi_{\hat{a}}\right)^\dagger \left(D^\mu \Pi^{\hat{a}}\right) + \ldots \tag{2.17}$$

2.4. Top partners

A ubiquitous aspect of modern composite Higgs models is partial compositeness. Partial compositeness solves the problem of the unexpectedly high top-quark mass in comparison to the other much lighter quarks, by introducing a so called top-partner. This is a composite particle that has the same standard model quantum numbers as the top-quark and therefore produces a mixed state with the elementary top-quark. The mass of the top-quark is then partially generated by the composite state, and the fundamental top-quark could be much lighter. It should be noted that this procedure can be done for all quark generations, however it is of explicit interest for the top-quark and hence the following discussion will focus on it.
The mixing is realized by introducing a linear coupling of the standard model quarks q_L and t_R to condensate operators $\mathcal{O}_F^{L/R}$, leading to the interaction

$$\mathcal{L}_{\text{int}} = -\lambda_L \bar{q}_L \mathcal{O}_F^L - \lambda_R \bar{t}_R \mathcal{O}_F^R + \text{h.c.}, \tag{2.18}$$

as first developed in [7]. This is done in contrast to the technicolor approach, where a bilinear operator is introduced to the Lagrangian

$$\mathcal{L}_{\text{int}} = \frac{\lambda_t}{\Lambda_{\text{UV}}^{d-1}} \bar{q}_L \mathcal{O}_S^c t_r + \frac{\lambda_b}{\Lambda_{\text{UV}}^{d-1}} \bar{q}_L \mathcal{O}_S b_R + \text{h.c.} \quad , \tag{2.19}$$

which can lead to the reintroduction of the naturalness problem. A further advantage of the partial compositeness way is that it gives the possibility of explaining the flavor

hierarchies by analyzing the dimensionality of the Yukawa-like operators and their scaling dimensions. Further details are discussed in [32].
Since the operators $\mathcal{O}_F^{L/F}$ are expected to be able to excite a single particle state from the vacuum they are associated with the fermionic resonances Q and T with a mass near the typical scale of the composite resonances m^* the effective mass Lagrangian can be estimated to be

$$\mathcal{L}_{mass} = -m^*\bar{Q}Q - m^*\bar{T}T - \frac{m^*}{g_*}\left(\lambda_L \bar{q}_L Q + \lambda_r \bar{t}_r T + \text{h.c.}\right). \tag{2.20}$$

2.5. Characterization of composite Higgs models

Since the MCHM is severely constrained and does not include top-partners in the form presented here, the question arises: what other (minimal) composite Higgs models are possible. We follow the characterization of composite Higgs models as is done in [42, 43].
In general, it is possible to have multiple sets of hyperquarks that are in different representations of the hyperquark gauge group G_{HC}. One usually splits the hyperquarks into one set that interacts with the strong sector of the standard model and one that interacts with the electroweak sector. The strong sector hyperquarks would then contribute to the construction of top-partners. Since the idea is to construct a minimal scenario, the assumption is that there is only one set of hyperquarks per sector. The hyperquarks associated with the electroweak sector shall be called Ψ and the ones associated with the strong sector χ. The minimal breaking patterns of the electroweak sector must be able to contain a $(\mathbf{2},\mathbf{2})$ representation of the custodial $\text{SU}(2)\times\text{SU}(2)$ symmetry that contains the complex standard model Higgs doublet. The strong sector needs to be able to contain the standard model QCD sector. Additionally, they carry hypercharge, and thus we require $H_\chi \supset \text{SU}(3)_c \times \text{U}(1)_X$ for the strong sector.
The global symmetry group is in general

$$U(n_\Psi) \times U(n_\chi) \cong \text{SU}(n_\Psi) \times \text{SU}(n_\chi) \times \text{U}(1)_\Psi \times \text{U}(1)_\chi, \tag{2.21}$$

where two additional U(1) arise. They form one linear combination that is anomaly free, while the other linear combination is responsible for the Adler-Bell-Jackiw anomaly. The anomaly free $\text{U}(1)_u$, contributes to the standard model hypercharge and is spontaneously broken, resulting in a standard model neutral pNGB a. This pNGB does not depend on the choice of the electroweak and strong sector groups, and is present in all minimal composite Higgs models. It has been studied in [10,

	$\Psi \in \mathrm{R}$	$\Psi \in \mathrm{PR}$	$\Psi, \tilde{\Psi} \in \mathrm{C}$
$\chi \in \mathrm{R}$	$\frac{\mathrm{SU}(5)}{\mathrm{SO}(5)}\,\frac{\mathrm{SU}(6)}{\mathrm{SO}(6)}$	$\frac{\mathrm{SU}(4)}{\mathrm{Sp}(4)}\,\frac{\mathrm{SU}(6)}{\mathrm{SO}(6)}$	$\frac{\mathrm{SU}(4)\times\mathrm{SU}(4)'}{\mathrm{SU}(4)_D}\,\frac{\mathrm{SU}(6)}{\mathrm{SO}(6)}$
$\chi \in \mathrm{PR}$	$\frac{\mathrm{SU}(5)}{\mathrm{SO}(5)}\,\frac{\mathrm{SU}(6)}{\mathrm{Sp}(6)}$	~~$\frac{\mathrm{SU}(4)}{\mathrm{Sp}(4)}\,\frac{\mathrm{SU}(6)}{\mathrm{Sp}(6)}$~~	~~$\frac{\mathrm{SU}(4)\times\mathrm{SU}(4)'}{\mathrm{SU}(4)_D}\,\frac{\mathrm{SU}(6)}{\mathrm{Sp}(6)}$~~
$\chi, \tilde{\chi} \in C$	$\frac{\mathrm{SU}(5)}{\mathrm{SO}(5)}\,\frac{\mathrm{SU}(3)\times\mathrm{SU}(3)'}{\mathrm{SU}(3)_D}$	~~$\frac{\mathrm{SU}(4)}{\mathrm{Sp}(4)}\,\frac{\mathrm{SU}(3)\times\mathrm{SU}(3)'}{\mathrm{SU}(3)_D}$~~	$\frac{\mathrm{SU}(4)\times\mathrm{SU}(4)'}{\mathrm{SU}(4)_D}\,\frac{\mathrm{SU}(3)\times\mathrm{SU}(3)'}{\mathrm{SU}(3)_D}$

Table 1: The minimal cosets that can be formed. The left and right cosets denote the electroweak and strong sector representations of the hyperquarks respectively. The two distinct representations of the hyperquark gauge group G_{HC} can be real (R), pseudo real (PR) or complex (C). The crossed-out combinations cannot form top-partners.

44, 45].

The anomaly free global symmetry group then is

$$\mathcal{G} = \mathrm{SU}(n_\Psi) \times \mathrm{SU}(n_\chi) \times \mathrm{U}(1)_u. \tag{2.22}$$

For the electroweak part of the composite Higgs model that leads to the possible minimal cosets of SU(5)/SO(5) for real representations, SU(4)/Sp(4) for pseudo-real representations, and $\mathrm{SU}(4) \times \mathrm{SU}(4)'/\mathrm{SU}(4)_D$ for complex representations. For the strong sector, this leads to the cosets SU(6)/SO(6) for real representations, SU(6)/Sp(6) for pseudo-real representations, and $\mathrm{SU}(3) \times \mathrm{SU}(3)'/\mathrm{SU}(3)_D$ for complex representations. Hence, we arrive at the minimal set of models that we summarize in Table 1, where the spontaneously broken and anomaly-free $\mathrm{U}(1)_u$ is not listed but present for all coset pairs.

In [44] it has been shown that the $Zb\bar{b}$ coupling can become a problem for the composite Higgs models with top-partners, as the new particles can modify this coupling via strong couplings to the top-quark. The $Zb\bar{b}$ coupling can however be protected by requiring that the top-partners transform equivalently under $\mathrm{SU}(2)_R$ and $\mathrm{SU}(2)_L$ by imposing a parity symmetry P_{LR} [46]. This leads to the classification of composite Higgs models that follows [9, 10] and is shown in Table 2.

Model Name	Gauge group G_{HC}	Ψ		χ	
M1	SO(7)	$5 \times \mathbf{F}$	R	$6 \times \mathbf{Spin}$	R
M2	SO(9)	$5 \times \mathbf{F}$	R	$6 \times \mathbf{Spin}$	R
M3	SO(7)	$5 \times \mathbf{Spin}$	R	$6 \times \mathbf{F}$	R
M4	SO(9)	$5 \times \mathbf{Spin}$	R	$6 \times \mathbf{F}$	R
M5	Sp(4)	$5 \times \mathbf{A}_2$	R	$6 \times \mathbf{F}$	PR
M6	SU(4)	$5 \times \mathbf{A}_2$	R	$3 \times (\mathbf{F}, \overline{\mathbf{F}})$	C
M7	SO(10)	$4 \times \mathbf{F}$	R	$4 \times (\mathbf{F}, \overline{\mathbf{F}})$	C
M8	Sp(4)	$4 \times \mathbf{F}$	PR	$6 \times \mathbf{A}_2$	R
M9	SO(11)	$4 \times \mathbf{Spin}$	PR	$6 \times \mathbf{F}$	R
M10	SO(10)	$4 \times \mathbf{Spin}, \overline{\mathbf{Spin}}$	C	$6 \times \mathbf{F}$	R
M11	SU(4)	$4 \times (\mathbf{F}, \overline{\mathbf{F}})$	C	$6 \times \mathbf{A}_2$	R
M12	SU(5)	$4 \times (\mathbf{F}, \overline{\mathbf{F}})$	C	$3 \times (\mathbf{A}_2, \overline{\mathbf{A}}_2)$	C

Table 2: The composite Higgs scenarios that fulfill the above-mentioned criteria as characterized in [9, 10]. The number of hyperfermions Ψ (electroweak sector) and χ (strong sector) and their representation are denoted by $\mathbf{F}$ for the fundamental, $\mathbf{A}_2$ for the two-index antisymmetric, and **Spin** for the spinorial.

2.6. The composite Higgs model M5

Most of the composite Higgs models characterized in Table 2 contain the doubly charged scalars we are interested in, namely all models containing the electroweak coset SU(5)/SO(5) (M1-M7).

In the following, we focus on the model M5, the study of which has motivated this thesis. We limit ourselves to a brief description of the electroweak sector of this model and refer to [45] for more details on the strong sector, and only show where the pNGBs that are of interest for this thesis originate from. For more details on the electroweak sector, we refer to [11]. We follow the introduction of the model M5 as is done in [42, 43].

The electroweak pNBGs originate from the breaking of the global SU(5) to SO(5), where the hyperquarks Ψ are gauged via an additional gauge group $G_{\text{HC}} = \text{Sp}(4)$

	Global symmetry			**SM gauge group**		
	SU(5)	SU(6)	U(1)	$SU(3)_C$	$SU(2)_L$	$U(1)_Y$
$\Psi_{1,2}$	**5**	**1**	$-\frac{\mathbf{3q_\chi}}{\mathbf{5}}$	**1**	**2**	$\frac{\mathbf{1}}{\mathbf{2}}$
$\Psi_{3,4}$	**5**	**1**	$-\frac{\mathbf{3q_\chi}}{\mathbf{5}}$	**1**	**2**	$-\frac{\mathbf{1}}{\mathbf{2}}$
Ψ_5	**5**	**1**	$-\frac{\mathbf{3q_\chi}}{\mathbf{5}}$	**1**	**1**	**0**
$\chi_{1,2,3}$	**1**	**6**	$\mathbf{q_\chi}$	**3**	**1**	**x**
$\chi_{4,5,6}$	**1**	**6**	$\mathbf{q_\chi}$	$\bar{\mathbf{3}}$	**1**	$\mathbf{-x}$

Table 3: Quantum numbers of the five electroweak sector hyperquarks Ψ in the two index antisymmetric representation $\mathbf{A_2}$ and the six strong sector hyperquarks χ in the fundamental representation $\mathbf{F}$. The U(1) charged is only determined up to a factor q_χ.

and form condensates below the composite Higgs scale. The model contains five hyperquarks Ψ in the electroweak sector and six hyperquarks χ in the strong sector. Table 3 summarizes their quantum numbers and representations under the global groups of the strong and electroweak sector as well as the standard model quantum numbers.

The electroweak pNGBs live in the coset SU(5)/SO(5). The SU(5) and SO(5) have 24 and 10 generators resepctivly; hence, there are 14 broken generators and, following that, 14 pNGBs. The pNGBs form a **14** of SO(5), which decomposes according to the breaking pattern discussed above:

$$\mathrm{SU}(5) \to \mathrm{SO}(5) \supset \mathrm{SU}(2)_L \times \mathrm{SU}(2)_R \supset \mathrm{SU}(2)_L \times \mathrm{U}(1)_Y. \tag{2.23}$$

Concretely, they decompose as

$$\mathbf{14} \to (\mathbf{1},\mathbf{1}) + (\mathbf{2},\mathbf{2}) + (\mathbf{3},\mathbf{3}) \tag{2.24}$$

$$\to \mathbf{1}_0 + \mathbf{2}_{\frac{1}{2}} + \mathbf{2}_{-\frac{1}{2}} + \mathbf{3}_1 + \mathbf{3}_0 + \mathbf{3}_{-1}, \tag{2.25}$$

where, for example, in the case of $\mathbf{3}_{-1}$ the index is the standard model hypercharge and the **3** represents a $SU(2)_L$ triplet.

We identify these mutiplets as sets of new particles as well as the standard model Higgs boson:

$$\eta + H + \tilde{H} + \pi_+ + \pi_0 + \pi_- \equiv \mathbf{1}_0 + \mathbf{2}_{\frac{1}{2}} + \mathbf{2}_{-\frac{1}{2}} + \mathbf{3}_1 + \mathbf{3}_0 + \mathbf{3}_{-1}. \tag{2.26}$$

Here the H is the standard model complex Higgs doublet, and the $\tilde{H}$ its dual. The $\eta \in \mathbf{1}_0$ is a singlet with respect to the standard model, and the (π_+, π_0, π_-) are the triplet components of the custodial $\mathrm{SU}(2)_L \times \mathrm{SU}(2)_R$ bitriplet $(\mathbf{3}, \mathbf{3})$. Finally, the Higgs VEV breaks the $\mathrm{SU}(2) \times \mathrm{SU}(2)$ to the diagonal $\mathrm{SU}(2)_D$, under which the pNGBs decompose as

$$\eta \equiv \mathbf{1}_0 \to \mathbf{1} \equiv \eta, \tag{2.27}$$

$$H + \tilde{H} \equiv \mathbf{2}_{\frac{1}{2}} + \mathbf{2}_{-\frac{1}{2}} \to \mathbf{1} + \mathbf{3} \equiv h + \phi, \tag{2.28}$$

$$\pi_- + \pi_0 + \pi_+ \equiv \mathbf{3}_1 + \mathbf{3}_0 + \mathbf{3}_{-1} \to \mathbf{5} + \mathbf{3} + \mathbf{1} \equiv \eta_5 + \eta_3 + \eta_1, \tag{2.29}$$

where the complex Higgs doublet decomposes into the three would be Goldstone bosons ϕ, that form the longitudinal degrees of freedom of the heavy vector bosons and the Higgs boson h analogously to the standard model. The η remains a standard model neutral singlet and the bitriplet decomposes into a singlet, triplet and quintuplet. The η_5 quintuplet contains the scalar pNGBs that motivate this thesis. Namely, it contains the doubly charged scalars η_5^{++}, that constitute the signal process for the machine learning task in this thesis. The decay chain of those scalars is model dependent and characteristic for the composite Higgs models with the $\frac{\mathrm{SU}(5)}{\mathrm{SO}(5)}$ electroweak coset. Whilst the name η_5 is common for those pNGBs, we will henceforth refer to them as $S^{\pm\pm}$ in accordance with [15], on which the work in this thesis is based.

In addition to the pNGBs there also arise other resonances, among them the fermionic resonances of the form $\chi\Psi\chi$ that act as the top partners of the model M5. Those resonances will, however, not be discussed further in this work.

3. Introduction to deep learning

In this chapter, we review the basics of machine learning, and in particular, of neural networks. First, we explain the general idea of machine learning using the example of a so-called multilayer perceptron (MLP). After this, we explain the training process and describe some common components of neural networks.

Machine learning is based on learning abstract information from data. Here, the way information is extracted from the data is not predetermined but rather learned iteratively. This allows the machine learning algorithms to find interdependencies and features that are not immediately obvious. This behavior is then used to make predictions about new data. For example, one could predict the membership to a set of classes or a value associated with the input. The former is called a classification task, whereas the latter is called a regression task.

One differentiates between the approaches of so called unsupervised and supervised learning. The latter relies on a dataset that is already labeled, meaning that for any given data point, the class or value of the corresponding data point is already known. In contrast, in unsupervised learning, one would, for example, try to determine the set of distinct classes by abstracting from the dataset. All applications that are discussed in this thesis fall into the category of supervised learning for classification tasks. Concretely, classifying a collision event at the high luminosity LHC as either a background or a signal event. In our scenario, a signal event would be an event that contains new particles from the composite Higgs sector. For this task, we study the application of various neural networks.

In general, a neural network is a universal function approximator, meaning that the network tries to approximate the output of a function $\hat{f}(x)$ given the input vector x, where $\hat{f}\colon\ \mathbb{R}^n \to \mathbb{R}^m,\ x \to \hat{f}(x)$. We call the approximating function f and the abstract target function $\hat{f}$. In the case of a classification task, one often talks about predicting labels instead of approximating a function, nonetheless the neural network learns a function that predicts the label.

For such a classification task, the produced output of the network $f(x)$ is a vector of the predicted scores, indicating the membership of the input to a certain class. In this case, every component of the vector f corresponds to a different class, and the value of the component f_j is the predicted score assigned to the input x for class j. To interpret the output of the neural network as an assigned probability, the output needs to be transformed, so that the sum of the output vector components is one and all entries are positive. For this, the softmax function is used. The softmax

function $\sigma(z)$ is an n dimensional generalization of the logistic function and reads

$$\sigma_i = \frac{\exp z_i}{\sum_j^n \exp z_j}, \tag{3.1}$$

where z is the neural network output of dimension n and $\sigma, z \in \mathbb{R}^n$ as well as $i, j \in \{1, ..., n\}$. The class with the highest predicted probability is then chosen as the predicted label y given the correct label $\hat{y}$. In the case of a prediction task, the output of the neural network does not necessarily need to be preceded by the softmax function.

The behavior as a universal function approximator is useful when the computation of the function $\hat{f}(x)$ is computationally expensive or the function itself is unknown. The latter is prominently the case in computer vision, where one, for example, tries to label an image as belonging to multiple classes (e.g. dog or cat). For this to be possible, it is important that the data used for training is already labeled, meaning that such tasks also fall in the realm of supervised learning and are somewhat similar to the problems tackled here. In fact, many techniques commonly used in computer vision are applied in this thesis.

One of the earliest types of neural networks is the multilayer perceptron (MLP), where all the layers are fully connected, as is visualized in Figure 2. There, each neuron of the previous layer is connected to all neurons of the next. For this reason, they are often simply called fully connected networks. MLPs are also responsible for giving neural networks their name, as their structure can be compared to the structure of interconnected neurons in the human brain.

It consists of an input layer, one or more hidden layers, and the output layer. The input layer transforms the input into a potentially higher-dimensional embedding, which is then further transformed by the hidden layers, and finally an output is produced by the output layer that projects the higher-dimensional embedding to the output dimension, which can, for example, be the number of classes. It should be noted that the hidden layers can have varying dimensionality. In principle, each layer could have an arbitrary hidden dimension.

Mathematically, this connectivity is realized by matrix multiplication. Each of the connections corresponds to an element of a matrix called the weight matrix $W \in \mathbb{R}^{M \times N}$. The components W_{ij}, with $i \in \{1, ..., M\}$ and $j \in \{1, ..., N\}$ are the learned parameters. Input, output, and the internal state of the network after each layer are represented by vectors of varying dimensionality, meaning the weight matrices W can have varying dimensions M and N. In addition to the weight matrices, a vector with learned components can be added to the state after the multiplication

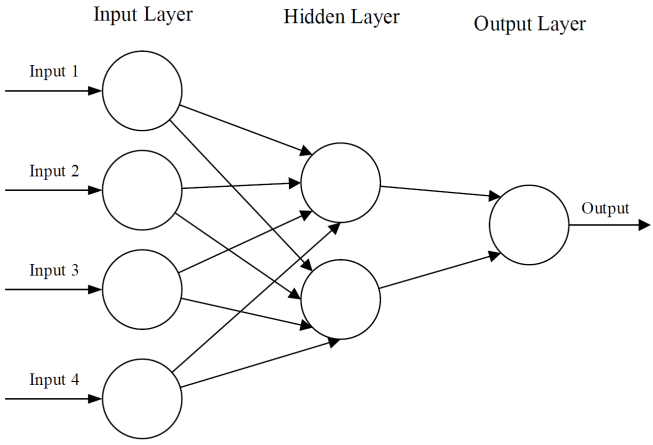


Figure 2: Visualization of a simple MLP. The neurons in the hidden layer are fully connected with the input and output layer. Taken from [47].

with the weight matrix. This vector is called a bias $b \in \mathbb{R}^M$. Together, the weights and biases of one layer form an arbitrary linear transformation of the input to the layer.

A simple series of such layers would be limited to linear functions in their expressiveness. Therefore, it is necessary to introduce nonlinearities into the model. For this purpose, the internal state vector is passed through a nonlinear function after each layer. This nonlinear function is called the activation function (a hint at neuron activation). The most common activation function is the rectified linear unit (ReLU) [48]:

$$\mathrm{ReLU}(x) = \max(0, x), \tag{3.2}$$

which is applied element-wise.

However, many other functions can be used, like the above-mentioned softmax or the $\tanh(x)$, with x being the input to the activation function, which can also be the hidden state h^l of the neural network after one or multiple hidden layers l. We denote an arbitrary activation function as σ.

The goal of the network is to approximate a function by learning from the data. To achieve this, there needs to be a metric by which the network is judged. This metric is the so-called loss function, sometimes also called the cost function. This metric could be something like the root mean squared error (RMSE) for a prediction task:

$$\mathrm{RMSE}\left(f(x), \hat{f}(x)\right) = \sqrt{\frac{1}{N}\sum_{i}^{N}\left(f(x_i)^2 - \hat{f}(x_i)^2\right)}, \tag{3.3}$$

where the error is calculated for a dataset of N data points and the $f(x_i)$ are the predicted (scalar) values for the $i-$th datapoint.
For the classification task with two classes, discussed in this thesis, the cross-entropy loss is used. The cross-entropy is used to compare two probability distributions and measures their dissimilarity. In the case of machine learning, the probability distribution estimated by the neural network is compared to the underlying ground truth, meaning the target distribution of the task at hand. A lower cross-entropy means the approximated probability distribution of the networks coincides better with the ground truth. The cross-entropy loss $\mathrm{CE}(p, \hat{y})$ of a binary classification task is calculated as

$$\mathrm{CE}(p, \hat{y}) = -\left(\hat{y} \ln(p) + (1 - \hat{y}) \ln(1 - p)\right), \tag{3.4}$$

where $\hat{y}$ is the binary ground truth label and p the predicted probability of the input belonging to that label. It should be noted here that the predicted probability and not the predicted label is used to calculate the cross-entropy.
This can also be generalized for multiple probabilities corresponding to multiple classes, where $\hat{y}$ and p become vectors. In the case of M distinct classes, the cross-entropy of a predicted probability distribution $p \in \mathbb{R}^M$ becomes

$$\mathrm{CE}(p, \hat{y}) = -\sum_{i}^{M} \hat{y}_i \log(p_i), \tag{3.5}$$

where $\hat{y}_i = 1$ indicates that the data-point belongs to this class and $\hat{y}_i = 0$ indicates that it does not, with $\hat{y} \in \mathbb{R}^M$. One can also see that the binary case in eq. (3.4) follows from eq. (3.5).
Since p is the predicted probability, there should always be a softmax proceeding the calculation of the cross-entropy. Therefore, all outputs of the models discussed in this thesis are the logits and the final softmax is always applied while calculating the cross-entropy.
In conclusion, the output $f(x)$ of an MLP with L layers given the input vector x can be written as a composition of linear transformations, realized by the weight matrices W^l and the biases b^l, and nonlinear activation functions σ^l of fitting dimension:

$$f(x) = \sigma^L\left(W^L \sigma^{L-1}\left(W^{L-1} ... \sigma^1\left(W^1 x + b^1\right) ... + b^{L-1}\right) + b^L\right), \tag{3.6}$$

where $l \in \{1, ..., L\}$ denotes the layer and each $W^l \in \mathbb{R}^{M \times N}$ is a rectangular matrix with possibly varying dimensions M and N.

3.1. The training process

A good network should achieve a low loss value. Hence, the parameters θ of the network need to be chosen in a way that minimizes the loss, meaning the task of training a neural network is fundamentally an optimization problem for the parameters θ of the network. Those parameters include the weights W_{ij}^l as well as the biases b_j^l, but can furthermore include other learned parameters of the network, such as the batch-normalization parameters that will be discussed later. Since the composition of many hidden layers leads to millions of free parameters, namely the weights and biases, the optimization problem is solved approximately.

The techniques used to solve the optimization problem are variants of the so-called gradient descent method. This is an iterative technique, where the neural network equation, illustrated by the example of equation (3.6) and the loss function are evaluated, and the gradient of the loss with respect to the model parameters θ is calculated. The exact form of the loss function depends on the problem at hand, however, in this thesis, we mainly use the above discussed cross-entropy loss. By taking a step in the opposite direction of the gradient, one converges to a (perhaps local) minimum of the optimization space. To perform the updates of the weights and biases, the gradients are calculated via backpropagation. Backpropagation uses the chain rule to compute the gradients for each learned parameter of the neural network efficiently. The update then consists of changing the parameters by the gradient, scaled by a factor λ. This factor is called the learning rate and is one of the most important so-called hyperparameters. Those are free parameters that are not associated with the weights and biases or other components of the neural network, but are rather parameters that change the training behavior of the network. Hyperparameters play a crucial role in successfully training a neural network, and finding a good set of hyperparameters constitutes one of the significant challenges during the training process.

The learning rate is arguably the most important of those hyperparameters and has an impact on the time needed for training, the performance, and the networks' ability to generalize. A high learning rate allows for fast training by quickly traversing across the optimization space while running the risk of not properly converging. The high learning rate will lead to big steps taken in the optimization space, which might overshoot the minimum, as illustrated in Figure 3 B and result in oscillating learning behavior. A too low learning rate will result in long training times and the problem of getting stuck in local minima. A good learning rate will efficiently converge to a minimum of the loss function, as demonstrated in Figure 3 A. This

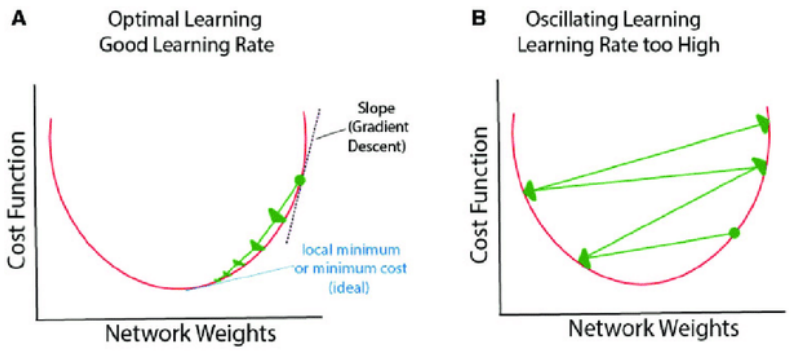


Figure 3: **Left:** An optimal learning rate decreases the cost function continuously and finds the optimal weight configuration as seen in A.
Right: A too high learning rate leads to oscillatory behavior and makes it harder to find the minimum in the optimization space.
Taken from [49].

minimum is ideally the global minimum of the optimization problem. However, in reality, this is often times a local minimum, which can pose a problem as this can lead to bad network performance, if the optimization process gets stuck in such a local minimum.

Due to the immense computational overhead of calculating the gradient for all training examples for each update-step and the slow update rate associated with this method, a stochastic method is used to approximate the gradients. This is called mini-batch gradient descent and uses a randomly chosen subset of the data to calculate the gradient. An update is performed with the estimated gradient, and the next subset is chosen. In machine learning, this subset is usually called a batch or mini-batch. Once all training examples have been used to update the network, one so-called epoch is finished, and different batches are created randomly. The batch size and the number of epochs used during training are further hyperparameters. Usually, the batch size is chosen to be rather small. This is called mini-batching. Mini-batching helps with overcoming local minima, as the small subsection of the dataset introduces noise into the calculated gradients. This helps the network find its way out of local minima due to the statistical fluctuation in the gradients. However, choosing the batch size too small comes with the risk of badly approximating the overall gradient landscape and thereby hindering efficient training.

The methods used to update the model parameters are not limited to the mini-batch stochastic gradient descent (SGD). There are a number of advanced optimization algorithms, usually called optimizer, used for training neural networks. All of them

are variants of the SGD that implement additional features to improve the training process. One of those methods is momentum, in which the gradient of the previous iterations is taken into consideration when performing the next update by calculating moving averages over the gradients. The gradient of the previous iterations is, for example, decayed exponentially, where the decay constant β is another hyperparameter. The gradient at iteration t is typically updated following

$$g_t \leftarrow \beta g_{t-1} + (1-\beta) g_t. \tag{3.7}$$

The momentum allows the optimization process to overcome small local minima, and rather continue in the direction of the global minima. Furthermore, it can also help with the problem of oscillating learning behavior as seen in Figure 3, by smoothing the gradient descent.

A commonly used optimizer, which is also used in this thesis, is called ADAM [50] and combines SGD with momentum and adaptive optimization algorithms. The adaptive learning rate used in this method allows for parameter-specific updates, increasing the training efficiency. Adam has a couple of parameters that control this adaptive behavior, which is realized through bias-corrected estimated moments. The algorithm can be summarized as follows:

ADAM Algorithm

while θ_t not converged **do**
- $t \leftarrow t + 1$
- $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
- $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t$
- $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2$
- $\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}$
- $\hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}$
- $\theta_t \leftarrow \theta_{t-1} - \lambda \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}$

end while

return θ_t

The first moment, which is the estimation of the gradient g_t, is estimated by an exponential moving average of the gradient mean m_t and the second moment is estimated analogously through the squared gradient v_t, where t denotes the optimization step. In general, the parameters associated with the exponential decay of the moving average need little tuning [50]. Good values for the exponential decay rates for the moment estimates are $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The factor $\epsilon = 10^{-8}$

guarantees numeric stability. With Adam even the learning rate λ requires less tuning than with other optimizers and a good starting choice is 0.001. Firstly, the first moment m_0 and the second moment v_0 are initialized to zero. Then, the following updates are performed on the initial model parameters θ_0 to best approximate the objective function $f(\theta)$. This optimization function is, in the context of this thesis, the loss function $\mathcal{L}$ and concretely the cross-entropy loss. Note here that the functional dependence is denoted as depending on the parameters θ and not the inputs x as above, since the goal of the optimization is to find the best parameters θ for any input x. From this, one can see that the initial choice of the parameters θ will always impact the training behavior of the model. This poses the question of how one can optimally initialize the weights of neural networks. As it turns out, this is to some extent dependent on the choice of activation function. However, all initializations should aim to avoid excessive reduction or magnification of the input values. For ReLU (and more general rectifiers), it has been shown in [51] that a sampling of the weights that is proportional to a zero-centered Gaussian with a standard deviation of $\sqrt{\frac{2}{N_l}}$, where N_l denotes the number of learned parameters in the layer l, leads to increased performance. This is known as Kaiming uniform or He normal initialization and is used for the initialization of all weight matrices of dense layers in this thesis.

Since in principle a neural network can approximate any function [52], it is plausible for very complex networks to learn a mapping from each training sample to its correct label, i.e. a lookup table. This is an extreme example of overfitting, where a network learns the dataset instead of general features of the data. This can be understood at the example of polynomials, where a polynomial of n-th degree can be described by $n+1$ parameters and can be used to fit $n+1$ data-points perfectly even if the underling ground truth is a polynomial of lower degree with some noise in the data as can be seen in the left part of Figure 4.

Overfitting manifests as a problem when the network is exposed to new data. As the network has not learned general features, its ability to correctly classify the unseen data points is limited. To check whether the network is learning general features or the data itself, one splits the available data into multiple subsets. The so-called validation set is used to evaluate the performance of the network during the training period, while the training set is used for training. The use of the validation set also allows comparison between different sets of hyperparameters. Different sets of hyperparameters are compared via the validation loss. For this, the best (meaning lowest) validation losses during the training of each model are compared. The set of

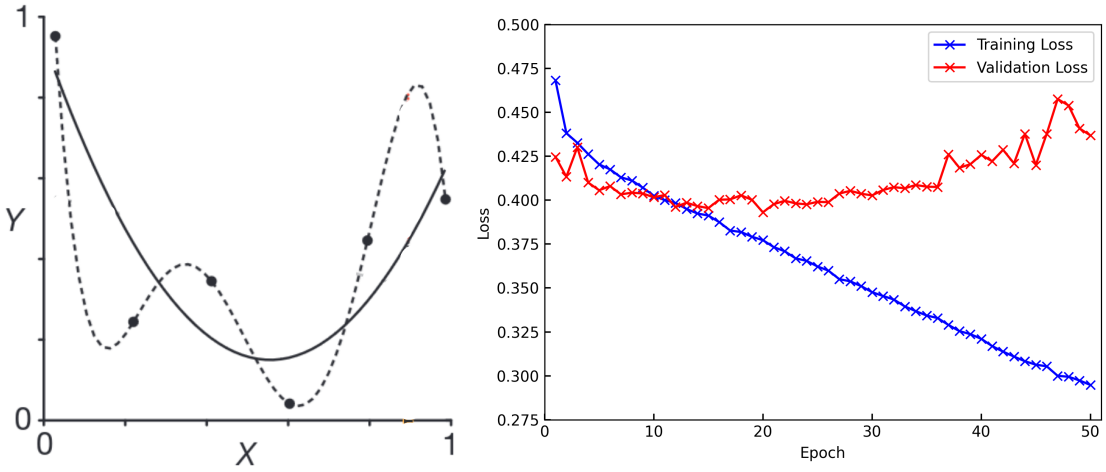


Figure 4: **Left:** Overfitting visualized by the example of polynomials, taken from [53]. The solid black line represents the ground truth distribution, from which the data-points are sampled with an artificially introduced variance. The dashed white line represent an over fitted polynomial, that matches the data-points perfectly but deviates from the ground truth distribution, fitting the variance instead of the underlying function.
Right: How overfitting manifests in a loss curve. The training- and validation loss diverge as the networks starts to learn the dataset instead of abstract features.

hyperparameters that produce the lowest validation loss is then chosen for the final model. Since the validation set is used during the optimization of the network and of the hyperparameters, the validation set is not completely new to the network. The performance of the model on the validation set has been used to optimize the hyperparameters, and should not be seen as an independent metric for the general performance of the network. For this reason, a hold-out test set is created, which is only used after all optimization and training have been performed. The test set is used to evaluate the final performance of the model after the hyperparameters have been chosen via the validation loss. Those results can then be presented as an independent (of the training process) metric.
Overfitting during training is manifested as a divergence between the training and validation loss. If the training loss continues to decrease and the validation loss starts to increase, this is a clear sign of overfitting. This is demonstrated in the right part of Figure 4.
To avoid this, regularization strategies can be implemented. One such method is the

so-called weight decay (also known as L_2 regularization or ridge regression in the case of regression tasks). The weight decay α is part of a penalty term added to the optimization problem that discourages model complexity. This penalty term $\mathcal{L}_{\text{req}}$ is, in the case of weight decay, the L_2 norm of the model parameters θ scaled with the hyperparameter α: $\mathcal{L}_{\text{req}} = \alpha \sum_i \theta_i^2$, with i counting all model parameters. Therefore, the optimization function given the loss function $\mathcal{L}$ (e.g. the cross-entropy in eq. 3.4) becomes $\mathcal{L} + \mathcal{L}_{\text{req}} = \mathcal{L} + \alpha \sum_i \theta_i^2$. It should be noted that the definition of the L_2 regularization in machine learning does not apply the square root as in the typical mathematical definition of the L_2 norm. The process of adding a regularization term is akin to the implementation of Lagrange multipliers for Lagrangian mechanics, where the imposed constraint is low model complexity.

The L_2 norm is not the only choice for a regularization term in the optimization function. One could also choose the L_1 norm, however in this thesis weight decay will always mean the hyperparameter associated with introducing a L_2 norm penalty term to the optimization function.

Lastly, we introduce methods to control the learning rate during the training process to reduce training times and improve the networks ability to generalize. Namely, learning rate scheduling. Here, the learning rate is changed dynamically during training, meaning the learning rate for the next set of iteration is dependent on the current epoch or iteration. This has the advantage that at the beginning of the training, a high learning rate that converges quickly can be chosen. Then, towards the end of training, the learning rate can be reduced to converge to a minimum. There is a plethora of different learning rate schedulers, of which two will be discussed in more detail. The so-called StepLR scheduler reduces the learning rate after a fixed number of epochs. One usually starts with a higher learning rate and then reduces the learning rate by an order of magnitude during the training. This step in the learning rate usually manifests as a sudden step in the loss function, as can be seen in Figure 5 at epoch 40. More complex schedulers have an oscillating learning rate. Those procedures were first proposed in [54] and have been refined to produce so-called super convergence [55] by implementing a one cycle learning rate policy. Super convergence aims to drastically reduce training time by cycling from a medium learning rate to one possibly multiple magnitudes larger and then to one that is magnitudes smaller than the initial learning rate. By this, the policy allows the model to find a direction in the optimization space what converges with the medium learning rate and then traverse quickly in this direction by increasing the learning rate and achieving convergence by then lowering it again.

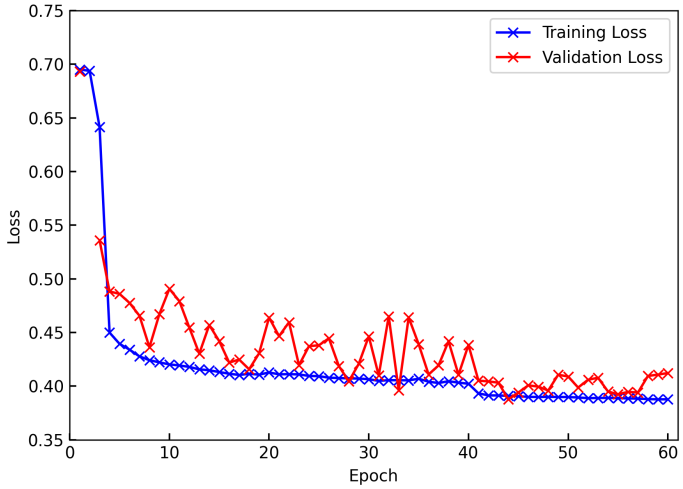


Figure 5: A loss curve with a StepLR learning rate scheduler. The learning rate is divided by ten at epoch 40. Subsequently, the training loss decreases further.

3.2. Neural network components

3.2.1. Additional regularization layers

In addition to the already mentioned components of neural networks, there are many common components used in neural networks. Some of them will be discussed in this section.

One such layer used to further reduce overfitting is the dropout layer. The dropout process sets a random subset of outputs (neurons) after a given layer to zero. This introduces a new hyperparameter, namely the dropout probability p. Setting a random subset of neuron activations to zero during each training iterations help to prevent the model from co-adapting [56]. Co-adaptation leads to neurons becoming overly reliant on each other during training, leading to redundancy, inefficiency, and poor generalization to new data. At test time, no dropout is applied, meaning all neurons are active and contribute to the model prediction. This can be interpreted as a set of thinned out networks being trained and then averaged during testing time [56]. The dropout probability p needs to be optimized and usually lies somewhere between 0 and 0.5. In this thesis, the dropout layer is always added after the activation function, setting some inputs of the next layer to zero.

Another layer that adds regularization are the batch normalization layers [57]. Those

layers normalize the input of a given mini-batch x around the mean $E(x)$ and scale them according to the variance $Var(x)$. This does not only add regularization but also helps the network learn by eliminating internal covariate shifts, where the distribution of inputs to each layer changes during training, due to updates performed to the preceding layers [57]. Batch normalization provides a stable input distribution to the next layer, which also addresses the problem of vanishing or exploding gradients [57], a common issue in training deep neural networks [58]. The problem of exploding gradients occur when the gradients of the loss function during the backpropagation process become very large, compared to their typical values. This leads to training instability, as the update to the weights is proportional to the value of the gradient. In case of large gradients, the updates performed during the backpropagation are also very large, leading to poor model convergence because the minima in the optimization space cannot be reached as the updates overshoot the minima. Vanishing gradients lead to a non-learning network, as the parameter updates are vanishing with the gradients before the minima has been reached. To combat this, [57] introduces the batch normalization layer

$$y = \frac{x - E(x)}{\sqrt{Var(x) + \epsilon}} \cdot \gamma + \delta, \tag{3.8}$$

where the input is shifted by the mean over the batch $E(x)$ and scaled by the batch variance $Var(x)$. The mean and variance are calculated for each dimension of the input across the batch. For example, for an image like input with dimension 50×50 with 3 channels, the mean of the 50×50 images is calculated for each of the 3 channels across the batch. Furthermore, the output of the batch normalization layer is then also scaled by the learned parameter γ and shifted by δ. This helps to conserve the expressiveness of the neural network and for example also allows learning to disregard the normalization by choosing $\delta = E(x)$ and $\gamma = \sqrt{Var(x)}$. As can be seen in the above example, the dimensionality of the parameters depends on the type of input to the batch normalization layer. The parameter ϵ is introduced to guarantee numerical stability and is set to $\epsilon = 10^{-5}$. During evaluation of the model, learned estimators for $E(x)$ and $Var(x)$ are used to normalize the inputs instead of the normalization across the batch. A batch normalization layer (short BatchNorm) is usually applied directly before the activation function, to center the inputs around the active region of the activation function.

3.2.2. Covolutional layers

In addition to the fully connected (dense) layers already introduced above, there are a couple of other commonly used layers with free parameters what are learned during the training process, one of which is the convolutional layer. They mainly find application in the field of computer vision and are especially useful for multidimensional inputs like images. This layer consists of a convolutional kernel of varying size (usually denoted (H, W)) where the entries of the kernel are learned parameters. The kernel is convoluted with the image and an output feature map is produced. Depending on the size of the kernel and the stride size (number of pixels between the applications of the convolutional kernel) this reduces the size of the original input image. To counteract this effect, padding is used. Hereby, the border of the image are padded with empty pixels, allowing the network to keep the same dimension after each convolutional layer or reduce the dimension in a controlled way. A convolutional layer can also have a bias, however in this thesis this bias is not added unless stated otherwise. Since almost all convolutional layers are followed by a BatchNorm, the free parameter δ fulfills a similar purpose and the network does not lose any expressiveness.
A typical image consists of three channels corresponding to the RGB values of the pixels meaning the input is of shape $(C_{\text{in}},\ H_{\text{in}},\ W_{\text{in}})$ where C_{in} denotes the number of input channels of the image. The convolutional kernel is then applied to all channels and the convolutions of each channel are summed up. Furthermore, a convolutional layer usually has multiple learned kernels, often also refered to as channels C or the width of the network. Each kernel generates its own feature map, so that after the layer the output has the possibly different shape $(C,\ H_{\text{out}},\ W_{\text{out}})$. In the case of quadratic kernel with size k and uniform stride s as well as uniform padding p, the output dimension can be calculated as

$$\left(C,\ \frac{H_{\text{in}} + 2 \cdot p - k}{s} + 1,\ \frac{W_{\text{in}} + 2 \cdot p - k}{s} + 1\right). \tag{3.9}$$

This means that for a stride of one and a kernel size of three a padding of one leads to preserved image sizes. In the case of convolutional neural networks, dropout layers are less common and are not used in this thesis.

3.2.3. Graph neural network layers

A common representation of a graph is the so-called adjacency matrix, which indicates which nodes of the graph are connected. Here $i, j \in 1, ..., N$, where N denotes

the number of nodes in the graph. If an element A_{ij} of the adjacency matrix A is not zero, this shows that the nodes i and j are connected. This also allows expressing various details of the connection between i and j by giving meaning to the elements A_{ij}. This is, however, not done in this thesis. An alternative to the adjacency matrix is the adjacency list, which is significantly more space-efficient for sparse graphs (graphs with low connectivity). For this, each node is given an arbitrary identifier, e.g. a number, and all pairs of connected nodes are listed as tuples of their identifiers.

In the case of graphs as the representation of the LHC event, defining a neural network is not so straight-forward. A feature of our data is that, in principle, the number of neighbors of each node in the graph is not predetermined. This makes it impossible to simply concatenate the features of all neighbors and pass them through an MLP. Hence, there is a need for an algorithm that can take in the full graph structure and make use of the graph connectivity. Furthermore, the typical representation of graphs, the adjacency matrix, is permutation-invariant, meaning the output of the network should not depend on the ordering of the nodes in the adjacency matrix either. The convolutions of CCNs depend on the grid structure of the input data, and MLPs always expect a fixed input size. Since those are not applicable to the case of graphs, a different approach is chosen. Namely, the message-passing algorithm that is the basis of some of the graph neural networks (GNNs) used in this thesis. Message passing consists of two stages. The message phase, in which each node receives a message from all its neighbors, followed by a permutationaly invariant aggregation of all messages and the update phase. For the messages, the features of each node are usually processed by a neural network, whose output is the message to its neighboring nodes. The permutationaly invariant aggregations can be, for example, the element-wise mean or maximum of all messages. The update consists of another network that takes the aggregated message and the node features of the local node and constructs a new updated node representation that incorporates the neighborhood information. Therefore, message-passing layers map one node representation to another updated representation and are therefore usable in succession, and deep graph neural networks can be constructed. A visual representation of the message-passing algorithm is shown in Figure 6. However, most GNNs do not stack many message-passing layers, as the more layers are stacked, the more distant nodes can influence the update process. For example, a two-layer GNN will aggregate information from its second order neighbors. The advantage of GNNs is their ability to contextualize local information in the updated graphs

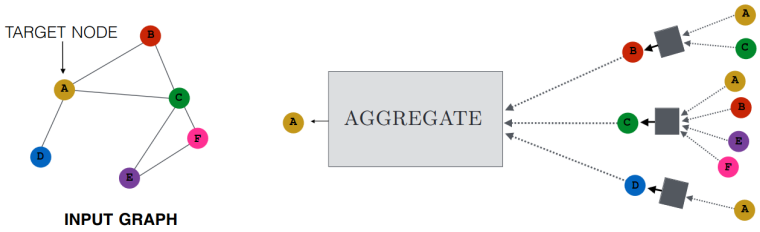


Figure 6: The message-passing algorithm constructs updated node embeddings by aggregating messages from the local graph neighborhood. Taken from [59]. The figure shows a two-layer message-passing scheme where node A can collect information from its second-order neighbors. Since all messages are constructed with the same aggregation network, the computational graph can be constructed without knowing the number of neighbors a priori.

representations, if too deep GNNs are constructed, one runs the risk of losing the local structure of the graph representation.

The variants of the message-passing algorithms are used to construct updated graph representations of the particle graph. At the end of the update process, the feature vectors of all nodes (constituent particles) are averaged and then passed through an MLP to project the feature vectors to the required output dimension.

4. Machine learning data in particle physics

This chapter will focus on the structure of the data that is used for training the various classifiers used in this thesis and how it was simulated. The data is fundamentally a detector simulation for events at the high-luminosity LHC. Firstly, we present the general structure of the data in sec. 4.1. Then we elaborate on the issues of the detector simulation as it was performed in [15] and outline how to correct them in sec. 4.2. Lastly, we present the details of the event simulation as performed for this thesis in sec. 4.3.

4.1. Data structure

For the machine learning task presented in this thesis, the data is divided into three datasets, containing three different representations of the LHC events.
During the simulation process, higher-level features of the events are calculated, such as the transverse momentum of the light and b-tagged jets and other kinematic features. In total, there are 106 such features that constitute the kinematic representation $\mathcal{K}$ of each event, which is given as

$$\mathcal{K} = \bigcup_i \{p_{x,y,z}, E, m, p_T\}_i || \bigcup_{i \neq j} M_{ij} || \bigcup_{i \neq j} \Delta R_{ij} || \{\vec{\not{P}}_T, S_T\}, \tag{4.1}$$

where $\{p_{x,y,z}, E, m, p_T\}_i$ is the set of momenta, energy and mass of the reconstructed objects i and M_{ij} and ΔR_{ij} are the invariant mass and the angular distance between two of those objects. The invariant mass is defined as $M_{ij} = \sqrt{(p_i + p_j)^\mu (p_i + p_j)_\mu}$, with p_i^μ being the associated four-vector of a reconstructed object. The angular distance between objects is calculated as $\Delta R_{ij} = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}$, with the pseudorapidity and detector angle differences $\Delta\eta$ and $\Delta\phi$ of the reconstructed objects. $\vec{\not{P}}_T$ is the missing transverse momentum of the event, and S_T the scalar sum of transverse momenta, which includes the isolated leptons present in the events, and $||$ denotes the concatenation of the variables into one flat list. The kinematic representation contains information about the three most energetic b-tagged jets and light jets as well as the two isolated leptons, that are required to be present in all events, which will be explained in section 4.3.
Furthermore, a low-level detector simulation is performed to generate constituent-level detector images. They are a more low-level representation of the above described kinematic data. Those images have a resolution of 50×50 pixels where the x-axis is the pseudo-rapidity η and encompasses a range from $\eta \in [-2.5, 2.5]$. The

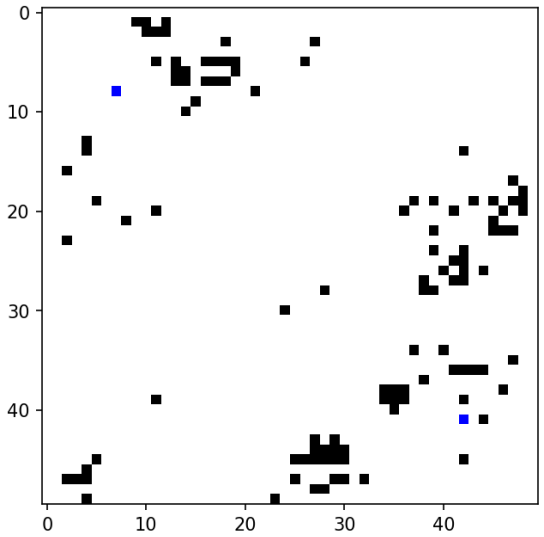


Figure 7: A typical jet image where the ϕ- (y-) and η- (x-) axes are converted to 50 detector pixels in each direction. In black, one can see the charged and neutral detector hits. The intensity of all pixels is set to be the same to better visualize the structure of detector images. The blue pixels are the two isolated leptons required to be present in all selected events.

y-axis is the detector angle ϕ and spans the range from $-\pi$ to π. The intensity of each pixel is given by the summed transverse momentum p_T of all constituent particles that fall within this detector pixel. Each image consists of three channels representing the charged and neutral hadrons, as well as one channel for the isolated leptons. A typical jet image can be seen in Figure 7, where the intensity of all active pixels is set to the same level to better visualize the image structure.
Seeing the image in Figure 7, one realizes that the detector images are fundamentally different from the images used in computer vision. In computer vision, one usually deals with black-and-white or colored images in which most pixels have values, whereas the detector images consist of mostly empty space. The application of convolutional neural networks is, therefore, not obvious at first. However, if one looks at the overall distribution of p_T in the images, one can see that the distributions are distinct. This can be seen in Figure 8 at the example of one signal and background process. It should be noted that these distributions can only be observed if all events are transformed into the same reference frame. This will be discussed in more detail later on. Those distributions have first been shown in [15] and the here shown distributions follow the same patterns, even though the simulation process used differs slightly, as will be discussed in sec. 4.3. The hypothesis

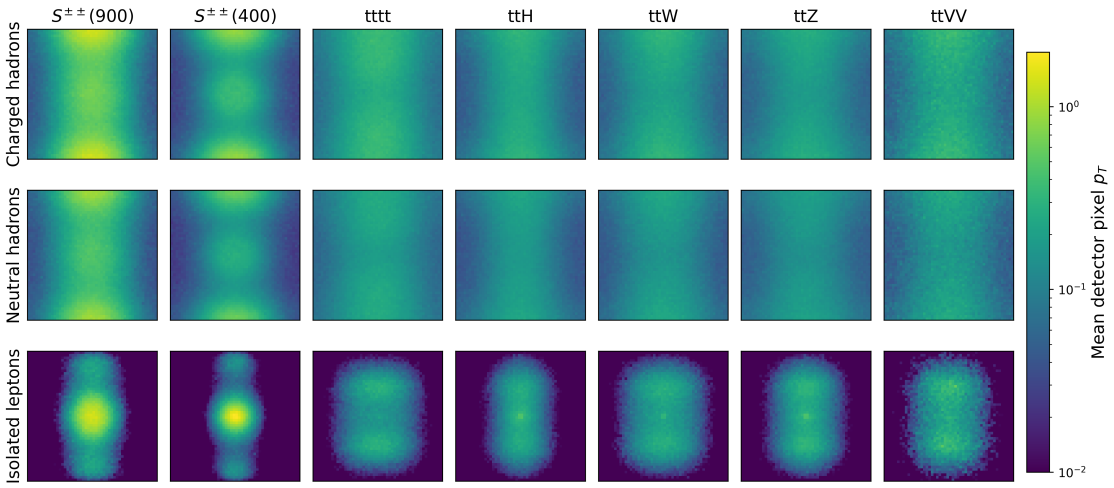


Figure 8: Distribution of the transverse momenta of the relevant background processes and signal processes ($m(S^{\pm\pm}) = 900$ GeV and $m(S^{\pm\pm}) = 400$ GeV). The detector images encompass the pseudorapidity region from $\eta = -2.5$ to $\eta = +2.5$ and the detector angle ϕ is displayed from -π to π. The detector images have a resolution of 50×50 pixels. The distributions of signals and background processes are distinctly different in shape and mean transverse momentum p_T per pixel.

is that the neural networks are able to identify to which distribution the images belong, since the shape of the distribution and the mean transverse momentum p_T per pixel are distinctly different. The signal processes are generally more energetic than the background processes, meaning they have a higher transverse momentum p_T. Figure 8 only shows the distribution of two mass points. However, this trend is observable for all mass points and is more pronounced for higher signal masses. Furthermore, the isolated lepton impacts of the signal processes are scattered less broadly along the pseudorapidity axes. They have a lower angular separation, since both same-sign leptons originate from the same doubly charged scalar $S^{\pm\pm}$, which decays into highly boosted particles, that in the end produce the same-sign leptons. Hence, they have a low angular separation in their rest frame (which is the reference frame for the distributions). On the other hand, the same-sign leptons of the background processes cannot originate from the same particle, which leads to a larger spread of the distribution.

In addition to the neutral, charged and lepton detector images, we can calculate additional input channels from the kinematic data. There is a b-tagging procedure

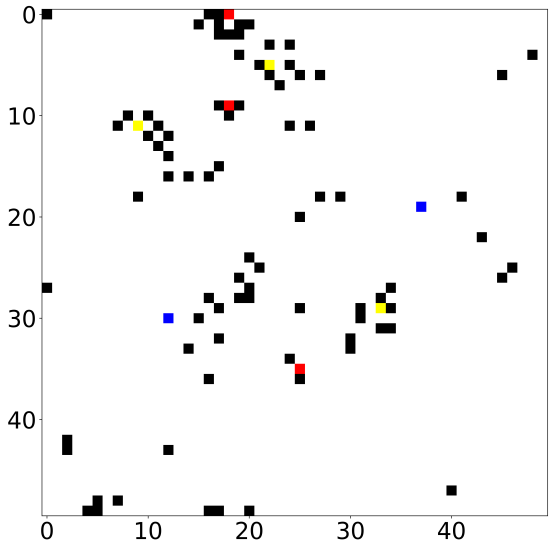


Figure 9: A jet image with added bottom (red) and light (yellow) jet-center positions. The ϕ- (y-) and η- (x-) axes are converted to 50 detector pixels in each direction. In black, one can see the charged and neutral detector hits. The intensity of all pixels is set to be the same to better visualize the structure of detector images. The blue pixels are the two isolated leptons.

applied to all reconstructed jets. Therefore, it is possible to construct additional jet images that display the jet-center positions of b-tagged and light jet-centers. We introduce a fourth (b-tagged jet-centers) and fifth (non-b-tagged jet-centers) detector image for the jet image part of the event representation. To calculate the images with jet-center positions, one first needs to calculate the η and ϕ coordinates from the kinematics and then transform them into the lepton rest frame and implement the artificial periodicity, as will be described in sec. 4.2. The value of the active pixels in the new image channels is the p_T of the reconstructed jets. A jet image example with the added jet-center position is shown in Figure 9. It can be seen that the jet-center positions indeed mark different jets. Since the intensity of all constituent particles is set to be the same for visualization purposes, it is hard to see if the positions lie in the exact center of the jet. In each image the three most energetic jets are marked, however the images may also contain constituent particles of other jets. We hypothesize that the addition of those new channels that incorporate some of the higher-level features of the LHC event into the jet image data, will improve the discriminatory power of the CNNs, as both of the jet-center images also display characteristic distributions as shown is Figure 10.

Finally, the constituent particles are presented as a point cloud (also called particle

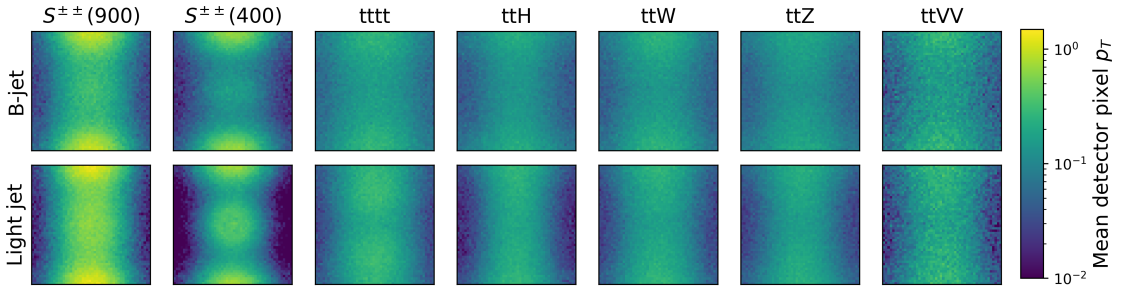


Figure 10: Mean transverse momentum p_T distributions of the bottom and light jet-center images. The signal process has a distinct distribution. Both the light and bottom jet-center images have a higher total p_T compared to the background events.

cloud). This was inspired by the data structure used for the successful "ParticleNet" [27]. In contrast to the images, the constituents are not summed up into detector pixels but are rather kept distinct. Therefore, it is possible to attribute more features to each constituent.

Each constituent is given a feature vector that contains information about its type (charged or neutral hadron, or isolated lepton) and its kinematics. The feature vectors contain the total energy E, the transverse momentum p_T, the detector angle ϕ, the pseudorapidity η and the momentum components $p_{x,y,z}$. Lastly, the particle type (charged or neutral hadron, or isolated lepton) is one-hot encoded in the feature vector, meaning we append three labels to the feature vector, which take the value one if the constituent is of the given particle class and zero otherwise.

4.2. Issues with the detector simulation

At the beginning of this thesis, it was planned to use the original data from [15]. However, there were multiple factors that led to the creation of a new dataset. Firstly, the dataset only encompasses around 80,000 training events, which might hinder the training process for more complex neural networks since those require a large variety of training examples, and this is not a particularly large dataset, especially considering that no data augmentation is used.

Moreover, the original data did not include the point cloud representation of the events. Such a particle cloud could have been calculated from the detector images; however, it would not resolve the individual constituent particles, as was done in [27] and would encompass fewer features per point in the cloud.

Most importantly, there were multiple issues with the simulation process in the preprint version of [15], which have been uncovered during the work for this thesis. Those issues lie mostly on the level of the DELPHES detector analysis, as will be discussed in the following.

Since all the data needs to be Lorentz-boosted into a common reference frame to make use of the underlying distributions, a shift in the η and ϕ coordinates has to be calculated and applied to all jet image pixels. We chose to boost all events into the rest frame of the isolated leptons, fixing the origin of the pseudorapidity-azimuth plane at the lepton rest frame center. All other constituent particles are then shifted accordingly. The shift in ϕ is unproblematic, since the detector images are periodic in ϕ. The η-coordinate is, however, restrained by the physical limits of the detector. The implementation of this is what has led to problems in the preprint version of [15].

The simulated data has, at first, no limits on η. Therefore, the physical detector limits need to be imposed. A limit of $\eta \in [-2.5, 2.5]$ has been chosen. However, during the generation of the data in the paper [15] a limit of $\eta \in [-3, 3]$ has been wrongfully applied. Furthermore, this wrong limit was not enforced consistently. Which meant the limit was only enforced for values of $\eta > 3$, but not for $\eta < -3$ for the neutral constituent particles of the jets.

This results in unobservable constituent particles being displayed in the jet images. Since the maximally allowed η-shift is $\delta\eta = 2.5$ this could, in the most extreme case, result in particles with an η-coordinate of almost $\eta = \pm 5$ being present in the jet images. A pseudorapidity of $\eta = 5$ corresponds to a relative angle to the beam axis of below $1°$. This would be, of course, a particle that is firmly outside the detector range and therefore could never be observed.

Therefore, it is necessary to correct the analysis. The correct cuts can be applied when simulating new data, or can be enforced during the training process as a mask that is laid over each image before it is passed to the neural networks. This, however, results in large chunks of the images being just empty space. Namely, the parts of the image that had a pseudorapidity coordinate of over $|\eta| = 2.5$ before the Lorentz-Boost but now lie within the jet image. Since no particles can be physically detected in this region, only empty pixels are displayed. This could be a problem for the convolutional neural networks, since it is unclear if the presence of so many empty pixels in some images could lead to training instability and therefore impaired learning.

The whole procedure also comes with another problem, as there are not just empty

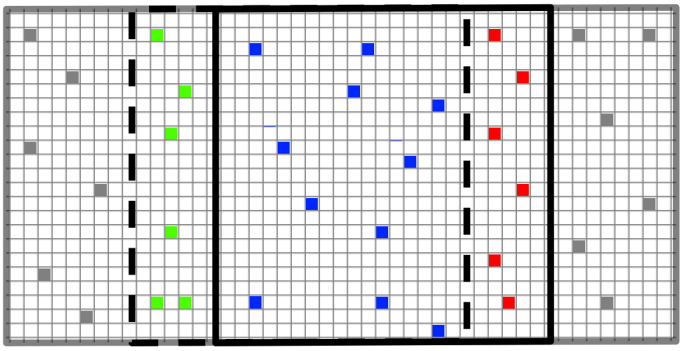


Figure 11: A schematic illustration of the issue that arises with the Lorentz-boost in the pseudorapidity η. The gray background grid symbolizes an extension of the 50×50 detector grid for visualization purposes. The dashed square gird is the simulated detector image before the $\eta-$transformation and includes the physical detector impacts (green and blue pixels). After the $\eta-$transformation (solid black line) the green impacts are shifted outside the detector image. If the cuts are applied incorrectly, unphysical (red) detector hits are contained in the jet images.

.

pixels being shifted into the jet images, but there are also constituents that have been detected that are now shifted outside of the jet image bounds. Those particles had an η-coordinate of less than $|\eta| = 2.5$ before the Lorentz-boost but have now acquired an apparent pseudorapidity of more than 2.5 and are therefore not displayed in the jet images. This leads to a lot of physical data being lost for the neural network analysis. Hence, it is desirable to implement methods that salvage the physical data points. The issue of unphysical data being displayed and physical data being lost is illustrated in Figure 11.

Two ideas have been tested for their feasibility. Firstly it is possible to simply enlarge the images to, for example, a size of 100×50 pixels, where all constituents are shifted to the lepton rest frame and all the nonphysical image space that corresponds to a pseudorapidity $|\eta| > 2.5$ holds just empty pixels. This leads to an off-center embedding of images like in Figure 7 in the bigger 100×50 pixel images. However, this also comes with the problem of having large empty spaces in the jet images and further requires a lot of computational resources to hold the larger images in the video-memory of the GPUs used for training. The alternative of implementing an artificial periodicity in η is therefore preferred. Here, all the pixels that fall outside the image bounds in the η direction are periodically shifted back into the frame.

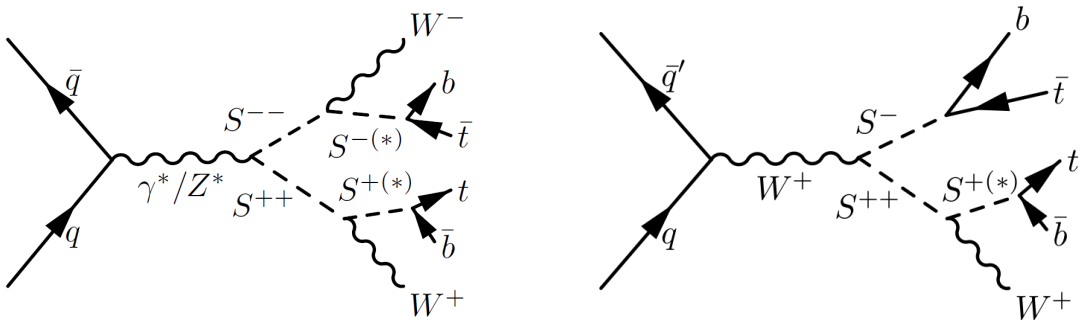
This artificial periodicity is, of course, unphysical. However, this representation is only used as input to the neural networks and hence does not necessarily need to be a physical image, as long as it only contains physical and observable particles. Furthermore, it is a reasonable compromise to salvage the physical data points and not incur the cost of increased memory usage and computational costs.
Since it is suspected that the ability of the convolutional neural networks to properly classify the jet images is dependent on the underlying differences in the p_T distributions, those have been calculated for a single background and signal channel as shown in Figure 8 and they do not differ significantly from the original distributions from [15].
Those changes to the data used for training the neural networks have also been applied to later releases of the paper [15].
Two other discrepancies in the code that was used to generate the original datasets were discovered. Namely, two preselection cuts in the DELPHES analysis were not imposed correctly. The preselection cuts will be discussed in detail in sec. 4.3, however, we mention the problematic cuts here for the sake of completeness.
The minimum p_T requirement for the muons (part of the isolated leptons) was set to be 10 GeV which is not in accordance with [15]. There is a cut of 20 GeV is proposed. This should, however, not have a big impact on the analysis. There was a similar issue in the calculation of the missing transverse momentum of the bottom-tagged jets. Here the cut is stated to be done at 25 GeV but was implemented as 20 GeV. Both discrepancies were fixed for the preceding evaluations in this thesis. Furthermore, it should be noted that the definition of the missing transverse momentum used in [15] which was originally defined in [60] and the DELPHES implementation are not congruent. In the definition in [60] the missing transverse momentum is calculated in regard to the transverse momenta p_T of all the leptons, photons, jets and soft tracks that pass a certain p_T threshold

$$\vec{\not{P}}_T = -\left(\sum \vec{p}_{T,l} + \sum \vec{p}_{T,\gamma} + \sum \vec{p}_{T,j} + \sum \vec{p}_{T,\text{track}}\right), \tag{4.2}$$

whereas in the DELPHES implementation, all the aforementioned particles are taken into account regardless of the minimum p_T requirement. This is physically useful since those constituents are not detected and therefore contribute to the missing transverse momentum, but it is in contradiction to the stated definition. We keep the implementation of the missing transverse momentum as described here.
The above-mentioned issues have not been fixed in the current version of [15], however, they should not influence the final analysis significantly, since the deviations

are minor.

4.3. Simulation process



(a) Production of the $S^{\pm\pm}$ pNGBs and the following decay chain.

(b) Production of singly charged pNGBs S^- and their decay chain.

Figure 12: Drell-Yan production of doubly (left) and singly (right) charged scalar pNGBs. Taken from [15]. The doubly charged scalars constitute the signal process for the machine learning task in this thesis.

A larger dataset is generally useful when training neural networks, as it prevents overfitting and allows the model to learn abstract features more easily. Furthermore, a larger dataset might also combat stability issues during training. Therefore, we aimed to generate a dataset that is almost ten times the size of the original dataset used in [15]. For this, we follow the simulation process in [15], as outlined in the following.

We consider the Drell-Yan production of the doubly charged scalars $S^{\pm\pm}$ described in sec. 2.6, which follow the decay path

$$q\bar{q} \to S^{++}S^{--} \to (W^+t\bar{b})(W^-\bar{t}b) \to (W^+W^+b\bar{b})(W^-W^-b\bar{b}), \tag{4.3}$$

as shown in Figure 12a. The decay is mediated by off-shell singly charged pNGBs $S^{\pm}$, that decay predominantly into third-generation quarks since their coupling to electroweak bosons is loop-suppressed. This produces a decay chain consisting of four b-quarks and four W-Bosons after the top-quarks decay. This decay chain is similar to the decay chain produced by $q\bar{q} \to S^{\pm\pm}S^{\mp}$ as shown in Figure 12b, with $S^{\mp}$ being from the same multiplet. The influence of those decay channels on the analysis has been studied in [15]. Here, the final states resulting from singly charged pNGBs are disregarded.

To simulate the LHC events, the public FeynRules [61] implementation of the eVLQ model [12, 62], is used to create a UFO file [63] with which, a new dataset is generated using MadGraph_aMC@NLO version 3.5.1 [64–66] to generate the events at a center of mass energy $\sqrt{s} = 14$ TeV and Pythia8 version 8.3.06 [67] is used to shower the events. We use the NNPDF2.3QED parton distribution function [68] with dynamical renormalization and factorization scales. To perform the detector simulation, Delphes 3.5.0 is used [69]. The Delphes configuration is based on [60] and the jet reconstruction is done using FastJet 3.4.1 [70]. An anti-k_T algorithm [71] is used with a cone radius of $r = 0.4$ and for b-tagging, a flat efficiency of 80% is assumed, whereas the probability of mistagging a c-jet to a b-jet is assumed to be 20%. Mistagging for a normal jet to a b-jet is set at 1%. This implementation follows the ATLAS report [72] as is done in [15]. We generate signal events with different masses $m(S^{\pm\pm})$ ranging from 400 GeV to 1000 GeV in steps of 100 GeV. A flat k-factor of 1.15 [73] is applied to the cross sections to take next to leading order (NLO) effects into consideration.

Table 4 summarizes the important background processes. Due to computational limitations, the $2 \to 4$ background processes $t\bar{t}t\bar{t}$ and $t\bar{t}VV$ are only generated at leading order (LO). Here V denotes the gauge bosons $W^{\pm}$ and Z. The other dominant backgrounds, $t\bar{t}V$ and $t\bar{t}H$ are generated at NLO. All background processes are otherwise simulated in the same way as the signal processes. Other backgrounds like VVV and VV are negligibly small [15] and are not taken into account for this thesis.

Process	$\epsilon_{\text{preselection}}$	Cross section [fb]	Events at 3 ab^{-1}
$S^{++}S^{--}$	$1.01 \cdot 10^{-2}$	$4.89 \cdot 10^{-2}$	146
$t\bar{t}W^{\pm}$	$2.21 \cdot 10^{-4}$	$9.82 \cdot 10^{-2}$	295
$t\bar{t}Z$	$1.40 \cdot 10^{-4}$	$1.29 \cdot 10^{-1}$	387
$t\bar{t}h$	$3.76 \cdot 10^{-4}$	$2.12 \cdot 10^{-1}$	636
$tt\bar{t}\bar{t}$	$1.52 \cdot 10^{-2}$	$1.80 \cdot 10^{-1}$	540
$t\bar{t}VV$	$1.43 \cdot 10^{-3}$	$2.71 \cdot 10^{-2}$	81

Table 4: Preselection efficiency and cross section as well as the expected number of events for the high luminosity LHC for the relevant background processes and the signal process with $m(S^{\pm\pm}) = 400$ GeV.

Numerous preselection cuts are imposed on the events to be selected for the analysis, which follow [15] with the in sec. 4.2 described deviations. Two same-sign isolated leptons are required for every event to reduce the standard model background. Isolated leptons are required to fulfill the isolation criterion $\frac{p_T(l)}{p_T(l)+\sum_i p_{T,i}} > 0.7$, where $\sum_i p_{T,i}$ is the sum of the transverse momenta of nearby particles which have a transverse momentum $p_T > 0.5\,\text{GeV}$ and an angular distance $\Delta R_{il} > 0.3$ to the next reconstructed objects. Furthermore, the isolated leptons need to have a minimum transverse momentum of $p_T > 20$ GeV and a pseudorapidity that falls within the detector $|\eta| < 2.5$.

The typical decay of the doubly charged scalar particle, as seen in Figure 12a, would imply a final state with four b-tagged and four non-b-tagged jets. This is, however, a very restrictive cut since this busy final state makes it difficult to identify all jets correctly. Hence, we choose the more inclusive search strategy proposed in [15] and only require three b- and non-b-tagged jets in the final state. For both types of jet, we select the three most energetic jets for the kinematic representation of the events discussed above and require them to have a minimum $p_T > 25$ GeV. Furthermore, the events have to have a minimum missing transverse momentum of $\vec{\not{P}}_T > 20$ GeV, which is defined as discussed above in eq. (4.2). Lastly, the events are required to have a scalar sum of transverse momenta of the reconstructed jets and same-sign leptons of $S_T > 400$ GeV. The preselection efficiencies are summarized in tab. 4. The preselection efficiencies denotes the part of events that pass the above-mentioned cuts. This table further shows the cross section after the preselection cuts as well as the expected events at the high-luminosity LHC with a total luminosity of 3 ab^{-1}.

During the simulation process, we experimented with implementing in-line decays, analogously to eq. (4.3), for the background processes. Implementing such in-line decays would decrease the simulation time significantly, since irrelevant decay paths that would hardly ever pass the preselection cuts could be eliminated and would not need to pass through Pythia and Delphes. The relevant decay paths could then be implemented on the MadGraph level of the simulation. However, this procedure has led to a couple of questions that have not been fully answered so far. Hence, all background processes are simulated without any in-line decays in this work and suggest studying this technique in further work.

5. Deep learning models for semi-hadronic final states

In this chaper we present the different neural networks that are applyed to the classification task in this thesis. All networks have been implemented using the PyTorch library [74]. The implementation of the models used in this thesis can be found at a publicly available GitHub repository and we refer there for the specific details of the networks [1].

All CNN models in this thesis use BatchNorm and all MLPs and GNN use BatchNorm and dropout.

Deep learning models require extensive hyperparameter tuning and sometimes even tuning of the architectural parameters like hidden dimensions. For this purpose, we use the OPTUNA framework [75]. This allows for an optimized search strategy in the hyperparameter space, where each run using a specific set of hyperparameters, is called a trial. Further, we use median pruning to terminate unpromising trials before completion. Median pruning terminates a trial if the trial's best intermediate result is worse than the median of intermediate results of previous trials at the same epoch. Usually, each trial is given a grace period of at least half the maximal training epochs after which median pruning is applied, which leads to a very lenient pruning scheme. We present the used hyperparameters in appendix A.1.

Some models have defining architectural parameters, such as the number of layers in a deep convolutional network model. Since those parameters define a plethora of different levels of model complexity, it is not feasible to optimize for all of them; rather, we pick a subset of such parameters, defining more and less complex model classes and compare those, whereas the number of hidden neurons in an MLP is a less architecture-defining feature and is keep as an optimization parameter.

In addition to such parameters, the most important hyperparameters to optimize are batch size, learning rate, weight decay, and drop-out rate. Those are always optimized for each model, where applicable. For this, we proceed in two stages. Firstly, the hyperparameters are varied over many orders of magnitude and a subset of the whole training-data is used. For this, we usually use around 100- to 200-thousand training examples. After this, the hyperparameter range is narrowed around the region what has been deemed optimal during the first search, and the optimization is tuned on a larger subset of around 500 thousand training examples.

[1] Full address: `https://github.com/ManuelSchmidt707/Machine-learning-methods-for-semi-hadronic-final-states-at-the-LHC`

Finally, the whole training-dataset is used to fine-tune the hyperparameters and employ learning rate schedulers. This is done manually for each network. The chosen hyperparameters for each model are summarized in the appendix A.1.

5.1. Jet image classification

5.1.1. Convolutional neural network CNL

We adapt the convolutional neural network architecture, termed CNL from [15] to perform as a baseline for the comparative study in this thesis. For this purpose, the hyperparameters are optimized on the new dataset for a fair comparison.
This architecture exists in different variants, taking as input either the charged and leptonic detector images or the charged or neutral hadronic and leptonic detector images. Here we focus on the variant that uses all three input channels and refer to it as the CNL model.
Using both the neutral and charged channels for this analysis is an optimistic assumption, since pileup effects for the neutral channel cannot be easily cleaned. There exist propositions for cleaning the pileup effects for charged particles [76]. However, those do not work for neutral particles, and other methods are being developed to tackle this issue [76, 77]. The impact of working only with the neutral and leptonic channel has been analyzed in [15] and we refer there for more details. The takeaway is that this reduces the overall network performance. In this thesis, we restrict ourselves to an analysis using all three channels. However, we expect that the same behavior would be observed here.
In [15] it was shown that combining multiple classifiers into one model working on multiple representations of the LHC event is beneficial. Hence, we expand on this strategy and take the CNLK model from [15], which works on jet images and the kinematic data, as a baseline.

5.1.2. ResNet and ResNeXt

Modern state-of-the-art CNNs implement additional architectural features such as residual connections, group convolutions, and "bottleneck blocks". Those types of features have been explored in [78], where the now ubiquitous residual connections (a specific type of skip connection) have been implemented for image recognition tasks with great success. The hypothesis is that instead of learning an unreferenced mapping, it is easier to learn a residual mapping, where, after a few convolutional layers, an identity mapping is added to the generated feature maps. Thus, the

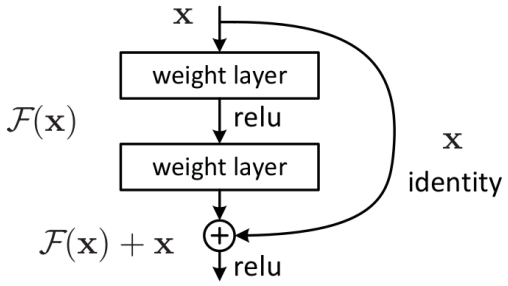


Figure 13: The basic residual convolution block used in the ResNet architecture, as first described in [78]. Two convolutional layers are used to learn the residual $\mathcal{F}$, which is combined with the identity mapping x. Figure also taken from [78]

.

mapping after a residual block $\mathcal{H}(x)$ is given by the learned residual $\mathcal{F}(x)$ and the identity maping of x as $\mathcal{H}(x) = \mathcal{F}(x) + x$, as depicted in Figure 13.
As discussed earlier, deep neural networks often suffer from vanishing gradients and gradient flow problems [58]. The residual connections further help with this problem. Moreover, very deep networks suffer from a degradation problem that is neither caused by gradient flow issues nor by overfitting but is rather related to the fact that deeper networks are harder to train [79, 80]. A deeper model should, in principle, always be able to replicate the performance of a shallower model, as the deeper model could always just consist of identity maps stacked onto the original model. However, experiments have shown that this is not the case, but residual connections can help alleviate the problem.
For this thesis, a ResNet network has been implemented that follows the design principles outlined in [78]. Each residual block consists of two 3×3 convolutional layers, each followed by a BatchNorm layer and a ReLU activation function. To keep the input and output dimensions of each residual block consistent, a stride and padding of (1,1) are used. An input layer with a 7×7 convolutional kernel, padding of (10,10) and a stride of 2 is used to bring the input into a more convenient dimensionality. To read out the last convolutional layer, adaptive 2D pooling is used, which is then passed through a linear output layer to project down to the output dimension of two.
The network consists of three stages, each of which consists of l residual blocks. During each stage, the dimensionality of the feature maps is kept constant by the design of the residual blocks. The first stage starts with 32 feature maps with 32×32

pixels. After each stage, the feature map is down-sampled with a convolutional layer with kernel size one and stride two and the number of kernels per convolutional layer is doubled. This leads to a halving of the feature map size after each stage. So for example, after the first stage, the feature map size is reduced to 16×16 and the number of convolutional kernels per layer is doubled to 64.
This results in a model class with easily scalable depth, where the total number of layers N is fixed as $N = 3 \cdot 2 \cdot l + 2$. In this thesis, two ResNet variants have been examined with $l = 3$ and $l = 18$, leading to one model with 20 layers, which shall be called ResNet-20 and one with 110 layers: ResNet-110.
The here described residual block can also be replaced by bottleneck blocks as described in [78]. There, the two 3×3 convolutional layers of each block are replaced by three convolutions. One of them has a lower input/output dimension. For this purpose, one usually uses a 1×1 convolution followed by a 3×3 with fewer feature maps followed by another 1×1 convolution to restore the original dimensionality.
A conceptual successor of the ResNet architecture is the so-called ResNeXt model type [81], where the bottleneck blocks are used in conjunction with the introduction of a new dimension to the convolutional setup, namely the cardinality. The cardinality introduces a number of C different paths in the residual bottleneck blocks. The paths each consist of a bottleneck setup with a 1×1 kernel, which reduces the dimensionality of the feature maps, followed by a 3×3 kernel with $\frac{F}{C}$ feature maps, where F represents the number of input feature maps. This is followed by another 1×1 convolution, which restores the original dimensionality. All paths are added together and then combined with the residual connection. This type of splitting can be thought off as a version of group convolution [82] and it has been shown in [81] that one can reformulate the above-described splitting as one residual block with the group convolution implemented in the 3×3 kernel, as is shown in Figure 14. Since it is easier to implement and more efficient, this approach is taken in this thesis. The addition of independent groups might allow the network to learn more distinct features, as the groups could focus on different aspects of the input, as those groups are completely independent of each other, which is not the case for normal convolutional layers. The projection to the output dimension works analogously to the ResNet implementation.

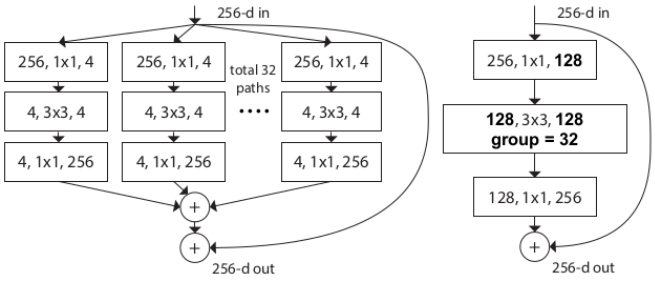


Figure 14: Depiction of the ResNeXt residual block, taken from [81].
Left: The block has a cardinality of $C = 32$ and splits the incoming 256 featuremaps into 32 paths with 4 paths each.
Right: This can also be thought of as a group convolution, with 32 groups in the 3×3 convolutional layer.

5.2. Classification based on sparse point clouds

The point cloud representation of the event-data is used to construct graphs, where the nodes represent the constituent particles. Each particle is given a feature vector that contains information about its type (charged hadron, neutral hadron or isolated lepton) and its four-vector. The feature vectors contain the total energy E, the transverse momentum p_T, the detector angle ϕ, the pseudorapidity η as well as the momentum components $p_{x,y,z}$. The later could be considered redundant; however, presenting the same information to the networks in different forms often proves useful. Finally, the node's feature vector contains information about the class of the particle it represents, namely whether it is a charged hadron, neutral hadron or an isolated lepton. This is one-hot encoded in the feature vector, meaning we append three labels to the feature vector, which take the value one if the constituent is of the given particle class and zero otherwise.
The connectivity of the graph is determined by a distance metric between the nodes. For this, two approaches are compared. Firstly, the graphs can be constructed by using the angular distance $\Delta R^2 = \Delta\eta^2 + \Delta\phi^2$ and secondly, by using the whole four-vector for the distance calculation. In both cases, the euclidean distance between two particles is used for the calculation. One could argue that for the case of four-vectors, one should take the Minkowski distance, however since we do not calculate physical distances this is a matter of choice. Based on those distances, we construct *knn*-graphs, meaning the k *n*earest *n*eighbors are considered connected. The graph is constructed on the 150 most energetic constituent particles, unless mentioned otherwise. This is done to keep the video-memory usage during training

constant, and is necessary to balance the size of the graph, based on the number of constituents included, against the maximal possible batch size.
A sensible alternative to the *knn*-approach is to connect the nodes based on their distance to other nodes, whereby all nodes within a certain radius would be connected. In the case of the angular distance as the distance metric, this gives the physical interpretation of all particles within a cone coming from the collision point being connected. However, this approach comes with the problem that it is not clear how many particles are going to be connected in each event, which leads to strongly varying video memory usage during training. While optimizing the batch size, this can easily lead to memory errors during training if one encounters a particularly strongly connected graph. For this technical reason, we have limited ourselves to the *knn*-graphs, even though this radius-based approach would be interesting for further research and has already been implemented.

5.2.1. ParticleNet

As a baseline for the comparison for graph based neural networks, we take the "ParticleNet" from [27] and modify the implementation of [83] to suit our data representation.
ParticleNet implements edge convolution [84] in conjunction with dynamic graph

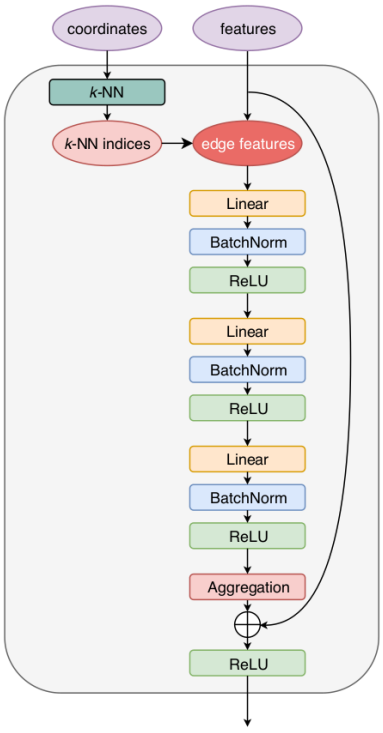


Figure 15: Structure of the ParticleNet edge convolution block. Taken from [27]. An edge convolution block takes in the position and features of each constituent particle and constructs a $knn-$graph. An MLP is used to implement the edge convolution. The feature vectors of each constituent particle are updated, and the new embeddings are used as coordinates and features for the next convolution block.

updates. The dynamical update of the graph, allows the network to learn the graph structure itself, by updating the connectivity of the graph with the learned embedding after each edge convolution layer. The structure of one such edge convolution block is depicted in Figure 15. Each block takes as input the coordinates to construct the *knn*-graph with 16 neighbors. The four-vectors in pseudorapidity-azimuth space are used as the coordinates and the features are the complete feature vector of each constituent in the point cloud, as described above. The subsequent blocks use the learned feature vectors as coordinates and features. The architectural makeup of the network follows [27] and employs three such edge convolution layers with the edge convolution realized through a three layer MLP as depicted in Figure 15.

5.2.2. Graph convolution networks

A simple network to apply to the classification task on graphs is the graph convolution network from [85]. Here, the matrix of activations h^l after layer l are updated by the layer-wise propagation rule

$$h^{l+1} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}h^l W^l\right), \tag{5.1}$$

where $\tilde{A}$ is the adjacency matrix with added self-connections (meaning each node is connected to itself), $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ is the degree matrix and W^l are learnable weight matrices. The non-linearity σ is chosen to be the ReLU function. This approach is motivated by a first-order approximation of localized spectral filters on graphs [86]. The above-described Graph Convolution Operator is implemented as a one-layer Graph Convolution Network (GCN) with a hidden dimension of 238 (the dimensionality of the updated feature vectors). After the graph convolution layers, the updated feature vectors are aggregated as the mean over all nodes, after which they are fed through an MLP to project to the output dimension. The input to the network is a $knn-$graph with five neighbors, and uses the angular distance as the distance metric to create the graphs.

5.2.3. Graph attention networks

Attention mechanisms have proven to be incredibly powerful since their application to transformers in the paper "Attention is all you need" [87]. The general concept of an attention mechanism is, however, not limited to transformers and can be used in the context of message-passing graph networks, as done in [88]. Here the feature vectors of the nodes $\vec{h}_i$ are transformed with the learnable weight matrix W

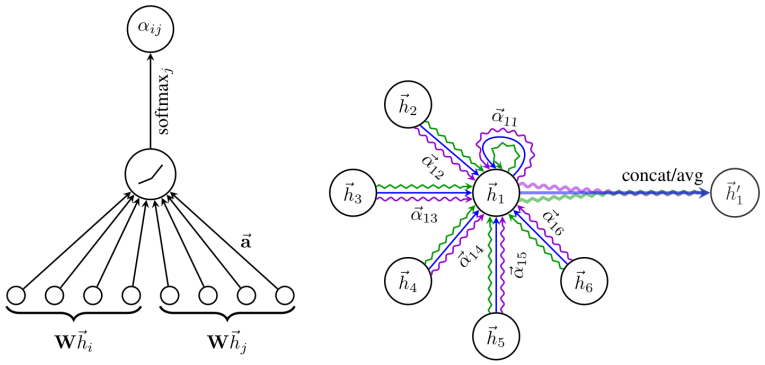


Figure 16: **Left:** The attention coefficient α_{ij} is calculated with a learned attention vector $\vec{a}$ by concatenating the transformed feature vectors $W\vec{h}_j$ of the neighboring node and the local node $W\vec{h}_i$.
Right: Multi-head attention is applied by calculating multiple (here up to three) independent attention mechanisms that are aggregated to update the local node.
Taken from [88].

as in the message passing scheme. In addition, for all neighbors j of the local node i an attention score α_{ij} is calculated with a learnable attention vector $\vec{a}$. This is illustrated in Figure 16. To calculate the attention score, the local and neighboring transformed feature vectors are concatenated, and the scalar product with the attention vector $\vec{a}$ is calculated and then normalized using the softmax function across all neighbors j. This way, the attention scores are more easily comparable for different nodes. This is done for each neighbor j. Additionally, to inject non-linearity into the attention calculation, before the application of the softmax function, a variant of the rectified linear unit is applied. Namely, the LeakyReLU, which is characterized by a hyperparameter μ and is slightly permeable for negative activations: $\text{LeakyReLU}(x) = \max(0, x) + \mu \min(0, x)$. All in all, the normalized attention scores α_{ij} are calculated as

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{a}^T \cdot (W\vec{h}_j || W\vec{h}_i)\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\text{LeakyReLU}\left(\vec{a}^T \cdot (W\vec{h}_j || W\vec{h}_i)\right)\right)}, \tag{5.2}$$

where $\mathcal{N}_i$ is the neighborhood of the node i, i.e. all nodes that i is connected to. The messages to the node i from its neighbors j are then weighted by the attention score α_{ij}. The input to the first graph attention layer is the *knn*-graph of the event, based on the angular distance calculation. The messages of the neighbors are then

weighted with the attention score α_{ij}. If multiple such attention mechanisms are used at once, we speak of multi-head attention. In this case, the weighted messages produces by each attention head are concatenated and used as the input of the next graph attention layer.
After the input has been passed through all graph attention layers, the updated graph embedding is aggregated as the mean over the features of all nodes. The aggregation is then projected to the output dimension with an MLP with three layers. We implement this network architecture to work with a *knn*-graph based on 5 neighbors (determined by the angular distance) and construct one model with one attention head (GAT-1) and another model with three attention heads (GAT-3). Each model has two graph attention layers.

5.3. High level feature analysis

For the high-level features, the most straight-forward approach is to pass the kinematic features through an MLP. This has been done in [15]. Here, the same network is implemented and trained on the new dataset to make a fair comparison. We will call this model FC, as is done in [15]. Since this network does not have any BatchNorm- or dropout-layers, we also implement a version of this network with those additional regularization layers (FC+Reg.). We also propose another MLP architecture, which has significantly shorter training times and fewer parameters. It consists of 6 fully connected layers, dropout layers and BatchNorm layers. It has a total of almost 2.5 million parameters, compared to the 11.7 million parameters of the FC models. We call this more efficient multilayer perceptron eMLP.
Furthermore, we implement a more sophisticated network architecture and compare it to the MLPs. Since the high-level features are fundamentally a list of related physical variables that represent different parts of the LHC event, an architecture with an attention mechanism is chosen, namely the Transformer structure from [87]. This architecture was first proposed for text translation tasks, and hence its application to this task is not obvious. However, the application of neural networks from image recognition tasks to jet images, which are very different from classical photographs, has already shown that such approaches might be fruitful.
Since the original encoder decoder setup from [87] is used for translation tasks (meaning generative tasks), we focus on the encoder part of the Transformer. The hypothesis is that the encoder structure will learn an embedding of the kinematic variables that can easily be read out with an MLP. Such encoder-only setups are

also commonly used in natural language processing, for example, in text classification or sentiment analysis. An example of such an encoder-only setup is the BERT model [89], which inspired this implementation. As the structure of the encoder part follows [87] exactly, we reference this paper for the details and describe only the free parameters in this thesis.

A Transformer encoder with six encoder layers and eight attention heads is implemented with a model dimension of 128 and a feed-forward dimension of 256. To construct an initial embedding with the model dimension, the kinematic data (106-dimensional) is padded with zeros and then fed directly into the encoder. After the encoder layers, the output is fed through a simple MLP to project to the output dimension.

6. Results

For the new dataset generated during the work on this thesis, over 1.5 billion LHC events have been simulated. This has resulted in a dataset of almost one million simulated LHC events that have passed the preselection cuts, as discussed in sec. 4.3. We partition the data into the training-, validation- and test set according to a 70/15/15 split. This reduces the relative size of the test set from 20% to 15% compared to [15], however, with 142,500 total test- and validation-samples, we still have more than five times the amount of data points than in [15], further increasing the statistical robustness. Moreover, we have 665 thousand training samples, which is eight times the original amount. All sets are composed of 50% signal and 50% background events. Each background process is weighted with the cross section to determine its respective portion in the dataset. For the training and validation set, the signal part of the dataset is composed of all signal mass points, ranging from $400 - 1000$ GeV in steps of 100 GeV in equal parts. The test set uses the same background mix but uses only one signal mass. The models are then successively evaluated with a test set containing each signal mass.

To compare the different neural network architectures, we proceed in two stages. Firstly, we select the model checkpoint with the lowest validation loss from all optimization runs of one architectural class. For this, we then calculate the network scores (logits) for all test set events. Then we can calculate the Receiver Operator Characteristic Curve (ROC-curve), which is commonly used to compare classifiers. The ROC-curve plots the true positive rate against the false positive rate for different threshold settings. The threshold denotes above which assigned probability an event is classified as a signal-event. The most basic threshold would be to consider everything that has an assigned signal probability of more than 50% to be a signal; however, in practice, we chose this threshold to optimize for the discovery and exclusion bounds. This will be discussed in more detail later. The ROC-curve gives a measure to quantify the discriminatory power of a model. Furthermore, the Area under the Curve (AUC) of the ROC-curve is a commonly used metric. It is 1 for a perfect classifier and 0.5 for random guessing. The AUC, whilst useful, should, however, not be taken as the predominant metric for our tasks since we select only one point on the ROC-curve as our working-point (WP). It should be noted that we will show an altered version of the classical ROC-curves, as is often done in particle physics (see ROC-curves in [15, 27]). When we refer to a ROC-curve, we will always mean a plot of one over the false positive rate against the true positive rate. This has

the physical interpretation of plotting the background rejection against the signal efficiency. The AUC refers to the area under the classical ROC-curve.
We use the ROC-curve to select the best model for each representation of the LHC-data and perform a physical analysis. For this, we calculate the 2σ exclusion bound and the 5σ discovery reach [90]. For the 2σ exclusion

$$Z_{\text{exc}} = \sqrt{-2 \ln \left(\frac{L(S+B|B)}{L(B|B)} \right)} \geq 1.64, \tag{6.1}$$

is required, where $L(x|n)$ is the likelihood of observing n events, where x events are expected. S and B are the number of signal and background events. For the discovery reach

$$Z_{\text{dis}} = \sqrt{-2 \ln \left(\frac{L(B|S+B)}{L(S+B|S+B)} \right)} \geq 5, \tag{6.2}$$

is required analogously. As mentioned above, one has to choose a threshold for the predicted probabilities to accept them as a signal. This is usually done in a way that maximizes eq. (6.2) [91], however, in [15] it has already been shown that this leads to cuts with very few background events remaining. This would then jeopardize the statistical validity of eq. (6.2) and eq. (6.1) and hence we adopt the same procedure as in [15] and impose a fixed number of background events that have to remain after the working-point has been selected and rescale the cross section iteratively. We have chosen a threshold of 20 background events, which corresponds to the working point at a background rejection of $\frac{1}{\epsilon_B} = 97$. For details on the imposed background threshold, we refer to appendix A.2.
It should be noted that there have been significant technical difficulties during the work for this thesis. The high-performance computational cluster at the University of Würzburg has experienced multiple outages during this time. The cluster was, however, vital for the event generation as well as the neural network training. This led to the necessity of cutting down on the computational time spent on the network optimizations. Furthermore, there are several further tests regarding some architectural choices that had to be cut out entirely.
Most importantly, one would like to average the results of different model checkpoints for each network with the same hyperparameters but different initializations. This would allow quantifying the run-to-run variance that is present in most neural networks. Unfortunately, this was simply not feasible, as it would require, for example, training each network around 20 times, as was done in [15]. We limit ourselves to the investigation of the run-to-run variance of a ResNet-20 model, by training

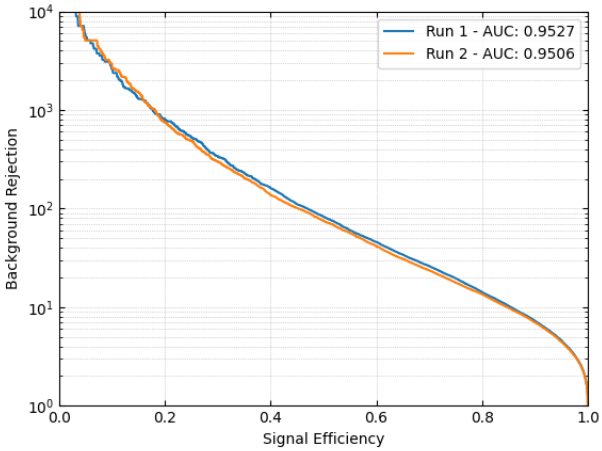


Figure 17: Comparison of two differently initialized trainings runs of the ResNet-20 model. There is a slight difference in the ROC-curves. Whilst it would be optimal to average the results of multiple runs to quantify the run-to-run variance, the differences between runs are small enough to compare different network architectures based on a single run.

two separate instances of the model with different initializations, and compare their ROC-curves in Figure 17 and conclude that the run-to-run variance has a visible but small impact on the ROC-curve. However, we suggest a more comprehensive analysis of the run-to-run variance of all models in this thesis for further work.
Finally, we have to mentioned that the interpretation of neural network results can be very challenging, due to their black-box nature. The inner workings of neural networks are not easily understood, and a thorough explanation of those mechanisms is beyond the scope of this thesis. In the analysis in the following sections, we will offer explanatory approaches for the results. However, there may still be additional hidden factors that influence the network results.

6.1. Model instability

Before we present the results of the different classifiers presented in this thesis, we address an anomaly that has manifested during the training of the CNN models in [15]. The models from [15] exhibit unusual behavior in the validation loss curve during the training period. An oscillation of the validation loss is observed, whereby the validation loss oscillates from low values to values up to an order of magnitude higher than its previous value, as can be seen in Figure 18. The training loss

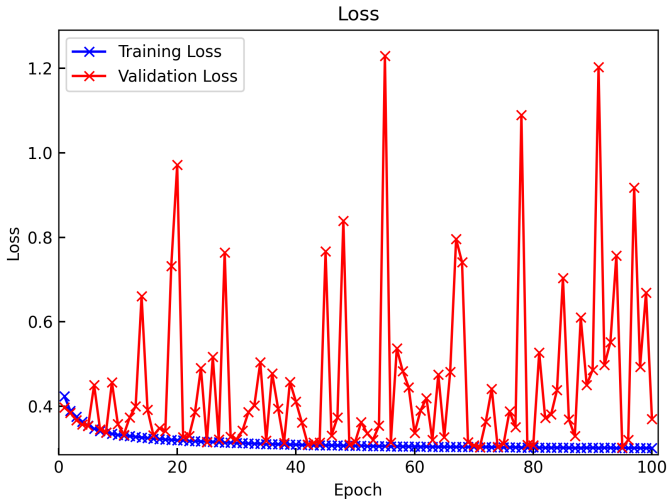


Figure 18: Oscillatory behavior of the CNL model form [15]. The validation loss drastically changes from epoch to epoch, oscillating between states that achieve a low validation loss and states that perform badly.

decreases as expected. This behavior is very atypical for well-tuned models and signifies an instability in the learning process. In the following chapter, potential reasons behind this instability will be examined.

One of the first approaches is to check if this behavior is dependent on the hyperparameters of the models. An extensive search across the hyperparameter space shows that this is not the case. The examined hyperparameters include the learning rate, batch size, weight decay as well as the optimizer itself. The aforementioned parameters have been varied over multiple orders of magnitude. However, it was found that, if the model trained well, the oscillations manifested. If a suboptimal set of hyperparameters was chosen, the instability did not always manifest itself. Instead, the validation loss did not reach as low values in the examples with the oscillations. Or, in the case of a very low learning rate, for example, the oscillations only showed up very late in the training process, e.g. only after multiple hundreds of epochs.

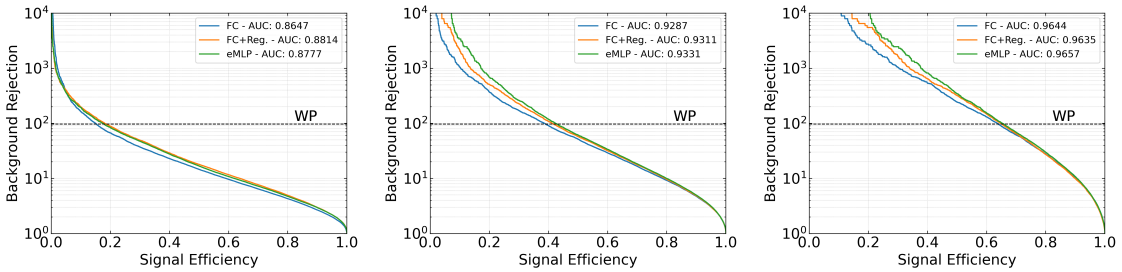
This means whether the anomaly manifest is dependent on the region of optimization space and not the value of the hyperparameter. With the lower learning rate, the training simply takes longer to reach the optimal region in the optimization space. It does not, however, change the oscillatory behavior. A similar behavior has been found for the batch size. With an increased batch size, fewer updates of the model parameters are performed, meaning that training needs to be done on more epochs to achieve comparable results. As in the tests for the learning rate, the anomaly manifests at higher epoch numbers. In conclusion, it can be said that the

hyperparameters cannot be solely responsible for the instability during training.
As mentioned earlier, a common problem in training deep neural networks are exploding or vanishing gradients. A strategy to overcome this problem has been proposed in [92]. There, the norm of the gradients is restrained to a maximal allowed value, mitigating the risk of exploding gradients. This is called gradient clipping.
If the gradient $\vec{g}$ has a norm $\|\vec{g}\|$ that is larger than an upper limit Λ the gradient is rescaled as $\hat{g} = \frac{\Lambda}{\|\vec{g}\|}\vec{g}$. This has been implemented in the training process of the CNNs used in this thesis. Here, it was found that if the value of the maximal gradient is chosen to be too small, the model trains slowly. This makes sense since smaller gradients mean that there is less distance traversed in optimization space. Hence, it takes longer to find better network configurations.
However, the gradient clipping does not prevent the oscillation of the validation loss. This behaviour is still observed at higher epoch numbers with the smaller maximally allowed gradients, and a higher value for the clipped gradients does not change the training behavior significantly.
Another reason for the instability during the training process could be the structure of the jet image data. Those images are mostly empty, as can be seen in Figure 7. In the case of the CNNs, this could lead to instability because it creates a kind of binary input to the network as a single pixel is either completely off or has a finite value. This is very different from the usual application of CNNs in computer vision, where every pixel has an RGB value and is hardly ever empty. This can make it hard to learn interesting feature maps that correspond to interesting features of the images. It is plausible that this is much more difficult with the hard features of the detector images.
To combat this issue, one can implement a form of data augmentation. A Gaussian blur is applied to the image during the training process. This corresponds to a sort of washing out of the detector hits, leading to fewer empty pixels in the jet image and softer features. Different sizes of the Gaussian kernel are tested, and the standard deviation of the Gaussian distribution used in the kernel is drawn at random before it is applied to one specific image. This further diversifies the features of each detector image. To limit the necessary computation time, this process is done only once before the training. So each image always has the same blur. However, this procedure did not significantly decrease the observed instability during the training process and does not increase model performance.
It is clear that the oscillatory behavior is not model-dependent, since all CNNs described in this thesis display this behavior. The above tests have been performed

for the ResNet and CNL models.
In conclusion, the anomaly does not depend on any hyperparameters of the model or training process. Therefore, the reason must lie elsewhere. To rule out that this behavior is inherent to the model, a new convolutional network has been implemented. This means that the anomaly is not dependent on either the hyperparameters or the model itself. This arguably means that this behavior is a consequence of the training data. It could either be that this behavior is intrinsic to this classification task or that it is a consequence of too few training samples. This could mean that the anomaly could disappear if the models are trained on a bigger dataset.
However, as we have described in section 4.3 we have created a dataset that is almost an order of magnitude bigger than the one used in [15]. The anomaly is present in both cases, making this unlikely.
Furthermore, we have found that the GAT type GNNs also display this behavior, even though to a lesser degree. The GCN network does not exhibit this behavior. As we have described above, the anomaly only manifests when the CNN models start to train well and the validation loss falls below a certain point. It is possible that the GCN model never reaches this point and therefore does not display this behavior. The GAT networks reach a performance that is comparable to the CNNs and also display this oscillatory behavior. However, it is less pronounced.
This leads us to believe that the oscillations might be intrinsic to this classification task. It should, however, also be mentioned that this should not be detrimental to the results that can be achieved with the classifiers presented here, since the model configurations that achieve a low validation loss are selected for the analysis. In [15] it has already been shown that the network performance also generalizes to the testset.

6.2. Classification based on high level kinematic features

We compare the different MLPs using the kinematic data in Figure 19. We take the MLP from [15], termed FC, as a baseline, however since this implementation does not have any batch normalization or dropout layers we also compare a version of this model with those added regularization layers. It is clear that the addition of regularization layers to the in MLP proposed in [15], increases test-set performance. Furthermore, the MLP implemented in this work (denoted as eMLP in Figure 19) achieves the same performance, with significantly fewer parameters and shorter training times as is elaborated in appendix A.1 and Table 8. With only



(a) $m(S^{\pm\pm}) = 300$ GeV (b) $m(S^{\pm\pm}) = 600$ GeV (c) $m(S^{\pm\pm}) = 900$ GeV

Figure 19: The ROC-curves of the MLPs trained on high level kinematic data. The addition of regularization layers significantly improves performance in the MLP described in [15] (called FC) for all signal masses. The newly implemented eMLP achieves the same performance with significantly fewer parameters and shorter training time.

around 20% of the parameters and an almost ten times shorter training time, the new model is significantly more resource efficient.

One could argue, that the performance of this new eMLP is even slightly better for higher signal masses, however this effect is minimal and might fall into the range of run-to-run variance. Since this network comes with reduced computational overhead we choose this model for further analysis and present the expected 95% confidence level upper limit on the cross section σ (exclusion limit) and the discovery reach cross section σ_5 of this model compared to the baseline in Figure 20, where we also display the cross section of the doubly charged scalar production of the $\frac{\mathrm{SU}(5)}{\mathrm{SO}(5)}$ model at 14 TeV [15]. This reference will be present in all further physical analyses in this thesis. As one can see, the added regularization and new network architecture improve the expected exclusion limit and discovery reach only slightly. The upper limit on the signal process cross section is improved to 670 GeV and the discovery reach to 450 GeV, which constitutes an improvement of around 20 GeV for both compared to the FC network. In general, we see that the expected exclusion limit and discovery reach get stronger for higher masses. This is due to the increased model performance for higher masses, as seen in 19. And the increase in the preselection efficiency $\varepsilon_{\text{preselection}}$ for higher masses. In addition to the MLPs, we have also implemented a Transformer architecture as described in section 5. However, this model has unusual training behavior that is not understood and produces results that are unreasonably good. Since we do not understand the training behavior of this model completely and thus do not trust the results completely, we present

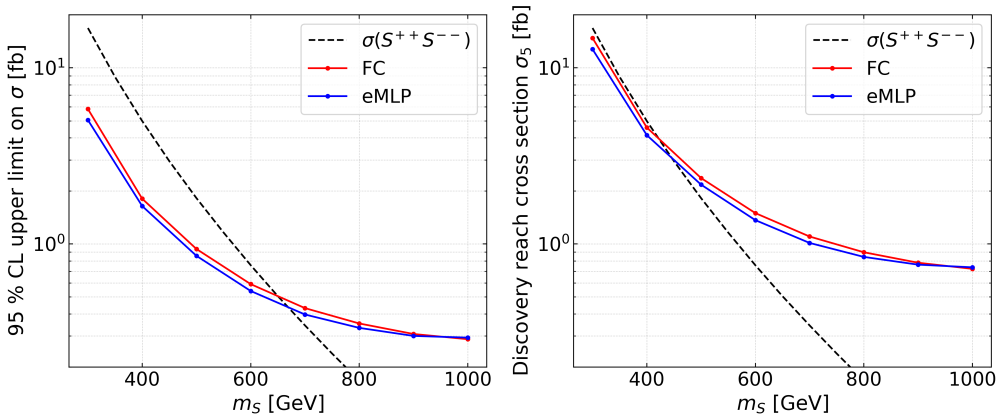
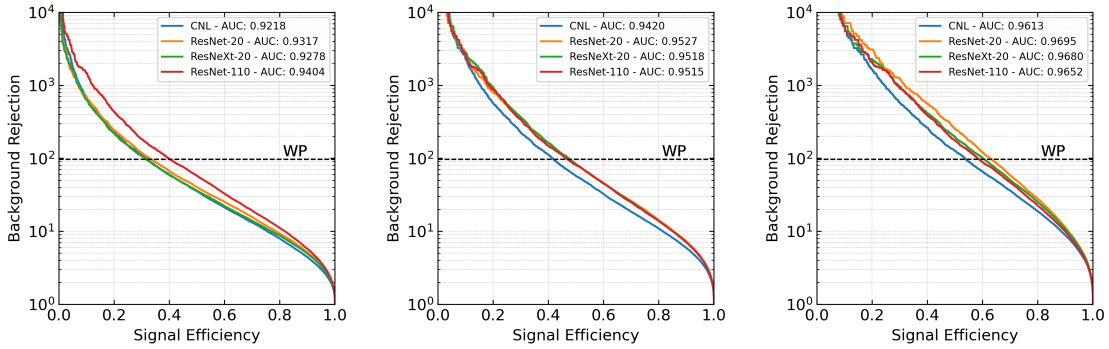


Figure 20: **Left:** Expected exclusion limit of the signal process at the high luminosity LHC with 3ab^{-1} integrated luminosity for the MLP networks. The eMLP raises the expected exclusion limit to 670 GeV.
Right: Expected discovery reach for the $S^{\pm\pm}$ production is increased to 450 GeV by the eMLP.
The dashed line is the cross section of the doubly charged scalar pNGB production from the $\frac{\mathrm{SU}(5)}{\mathrm{SO}(5)}$ model at 14 TeV [15].

the results in appendix A.3 with reservations and do not include the Transformer analysis in the main part of this thesis.

6.3. Jet image classification

We present the comparison of the different CNNs. Figure 21 shows the ROC-curves of the CNNs for three different signal mass points. While the exact hierarchy of the model performances varies, there are clear trends visible. The ResNet architectures outperform the CNL architecture. In general, the models perform better for higher masses. This is most likely due to the fact that the higher signal masses produce events that are significantly more energetic (see Figure 8) and hence can be better distinguished.
It might be surprising that the ResNet-20 outperforms the deeper ResNet-110 for higher mass points; however, this might be due to the fact that the bigger networks start to overfit more easily at higher masses. This problem might be more pronounced for higher masses since, those are easier to seperate in the first place. Since more complex models overfit easier on less challenging tasks, this might explain why this effect is only observed for the higher masses. During the training, the dataset consists of a mix of all signal masses, hence the effect of overfitting on

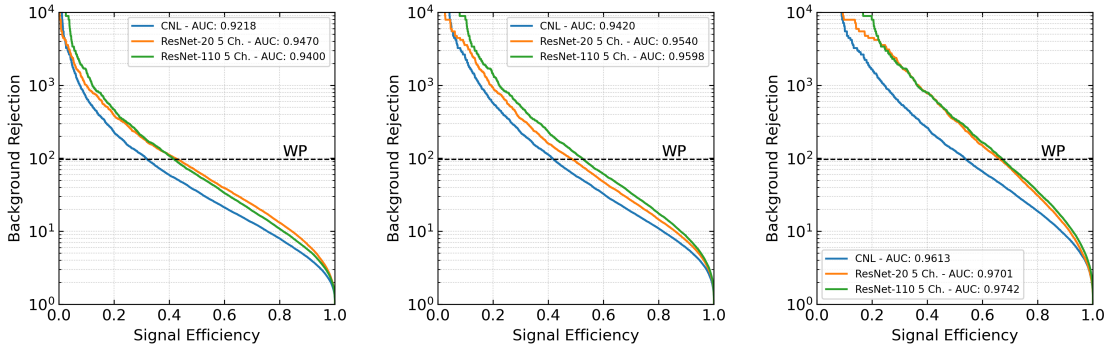


(a) $m(S^{\pm\pm}) = 300$ GeV (b) $m(S^{\pm\pm}) = 600$ GeV (c) $m(S^{\pm\pm}) = 900$ GeV

Figure 21: The ROC-curves of the convolutional neural networks. The ResNet architectures outperform the CNL model. For lower signal masses the ResNet-110 performs best, whilst for high signal masses the ResNet-20 has the best performance.

higher masses is not so obvious during training, while we evaluate for each mass separately. We have implemented numerous measures to counteract overfitting, as discussed in earlier sections; however, it is often not possible to completely prevent it. For low signal masses, the ResNet-110 architecture outperforms the ResNet-20, where the ResNet-20 model performes more comparable to the base-line CNL model for lower signal efficiencies. The advantage of the ResNet-20 is in this case only visible for high signal efficiencies. The performance of the ResNeXt-20 model is, surprisingly, not better (or slightly worse) than the performance of the ResNet-20 model. However, it needs to be mentioned that this can be due to computational limitations at the end of the work for this thesis. The high performance computation cluster of the university of Würzburg had outages around this time, which limited the hyerparameter optimization. Since the ResNeXt-20 model requires more computational resources than the ResNet-20 and the optimization was started after the other models, it got less optimization time than the other networks. However, we do not expect that a significant improvement in performance could be achieved with more optimization time, rather we expect the general trend to be the same, with the depth of the models being most important. Training an implementation of, for example, a ResNeXt-110 network lies beyond the scope of this thesis.

As described in section 4.1 we have calculated additional input channels for the CNN networks, that contain the b-tagged and light jet-center positions. We adapt the ResNet-20 and ResNet-110 models to expect five input channels and retrain the



(a) $m(S^{\pm\pm}) = 300$ GeV (b) $m(S^{\pm\pm}) = 600$ GeV (c) $m(S^{\pm\pm}) = 900$ GeV

Figure 22: The ROC-curves of the convolutional neural networks. The ResNet architectures outperform the CNL model, and the addition of jet-center images as input to the CNNs further increases performance.

models with the extended jet image dataset. The impact of introducing the new jet-center images is shown in Figure 22.

There it can be seen that this further increases network performance and is most significant for a signal mass of 600 GeV. At lower signal masses, the ResNet-20 with 5 input channels performs better than the ResNet-110 (5 Ch.) for higher signal efficiencies but worse for lower signal efficiencies. For higher signal masses, the ResNet-110 (5 Ch.) performs better for higher signal efficiencies, but otherwise the networks are similar.

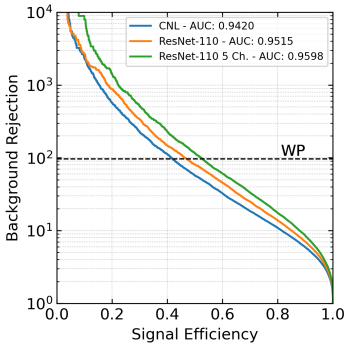


Figure 23: ROC-curve of the ResNet-110 with 3 and 5 input channels compared to the CNL model. The ResNet-110 outperforms the CNL baseline and the addition of the jet-center images further improves the performance. It can be said, that the addition of kinematic data into the jet images is as important as the model choice.

The mass point of $m(S^{\pm\pm}) = 600$ GeV is of especial interest, since it is near the middle of the region where the discovery and exclusion bounds, that have been calculated in [15], lie. Therefore, it is sensible to judge the networks based on this point, since the performance differences in this region are important for the physical analysis. Hence, the sensitivity of the models in this region is arguably of higher import to the bounds that can be achieved. Especially at this mass point, the addition of the b- and non-b-tagged jet-centers seems to be important for the increased model performance of the ResNet-110. Furthermore, it can be said that the addition of the jet-centers is as important as the architecture choice itself. This is illustrated for the signal mass of 600 GeV in Figure 23, where the effect is strongest.
Since the ResNet-110 model with 5 input channels generally performs the best and especially in the 600 GeV signal mass region, we chose this model for all further analysis performed in this thesis and present the 95% upper confidence limit on the cross section σ and the discovery reach cross section σ_5 of the ResNet-110 compared to the CNL model in Figure 24.
It can be seen that the CNNs also follow the trend of imposing stronger bounds and increasing the discovery reach for higher masses, as discussed in the previous section.

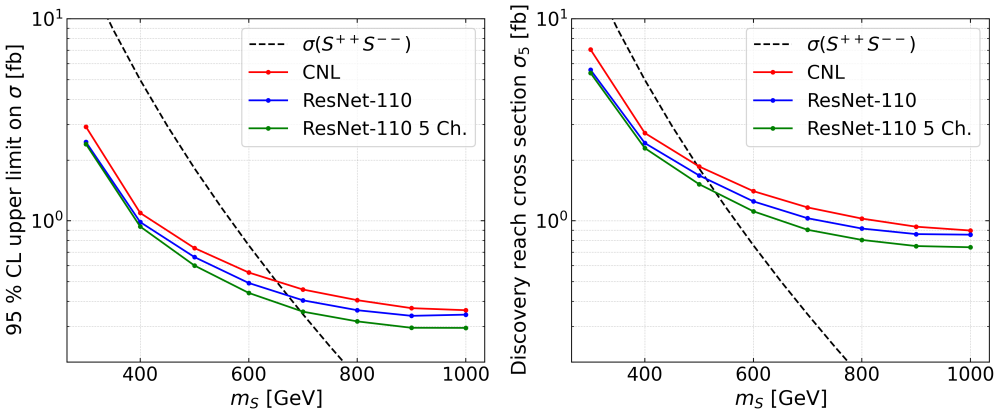
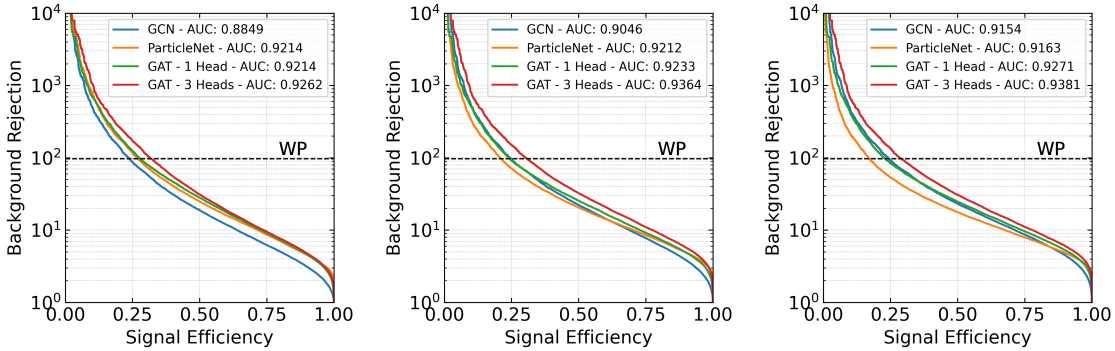


Figure 24: **Left:** Expected exclusion limit of the signal process at the HL-LHC for the CNN networks. The ResNet-110 architecture including the jet-center images improves the exclusion limit by around 50 GeV compared to the CNL baseline.
Right: Expected discovery reach for the $S^{\pm\pm}$ production is increased by 30 GeV through the ResNet-110 architecture with additional jet-center images compared to the CNL baseline.

The ResNet-110 (5 Ch.) model produces an expected exclusion limit of almost 700 GeV, whereas the CNL model can only exclude the process up to 650 GeV. With the ResNet-110 (5 Ch.) the signal process can be discovered up to 530 GeV, compared to the 500 GeV with the CNL model.
It should be noted that we have found that the CNN networks in general perform worse on the new dataset, compared to the results of [15]. This might be due to the differences during the simulation process, as well as different versions of the Monte-Carlo generators that were used. It was however also found that a CNN can distinguish the two datasets for one background process. We elaborate on this issue in appendix A.4.

6.4. Graph neural networks



(a) $m(S^{\pm\pm}) = 300$ GeV (b) $m(S^{\pm\pm}) = 600$ GeV (c) $m(S^{\pm\pm}) = 900$ GeV

Figure 25: The ROC-curves of the graph neural networks. The GAT network with three attention heads outperforms all other networks.

We compare the in sec. 5.2 described models and present their ROC-curves in Figure 25. At a first glance, the graph convolution network (GCN) performs the worst across all mass points, as judged by the AUC metric. This is not surprising, since, with only one layer and around 60,000 parameters, it is a very simple network. However, this type of GNN has the advantage that it is fast to train and easy to optimize, which is a significant challenge for the other GNNs compared in this thesis. For details concerning the training time and number of parameters of the models, we refer to appendix A.1. This worse performance is mostly due to the low discriminatory power at high signal efficiencies. For lower signal efficiencies, the GCN performs comparable to the GAT with one attention head (GAT-1). For the lower signal masses up to $m(S^{\pm\pm}) = 600$ GeV, this is however not the case. There, the GCN

performs strictly worse than the other networks. This performance is nonetheless surprisingly good for such a simple model, since the lower signal efficiency region is more important for the calculation of the physical bounds. The GCN even outperforms the ParticleNet for higher signal masses at the working point.

While the AUC of the ParticleNet is on par with the GAT networks, the performance for low signal efficiencies is worse. Especially around the working point and for higher signal masses, the ParticleNet performs worse than the other networks. The GAT network with three attention heads (GAT-3) outperforms all other networks consistently and is therefore chosen for all further evaluations in this thesis. In Figure 26 we show the exclusion limit and the discovery reach of the GAT-3 network compared to ParticleNet. We also present the analysis for the GCN network, due to its surprising performance for its low model complexity. For all models, the expected exclusion bound and discovery reach start to rise again for higher signal masses. In contrast to the CNNs, the GNNs have more discriminatory power for lower signal masses rather than higher ones. This can already be seen in Figure 25, where the performance of the networks decreases at the working point for higher signal masses. As we have discussed earlier, the higher mean p_T in the signal events is an important feature for the networks to discriminate signals from the background.

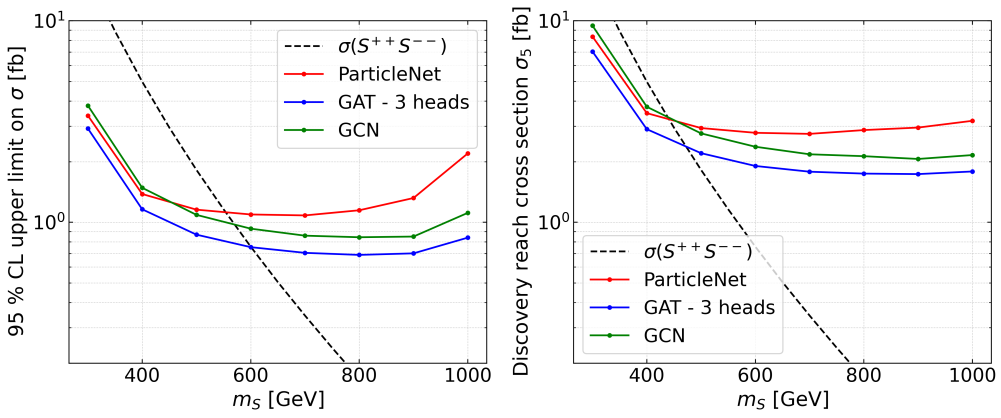


Figure 26: **Left:** Expected exclusion limit of the signal process at the HL-LHC with 3ab^{-1} integrated luminosity for the GNN networks. The GAT-3 architecture increases the expected limit by almost 50 GeV compared to the ParticleNet. The GCN slightly beats the ParticleNet.
Right: Compared to the ParticleNet baseline, the expected discovery reach for the $S^{\pm\pm}$ production is increased by 30 GeV by the GAT-3 model. The GCN is on par with ParticleNet.

In the kinematic and jet image representations, this information is easily accessible to the networks. However, in the graph representation, this information might not be so readily available for the networks due to their local nature. Each node in the graph aggregates information over only a limited number of the other nodes in the graph, where the number of layers determines the depth of the aggregation. While there is also a global readout to project down to the output dimension of the networks, this could make the aggregation of the total p_T information of an event more difficult. This can explain why network performance does not increase for higher masses. Giving an exact reason for why network performance gets worse for higher masses is, however, difficult due to the black box nature of neural networks. Further investigation would be required. The preselection efficiency rises for higher signal masses, which would result in stronger bounds and a higher discovery reach for high masses if this effect were not counteracted by the worst network performance.
In general, the graph networks perform worse than the CNNs, only allowing discovery up to around 470 GeV and exclusion up to 600 GeV, for the GAT-3 model. The GAT-3 model improves the discovery reach by around 30 GeV compared to ParticlNet and the GCN. For the exclusion limit, the GCN improves upon the ParticleNet by 20 GeV and the GAT-3 enhances the exclusion bound by almost another 30 GeV.

6.5. Combining multiple classifiers

As has already been shown in [15] it is beneficial to combine multiple classifiers that work on the different representations of the events via an MLP to produce a combined prediction. In [15] this was done by simply concatenating the outputs of the last neuron layers before the projections to the output-dimension and training both networks conjointly. Since we have investigated a new representation of the event, namely the point clouds, it is sensible to construct a combined classifier that works on all three representations at once.
We hypothesize, that the addition of the graph networks for the point-cloud data will further increase the predictive power of the model, since the graph networks focus on the local substructure of the jets with their sparse connectivity and few GNN layers. In contrast, the kinematic-models work on the already abstracted high level features of the event and the deep convolutional networks have a high enough field of view to learn the global structure of the jet images. We refer to the combined classifier that works on all three representations, as multi representation classifier (MRC).

Since there are quite a few models for each representation, we restrict the analysis to a "best in class" approach. For each type of representation, the best performing model is chosen and combined with the others, to form a single classifier. To compare the influence of the different representations on the network performance, we also compare all two network combinations.
However, the training of such complex combined models with more than 35 million parameters is quite challenging and also comes with immense computational requirements. Instead of training the combined model with freshly initialized parameters, the parameters of the already optimized and trained single representation models used for the previous analysis are used. Since those models project the latent-space information down to the two-dimensional output space, two options present themselves.
Firstly, one could only combine the two-dimensional output of all models and feed this forward through a combining MLP, that learns the best way of weighting this information as well as utilizing potential correlations of the network outputs. In this way, only the combining MLP needs to be trained, while the other network-parameters are frozen. This significantly reduces the training time and is rather convenient to implement but poses the risk of losing a lot of valuable information contained in the latent-space representations learned by the classifiers.
To avoid this, we chose a different approach, which we visualize in Figure 27. We

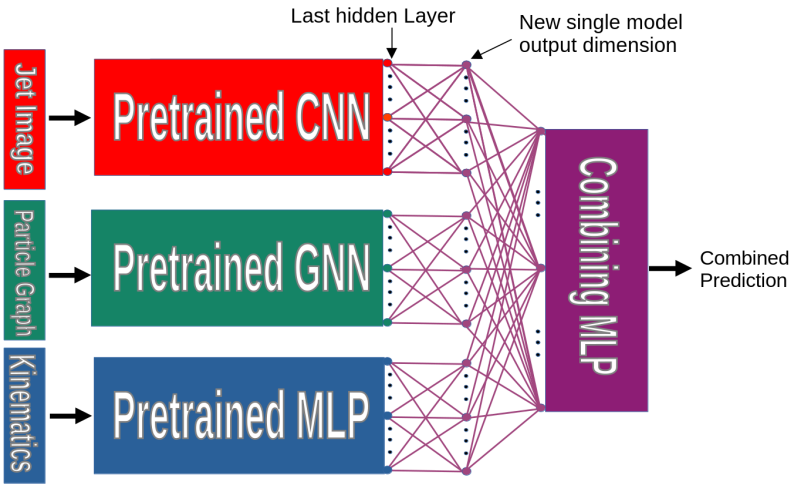
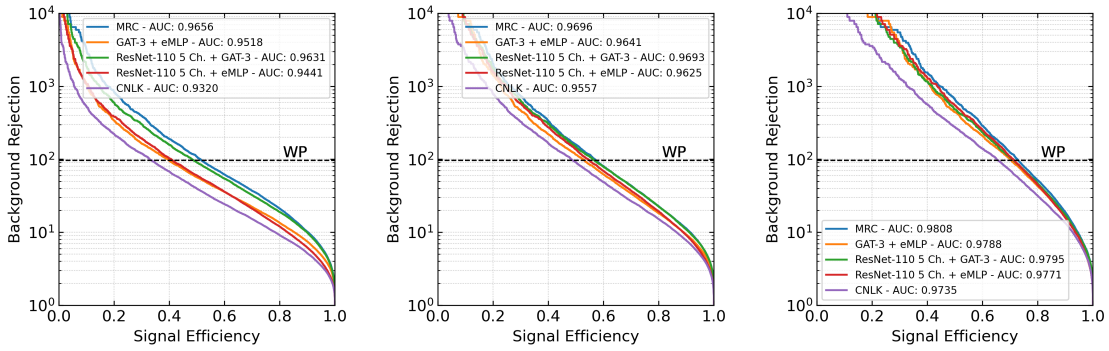


Figure 27: The multi representation classifier (MRC) leverages the pretrained single representation models. The last two-dimensional output layer of each model is substituted with a higher dimensional hidden layer. The models are then combined with an MLP to produce a single output.

load all wights except the last layer that projects down to the two-dimensional output. This layer is instead replaced with a newly initialized layer with higher dimensionality. We do this by extracting the trained parameters from the single representation networks and update the parts of the combined classifier that correspond to the single representation networks.

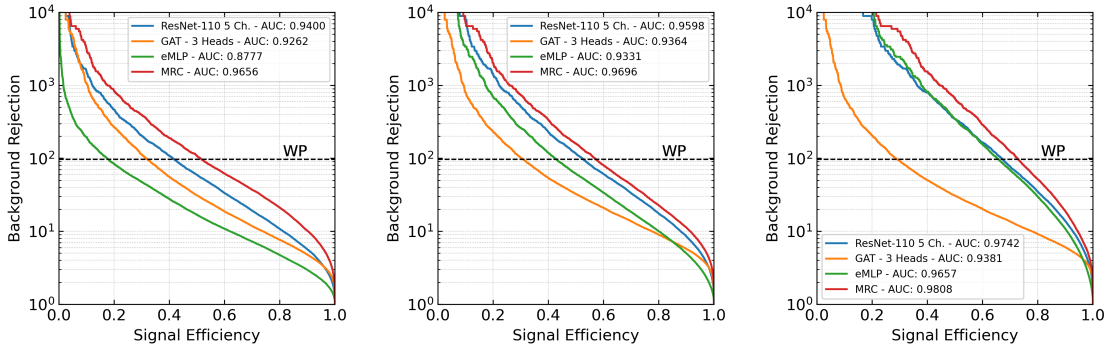
This allows the combined classifier access to the latent-space information of all models. We then proceed to freeze all pretrained parameters and train the combined classifier. The MLP used to combine the latent space information of the single representation models consists of three linear layers with BatchNorm and dropout. It learns rather quickly and only few epochs without any scheduling are needed to achieve good network performance. The CNLK model from [15] that uses the kinematic and jet image representation, is used as a baseline for our comparison. It combines the convolutional neural network CNL and the multilayer perceptron FC. We optimize and train this network on our larger dataset for a fair comparison. We present the ROC-curves in Figure 28.

We see that all combinations beat the baseline CNLK model for all masses. Further, the impact of adding the eMLP working on the high level kinematic data to one of the other models increases the performance less than adding the graph based model and the image based model. This leads us to believe that the graph and image based models do indeed learn complementary information about the events. The



(a) $m(S^{\pm\pm}) = 300$ GeV (b) $m(S^{\pm\pm}) = 600$ GeV (c) $m(S^{\pm\pm}) = 900$ GeV

Figure 28: Combining multiple classifiers leads to improved results. The combination of graph and image based classifiers significantly improves performance, indicating the networks learn complementary information from the events that results in better classification. Combining models that work on all three representations leads to the best results.



(a) $m(S^{\pm\pm}) = 300$ GeV (b) $m(S^{\pm\pm}) = 600$ GeV (c) $m(S^{\pm\pm}) = 900$ GeV

Figure 29: The comparison of the single models to the multi representation classifier (MRC), shows that this approach leads to significantly improved results for all signal masses. The advantage of the MRC gets smaller for higher masses as the CNN and MLP models start to perform similarly. The inclusion of the GNN still improves performance, even though its performance diminishes for higher masses.

graph based models work on the constituent level jet-substructure, while the deep ResNet-110 sees the whole image and is able to extract more global features. The kinematic representation seems to be more similar to the jet images than to the graph representation, as it is also a high level analysis of the event. This also manifests in the similar behavior with increasing signal masses of the respective models. Both network types gain discriminatory power for higher masses, whereas the graph based models exhibit contrary behavior.

The MRC based on all three representations increases performance further. However, the significant part of the performance is gained by combining graph and image based models. Figure 29 highlights the importance of combining the representations. We see that the increase in performance by combining different physical representations as input to a neural network is as significant as the choice of network itself. This advantage persists even at higher signal masses, where the eMLP and ResNet-110 start to perform similarly at the working point. There, the graph based model still contributes useful information to the classification process.

In Figure 30 we present the exclusion bound and discovery reach produced by the MRC compared to the CNLK model. We see that the CNLK model excludes the signal process up to 680 GeV and the MRC increases this upper limit by around 30 GeV to 710 GeV. The discovery reach is increased from 520 GeV to 550 GeV by

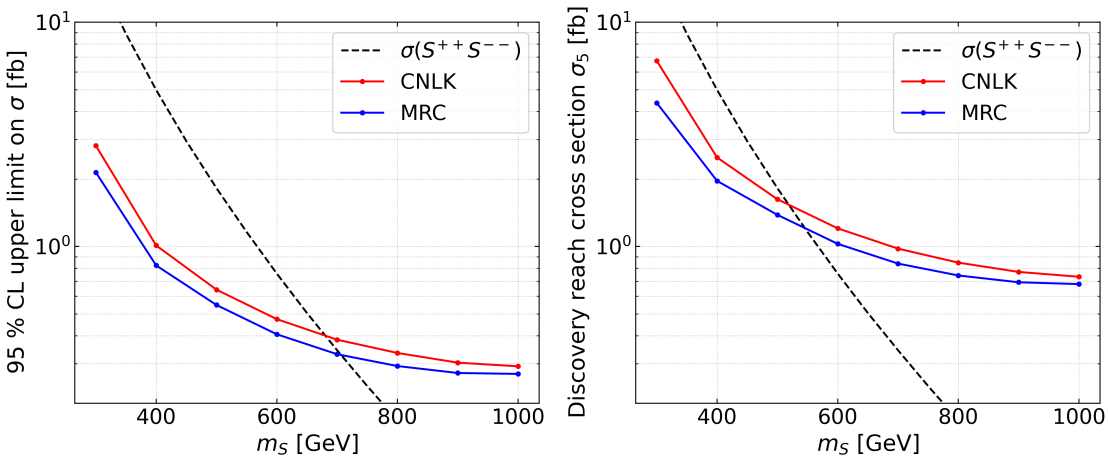


Figure 30: **Left:** The MRC increases the expected exclusion limit to 710 GeV, which constitutes a 30 GeV increase over the CNLK baseline.
Right: The expected discovery reach for the signal process is also increased by 30 GeV by the MRC. This allows the discovery up to 550 GeV.

the MRC. The exclusion limit is not increased significantly by the MRC compared to the ResNet-110 (5 Ch.) as the MRC only achieves a 10 GeV higher exclusion limit. This is mainly due to the fact that the MRC does not increase the network performance that much for the mass region from 500-700 GeV. The performance improvement is more significant for the higher and lower signal masses. Furthermore, the ResNet-110 (5 Ch.) already has access to a part of the kinematic data in form of the jet-center images. Nonetheless, the MRC achieves the highest exclusion limit and discovery reach of all models compared in this thesis.

7. Conclusion and outlook

Composite Higgs models produce beyond the standard model signal processes that are accessible to searches at the high luminosity LHC. In this thesis we have studied the decay process of a doubly charged scalar $S^{\pm\pm}$ that decays to a $W^+t\bar{b}$ ($W^-\bar{t}b$) system, motivated by such composite Higgs models. We focus on a search targeting a final state with high hadronic activity and two same-sign isolated leptons. For this, we simulate a dataset that contains three different representations of each event. We compare different deep neural networks for each representation and surpass our benchmark in each case. Our dataset is larger and therefore provides better statistics and allows for easier training of more complex networks compared to previous work done in [15].

We represent each LHC event in distinct ways that allow us to access different features contained in each representation, allowing for discrimination between signal and background processes.

The most abstract representation is a set of high-level kinematic data about the isolated leptons and reconstructed jets. We use a new MLP, called eMLP, on this representation, achieving the same performance as our benchmark with fewer parameters and training time. The benchmark is an MLP from [15] proposed for this task, which we first improve with the addition of regularization layers and then further surpass with the eMLP. This network achieves an exclusion limit of 670 GeV and a discovery reach of 450 GeV, which constitutes a 20 GeV improvement compared to the benchmark for both.

Furthermore, we implement a version of jet images that incorporates an artificial periodicity in the pseudorapidity, which allows the CNNs to access all physical detector information. In the process of our event generation, we have uncovered several issues with the detector simulation in [15] and worked to correct them. We have further enriched the information content of the jet images by adding channels that represent the b-tagged and light jet-center positions. We compare our ResNet and ResNeXt based models to the CNN (called CNL) proposed in [15] that was designed for this search. All our models surpass this benchmark, and the inclusion of the jet-center position further increases the discriminatory power of the ResNet models. We find the 110-layer deep ResNet-110 implementation with the added jet-center positions allows the discovery of the signal process up to 530 GeV and achieves an exclusion limit of 700 GeV, which is an 50 GeV and 30 GeV to the benchmark model, respectively.

The last representation of the LHC events are point clouds, on which we construct graphs. We compare the performance of a GCN and two GAT networks to ParticleNet [27] and find that a GAT network with 3 attention heads performs best. It can discover the signal process up to 470 GeV and exclude it up to 600 GeV.
Combining the best performing models into a single network (MRC), which can learn from all three representations, and leveraging the pretrained networks to make training such a complex classifier possible, yields further improvements to discovery reach and exclusion limit. We find that the upper limit on the model cross section increases to 710 GeV, constituting a 30 GeV improvement on the combined classifier CNLK from [15]. The discovery reach is also increased by 30 GeV compared to this network. The MRC allows the discovery up to 550 GeV.
We find that most of the performance increase is gained by combining the jet image based model and the graph based model, indicating that they learn complementary information about the events. The incorporation of the jet-center positions into the jet images already constitutes a combining of the high-level kinematic and image based event representations. The addition of graph based networks injects information about the local substructure into the classification process.
The choice of network architecture is clearly important and can improve performance on the classification task, as has been shown here. However, importantly, we also show that the way in which the data is represented and combinations of such representations that highlight different physical aspects of the event are just as important.
Even tough we have presented various machine learning methods and applied them to the doubly charged scalar signal process in this thesis, there is ample opportunity to expand on this work.
We have uncovered differences in the structure of the data of the $t\bar{t}t\bar{t}$ background in our dataset compared to the data used in [15]. This difference can be exploited by neural networks to discriminate between the two datasets, as we discuss in appendix A.4, and is most likely due to different Monte-Carlo generators used during part of the simulation chain. This poses the question of the importance of the choice of generator on the physical results. We suggest investigating this in further research.
The application of Transformer based models in this thesis yields results that are not completely understood yet. Nonetheless, we believe that the application of Transformers to the kinematic and point cloud representations could be beneficial and should be studied further.

Even though GNNs have not performed better than the other network types, the graph based representation allows for the encoding of additional physical knowledge into the graph structure. We have already seen that the addition of jet-centers to the images has yielded fruitful results. We suggest doing the same for the graph representation. By connecting the graph based on the reconstructed jets, more information about the jet substructure can be encoded into this representation. For example, one could connect the constituent particles to an additional node representing the jet itself and further connect the jet nodes to an event-level node, inducing a hierarchy into the graph that represents the different structures of the event.
Moreover, one could apply networks that can work on time-series, e.g., temporal graph networks, to evolving particle graphs that incorporate the angular separation of constituents as an artificial time. The graph connectivity would be determined by the angular distance of constituents and would increase with progression through this artificial time.
Finally, we note that, the approaches presented in this thesis are not limited to the search for doubly charged scalars originating from composite Higgs models. One could easily apply the techniques presented here to other beyond the standard model searches with busy final states. The application of some of the here discussed methods to jet-tagging is also feasible in light of publicly available jet-tagging datasets [93]. Especially, the combination of multiple physical representations is widely applicable.

A. Appendix

A.1. Network hyperparameters

Network	Learning Rate	Weight Decay	Dropout	batch size
CNL	9.6×10^{-4}	6×10^{-4}	0	179
CNLK	6×10^{-4}	1×10^{-4}	0	180
ResNet-20	5×10^{-5}	1×10^{-4}	0	350
ResNet-110	9.5×10^{-4}	1×10^{-5}	0	110
ResNeXt-20	2.5×10^{-2}	1×10^{-9}	0	457
ParticleNet	3×10^{-4}	1×10^{-4}	0	200
GCN	2.5×10^{-4}	3×10^{-7}	0.3	177
GAT 1 head	3×10^{-4}	1×10^{-9}	0.3	16
GAT 3 heads	3×10^{-4}	3×10^{-8}	0.05	21
FC	9×10^{-4}	8×10^{-9}	0	59
FC+Reg.	3.3×10^{-3}	3×10^{-10}	0.37	102
MLP	1×10^{-4}	1×10^{-5}	0.3	200
Transformer	1.1×10^{-5}	0	0.05	64
MRC	5×10^{-3}	2×10^{-8}	0.3	127

Table 5: Summary of the hyperparameters that were used for the final model checkpoints of the networks used in this thesis.

We summarize the hyperparameters used for training the neural networks in this thesis in Table 5. They have been determined through Optuna trials as described in section 5 and then manually fine-tuned.
The GAT networks use the LeakyReLU activation function instead of the ReLU used for all other models. The "negative_slope" hyperparameter μ associated with this has been set to $\mu = 0.05$ (determined through Optuna trials).
The learning rates listed in Table 5 are the initial learning rates at the beginning of the training. Most Networks make use of learning rate scheduling, as described at the end of section 3.1. We have used the Pytorch [74] implementation of the "StepLR" and "OneCycleLR" learning rate scheduler. In the following, we use the nomenclature used in the Pytorch documentation to refer to the different parameters associated with the schedulers and refer there for the details of the

Network	div_factor	final_div	anneal_strategy	three_phase	epochs
ResNet-20	3	15	cos	False	60
ParticleNet	10	3×10^3	linear	True	20
MLP	10	30	cos	False	35

Table 6: The "OneCycleLR" scheduling scheme used for the listed models. The maximal learning rate is the learning rate from Table 5 multiplied with the "div_factor".

Model	γ	Step size	Max epochs
GAT - 1 head	0.1	65	170
GAT - 3 heads	0.2	38	80
ResNet-110	0.1	35	80
GCN	0.5	45	100

Table 7: These models use the "StepLR" learning rate scheduler, which reduces the learning rate with a factor $\gamma < 1$ every fixed number of epochs (Step size).

scheduling procedure. We summarize the models using the "OneCycleLR" in Table 6 and the models using the "StepLR" in Table 7. All other models do not use any scheduling. The CNL, CNLK, FC and FC+Reg. models do not use scheduling as they function as the baseline comparison. For the ResNeXt-20 model, we could not implement a scheduling scheme, due to the aforementioned technical difficulties. We can see that the models that use the "OneCycleLR" require fewer epochs to train than the other models, proving this scheduling scheme to be effective here. However, it is more complicated to effectivly implement, as the parameters of the scheduler require more tuning. This is the reason why not all networks use this scheme, as it would simply require too much time to tune the parameters for all networks. On the other hand, the "StepLR" policy has proven to be easy and effective. It requires little tuning and almost always benefits the training process. However, with this policy the training usually takes longer, since the optimization process is open-ended, meaning in contrast to the "OneCycleLR" there is not an a priori fixed number of epochs. Rather, we let the training continue until there is no benefit in further reducing the learning rate.

Table 8 shows the required training time on a NVIDIA P100 GPU (NVIDIA GTX 970 for the highlited models) and the number of learned parameters for each network.

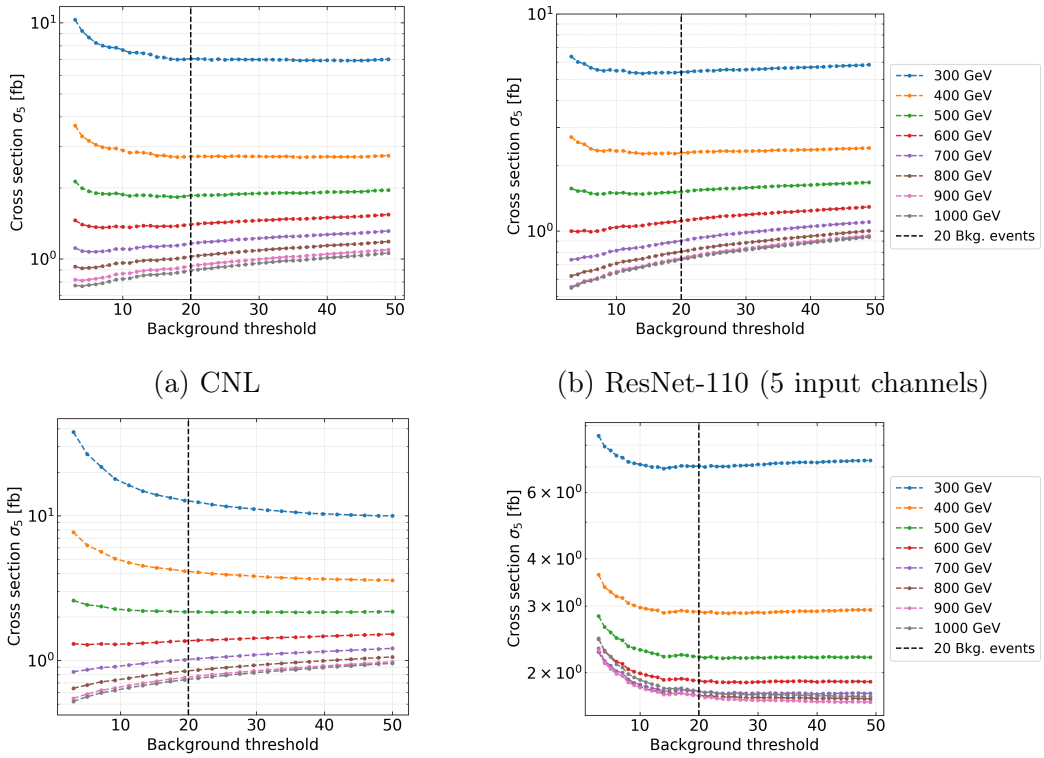
Network	Time per epoch in min	Parameters in 10^3
CNL	1.3	880
CNLK	1.6	6,195
ResNet-20	5.6	1,204
ResNet-110	24.6	7,023
ResNeXt-20	10.1	699
ParticleNet	16.9	367
GCN	1.3	60
GAT 1 head	13.7	1,985
GAT 3 heads	67.7	23,126
MRC	126.8	36,396
FC	2.3	11,694
FC+Reg.	2.6	11,712
eMLP	0.3	2,482
Transformer	2.4	945

Table 8: The number of learned parameters and the required training time per epoch for all networks studied in this thesis. The highlighted networks were trained on an NVIDIA GTX 970 GPU and all other networks were trained on NVIDIA P100 GPUs.

The GNNs require significantly longer training times due to their complex learning algorithms. In contrast, the MLP type architectures (MLP, FC and FC+Reg.) have a lot of parameters but require less training time (even on the significantly worse GTX 970) than the more complex models, due to their highly parallelizable structure and simple implementation.

A.2. Background threshold

Figure 31 shows how the discovery reach σ_5 varies with the chosen background threshold for a selection of models. In contrast to [15] we select the same threshold for all models. This eliminates the influence of the chosen threshold on the comparison of different networks. With 20 background events, a statistical analysis is possible. Furthermore, this threshold lies in a region where the discovery reach hardly varies with the chosen background threshold. Whilst Figure 31 shows only



(a) CNL

(b) ResNet-110 (5 input channels)

(c) eMLP

(d) GAT - 3 heads

Figure 31: The discovery reach cross section σ_5 depending on the chosen background threshold for. We chose the same threshold for all models. At 20 background events, the cross section remains almost constant when changing the background threshold slightly. Furthermore, with 20 background events there are enough events left for a statistical analysis.

the behavior of a selection, the other models follow similar trends. It is noticeable that for low thresholds the cross section curves of low and high masses diverge for CNN type models and the MLP type models as can be seen in figures 31a-c. On the other hand, for the GNN type models they do not diverge, as can be seen in Figure 31d. This trend is common to all graph based models in this thesis.
With a chosen background threshold of 20 events, we achieve a background rejection of $\frac{1}{\epsilon_B} = 97$ for all models. This is the working point for the analysis performed in this thesis. However, it must be mentioned that other reasonable thresholds could be chosen.

A.3. The Transformer model

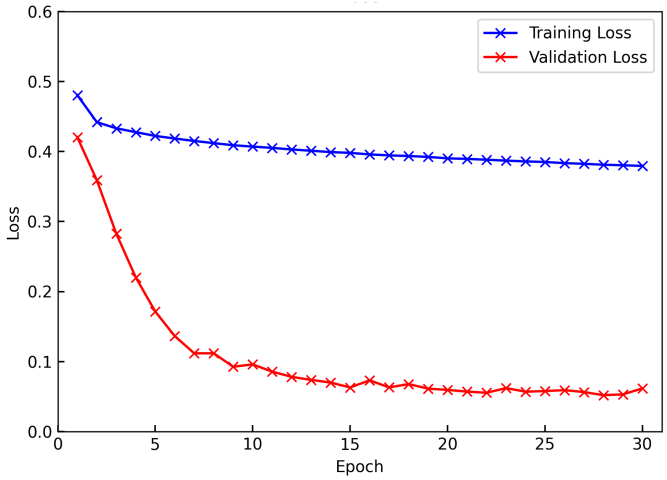
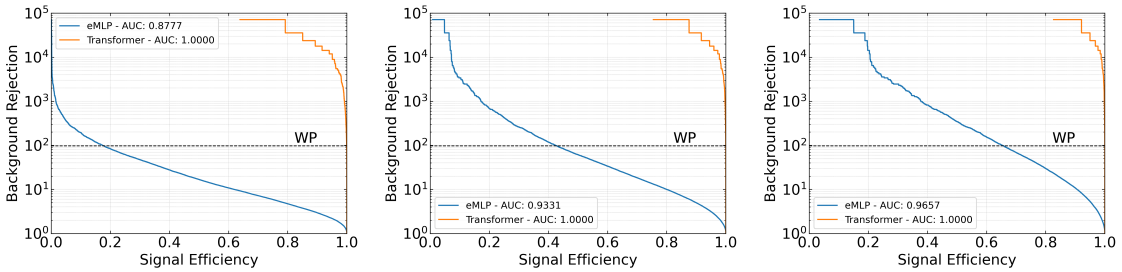


Figure 32: Cross-entropy loss curves of the Transformer model during training. The training loss lies well above the validation loss. This is unusual training behavior, that is not completely understood. The model achieves unreasonable good performance. For example, it achieves an accuracy of over 99.5%.

The Transformer encoder model as described in section 5 exhibits unusual training behavior, whereby the validation loss surpasses the training loss significantly. The validation loss can usually lie under the training loss curve at the beginning of the training in the first few epochs, where the model rapidly learns, however we see in Figure 32 that the validation loss consistently lies well below the training loss. In theory, this can happen, if the model uses a high dropout rate. During the validation, the dropout is disabled and the full model is active. This often results in a better performance during validation compared to training. However, here we have a dropout rate of only 0.05 and have also observed this behavior with dropout completely disabled. As can be seen in Figure 32 this is also not a one time occurrence, since the model achieves consistently lower validation loss for multiple epochs.
We observe similar behavior if we switch the role of training and validation set during training. Meaning, we train on the validation set and validate on the training set. This shows that this behavior is not inherent to the here used validation set.
Furthermore, we have not observed any unexpected behavior of the training loss during the epochs. Figure 32 only shows the mean loss per epoch. To study the behavior of the loss during the epochs, we have monitored the loss per batch over multiple epochs. If the models would somehow end one training epoch with a sig-



(a) $m(S^{\pm\pm}) = 300$ GeV (b) $m(S^{\pm\pm}) = 600$ GeV (c) $m(S^{\pm\pm}) = 900$ GeV

Figure 33: The ROC-curves of the Transformer model. We see a near perfect discrimination for all signal masses. This leads to a signal efficiency of almost one at the working point and an AUC of 1.0000 rounded to the fourth digit. This performance seems unreasonable.

nificantly lower training loss and then start the next epoch with a higher loss again, this would signify that there is some influence on the model parameters after each epoch. This could explain the here observed behavior since, the model state at the end of the training epoch is better than at the beginning and the validation loss is calculated at the end of a training epoch. The validation loss would be evaluated on this better model state. However, we have not observed any significant change of the training loss per batch from one epoch to the next.

This training behavior also persists for different initzializations of the model and also manifest if the architecture is changed slightly, e.g. Transformer encoders with more encoder layers or attention heads.

We suggest studying this behavior in further research.

Nonetheless, we perform a test set analysis of the Transformer model and show its ROC-curves in Figure 33. The test set has never been seen by the Transformer model and should give an independent metric for its performance. In Figure 33 we can see that the Transformer model even performs beyond what is expected to be reasonable on the test set. The Transformer achieves an AUC of up to 1.0000 (rounded to the fourth digit), which would indicate a nearly perfect classifier. Such a performance solely based on the kinematic data seems unrealistic. Such surprisingly good performance could, in theory, be explained by data-leakage. Data-leakage happens when information that should not be accessible to the network is present in the features used to classify the sample. Since we use multiple different models on the kinematic representation, and we only see this behavior with the Transformer, we do not expect this to be the case. If data-leakage were to occur, we should also

observe this with other models, since the same training procedure and data are used for each model. This also does not explain the discrepancy between the training and validation loss.
Due to the unusual training behavior and this surprising performance, we present this result with reservations. As we can see in Figure 33 the signal efficiency at the working point is basically 1.0 and remains there even when the background rejection of the working point is changed by an order of magnitude. This makes the physical analysis almost exclusively dependent on the choice of the working point.

A.4. Differences between datasets

As mentioned in section 6.3 the CNN type models have performed worse on the new event data generated during the work for this thesis as compared to the data of [15]. We have tested the same model on both datasets, while fixing all other variables, e.g. the hyperparameters[2] and training set size. We have observed that the CNNs perform systematically worse on the new data. As discussed in section 4.2 there were some differences in the process used to generate the events. The main issue of the wrong η-cut was, however, fixed in the newest version of [15] and cannot be responsible for the difference. We do not suspect that the other differences in the imposed cuts can be responsible for this either, since the cuts differ only slightly.
The different versions of the Monte-Carlo generators can contribute to the different neural network performances. For example, we use MadGraph_aMC@NLO version 3.5.1 while in [15] version 3.4.0 is used. To further investigate the differneces between the datasets, we have trained a CNN classifier to distinguish between the two datasets. For this, we have created a mix of both datasets for each process (signal and background processes) and have labeled the new data as class 1 and the other as class 0. We then use the jet images to train the ResNet-20 classifier and find that the network is able to distinguish the $t\bar{t}t\bar{t}$-background with an accuracy of over 80%, without much hyperparameter tuning. After consulting with the authors of [15], we have found another difference during the event generation. Namely, whilst they also simulate the $t\bar{t}t\bar{t}$-background at leading order (LO) they do so by using the aMC@NLO codebase. This codebase is usually used for the NLO event generation, does however also have the capability to generate LO events. They do so by setting the QCD flag during the event generation:

```
generate p p > t t~ t t~ [QCD]
```

[2]Using optimized hyperparameter on each dataset leads to the same results.

They then define the order as LO later on:

```
order=LO
```

Whereas, we do not set this flag and therefore use the MadGraph codebase. Since this is the only significant difference between the datasets, we presume that this is at least partially responsible for the performance difference. Since the influence of different Monte-Carlo generators on the neural network performance and hence the physical analysis is significant, we suggest studying this in further work.

Acronyms

AUC Area Under the Curve.

BatchNorm Batch Normalization (Layer).

CCWZ Callan-Coleman-Wess-Zumino.

CNL CNN taking Charged, Neutral and Leptonic input.

CNLK CNN taking Charged, Neutral, Leptonic and Kinematic input.

CNN Convolutional Neural Network.

CP Charge Parity.

eMLP efficient Multilayer Perceptron.

EW Electro Weak.

FC Fully Connected (Network).

GAT Graph Attention (Network).

GCN Graph Convolution Network.

GNN Graph Neural Network.

GUT Grand Unified Theory.

LHC Large Hadron Collider.

LO Leading Order.

MCHM Minimal Composite Higgs Model.

MLP Multilayer Perceptron.

MRC Multi Representation Classifier.

NLO Next to Leading Order.

pNGB pseudo Nambu-Goldstone Boson.

QCD Quantum Chromodynamics.

ReLU Rectified Linear Unit.

ROC-curve Receiver Operator Characteristic Curve.

SGD Stochastic Gradient Decent.

SM Standard Model.

VEV Vacuum Expectation Value.

References

[1] **CMS** Collaboration, S. Chatrchyan *et al.*, "Observation of a New Boson at a Mass of 125 GeV with the CMS Experiment at the LHC," *Phys. Lett. B* **716** (2012) 30–61, arXiv:1207.7235 [hep-ex].

[2] **ATLAS** Collaboration, G. Aad *et al.*, "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** (2012) 1–29, arXiv:1207.7214 [hep-ex].

[3] D. B. Kaplan and H. Georgi, "SU(2) $\times$ U(1) breaking by vacuum misalignment," *Physics Letters B* **136** no. 3, (1984) 183–186.

[4] D. B. Kaplan, H. Georgi, and S. Dimopoulos, "Composite higgs scalars," *Physics Letters B* **136** no. 3, (1984) 187–190.

[5] R. Contino, Y. Nomura, and A. Pomarol, "Higgs as a holographic pseudoGoldstone boson," *Nucl. Phys. B* **671** (2003) 148–174, arXiv:hep-ph/0306259.

[6] K. Agashe, R. Contino, and A. Pomarol, "The Minimal composite Higgs model," *Nucl. Phys. B* **719** (2005) 165–187, arXiv:hep-ph/0412089.

[7] D. B. Kaplan, "Flavor at ssc energies: A new mechanism for dynamically generated fermion masses," *Nuclear Physics B* **365** no. 2, (1991) 259–278.

[8] G. Ferretti and D. Karateev, "Fermionic UV completions of Composite Higgs models," *JHEP* **03** (2014) 077, arXiv:1312.5330 [hep-ph].

[9] G. Ferretti, "Gauge theories of Partial Compositeness: Scenarios for Run-II of the LHC," *JHEP* **06** (2016) 107, arXiv:1604.06467 [hep-ph].

[10] A. Belyaev, G. Cacciapaglia, H. Cai, G. Ferretti, T. Flacke, A. Parolini, and H. Serodio, "Di-boson signatures as Standard Candles for Partial Compositeness," *JHEP* **01** (2017) 094, arXiv:1610.06591 [hep-ph]. [Erratum: JHEP 12, 088 (2017)].

[11] A. Agugliaro, G. Cacciapaglia, A. Deandrea, and S. De Curtis, "Vacuum misalignment and pattern of scalar masses in the SU(5)/SO(5) composite Higgs model," *JHEP* **02** (2019) 089, arXiv:1808.10175 [hep-ph].

[12] A. Banerjee *et al.*, "Phenomenological aspects of composite Higgs scenarios: exotic scalars and vector-like quarks," arXiv:2203.07270 [hep-ph].

[13] A. Banerjee, D. B. Franzosi, and G. Ferretti, "Modelling vector-like quarks in partial compositeness framework," *JHEP* **03** (2022) 200, arXiv:2202.00037 [hep-ph].

[14] G. Cacciapaglia, T. Flacke, M. Kunkel, W. Porod, and L. Schwarze, "Exploring extended Higgs sectors via pair production at the LHC," *JHEP* **12** (2022) 087, arXiv:2210.01826 [hep-ph].

[15] T. Flacke, J. H. Kim, M. Kunkel, P. Ko, J. S. Pi, W. Porod, and L. Schwarze, "Uncovering doubly charged scalars with dominant three-body decays using machine learning," *JHEP* **11** (2023) 009, arXiv:2304.09195 [hep-ph].

[16] J. Cogan, M. Kagan, E. Strauss, and A. Schwartzman, "Jet-Images: Computer Vision Inspired Techniques for Jet Tagging," *JHEP* **02** (2015) 118, arXiv:1407.5675 [hep-ph].

[17] L. de Oliveira, M. Kagan, L. Mackey, B. Nachman, and A. Schwartzman, "Jet-images — deep learning edition," *JHEP* **07** (2016) 069, arXiv:1511.05190 [hep-ph].

[18] P. Baldi, K. Bauer, C. Eng, P. Sadowski, and D. Whiteson, "Jet Substructure Classification in High-Energy Physics with Deep Neural Networks," *Phys. Rev. D* **93** no. 9, (2016) 094034, arXiv:1603.09349 [hep-ex].

[19] J. Barnard, E. N. Dawe, M. J. Dolan, and N. Rajcic, "Parton Shower Uncertainties in Jet Substructure Analyses with Deep Neural Networks," *Phys. Rev. D* **95** no. 1, (2017) 014018, arXiv:1609.00607 [hep-ph].

[20] P. T. Komiske, E. M. Metodiev, and M. D. Schwartz, "Deep learning in color: towards automated quark/gluon jet discrimination," *JHEP* **01** (2017) 110, arXiv:1612.01551 [hep-ph].

[21] **ATLAS** Collaboration, "Quark versus Gluon Jet Tagging Using Jet Images with the ATLAS Detector,". https://inspirehep.net/files/a5d847032c1ad4abbde281f16fc0474f.

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

[23] S. Macaluso and D. Shih, "Pulling Out All the Tops with Computer Vision and Deep Learning," *JHEP* **10** (2018) 121, arXiv:1803.00107 [hep-ph].

[24] S. Choi, S. J. Lee, and M. Perelstein, "Infrared Safety of a Neural-Net Top Tagging Algorithm," *JHEP* **02** (2019) 132, arXiv:1806.01263 [hep-ph].

[25] I. Henrion, J. Brehmer, J. Bruna, K. Cho, K. Cranmer, G. Louppe, and G. Rochette, "Neural message passing for jet physics," *Deep Learning for Physical Sciences Workshop at the 31st Conference on Neural Information Processing Systems (NIPS)* (2017) . https://orbi.uliege.be/bitstream/2268/226446/1/nips_dlps_2017_29.pdf.

[26] F. Ma, F. Liu, and W. Li, "Jet tagging algorithm of graph network with Haar pooling message passing," *Phys. Rev. D* **108** no. 7, (2023) 072007, arXiv:2210.13869 [hep-ex].

[27] H. Qu and L. Gouskos, "Jet tagging via particle clouds," *Physical Review D* **101** no. 5, (Mar., 2020) . http://dx.doi.org/10.1103/PhysRevD.101.056019.

[28] L. Susskind, "Dynamics of spontaneous symmetry breaking in the weinberg-salam theory," *Phys. Rev. D* **20** (11, 1979) 2619–2625. https://link.aps.org/doi/10.1103/PhysRevD.20.2619.

[29] M. J. Dugan, H. Georgi, and D. B. Kaplan, "Anatomy of a composite higgs model," *Nuclear Physics B* **254** (1985) 299–326.

[30] J. Goldstone, "Field theories with « superconductor » solutions," *Il Nuovo Cimento* (1961) . https://doi.org/10.1007/BF02812722.

[31] J. Goldstone, A. Salam, and S. Weinberg, "Broken symmetries," *Phys. Rev.* **127** (Aug, 1962) 965–970.

[32] G. Panico and A. Wulzer, *The Composite Nambu-Goldstone Higgs*. Springer International Publishing, 2016. http://dx.doi.org/10.1007/978-3-319-22617-0.

[33] L. Susskind, "Dynamics of spontaneous symmetry breaking in the weinberg-salam theory," *Phys. Rev. D* **20** (Nov, 1979) 2619–2625. https://link.aps.org/doi/10.1103/PhysRevD.20.2619.

[34] S. Weinberg, "Implications of dynamical symmetry breaking," *Phys. Rev. D* **13** (Feb, 1976) 974–996. https://link.aps.org/doi/10.1103/PhysRevD.13.974.

[35] S. Weinberg, "Implications of dynamical symmetry breaking: An addendum," *Phys. Rev. D* **19** (Feb, 1979) 1277–1280. https://link.aps.org/doi/10.1103/PhysRevD.19.1277.

[36] S. Coleman, J. Wess, and B. Zumino, "Structure of phenomenological lagrangians. i," *Phys. Rev.* **177** (Jan, 1969) 2239–2247. https://link.aps.org/doi/10.1103/PhysRev.177.2239.

[37] C. G. Callan, S. Coleman, J. Wess, and B. Zumino, "Structure of phenomenological lagrangians. ii," *Phys. Rev.* **177** (Jan, 1969) 2247–2250. https://link.aps.org/doi/10.1103/PhysRev.177.2247.

[38] S. L. Adler, "Axial-vector vertex in spinor electrodynamics," *Phys. Rev.* **177** (Jan, 1969) 2426–2438. https://link.aps.org/doi/10.1103/PhysRev.177.2426.

[39] J. S. Bell and R. Jackiw, "A PCAC puzzle: $\pi^0 \to \gamma\gamma$ in the σ model," *Nuovo Cim. A* **60** (1969) 47–61.

[40] J. Wess and B. Zumino, "Consequences of anomalous ward identities," *Physics Letters B* **37** no. 1, (1971) 95–97.

[41] E. Witten, "Global aspects of current algebra," *Nuclear Physics B* **223** no. 2, (1983) 422–432.

[42] L. Schwarze, "LHC Phenomenology of the Electroweak pNGBs in the SU(5)/SO(5) Coset," Master's thesis, Julius-Maximilians-Universität Würzburg, 2022.

[43] M. Kunkel, "LHC Phenomenology of Top Partners in Models with SU(6)/Sp(6)," Master's thesis, Julius-Maximilians-Universität Würzburg, 2021.

[44] G. Ferretti, "UV Completions of Partial Compositeness: The Case for a SU(4) Gauge Group," *JHEP* **06** (2014) 142, arXiv:1404.7137 [hep-ph].

[45] G. Cacciapaglia, T. Flacke, M. Kunkel, and W. Porod, "Phenomenology of unusual top partners in composite Higgs models," *JHEP* **02** (2022) 208, arXiv:2112.00019 [hep-ph].

[46] K. Agashe, R. Contino, L. Da Rold, and A. Pomarol, "A Custodial symmetry for $Zb\bar{b}$," *Phys. Lett. B* **641** (2006) 62–66, arXiv:hep-ph/0605341.

[47] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *CoRR* **abs/1511.08458** (2015) , 1511.08458. http://arxiv.org/abs/1511.08458.

[48] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, G. Gordon, D. Dunson, and M. Dudík, eds., vol. 15 of *Proceedings of Machine Learning Research*, pp. 315–323. PMLR, Fort Lauderdale, FL, USA, 11–13 apr, 2011. https://proceedings.mlr.press/v15/glorot11a.html.

[49] C. Krittanawong, K. Johnson, R. Rosenson, Z. Wang, M. Aydar, U. Baber, J. Min, W. Tang, J. Halperin, and S. Narayan, "Deep learning for cardiovascularmedicine: A practical primer," *European heart journal* **40** (02, 2019) .

[50] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization." 2017. https://arxiv.org/abs/1412.6980.

[51] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *CoRR* **abs/1502.01852** (2015) , 1502.01852. http://arxiv.org/abs/1502.01852.

[52] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems* **2** no. 4, (1989) 303–314. https://doi.org/10.1007/BF02551274.

[53] M. K. . N. A. Jake Lever, "Model selection and overfitting," *Nature Methods* **13** (2016) .

[54] L. N. Smith, "Cyclical learning rates for training neural networks," *CoRR* **abs/1506.01186** (2017) , 1506.01186 [cs.CV]. http://arxiv.org/abs/1506.01186.

[55] L. N. Smith and N. Topin, "Super-convergence: Very fast training of residual networks using large learning rates," *CoRR* **abs/1708.07120** (2017) , 1708.07120. http://arxiv.org/abs/1708.07120.

[56] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research* **15** no. 56, (2014) 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html.

[57] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR* **abs/1502.03167** (2015) , 1502.03167. http://arxiv.org/abs/1502.03167.

[58] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks* **5** no. 2, (1994) 157–166. https://ieeexplore.ieee.org/document/279181.

[59] W. L. Hamilton, "Graph representation learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning* **14** no. 3, 1–159. https://link.springer.com/book/10.1007/978-3-031-01588-5.

[60] J. H. Kim, M. Kim, K. Kong, K. T. Matchev, and M. Park, "Portraying double higgs at the large hadron collider," *Journal of High Energy Physics* **2019** no. 9, (Sept., 2019) . http://dx.doi.org/10.1007/JHEP09(2019)047.

[61] A. Alloul, N. D. Christensen, C. Degrande, C. Duhr, and B. Fuks, "Feynrules 2.0 — a complete toolbox for tree-level phenomenology," *Computer Physics Communications* **185** no. 8, (Aug., 2014) 2250–2300. http://dx.doi.org/10.1016/j.cpc.2014.04.012.

[62] http://feynrules.irmp.ucl.ac.be/wiki/NLOModels.

[63] C. Degrande, C. Duhr, B. Fuks, D. Grellscheid, O. Mattelaer, and T. Reiter, "Ufo – the universal feynrules output," *Computer Physics Communications* **183** no. 6, (June, 2012) 1201–1214. http://dx.doi.org/10.1016/j.cpc.2012.01.022.

[64] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H.-S. Shao, T. Stelzer, P. Torrielli, and M. Zaro, "The automated computation of tree-level and next-to-leading order differential cross sections, and their

matching to parton shower simulations," *Journal of High Energy Physics* **2014** no. 7, (July, 2014) . http://dx.doi.org/10.1007/JHEP07(2014)079.

[65] R. Frederix, S. Frixione, V. Hirschi, D. Pagani, H.-S. Shao, and M. Zaro, "The automation of next-to-leading order electroweak calculations," *Journal of High Energy Physics* **2018** no. 7, (July, 2018) . http://dx.doi.org/10.1007/JHEP07(2018)185.

[66] V. Hirschi and O. Mattelaer, "Automated event generation for loop-induced processes." *J. High Energ. Phys. 2015, 146 (2015).* (2015) . https://doi.org/10.1007/JHEP10(2015)146.

[67] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten, S. Mrenna, S. Prestel, C. O. Rasmussen, and P. Z. Skands, "An introduction to pythia 8.2," *Computer Physics Communications* **191** (2015) 159–177. https://www.sciencedirect.com/science/article/pii/S0010465515000442.

[68] R. D. Ball, V. Bertone, S. Carrazza, L. Del Debbio, S. Forte, A. Guffanti, N. P. Hartland, and J. Rojo, "Parton distributions with qed corrections," *Nuclear Physics B* **877** no. 2, (Dec., 2013) 290–320. http://dx.doi.org/10.1016/j.nuclphysb.2013.10.010.

[69] T. D. . collaboration., de Favereau, J., Delaere, and C. et al., "Delphes 3: a modular framework for fast simulation of a generic collider experiment." *J. High Energ. Phys. 2014, 57 (2014).* (2014) . https://doi.org/10.1007/JHEP02(2014)057.

[70] M. Cacciari, G. Salam, and G. Soyez, "Fastjet user manual." *Eur. Phys. J. C 72, 1896 (2012).* (2012) . https://doi.org/10.1140/epjc/s10052-012-1896-2.

[71] M. Cacciari, G. P. Salam, and G. Soyez, "The anti-ktjet clustering algorithm," *Journal of High Energy Physics* **2008** no. 04, (Apr., 2008) 063–063. http://dx.doi.org/10.1088/1126-6708/2008/04/063.

[72] **ATLAS** Collaboration, "Technical Design Report for the ATLAS Inner Tracker Pixel Detector," tech. rep., CERN, Geneva, 2017. http://cds.cern.ch/record/2285585.

[73] B. Fuks, M. Klasen, D. R. Lamprea, and M. Rothering, "Precision predictions for electroweak superpartner production at hadron colliders with resummino,"

The European Physical Journal C **73** no. 7, (July, 2013) . http://dx.doi.org/10.1140/epjc/s10052-013-2480-0.

[74] A. Paszke, S. Gross, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *CoRR* **abs/1912.01703** (2019) , 1912.01703. http://arxiv.org/abs/1912.01703.

[75] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," *CoRR* **abs/1907.10902** (2019) , 1907.10902. http://arxiv.org/abs/1907.10902.

[76] D. Bertolini, P. Harris, M. Low, and N. Tran, "Pileup per particle identification," *Journal of High Energy Physics* **2014** no. 10, (Oct., 2014) . http://dx.doi.org/10.1007/JHEP10(2014)059.

[77] A. Sirunyan, A. Tumasyan, *et al.*, "Pileup mitigation at cms in 13 tev data," *Journal of Instrumentation* **15** no. 09, (Sept., 2020) P09018–P09018. http://dx.doi.org/10.1088/1748-0221/15/09/P09018.

[78] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR* **abs/1512.03385** (2015) , 1512.03385. http://arxiv.org/abs/1512.03385.

[79] K. He and J. Sun, "Convolutional neural networks at constrained time cost," *CoRR* **abs/1412.1710** (2014) , 1412.1710. http://arxiv.org/abs/1412.1710.

[80] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *CoRR* **abs/1505.00387** (2015) , 1505.00387. http://arxiv.org/abs/1505.00387.

[81] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," *CoRR* **abs/1611.05431** (2016) , 1611.05431. http://arxiv.org/abs/1611.05431.

[82] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds., vol. 25. Curran Associates, Inc., 2012. https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[83] J. Bardhan, A. Sinha, and K. Shah, "Particlenet pytorch geometric," https://github.com/Jai2500/particlenet, 2021.

[84] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Trans. Graph.* **38** no. 5, (Oct, 2019) . https://doi.org/10.1145/3326362.

[85] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *CoRR* **abs/1609.02907** (2016) , 1609.02907. http://arxiv.org/abs/1609.02907.

[86] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory." 2009. https://arxiv.org/abs/0912.3848.

[87] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need." 2023. https://arxiv.org/abs/1706.03762.

[88] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks." 2018. https://arxiv.org/abs/1710.10903.

[89] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, eds., pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota, June, 2019. https://aclanthology.org/N19-1423.

[90] G. Cowan, K. Cranmer, E. Gross, and O. Vitells, "Asymptotic formulae for likelihood-based tests of new physics," *The European Physical Journal C* **71** no. 2, (Feb., 2011) . http://dx.doi.org/10.1140/epjc/s10052-011-1554-0.

[91] L. Huang, S.-b. Kang, J. H. Kim, K. Kong, and J. S. Pi, "Portraying double higgs at the large hadron collider ii," *Journal of High Energy Physics* **2022** no. 8, (Aug., 2022) . http://dx.doi.org/10.1007/JHEP08(2022)114.

[92] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks." 2013. https://arxiv.org/abs/1211.5063.

[93] H. Qu, C. Li, and S. Qian, "Particle Transformer for Jet Tagging," arXiv:2202.03772 [hep-ph].

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet und die Arbeit keiner anderen Prüfungsbehörde unter Erlangung eines akademischen Grades vorgelegt habe.

Würzburg, den 15.07.2024 .....

Manuel Schmidt

Acknowledgements

First and foremost, I would like to thank my father, Andreas, for supporting me throughout all my studies and always encouraging me. Without his support, I would never have been able to achieve all that I have.
My gratitude also extends to my supervisor, Prof. Dr. Werner Porod, who guided me through my work for this thesis and taught me the intricacies of standard model physics and beyond.
I would further like to thank Manuel Kunkel, who assisted me with all the technical aspects of this work and patiently answered all my questions regarding physics, programming and more.
Finally, I would like to thank all my friends for making the time spent studying physics both fun and interesting.