

Polarization studies in same-sign W-boson pair production at the LHC with the ATLAS detector

Master-Arbeit
zur Erlangung des Hochschulgrades
Master of Science
im Master-Studiengang Physik

vorgelegt von

Erik Bachmann
geboren am 28.12.1997 in Dresden

Institut für Kern- und Teilchenphysik
Fakultät Physik
Bereich Mathematik und Naturwissenschaften
Technische Universität Dresden
2023

Eingereicht am 12. April 2023

1. Gutachter: Dr. Frank Siegert
2. Gutachter: Prof. Dr. Arno Straessner

Abstract

The scattering of massive electroweak gauge bosons offers a great opportunity to study the Standard Model's innermost gauge symmetry structure and the electroweak sector. Especially the scattering of longitudinally polarized bosons is sensitive to the concrete realization of the electroweak symmetry breaking due to the unitarity violating energy dependency of the scattering amplitude without the Higgs boson. At the LHC, the pair production of same-charged $W^\pm$ bosons with the final state $\ell^\pm\nu\ell^\pm\nu jj$ is one of the most promising vector boson scattering channels due to the comparatively high cross-section of the electroweak production mode.

In this thesis, this process is investigated for a center of mass energy $\sqrt{s} = 13$ TeV and an integrated luminosity of 139 fb^{-1} . As part of this study, a framework for automated and distributed hyperparameter optimization and input variable selection of artificial neural networks has been developed. It is applied to optimize multilayer perceptrons (MLPs) for the discrimination of longitudinally polarized $W^\pm$ -boson pairs against the other polarizations and background processes. Maximum likelihood fits to Asimov data, including only statistical uncertainties, are performed, from which a discovery significance of 1.4σ is obtained. A comparative study using gradient-boosted decision trees shows an 11% improvement of the significance when using neural networks. An outlook for the High-Luminosity LHC (HL-LHC) results in an expected significance of 6.9σ with the use of MLPs, compared to 3.2σ obtained with the most sensitive single observable, indicating that observation is likely achievable with the HL-LHC.

Kurzdarstellung

Die Streuung von massiven, elektroschwachen Eichbosonen bietet eine Möglichkeit, die innere Eichstruktur des Standardmodells und den elektroschwachen Sektor zu untersuchen. Insbesondere die Streuung longitudinal polarisierter Bosonen ist aufgrund der unitaritätsverletzenden Energieabhängigkeit der Streuamplitude ohne das Higgs-Boson sensitiv darauf, wie die Elektroschwachen Symmetrie gebrochen ist. Am LHC ist die Paarproduktion von gleich geladenen $W^\pm$ -Bosonen mit dem Endzustand $\ell^\pm \nu \ell^\pm \nu jj$ aufgrund des vergleichsweise hohen Wirkungsquerschnitts des elektroschwachen Produktionskanals einer der vielversprechendsten Vektorboson-Streuprozesse.

In dieser Arbeit wird dieser Prozess für eine Kollisionsenergie von $\sqrt{s} = 13$ TeV und eine integrierte Luminosität von 139 fb^{-1} untersucht. Im Rahmen dieser Studie wurde eine Anwendung zur automatisierten und verteilten Hyperparameteroptimierung und Selektion der Eingabe-Variablen für künstliche neuronale Netze entwickelt. Sie wird angewandt, um mehrschichtige Perzeptrons (MLPs) darauf zu optimieren, longitudinal polarisierte Paare von $W^\pm$ -Bosonen von anderen Polarisationskomponenten und den Hintergrundprozessen zu trennen. Mit Maximum-Likelihood-Fits gegen Asimov-Daten, die nur statistische Unsicherheiten berücksichtigen, wurde eine Signifikanz von $1,4\sigma$ erreicht. Eine vergleichende Studie unter Verwendung von geboosteten Entscheidungsbäumen zeigt eine 11%ige Verbesserung der Signifikanz bei der Verwendung neuronaler Netze. Ein Ausblick für den High-Luminosity LHC (HL-LHC) ergibt eine erwartete Signifikanz von $6,9\sigma$ bei Verwendung von MLPs, verglichen mit $3,2\sigma$ erzielt mit der empfindlichsten Observable. Dies deutet darauf hin, dass eine Beobachtung des untersuchten Prozesses mit dem HL-LHC vermutlich möglich sein wird.

Contents

1	Introduction	1
2	Theoretical foundation	3
2.1	The Standard Model of Particle Physics	3
2.1.1	Electroweak Theory	4
2.1.2	The Electroweak Symmetry Breaking	5
2.2	Vector Boson Scattering	8
2.2.1	The $W^\pm W^\pm jj$ -EW Process	8
2.2.2	Polarization in Vector Boson Scattering	11
3	Experimental methods	13
3.1	The Large Hadron Collider	13
3.2	The ATLAS detector	14
3.2.1	The ATLAS coordinate system	15
3.2.2	Inner Detector	16
3.2.3	Calorimeter System	16
3.2.4	Muon Spectrometer	17
3.3	Multivariate Analysis	17
3.3.1	Boosted Decision Trees	18
3.3.2	Multilayer Perceptrons	22
3.3.3	Generalization, Underfitting & Overfitting	26
4	Distributed Hyperparameter Optimization for Neural Networks	31
4.1	Hyperparameter optimization algorithms	31
4.2	Hyperparameter optimization framework	37
4.2.1	Overview of the optimization	38
4.2.2	Custom metrics & overfitting conditions	39
4.2.3	Input data preprocessing	40
4.2.4	Experiment execution	41
4.2.5	Experiment evaluation	47
4.2.6	Input variable selection	51

5	Extraction of the Longitudinal Polarization of $W^\pm$-boson pairs	55
5.1	The $W^\pm W^\pm jj$ -EW signal region	55
5.1.1	Object selection	55
5.1.2	Event selection	57
5.1.3	Monte Carlo samples and expected yields	58
5.2	Fit strategy and polarization sensitive observables	61
5.3	Validation of the $W^\pm W^\pm jj$ -EW polarization samples	65
5.4	Study of the LL polarization using boosted decision trees	68
5.4.1	Figures of merit	68
5.4.2	Preprocessing of input variables	70
5.4.3	Variable & hyperparameter optimization	73
5.4.4	Results	75
5.5	Study of the LL polarization using multilayer perceptrons	82
5.5.1	Input variables, data preprocessing and general settings	82
5.5.2	Determination of the optimal settings for the <code>TPESampler</code> and the re- porting of results to Tune	84
5.5.3	Determination of the optimal hyperparameter search space	86
5.5.4	Overfitting conditions	93
5.5.5	Input variable optimization	100
5.5.6	Results	104
5.6	Outlook for the High-Luminosity LHC	107
6	Summary and Outlook	111
A	Mathematical definitions	113
B	Monte Carlo samples	114
C	Comparison plots for the full, unpolarized and polarized samples	116
D	Distributions of polarization sensitive observables	121
E	Preprocessed signal and background distributions of BDT input variables	128
F	Configuration of the optimization framework	134
	List of Figures	141
	List of Tables	144
	Bibliography	146

1 Introduction

Particle physics is the field of research studying the fundamental building blocks of the universe and their interactions. The theory that describes all known elementary particles, their properties and interactions, is called the Standard Model of particle physics. Developed over the last half century, the Standard Model was finally completed in 2012 with the discovery of the last missing particle, the Higgs boson, at the Large Hadron Collider (LHC) [1, 2]. It represents one of the most thoroughly tested theories and proved to make extremely accurate prediction. Nonetheless, the Standard Model is known to be incomplete, being unable to explain Dark Matter, the matter-antimatter asymmetry or the origin of the elementary fermion masses. Over the years, many theories have been developed, extending the Standard Model in an attempt to explain its deficiencies, often predicting additional particles. However, to date, no experimental evidence has been found to support these predictions.

After the discovery of the Higgs boson as the last particle predicted by the Standard Model, an era of precision measurements began. The properties of the elementary particles and their interactions are precisely studied, in the hope to find deviations from the Standard Model that hint at new, unknown physics. With the Run 2 of the LHC with a center of mass energy of $\sqrt{s} = 13$ TeV completed, an even larger dataset is available to investigate even rarer processes that have so far not been observed.

One of these rare processes is the scattering of longitudinally polarized massive vector bosons. While the existence of the Higgs boson confirmed the mechanism of electroweak spontaneous symmetry breaking (EWSB), its concrete realization is still uncertain. Since the longitudinal polarization of massive vector bosons is a direct consequence of the EWSB, polarized production of vector bosons is a promising group of processes to complement direct Higgs searches in the study of the EWSB. Moreover, without the existence of the Higgs boson, the scattering of longitudinally polarized massive vector bosons violates unitarity, leading to unphysical predictions for high energies. Thus, the scattering of longitudinally polarized massive vector bosons is considered to be very sensitive to the exact realization of the EWSB. Besides that, the existence of diagrams containing triple and quartic gauge couplings in the vector boson scattering process allows to study the innermost gauge symmetry structure of the Standard Model.

The vector boson scattering channel investigated in this work is the electroweak production of same-charged $W^\pm$ -boson pairs in the fully leptonic decay channel. This process is of particular

interest due to the suppression of the QCD-induced production mode and subsequently higher signal to irreducible background ratio. It was first observed in 2018 with data at $\sqrt{s} = 13$ TeV by the CMS collaboration [3] and shortly after confirmed by ATLAS [4]. The longitudinal-longitudinal (LL) polarization component, however, has so far not been observed. Previous results published by CMS showed an observed (expected) 95 % CL upper limit of 1.17 (0.88) fb on the production cross-section [5].

As part of the ongoing ATLAS study of polarization in the pair production of same-charged $W^\pm$ -bosons, the aim of this work is to investigate the use of artificial neural networks to separate the LL polarization component from the other polarizations and background processes in order to increase the sensitivity of the measurement. As part of this study, a framework for automated and distributed hyperparameter optimization and input variable selection of neural networks has been developed, tailored to the use in particle physics analyses.

This thesis is structured as follows: In Chapter 2, an introduction to the Standard Model with special focus on the electroweak theory and the EWSB is given and the same-charged $W^\pm$ -boson pair production process is discussed. Additionally, a short introduction to polarization of intermediate vector bosons is given. The LHC and the ATLAS detector are briefly presented in Chapter 3, followed by a detailed description of the multivariate techniques used in this thesis. In Chapter 4, multiple hyperparameter optimization algorithms are presented and the all relevant parts of the optimization framework are explained. In the first part of Chapter 5, selection criteria and the Monte Carlo samples used to model the signal and background processes are summarized. This is followed by a discussion of polarization sensitive observables and a study using boosted decision trees (BDTs) to separate the LL polarization component from the other polarizations and background processes. Maximum likelihood fits on the BDT-output are performed to Asimov data to obtain the expected discovery significance of the LL polarization component. In the second half of Chapter 5, the optimization framework is applied to optimize the hyperparameters and set of input variables of multilayer perceptrons (MLPs) that are trained to perform the same binary classification task as the BDTs. Maximum likelihood fits to Asimov data are performed on the output distributions of the best performing MLPs to obtain the expected discovery significance of the LL polarization component. Additionally, an outlook for the study of this process at the High-Luminosity LHC is given. A summary and an outlook for further studies is given in Chapter 6.

2 Theoretical foundation

The goal of particle physics is to probe nature’s most fundamental building blocks and their interactions. In order to interpret our measurements, a theory is necessary to make predictions about the measurement outcome given our current understanding of the universe. In looking for disagreements between predictions and observations, new, yet unknown phenomena can be discovered. In this chapter, a brief overview of the most successful theory of particle physics to date, the Standard Model, is given and the process studied in this thesis is introduced. It is heavily inspired by the approaches presented in Ref. [6–9].

2.1 The Standard Model of Particle Physics

The Standard Model of particle physics (SM) is a relativistic quantum field theory describing the three fundamental interactions that are accessible with current particle physics experiments: the electromagnetic, the weak and the strong interaction. It is formulated as a non-Abelian gauge theory with the gauge group $SU(3)_C \otimes SU(2)_L \otimes U(1)_Y$ where each interaction is connected to a continuous symmetry. The subgroup $SU(3)_C$ represents the strong interaction and is the basis for Quantum Chromodynamics (QCD) while the $SU(2)_L \otimes U(1)_Y$ subgroup corresponds to the Electroweak theory describing the unified electromagnetic and the weak interactions.

As per the Noether theorem, any continuous symmetry is associated with a conserved quantity. In the SM, this gives rise to the charges of particles. Additionally, the requirement of local gauge invariance of the Lagrangian density of the SM leads to the emergence of additional vector fields, the so-called gauge bosons. Each gauge boson is associated with one of the interactions and only couples to particles carrying the corresponding charge. The strong interaction is mediated by eight different types of gluons g coupling to the color charge $\vec{C}$. The W^+ , W^- and Z -bosons transmit the weak interaction and couple to the third component of the weak isospin I_3^W . The photon γ is the mediator of the electromagnetic interaction and couples to the electric charge Q . The charges of all matter particles (fermions) and gauge bosons are summarized in Tab. 2.1.

Table 2.1: Overview of the charges of all elementary particles in the Standard Model. C is used as a placeholder for the color charges red, green and blue while $\bar{C}$ is the corresponding anti-charge. For the fermions, only the charges of particles are shown; anti-particles carry charges of opposite sign. Additionally, the shown values of the third component of the weak isospin I_3^W corresponds to left-handed fermions (right-handed anti-fermions) as right-handed fermions (left-handed anti-fermions) do not take part in the weak interaction, thus for them $I_3^W = 0$.

	Fermions				Bosons				
	Leptons		Quarks		Gauge bosons				Higgs boson
	ν_e, ν_μ, ν_τ	e, μ, τ	u, c, t	d, s, b	γ	g	Z	$W^\pm$	H
Electric charge Q	0	-1	$2/3$	$-1/3$	0	0	0	± 1	0
Weak isospin I_3^W	$1/2$	$-1/2$	$1/2$	$-1/2$	0	0	0	± 1	$-1/2$
Color charge $\vec{C}$	$\vec{0}$	$\vec{0}$	C	C	$\vec{0}$	$C\bar{C}'$	$\vec{0}$	$\vec{0}$	$\vec{0}$

2.1.1 Electroweak Theory

As first proven experimentally in 1957 by C. S. Wu et al. [10], the weak interaction violates parity, acting only on left-handed fermions and right-handed anti-fermions. As a result, the left- and right-handed components of a fermion field ψ :

$$\psi_{L/R} = \frac{1}{2}(1 \mp \gamma_5)\psi, \quad (2.1)$$

need to transform differently under the $SU(2)_L \otimes U(1)_Y$ gauge transformations, forming $SU(2)_L$ -doublets Ψ_L for left-handed fermions and -singlets ψ_R for right-handed fermions:

$$\begin{aligned} \Psi_L &\rightarrow e^{i\alpha_a(x)T^a} e^{i\beta(x)Y} \Psi_L, \\ \psi_R &\rightarrow \mathbb{1} e^{i\beta(x)Y} \psi_R. \end{aligned} \quad (2.2)$$

Here, $\alpha_a(x)$ and $\beta(x)$ are the space-time dependent gauge parameters and the weak hypercharge¹ Y and $T^a = \frac{1}{2}\sigma^a$ are the generators of the $U(1)_Y$ and $SU(2)_L$ groups, respectively. σ^a and γ_5 denote the Pauli matrices and the fifth Dirac matrix as defined in Appendix A.

Using the above defined notation, the Lagrangian of the electroweak theory can be written as:

$$\mathcal{L}_{\text{EW}} = \mathcal{L}_{\text{fermion}} + \mathcal{L}_{\text{gauge}} = \sum_{f_L, f_R} \bar{\psi}_f i\gamma^\mu D_\mu \psi_f - \frac{1}{4} W_{\mu\nu}^a W^{a\mu\nu} - \frac{1}{4} B_{\mu\nu} B^{\mu\nu}, \quad (2.3)$$

with the adjoint spinor $\bar{\psi} = \psi^\dagger \gamma^0$ and the sum over all fermions $f_{L/R}$. As mentioned before, four additional vector fields W_μ^a and B^μ generated by the $SU(2)_L$ and $U(1)_Y$, respectively, were

¹The weak hypercharge Y is connected to the electric charge and the weak isospin via $Q = Y + I_3^W$.

introduced by minimal coupling to ensure local gauge invariance:

$$\begin{aligned} D_\mu \Psi_L &= (\partial_\mu + ig_W T^a W_\mu^a + ig_Y Y B_\mu) \Psi_L, \\ D_\mu \psi_R &= (\partial_\mu + ig_Y Y B_\mu) \psi_R. \end{aligned} \quad (2.4)$$

Their infinitesimal transformations need to take the form:

$$\begin{aligned} W_\mu^a &\rightarrow W_\mu^a - \frac{1}{g_W} \partial_\mu \alpha^a(x) - \epsilon^{abc} \alpha^b(x) W_\mu^c, \\ B_\mu &\rightarrow B_\mu - \frac{1}{g_Y} \partial_\mu \beta(x), \end{aligned} \quad (2.5)$$

so that the Lagrangian is invariant under simultaneous transformations of the fermion fields according to Eq. 2.2 and the gauge fields. It can be seen that $\mathcal{L}_{\text{fermion}}$ describes the free fermions and their interactions with the electroweak gauge fields while $\mathcal{L}_{\text{gauge}}$ with the field strength tensors

$$\begin{aligned} W_{\mu\nu}^a &= \partial_\mu W_\nu^a - \partial_\nu W_\mu^a + g_W \epsilon^{abc} W_{b,\mu} W_{c,\nu}, \\ B_{\mu\nu} &= \partial_\mu B_\nu - \partial_\nu B_\mu, \end{aligned} \quad (2.6)$$

contains the dynamics of the gauge fields and predicts cubic and quartic gauge boson self-interactions. Example Feynman diagrams of these self-interactions² are depicted in Fig. 2.1.

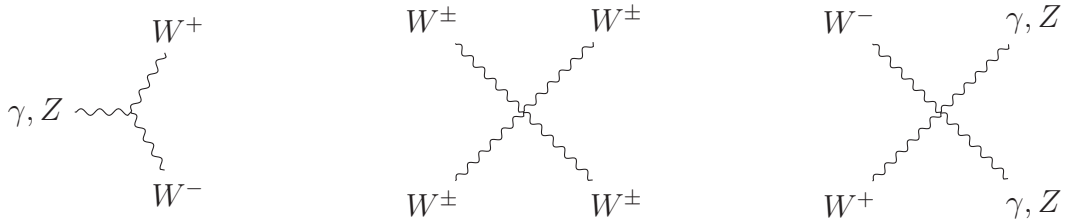


Figure 2.1: Example Feynman diagrams of gauge boson self-interactions

2.1.2 The Electroweak Symmetry Breaking

In the formulation of the electroweak theory presented in the previous section, as proposed in 1961 by S. Glashow [11], all particles are still massless which contradicts clear experimental evidence of the masses of fermions and the $W^\pm$ and Z -bosons. Since mass terms for the fermions and gauge bosons:

$$\begin{aligned} \mathcal{L}_{m_f} &= m_f \psi \bar{\psi}, \\ \mathcal{L}_{m_b} &= \frac{1}{2} m_b^2 V_\mu V^\mu, \end{aligned} \quad (2.7)$$

²In these Feynman diagrams, the physical gauge fields introduced below have been used.

are not $SU(2)_L \otimes U(1)_Y$ gauge invariant, they cannot be included explicitly. This problem was solved by A. Salam and S. Weinberg [12, 13] using the Brout-Englert-Higgs mechanism [14–16] to generate gauge invariant mass terms via spontaneous symmetry breaking. For this, two new complex scalar fields need to be introduced, forming an $SU(2)_L$ doublet with hypercharge $Y = \frac{1}{2}$:

$$\Phi = \begin{pmatrix} \phi^+ \\ \phi^0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} \phi_1 + i\phi_2 \\ \phi_3 + i\phi_4 \end{pmatrix}, \quad (2.8)$$

and the Lagrangian is extended by the kinematic terms of Φ and a new potential $V(\Phi)$:

$$\mathcal{L}_\Phi = (D^\mu \Phi)^\dagger (D_\mu \Phi) - V(\Phi) = (D^\mu \Phi)^\dagger (D_\mu \Phi) - \mu^2 \Phi^\dagger \Phi - \lambda (\Phi^\dagger \Phi)^2. \quad (2.9)$$

For $\mu^2 < 0$ and $\lambda > 0$, this potential has an infinite set of degenerate, non-vanishing minima. Since $V(\Phi)$ is invariant under $SU(2)_L \otimes U(1)_Y$, the Lagrangian remains gauge invariant while the ground state spontaneously breaks the symmetry by choosing one of the minima³, e.g.:

$$\Phi_0 = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ v \end{pmatrix}, \quad (2.10)$$

with the vacuum expectation value $v = \sqrt{-\frac{\mu^2}{\lambda}} = 246$ GeV. Expanding Φ around this minimum:

$$\Phi(x) \approx \begin{pmatrix} \tilde{\phi}_1(x) + i\tilde{\phi}_2(x) \\ \frac{v+H(x)}{\sqrt{2}} + i\tilde{\phi}_4(x) \end{pmatrix}, \quad (2.11)$$

which can be simplified by a gauge transformation to:

$$\Phi(x) \approx \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ v + H(x) \end{pmatrix}, \quad (2.12)$$

and inserting into the kinetic term of Eq. 2.9 yields:

$$(D_\mu \Phi)^\dagger (D^\mu \Phi) = \frac{1}{2} \partial_\mu H \partial^\mu H + \frac{1}{8} (v + H)^2 \{ |g_W W_\mu^3 - g_Y B_\mu|^2 + g_W^2 |W_\mu^1 + iW_\mu^2|^2 \}. \quad (2.13)$$

To obtain the physical gauge bosons, the mass matrix needs to be diagonalized, resulting in the following mass states:

$$W_\mu^\pm = \frac{1}{\sqrt{2}} (W_\mu^1 \mp iW_\mu^2) \quad Z_\mu = \frac{g_W W_\mu^3 - g_Y B_\mu}{\sqrt{g_W^2 + g_Y^2}} \quad A_\mu = \frac{g_W W_\mu^3 + g_Y B_\mu}{\sqrt{g_W^2 + g_Y^2}}. \quad (2.14)$$

³While not proven here, this choice ensures that the $U(1)_Q$ symmetry of quantum electrodynamics remains unbroken, otherwise the photon would also become massive.

Using these physical fields, Eq. 2.9 becomes:

$$\mathcal{L}_\Phi = \underbrace{m_W^2 W_\mu^+ W^{-\mu} + \frac{1}{2} m_Z^2 Z_\mu Z^\mu}_{\text{W,Z mass terms}} + \underbrace{\left(vH + \frac{H^2}{2} \right) \left\{ 2 \frac{m_W^2}{v^2} W_\mu^+ W^{-\mu} + \frac{m_Z^2}{v^2} Z_\mu Z^\mu \right\}}_{\text{W,Z-Higgs interaction}} + \underbrace{\frac{1}{2} \partial_\mu H \partial^\mu H - \lambda v^2 H^2}_{\text{Higgs-Lagrangian}} - \underbrace{\lambda v H^3 - \frac{\lambda}{4} H^4}_{\text{Higgs-self-interaction}}, \quad (2.15)$$

with

$$m_W = \frac{g_W v}{2} \quad m_Z = \frac{1}{2} v \sqrt{g_W^2 + g_Y^2} \quad m_A = 0. \quad (2.16)$$

In addition to the masses of the $W^\pm$ and Z -bosons, the electroweak symmetry breaking gives rise to an additional massive boson with mass $m_H = \sqrt{2\lambda}v$, called the Higgs boson. In Eq. 2.15, coupling of the Higgs boson to the massive electroweak gauge bosons and Higgs-self-interactions are visible. Feynman diagrams of the possible interaction between the Higgs, $W^\pm$ and Z -bosons are shown in Fig. 2.2.



Figure 2.2: Feynman diagrams of the interactions between the Higgs and the massive electroweak gauge bosons.

Finally, gauge invariant fermion mass terms can be added by hand by introducing couplings between the fermions and the Higgs doublet:

$$\mathcal{L}_{\text{Yukawa}} = y_1^f \bar{\Psi}_L^f \Phi \psi_R^f + y_2^f \bar{\Psi}_L^f (i\sigma_2 \Phi^*) \psi_R^f + h.c., \quad (2.17)$$

resulting in the fermion masses

$$m_f = \frac{y_{1,2}^f}{\sqrt{2}} v, \quad (2.18)$$

and couplings between the fermions and the Higgs boson. The Yukawa couplings $y_{1,2}^f$ are free parameters of the SM and need to be determined experimentally. Neutrino masses are not part of the Standard Model.

2.2 Vector Boson Scattering

As seen in the Lagrangian of the electroweak theory, specifically the field strength tensors in Eq. 2.6, the Standard Model predicts cubic and quartic self-interactions between the electroweak gauge bosons. Additionally, the kinetic term of the Higgs-field gives rise to couplings between the Higgs boson and the massive electroweak gauge bosons. Collectively, the gauge boson self-interactions and the production and exchange of a Higgs boson are referred to as *Vector Boson Scattering* (VBS). Within this thesis, the scattering of two same-charged $W^\pm$ -bosons is investigated and is introduced in the following section.

Due to the presence of vertices between the electroweak gauge bosons and the Higgs boson, VBS is considered to be very sensitive to the exact mechanism of the electroweak symmetry breaking (EWSB). Furthermore, the longitudinal polarization state of massive gauge bosons is a direct consequence of the EWSB, as discussed later in this chapter. This makes the scattering of longitudinally polarized gauge bosons a prime candidate to complement direct Higgs searches in the study of the EWSB.

2.2.1 The $W^\pm W^\pm jj$ -EW Process

In Fig. 2.3, the Feynman diagrams for all leading-order interactions between two same-charged $W^\pm$ -bosons are shown.

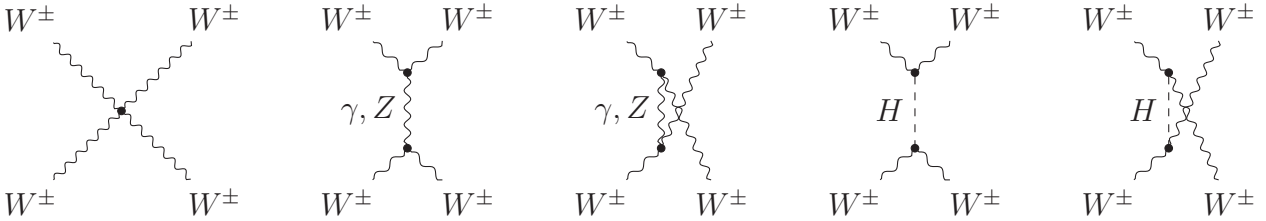


Figure 2.3: Leading order Feynman diagrams of the interaction of two same-charged $W^\pm$ -bosons.

At a p-p collider like the LHC, the $W^\pm$ -bosons are radiated off by quarks or anti-quarks in the accelerated protons. Since these initial state quarks only undergo an electroweak interaction, they are only weakly deflected from the beam axis, resulting in two hadronic jets with a large rapidity separation ΔY and high invariant mass m_{jj} in the detector. These two so-called *tagging jets* provide a characteristic experimental signature of VBS at the LHC.

Due to their extremely short lifetime, the $W^\pm$ -bosons cannot be measured directly in the detector. Instead, their existence must be inferred from their decay products. Despite the low branching fraction, only the fully leptonic decay channel where both $W^\pm$ -bosons decay leptonically, $W^\pm \rightarrow \ell^\pm \nu$, is considered due to the overwhelmingly large cross-section of background processes with multi-jet final states at a hadron collider like the LHC. Thus, the

process to be studied, denoted $W^\pm W^\pm jj$, is given by:

$$qq \rightarrow \ell_1^\pm \nu_{\ell_1} \ell_2^\pm \nu_{\ell_2} jj, \quad (2.19)$$

with ν denoting a neutrino in the W^+W^+ case and an anti-neutrino for W^-W^- . A depiction of the VBS-diagrams including the production and decay of the $W^\pm$ -bosons is shown in Fig. 2.4.

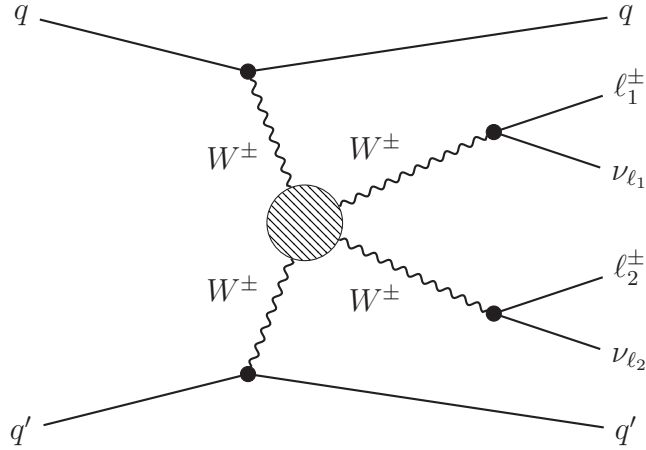


Figure 2.4: Feynman diagram of the $W^\pm W^\pm$ vector boson scattering at leading order at the LHC. The blob is a placeholder for the interactions shown in Fig. 2.3.

At leading order, three gauge invariant contributions to the $W^\pm W^\pm jj$ process exist. The first contribution contains the VBS diagrams shown in Fig. 2.4. As it can be seen, VBS is of order $\mathcal{O}(\alpha_{EW}^6 \alpha_S^0)$ in the electroweak and strong coupling parameters α_{EW} and α_S , respectively. In order to obtain a gauge invariant set of Feynman diagrams, all diagrams of the same order and final state must be included in the process definition. A subset of the non-VBS diagrams of the same order are shown in Fig. 2.5. Collectively, these diagrams define the signal process, denoted $W^\pm W^\pm jj$ -EW throughout this thesis.

The second contribution contains QCD-induced production diagrams of order $\mathcal{O}(\alpha_{EW}^4 \alpha_S^2)$ and is correspondingly referred to as $W^\pm W^\pm jj$ -QCD throughout this thesis. An exemplary subset of contributing diagrams can be found in Fig. 2.6. Due to electric charge conservation, no leading-order diagrams exist with two gluons in the initial state. As a result, $W^\pm W^\pm jj$ is unique among the different processes of massive vector boson pair production in association with jets, being the only process where the fiducial cross-section of the QCD-induced contributions are significantly smaller than the cross-section of the electroweak production process. [6]

Finally, interferences between the two aforementioned processes at $\mathcal{O}(\alpha_{EW}^5 \alpha_S)$ make up the final contribution to the $W^\pm W^\pm jj$ process and will be denoted $W^\pm W^\pm jj$ -Int throughout this thesis.

Since the $W^\pm W^\pm jj$ -QCD and $W^\pm W^\pm jj$ -Int processes can be separated from the $W^\pm W^\pm jj$ -EW signal process in a gauge invariant way, both are used as background processes in this thesis.

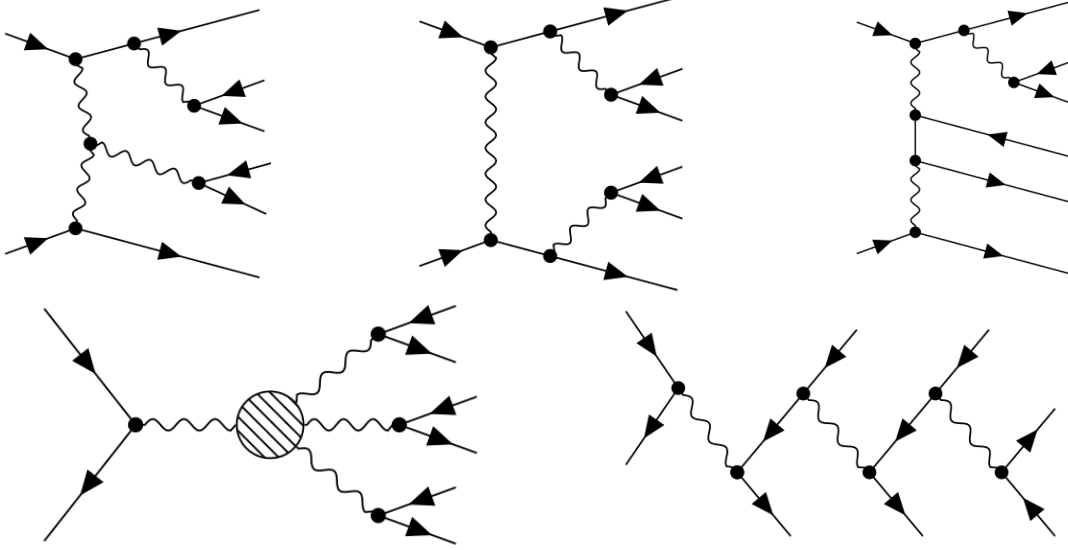


Figure 2.5: Examples of non-VBS Feynman diagrams of order $\mathcal{O}(\alpha_{EW}^6 \alpha_S^0)$ contributing to the $qq \rightarrow \ell_1^\pm \nu_{\ell_1} \ell_2^\pm \nu_{\ell_2} jj$ process (diagrams taken from Ref. [6]).

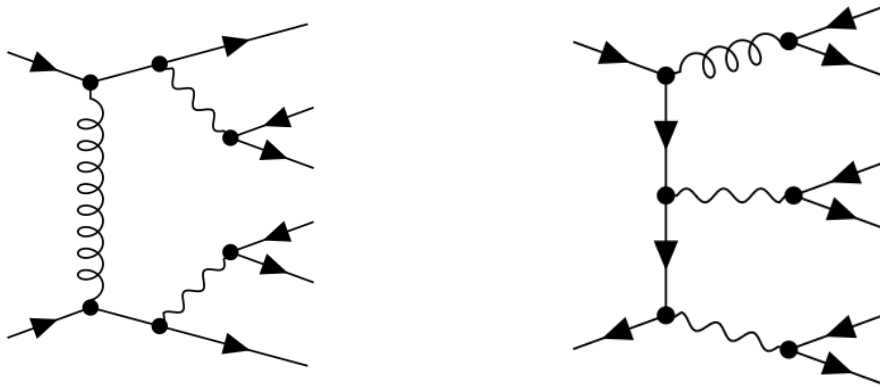


Figure 2.6: Examples of Feynman diagrams contributing to the $W^\pm W^\pm jj$ -QCD process (diagrams taken from Ref. [6]).

2.2.2 Polarization in Vector Boson Scattering

Polarization is defined as the alignment of the spin $\vec{S}$ of a particle along a given axis. Commonly, the particle's momentum $\vec{p}$ is used as the quantization axis, giving the helicity h :

$$h = \vec{S} \cdot \frac{\vec{p}}{|\vec{p}|}. \quad (2.20)$$

Since the helicity is only Lorentz-invariant for massless particles, the polarization of a massive vector boson depends on the frame of reference it was defined in. Within this thesis, the polarization is defined in the center of mass frame of the two $W^\pm$ -bosons. Other choices are for example the laboratory frame or the center of mass frame of the initial state partons, however, these reference systems are not investigated in this thesis.

For a massive spin S particle, the projection of its spin onto an axis has $2S + 1$ possible eigenvalues between $-S$ and S while for massless particles, only $\pm S$ are possible. Thus, the helicity of the massive vector bosons can take on 3 different values: 0 and ± 1 . The values $h = \pm 1$ correspond to the alignment and anti-alignment of the particle's spin and momentum directions and are collectively referred to as *transverse polarization*. A helicity of $h = 0$ is called *longitudinal polarization* and means that the spin is orthogonal to the direction of the particle's momentum. Within this thesis, transversely polarized $W^\pm$ -bosons will be denoted $W_T^\pm$, longitudinally polarized bosons as $W_\ell^\pm$.

Since the longitudinal polarization state only exists for massive vector bosons, it is a direct consequence of the EWSB. The three additional degrees of freedom that are introduced (one additional polarization state for each W^+ , W^- and Z -bosons) correspond to the massless Nambu-Goldstone-bosons $\tilde{\phi}_1$, $\tilde{\phi}_2$ and $\tilde{\phi}_4$ that were eliminated using a gauge transformation (see Eq. 2.11 and 2.12). In addition to that, as outlined in Ref. [6, 17], the individual cross-sections of triple and quartic gauge interactions of longitudinally polarized $W^\pm$ -boson pairs grow $\propto s^2$ for high center of mass energies $\sqrt{s} = m_{WW} \gg 2m_W$. Summing all pure gauge boson interactions cancels the divergence ("gauge cancellations"), but a rise $\propto s$ remains. This quadratic growth of the scattering cross-section with the center of mass energy violates unitarity and is not physical [18]. The inclusion of the interaction with Higgs boson in the calculation is necessary to restore unitarity, showing that the scattering of longitudinally polarized $W^\pm$ -boson pairs is very sensitive to the exact realization of the EWSB. Such a unitarity-violating energy dependence does not occur for the scattering of transversely polarized $W^\pm$ -boson pairs. In Fig. 2.7, the energy dependence of the scattering cross-section is visualized for different gauge boson combinations, both with and without the inclusion of the Higgs boson.

Since for observable processes, vector bosons only occur as intermediate particles, two peculiarities arise in the definition of polarized cross-sections. Both difficulties and the necessary surrounding theoretical framework have been discussed in Ref. [7]. In essence, the definition of the polarization for intermediate vector bosons requires the factorization of the full matrix

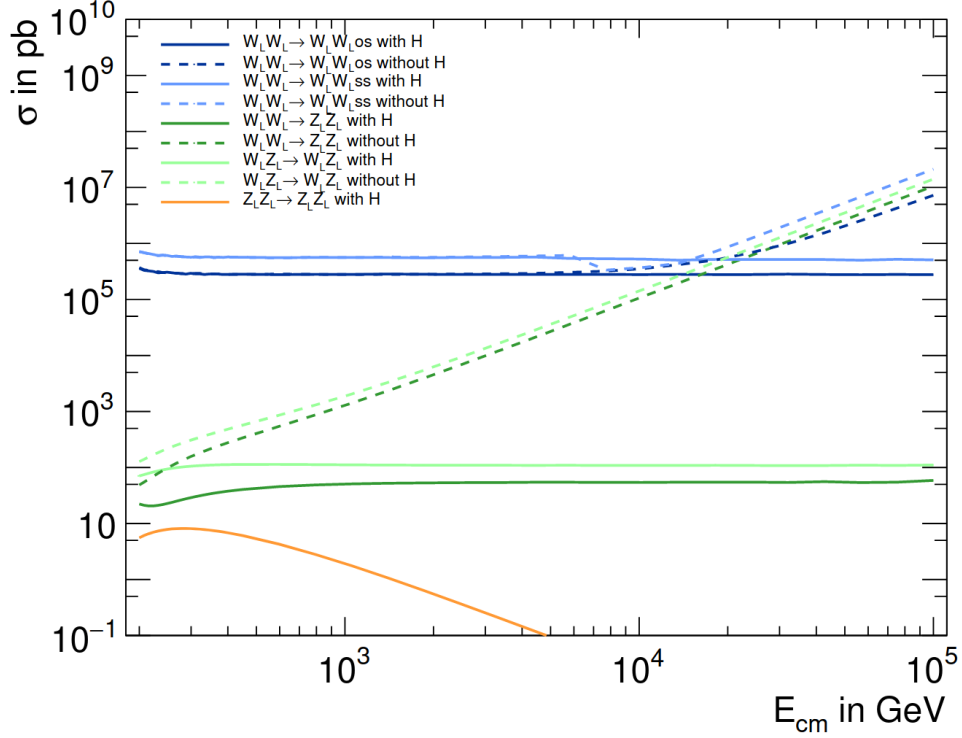


Figure 2.7: Scattering cross-section for all combinations of longitudinally polarized massive gauge boson pairs, $V_L V_L \rightarrow V_L V_L$, as a function of the center of mass energy. Solid lines correspond to the SM predicted cross-sections (with the Higgs boson) while dashed lines show the case without the Higgs boson. Figure taken from Ref. [19]

element into vector boson production and decay (here for simplicity only for a single vector boson):

$$\mathcal{M} \propto \sum_{\lambda} \mathcal{M}_{\lambda}^{\mathcal{P}} \mathcal{M}_{\lambda}^{\mathcal{D}} =: \sum_{\lambda} \mathcal{M}_{\lambda}^{\mathcal{F}}. \quad (2.21)$$

Here, $\mathcal{M}_{\lambda}^{\mathcal{F}}$ corresponds to the full polarized amplitude, in this case for a single vector boson with polarization λ . For this factorization to be possible, only double-resonant diagrams can be included. Since, as seen in Fig. 2.4, not all diagrams contributing to the $W^{\pm}W^{\pm}jj$ -EW process are double-resonant, a suitable approximation is necessary to exclude these single- and non-resonant diagrams from the calculation. The specific approximation used in this thesis will be introduced briefly in Sec. 5.3.

The second peculiarity is the emergence of interference terms between different polarizations due to the squaring of the matrix element:

$$\underbrace{|\mathcal{M}|^2}_{\text{coherent sum}} = \underbrace{\sum_{\lambda} |\mathcal{M}_{\lambda}^{\mathcal{F}}|^2}_{\text{polarized contributions, incoherent sum}} + \underbrace{\sum_{\lambda \neq \lambda'} \mathcal{M}_{\lambda}^{\mathcal{F}} \mathcal{M}_{\lambda'}^{*\mathcal{F}}}_{\text{interferences}}, \quad (2.22)$$

which only vanish in the absence of lepton cuts. The influence of these interferences will also be discussed in Sec. 5.3.

3 Experimental methods

The investigations presented in this thesis were performed as part of the current ATLAS study of polarization in the pair production of same-charged $W^\pm$ -bosons. Thus, while all results are based on Monte Carlo simulation and no actual measured data was used, the simulations as well as the selection criteria discussed in Sec. 5.1 are specific to the ATLAS detector at the Large Hadron Collider (LHC). Therefore, a short overview over the LHC and the ATLAS detector is provided in this chapter; detailed information can be found in the corresponding design reports available at Ref. [20] and [21], respectively. Additionally, a brief introduction to the likelihood function and its application in determining the discovery significance of a process is given. Finally, the multivariate techniques used in this thesis are discussed in detail.

3.1 The Large Hadron Collider

The Large Hadron Collider (LHC) is a circular particle accelerator and collider. Built between 2000 and 2008 at the European Organization for Nuclear Research (CERN¹) in the 26.7 km long tunnel of the previous Large Electron-Positron Collider (LEP) on the France-Switzerland border near Geneva, it is the largest and most powerful particle accelerator in the world. It is designed to collide two counter-rotating beams of protons at a center of mass energy of $\sqrt{s} = 14$ TeV or heavy ions (Pb) with an energy of 2.8 TeV per nucleon. The collisions occur at four interaction points where the two beams are crossed and the four main experiments are situated: ATLAS [21], CMS [22], ALICE [23] and LHCb [24]. While ATLAS and CMS are general purpose, high luminosity detectors, LHCb is specialized to study the physics of bottom quarks and ALICE focuses on studying the quark gluon plasma using heavy ion collisions.

To achieve the high beam energies, the particles are pre-accelerated using multiple accelerators of increasing size and energy. This system of accelerators is shown in Fig. 3.1. The beam of the final pre-accelerator, the Super Proton Synchrotron (SPS), is injected into two separate beam pipes to achieve counter-rotating beams. Within the LHC, the beam is further accelerated to the desired beam energy using superconducting radio-frequency cavities. A total of 1232 superconducting dipole magnets, operating at a temperature of 1.9 K and producing a maximum magnetic field strength of 8.3 T, are used to bend the trajectory of the particle beams.

Each beam itself consists of 2808 particle bunches of about $2 \cdot 10^{11}$ particles, resulting in a

¹CERN is an abbreviation for the original french name: Conseil Européen pour la Recherche Nucléaire.

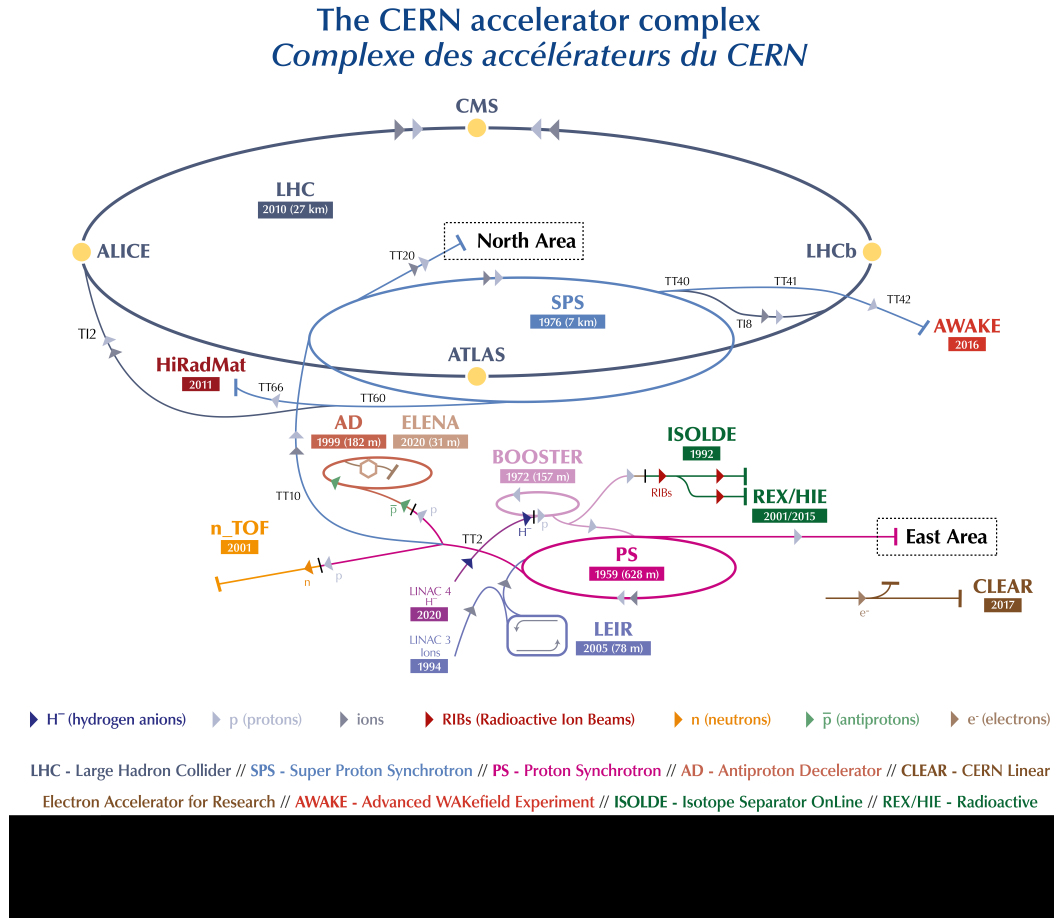


Figure 3.1: Schematic depiction of the CERN accelerator complex. Figure taken from Ref. [25].

bunch spacing of 25 ns, i.e. at each intersection point two opposing bunches cross every 25 ns. Per bunch crossing, between 20 and 40 proton-proton interactions occur (“pileup”).

So far, the design center of mass energy of $\sqrt{s} = 14$ TeV has not yet been reached: during Run 2, only $\sqrt{s} = 13$ TeV was achieved while for the current Run 3, which began in 2022, the energy could be raised to $\sqrt{s} = 13.6$ TeV. However, during the previous run, more than twice the design peak instantaneous luminosity was reached, resulting in a total delivered integrated luminosity of 156 fb^{-1} for Run 2, as reported by the ATLAS collaboration [26].

3.2 The ATLAS detector

The ATLAS² detector is one of the two general purpose detectors operated at the LHC. It consists of concentric layers around the beam axis and is built forward-backward symmetric around the interaction point, resulting in a cylindrical shape of 44 m in length and 25 m in height. It is sectioned into the central barrel region and one end-cap region pipe on either side

²A Toroidal LHC ApparatuS

perpendicular to the beam pipe. A schematic depiction of the ATLAS detector is shown in Fig. 3.2.

The ATLAS detector consists of four major components: the inner detector, the calorimeter system, the muon spectrometer and the magnet system. Each component will be briefly presented below. During Run 2, the ATLAS detector recorded a total integrated luminosity of 139 fb^{-1} [26].

3.2.1 The ATLAS coordinate system

For the ATLAS detector, a right-handed coordinate system with origin at the interaction point (nominal vertex) is used. The z -axis is parallel to the beam axis, the x -axis points towards the center of the LHC and the y -axis (consequently) points upwards. Thus, the x - y plane forms the transverse plane perpendicular to the beam axis. Instead of Cartesian coordinates, the spherical coordinates given by ϕ , θ and R are often used.

At a proton-proton collider like the LHC, the momentum component along the beam axis p_z of the initial state partons, and thus the z -component of the total momentum, is unknown. Since this leads to a boost of the center of mass frame of the hard interaction along the z -axis with respect to the laboratory frame, observables that are invariant under a boost along the z -axis are desirable. An example of such an observable are differences in *rapidity* Y which is defined as:

$$Y := \frac{1}{2} \ln \left(\frac{E + p_z}{E - p_z} \right). \quad (3.1)$$

For massless particles, the rapidity is identical to the geometrically defined *pseudo-rapidity* η :

$$\eta := -\ln \tan \left(\frac{\theta}{2} \right), \quad (3.2)$$

which is commonly used instead of the angle θ .

While the total momentum of the initial partons is now known, their momentum component perpendicular to the beam axis, called *transverse momentum*:

$$p_T = \sqrt{p_x^2 + p_y^2}, \quad (3.3)$$

is negligible. Applying momentum conservation thus implies that the total transverse momentum of all final state particles must be zero. The difference of the total transverse momentum to zero, called *missing transverse momentum* p_T^{miss} , can be used to infer the existence of undetected particles like neutrinos.

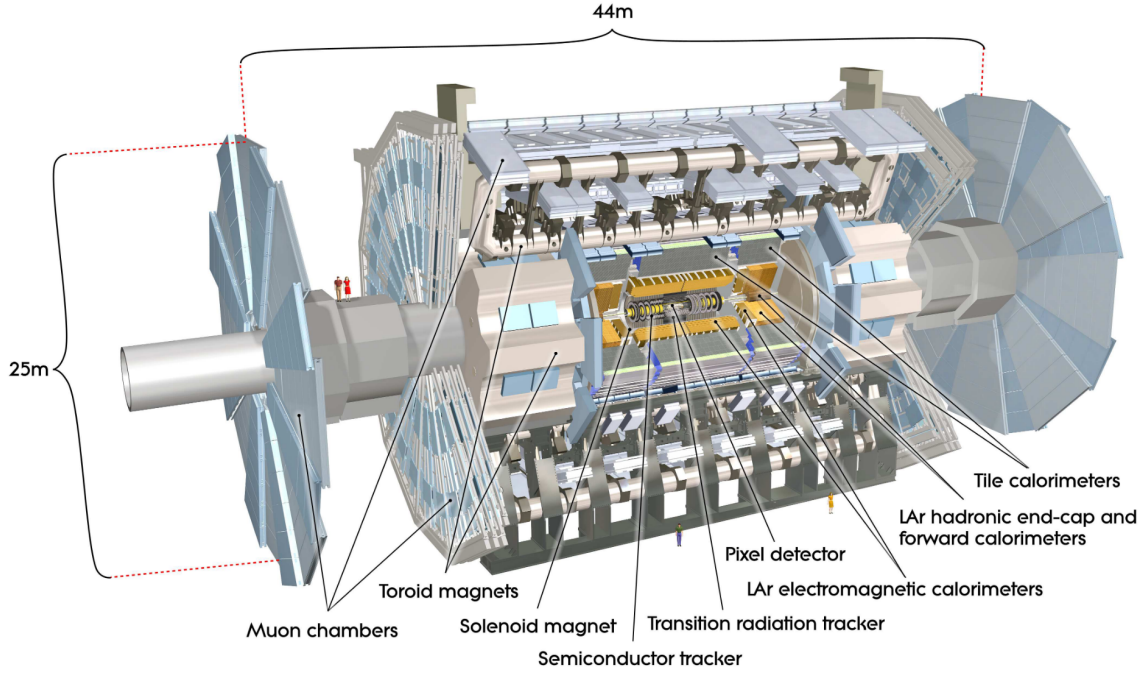


Figure 3.2: Cut-away view of the ATLAS detector. Figure taken from Ref. [21].

3.2.2 Inner Detector

The inner detector (ID) is makes up the first layer of the ATLAS detector and is used to measure the trajectories of charged particles originating from the interaction point. A precise reconstruction of particle tracks is crucial for example for momentum measurement, particle identification, vertex reconstruction, flavor tagging and pileup rejection. The ID covers a region of $|\eta| < 2.5$ and is surrounded by a superconducting solenoid generating a homogeneous magnetic field of 2 T to allow the measurement of particle momenta and electric charge.

The ID is further subdivided into three subdetectors. The innermost parts are the silicon pixel detector and the semiconductor tracker (SCT). They are arranged in concentric layers in the barrel region and contain disks perpendicular to the beam axis in the end-caps. Both cover a region of $|\eta| < 2.5$ and are designed for high precision tracking. They are surrounded by the straw-tube Transition Radiation Tracker (TRT) which covers the region $|\eta| < 2$. While featuring a lower granularity and thus a lower resolution, it allows a larger number of hits per particle, improving the accuracy of the momentum measurement. Additionally, the detection of transition radiation can be used to improve the particle identification. In the barrel region, the straws of the TRT are oriented parallel to the beam axis while they are arranged in wheels in the end-cap region.

3.2.3 Calorimeter System

The calorimeter system of the ATLAS detector is designed to provide precise energy measurements as well as information for particle identification over a large pseudo-rapidity range

$|\eta| < 4.9$. The calorimeter system is subdivided into two main components: the Electromagnetic Calorimeter and the Hadronic Calorimeter. As their names suggest, they are specialized for different types of particles according to their primary method of interaction with matter. Above a certain energy threshold, electrons and photons primarily interact via bremsstrahlung and pair production. The liquid argon (LAr) based Electromagnetic Calorimeter as the inner part of the calorimeter system thus uses lead absorber plates to initiate electromagnetic showers. It consists of a barrel part covering the range of $|\eta| < 1.475$ and two coaxial wheels in the end-cap region spanning the range $1.375 < |\eta| < 3.2$. Together, the barrel region and the outer wheel ($1.375 < |\eta| < 2.5$) feature a high granularity over the range covered by the inner detector and are therefore suited for precision measurements of electrons and photons. Since the hadronic shower created by strongly interacting particles is not stopped in the Electromagnetic Calorimeter, the Electromagnetic Calorimeter is surrounded by the Hadronic Calorimeter designed to measure the energy of jets. It consists of the Tile calorimeter whose barrel covers the range of $|\eta| < 1.0$ and two extended barrel parts cover $0.8 < |\eta| < 1.7$ as well as the LAr hadronic end-cap calorimeter extending the acceptance between $1.5 < |\eta| < 3.2$. Finally, the LAr forward calorimeter covers the range of $3.1 < |\eta| < 4.9$ closest to the beam pipe. It consists of three modules per end-cap, with the innermost module, made of copper, being specialized for electromagnetic measurements and the outer two modules, made of tungsten, measuring the energy of hadronic interactions.

3.2.4 Muon Spectrometer

Since the interaction of muons via bremsstrahlung is suppressed due to their high mass, they are not stopped by the calorimeter. Thus, as the outermost layer of the ATLAS detector, the muon spectrometer is used to measure the momentum of muons produced in the collision. Since, besides muons, all particles observable with the ATLAS detector are typically stopped by the calorimeter system, the signature of muons is very clean. This leads to very good reconstruction and accurate identification of muons. In order to measure their momenta, the muon spectrometer is equipped with three toroidal, air-filled magnets, one barrel and two end-cap magnets.

3.3 Multivariate Analysis

In the quest to find ever smaller signals in increasingly larger datasets, multivariate techniques for signal discrimination have long become a standard tool in many analyses. In this chapter, an introduction is given to two popular techniques that have been used in this thesis to classify signal and background events.

3.3.1 Boosted Decision Trees

In this thesis, the boosted decision trees from *ROOT's Toolkit for Multivariate Data Analysis* (*TMVA*) are used. A brief introduction to the basic principles of BDTs is given below. More information including details of the technical implementation can be found in the *TMVA Users Guide* [27]. Throughout this section, parameters of the BDT that are customizable in TMVA are introduced with the name of the corresponding setting, indicated by the `monospace`-font. The building blocks of a boosted decision tree (BDT) are binary decision trees which are a type of tree-structured classifier. An example of such a decision tree can be seen in Fig. 3.3. For a given set of input variables, a decision tree evaluates a series of binary conditions – so-called splitting conditions – in order to classify the corresponding event. Which condition is checked next always depends on the value of the previous condition, resulting in a tree-like structure. This way, the whole phase space is subdivided into a large number of hypercubes, which are eventually identified as signal-like or background-like. In contrast to that, a cut-based analysis can only select a single hypercube as the signal region.

The splitting conditions are derived recursively during the training (or growing) of the decision tree using labeled training data (pairs of vectors of input variables and corresponding target labels, 1 for signal events, -1 for background events). Starting at the root node, a grid search is performed to find the condition that best separates signal and background for the full training dataset. For each input variable, a fixed number of equidistant cuts (`nCuts`) over the full range of the input variable is tested. Once the best splitting condition is found and the node is split, the same procedure is applied to the two created child nodes using the corresponding subsets of training events. This is repeated until the full tree is built.

The splitting of a node is based on the gini-index $p(1 - p)$, with p being the signal fraction or signal purity of the events that get to the node. It is a measure of how well signal and background are mixed and reaches its maximum value at $p = 0.5$. For higher signal or background fractions, the gini-index decreases and has a minimum value of 0 for $p = 0$ (background-only) and $p = 1$ (signal-only). A node is only split if its gini-index is larger than the sum of the gini-indices of the two created child nodes, weighted by their relative fractions of events. During the grid search, the splitting condition that minimizes the weighted sum of the gini-indices of the child nodes is selected.

Once a node is split, for each of the child nodes, two stopping conditions are checked to decide if the node should be split further:

- number of splits in this chain is equal or larger than `MaxDepth`
- fraction of events that reach this node is smaller than `MinNodeSize`

If any of the stopping conditions is fulfilled, the node is not split further and becomes a leaf node.

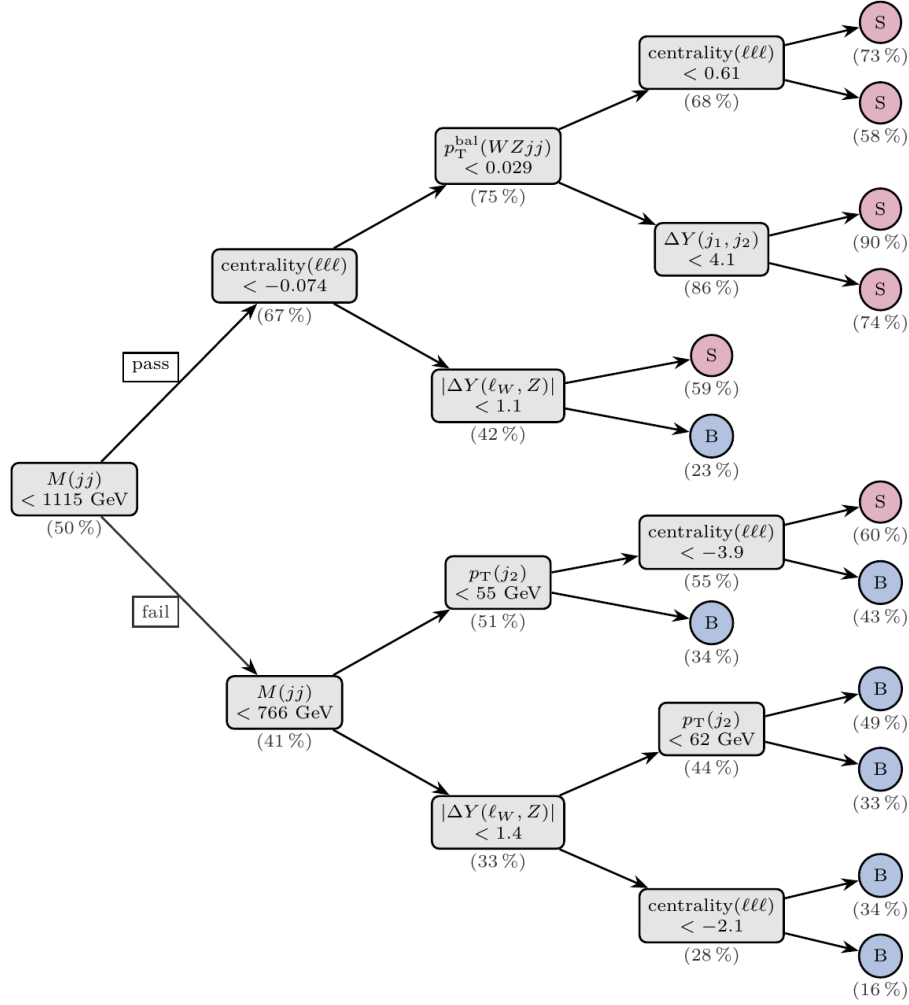


Figure 3.3: Example of a binary decision tree. The condition that is evaluated in each node is given in the gray boxes, the percentage below each node denotes the signal purity in this node. Figure taken from Ref. [18]

All leaf nodes are classified according to the class that the majority of its events belong to: if the node's signal purity p is higher than 0.5, it is classified as signal, otherwise, it is classified as background. During inference, an event is categorized based on which leaf node it ends up in. In order to improve both the performance and the robustness of the classifier, multiple trees are combined to form an ensemble. The output of the classifier is then given as a weighted sum of the outputs of the individual trees in the ensemble. This technique is called Boosting. The idea of this approach is to represent a target function $f(x)$ (in our case a classifier) as a weighted sum of parametrized base functions $b(x; \gamma_m)$:

$$f(x) = \sum_{m=1}^M \beta_m b(x; \gamma_m). \quad (3.4)$$

Typically, the discriminative power of each individual base function is low, giving rise to the name *weak learners*.

In this thesis, *TMVA*'s implementation of Gradient Boosting was used which closely follows the procedure described in Chapter 10 of Ref. [28]. This method introduces a differentiable loss function $L(y, f(x))$ as a measure of the deviation between the model response $f(x)$ and the target labels y . For classification, this is the binomial log-likelihood loss:

$$L(y, f(x)) = \ln(1 + e^{-2f(x)y}). \quad (3.5)$$

The goal of the optimization is to find the set of parameters $\{\beta_m, \gamma_m\}_1^M$ of the base functions $b(x; \gamma_m)$ so that the total loss on the training dataset is minimal:

$$\min_{\{\beta_m, \gamma_m\}_1^M} \sum_{i=1}^N L\left(y_i, \sum_{m=1}^M \beta_m b(x_i; \gamma_m)\right). \quad (3.6)$$

Since this is a very computationally expensive optimization problem, an iterative approach is used to approximate the desired function f in a greedy fashion by adding new basis functions to the expansion without altering the parameters of previous functions. At each step of the iteration, the parameters of the next basis function are determined via:

$$(\beta_m, \gamma_m) = \arg \min_{\beta, \gamma} \sum_{i=1}^N L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)), \quad (3.7)$$

and the model is subsequently updated:

$$f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m). \quad (3.8)$$

When applying this to the boosting of trees, the base functions $b(x; \gamma_m)$ are typically regression trees which could be understood as a generalization of decision trees. As previously discussed, a decision tree partitions the whole phase space into several disjoint regions R_j , each of which is represented by a leaf node j . The output of the tree in each region is constant,

$$x \in R_j \Rightarrow f(x) = \gamma_j, \quad (3.9)$$

so that the tree can be expressed as

$$T(x; \theta) = \sum_{j=1}^J \gamma_j I(x \in R_j), \quad (3.10)$$

with $\theta = \{R_j, \gamma_j\}_1^J$.

In the case of decision trees, γ_j is limited to only ± 1 , for regression trees the value of γ_j can be arbitrary. The growth of such a regression tree follows the same algorithm that has been outlined before, only the gini-index as the node splitting condition is replaced by a condition

based on the mean squared error in each region and the tree complexity.

With Eq. 3.10, the optimization problem (Eq. 3.7) can be reformulated as:

$$\begin{aligned} \hat{\theta}_m &= \arg \min_{\theta_m} \sum_{i=1}^N L(y_i, f_{m-1}(x_i) + T(x_i; \theta_m)), \\ \text{with } \theta_m &= \{R_{jm}, \gamma_{jm}\}_1^{J_m}, \end{aligned} \quad (3.11)$$

meaning that at each optimization step, the tree that maximally reduces the total loss is searched. This optimization problem is solved in two steps by first finding the regions R_{jm} of the leaf nodes and then the corresponding constants γ_{jm} .

Unfortunately, no simple, fast algorithm exists for finding the regions R_{jm} directly. Instead, a gradient descent approach is used to find an approximate solution. Since the direction of the fastest decrease of the total loss is given by its negative gradient, it is natural to choose a tree that approximates the negative gradient as good as possible. This is done by first calculating the gradient of the loss with respect to the model response:

$$g_{im} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f=f_{m-1}}, \quad (3.12)$$

and then fitting a regression tree to the gradients:

$$\tilde{\theta}_m = \arg \min_{\theta} \sum_{i=1}^N (-g_{im} - T(x_i; \theta))^2. \quad (3.13)$$

The regions $\tilde{R}_{jm}$ given by this regression tree are approximations to the regions R_{jm} that solve Eq. 3.11. The corresponding constants γ_{jm} can then be found very easily using:

$$\gamma_{jm} = \arg \min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma). \quad (3.14)$$

The model is consequently updated according to:

$$f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm}), \quad (3.15)$$

and the process is repeated until the desired number of trees **NTrees** is reached.

In order to become less susceptible to random fluctuations in the training dataset, two modifications can be applied to the boosting procedure. Firstly, a **Shrinkage** parameter ν can get introduced as a constant scaling factor to each tree, i.e. a learning rate. For values between 0 and 1, it reduces the contribution of each individual tree, which improves both the robustness and the generalization performance of the BDT. Secondly, a technique called *bagging* can be

employed where the regression tree in each iteration is trained on a random fraction of the training dataset (`BaggedSampleFraction`). It was shown that bagging, when used together with shrinkage, usually results in better performance of the BDT on unseen data. [28]

3.3.2 Multilayer Perceptrons

Even though researchers started investigating artificial neural networks (NNs) more than 50 years ago, with the algorithm to efficiently train them being (re-)discovered in the 1980s, it took until around 2010 for neural networks to start to gain widespread popularity. The main driving forces for the adoption of NNs were the development of graphical processing units (GPUs) for very fast execution of highly parallelizable applications, allowing the use of a small number of GPUs in place of a large cluster of CPUs, as well as many algorithmic improvements that allowed training more powerful neural networks. [29]

Since then, artificial neural networks have become an essential tool in many high-energy physics analyses [30]. In this section, an introduction is given to the type of neural network used for this thesis, the multilayer perceptron (MLP). Further details can be found in Ref. [29, 31, 32]. The fundamental building blocks of neural networks are *artificial neurons*. Each artificial neuron has multiple inputs and a single output, with each input connection having a weight assigned to it. The inputs can either be input variables that are given to the model or the outputs of other neurons. The neuron's output can again be sent to multiple other neurons. Within each neuron, the sum of the neuron's input values, weighted with the corresponding input weights, is calculated, and a *bias* term is added. This sum, called the *activation*³, is then transformed using a non-linear *activation function* to allow a non-linear relationship between the neuron's input vector and its output value. Usually, multiple independent neurons are aggregated into a layer; the layer's output can therefore be expressed using a matrix multiplication between the input vector x , a weights matrix w and a bias vector b :

$$a_i(x) = A \left(\sum_j w_{ij} x_j + b_i \right). \quad (3.16)$$

In a multilayer perceptron, multiple such layers are stacked: the output vector of each layer is used as the input for each neuron in the next layer, forming so-called *fully connected* or *dense layers*. The first layer, called the *input layer*, uses the input variables that are given to the MLP as its input. The output of the last layer of the MLP, the *output layer*, represents the result of the processing. Depending on the application, this output layer consists of a single or multiple neurons; for binary classification, a single output neuron is used. All layers between the input and the output layer are called *hidden layers*. In Fig. 3.4, an example of an MLP

³The term *activation* seems to be used inconsistently in the literature. Throughout this thesis, the convention of Goodfellow et al. [33] to refer to a neuron's output before the non-linear activation function as its *activation* is followed.

with two hidden layers and a single output neuron is shown.

Over the years, many different activation functions have been used inside the hidden layers. In the early days of neural networks, sigmoid functions like the logistic sigmoid or the hyperbolic tangent were popular, motivated by the firing of biological neurons. However, their computational complexity and the fact that their outputs become saturated for high and low input values (“vanishing gradient problem”) limited the number of layers that could be used. In the 2010s, the *Rectified Linear Unit (ReLU)* that is defined as $ReLU(x) = \max(0, x)$ became popular, solving the two main problems of sigmoid activation, allowing the training of deeper neural networks and consequently improving their performance significantly. In recent years, many new activation functions have been invented to improve upon the performance of ReLU. The most commonly addressed problems with ReLU are the lack of negative output values, its limited non-linearity, and the unbounded output for large positive inputs. However, most of the observed improvements are application specific, meaning that the optimal choice depends on the network architecture and the type of problem that should be solved. [35]

The layer’s weights matrices are determined during the training of the neural network. Like with BDTs, this is done using a labeled training dataset consisting of pairs of vectors of input variables and corresponding target values. This training technique is called *supervised learning*. In order to evaluate the performance of the NN, a differentiable loss function is introduced as a measure for the difference between the NN output $f(x)$ and the target value y . The specific choice of which function to use depends on the application; for binary classification, the most common loss function is the binary cross-entropy loss, which, assuming the two classes are encoded with target labels 0 and 1, is defined as:

$$L(f(x), y) = -[y \log(f(x)) + (1 - y) \log(1 - f(x))]. \quad (3.17)$$

The goal of the training is to find the model parameters $W = \{w_{ij}^{(l)}\}_1^N$ that minimize the average loss on the training dataset. Deep learning models can easily have on the order of hundreds of thousands, sometimes even millions of parameters; this can, therefore, not be done analytically. Instead, an approximate solution is found using a gradient descent approach. Since the NN output is a differentiable function of its parameters⁴, and the average loss is a differentiable function of the NN output, the gradient of the average loss function with respect to the NN parameters can be calculated directly, and the parameters can be moved in the opposite direction of the gradient. This procedure can be repeated iteratively until a minimum of the average loss is found; each of these iterations is called an *epoch*.

This algorithm, called *batch gradient descent*, is guaranteed to converge to at least a local minimum. However, especially for large training datasets, it performs many redundant calculations

⁴This is not strictly true for all activation functions, as, e.g., ReLU is not differentiable at 0. Due to the limited numerical precision, such values can be reached with finite probability, so that surrogate derivatives need to be chosen. In the case of ReLU, most libraries choose $ReLU'(0) = 0$. [36]

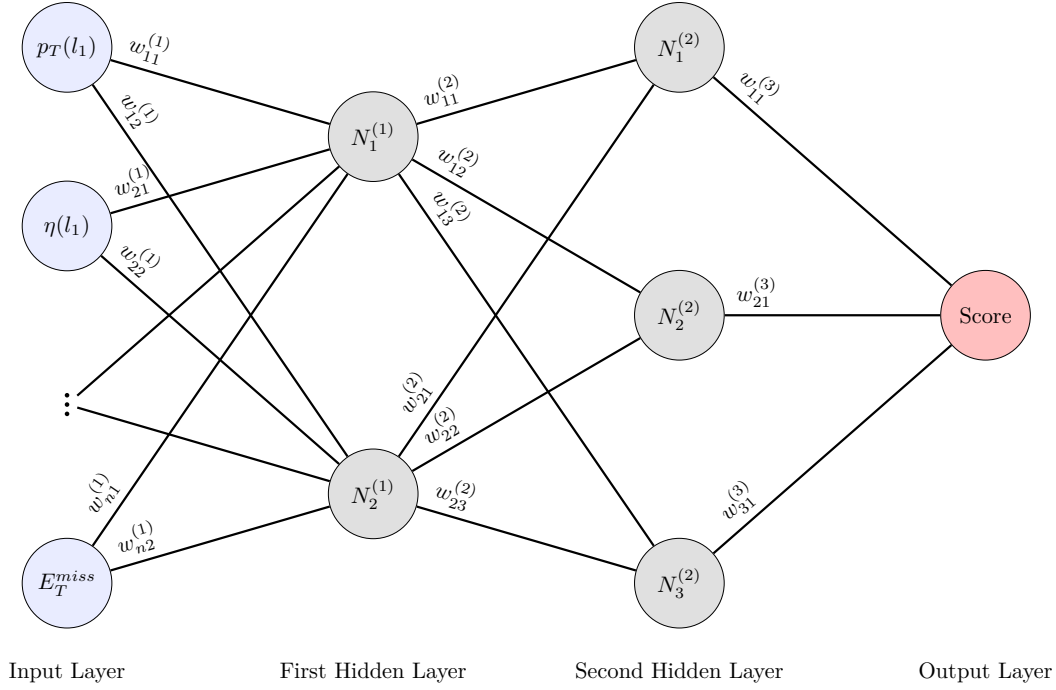


Figure 3.4: Schematic depiction of a multilayer perceptron with two hidden layers, two neurons in the first hidden layer, three neurons in the second hidden layer, and a single output neuron. Each circle represents an artificial neuron, and each line corresponds to a connection between two neurons. Each connection has a weight $w_{ij}^{(l)}$ assigned to it. Figure taken from Ref. [34]

because it recomputes many very similar gradients before the model's parameters are updated, resulting in very slow convergence. Additionally, batch gradient descent can only reach the nearest local minimum, which may be a suboptimal solution.

By dividing the training dataset into smaller chunks, so-called batches, calculating the average gradient of the loss function on this batch, and then updating the model parameters, the redundancy is significantly reduced. This *mini-batch gradient descent* – often called *SGD* in the literature – greatly speeds up the convergence by applying multiple parameter updates per epoch. Additionally, due to the higher variance of the parameter updates introduced by averaging the gradient over a smaller sample, SGD can jump out of local minima and, therefore, potentially reach better minima. However, for very small batch sizes, the fluctuation of the gradient can ultimately prevent convergence to the exact minimum. It is, therefore, essential to carefully choose the batch size.

With N_{batch} the number of training events per batch, the update rule for SGD can be formulated as:

$$w_{ij,t}^{(l)} = w_{ij,t-1}^{(l)} - \frac{\alpha}{N_{\text{batch}}} \sum_{k=1}^{N_{\text{batch}}} \left[\frac{\partial L(f(x_k))}{\partial w_{ij}^{(l)}} \right]_{f=f_{t-1}}, \quad (3.18)$$

or when using θ as the vector of model parameters:

$$\theta_t = \theta_{t-1} - \frac{\alpha}{N_{\text{batch}}} \sum_{k=1}^{N_{\text{batch}}} \nabla_{\theta} L(f_{t-1}(x_k)). \quad (3.19)$$

The parameter α is called the *learning rate* and controls the step size of each update.

Although it is certainly possible to successfully train a neural network with vanilla mini-batch gradient descent, various challenges with SGD motivated the development of more sophisticated gradient descent algorithms:

- choosing a good learning rate α can be difficult; too small values result in very slow or even suboptimal convergence, too large values can hinder convergence
- varying the learning rate over the course of the training using learning rate schedulers can improve convergence while significantly reducing the training time. However, the schedules must be defined before the training and can thus not be dynamically adapted to the dataset
- avoiding getting trapped in suboptimal local minima or saddle points can be very challenging; especially saddle points, in the vicinity of which the gradient is particularly small, are notoriously hard for SGD to escape

An overview of how different gradient descent algorithms systematically tried to overcome these issues is given in Ref. [32].

Below, the gradient descent algorithm that was used in this thesis, the *Adaptive Moment Estimation* optimizer (ADAM), is introduced briefly; a detailed description can be found in Ref. [37]. ADAM tries to solve SGD's shortcomings by keeping track of past gradients and squared gradients and uses them to

- incorporate the past gradients into the parameter update: the direction of the update does not only depend on the current gradient but on past gradients too. This is usually compared to the momentum of a ball rolling down a hill, giving this addition the name *momentum*. The momentum term allows ADAM to skip over shallow local minima and escape saddle points.
- scale the learning rate α dynamically and independently for each parameter based on the fluctuation of the gradient: when the gradient is similar for many iterations, increase the step size; when the gradient is fluctuating, which happens in the vicinity of a local minimum, reduce the step size. This allows a larger step size for the initial phase of the training while automatically decreasing the learning rate near the end, resulting in significantly reduced training times.

A single parameter update of ADAM can be written as:

$$\begin{aligned}
g_t &= \frac{1}{N_{\text{batch}}} \sum_{k=1}^{N_{\text{batch}}} \nabla_{\theta} L(f_{t-1}(x_k)), \\
m_t &= \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t, \\
v_t &= \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2, \\
\hat{m}_t &= \frac{m_t}{1 - \beta_1^t}, \\
\hat{v}_t &= \frac{v_t}{1 - \beta_2^t}, \\
\theta_t &= \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}.
\end{aligned} \tag{3.20}$$

Here, α, β_1, β_2 and ϵ are scalar constants. In the first line, the average gradient on the batch is calculated, just like for SGD. In the second and third lines, estimates for the first and second moments of past gradients are derived as exponential moving averages of the gradient and the squared gradient (g_t^2 denotes the element-wise square $g_t \odot g_t$). The constants β_1 and β_2 define the decay rates and, thus, the importance of past gradients. Since m_0 and v_0 are initialized to zero, these estimates are strongly biased towards zero for the first few iterations of the optimization; this is corrected in lines four and five. In the last line, the parameter update is calculated. Here, the square root and the division denote element-wide operations; the constant ϵ is added for numerical stability. The authors of ADAM suggest default values of $\alpha = 0.01$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$.

A summary of the full training procedure is given as a schematic depiction in Fig. 3.5.

3.3.3 Generalization, Underfitting & Overfitting

In the previous two sections, boosted decision trees and multilayer perceptrons were introduced. In the discussion of the algorithms that are used to train these models, the goal of the optimization was formulated as a minimization of a loss function on the training dataset. However, for any practical application, minimizing the error of the classifier on known, labeled data is not the ultimate goal; instead, training a model that can generalize to new, unseen data is desired. A detailed discussion of the generalization of machine learning models and the associated challenges is given in chapters 5.2 and 7 of Ref. [33], on which this section is based. In order to estimate the generalization error of a machine learning model, its performance is evaluated on a separate dataset not used for training, the *testing dataset*. When assuming that the two datasets are independently generated by the same underlying process, meaning that they are *independent* and *identically distributed* (“i.i.d assumptions”), the expected testing error is larger or equal to the expected training error. In practice, the testing dataset is often generated by splitting off a random fraction of the available dataset – when the resulting testing

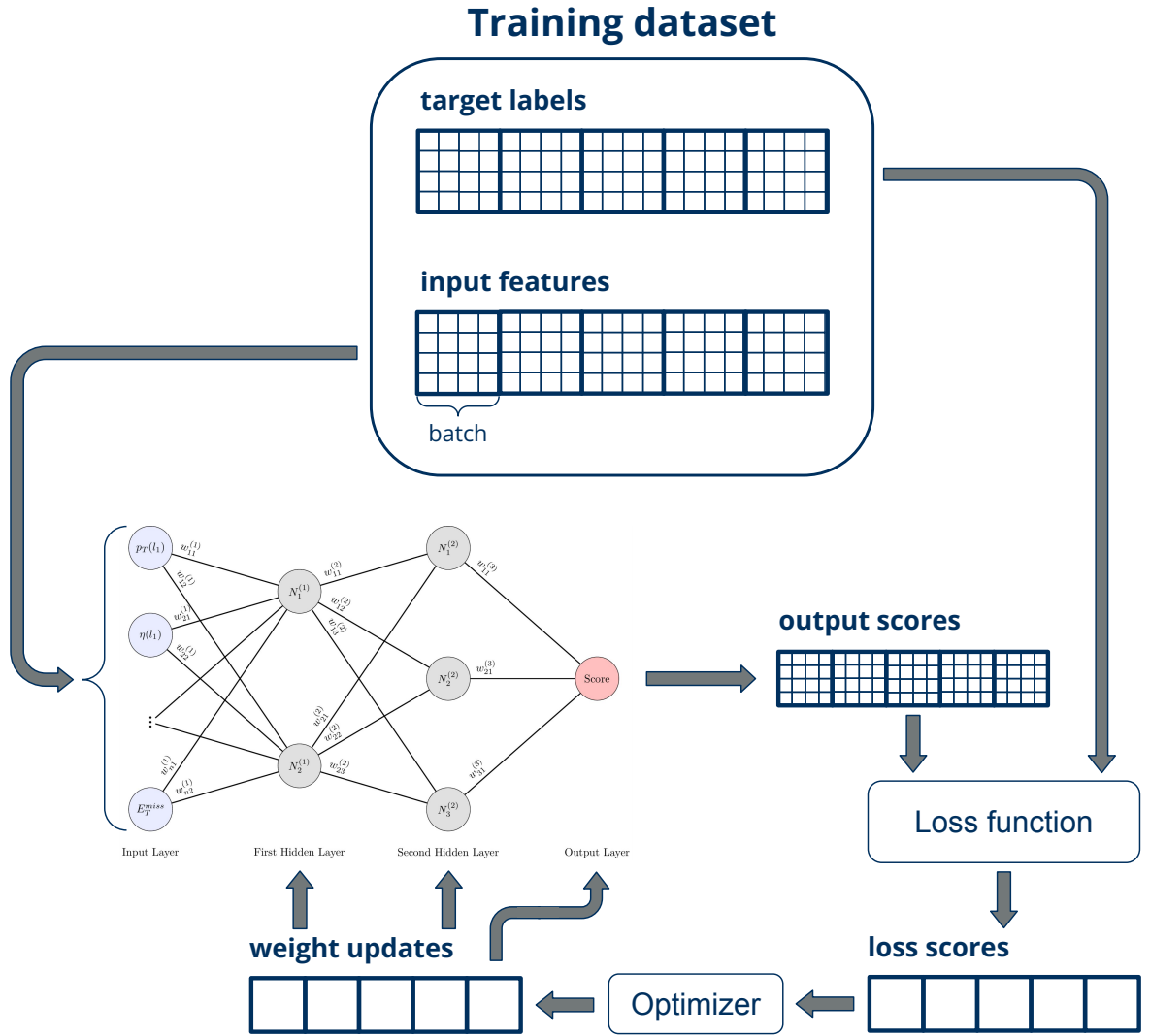


Figure 3.5: Schematic depiction of a single training epoch of a multilayer perceptron using gradient descent. The depiction of the MLP is taken from Ref. [34]

dataset is large enough, the i.i.d assumptions are usually fulfilled.

When an appropriate testing dataset is available, finding the model with the lowest possible generalization error consequently means to simultaneously

- minimize the training error, and
- minimize the difference between the training and testing error

These two goals are directly tied to two important challenges in machine learning: *Overfitting* and *Underfitting*. A model is overfitting when the difference between the training and testing error is too large, while a model that is not able to reach a low enough training error is underfitting. Both depend closely on the *capacity* of a model: a model with low capacity is prone to underfitting since it may not be able to follow the underlying distributions in the

training dataset, resulting in a high training error but a small difference between training and testing error (good generalization); a high capacity model, on the other hand, is more likely to overfit as it can not only follow the general trend of the training data but also fit to its noise, resulting in a low training error but a large gap between training and testing error (bad generalization). In both cases, the generalization error is high.

For the best performance on unseen data, the model's capacity must be high enough to solve the problem it is intended for but low enough to prevent the model from fitting to the noise in the training data. The optimal capacity, therefore, depends on the problem's true complexity and the amount of available training data. For a given problem, the generalization error as a function of the model capacity follows a U-shape, which can be seen in Fig. 3.6. A good visualization of this dependency of the generalization error on the model capacity is given in Fig. 3.7 using different order polynomials that are fitted to a small, noisy dataset.

One method to tune a model's capacity to the specific task is to choose its *hypothesis space* or *representational capacity*, which is the set of functions that the model can represent. For the example of fitting a polynomial to a dataset, this corresponds to choosing the degree of the polynomial, which, as seen in Fig. 3.7, has a strong influence on the generalization error. When applied to boosted decision trees, this would mean to carefully set the number of trees `NTrees` of the boosting as well as the two stopping conditions, the maximum number of splits in each chain `MaxDepth` and the minimum fraction of total events in each node `MinNodeSize`, of the growing algorithm for each decision tree. For multilayer perceptrons, the hypothesis space is mainly given by the number of hidden layers and the number of neurons in each layer.

A different way to control the capacity of the model is to employ regularization. Instead of directly limiting the space of functions that are allowed to be solutions of the training procedure, regularization gives the learning algorithm a preference for some solutions over others. This way, all functions in the hypothesis space can still be chosen by the training algorithm, but “unpreferred” solutions need to be a significantly better fit to the training data in order to be selected. For boosted decision trees, examples of regularization include the shrinkage parameter (to reduce the weight of each individual tree in the expansion) and the bagging technique (to train each tree on a random subset of the training data). Both were introduced in Sec. 3.3.1. For neural networks, a popular type of regularization are parameter norm penalties where a penalty term $\Omega(\theta)$ is added to the loss function:

$$\tilde{L}(f(x), y; \theta) = L(f(x), y; \theta) + \lambda \Omega(\theta). \quad (3.21)$$

The strength of the regularization is controlled by the parameter $\lambda \in [0, \infty)$. Typically, the L^1 or L^2 norms are used, consequently giving the names *L1-regularization* and *L2-regularization* and resulting in the regularization terms $\lambda_1 \sum_i |\theta_i|$ and $\lambda_2 \sum_i \theta_i^2$ respectively. Another widely used regularization technique for neural networks is *Dropout*, where for each training iteration, the outputs of a random subset of neurons is set to zero. This approximates the effect of training a very large

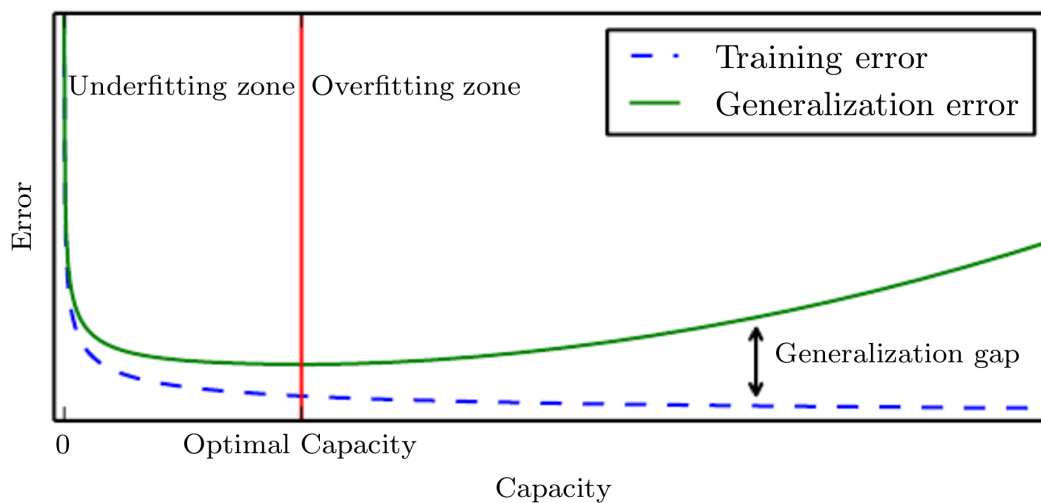


Figure 3.6: Generalization and training error as a function of model capacity. Figure taken from Ref. [33].

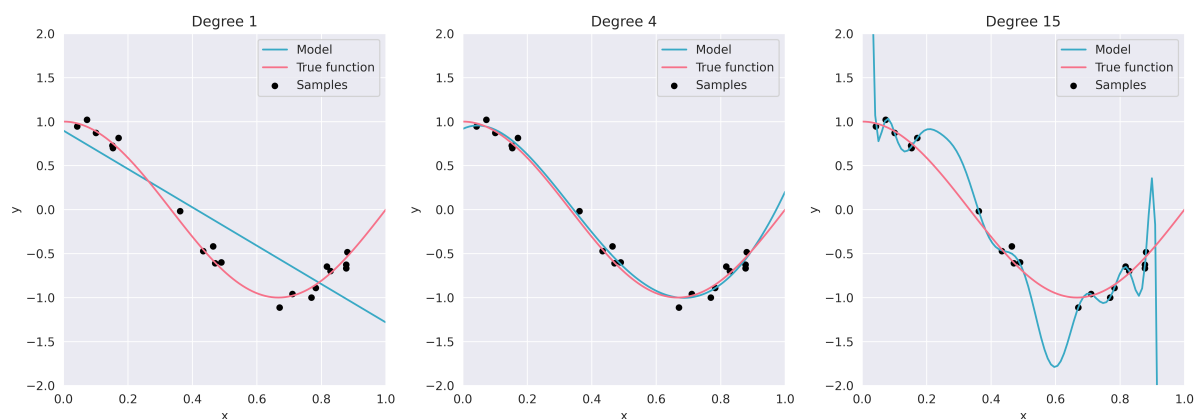


Figure 3.7: Visualization of the dependence of the generalization error on the model capacity using degree 1, 4, and 15 polynomials that are fitted to 20 data points following a cosine function plus Gaussian noise. The degree 1 polynomial on the left cannot follow the data samples, resulting in large training and testing errors (underfitting). The degree 15 polynomial on the right is capable of following the data points very closely and has fitted to the noise in the data, resulting in very low training error and high testing error (overfitting). The degree 4 polynomial in the middle follows the true function almost perfectly, resulting in low training and testing errors (optimal capacity).

ensemble of smaller networks on subsets of the training data and averaging their predictions to get the final output. [39] A different type of regularization is a technique called *Early Stopping*. When training sufficiently complex models that have the capacity to overfit the training data, the training error steadily declines over the course of the training while the generalization error reaches a minimum, after which it will increase again. Early Stopping tries to terminate the training when the generalization error of the model is minimal by calculating the error on a third independent dataset, the *validation dataset*, at the end of each epoch. Under the same i.i.d. assumptions as for the testing dataset, the expected generalization error is minimal when the validation error is minimal, consequently terminating the training once the validation error starts to increase is expected to result in a model with the best generalization error. An example of the trend of training and validation errors when training an MLP is shown in Fig. 3.8.

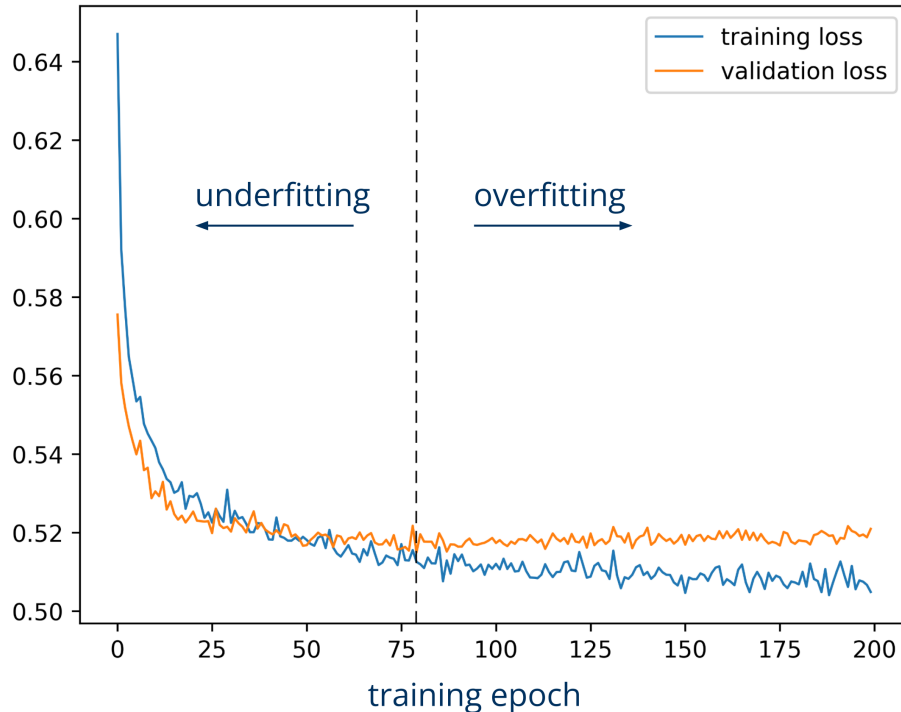


Figure 3.8: Training and validation loss as a function of the training epoch for the training of a multilayer perceptron. The dashed vertical line marks the epoch of the lowest validation loss and consequently the lowest expected generalization error. Terminating the training before reaching this point would result in an underfitted model while continuing the training for longer will result in overfitting.

4 Distributed Hyperparameter Optimization for Neural Networks

In the previous chapter, an introduction to boosted decision trees and artificial neural networks was given, and their respective training algorithms were discussed. It was shown that the generalization performance of machine learning methods depends on the model capacity and has to be tuned to the given task to prevent over- and underfitting. This is done by carefully selecting the representational capacity of the model and by employing regularization. In contrast to the model parameters learned during the model's training, the parameters that govern the representational capacity and the strength of the regularization – the so-called *hyperparameters* – need to be set manually and are fixed throughout the training. The process of optimizing the values of these hyperparameters to achieve the best generalization error is called hyperparameter optimization. In this chapter, multiple hyperparameter optimization algorithms will be discussed, and a framework to perform distributed hyperparameter optimization, tailored to the use in particle physics analyses, will be presented. The source code of this framework is available at <https://gitlab.cern.ch/atlas-germany-dresden-vbs-group/optima>.

4.1 Hyperparameter optimization algorithms

By far the most commonly used method of searching for a good set of hyperparameters is manually trying different values. Given a decent starting point and a solid understanding of both the machine learning technique and the training algorithm that should be used, each hyperparameter can be optimized by interpreting the validation error¹ and the gap between training and validation error, deciding if the model's capacity should be increased or decreased and adjusting the hyperparameters accordingly. However, especially for large numbers of hyperparameters – neural networks, for example, can easily have tens of hyperparameters – or when no good starting point is known, this can be a very challenging and time-consuming endeavor. Since the search for the best hyperparameters is in itself an optimization problem, automated procedures can be used to reduce the manual labor and the need for previous

¹Typically, the validation dataset is used for hyperparameter optimization instead of the testing dataset. Since the hyperparameters are chosen based on which parameters result in the lowest validation error, meaning they are learned from the validation dataset, the validation error will underestimate the generalization error to some degree. Therefore, the testing dataset is left out for the final check at the very end. [33]

experience in exchange for a potentially higher computational cost. [33]

One of the simplest search algorithms that can be used for hyperparameter optimization is *grid search*. For each hyperparameter to tune, a finite set of values is specified by the user. The grid search algorithm will then train a machine learning model for each element in the Cartesian product of these sets. The outcome of the optimization is the hyperparameter combination that resulted in the lowest validation error. Due to its simplicity, grid search can be easily parallelized, requiring minimal communication between parallel instances. However, since the number of combinations that need to be tried grows exponentially with the number of hyperparameters, grid search is most useful when the number of hyperparameters is small, while large searches may not be possible due to the exponentially increasing computation cost. [33]

In order to reduce the necessary computation for large numbers of hyperparameters, instead of specifying fixed values for each parameter, the marginal distribution of each hyperparameter can be provided. These are typically chosen to be uniform or log-uniform distributions. In order to choose a hyperparameter combination, each marginal distribution is sampled to give the joint hyperparameter setting. This technique is called random sampling and was shown to converge significantly faster to a good solution for cases where some hyperparameters have a significantly greater influence on the generalization error of the model than others, resulting in a lower *effective dimensionality* of the problem. The approach is visualized in Fig. 4.1 for the example of two hyperparameters with only one of them being important. In cases where all hyperparameters are important, random search performs similarly to grid search when the number of hyperparameters is sufficiently large. Like grid search, random search can be trivially parallelized to use available resources efficiently and should therefore be preferred to grid search when the number of hyperparameters is large. [33, 40]

Since random search does not make any assumptions about the optimization problem at hand and, given enough time, will produce results arbitrarily close to the optimum, it can act as a useful baseline. However, it is usually outperformed by other, more sophisticated optimization algorithms [41–43]. Chapter 1 of Ref. [41] outlines the developments in hyperparameter optimization algorithms over the past ten years. Although no single approach proved superior for every application, *Bayesian optimization* was demonstrated to give very good results on many different problems.

Bayesian optimization is a model-based optimization algorithm. The idea is to build a probabilistic model $p_M(y|\theta)$ of the validation error y as a function of the hyperparameters θ based on past observations. Using this model, the most promising hyperparameter combinations to sample next can be extracted. This is done by maximizing an *acquisition function*, most commonly the *expected improvement*, which for minimization is defined as:

$$\text{EI}_{y^*}(\theta) := \int_{-\infty}^{y^*} (y^* - y) p_M(y|\theta) dy, \quad (4.1)$$

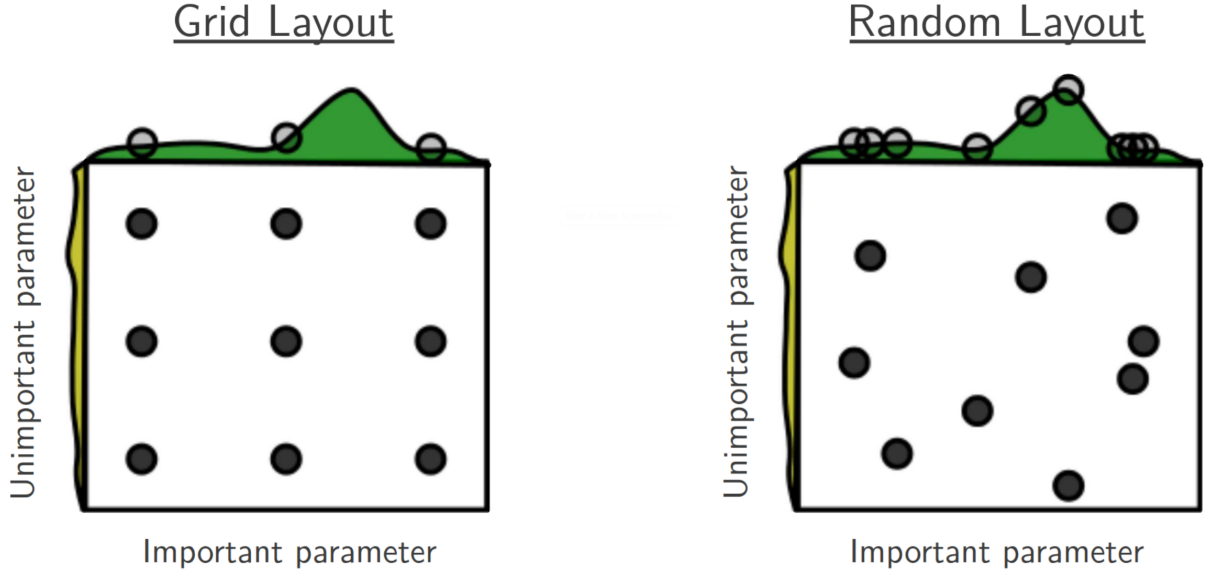


Figure 4.1: Comparison of grid search and random search for a 2D optimization problem where the final score is dominated by only one of the two parameters. In this example, the score is given as $f(x, y) = g(x) + h(y) \approx g(x)$. Above each square, $g(x)$ is shown in green while $h(x)$ is drawn in yellow to the left of each square. Since for nine trials, grid search evaluates $h(x)$ only at three positions, the resulting score is not optimal. For random search, each function is sampled at nine distinct positions, resulting in a higher score because $h(x)$ is sampled better while having the same computational cost. Figure taken from Ref. [40]

with y^* being the best validation error seen in previous iterations. In contrast to grid search and random search, this approach can therefore utilize the knowledge acquired during previous iterations to explore more promising regions of the hyperparameter space. This can be seen in Fig. 4.2 where grid, random and Bayesian search are applied to the minimization of a 2-dimensional function. [41]

The function used to model $p(y|\theta)$ is called the *surrogate function*. A common example of this is the Gaussian process. In Fig. 4.3, the process of searching for the minimum of a simple function using a Gaussian process and the expected improvement is shown to illustrate how Bayesian optimization works. In this thesis, however, *Tree Parzen Estimators* (TPEs) were used to model the validation error as they have been shown to outperform Gaussian processes, scale better to higher numbers of data points, and can handle categorical search spaces (e.g., when optimizing the activation function of a neural network) more easily [41, 42]. Instead of directly modeling $p(y|\theta)$, TPEs apply Bayes' theorem:

$$p(y|\theta) = \frac{p(\theta|y)p(y)}{p(\theta)}, \quad (4.2)$$

and instead model $p(\theta|y)$ and $p(y)$. Using two kernel density estimations $\ell(\theta)$ and $g(\theta)$, $p(\theta|y)$

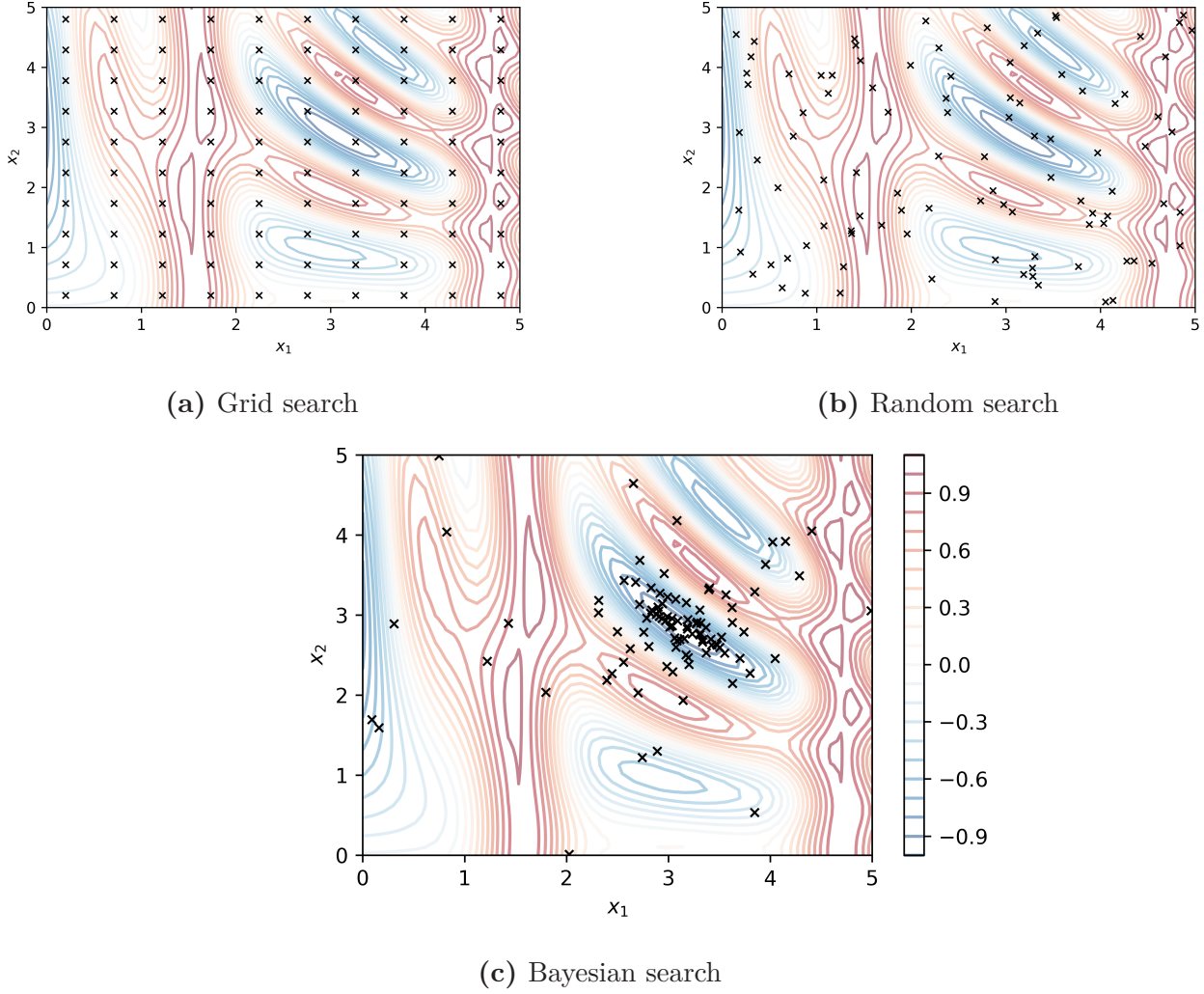


Figure 4.2: Visualization of different hyperparameter optimization algorithms applied to the minimization of $f(x, y) = -(\sin(x))^{10} - \left(1 - 0.1(y - 3)^2\right) \cos(10 + y * x) \cos(x)$ for $0 \leq x, y \leq 5$. In the contour plots, red areas belong to function values near 1, while blue areas belong to function values near -1. For the Bayesian search, the `TPESampler` from the Optuna framework [44] was used. For all three algorithms, 100 trials were sampled. Since $f(x, y)$ depends strongly on both x and y , random search does not have an advantage over grid search. The Bayesian search, however, could sample the function much more efficiently.

is represented as:

$$p(\theta|y) = \begin{cases} \ell(\theta) & \text{if } y < y^* \\ g(\theta) & \text{if } y \geq y^* \end{cases}. \quad (4.3)$$

The specific choice of what type of kernel density estimator to use depends on the chosen search space $p(\theta)$; details can be found in Sec. 4.2 of Ref. [42]. The density $\ell(\theta)$ is build using observations $\{\theta^{(i)}\}$ where the corresponding validation error $L(\theta^{(i)})$ was below a threshold y^* and $g(\theta)$ is build from the remaining observations. This divides $p(\theta|x)$ into a distribution of “good” and a distribution of “bad” observations. The threshold y^* is chosen to be a certain quantile α of the observed values, meaning that $p(y < y^*) = \alpha$. $p(y)$ itself is also modeled using a kernel density estimator, the specific choice of which is not important according to Ref. [42]. When using the expected improvement acquisition function with TPE, it can be shown that:

$$EI_{y^*}(\theta) = \int_{-\infty}^{y^*} (y^* - y) \frac{p(\theta|y)p(y)}{p(\theta)} dy = \frac{\alpha y^* - \int_{-\infty}^{y^*} yp(y)dy}{\alpha + (1 - \alpha) \frac{g(\theta)}{\ell(\theta)}} \propto \left(\alpha + (1 - \alpha) \frac{g(\theta)}{\ell(\theta)} \right)^{-1}. \quad (4.4)$$

The maximization of the expected improvement, therefore, corresponds to a minimization of the fraction $g(\theta)/\ell(\theta)$, meaning that hyperparameter combinations θ that are much less likely to be “bad” than to be “good” should be sampled. [42]

For all discussed optimization algorithms, the hyperparameters were selected and evaluated by fully training the corresponding machine learning model. As a result, the hyperparameters for each training instance remain constant for the entire training. *Population Based Training* (PBT) [45] is an optimization algorithm that searches for a more general *hyperparameter schedule* which describes the value of each hyperparameter as a function of the training iteration. For that, PBT creates a *population* of models that are trained in parallel and whose hyperparameters are initially selected using random sampling. After a certain number of training iterations, the *perturbation interval*, all models are evaluated and compared. Good-performing models continue training while under-performing models undergo an *exploitation-exploration* cycle. The specific method for comparing and selecting the models and doing the exploitation and exploration depends on the problem; multiple options are given by the authors in Ref. [45]. For this thesis, the members of the population are ranked by their validation errors. Each member in the bottom k fraction of the population is then replaced by a copy of a randomly selected model from the top k fraction (exploitation). This copy includes both the training progress (e.g., the weights $w_{ij}^{(l)}$ and biases $b_i^{(l)}$ for MLPs) and the hyperparameters. After that, each hyperparameter of the copied model is either increased or decreased by multiplying it with a perturbation factor (e.g., 1.2 or 0.8) or resampled from the corresponding marginal distribution (exploration). The training is then continued for another perturbation interval, after which the process is repeated. This is done until all models are fully trained. A visualization of the described optimization procedure can be seen in Fig. 4.4.

While the hyperparameters schedule that is found by PBT has been shown to outperform the

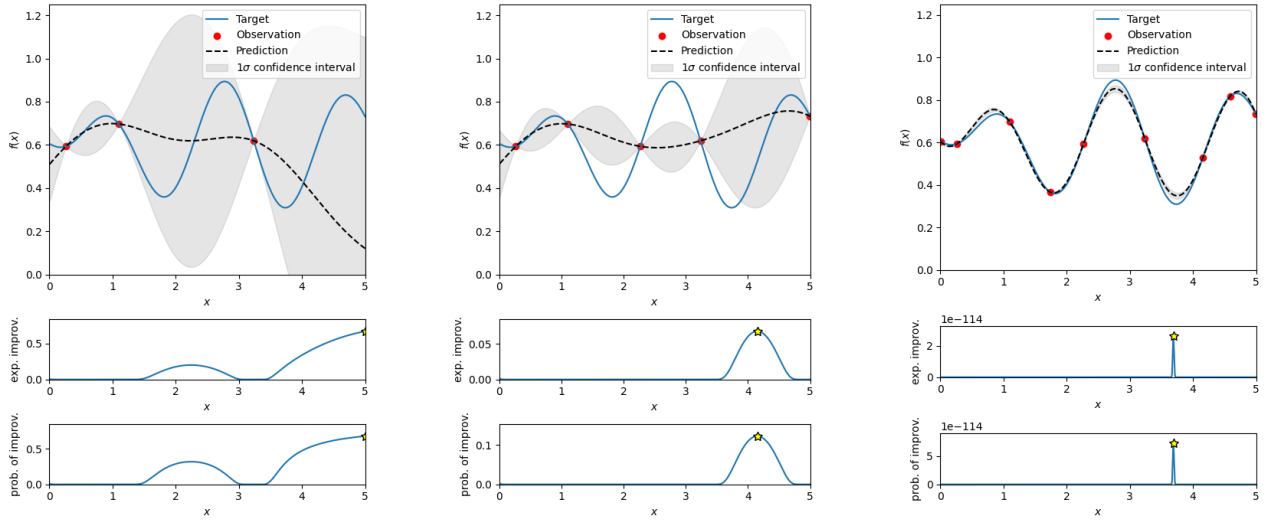


Figure 4.3: Bayesian optimization using a Gaussian process applied to the minimization of an example function $f(x) = -0.3 \left((\sin(3.2))^{10} + \left(1 - 0.1 (x - 3.2)^2 \right) \cos(10 + 3.2x) \right)$ in the range $0 \leq x \leq 5$. The optimization state is shown after 3, 5, and 9 observations (left, middle and right plots, respectively). In each plot, the target function $f(x)$ is drawn in blue, and the points where $f(x)$ was already sampled (observations) are drawn in red. The dashed line corresponds to the mean of the Gaussian process, and the gray area denotes the 1σ uncertainty region. Below each plot, the expected improvement (see Eq. 4.1) and the probability of improvement (defined as $PI(x) = \int_{-\infty}^{y^*} p(y|x)dy$) is shown. In order to select the next point to sample, the expected improvement was used. The `GaussianProcessRegressor` from the `scikit-learn` library [38] got utilized to fit the Gaussian process.

results of other Bayesian optimization approaches for some applications [45], it also limits the hyperparameters that can be optimized with PBT as not all hyperparameters can be adjusted without restarting the training (e.g., the number of layers in an MLP). Additionally, since PBT depends on the training progress of a population of models, PBT can only be used with a sufficiently large number of parallel training instances, in contrast to the other discussed optimization algorithms.

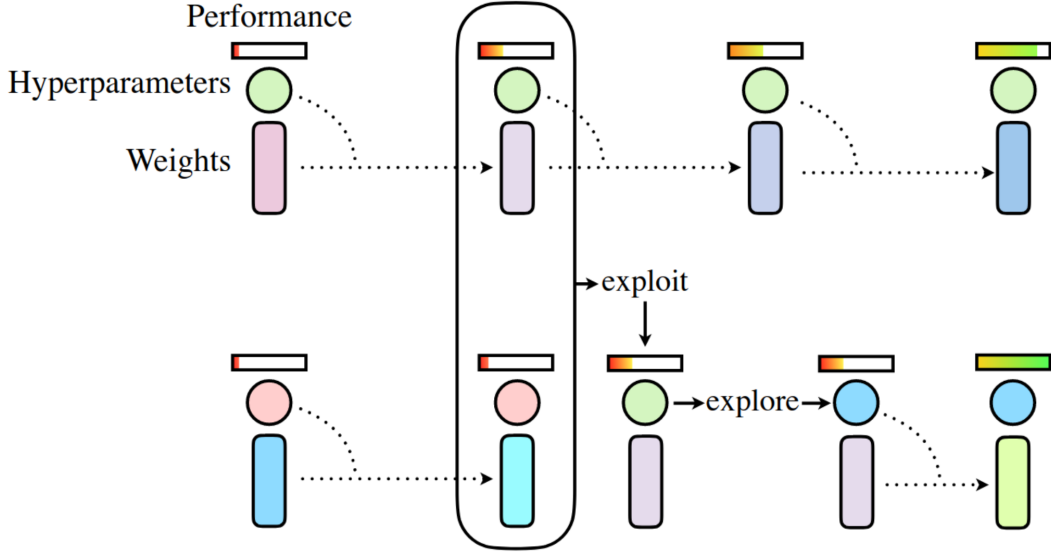


Figure 4.4: A visualization of the optimization procedure when using Population Based Training. The performance of each member of the population is represented by the corresponding bar, while the values of its hyperparameters and weights are symbolized by the color of the corresponding circle and rectangle. Figure taken from Ref. [45]

4.2 Hyperparameter optimization framework

The framework developed as part of this thesis to perform the hyperparameter optimization of neural networks was built using the open-source Python library *Tune* [46]. *Tune* is a library for scalable hyperparameter search and provides a flexible API to optimize models from a large collection of machine learning frameworks. Instead of providing its own search algorithms, it natively integrates with many different hyperparameter optimization libraries, providing a unified interface to switch between different optimization algorithms easily. *Tune* runs on top of *Ray*, an open-source platform for distributed computing of machine learning and general Python applications. *Ray* enables the quick and flexible assembly of computing resource clusters by launching *Ray nodes* on each machine and handles the scheduling of computing tasks. This facilitates the execution of many parallel training instances across multiple machines while allowing flexible resource requirements for each individual instance, which, together with the built-in interface to multiple optimization libraries, was the primary motivation to use *Tune* for this thesis.

In its current state, the optimization framework is built to optimize the hyperparameters of Keras [47] neural networks for supervised learning tasks. In order to keep the framework as flexible as possible, large parts of it can be configured using a `config`-file. For multilayer perceptrons applied to binary classification, which is the application that was used for this thesis, built-in default functions are implemented so that only minimal configuration is necessary to apply the framework to a given problem. However, by providing the functions to load and preprocess the training data as well as the `build`, `update`, `compile` and `evaluate` functions – all of which are discussed in this section – in the `config`-file, the framework can easily be extended to arbitrary Keras model architectures and supervised learning tasks. Consequently, the following sections cover both general concepts that apply to all types of optimization tasks that can be performed with this framework and details specific to the use of MLPs for binary classification.

4.2.1 Overview of the optimization

A single hyperparameter optimization run consists of up to three optimization steps: an input variable selection, the main hyperparameter optimization using Bayesian optimization, and a “fine-tuning” step using Population Based Training.

Initially, the input data must be loaded into memory and prepared for the training. When using vectorized input data, the built-in preprocessing function that scales and splits the input data can be used, details of which are described in Sec. 4.2.3.

If it is desired to minimize the number of inputs of the model, an input variable selection can be performed to remove inputs from the training data that do not improve or even impair the training result. Since this selection should preferably be made using a model representative of the final result of the optimization, a pre-optimization using the same settings as for the main optimization, but with fewer trials, is performed. The result of this pre-optimization can then be used to gauge the importance of each input variable. For vectorized input data with independent input vector components, meaning that the input vector does not have any internal structure and stays meaningful when a single input variable is removed², the built-in variable selection algorithm that is explained in Sec. 4.2.6 can be used. Once the important input variables are determined, the remaining input variables are dropped from the datasets before the next optimization is started.

Both the pre- and the main optimization use Bayesian optimization to find good hyperparameters. Sec. 4.2.4 describes the procedure followed during both optimization steps. At the end of each optimization, an evaluation function is called to extract the best hyperparameters, perform a k-fold cross-validation, and evaluate the performance of the obtained models; a detailed description of this evaluation step can be found in Sec. 4.2.5.

²This is, for example, not the case when the inputs encode a time series or are subdivided into groups (e.g., every block of four input variables encodes the four-momentum of a particle).

As a final optimization step, all hyperparameters that can be changed throughout the training get re-optimized using Population Based Training. The remaining hyperparameters are fixed to the values derived during the main optimization. The general procedure of the optimization is identical to the previous two optimizations, while the evaluation step differs slightly; these differences will also be discussed in the corresponding section 4.2.5.

The neural networks trained during the cross-validation at the end of each optimization step are saved to disk for later use in production. The results of the evaluations for each step are saved as clear-text files.

4.2.2 Custom metrics & overfitting conditions

In order to define a goal of the optimization, metrics are needed to assess how well the neural networks perform. Keras includes a long list of different metrics to judge the performance of neural networks for general classification or regression tasks. These metrics can get directly used within the optimization framework. For that, a reference to the corresponding metric's class together with a list of necessary arguments and a unique name for the metric (e.g. for use in the results files) is to be added to the configuration file. However, depending on the application, constructing metrics more specific to the task at hand can be very useful. In order to allow the definition of a wide variety of metrics, all functions that accept an array of neural network predictions, the array of corresponding target labels, and an array of event weights and returns either a number or a Boolean can be used to define a *custom metric*. Like the Keras metrics, they are defined by providing a function reference and a name in the configuration file. The values of all custom metric are calculated separately on the training and the validation dataset at the end of each epoch.

As discussed in Sec. 3.3.3, overfitting is characterized by both the training error as well as the gap between training and generalization error. For applications sensitive to the difference between training and generalization performance, it may be crucial to limit this gap. For example, it is not uncommon to reuse the training data of a BDT or neural network classifier for the profile likelihood fit in particle physics analyses. Consequently, a difference between the classifier performance on the training data and new, unseen data – which, in this case, is the measured data – can be detrimental to the validity of the prediction. Since the validation dataset is not used for the training, comparing the value of a metric on the training and the validation datasets can be used as a measure for the generalization gap. By providing a list of metric names together with a function that accepts the values of these metrics, multiple metrics can get combined to form *composite metrics*. Utilizing such a composite metric provides a flexible way to quantify the difference between the training and validation performance. This can, for example, be used to include a “penalty” term for large generalization gaps directly in the target metric that is used for the optimization. Like custom metrics, composite metrics are defined in the configuration file, can return either a number or a Boolean, and are evaluated at

the end of each epoch.

In order to set a hard limit on the difference between training and validation error, *overfitting conditions* as special composite metrics can be defined in the configuration file. Their return value should be **True** when the generalization gap is too large and **False** otherwise. Overfitting conditions are evaluated at the end of each epoch and influence the termination of the training as part of the Custom Early Stopping described in Sec. 4.2.4.

4.2.3 Input data preprocessing

Multilayer perceptrons have been shown to converge faster when the input variables are scaled to have zero mean and the same variance [48]. Additionally, many weight initialization schemes expect the inputs to be normally distributed [49, 50]. Since the input variables usually follow very different distributions, data preprocessing is necessary. The built-in preprocessing that can be used with vectorized input data works in two steps. First, a user-specified non-linear transformation is applied to each input variable – this can be used to make the distributions of each input variable as symmetric as possible. After that, the inputs are automatically scaled to a mean of zero and a variance of one using a linear transformation. Figure 4.5 shows the distribution of an example input variable before and after the scaling.

When using event weights, weights for signal events are reweighted to the same total weight as background events to ensure the same contribution from both classes during training. Additionally, the average weight is normalized to a value of 1 to circumvent any numerical issues that may arise from very low or very high average weights.

Once the inputs are properly scaled and the event weights are adjusted, the dataset needs to be split into the training, validation, and testing datasets. This splitting can be done randomly, with the fractions of events to be used for each dataset configurable in the configuration file. Alternatively, an array of event numbers can be provided together with the input data, which assigns a number to each event in the input data. Using these event numbers and a custom splitting condition that can be defined in the configuration file, more control can be taken over how the input data is split. This is, for example, useful when doing event classification in a particle physics analysis to ensure that different processes are equally represented in each dataset. By default, the following conditions are evaluated for each event in order to assign it to a dataset

- training: $(\text{EventNumber} + C) \bmod N \neq 0$ and $(\text{EventNumber} + C + 1) \bmod N \neq 0$
- validation: $(\text{EventNumber} + C) \bmod N = 0$
- testing: $(\text{EventNumber} + C + 1) \bmod N = 0$

These selections result in three disjoint datasets. The parameter N defines the fraction of events that are used for training (e.g., for $N = 5$, this will be a 60:20:20 split between training,

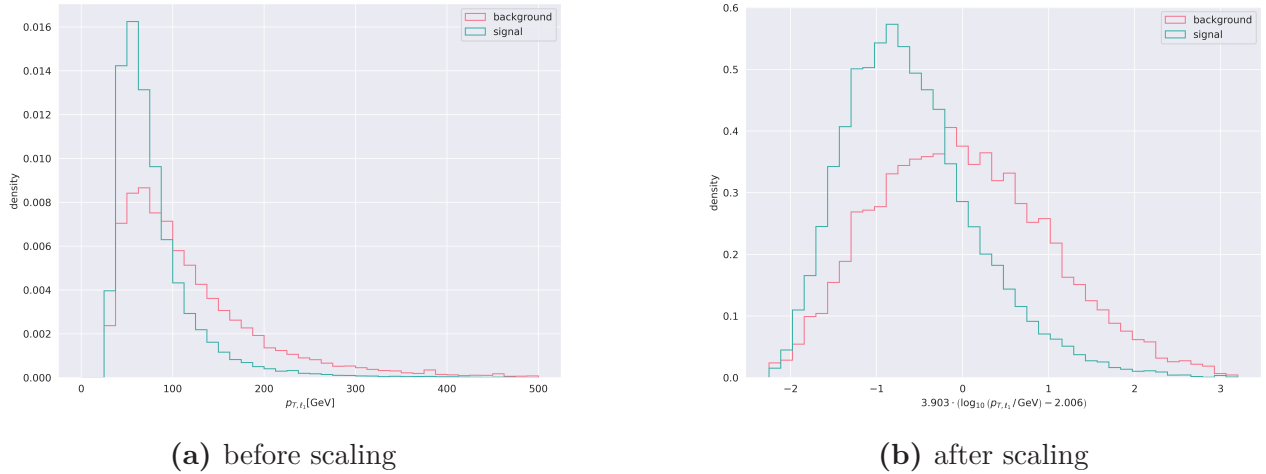


Figure 4.5: Distribution of the transverse momentum of the leading lepton for the signal and background processes described in 2.2.1 before and after the input variable scaling. The distributions of transverse momenta typically show a long tail towards higher momenta values. By applying a logarithmic transformation, most of the asymmetry can be corrected, resulting in a distribution much closer to a Gaussian. Since each event in the dataset is weighted equally when determining the parameters of the linear transformation, unequal numbers of events per class when performing classification can result in larger absolute mean values for underrepresented classes, which is the reason for the shift of the signal distribution in the right plot.

validation, and testing) while the parameter C acts as an offset, resulting in N possible splittings that are used for cross-validation (see Sec. 4.2.5). When not using the built-in splitting conditions, the custom splitting condition needs to be defined in a way that allows multiple different splitting in order to be able to use cross-validation.

4.2.4 Experiment execution

In Tune, each training instance is called a *trial*; an optimization that combines multiple trials to find the best hyperparameters (or hyperparameter schedule) is called an *experiment*. To run an experiment, Tune needs to be provided with a *search algorithm* to select the hyperparameters for each trial, a *search space* to sample the hyperparameters from, a *trial scheduler* that schedules which trials are executed and can manipulate running trials as well as a **trainable** function that is executed to do the actual training.

While the search algorithms and the trial schedulers are included in Tune, the **trainable** function is specific to the used machine learning method and the task it should solve and thus needs to be implemented by the user. It needs to accept the hyperparameters as an input, build the corresponding machine learning model, and perform the training. Tune controls each trial in what the developers call a “cooperative control mode”: the **trainable** is expected to *report* the training progress to Tune in regular intervals, e.g., at the end of each training epoch, and to check if Tune wants it to save a snapshot of the training state or restore the training

from a previous snapshot. The intermediate training results are used by the search algorithm together with the results from already terminated trials to select the hyperparameters for new trials. Optionally, the training of underperforming trials can also be terminated early using a suitable trial scheduler. The ability to save and restore the training state is needed by some trial schedulers that rely on changing a trial’s hyperparameters throughout the training, an example of which is Population Based Training.

In the following sections, the search algorithms and trial schedulers used in this framework are presented, as well as the built-in `trainable` function and its dependencies.

Search algorithms and Trial schedulers

The default search algorithm that is used for the pre- and the main optimization step is the `TPESampler` from the Optuna library [44]. It implements an optimization algorithm based on Tree Parzen Estimators (see Sec. 4.1) and was found to perform well for the optimizations done in this thesis. To reduce the computation cost, the `ASHAScheduler` that implements the *Asynchronous Successive Halving Algorithm* [51] is used to early stop underperforming trials based on their intermediate training results. ASHA is an asynchronous modification to the popular Hyperband algorithm that is based on the assumption that “good” trials will perform above average for the entire training. Hyperband divides all trials in an optimization into multiple “rungs” based on their performance throughout the training. All trials are initially trained for a certain minimum number of epochs r , after which their validation errors are compared. For a given reduction factor η , e.g., 2, the best performing $1/\eta$ fraction of trials can continue training (are *promoted* to the next rung) while the remaining trials get terminated. This reduction is repeated after every $r\eta^i$ epochs, with i being the number of reductions that have already occurred. This ensures that for each rung, the same resource budget is allocated. ASHA modifies this approach by determining the decision thresholds for each rung on the fly, thereby removing down-times that occur while waiting for all trials in a rung to finish. Both the `TPESampler` and the ASHA scheduler can support many parallel trials as well as a large number of total trials in the experiment, allowing efficient usage of available HPC computing resources. This has been verified as part of this thesis by performing optimizations with up to 400 parallel training instances on 1 CPU core each and up to 10000 trials in total.

During the fine-tuning step that is performed after the main optimization, Population Based Training is used to try and find hyperparameter schedules that improve upon the fixed hyperparameter combination derived with the TPE-based approach. A variant of the PBT algorithm is implemented as a trial scheduler in Tune. All hyperparameters not changeable during the training are fixed to the optimized values from the previous step. For all remaining hyperparameters, initial values for each member of the population are randomly sampled from the search space using the default `Random Sampler`. After a burn-in period of `burn_in_period` epochs to reduce the influence of the random weight and bias initialization, the iterative

exploitation-exploration procedure described in Sec. 4.1 begins. In Tune, this is realized by interrupting the training of a trial and re-executing the corresponding `trainable`, providing a snapshot of the training state of a different trial that should be restored from and an updated set of hyperparameters. The saving and reloading of snapshots and updating of an existing model will be discussed in the following section. Since the exploitation and exploration for each member of the population are done asynchronously, PBT can scale well to large numbers of parallel trials. This has again been verified in this thesis for population sizes of up to 400 parallel trials.

It is important to note that the perturbation of continuous hyperparameters, which is done by multiplication with one of two constants defined in `perturbation_factors` (e.g., 1.2 and 0.8, see Sec. 4.1), does not respect the bounds defined by the search space and can thus result in invalid values which would need to be corrected in the `trainable` function. This peculiarity does not occur when perturbing hyperparameters with categorical search space, which is done by randomly choosing one of the neighboring values in the search space, or when resampling. The common hyperparameter search space for all optimization steps is defined in the configuration file. It allows all marginal distributions supported by Tune, which includes continuous and discrete uniform and log-uniform, as well as normal and categorical search spaces.

The trainable function

During the experiment, Tune starts the training of each trial by calling the `trainable` function and providing the dictionary of hyperparameters. Within the `trainable`, the training and validation datasets are converted to TensorFlow datasets³ and split into batches of size `batch_size` and the Custom Early Stopper – which will be presented in the following section – is initialized. When instructed to restore from a snapshot, the corresponding Keras model is reloaded from the disk, and the Early Stopper state is updated accordingly.

The `trainable` itself depends on three optimization-specific functions that need to be provided by the user, either by using the built-in variants or by defining them in the configuration file:

- A `build` function that takes the dictionary of hyperparameters and returns a Keras model.
- An `update` function that takes an existing Keras model and the new dictionary of hyperparameters and updates the values of all those hyperparameters that can be changed without losing the training state of the model. This is necessary for the PBT scheduler to adjust the hyperparameters throughout the training.
- A `compile` function that creates the loss function and the optimizer (or updates its

³This needs to be done within the `trainable` because large input parameters need to be serialized, saved to Ray's shared object store, and reloaded on the worker that executes the `trainable`. Tensorflow datasets, however, are not serializable while, e.g., Numpy-arrays are.

parameters when reloading from a snapshot) and binds them to the Keras model (*compiles* the model)

The built-in `build` function can create multilayer perceptrons for binary classification (i.e. MLPs with a single output neuron with logistic sigmoid activation) with the following adjustable hyperparameters:

- The number of fully connected hidden layers `num_layers`.
- The number of neurons in hidden layer i `units_i`.
- The activation function of the hidden layers `activation`. Supported activation functions are
 - logistic sigmoid: $A(x) = 1/(1 + e^{-x}) =: \sigma(x)$
 - hyperbolic tangent: $A(x) = \tanh x$
 - Rectified Linear Unit (ReLU): $A(x) = \max(0, x)$
 - Leaky ReLU: $A(x) = \max(\alpha x, x), 0 \leq \alpha \leq 1$
 - Swish: $A(x) = x \cdot \sigma(\beta x)$ ([52])
 - Mish: $A(x) = x \cdot \tanh(\ln(1 + e^x))$ ([53])
 - SELU: $A(x) = \lambda(\text{ReLU}(x) + (\alpha e^x - \alpha) \cdot \text{ReLU}(-x))$ ([54])
 - SPLASH: $A(x) = \sum_{s=1}^{(S+1)/2} (a_+^S \cdot \text{ReLU}(x - b^S) + a_-^S \cdot \text{ReLU}(-x - b^S))$ ([55])
- The initializers used to set the initial values of the weights matrix and the bias vector of each hidden layer `kernel_initializer` and `bias_initializer`. These can either be selected automatically based on the given activation function or overwritten with any Keras initializer. The automatic choices are as follows:
 - logistic sigmoid: $w_{ij}^{(l)} \sim \mathcal{N}(0, 1/N^{(l-1)})$, $b_i^{(l)} = 0$ [56]
 - hyperbolic tangent: $w_{ij}^{(l)} \sim \mathcal{O}(2)$, $b_i^{(l)} \sim \mathcal{N}(0, 0.104)$ [57]
 - Rectified Linear Unit and Leaky ReLU: $w_{ij}^{(l)} \sim \mathcal{N}(0, 2/N^{(l-1)})$, $b_i^{(l)} = 0$ [49]
 - Swish: $w_{ij}^{(l)} \sim \mathcal{N}(0, 2.952/N^{(l-1)})$, $b_i^{(l)} \sim \mathcal{N}(0, 0.04)$ [50]
 - SELU: $w_{ij}^{(l)} \sim \mathcal{N}(0, 1/N^{(l-1)})$, $b_i^{(l)} = 0$ [54]
 - Mish and SPLASH: $w_{ij}^{(l)} \sim \mathcal{N}(0, 2/N^{(l-1)})$, $b_i^{(l)} = 0$

Here, $\mathcal{N}(\mu, \sigma^2)$ denotes a normal distribution with mean μ and variance σ^2 while $\mathcal{O}(\sigma)$ produces an orthogonal matrix that is built by first generating a random matrix whose elements are drawn from $\mathcal{N}(0, \sigma^2)$ and then performing QR factorization to obtain an orthogonal matrix Q . No investigation of the best initialization schemes could be found in the literature for the Mish and SPLASH activation functions. Therefore, the choice to use the Glorot-initialization as with the ReLU activation is arbitrary.

- The fraction of randomly deactivated neurons (Dropout rate) `dropout`. When using the SeLU activation function, the modified “Alpha-Dropout” [54] suggested by the author is used instead of regular Dropout. Alpha-Dropout sets the value of deactivated neurons to $-\lambda\alpha$ instead of zero to keep the self-normalizing properties of SELU.
- The strength of the L1- and L2-regularization (see Sec. 3.3.3) on the weights matrices $w_{ij}^{(l)}$ and bias vectors $b_i^{(l)}$ `l1_lambda` and `l2_lambda`. For each hidden layer, the total loss is increased by a loss penalty:

$$\Delta L^{(l)} = \lambda_1 \left(\sum_{i,j} |w_{ij}^{(l)}| + \sum_i |b_i^{(l)}| \right) + \lambda_2 \left(\sum_{i,j} \left(w_{ij}^{(l)} \right)^2 + \sum_i \left(b_i^{(l)} \right)^2 \right). \quad (4.5)$$

For all activation functions other than SELU⁴, batch normalization (BN) is applied to each hidden layer. BN transforms a layer’s activations using two consecutive linear transformations before applying the activation function. Details on how the parameters of these transformations are calculated can be found in Ref. [58]. According to the authors, batch normalization reduces the change of the output distributions of each hidden layer over the course of the training, allowing for higher learning rates and thus accelerating the training. In addition to that, the noise that is introduced by calculating the parameters of the first transformation for each individual batch can act as a regularizer.

When reloading the training state from a snapshot, an already existing Keras model is reloaded, and the `build`-function is consequently not needed. However, updating the model’s hyperparameters may be necessary, which is done within the `update` function. When using the built-in functions for MLPs, adjustable hyperparameters of the model are the Dropout rate `dropout` as well as the strength parameters of the L1 and L2 regularization `l1_lambda` and `l2_lambda`.

Once the Keras model is built (or loaded and updated), the `compile`-function is called to create the optimizer and the loss function and bind them to the model. With the built-in `compile`-function, the ADAM optimizer whose update rule was explained in Sec. 3.3.2 is used; its parameters α , β_1 , β_2 and ϵ (see Eq. 3.20) are also tunable hyperparameters. As a loss function, any callable accepted as a loss function by Keras can be used. If desired, multiple loss functions can be provided in the hyperparameter search space, allowing the optimization of the loss function. By default, an extension of the binary cross-entropy with a separate weight w_s for signal events is used:

$$L(f(x), y) = -[w_s y \log(f(x)) + (1 - y) \log(1 - f(x))]. \quad (4.6)$$

The motivation for this weighted loss is explained in Sec. 5.5.3.

⁴With the SELU activation function, the neuron’s outputs should naturally converge to zero mean and unit variance according to Ref. [54]; thus, batch normalization is not needed.

Finally, the training of the neural network is started by calling its `fit`-function, providing the training and validation datasets, and setting the Custom Early Stopper instance as a callback. The termination of the training is triggered by either the Early Stopper or the used trial scheduler.

Custom Early Stopping

As discussed in Sec. 3.3.3, Early Stopping is a popular regularization technique that dynamically decides when to terminate the training of a neural network based on the error on the validation dataset. To achieve this, regular feedback on the performance of the neural networks is required. Fortunately, Keras' *Callbacks API* allows to flexibly define actions that should be executed at various stages of the training and automatically provides the current values of all registered Keras metrics. The Early Stopping functionality is already implemented in the built-in `EarlyStopping` callback. At the end of each epoch, Keras provides a dictionary with the average values of all registered Keras metrics over the past epoch to the Early Stopping callback, which picks the one metric of the user's choice and compares it with the best value that was seen before. The training is terminated when this metric has not improved for `patience` epochs. Usually after the termination, the weights of the neural network are reverted to their values after the best epoch; however, this is not necessary in this case due to the checkpointing that will be discussed later.

To extend Early Stopping to the custom and composite metrics that were discussed in Sec. 4.2.2, a subclass of the `EarlyStopping` class was created, which overwrites the `on_epoch_end` function that is called by Keras at the end of each epoch. When executed, the neural network predictions on the training and validation datasets are computed. Together with the corresponding target labels and event weights, these predictions are used to calculate the values of all custom metrics on the training and validation datasets. Once this is done, the composite metrics can be constructed from the custom and the Keras metrics. After updating the dictionary that was provided by Keras, the `on_epoch_end` function of the superclass is called to perform the Early Stopping on custom or composite metrics.

In addition to the Early Stopping functionality, the computed metrics can now also be used to evaluate the overfitting conditions, allowing to incorporate these conditions directly into the Early Stopping. Logically, an epoch is only considered an improvement when both the metric to monitor improved and all overfitting conditions passed. By adding an additional `overfitting_patience`, independent control over terminating due to lack of improvement and excessive generalization gap become possible. Typically, the `overfitting_patience` can be set to a lower value than the `patience`. While temporary phases without an improvement are possible during the training, the gap between training and validation error is typically monotonically increasing.

Since the calculated values of the metrics are only available in this callback, including the

reporting to Tune in the same function reduces the computational overhead. The report includes the values of all Keras, custom, and composite metrics, as well as their values for the best epoch and the last epoch that passed all overfitting conditions. Due to a limitation in Tune, only the most recently reported metric values for each trial are considered when suggesting new hyperparameters or scheduling trials with PBT. Including both the best and “last valid” metric values in the report, which can both be set as the target metric for the optimization, allows to take finer control over how Tune evaluates the performance of a trial:

- Best value: this prevents a possible performance degradation near the end of the training from reducing the *score* of the trial (and thus the hyperparameters of this trial). However, this can be sensitive to outliers that are not representative for the overall performance of the trial, which may arise due to statistical fluctuations, especially when using a small validation dataset
- Last valid value: this gives Tune potentially more recent and, on average, more representative feedback on the performance of the trial, which may be especially beneficial for the scheduler. However, potential performance degradation near the end of the training will reduce the trial’s score, resulting in a too conservative estimate of the hyperparameter’s potential.

Overall, neither of the two options has a clear advantage over the other, and the best strategy depends on the specific problem. In any case, however, using a raw metric value as the target of the optimization will always be suboptimal when overfitting conditions are defined since the raw metric may still improve while the generalization gap is already considered too large. In such a situation, the potential of the hyperparameters is overestimated, and the overestimation becomes even larger when the generalization gap grows faster.

Finally, the current state of the neural network and its state after the best epoch (including the state of the optimizer) as well as the current state of the Early Stopper are checkpointed, forming the snapshot of the training state that can be given to the `trainable` to restore on. Also, saving the best state of the model allows to either restore on the last or the highest scoring model state when using PBT. Again, which strategy performs best needs to be tested for the specific problem. In both cases, the last state of the Early Stopper is used; otherwise, infinite loops where PBT keeps restoring on the same checkpoint are possible when the perturbation interval is shorter than both the `patience` and `overfitting_patience`.

4.2.5 Experiment evaluation

After the termination of the experiment, the best hyperparameter combination needs to be extracted from the experiment results. While Tune automatically reports the trial with the highest final score, manually evaluating the reported metric values allows for a more flexible

and robust selection strategy. Fortunately, all intermediately reported metric values of all trials are saved, allowing to perform the evaluation on the raw metric values instead of the “best” and “last valid” metrics that were discussed at the end of the previous section.

An overview of the optimization can be generated by concatenating all reported sets of metric values for all trials and sorting them by the time of the report. Since each report corresponds to a training epoch, this can be used to visualize the optimization progress as a function of the total number of epochs. The results are displayed by generating two sets of plots. The first type of plot contains the raw values of each metric on the training and validation dataset for every epoch. It provides a general overview of how the average performance of the neural networks evolved throughout the optimization. Epochs that did not pass the overfitting conditions are included as semi-transparent dots. In the second set of plots, the values of each metric corresponding to the epoch with the best target metric seen up to this point are drawn. For the target metric, this corresponds to a running extremum of the metric value on the validation dataset and the corresponding value on the target dataset and thus shows a monotonically increasing or decreasing value of the validation metric value. This gives an insight into the convergence of the optimization and can be useful when comparing different optimization algorithms (see Sec. 5.5.2). Epochs that failed the overfitting conditions are not considered when finding the best value of the target metric. Example images for both sets of plots are shown in Fig. 4.6

Two approaches are used to select the best trial from the reported metric values. The first variant searches for the epoch with the best target metric value that fulfills all overfitting conditions. Once this trial is found, the corresponding hyperparameters are saved, and plots showing the evolution of each metric throughout the training of this trial are drawn, an example of which can be seen in Fig. 4.7. Similar to reporting the best seen metric value to Tune, which was discussed in the previous section, this can be susceptible to outliers; however, the implications are subtly different. While an outlier reported as the best value to Tune may influence which new hyperparameters are sampled next, the impact on the final result should be relatively small as long as strong outliers are rare compared to the total number of trials. During trial selection, however, a single outlier can potentially cause suboptimal hyperparameters to be returned as a result of the optimization.

Thus, a second approach is implemented that fits a function to the metric values and uses the extreme value of the fit function to select the best trial. Three fit functions are available in the optimization framework:

$$\begin{aligned}
 f_1(x) &= Ae^{-a(x-c)^2} + Be^{-bx} + Cx^2 + Dx + \text{const.} \\
 f_2(x) &= Ax^{-2} + Bx^{-1} + Cx^2 + Dx + \text{const.} \\
 f_3(x) &= Ae^{-a(x-c)^2} + Be^{-bx} + Cx^2 + Dx + Ax^{-2} + Bx^{-1} + \text{const.}
 \end{aligned}
 \tag{4.7}$$

These functions are motivated by the expectation that for reasonable hyperparameters, the

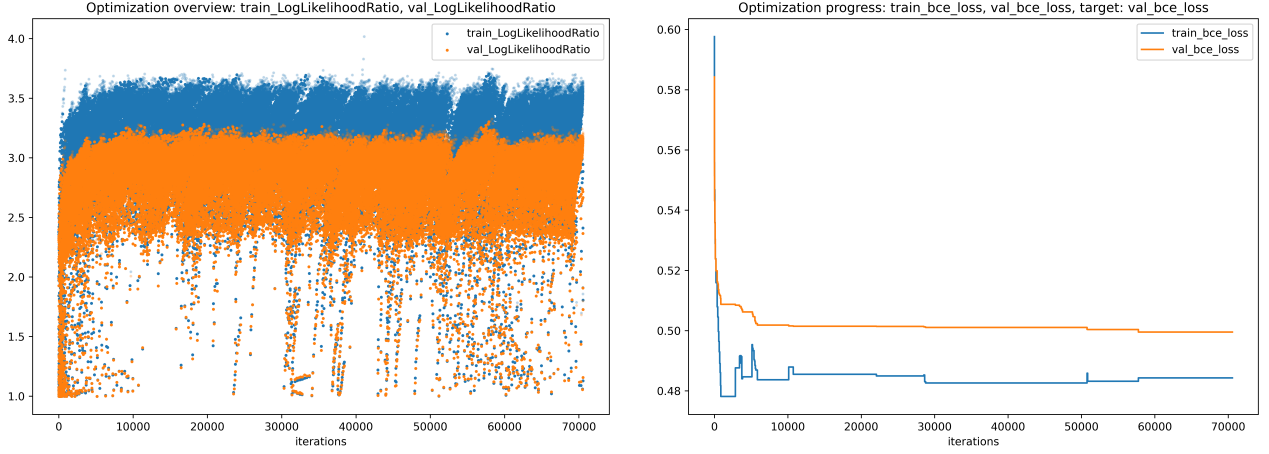


Figure 4.6: Examples of the two types of overview plots that are generated at the end of each optimization. The image on the left contains the values for the metric that is based on the Log-Likelihood ratio (see Sec. 5.5.3) on the training and validation datasets for each epoch, sorted by the timestamp of the report. Epochs that failed the overfitting conditions are drawn as semi-transparent dots. The image on the right shows the running minimum of the validation binary cross-entropy loss, which was used as the target for the optimization, and the corresponding value on the training dataset.

metrics should converge towards an extreme value with increasing training iterations (thus the terms $\propto e^{-x^2}$, $\propto e^{-x}$, $\propto x^{-2}$ and $\propto x^{-1}$) and potentially worsen again due to beginning overfitting near the end of the training (modeled by the terms $\propto x^2$ and $\propto x$). To ensure that the absolute values of the exponential terms are always decreasing, the bounds $0 \leq a < \infty$, $-\infty < c \leq 0$, and $0 \leq b < \infty$ are enforced. A least squares fit is performed using scipy's `curve_fit` function to derive the parameter values. The fit function values at each training iteration are subsequently used to evaluate the overfitting conditions and find the target metric's best value. When the best trial is determined, plots showing the evolution of each metric throughout the training of this trial with the corresponding fit are generated, an example of which is displayed in Fig. 4.7.

An R-squared based goodness of fit condition is implemented to prevent metric values, erroneously predicted due to a bad fit, from causing the selection of suboptimal hyperparameters. Such bad fit can be caused by trials whose metrics do not follow the expected trend and typically happens for very bad hyperparameters. Fits whose R-squared value is below a threshold defined in the configuration file are rejected. Additionally, to prevent overfitting of the fit function, at least twice as many data points as fit parameters are required; otherwise, the fit will not be performed and the corresponding trial is not considered. While both measures were found to reliably reject bad trials while keeping the vast majority of good trials for the MLPs that were used for this thesis, this cannot be generally guaranteed. Thus, the hyperparameters of the best trials selected by both approaches are kept for the evaluation. For all those cases where one of the two hyperparameter combinations must be chosen automatically, e.g., when setting the fixed hyperparameters for the PBT step based on the results from the main optimization,

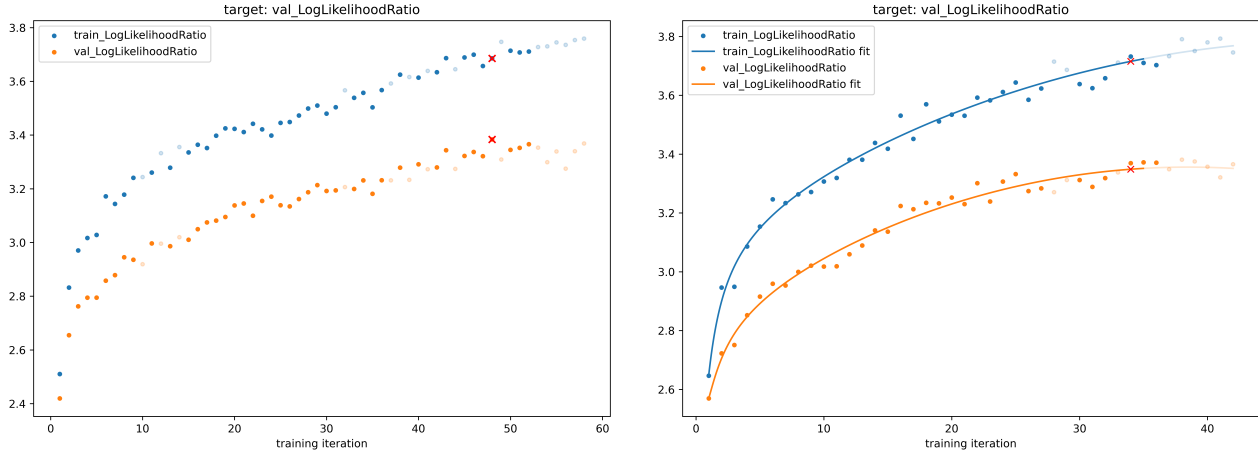


Figure 4.7: Examples for the plots generated during the selection of the best trial, showing the evolution of the target metric throughout the training for said trial. The image on the left corresponds to the approach that chooses the best trial solely based on the best target metric value while excluding epochs that fail the overfitting conditions. The image on the right corresponds to the approach that applies a fit function to the metrics, evaluates the overfitting conditions with the fit predictions of the metric values, and selects the trial based on the best fit function value that passed all overfitting conditions. In both images, the red cross denotes the training iteration selected as the best epoch. Iterations whose raw metric values failed the overfitting conditions are drawn semi-transparent. In the right image, training phases that fail the overfitting conditions based on the fit function values are represented with a semi-transparent line. The function f_2 given in Eq. 4.7 was used for the fit.

options to define the preferred version are available in the configuration file.

It is important to note that for PBT, jumps in the metric values are very likely due to the cloning of the training states from different trials; thus, the fitting approach is likely to fail for PBT.

For both sets of best hyperparameters, k -fold cross-validation is performed to assess the fluctuation of the performance estimate of the neural network. These fluctuations are mainly caused by the stochastic nature of the model training and statistical uncertainties when using small validation datasets. For k -fold cross-validation, the combined training and validation dataset is split into k non-overlapping partitions. Each partition is used once as the validation dataset while the remaining $k - 1$ are used for training, resulting in k fully trained models for each of the two hyperparameter combinations. Similar to the splitting of the full dataset into training, validation and testing sets for the optimization, this k -fold splitting can be done randomly or based on an array of event numbers and a custom splitting condition.

For the first two optimization steps (i.e. the pre- and main optimization steps), the hyperparameters are fixed, and thus the training is performed by calling the `trainable` function and providing the dictionary of hyperparameters. Since the training is independent for all models, it can be asynchronously executed in a distributed fashion using the Ray Task API. This is not possible for the fine-tuning step, as hyperparameters need to get adjusted during

the training. Instead, the `PopulationBasedTrainingReplay` scheduler is used together with Tune to apply the same perturbations to the training as during the optimization. Because of technical limitations within Tune, this requires the models to be trained sequentially.

When the total amount of training data is sufficiently large so that differences between the various splits can be assumed to be small, the same Custom Early Stopping technique used during the optimization should also be applied during the cross-validation. However, when significant differences between the splits can be expected, the overfitting conditions could terminate the training at vastly different states, introducing a significant variation between the different models. In this situation, it can be beneficial to instead use a fixed number of epochs equal to the “best” number of training iterations that were found during the trial selection. This behavior can be selected in the configuration file.

Once all k models for both sets of hyperparameters are trained, each model is given to the `evaluate` function that evaluates the performance of each model. In the built-in version of this function (that is applicable to binary classification tasks), the normalized and weighted output distribution is drawn as a histogram and the values of all custom and composite metrics are calculated. The mean and standard deviations are calculated from the k values for each metric, and the result is saved to a text file.

4.2.6 Input variable selection

When using multilayer perceptrons, including inputs with low discriminative power can result in a larger number of local minima of the error function, thus increasing the likelihood of the training algorithm getting stuck at a suboptimal minimum. As a consequence, the generalization of the trained model can be impaired, resulting in worse performance on new, unseen data. [59] The process of removing variables from the dataset that are not strictly necessary is called *input variable selection*.

The selection algorithm that is included in the optimization framework implements a *backwards elimination* approach to determine which variables are redundant or uninformative. This procedure starts with a model which was trained using the full set of available variables and iteratively removes the most unimportant input until a stopping condition is met. As previously mentioned in Sec. 4.2.1, this input variable selection is based on the results of the pre-optimization step. An option to decide which of the two sets of hyperparameters to use (see Sec. 4.2.5) is available in the configuration file.

Selecting the best input variables using a single model bears the risk of optimizing them only for one specific training-validation splitting, especially when the validation dataset is small. Thus, all k models trained during the k -fold cross-validation are used as the starting point of the backwards elimination. Consequently, the impact of each individual input variable is also evaluated for all k data splittings.

To determine the importance of a variable, a measure of the neural network performance needs

to be specified. As with the hyperparameter optimization, any metric included in Keras or any custom metric can be used for this purpose – see Sec. 4.2.2 for more details. Before starting the variable selection, the value of this metric on the validation dataset of each of the k models is calculated. These values are used as a baseline.

During each iteration of the backwards elimination, the relative importance of every variable must be estimated. Depending on the computational complexity of the problem, one of two methods can be used to achieve this:

retrain-method When the cost of retraining a neural network is low, the importance of a variable can be assessed by removing it from the dataset, retraining the model and evaluating the difference in performance on the validation dataset. This is achieved by repeating the cross-validation described in Sec. 4.2.5 for the chosen set of hyperparameters. When n input variables are remaining in the current iteration, this results in $n \cdot k$ retrained neural networks, each of which had a single input variable removed for the training. For each model, the value of the selected metric is calculated on the corresponding validation dataset, resulting in a list of k metric values for each input variable.

shuffle-method For larger neural networks where retraining is not feasible, the importance of a variable must be gauged by its influence on the prediction of the existing models. Since a variable cannot be completely removed from the inputs of a given model, the values of an input variable are instead randomly shuffled between different events while keeping the order of the remaining variables and the target labels fixed. This preserves the distribution of the input variable but assigns a different value to each event. The predictions of the k cross-validation models using the corresponding shuffled validation datasets are then used to calculate the values of the chosen metric. As with the **retrain-method**, this results in k metric values for each input variable.

A high relative importance of an input variable corresponds to a relatively large performance degradation, and thus a significant worsening of the chosen metric when removing it. Therefore, the relative change of the metric value is used as a measure of variable importance. It is calculated individually for each of the k splittings to obtain k relative changes. In the configuration file, an option is available to decide if the baseline metric values or the best values that were achieved so far during the input variable selection should be used as the reference for the relative changes. When using the **shuffle-method**, removing a variable should never result in a performance improvement⁵; therefore, the baseline metric values should be used as the reference. For the **retrain-method** however, removing an unimportant input variable can certainly improve the model performance. In this case, the choice of which reference to use depends on the specific task:

⁵For this to be possible, shuffling a variable would need to introduce new information that can be used to classify an event, which is not the case.

- (a) Low number of redundant variables and sufficiently large training dataset: the relative changes are dominated by the importance of the input variable that was removed. This means that over the course of the variable selection, a monotonic improvement of the selected metric can be expected until the optimal set of variables is reached. Thus, the best seen metric values should be used as a reference.
- (b) High number of redundant variables or relatively small training dataset: the relative changes for redundant variables are dominated by random fluctuations due to the stochastic nature of the training. Using the best seen metric values could lead to premature termination of the variable selection. Thus, the baseline metric values should be used.

Once the k relative changes are determined for each input variable, the variable with the best mean (or median, depending on the choice in the configuration file) relative change $\bar{\delta}_m$ is selected as a candidate for removal. If $\bar{\delta}_m$ indicates an improvement or the worsening is smaller than a configurable threshold (i.e. $\bar{\delta}_m < \delta_{\max}$ when small metric values are good), the input variable is removed and the next iteration of the backwards elimination begins. Otherwise, the input variable selection is terminated.

5 Extraction of the Longitudinal Polarization of $W^\pm$ -boson pairs

In the previous chapter, a framework for automated and distributed hyperparameter optimization was presented. In the following sections, its application in the study of the polarization in the electroweak production of same-charged $W^\pm$ -boson pairs is discussed. For that, the $W^\pm W^\pm jj$ -EW signal region and the used Monte Carlo samples as well as observables sensitive to the polarization of the two $W^\pm$ -bosons are introduced. In order to increase the sensitivity of the measurement of longitudinally polarized (LL) $W^\pm$ -boson pairs, the hyperparameter optimization framework was used to optimize multilayer perceptrons to separate the LL polarization component from the other polarizations and the background processes. The performance of the resulting neural networks is compared to boosted decision trees trained on the same binary classification task by performing binned maximum likelihood fits to extract the expected significance to measure the LL polarization.

5.1 The $W^\pm W^\pm jj$ -EW signal region

Although only Monte Carlo simulated proton collisions are used for this thesis, these events include a simulation of the final state particles' interaction with the detector. Consequently, all physics objects need to be reconstructed from the detector response. Object selection criteria containing requirements on the kinematic properties of each type of object, e.g. leptons or jets, are applied to filter which physics object to include in the analysis. Based on the properties of these objects, an event selection is used to discard events that are not of interest. Both selections are based on the current ATLAS study of (unpolarized) $W^\pm W^\pm$ scattering at 13 TeV detailed in Ref. [60] and will be summarized in the following two sections.

5.1.1 Object selection

The object selection applied to the objects of interest, namely muons, electrons, jets and missing transverse energy E_T^{miss} , is performed in two steps. A loose selection (preselection) is applied to all physics objects, the selected objects of which undergo an overlap removal to avoid double-counting objects that can be reconstructed by more than one reconstruction

algorithm. A tighter (**signal**) selection is applied to all physics objects remaining after the overlap removal.

Leptons

In the object selection for muons and electrons, criteria to access the quality of the identification (ID) are included. For muons, these are based on recommendations by the Muon Combined Performance group. For electrons, a likelihood discriminant is used which considers, amongst others, the quality of the track, the shape of the shower and the matching between the track and the calorimeter cluster. Since electrons and photons both cause a shower in the electromagnetic calorimeter while only the electrons create a track in the inner detector, the quality of the track influences the ambiguity of the decision electron/photon. Thus, for **signal** electrons, an additional criterion based on the “author” quantity, which has a value of 1 for unambiguous classification as an electron, is included. Additionally, a selection based on the Electron Charge ID Selector Tool (ECIDS) is included for **signal** electrons to suppress backgrounds due to electron charge mis-identification.

In addition to the criteria on the quality of the object identification, kinematic (transverse momentum p_T) as well as geometric (pseudo-rapidity η , longitudinal and transverse impact parameters z_0 and d_0) acceptance criteria and requirements on the lepton isolation are included. All lepton selection criteria are collected in Tab. 5.1.

Jets

Jets are reconstructed using the particle flow algorithm [61], combining information from the calorimeter and the inner tracker, and clustered using the anti- k_T algorithm [62, 63] with a resolution parameter of $R = 0.4$. The calibrated jets (see Ref. [60], Sec. 4.5 for more details on the calibration) are required to have a transverse momentum $p_T > 20$ GeV and pseudo-rapidity $|\eta| < 4.5$ for the **preselection**, which is further restricted to $p_T > 25$ GeV for the **signal** selection. In order to suppress pile-up jets, the Jet Vertex Tagger JVT with the **tight** working point [64] was applied to jets with $p_T < 60$ GeV. All surviving jets are b -tagged using the DL1r tagger with the 85 % efficiency working point.

Missing transverse energy

The missing transverse energy E_T^{miss} is calculated using all calibrated preselected objects. Particle flow jets are directly included in the hadronic hard term. Tracks that are associated with the primary vertex but are not matched to any reconstructed object are included in a track-based soft term.

Table 5.1: Muon and electron selection criteria based on the current ATLAS study of $W^\pm W^\pm$ scattering at 13 TeV. [60]

	Muons		Electrons	
	preselection	signal selection	preselection	signal selection
ID	Loose	Medium	LooseLH	TightLH
p_T	> 3 GeV	> 27 GeV	> 4.5 GeV	> 27 GeV
$ \eta $	< 2.7	< 2.5	< 2.47	< 2.47 excl. $1.37 \leq \eta \leq 1.52$
$ z_0 \times \sin \theta $	< 1.5 mm	< 0.5 mm	< 0.5 mm	< 0.5 mm
$\left \frac{d_0}{\sigma_{d_0}} \right $	< 15	< 3	< 5	< 5
Isolation	None	FixedCutPflowTight	None	Gradient
Author	–	–	–	== 1
ECIDS	–	–	–	pass

Overlap removal

Lepton-lepton Two electrons are overlapping, when the corresponding reconstructed objects have overlapping electromagnetic showers or a common track in the inner detector; the electron with the higher p_T is kept. For an overlapping electron-muon pair sharing a track in the inner detector, the muon is kept unless it was not tagged in the muon spectrometer.

Electron-jet Jets overlapping with an electron candidate with $\Delta R_{e,j} < 0.2$ are removed unless they are b -tagged. For larger distances $0.2 < \Delta R_{e,j} < \min(0.4, 0.04 + 10 \text{ GeV}/p_T^e)$, the electron is removed.

Muon-jet A jet overlapping with a muon candidate with $\Delta R_{e,j} < 0.2$ is removed if it has less than three associated tracks with $p_T > 0.5$ GeV or if both $p_T^\mu/p_T^j > 0.5$ and the ratio of p_T^μ to the p_T sum of tracks associated to the jet whose $p_T > 0.5$ GeV is larger than 0.7. Again, b -tagged jets are excluded from removal. For all remaining jets, the muon is removed if $\Delta R_{e,j} < \min(0.4, 0.04 + 10 \text{ GeV}/p_T^\mu)$.

5.1.2 Event selection

In order to select events that reflect the fully leptonic final state and the expected experimental signature of the $W^\pm W^\pm jj$ -EW process described in Sec. 2.2.1, the event selection criteria collected in Tab. 5.2 are applied to each event, defining the *signal region*.

Exactly two **signal** leptons with the same electric charge, at least two **signal** jets and missing transverse momentum of at least 30 GeV (due to the two neutrinos) are required in each event in order to obtain the desired final state.

Drell-Yan backgrounds, e.g. $Z \rightarrow e^+e^-$ where one of the electrons undergoes a charge-flip, are suppressed by applying a Z-veto selection on the dilepton invariant mass $|m_{\ell\ell} - m_Z| > 15$ GeV

and requiring both leptons to lie in the barrel region ($|\eta_e| < 1.37$) in the ee -channel. Additionally, a minimum dilepton invariant mass of 20 GeV is required to reduce the modeling uncertainty of low-mass Drell-Yan processes.

In order to reduce contributions from background processes with more than two leptons in the final state, e.g. the fully leptonic final states of $W^\pm Z$ and ZZ , events with more than two **preselection** leptons surviving the overlap removal are vetoed. Additionally, events are also removed when their third lepton does not pass the overlap removal but forms a dilepton system with invariant mass close to the Z-boson mass $|m_{\ell\ell} - m_Z| < 15$ GeV with one of the two **signal** leptons of the same flavor and opposite electric charge.

Contributions from top quark processes are reduced by employing a b -jet veto, rejecting events where at least one **signal** jet is b -tagged.

The two **signal** jets with the highest transverse momentum are required to fulfill $p_T > 65$ GeV and $p_T > 35$ GeV, respectively. They are selected as the tagging jets and used to calculate the corresponding observables m_{jj} and ΔY_{jj} .

In order to further purify the signal region and especially enhance the $W^\pm W^\pm jj$ -EW relative to the $W^\pm W^\pm jj$ -QCD process, the experimental signature of the signal process discussed in Sec. 2.2.1 is utilized by requiring a large invariant mass and rapidity separation of the two tagging jets, $m_{jj} > 500$ GeV and $\Delta Y_{jj} > 2$, respectively.

5.1.3 Monte Carlo samples and expected yields

Within this chapter, the irreducible background processes discussed in Sec. 2.2.1 as well as all reducible backgrounds are modeled using Monte Carlo simulation. The full list of MC samples used can be found in Tab. B.1.

The different polarizations of the $W^\pm$ -bosons in the $W^\pm W^\pm jj$ -EW process are simulated using MadGraph 2.9.5 [65] (AthGeneration-21.6.79) interfaced to Pythia8 [66] for the modeling of the parton shower, hadronization and underlying event. Technical details on the simulation of polarized cross sections using MadGraph can be found in Ref. [9]. Three samples are available corresponding to the $W_L^\pm W_L^\pm$, $W_L^\pm W_T^\pm$ and $W_T^\pm W_T^\pm$ polarization components, collectively referred to as “polarized samples”. For all three samples, the polarization was defined in the center-of-mass frame of the two $W^\pm$ -bosons. In order to access the contribution of interferences between different polarizations which were discussed in Sec. 2.2.2, an additional (“unpolarized”) sample is available, simulated via the same decay-chain syntax but without polarization information. Finally, to gauge the influence of the approximation needed for the factorization of the process into production and decay of the two $W^\pm$ -bosons (see Ref. [9]), which in turn is necessary to define the polarization of the bosons (see Sec. 2.2.2), the nominal $W^\pm W^\pm jj$ -EW sample of the current ATLAS study of $W^\pm W^\pm$ scattering at 13 TeV [60] is used. This sample was produced using MadGraph5_aMC@NLO 2.6.7 [67] interfaced to Herwig 7.2 [68] and models the full $W^\pm W^\pm jj$ -EW process, i.e. includes the contributions of the non-resonant diagrams

Table 5.2: Event selection defining the signal region of the $W^\pm W^\pm jj$ -EW process based on the current ATLAS study of $W^\pm W^\pm$ scattering at 13 TeV [60].

Exactly two signal leptons with the same electric charge
Veto events with more than two preselected leptons
ee channel: $ \eta < 1.37$
$m_{\ell\ell'} \geq 20$ GeV
ee channel: $ m_{ee} - m_Z > 15$ GeV
$E_T^{\text{miss}} \geq 30$ GeV
At least two jets
Leading jet: $p_T > 65$ GeV
Subleading jet: $p_T > 35$ GeV
b -jet veto
$ \Delta Y_{jj} > 2$
$m_{jj} \geq 500$ GeV

(see Sec. 2.2.1) that are not included in the unpolarized sample. This sample will be referred to as the “full” sample within this thesis.

Contributions of prompt background processes are modeled with the same Monte Carlo samples used in the current ATLAS study of $W^\pm W^\pm$ scattering [60]. The $ZZ + \text{jets}$, $t\bar{t}V$, tV , single top-quark and triple boson processes are summarized in the “other prompt” category for the rest of this chapter. While data-driven methods were used to estimate the $V + \gamma$, charge mis-identification and non-prompt backgrounds in Ref. [60], purely Monte Carlo based modeling was used for this thesis due to the need to train the multivariate classifiers on the simulated events.¹ The charge mis-identification background is modeled using MC samples for $Z \rightarrow e^+e^-$ decays. The photon conversion background is estimated using MC samples of $W + \gamma$ and $Z + \gamma$ production, which are combined into the “Vgamma” category. The contribution of non-prompt backgrounds is modeled with MC samples of leptonic $W + \text{jets}$ and semi-leptonic and hadronic $t\bar{t}$ production.

In Tab. 5.3, the expected event yields obtained using the aforementioned Monte Carlo samples are compared to the corresponding yields provided in Ref. [60]. Overall, a reasonably good agreement is achieved, with the non-prompt background contribution expectedly showing the largest deviation.

¹When performing the training directly using data-driven estimates for the non-prompt backgrounds, derived using the fake factor method (see Ref. [8]), a significant number of events would carry a negative event weight due to the subtraction of prompt events. This in itself could cause problems during the training. In addition to that, since the loss calculated for each event is multiplied with the event weight, the loss for data events and MC subtraction events would have a different sign. As a result, the classifier would be motivated to exploit possible mismodeling to separate MC and data events of the same process.

Table 5.3: Estimated event yields and statistical uncertainties of the $W^\pm W^\pm jj$ -EW (“full”) and all relevant background processes as well as the event yields corresponding to the individual polarization components. In addition to that, the sum of the three polarization contributions (“polsum”) and the expected yield using the unpolarized sample (“unpol”) are shown. The yields for the (full) $W^\pm W^\pm jj$ -EW and background processes are compared to the corresponding estimations given in the support note of the current ATLAS study of $W^\pm W^\pm$ scattering at 13 TeV [60], denoted as “VBS note”.

Process	Event yield	
	this study	VBS note
$W^\pm W^\pm jj$ -QCD	23.65 ± 0.14	24.04 ± 0.16
$W^\pm W^\pm jj$ -Int	7.37 ± 0.07	7.60 ± 0.08
$W^\pm Z$	115.1 ± 1.2	97.7 ± 1.1
Other prompt	2.83 ± 0.29	7.08 ± 0.26
$V + \gamma$	11.3 ± 1.6	10.9 ± 1.3
Charge-flip	5.0 ± 1.3	10.11 ± 0.31
Non-prompt	26 ± 4	56 ± 5
$W^\pm W^\pm jj$ -EW (full)	227.8 ± 0.5	234.90
$W^\pm W^\pm jj$ -EW (unpol)	221.3 ± 1.1	
$W^\pm W^\pm jj$ -EW (polsum)	222.9 ± 0.8	
$W_L^\pm W_L^\pm jj$ -EW	20.00 ± 0.14	
$W_L^\pm W_T^\pm jj$ -EW	65.0 ± 0.4	
$W_T^\pm W_T^\pm jj$ -EW	137.9 ± 0.7	
Total MC	414 ± 4	448 ± 6

5.2 Fit strategy and polarization sensitive observables

Due to their very short lifetime, vector bosons, and consequently their polarization, cannot be measured directly in the detector. Instead, their existence has to be inferred from their decay products. As a result, the polarization of the $W^\pm$ -bosons in the $W^\pm W^\pm jj$ -EW process can only be investigated using its influence on the kinematics of the final state particles. The first step towards measuring the longitudinal-longitudinal (LL) polarization component is therefore to find observables that are sensitive to the polarization of the $W^\pm$ -bosons.

The selection of polarization sensitive observables does not have to be stringent, as, for both investigated multivariate techniques, the set of input variables is automatically optimized. Thus, instead of defining an objective selection criterion, all available observables got included that were visually found to show differences between the polarization components. The resulting set of observables is shown in Tab. 5.4.

For each observable, a binned maximum likelihood fit to Asimov-data was performed to find the most sensitive observable. For these fits, the contribution of the $W^\pm W^\pm jj$ -EW process was assumed to be sufficiently described by the sum of the three polarization components; this assumption is verified in Sec. 5.3. The LL polarization component was chosen as the signal while all other polarizations and the background processes were set as the background. In order to incorporate the fact that none of the individual polarizations have been observed so far, the sum of the polarization components is included as an unconstrained nuisance parameter. No systematic uncertainties are included for the background processes.

Before each fit, a binning optimization is performed. Starting with 100 bins, neighboring bins are merged until each bin contains at least 10 background events. The exact bin edges are chosen to maximize the significance estimate suggested in Ref. [69]. Once this is completed, bins are further merged until the estimated significance decreased by 0.01.

Tab. 5.5 displays the expected significances extracted for each observables using the described fit strategy. Unsurprisingly, due to the low expected yield of the LL polarization component, even the most sensitive observable – the early-projected massless transverse invariant mass of the dilepton-MET system m_{o1} – results in a very low significance of only 0.69 standard deviations. For the two most sensitive observables, the normalized distributions of all three polarizations, the stacked distribution showing the contribution of each process, and the binning-optimized stacked distribution used for the Asimov fit are shown in Fig. 5.1. The normalized and stacked distributions for the remaining observables can be found in Appendix D.

Table 5.4: List of polarization sensitive observables and corresponding descriptions.

Observable	Description
η^{j1}	pseudo-rapidity of the leading jet
η^{j2}	pseudo-rapidity of the subleading jet
Y^{j1}	rapidity of the leading jet
Y^{j2}	rapidity of the subleading jet
p_T^{j1}	transverse momentum of the leading jet
p_T^{j2}	transverse momentum of the subleading jet
$\eta^{\ell1}$	pseudo-rapidity of the leading lepton
$\eta^{\ell2}$	pseudo-rapidity of the subleading lepton
$p_T^{\ell1}$	transverse momentum of the leading lepton
$p_T^{\ell2}$	transverse momentum of the subleading lepton
ΔY_{jj}	rapidity separation of the tagging jets
$\Delta \Phi_{jj}$	azimuthal distance of the tagging jets
ΔR_{jj}	angular distance of the tagging jets
$\Delta \eta_{\ell\ell}$	pseudo-rapidity separation of the two leptons
$\Delta \Phi_{\ell\ell}$	azimuthal distance of the two leptons
$\Delta R_{\ell\ell}$	angular distance of the two leptons
$m_{\ell\ell}$	invariant mass of the dilepton system
$p_{T,\ell\ell}$	p_T of the dilepton system
$a_{T,\ell\ell}$	p_T of the dilepton system relative to its thrust axis
$ \cos(\vartheta_{CS}^{\ell\ell}) $	angle between the two leptons in the Collins-Soper reference frame
$z^{\ell1}$	Zeppenfeld variable of the leading lepton
$z^{\ell2}$	Zeppenfeld variable of the subleading lepton
$\min(\Delta R_{j\ell})$	minimal angular distance between the leptons and the tagging jets
R_{Pt}	p_T ratio of the leptons and the tagging jets
E_T^{miss}	missing transverse energy
m_T^{WW}	transverse mass of the dilepton system and E_T^{miss}
m_{o1}	early-projected massless transverse inv. mass of the dilepton system and E_T^{miss}

Table 5.5: Significance of the $W_L^\pm W_L^\pm jj$ -EW contribution obtained by performing a binned maximum likelihood fit to Asimov data on different observables. Only statistical uncertainties are included in the fit.

Observable	Significance [σ]
m_{o1}	0.691
$p_T^{\ell_1}$	0.664
m_T^{WW}	0.621
$ \cos(\vartheta_{CS}^{\ell\ell}) $	0.430
$m_{\ell\ell}$	0.417
E_T^{miss}	0.403
$p_T^{\ell_2}$	0.400
$\Delta\Phi_{\ell\ell}$	0.355
$p_{T,\ell\ell}$	0.331
R_{Pt}	0.322
$\Delta\Phi_{jj}$	0.303
$\min(\Delta R_{j\ell})$	0.297
$p_T^{j_1}$	0.252
$\Delta\eta_{\ell\ell}$	0.234
ΔR_{jj}	0.226
$\Delta R_{\ell\ell}$	0.173
$a_{T,\ell\ell}$	0.156
ΔY_{jj}	0.147
η^{j_1}	0.144
Y^{j_1}	0.143
$p_T^{j_2}$	0.096
η^{ℓ_2}	0.081
η^{ℓ_1}	0.064
Y^{j_2}	0.050
z^{ℓ_1}	0.030
η^{j_2}	0.024
z^{ℓ_2}	0.016

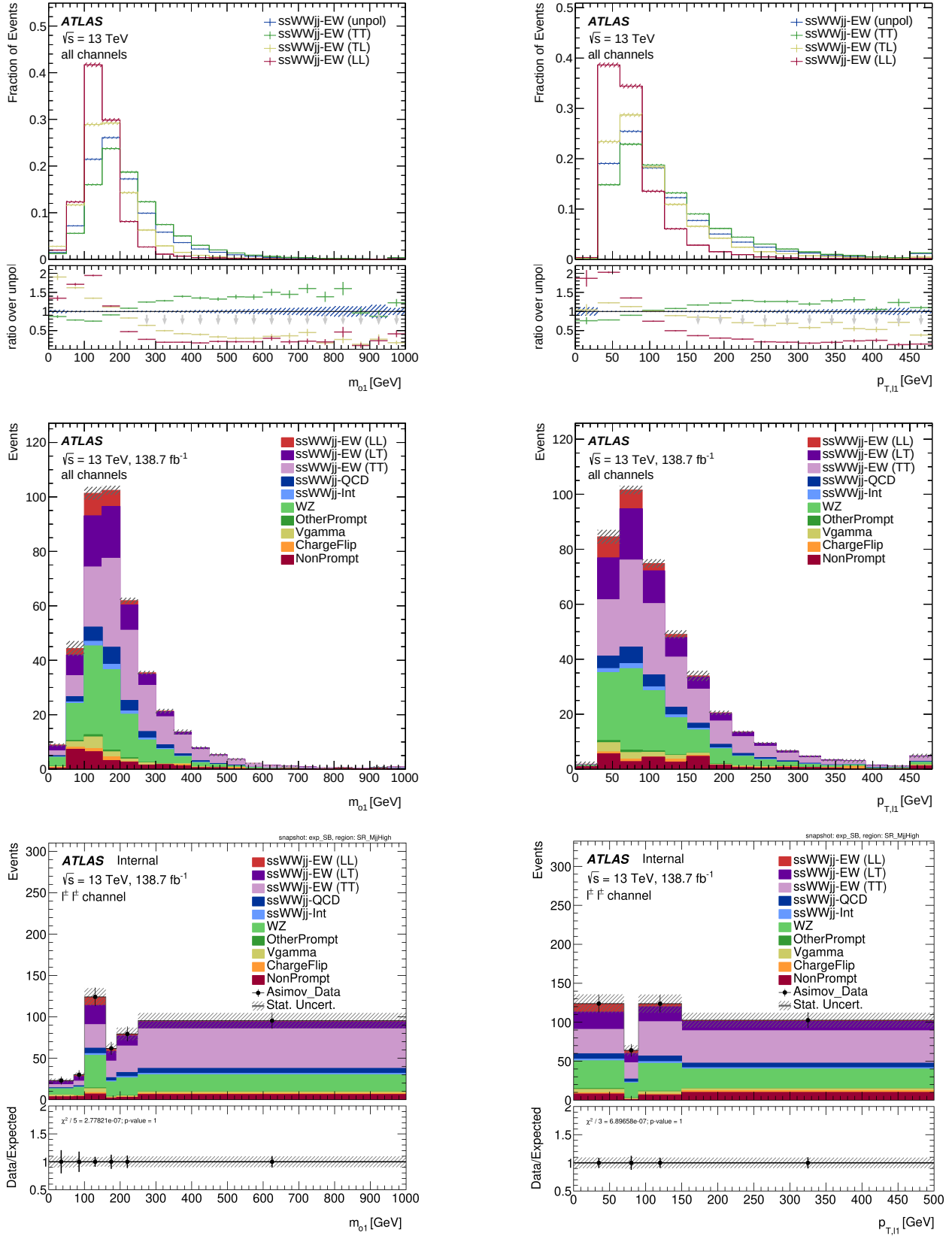


Figure 5.1: Distributions of the early-projected massless transverse invariant mass of the dilepton-MET system m_{o1} (left) and the transverse momentum of the leading lepton $p_{T,l1}$ (right) of all contributing processes in the $W^{\pm}W^{\pm}jj$ -EW signal region. The drawn error bars correspond to statistical uncertainties. For each observable, the normalized distributions of the three polarization components and the unpolarized sample (top), the stacked distributions of all contributing processes (center) and the binning optimized stacked distributions used for the Asimov fit (bottom) are shown.

5.3 Validation of the $W^\pm W^\pm jj$ -EW polarization samples

In order to describe the contribution of the $W^\pm W^\pm jj$ -EW process as the sum of the contributions of three different polarization states of the two intermediate $W^\pm$ -bosons, longitudinal-longitudinal (LL), longitudinal-transverse (LT) and transverse-transverse (TT), two approximations are necessary. Firstly, as explained in Sec. 2.2.2, the polarization of the two $W^\pm$ -bosons can only be defined for double-resonant diagrams. As mentioned in Sec. 5.1.3, MadGraph's decay-chain syntax is used to exclude single- and non-resonant diagrams and to limit the virtuality of the intermediate $W^\pm$ -bosons for the polarized and unpolarized $W^\pm W^\pm jj$ -EW samples; more details on the decay-chain syntax can be found in Sec. 2.4 of Ref. [70] and Sec. 4.2 of Ref. [9]. Secondly, interferences between different polarizations are not included when only considering the contributions of the three polarization components LL, LT and TT. Within this section, the impact of both approximations is investigated to ensure that the sum of the three polarization contributions provides a sufficiently good description of the full $W^\pm W^\pm jj$ -EW process.

Comparing the total expected event yields using the full, the unpolarized and the sum of the polarized samples provides an initial insight into the influence of the two approximations. They are shown in Tab. 5.6 together with the corresponding ratios. The comparison of the unpolarized to the full sample shows an overall contribution of single- and non-resonant diagrams of around 2.8 %. The contribution from interferences between different polarizations is significantly smaller, showing only a difference of about 0.7 % between the unpolarized and the sum of the polarized samples, which is on the order of the Monte Carlo statistical uncertainties. The unpol / full comparison indicates that the majority of the differences between the full and the sum of the polarized samples can likely be attributed to single- and non-resonant diagrams. A similar result was also found on truth level in Ref. [7] where the $W^\pm W^\pm jj$ -EW process was investigated using the SHERPA event generator.

In order to assess the influence of the approximations on differential distributions, the full, unpolarized and sum of the polarized samples were compared for all polarization sensitive observables. For lepton observables, the deviations between the three samples were mostly

Table 5.6: Comparison of the expected event yields of the $W^\pm W^\pm jj$ -EW process in the signal region using the full, unpolarized and sum of the polarized samples.

Process	Event yield
$W^\pm W^\pm jj$ -EW (full)	227.8 $\pm$ 0.5
$W^\pm W^\pm jj$ -EW (unpol)	221.3 $\pm$ 1.1
$W^\pm W^\pm jj$ -EW (polsum)	222.9 $\pm$ 0.8
unpol / full	0.972 ± 0.005
polsum / unpol	1.007 ± 0.006
polsum / full	0.978 ± 0.004

constant and on the order of a few percent for most bins. Observables containing information on the tagging jets (both jet and jet-lepton observables), however, show a significant discrepancy between the full and the other samples of up to 15-20% per bin. Examples for both types of observables are shown in Fig. 5.2, the full set of observables can be found in Appendix C.

Since the full sample was produced with a different version of MadGraph and potentially different settings than the other $W^\pm W^\pm jj$ -EW samples, it is difficult to assess the origin of the observed discrepancy with certainty. In order to better investigate these differences, it may be beneficial to produce an additional Monte Carlo sample of the full $W^\pm W^\pm jj$ -EW process with settings comparable to the unpolarized and polarized samples. Additionally, it will probably be necessary to include the observed differences as a systematic uncertainty on the Monte Carlo prediction in the polarization study. Since both the inclusion of systematic uncertainties and the production of Monte Carlo samples are beyond the scope of this thesis, this discrepancy will not be investigated further.

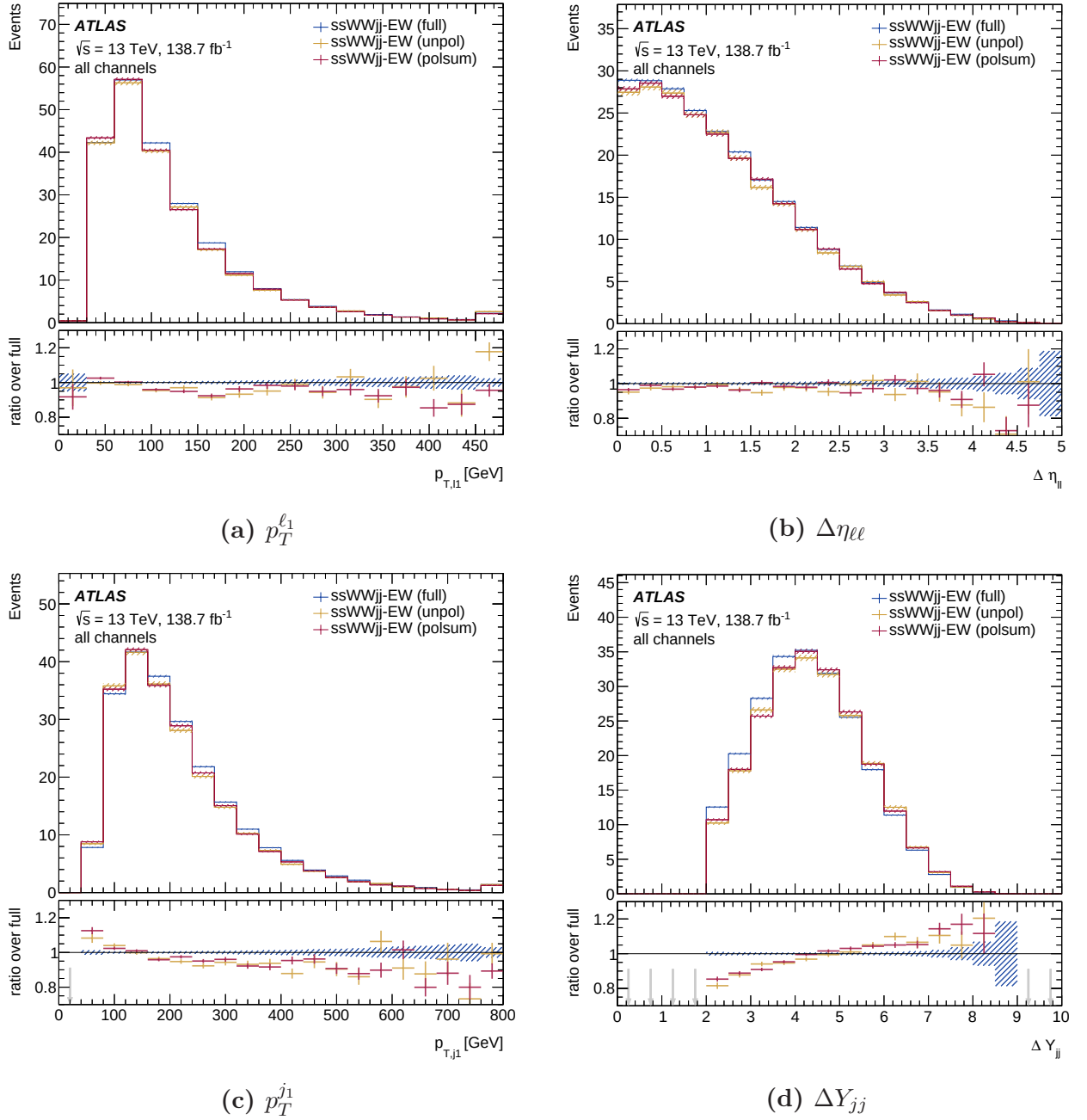


Figure 5.2: Comparison of the full (blue), the unpolarized (yellow) and sum of the polarized samples (red) of the $W^\pm W^\pm jj$ -EW process using two exemplary lepton (top) and jet observables (bottom) in the $W^\pm W^\pm jj$ -EW signal region. The full sample was produced using MadGraph5_aMC@NLO 2.6.7 interfaced to Herwig 7.2 while the unpolarized and polarized samples were generated with MadGraph 2.9.5 (AthGeneration-21.6.79) interfaced to Pythia8. Only statistical uncertainties are shown.

5.4 Study of the LL polarization using boosted decision trees

As detailed in Sec. 5.2, even the observable that was found to be the most sensitive to the polarization of the intermediate $W^\pm$ -bosons resulted in a discovery significance of the LL polarization of less than 0.7 standard deviations. In order to greatly improve the sensitivity of the measurement, multiple observables can be combined into a new variable, making use of correlations that exist between different observables. The distribution of this variable can then be used to extract the LL polarization component. This approach is called *multivariate analysis*.

Within this section, the use of gradient-boosted decision trees (BDTs) from the TMVA library for this purpose is investigated. The results of this study are intended as a baseline for the application of multilayer perceptrons which will be discussed in detail in Sec. 5.5.

The BDTs are trained to classify individual events as signal-like (LL) or background-like (TL, TT and other background processes). A detailed explanation of the working principles of BDTs can be found Sec. 3.3.1. The optimization and evaluation of the BDTs loosely follows the approach presented in Ref. [18].

5.4.1 Figures of merit

In order to quantify the performance of the BDTs, figures of merit (FoMs) are built from the BDT's output distribution. For binary classification, the goal is to achieve a large separation between the two classes, i.e. a large difference between the output distributions for signal and background events. Thus, FoMs calculated from the signal and background distributions of the BDT on the testing dataset can be used as a measure of the BDTs generalization error. In addition to that, FoMs that compare the output distributions of the same class on training and testing data allow to quantify the generalization.

The list of FoMs that are used to evaluate the BDTs is given below. All FoMs are calculated from two binned output distributions, either the signal and background distributions on the same dataset or the output distribution of the same class on the training and testing datasets. Here, S_i and B_i denote the signal and background events in bin i , respectively, while $n_{1,i}$ and $n_{2,i}$ are used for the bin values of the normalized distributions.

Reduced χ^2 $f_{\text{red},\chi^2}(\mathbf{n}_1, \mathbf{n}_2)$. The reduced χ^2 -values is used as a measure for the compatibility between two normalized distributions:

$$f_{\text{red},\chi^2}(\mathbf{n}_1, \mathbf{n}_2) = \frac{1}{N_{\text{dof}}} \sum_{i=1}^{N_{\text{bins}}} \frac{(n_{1,i} - n_{2,i})^2}{\Delta n_{1,i}^2 + \Delta n_{2,i}^2}. \quad (5.1)$$

N_{dof} denotes the degrees of freedom, $\Delta n_{k,i}$ corresponds to the absolute uncertainty in bin i of distribution k . Reduced χ^2 -values around 1 are expected for agreeing distributions while larger values indicate disagreement.

Kolmogorov-Smirnov $f_{\text{KS}}(\mathbf{n}_1, \mathbf{n}_2)$. Like the red. χ^2 -value, the *Kolmogorov-Smirnov* (KS) test can be used to judge the dissimilarity between two normalized distributions:

$$f_{\text{KS}}(n_1, n_2) = \max_k \left| \sum_{i=1}^k n_{1,i} - n_{2,i} \right|. \quad (5.2)$$

Since f_{KS} corresponds to the largest difference between the integral of the two distributions, the KS-test is less susceptible to random fluctuations than the χ^2 test and therefore better suited to judge shape differences between the distributions indicating a generalization gap.

Non-Overlap $f_{\text{N-O}}(\mathbf{n}_1, \mathbf{n}_2)$. The overlap f_{O} of two normalized distributions, given as the common area under both distributions, can be used as a measure of similarity. For convenience, the non-overlap defined as:

$$f_{\text{N-O}}(n_1, n_2) = 1 - f_{\text{O}} = 1 - \sum_{i=1}^{N_{\text{bins}}} \min(n_{1,i}, n_{2,i}), \quad (5.3)$$

is used here. Non-overlap values near 1 indicate a large disagreement while $f_{\text{N-O}} = 0$ stands for identical distributions.

Separation $f_{\text{Sep}}(\mathbf{n}_1, \mathbf{n}_2)$. The separation is defined as:

$$f_{\text{Sep}}(n_1, n_2) = \sum_{i=1}^{N_{\text{bins}}} \frac{(n_{1,i} - n_{2,i})^2}{n_{1,i} + n_{2,i}}. \quad (5.4)$$

Like the non-overlap, the value of $f_{\text{Sep}}(n_1, n_2)$ is closer to 1 for dissimilar and 0 for identical distributions.

Approximated AUC $f_{\text{AUC}}(s, b)$. In the context of particle physics, the receiver operating characteristic (ROC) curve of a classifier is most often defined as the signal efficiency as a function of the background rejection. The area under the ROC curve (AUC) is one of the most widely used metrics to quantify the performance of a classifier. For a perfect classifier (and correspondingly non-overlapping output distributions), the AUC has a value of 1 while random guessing (or identical distributions) correspond to a value of 0.5. Since only the binned output distributions of the BDTs were available, the determined AUC value is only an approximation of the true ROC integral.

Estimated discovery significance $Z_{\text{SoB}}(S, B)$. In order to account for the normalization of the signal and background output distributions, an estimated discovery significance is used, defined as:

$$Z_{\text{SoB}} = \max_i \sum_{j \geq i}^{N_{\text{bins}}} \frac{S_j}{\sqrt{B_j + (\Delta_{\text{syst}} \cdot B_j)^2}}. \quad (5.5)$$

This estimate takes both the statistical and systematic uncertainties of the number of background events per bin into account. Of all the available FoMs, this quantity should be closest to the significance that will later be extracted from the maximum likelihood fit.

Estimated total significance $Z_{\text{tot}}(S, B)$. This metric is similar to the estimated discovery significance but includes the statistical uncertainty of the number of signal events per bin. It is defined as:

$$Z_{\text{tot}} = \max_i \sum_{j \geq i}^{N_{\text{bins}}} \frac{S_j}{\sqrt{S_j + B_j + (\Delta_{\text{syst}} \cdot B_j)^2}}. \quad (5.6)$$

5.4.2 Preprocessing of input variables

For the training of the BDTs, all polarization sensitive observables (see Sec. 5.2) were included. In addition to that, it was decided to include the azimuth angles of the tagging jets, the two leptons and the missing transverse energy. The reason for this was mainly to keep consistency with the study using multilayer perceptrons, where all low-level observables were included for the training (see Sec. 5.5).

The azimuth angle of a particle on its own cannot carry any information due to the rotational symmetry around the beam axis, only angular differences between two particles can. However statistical fluctuations that may be present in the distributions of the azimuth angles could still lead to overfitting. In order to circumvent this potential issue, the rotational symmetry was incorporated into the definition of the azimuth angles by redefining the zero point of the azimuth axis for each event based on the azimuth angle of the leading lepton:

$$\begin{aligned} \phi^{\ell_1} &\rightarrow \phi_{\text{recal}}^{\ell_1} = 0, \\ \phi^{\ell_2} &\rightarrow \phi_{\text{recal}}^{\ell_2} = \phi^{\ell_2} - \phi^{\ell_1}, \\ \phi^{j_1} &\rightarrow \phi_{\text{recal}}^{j_1} = \phi^{j_1} - \phi^{\ell_1}, \\ \phi^{j_2} &\rightarrow \phi_{\text{recal}}^{j_2} = \phi^{j_2} - \phi^{\ell_1}, \\ \phi^{\text{MET}} &\rightarrow \phi_{\text{recal}}^{\text{MET}} = \phi^{\text{MET}} - \phi^{\ell_1}. \end{aligned} \quad (5.7)$$

Using these definitions, the azimuth angles are transformed into angular differences between multiple particles. Since now ϕ_{recal} is trivial and $\phi_{\text{recal}}^{\ell_2} \equiv \Delta\Phi_{\ell\ell}$, ϕ_{recal} and $\Delta\Phi_{\ell\ell}$ are removed from the set of input variables.

As mentioned in 3.3.1, during the training of each BDT, a fixed number of equidistant cuts are tested for each input variable. The outer limits of the range of tested cut values are set to the minimum and maximum values of the distribution. For observables with strongly skewed distributions like transverse momenta, this results in oversampling of the tail of the distribution where only a small fraction of events are located while the region around the distribution's peak may be undersampled. In order to mitigate this effect, observables with long tails are scaled by applying a logarithmic or square root transformation. Additionally, the effect of outlier events can be prevented by manually reducing the range of the input variables. Similar to over- and underflow bins in histograms, the values of events outside of the specified range was set to the minimum or maximum allowed value. Finally, the effective distance between different cut values can be reduced for symmetric observables (e.g. pseudo-rapidities) by only taking the absolute value of the observable. While this potentially disregards information on correlations with other observables, it was found to lead to slightly better performance of the resulting BDT. As an example, the signal and background distribution of the transverse momentum of the leading lepton before and after the preprocessing are shown in Fig. 5.3. The scaled distributions of the remaining input variables can be seen in Appendix E.

A summary of the preprocessing applied to all used input variables as well as the values of some figures of merit calculated using the signal and background distribution of each observable is displayed in Tab. 5.7.

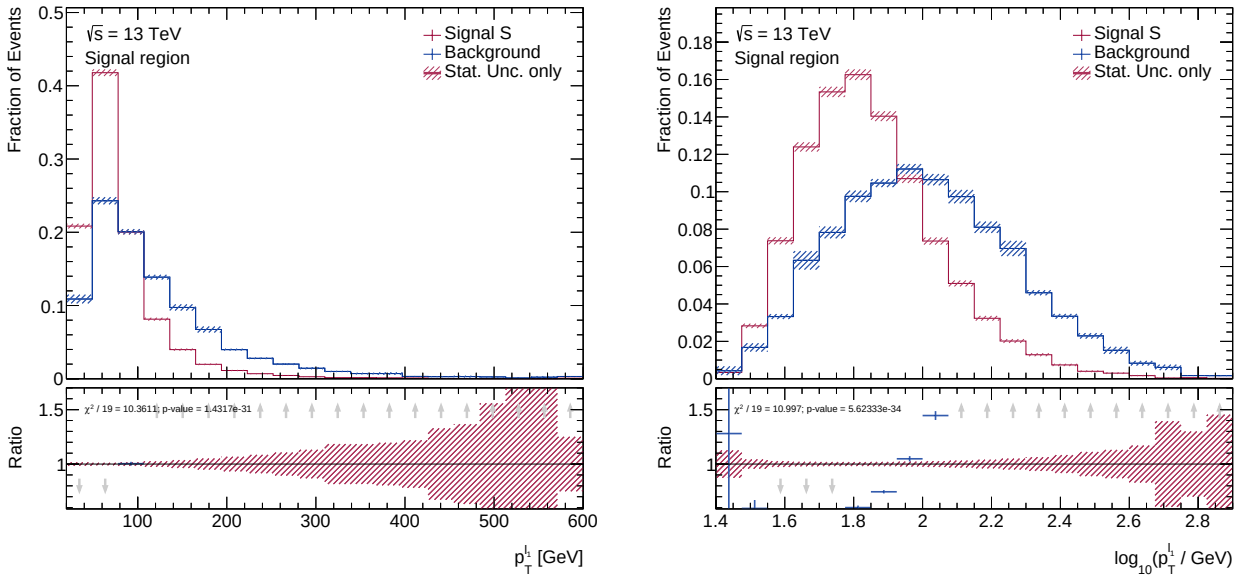


Figure 5.3: Normalized signal and background distribution of $p_T^{\ell_1}$ before (left) and after (right) preprocessing in the $W^\pm W^\pm jj$ -EW signal region. The contribution of the LL polarization component of the $W^\pm W^\pm jj$ -EW process is used for the signal distribution while the background distribution is given by the sum of the remaining polarizations and background processes. Only statistical uncertainties are shown.

Table 5.7: The values of the non-overlap $f_{\text{N-O}}$, separation f_{Sep} , reduced χ^2 $f_{\text{red. } \chi^2}$, Kolmogorov-Smirnov test f_{KS} and the approximated area under the ROC curve f_{AUC} for the signal and background distributions of all observables used for the BDT training, ranked by decreasing non-overlap. Additionally, the preprocessing applied to each observable is shown.

Observable	Figures of merit					Preprocessing		
	$f_{\text{N-O}}$	f_{Sep}	$f_{\text{red. } \chi^2}$	f_{KS}	f_{AUC}	Function	min	max
$p_T^{\ell_1}$	0.29	0.110	151.71	0.29	0.64	log		2.9
m_T^{WW}	0.28	0.108	177.62	0.28	0.64	log		3.2
m_{o1}	0.27	0.099	167.36	0.26	0.57	sqrt	3	30
$\min(\Delta R_{j\ell})$	0.21	0.060	86.64	0.21	0.61			4
ΔR_{jj}	0.20	0.069	116.70	0.20	0.60			9
$m_{\ell\ell}$	0.17	0.047	72.97	0.17	0.56	log		3.2
$ \cos(\vartheta_{CS}^{\ell\ell}) $	0.17	0.037	50.19	0.17	0.58			
$\Delta\Phi_{jj}$	0.17	0.041	54.67	0.17	0.57			
$p_{T,\ell\ell}$	0.17	0.048	63.79	0.17	0.54	sqrt		25
ΔY_{jj}	0.16	0.042	58.55	0.16	0.58			8.5
$p_T^{\ell_2}$	0.16	0.042	69.99	0.16	0.55	log		2.5
R_{Pt}	0.16	0.037	48.69	0.16	0.56	log		1
η^{j1}	0.15	0.030	48.72	0.15	0.57	abs		
Y^{j1}	0.15	0.030	47.72	0.15	0.57	abs		
z^{ℓ_1}	0.13	0.034	42.22	0.13	0.54	sqrt		1.2
E_T^{miss}	0.13	0.035	111.61	0.13	0.53	log		2.8
z^{ℓ_2}	0.13	0.031	49.22	0.13	0.55	sqrt		1.2
$\Delta\eta_{\ell\ell}$	0.13	0.021	29.04	0.12	0.54	sqrt		
$\phi_{\text{recal}}^{\ell_2}$	0.12	0.018	27.05	0.06	0.48			
p_T^{j1}	0.11	0.024	48.16	0.11	0.53	log		3
ϕ_{recal}^{j2}	0.08	0.008	13.74	0.05	0.48			
$\phi_{\text{recal}}^{\text{MET}}$	0.07	0.007	8.94	0.04	0.48			
$a_{T,\ell\ell}$	0.07	0.013	24.54	0.05	0.47	sqrt		18
η^{j2}	0.06	0.007	10.83	0.06	0.51	abs		
Y^{j2}	0.06	0.007	10.74	0.06	0.51	abs		
$\Delta R_{\ell\ell}$	0.06	0.006	8.20	0.05	0.48			5.4
η^{ℓ_1}	0.05	0.005	6.00	0.05	0.50	abs		
p_T^{j2}	0.05	0.005	17.20	0.03	0.47	log		2.8
η^{ℓ_2}	0.04	0.002	3.17	0.04	0.50	abs		
ϕ_{recal}^{j1}	0.04	0.002	3.18	0.01	0.46			

5.4.3 Variable & hyperparameter optimization

As with neural networks, the hyperparameters of the BDT and the set of used input variables need to be optimized to achieve the best performance. For this optimization, the full dataset consisting of 21799 signal and 222067 background events is split into a training and a testing sample based on the event number:

- training: `EventNumber mod 5  $\neq$  0`
- testing: `EventNumber mod 5 = 0`

The performance of a BDT is evaluated based on the estimated discovery significance Z_{SoB} calculated on the testing dataset. In the current ATLAS study of $W^\pm W^\pm$ scattering at 13 TeV, a systematic uncertainty on the total expected yield of around 6% is assumed [60]. It was therefore decided to include a slightly more conservative 10% systematic uncertainty on the background yield in Z_{SoB} :

$$Z_{\text{SoB}} = \max_i \sum_{j \geq i}^{N_{\text{bins}}} \frac{S_j}{\sqrt{B_j + (0.1B_j)^2}}. \quad (5.8)$$

The input variable and hyperparameter optimization was performed in three consecutive steps. In the first step, an initial set of input variables was built using a fixed starting set of hyperparameters that can be found in Tab. 5.8, an explanation of the hyperparameters can be found in Sec. 3.3.1. Using this initial input variable set, the BDT's hyperparameters are optimized during the second step. Once the best hyperparameters are determined, the set of input variables is fine-tuned during the third step.

Originally it was intended to reuse the training dataset during the maximum likelihood fit. For this, the generalization gap needs to be as small as possible in order to avoid over-estimation of the BDTs performance. The influence of reusing the training dataset for the fit will be discussed in detail in Sec. 5.5.4. In addition to that, the hyperparameter optimization consistently produced a few BDTs that, by chance, performed very well on the chosen testing dataset while performing poorly when retrained on a different training/testing split. Fortunately, these BDTs

Table 5.8: Set of hyperparameters used during the first step of the BDT optimization.

Hyperparameter	Value
<code>NTrees</code>	400
<code>MaxDepth</code>	3
<code>MinNodeSize</code>	2%
<code>nCuts</code>	20
<code>Shrinkage</code>	0.02
<code>BaggedSampleFraction</code>	0.6

always showed a large generalization gap. In order to both limit the generalization gap and reject the problematic BDTs, a set of overfitting conditions got defined:

$$\begin{aligned}
 f_{\text{KS, sig.}} &< 0.02, \\
 f_{\text{KS, bkg.}} &< 0.05, \\
 \frac{Z_{\text{tot, train.}} - Z_{\text{tot, test.}}}{Z_{\text{tot, train.}}} &< 0.05.
 \end{aligned} \tag{5.9}$$

Using a Kolmogorov-Smirnov (KS) test, shape differences between the output distributions on the training and testing dataset are limited. The different limits for signal and background distributions were found to be necessary due to the larger fluctuations of the background distributions, which are mainly driven by low Monte Carlo statistics of the non-prompt background. In addition to the KS-based conditions, requiring a small difference between the estimated total significance on training and testing data was found to further improve the rejection. During all three optimization steps, a BDT that fails any of these conditions is disregarded.

The initial set of input variables is built iteratively, starting with a single observable. As the starting variable, the transverse momentum of the leading lepton $p_T^{\ell_1}$ was chosen because it resulted in the largest non-overlap of the preprocessed signal and background distributions (as shown in Tab. 5.7). During each iteration, that input variable was added, that resulted in the largest increase of the estimated discovery significance while passing all overfitting conditions. If multiple input variable were already in use, it was also checked if the removal of one of the previously added variables led to a larger improvement. Once neither adding nor dropping a variable resulted in an improvement while passing the overfitting conditions, the hyperparameter optimization was started.

During initial testing it was found that both the number of cuts `nCuts` and the bagging fraction `BaggedSampleFraction` had a neglectable impact on the BDT's performance, as long as their values were reasonable. Therefore, it was decided to not optimize these two parameters. For the remaining hyperparameters, an exhaustive grid search over the search space defined in Tab. 5.9 was performed. In total, 3528 hyperparameter combinations were tried. The combination that resulted in the highest estimated discovery significance while passing all overfitting conditions was selected.

Table 5.9: Search space of the BDT hyperparameter optimization

Hyperparameter	Considered values
<code>NTrees</code>	400, 500, ..., 1500
<code>MaxDepth</code>	3, 4, 5
<code>MinNodeSize</code>	1%, 1.5%, ..., 4%
<code>Shrinkage</code>	0.0025, 0.005, 0.0075, 0.01, 0.015, 0.02, 0.03, ..., 0.1

Using the optimized hyperparameters, the fine-tuning of the input variable set was performed. Starting from the list of input variables derived during the first step, the same iterative procedure was followed, with one exception: in addition to adding a single input variable at a time, variables could also get added in pairs. The same termination condition as during the first optimization step was applied.

5.4.4 Results

In Tab. 5.10, the optimized hyperparameters are collected. In order to get a better insight into how the optimization arrived at these parameters, the surrounding of the best hyperparameter combination was investigated. For simplicity, this was done by fixing all but one hyperparameter to the optimal value and varying the remaining parameter within its value range given by the hyperparameter search space. In Fig. 5.4 and 5.5, the trends of the estimated discovery significance and the values of the three quantities that are constrained by the overfitting conditions are shown. It can be seen that the values of all hyperparameters are constrained by the requirements on the value of the KS test on the signal distributions and on the relative differences of the estimated total significances on the training and testing datasets (Z -ratio). While the value of the **Shrinkage** is already close to the local maximum, increasing the number of trees and decreasing the minimum node size both result in a significant increase of the estimated discovery significance in exchange for a larger generalization gap. Loosening the overfitting conditions could therefore result in a BDT with better generalization performance. However, as already mentioned in the previous section, this would also increase the risk of selecting a BDT that only performs well on the specific testing dataset used during the optimization while generalizing poorly to new data.

In Tab. 5.11, the input variables that were selected during the two input variable optimizations are shown. In total, 16 input variables were chosen. The evolution of the values of the estimated discovery significance, the KS tests on the signal and background distributions and the Z -ratio over the course of the optimization are shown in Fig. 5.6. It can be seen that while the value of the KS test on the signal distributions and the Z -ratio steadily increase when adding additional input variables, the value of the KS test for the background distribution initially decreases. This indicates that, for a given set of BDT hyperparameters, generalization to unseen background

Table 5.10: Set of hyperparameters that result from the performed hyperparameter optimization.

Hyperparameter	Value
NTrees	400
MaxDepth	5
MinNodeSize	4%
Shrinkage	0.04

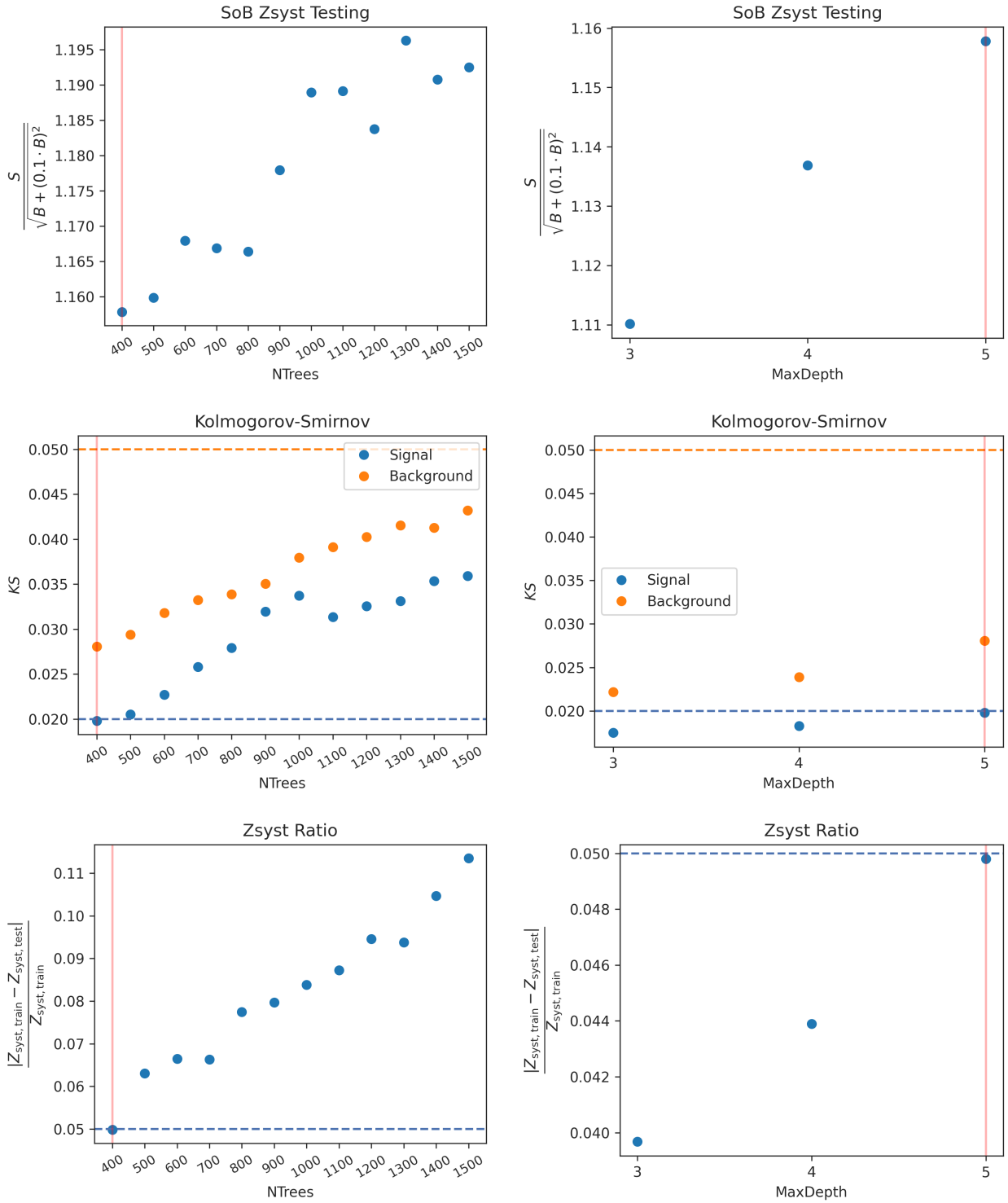


Figure 5.4: Change of the estimated discovery significance (top), the values of the KS tests on the signal and background distributions (center) and the relative differences of the estimated total significances on the training and testing datasets Z-ratio (bottom) when varying **NTrees** (left) and **MaxDepth** around their optimized values. The limits on the values of the KS test and the Z-ratio are shown as dashed horizontal lines, the parameter value that was selected during the hyperparameter optimization is marked using a red vertical line.

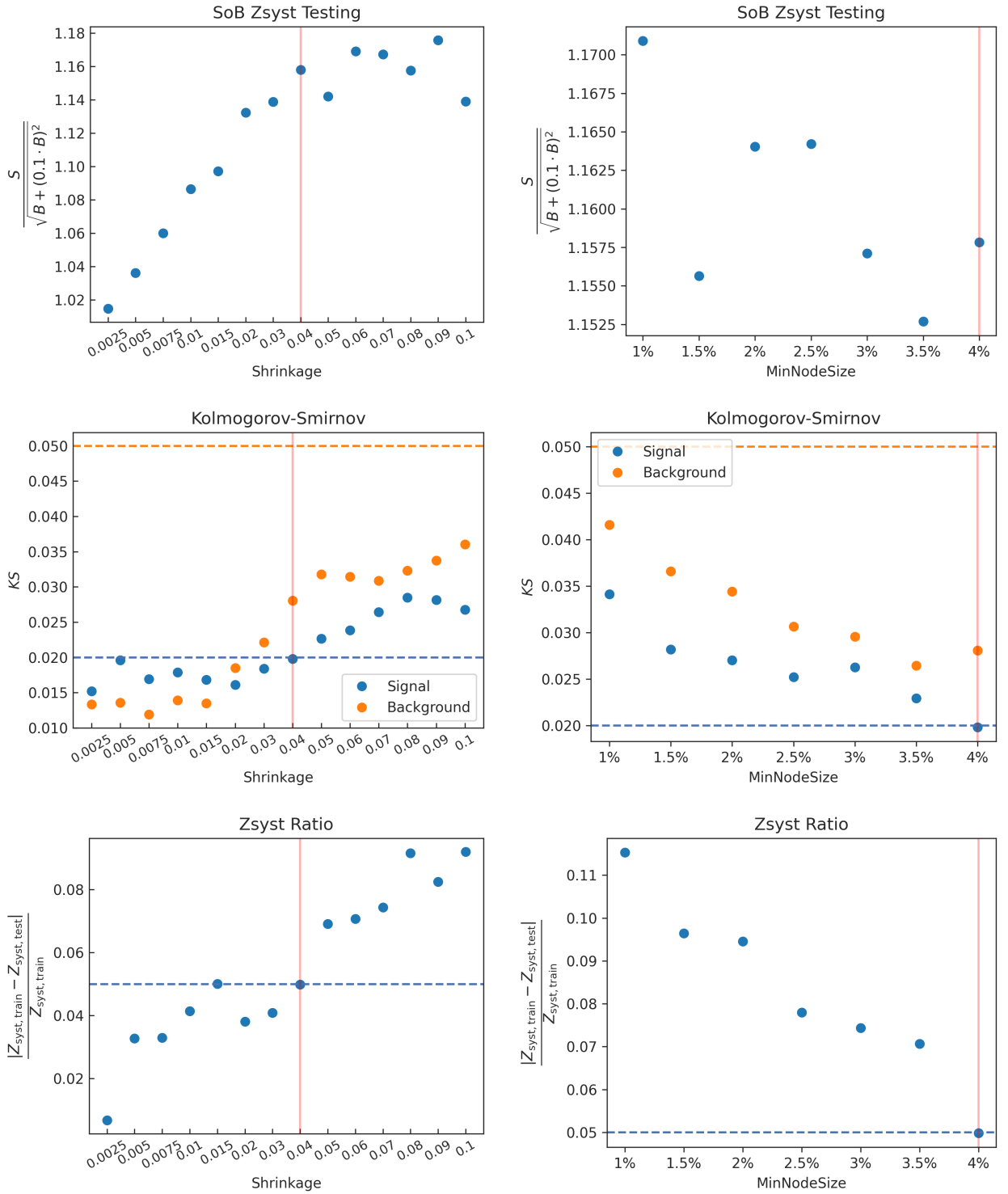


Figure 5.5: Change of the estimated discovery significance (top), the values of the KS tests on the signal and background distributions (center) and the relative differences of the estimated total significances on the training and testing datasets Z-ratio (bottom) when varying **Shrinkage** (left) and **MinNodeSize** around their optimized values. The limits on the values of the KS test and the Z-ratio are shown as dashed horizontal lines, the parameter value that was selected during the hyperparameter optimization is marked using a red vertical line.

events improves when adding additional input variables, while the same was not observed for signal events. A possible reason for this difference could be the presence of high-weight background events due to low Monte Carlo statistics but consistently small weights for signal events.

Table 5.11: List of input variables in the sequence they were added during the input variable optimization. Input variables above the dashed line were added in the first variable optimization step, variables below in the second.

Iteration	Observable
1	$p_T^{\ell_1}$
2	ΔR_{jj}
3	$p_T^{j_2}$
4	E_T^{miss}
5	$\Delta\eta_{\ell\ell}$
6	m_T^{WW}
7	$m_{\ell\ell}$
8	$a_{T,\ell\ell}$
9	$\Delta\Phi_{jj}$
10	$\phi_{\text{recal}}^{j_1}$
11	$ \cos(\vartheta_{CS}^{\ell\ell}) $
12	$\Delta\eta_{\ell\ell}$
13	m_{o1}
14	$\phi_{\text{recal}}^{\text{MET}}$
15	$\Delta\Phi_{\ell\ell}$ and $p_{T,\ell\ell}$

With the optimized set of input variables and hyperparameters, a 5-fold cross-validation was performed, resulting in five BDTs whose corresponding testing sets make up the entire dataset. The event splitting was performed according to:

- training: $(\text{EventNumber} + C) \bmod 5 \neq 0$,
- testing: $(\text{EventNumber} + C) \bmod 5 = 0$,

with $C = 0, 1, \dots, 4$. For the BDT corresponding to $C = 0$, i.e. the same splitting used during the optimization, the normalized signal and background distributions on the training and testing datasets are shown in Fig. 5.7a. Overall, a good agreement between the distributions on the training and testing dataset is visible, with only minor shape differences indicating a small generalization gap. In Fig. 5.7b and 5.7c, the normalized output distributions for the three polarization components and the stacked output distribution for all contributing processes are shown. As expected, the BDT output shows much larger shape differences between the different polarization components than any individual observable, proving the effectiveness of multivariate analysis. However, it is also apparent that the separation between the polarizations, especially LL and LT, is still not optimal. The non-polarization backgrounds, on the other

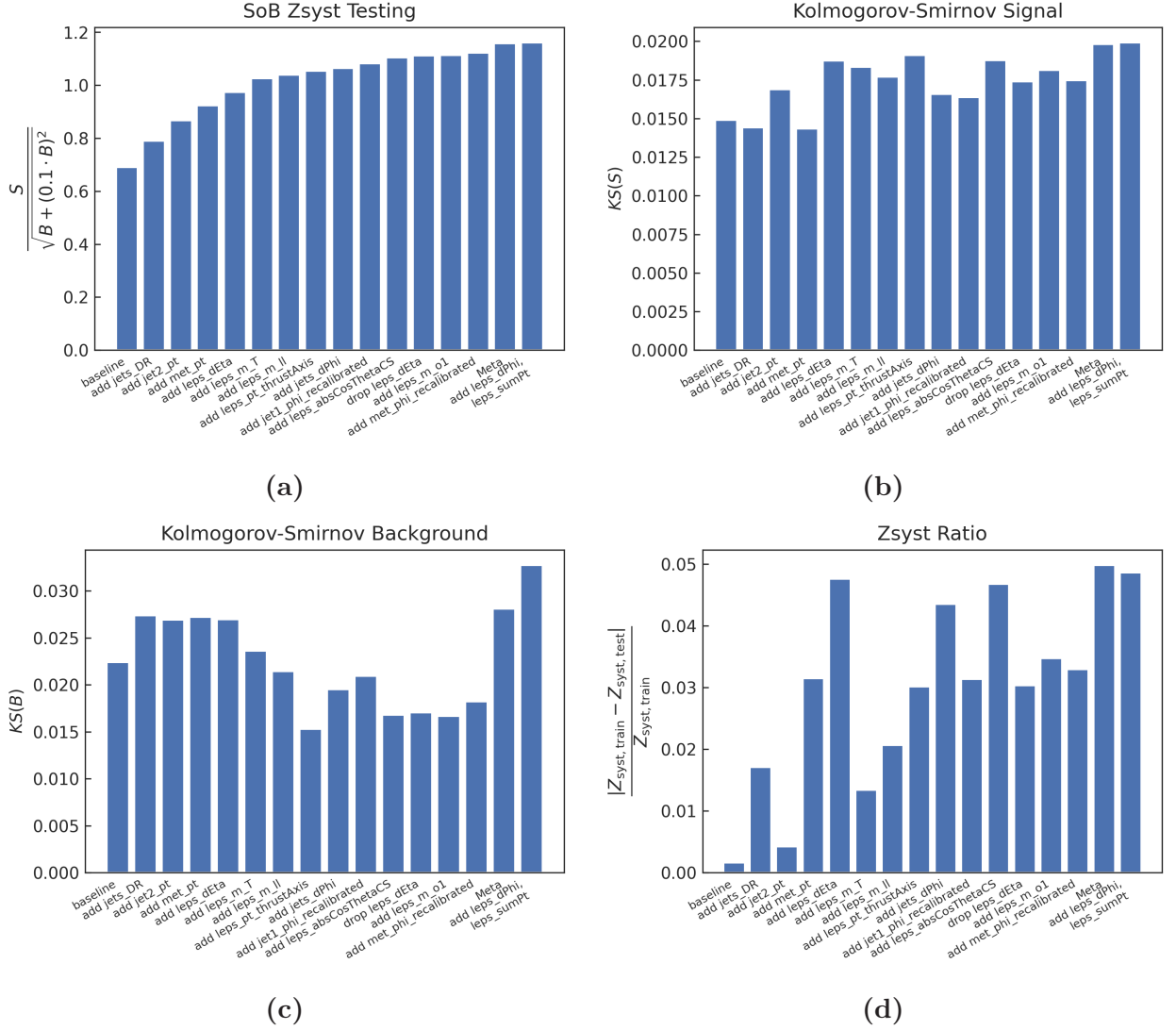


Figure 5.6: Evolution of the (a) estimated discovery significance, value of the KS tests on (b) the signal and (c) the background distributions and (d) the relative differences of the estimated total significances on the training and testing datasets over the course of the two input variable optimizations. The value of the metrics after the hyperparamter optimization is included and labeled as “Meta”.

hand, are classified much better, showing a clear peak in the stacked output distribution at low BDT scores.

In order to assess the generalization performance of the BDTs, the stacked output distributions of all five BDTs on their respective testing datasets were calculated. Since each testing dataset corresponds to one fifth of the full dataset, the distributions were scaled by a factor of 5 to achieve the same integrated luminosity of the full dataset. For each output distribution, a maximum likelihood fit was performed, following the fit strategy detailed in Sec. 5.2. The binning optimized stacked output distribution of the BDT corresponding to $C = 0$ event splitting is shown in Fig. 5.7d. The expected significances resulting from the five maximum likelihood fits are collected in Tab. 5.12. Averaging the five significances yields a total expected

significance of 1.259 ± 0.039 standard deviations, which corresponds to an 80% higher significance compared to the most sensitive single observable m_{o1} (see Sec. 5.2). It is worth noting that the majority of the fluctuations of the sensitivities between the different event splittings can be attributed to low Monte Carlo statistics, resulting in systematic differences between the individual splittings. This will be discussed in more detail in Sec. 5.5.4.

Table 5.12: Expected discovery significances of the $W_L^\pm W_L^\pm jj$ -EW polarization component for the outputs of the five final BDTs, evaluated on their respective testing datasets. Additionally, a total expected significance with uncertainties is given, calculated as the mean and the standard deviation of the five individual significances, respectively.

Event splitting C	exp. significance $[\sigma]$
0	1.256
1	1.215
2	1.278
3	1.316
4	1.230
Total	1.259 ± 0.036

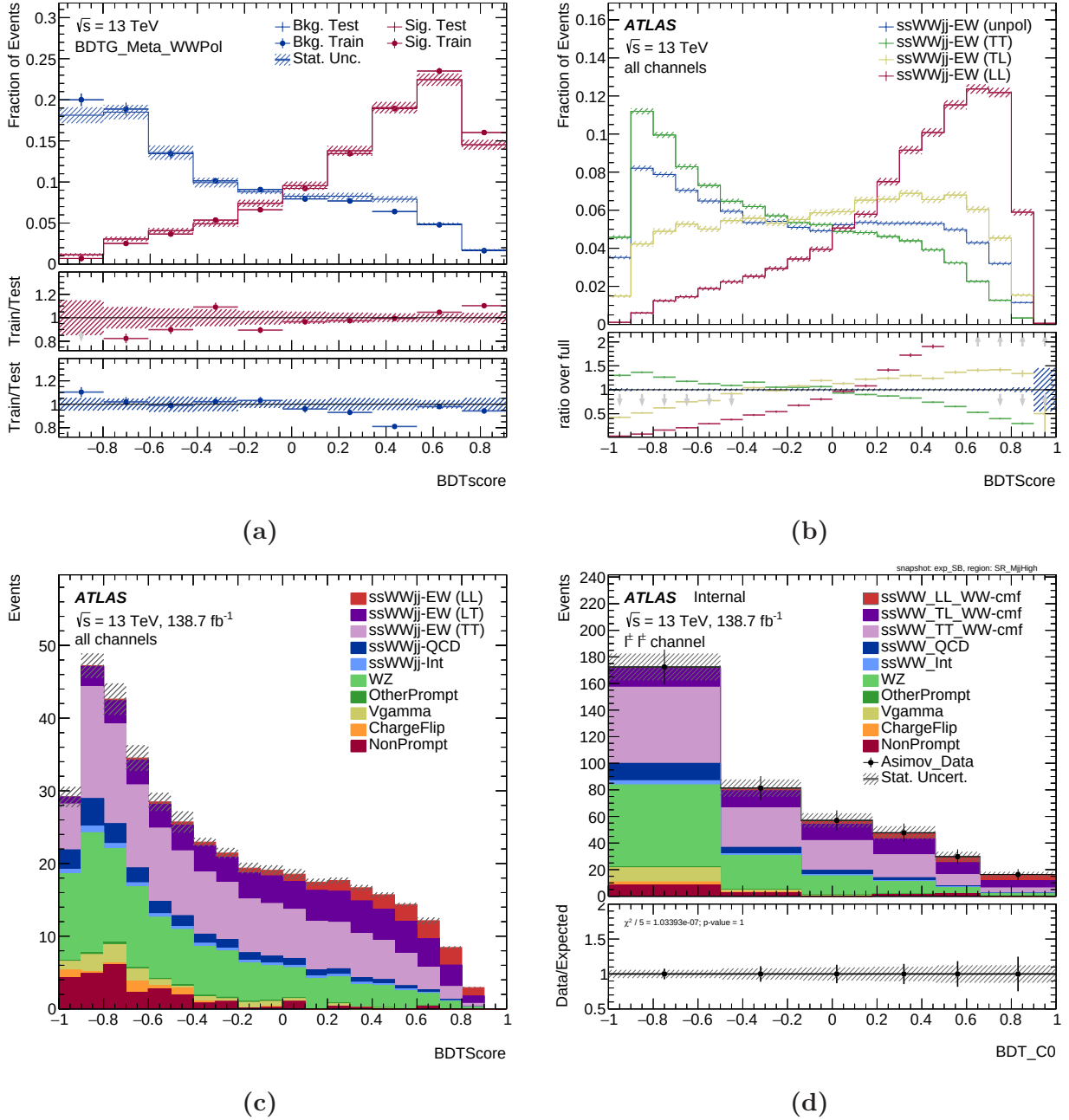


Figure 5.7: (a) Normalized signal and background distributions on the training and testing dataset, (b) the normalized output distributions for the three polarization components of the $W^\pm W^\pm jj$ -EW process and (c) the stacked output distribution for all contributing processes using the full dataset and (d) the binning optimized stacked output distribution using the testing dataset for the final BDT corresponding to the $C = 0$ event splitting in the $W^\pm W^\pm jj$ -EW signal region. For figure (a), the contribution of the LL polarization component of the $W^\pm W^\pm jj$ -EW process is used for the signal distribution while the background distribution is given by the sum of the remaining polarizations and background processes. Only statistical uncertainties are shown in all plots.

5.5 Study of the LL polarization using multilayer perceptrons

In order to further improve the sensitivity of the measurement, an artificial neural network should be used to perform the signal discrimination in place of the boosted decision tree. In order to achieve that, the hyperparameter optimization framework presented in Sec. 4.2 was applied to optimize the hyperparameters and the set of used input variables of a multilayer perceptron (MLP). A detailed introduction to MLPs has been given in Sec. 3.3.2. Since the optimization framework was built with this specific task in mind, all built-in functionality could be utilized. The configuration files for all optimizations presented in this section are available at https://gitlab.cern.ch/atlas-germany-dresden-vbs-group/CAF-Scripts/-/tree/KerasOnRootfile_dev_erik/scripts/KerasOnRootfile/configs/thesis; the relevant parts of the configuration can also be found in Appendix F.

5.5.1 Input variables, data preprocessing and general settings

In a previous work by Baldi et al. [71] it was shown that for sufficiently deep MLPs and large enough training datasets, the inclusion of low-level variables is crucial to reach the highest possible classification accuracy of particle collision events. While it was expected that with the available dataset, low-level variables alone are not sufficient to reach optimal separation between the signal and background processes, in contrast to the findings in Ref. [71], they were still expected to provide additional information usable by the MLPs. Thus, in addition to the polarization sensitive observables that were discussed in Sec. 5.2, it was decided to include all remaining low-level observables. As mentioned before in Sec. 5.4.2, only the azimuth angles of the leptons, the tagging jets and the missing transverse energy needed to be added. The same redefinition of the zero point of the azimuth axis (see Eq. 5.7) as for the BDTs is applied here. The procedure introduced in Sec. 4.2.3 was used to preprocess the dataset, applying log and square root transformations to the same input variables as for the BDT training (see Tab. 5.7). No absolute values or cut-offs were used. Due to the similarities to the distributions of the BDT input variables shown in Appendix E – the applied linear transformation does not change their shapes, and the omission of the absolute value only mirrors the distribution on the y-axis – the scaled input distributions of the MLP input variables are not shown.

For all optimizations presented in this section, the dataset was split into a training and a validation set according to:

- training: $(\text{EventNumber} + 1) \bmod 5 \neq 0$
- validation: $(\text{EventNumber} + 1) \bmod 5 = 0$

This corresponds to the $C = 1$ event splitting used for the training of the final BDTs. The motivation to use this splitting instead of the $C = 0$ splitting of the BDT optimization is the presence of systematic differences that were observed between different splittings. This finding

was briefly mentioned in the previous section and will be discussed in more detail in Sec. 5.5.4. The $C = 0$ splitting was observed to regularly result in better performance of the MLP on the validation than on the training dataset, while the $C = 1$ splitting was found to be closest to the “average” event splitting. Because of these systematically different splittings, it was also decided to not include a testing dataset since differing performance on validation and testing data would not be indicative of over-specialization to the validation data. Instead, a slightly larger training dataset got used to improve the Monte Carlo statistic of the non-prompt background contribution. In further studies, the effect of excluding backgrounds with high statistical uncertainties could be investigated.

In order to perform the five-fold cross-validation that is part of the evaluation of each optimization step as described in Sec. 4.2.5, the following event splitting was chosen:

- training: $(\text{EventNumber} - k + 1) \bmod 5 \neq 0$
- validation: $(\text{EventNumber} - k + 1) \bmod 5 = 0$

for $k = 0, 1, \dots, 4$. This produces the same datasets as for the cross-validation performed for training of the final BDTs, only in a different order. Since the significances of all five cross-validation models are averaged, the final result is not influenced by this choice.

For all optimizations presented in this section, a few common settings were chosen based on initial manual testing and were not subject to optimization. These settings will now be presented briefly.

During the manual testing phase, most models were trained for 50 to 100 epochs before their validation losses stopped decreasing. Based on these findings, setting the `patience` and `overfitting_patience` parameters of the Custom Early Stopping discussed in Sec. 4.2.4 to 10 epochs and 6 epochs, respectively, seemed appropriate.

In order to reduce the impact of the weight initialization on the trial scheduling performed by the `ASHAScheduler` (see Sec. 4.2.4), the `ASHA_grace_period` (minimum number of epochs r before the first reduction) was set to a conservative value of 10 epochs. For the `ASHA_reduction_factor` η , a value of 2 was selected. In addition to these two parameters, the implementation of ASHA in Tune allows to control the maximum number of reductions using the `ASHA_max_t` parameter. Setting it to a value of 75 epochs² results in a total of 3 reductions after 10, 20 and 40 epochs. When Population Based Training was used, the number of hidden layers and the number of neurons in each layer were fixed to the values of the best trial from the previous optimization step, selected using the fit function evaluation approach (see Sec. 4.2.5). The implementation of PBT included in Tune allows to set a grace period before the first perturbation. Setting this `burn_in_period` parameter to 6 epochs was found to be sufficient to identify underperforming trials. For the `perturbation_interval`, a value of 4 epochs was chosen.

²Since the option to terminate trials after `ASHA_max_t` epochs was disabled, any value of `ASHA_max_t` between 41 and 79 would achieve the same result.

5.5.2 Determination of the optimal settings for the TPESampler and the reporting of results to Tune

As explained in Sec. 4.2.4, each trial needs to report its performance to Tune in regular intervals. It was explained that, in the optimization framework, two options on how to handle the reporting are implemented. Before performing the first actual optimizations, it was therefore necessary to investigate the influence of the reporting method.

In addition to that, Optuna provides two versions of the **TPESampler** which use different methods to build the densities $\ell(\theta)$ and $g(\theta)$ of “good” and “bad” observations (see Sec. 4.1). The “independent” **TPESampler** represents these densities using multiple one-dimensional kernel density estimators, with one KDE per hyperparameter. The “multivariate” **TPESampler** on the other hand uses a single multidimensional KDE, allowing to capture the interactions between different parameters. More details can be found in Ref. [72]. While the multivariate approach promises superior modeling of the neural network’s performance, benchmarks by the Optuna developers showed that this advantage does not materialize in all applications.

In order to assess the influence of both settings on the optimization result, ten optimization runs (*experiments*) of 10000 trials each were performed for each combination of settings. Only 25000 Monte Carlo events ($\approx 10\%$ of the full dataset) were used for these optimizations to reduce the computing requirements. Because of the large imbalance between the available signal and background events, keeping the same signal to background ratio as for the full dataset would result in the use of only ≈ 2200 signal events. Since the absolute performance of the optimized neural networks was not important for this test but a stable convergence of the search algorithms was desired, it was instead decided to use an equal number of signal and background events. Only the main optimization step was performed, using the full set of input variables, to further reduce the resource requirements and possible variability between different runs, omitting the pre-optimization, input variable selection and fine-tuning steps.

With the exception of the activation function, kernel- and bias initializer and the loss function, all tunable hyperparameters that are supported by the built-in **build**-function were included in the search space. An overview over all hyperparameters was given in Sec. 4.2.4. The marginal distributions for each hyperparameter in the search space are collected in Tab. 5.13. Instead of directly including the β_2 parameter of the ADAM optimizer (see Eq. 3.20), it was decided to provide the marginal distribution of $1 - \beta_2$ to allow the log-uniform sampling around the default value $\beta_2 = 0.999$.

The Swish function was chosen as the activation function due to restrictions of the *Lightweight Trained Neural Network* (lwttn) [73] format that is intended to be used for the final model in the full analysis. As of the time of writing, the only interesting activation functions supported by the lwttn-format are the exponential linear unit (ELU, see Ref. [74]) and the Swish activation function. In Ref. [50], the weight initialization of MLPs with Swish-activation was investigated, an equivalent study for the ELU activation function was not found. Therefore, it was decided

Table 5.13: Hyperparameter search space used for the determination of the best settings for the TPESampler and the reporting of results to Tune.

Hyperparameter	Distribution	Min	Max	Step
num_layers	uniform	3	6	1
units	log-uniform	4	128	1
l1_lambda	log-uniform	10^{-18}	0.1	–
l2_lambda	log-uniform	10^{-18}	0.1	–
dropout	uniform	0	0.2	–
batch_size	log-uniform	4	256	1
learning_rate	log-uniform	10^{-5}	0.01	–
Adam_beta_1	uniform	10^{-4}	0.99	–
one_minus_Adam_beta_2	log-uniform	10^{-5}	0.9999	–
Adam_epsilon	log-uniform	10^{-10}	1	–
loss_signal_weight	log-uniform	0.1	10	–

use the Swish activation function and consequently the weight and bias initialization schemes proposed in [50] for all optimizations in this thesis.

Since within this thesis, no alternatives to the binary cross-entropy loss function are investigated, the signal weighted version of the binary cross-entropy loss defined in Eq. 4.6 was set as the loss function for all optimizations. Additionally, its value on the validation dataset was also used as the target of the optimization. An alternative optimization target will be discussed in Sec. 5.5.3. Since the weight of signal events w_s (see Eq. 4.6) was included in the search space, the loss values reported by Keras cannot be directly used for the optimization. Instead, the binary cross-entropy loss with fixed $w_s = 1$ was defined as a custom metric.

The relevant parts of the configuration file for the optimizations corresponding to the “multivariate” TPESampler and reporting the best results to Tune can be found in Appendix F.1. For the other optimizations, only the options `optimize_on_best_value` and `use_multivariate_TPE` needed to be changed.

In order to evaluate the convergence of each search algorithm, the same technique that is applied during the experiment evaluation detailed in Sec. 4.2.5 is used. For each optimization run, the validation loss values reported by each trial are concatenated and sorted by the time of the report. This yields ten time-sorted lists of validation losses for each setting where the number of entries in each list equals the total number of trained epochs (summed across all trials) in the corresponding optimization run. In these lists, the i -th row contains the validation loss value obtained after the i -th finished epoch, counted across all trials in the experiment. In order to obtain a measure of the progression of each optimization run, a running minimum is applied to each list. As a result, the i -th list entries now contain the *lowest* validation loss value seen until epoch i in the corresponding experiment. Finally, for each optimization setting, the mean value of each row over the ten lists is calculated to obtain the average best validation loss seen until each epoch; the corresponding standard deviations are used as its uncertainty.

The trends of these average values and their uncertainties for all four optimization settings are shown in Fig. 5.8. It can be seen that the “independent” `TPESampler` consistently outperformed the “multivariate” version, resulting in lower minimum validation loss values for the entire duration of the experiment. While such an impact on the average best validation loss cannot be found for the two investigated reporting methods, a clear influence on the variability between different experiment runs is visible. Throughout the majority of the experiment, the standard deviation between the ten experiment runs using the same settings is significantly smaller for the method that reports the “last valid” metric value. Thus, for the remaining optimizations presented in this section, the “independent” `TPESampler` and the “last valid” reporting method are used.

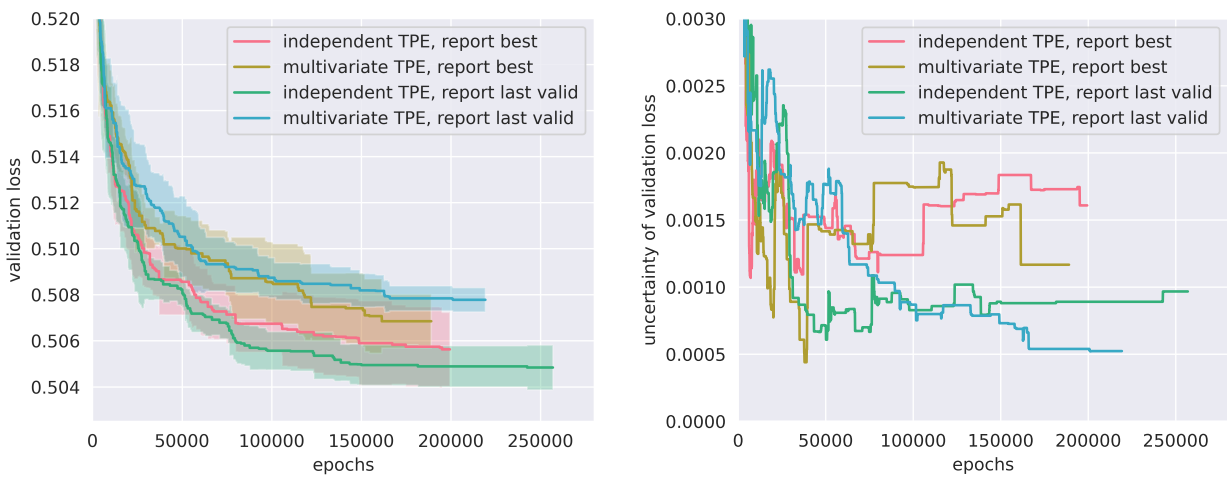


Figure 5.8: Minimum validation loss as a function of the total number of epochs for four different optimization settings. For each setting, ten optimization runs were performed. On the left, the mean minimum validation loss is displayed. The uncertainty bands correspond to the standard deviation across the ten optimization, which is shown on the right.

5.5.3 Determination of the optimal hyperparameter search space

Due to the limited number of hyperparameter combinations that can reasonably be sampled within an optimization run, the inclusion of unimportant hyperparameters can easily lead to degraded optimization results because of the increased dimensionality of the search space. While all potentially interesting hyperparameters were included in the optimization when searching the best `TPESampler`-settings and reporting method, the usefulness of each parameter for the final optimization result first needs to be proven.

In order to arrive at the smallest reasonable hyperparameter search space, multiple optimizations with differing search spaces are performed. For hyperparameters that were not optimized, the default values shown in Tab. 5.14 were used. The defaults for the parameters of the ADAM optimizer correspond to the default values suggested by the developers (see Ref. [37]); the default batch size was set to a value that was found to work reasonably well during manual

testing. All hyperparameters included in the search space were sampled from the marginal distributions given in Tab. 5.13.

Table 5.14: Default values for hyperparameters not included in the hyperparameter search space.

Hyperparameter	Default value
l1_lambda	0
l2_lambda	0
dropout	0
batch_size	64
learning_rate	0.001
Adam_beta_1	0.9
one_minus_Adam_beta_2	0.001
Adam_epsilon	10^{-7}
loss_signal_weight	1

Like for the optimizations presented in the previous section, only the main optimization step using the full set of input variables was executed in order to reduce the resource requirements. In contrast to the previous optimizations, however, only 2000 instead of 10000 trials were sampled and the entire dataset was used to train the MLPs. The necessary adaptations of the configuration file can be found in Appendix F.2.

In order to evaluate the optimization result for each tested hyperparameter search space, the five cross-validation models whose hyperparameters were selected using the fit function evaluation method (see Sec. 4.2.5) were chosen. For all models, the stacked output distributions on the respective validation datasets were calculated. Equivalent to the evaluation of the final BDTs (see Sec. 5.4.4), these distributions were scaled by a factor of 5 to reach the same luminosity as the full dataset. For each output distribution, a maximum likelihood fit was performed to determine the expected significance of the LL polarization component, following the fit strategy detailed in Sec. 5.2. Finally, the mean of the five significances was calculated and the corresponding standard deviation was used as a measure for the statistical uncertainty. It is however important to note that, due to the systematic differences between different event splittings which will be discussed in the next section, the standard deviations are likely overestimations of the actual statistical uncertainty of the mean significance.

In Tab. 5.15, the iterative process followed to build the optimal hyperparameter search space is summarized. Starting with only the number of hidden layers and the number of units per hidden layer, the search space was increased by iteratively trying multiple combinations of additional hyperparameters and selecting the best combination for the next iteration. Since a full optimization needed to be performed for each tested combination of hyperparameters, only combinations considered to most likely result in an improvement were tested during each iteration. Due to the likely overestimation of the statistical uncertainty of the expected

significance, instead of following common rounding rules, the same number of digits are shown for all stated significances within this chapter – irrelevant of the associated uncertainty estimate – to better show the differences between optimization runs.

Table 5.15: Overview of the optimizations performed to determine the minimal reasonable hyperparameter search space for the binary cross-entropy optimization target. The hyperparameters `one_minus_Adam_beta_2` and `loss_signal_weight` have been shortened to `Adam_beta_2` and `signal_weight`, respectively. The horizontal lines separate hyperparameter combinations that were investigated within one iteration. Bold significances denote hyperparameter combinations that were selected for the next iteration.

Nr.	Exp. significance [σ]	num_layers	units	l1_lambda	l2_lambda	dropout	batch_size	learning_rate	Adam_beta_1	Adam_beta_2	Adam_epsilon	signal_weight
1	1.246 ± 0.043	✓	✓									
2	1.284 ± 0.055	✓	✓	✓								
3	1.243 ± 0.025	✓	✓		✓							
4	1.225 ± 0.038	✓	✓	✓	✓							
5	1.298 ± 0.049	✓	✓			✓						
6	1.307 ± 0.057	✓	✓	✓		✓						
7	1.311 ± 0.053	✓	✓		✓	✓						
8	1.324 ± 0.056	✓	✓	✓	✓	✓						
9	1.328 ± 0.047	✓	✓			✓	✓					
10	1.342 ± 0.052	✓	✓			✓	✓	✓	✓	✓	✓	
11	1.361 ± 0.030	✓	✓	✓		✓	✓	✓	✓	✓	✓	
12	1.343 ± 0.048	✓	✓		✓	✓	✓	✓	✓	✓	✓	
13	1.326 ± 0.048	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
14	1.340 ± 0.053	✓	✓			✓	✓	✓	✓	✓	✓	✓
15	1.318 ± 0.056	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓

Using this iterative process, the hyperparameter search space corresponding to the Nr. 11 in Tab 5.15 was found to result in the best performing MLPs, i.e. the highest average expected significance. This search space was thus used for all remaining optimizations which used the binary cross-entropy loss as the optimization target. The resulting optimized hyperparameters are collected in Tab. 5.16.

In Fig. 5.9a, the normalized signal and background distributions of the MLP corresponding to the $k = 0$ cross-validation splitting on the training and validation dataset are shown. For the signal distribution, a slight generalization gap of about the same size as for the final BDTs (see Fig. 5.7a) is visible, especially for low DNN scores. Since no overfitting conditions were enforced during the optimization, the presence of a visible generalization gap is not

Table 5.16: Optimized values of the hyperparameters corresponding to optimization number 11.

Hyperparameter	Optimized value
num_layers	3
units	109
l1_lambda	$2.53 \cdot 10^{-9}$
dropout	0.288
batch_size	222
learning_rate	$7.69 \cdot 10^{-4}$
Adam_beta_1	0.271
one_minus_Adam_beta_2	0.0237
Adam_epsilon	$2.79 \cdot 10^{-5}$

surprising. The influence of this gap on the expected significance will be discussed in the next section. In Fig. 5.9b-5.9d, the normalized distributions of the three polarization components and the stacked and binning optimized distributions of all contributing processes are displayed. Comparing these distributions with the results obtained with the final BDTs in Fig. 5.7, a slightly better classification of the LL polarization component is visible. The distributions of the remaining processes are very similar to the distributions obtained with the BDTs, and especially the LT polarization component is still not separated well from the signal process. This overall slightly better separation between signal and background is confirmed by the maximum likelihood fit, resulting in an expected significance of the LL polarization component of 1.361 ± 0.030 standard deviations, around 8% higher than the result obtained with the final BDTs.

When training a neural network using binary cross-entropy loss, the misclassification of signal and background events are penalized equally. Assuming the same total weight of both classes, this motivates the network to split the training data into two equally pure classes. However, when applying the NN as the final discriminator in a particle physics analysis, the classification accuracy of signal and background events are not necessarily of equal importance. [75] Instead, the ultimate goal when searching for the existence of a process is to maximize the test statistic used for the maximum likelihood fit (see e.g. Ref. [76]). Thus, the best solution would be to build a loss function to directly optimize the value of this test statistic. Due to the differentiability requirement of the loss function, a differentiable approximation of the binned output distributions, e.g. based on kernel density estimators, would be necessary. So far, this approach has not yet been implemented but may be investigated in the future.

In Ref. [75], two loss functions to directly optimize the discovery significance based on estimates of the statistical significance $S/\sqrt{S+B}$ and the Asimov estimate Z_A are proposed. However, these loss functions are only applicable to cut-and-count analyses and have only been shown to improve the result in the case of high systematic uncertainties. Both aspects are not fulfilled

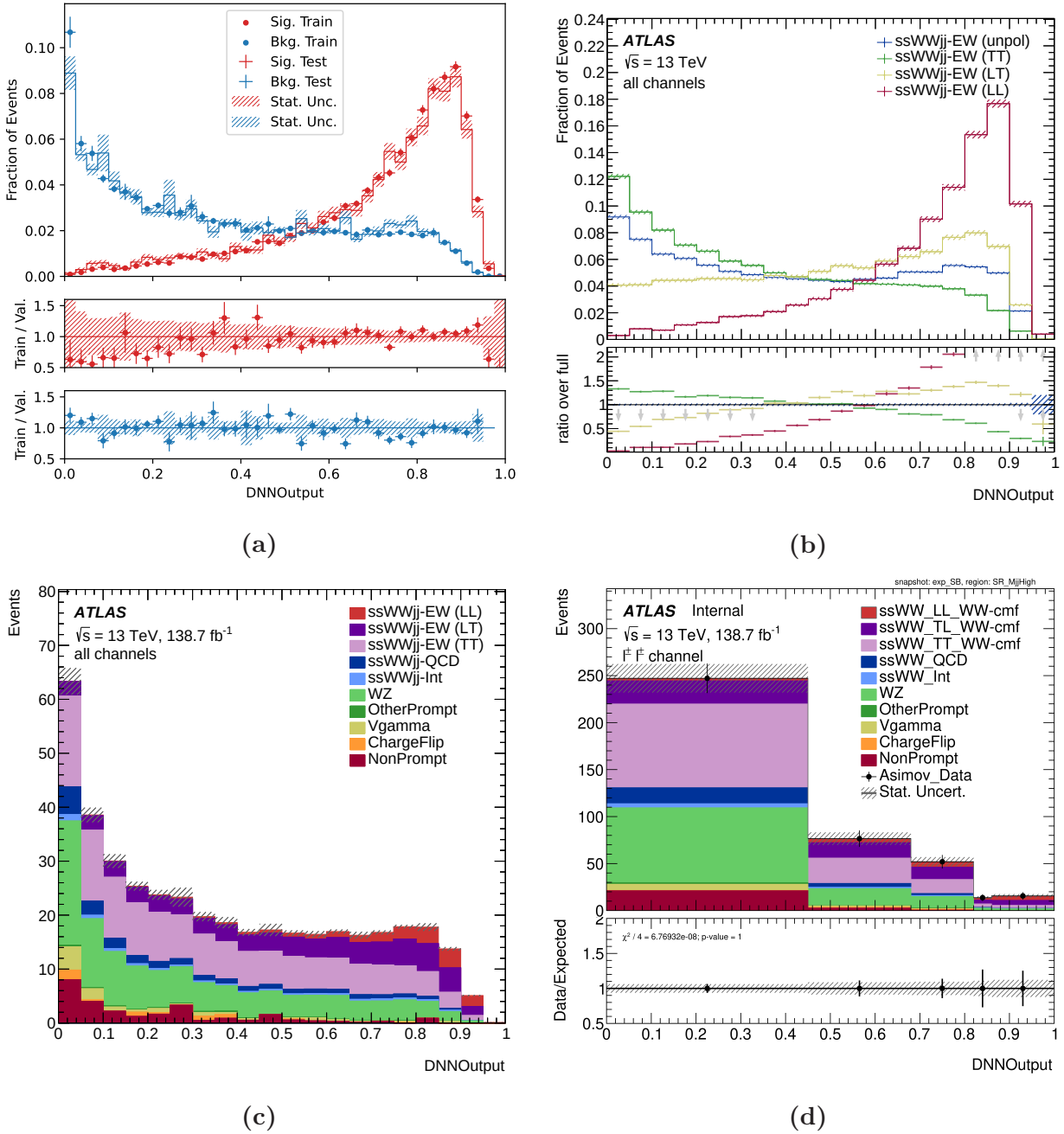


Figure 5.9: (a) Normalized signal and background distributions on the training and validation dataset, (b) the normalized output distributions for the three polarization components of the $W^\pm W^\pm jj$ -EW process and (c) the stacked output distribution for all contributing processes using the full dataset and (d) the binning optimized stacked output distribution using the validation dataset in the $W^\pm W^\pm jj$ -EW signal region for the MLP obtained from the hyperparameter optimization with search space Nr. 11 (see Tab. 5.15) corresponding to the $k = 0$ cross-validation event splitting. For figure (a), the contribution of the LL polarization component of the $W^\pm W^\pm jj$ -EW process is used for the signal distribution while the background distribution is given by the sum of the remaining polarizations and background processes. Only statistical uncertainties are shown in all plots.

for this analysis.

As an intermediate step towards a loss function to directly optimize the test statistic, the *Asimov log-likelihood ratio* (LLR), assuming only statistical uncertainties:

$$\begin{aligned}
q_0^{\text{Asimov}} &= -2 \ln \frac{L(0)}{L(\hat{\mu} = 1)} \\
&= -2 \ln \left(\prod_{i=1}^{N_{\text{bins}}} \frac{\frac{B_i^{N_i}}{N_i!} e^{-B_i}}{\frac{(S_i+B_i)^{N_i}}{N_i!} e^{-(S_i+B_i)}} \right) \\
&= -2 \ln \left(\prod_{i=1}^{N_{\text{bins}}} \left(\frac{B_i}{S_i + B_i} \right)^{S_i+B_i} e^{S_i} \right) \\
&= -2 \sum_{i=1}^{N_{\text{bins}}} \left((S_i + B_i) \ln \left(\frac{B_i}{S_i + B_i} \right) + S_i \right),
\end{aligned} \tag{5.10}$$

was implemented as a custom metric in the optimization framework. Here, $L(\mu)$ denotes the binned likelihood function for measured event yields n_i in the individual bins i . Assuming only statistical uncertainties, which was done for the definition of the LLR, L is defined as:

$$L(\mu) = \prod_{i=1}^{N_{\text{bins}}} \text{Pois}(n_i | \mu s_i + b_i) = \prod_{i=1}^{N_{\text{bins}}} \frac{(\mu s_i + b_i)^{n_i}}{n_i!} e^{-(\mu s_i + b_i)}. \tag{5.11}$$

Given the neural network predictions, corresponding target labels and event weights and the total expected signal and background yields, histograms of weighted signal and background events with 100 bins are created. The total signal and background weights are scaled to reach the expected yields for each class. Bins are merged until at least 10 weighted events are contained within each bin to improve the resistance to statistical fluctuations. With the obtained two histograms, the value of q_0^{Asimov} is calculated and returned to the optimization framework.

By setting this custom metric as the target of the optimization, the hyperparameter optimization now evaluates the performance of each MLP based on the achieved LLR. However, this does not yet have an influence on the training of each neural network and thus on the importance of each class. In order to allow the tuning of the relative importance of misclassifying signal and background events, a signal weight was introduced that is applied to the loss values of signal events, as already shown in Eq. 4.6. The value of this signal weight was used as an additional hyperparameter `loss_signal_weight` and was already included in the previous optimizations. Starting from the hyperparameter search space found to be optimal when using the binary cross-entropy loss as the optimization target, four additional optimizations using the LLR target were performed. Except for the additional custom metric and the changed optimization target, all optimization settings were left unchanged. The relevant changes can be found in Appendix F.2. An overview of the hyperparameters that were included in the search spaces for

these optimizations and the corresponding mean expected significances is given in Tab. 5.17. It can be seen that the best performing MLPs were produced by optimization number 18, resulting in a mean expected significance of the LL polarization component of 1.35 ± 0.08 standard deviations. The corresponding optimized hyperparameter values can be found in Tab. 5.18.

Table 5.17: Overview of the optimizations performed to determine the minimal reasonable hyperparameter search space for the Asimov log-likelihood ratio optimization target. The hyperparameters `one_minus_Adam_beta_2` and `loss_signal_weight` have been shortened to `Adam_beta_2` and `signal_weight`, respectively.

Nr.	Exp. significance [σ]	num_layers	units	l1_lambda	l2_lambda	dropout	batch_size	learning_rate	Adam_beta_1	Adam_beta_2	Adam_epsilon	signal_weight
16	1.312 ± 0.086	✓	✓	✓		✓	✓	✓	✓	✓	✓	
17	1.336 ± 0.082	✓	✓			✓	✓	✓	✓	✓	✓	
18	1.350 ± 0.075	✓	✓			✓	✓	✓	✓	✓	✓	✓
19	1.338 ± 0.067	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓

Possible reasons for why the usage of the LLR could not improve the performance of the MLPs are the facts that neither Monte Carlo statistical uncertainties nor the additional nuisance parameter for the normalization of the polarization sum are considered. Both aspects can be incorporated into the definition of the custom metric and could be investigated in future studies.

Although the LLR did not improve upon the results obtained by optimizing on the binary cross-entropy loss, this metric was nevertheless used as a second optimization target for the following investigations in this section due to its motivation presented above. For all optimizations using this metric as the optimization target, the hyperparameter search space of optimization

Table 5.18: Optimized values of the hyperparameters corresponding to optimization number 18.

Hyperparameter	Optimized value
num_layers	4
units	104
dropout	0.223
batch_size	215
learning_rate	$1.47 \cdot 10^{-3}$
Adam_beta_1	0.052
one_minus_Adam_beta_2	0.0266
Adam_epsilon	$6.73 \cdot 10^{-6}$

number 18 is used.

In Fig. 5.10, the output distributions of the MLP corresponding to the $k = 0$ cross-validation splitting are shown. It can be seen that all distributions are strongly skewed towards smaller output values compared to their counterparts when optimizing for the best binary cross-entropy loss (see Fig. 5.9). This is caused by the signal weight hyperparameter w_s , whose optimal value was found to be around 0.18. Similar values were also obtained for optimization number 19 as well as for all remaining unconstrained³ optimizations that will be presented in the following sections. This confirms that low values of w_s consistently result in a higher LLR.

5.5.4 Overfitting conditions

So far in this chapter, all stated expected significances were determined by averaging the results of multiple maximum likelihood fits, each using a different classifier and the corresponding subset of the dataset that was not used for the training. While this approach provides a good estimate of the best generalization performance that can be achieved with a given method, it is not applicable to the final analysis where only a single maximum likelihood fit will be performed. The most straightforward solution would be to instead select only one of the k cross-validation models and use the entire dataset, including the training data, for the final fit. The potential issue that arises when reusing the training data is the introduced dependency of the fit result on the classifier's generalization gap. It is therefore of great importance to investigate the influence of the generalization gap on the expected and observed significances.

The inclusion of training data in the fit for a non-vanishing generalization gap implies a better separation of signal and background for the Monte Carlo simulated contributions than for measured data. Intuitively one would assume that this mismodeling, on average, results in lower observed than expected significances. This is, however, not the case when the region of the highest signal contribution (i.e. the bins corresponding to high output scores of the classifier) is background dominated, which typically happens when the expected signal yields are much smaller than the yields of background processes. In this situation, the generalization gap results in an underprediction of the background contribution that is not compensated by the overprediction of the signal contribution. Thus, on average, an excess will be observed in the region of the highest signal contribution, caused by the mismodeling introduced by reusing the training dataset for the final maximum likelihood fit.

This effect is visualized in Fig. 5.11 using a prefit distribution of the $k = 0$ cross-validation MLP of the best optimization with the LLR target from Sec. 5.5.3. For this figure, the training dataset was used for the stacked signal and background distributions while the predicted yields using the validation data were used for the “observed” data. Both the stacked distributions and the simulated data were scaled to the same integrated luminosity as the full dataset and the

³For some optimizations with very strict overfitting conditions (see Sec. 5.5.4), optimized values of w_s between 0.5 and 2 were obtained.

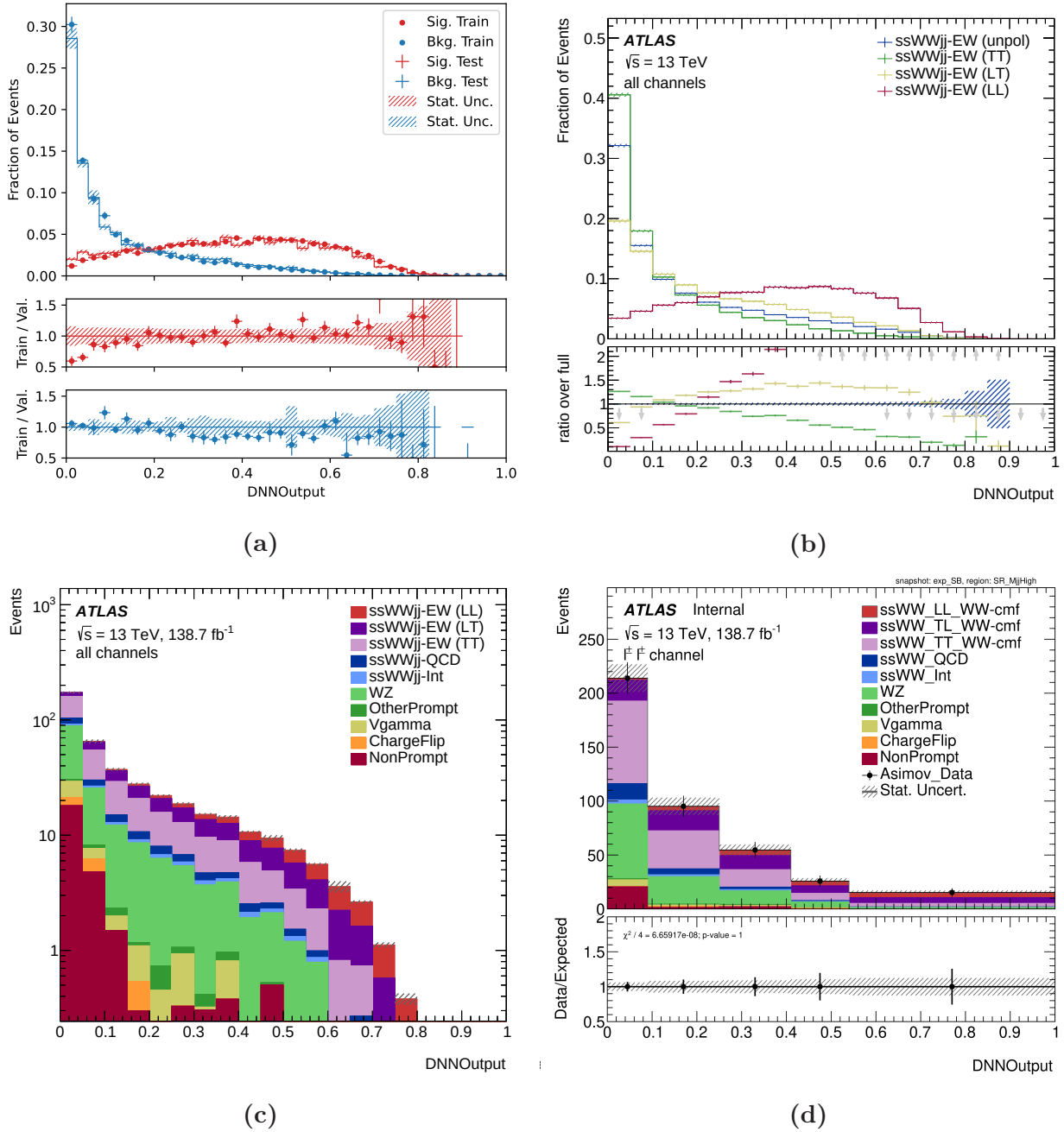


Figure 5.10: (a) Normalized signal and background distributions on the training and validation dataset, (b) the normalized output distributions for the three polarization components of the $W^\pm W^\pm jj$ -EW process and (c) the stacked output distribution for all contributing processes using the full dataset and (d) the binning optimized stacked output distribution in the using the validation dataset in the $W^\pm W^\pm jj$ -EW signal region for the MLP obtained from the hyperparameter optimization with search space number. 18 (see Tab. 5.17) corresponding to the $k = 0$ cross-validation event splitting. For figure (a), the contribution of the LL polarization component of the $W^\pm W^\pm jj$ -EW process is used for the signal distribution while the background distribution is given by the sum of the remaining polarizations and background processes. Only statistical uncertainties are shown in all plots.

binning optimization described in Sec. 5.2 was applied. A clear shape difference between the stacked and the simulated data distributions is visible, resulting in an underprediction in the bins with the highest signal contribution. During the fit, this excess is corrected by increasing the signal strength to a value larger than one, resulting in systematically higher observed than expected significances.

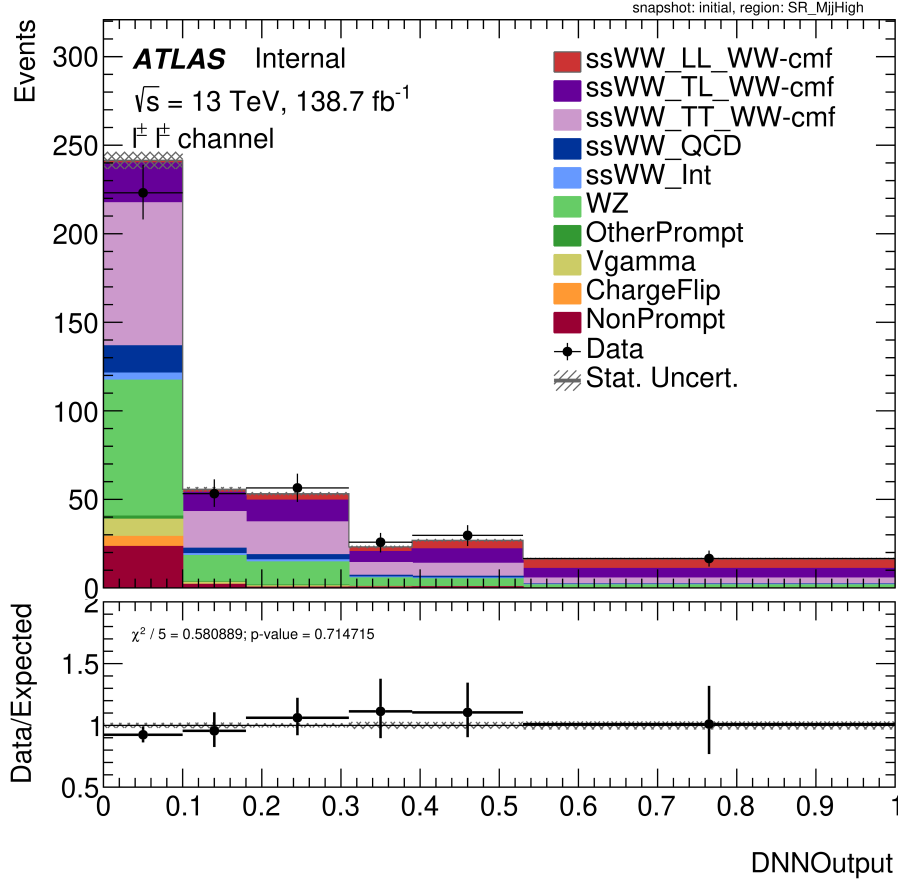


Figure 5.11: Visualization of the influence of the generalization gap on the observed significance using the $k = 0$ cross-validation MLP of the best optimization with Asimov log-likelihood ratio target from Sec. 5.5.3. For this profit distribution, the training dataset was used for the stacked contributions of all contributing processes in the $W^\pm W^\pm jj$ -EW signal region while the “observed” data corresponds to the the sum of the expected yields of these processes as predicted with the validation dataset. Both distributions were scaled to the integrated luminosity of the full dataset and the binning optimization described in Sec. 5.2 was applied. Only statistical uncertainties are shown.

With the overfitting conditions described in Sec. 4.2.2, the optimization framework provides a method to constrain the size of the generalization gap during the training. In order to investigate if the mismodeling introduced by reusing the training data during the fit can be sufficiently reduced, overfitting conditions based on the ratio of the values of target metric (binary cross-entropy loss l or LLR f_{LLR}) on the training and validation datasets were defined. For optimizations using the binary cross-entropy loss target, the overfitting conditions were of

the form:

$$\frac{l^{\text{train.}}}{l^{\text{val.}}} < c, \quad (5.12)$$

while for the optimizations with the LLR target, the condition:

$$\frac{f_{\text{LLR}}^{\text{val.}}}{f_{\text{LLR}}^{\text{train.}}} < c, \quad (5.13)$$

was applied. As explained in Sec. 4.2.2, the generalization gap of an MLP is considered too large, if the respective metric values fulfill the condition.

During the investigations presented in this section, significant systematic differences between the individual event splittings were observed. In Tab. 5.19, the expected event yields using the validation datasets corresponding to the five cross-validation splittings, scaled to the same integrated luminosity of the full dataset, are shown. In can be seen that the majority of the differences can be attributed to the non-prompt background contributions.

Due to these differences, the size of the observed generalization gap – i.e. the difference between the value of the target metric on the training and validation datasets – strongly depends on the specific event splitting. As a result, applying the same overfitting condition to the training of all five cross-validation models caused the termination of the training at vastly different points. For very strict overfitting conditions (i.e. c values near 1), the observed generalization gap for the $k = 1$ cross-validation model exceeded the allowed threshold already after the first epoch while for the $k = 2$ splitting, the training continued for more than 50 epochs. In order to mitigate this problem, it was decided to not evaluate the overfitting condition during the cross-validation and instead use the optimal number of epochs determined from the experiment evaluation (see Sec. 4.2.5). This can be realized by disabling the option `use_OT_conditions_for_crossvalidation` in the configuration file. As the overfitting condition is evaluated during the optimization and the experiment evaluation too, the threshold value c still controls the allowed size of the observed generalization gap for the $k = 0$ event splitting and thus still influences the generalization of the MLPs. However, all further investigations are now specific to the event splitting used during the optimization and cannot be generalized to other possible splittings. Therefore, this approach is not recommended when the differences between individual splittings are small.

For both the binary cross-entropy and LLR optimization targets, multiple optimizations for different values of c were performed to determine the dependency of the generalization gap – and thus the overestimation of the observed significance – on the threshold value c of the overfitting condition. All optimizations used the hyperparameter search spaces determined in the previous section. For all hyperparameters not included in the search space, the default values given in Tab. 5.14 were set. The settings were kept the same, except for the `use_OT_conditions_for_crossvalidation`-flag and the overfitting conditions, whose definitions can be found in see Appendix F.3.

The evaluation of each optimization was performed using the five cross-validation models whose

Table 5.19: Expected event yields using the validation datasets corresponding to the k cross-validation splittings, scaled to the same integrated luminosity of the full dataset. Additionally, the estimated discovery significance $S/\sqrt{B}$ is given for each splitting, with S and B the expected yields of the $W_L^\pm W_L^\pm jj$ -EW and the sum of all background contributions, respectively.

Process	Event yield				
	$k = 0$	$k = 1$	$k = 2$	$k = 3$	$k = 4$
$W_L^\pm W_L^\pm jj$ -EW	19.82	20.37	19.59	19.96	20.26
$W_L^\pm W_T^\pm jj$ -EW	63.49	65.34	65.71	64.53	65.71
$W_T^\pm W_T^\pm jj$ -EW	137.96	138.89	138.65	135.32	138.81
$W^\pm W^\pm jj$ -QCD	23.97	23.95	23.57	23.09	23.67
$W^\pm W^\pm jj$ -Int	7.32	7.26	7.42	7.41	7.42
$W^\pm Z$	114.21	116.98	117.06	112.30	115.01
Other prompt	2.89	2.37	2.75	3.32	2.83
$V + \gamma$	8.77	12.57	14.61	9.20	11.16
Charge-flip	2.04	2.65	6.19	6.22	7.86
Non-prompt	24.48	14.30	43.68	23.52	25.76
Total MC	404.96	404.68	439.24	404.88	418.50
$S/\sqrt{B}$	1.010	1.039	0.956	1.017	1.015

hyperparameters were selected based on the best value of the target metric⁴ (see Sec. 4.2.5). For each MLP, three maximum likelihood fits were done. The first two fits were performed on Asimov data, using only the training dataset for the first and the validation dataset for the second fit; both sets were scaled to the same integrated luminosity as the full dataset. For the third fit, the scaled training dataset was used for the stacked distributions and the scaled validation set for the simulated “observed” data, as explained above. As with all results presented previously, the mean of the five obtained significances for each type of fit was calculated and the standard deviation was used as the uncertainty. The first two obtained mean significances can be interpreted as measures for the performance of the MLP on the training data σ_{train} and on unseen data σ_{exp} , respectively, while their difference acts as a measure for the generalization gap. The third mean significance corresponds to the “expected observed” significance $\hat{\sigma}_{\text{obs}}$ when using only the training dataset for the maximum likelihood fit. Its difference to the mean expected significance obtained using the validation dataset σ_{exp} can thus be interpreted as an upper bound on the expected overestimation of the observed significance when using the full dataset.

The obtained values of σ_{exp} , $\hat{\sigma}_{\text{obs}}$, $\sigma_{\text{train}} - \sigma_{\text{exp}}$ and $\hat{\sigma}_{\text{obs}} - \sigma_{\text{exp}}$ for the optimizations performed using the binary cross-entropy loss optimization target are collected in Tab. 5.20. Tab. 5.21

⁴At the time these investigations were done, the R-squared based condition on the fit performed during the experiment evaluation (see Sec. 4.2.5) was not yet implemented. The fit-based evaluation was therefore less reliable than the method based on the best metric value, especially for very strict overfitting conditions that resulted in few trained epochs.

compiles the same quantities for the optimizations with the LLR target. Additionally, the mean significances obtained by the three maximum likelihood fits performed for each optimization as a function of the threshold c of the overfitting condition for both types of optimizations are shown in Fig. 5.12. It is important to note that $\hat{\sigma}_{\text{obs}}$ and consequently $\hat{\sigma}_{\text{obs}} - \sigma_{\text{exp}}$ are especially sensitive to the systematic differences between the individual event splittings, resulting in the large standard deviations that are used to indicate the uncertainties.

While a decrease of the generalization gap and the resulting overestimation of the observed significance for increasing values of c is seen for the optimizations with the binary cross-entropy loss target, no sufficient reduction was achieved even for $c = 0.99$. As values of $c \geq 1$ would require the MLP to perform at least as good on the validation as on the training dataset, larger values of c were not tested.

For the optimizations using the LLR target, a stronger dependence of the generalization gap on the value of c is visible. For $c \leq 0.9$, the overfitting condition was loose enough to not significantly effect the optimization, i.e. the generalization gap of the best performing MLPs was naturally small enough without any constraint. Thus, the overestimation of the observed significance stays mostly constant. For values $0.9 < c < 0.95$, an almost linear decrease of $\hat{\sigma}_{\text{obs}} - \sigma_{\text{exp}}$ is observed, indicating that the overfitting condition effectively constrained the generalization gap. For larger values of c , however, significant fluctuations without a further reduction of the generalization gap are visible. Additionally, a significant decrease of σ_{exp} is observed, indicating that the constraint on the generalization gap is hampering the performance of the MLPs. For such high c values, the fluctuations occur as the experiment evaluation was dominated by outliers where the metric values of a single epoch, by chance, happened to pass the strict overfitting conditions. This issue can be prevented by using the fit function evaluation method (see Sec. 4.2.5). Unfortunately, at the time these investigations were performed, this method was not fully implemented yet and led to erroneous behavior for low numbers of trained epochs. Due to the high resource requirements of these investigations and the fact that the performance of the classifier was already strongly reduced by the strict overfitting conditions, it was decided to not repeat the optimizations once the fit function evaluation method was completed.

Since neither of the two overfitting conditions was able to sufficiently reduce the expected overestimation of the observed significance without a simultaneous reduction of generalization performance, the training data cannot be reused for the maximum likelihood fit. Within the analysis group, it is currently being investigated to instead combine the output distributions of the five cross-validation models on their respective validation datasets into a single histogram for the final fit. This would remove the dependency of the fit result on the generalization gap. Thus, no overfitting condition was used for the remaining optimizations.

While the application of overfitting conditions proofed to be insufficient for the specific process investigated in this thesis, it is nevertheless possible that – for analyses with higher expected

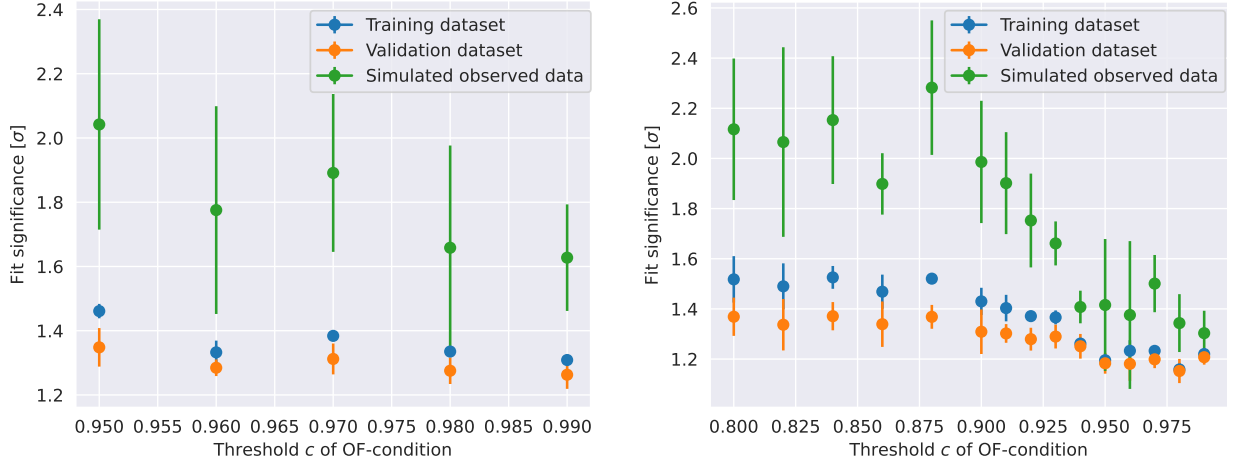


Figure 5.12: The mean significances obtained by the three maximum likelihood fits performed for each optimization as a function of the threshold c of the overfitting condition for the optimization performed using the binary cross-entropy loss (left) and the Asimov log-likelihood ratio (right) optimization targets

signal yields – the use of overfitting conditions could still allow the reuse of the training data for the final fit without a large sacrifice in performance. Especially the condition based on the LLR has been shown to allow an effective reduction of the expected overestimation of the observed significance. This can be explained by the fact that, like $\hat{\sigma}_{\text{obs}} - \sigma_{\text{exp}}$, the ratio of the validation to training LLR is mainly determined by deviations of the background distribution between the training and validation datasets in the signal-rich parts of the output distribution. As a result, the overfitting condition based on the LLR places a more selective constraint on the generalization gap, focusing more on those deviations between the training and validation distributions that have a higher influence on the fit result. In future studies, a version of the LLR that implements the fit to simulated “observed” data could be investigated for its capacity to constrain the generalization gap even more selectively.

Table 5.20: Expected significance σ_{exp} , “expected observed” significance $\hat{\sigma}_{\text{obs}}$, estimated generalization gap $\sigma_{\text{train}} - \sigma_{\text{exp}}$ and the expected overestimation of the observed significance $\hat{\sigma}_{\text{obs}} - \sigma_{\text{exp}}$ for different values of the threshold c of the overfitting condition for optimizations with the binary cross-entropy loss optimization target. The uncertainties of all values are given by the standard deviations of the corresponding quantity for the five performed maximum likelihood fits.

c	σ_{exp}	$\hat{\sigma}_{\text{obs}}$	$\sigma_{\text{train}} - \sigma_{\text{exp}}$	$\hat{\sigma}_{\text{obs}} - \sigma_{\text{exp}}$
0.95	1.348 ± 0.060	2.042 ± 0.327	0.113 ± 0.050	0.694 ± 0.343
0.96	1.285 ± 0.026	1.776 ± 0.323	0.048 ± 0.037	0.491 ± 0.309
0.97	1.312 ± 0.048	1.891 ± 0.246	0.072 ± 0.053	0.579 ± 0.275
0.98	1.275 ± 0.041	1.658 ± 0.318	0.060 ± 0.051	0.383 ± 0.324
0.99	1.263 ± 0.044	1.627 ± 0.166	0.046 ± 0.053	0.364 ± 0.183

Table 5.21: Expected significance σ_{exp} , “expected observed” significance $\hat{\sigma}_{\text{obs}}$, estimated generalization gap $\sigma_{\text{train}} - \sigma_{\text{exp}}$ and the expected overestimation of the observed significance $\hat{\sigma}_{\text{obs}} - \sigma_{\text{exp}}$ for different values of the threshold c of the overfitting condition for optimizations with the Asimov log-likelihood ratio optimization target. The uncertainties of all values are given by the standard deviations of the corresponding quantity for the five performed maximum likelihood fits.

c	σ_{exp}	$\hat{\sigma}_{\text{obs}}$	$\sigma_{\text{train}} - \sigma_{\text{exp}}$	$\hat{\sigma}_{\text{obs}} - \sigma_{\text{exp}}$
0.80	1.369 ± 0.076	2.116 ± 0.282	0.149 ± 0.041	0.747 ± 0.263
0.82	1.337 ± 0.103	2.065 ± 0.378	0.153 ± 0.027	0.728 ± 0.307
0.84	1.371 ± 0.056	2.153 ± 0.255	0.155 ± 0.042	0.782 ± 0.282
0.86	1.339 ± 0.091	1.898 ± 0.122	0.130 ± 0.058	0.559 ± 0.170
0.88	1.368 ± 0.048	2.282 ± 0.268	0.152 ± 0.057	0.914 ± 0.308
0.90	1.309 ± 0.089	1.986 ± 0.244	0.121 ± 0.044	0.677 ± 0.217
0.91	1.302 ± 0.037	1.902 ± 0.203	0.101 ± 0.036	0.599 ± 0.201
0.92	1.279 ± 0.045	1.752 ± 0.187	0.092 ± 0.046	0.473 ± 0.195
0.93	1.290 ± 0.047	1.661 ± 0.088	0.076 ± 0.060	0.371 ± 0.117
0.94	1.251 ± 0.049	1.408 ± 0.065	0.011 ± 0.058	0.157 ± 0.093
0.95	1.183 ± 0.041	1.416 ± 0.263	0.012 ± 0.039	0.232 ± 0.265
0.96	1.181 ± 0.068	1.376 ± 0.295	0.052 ± 0.046	0.195 ± 0.251
0.97	1.199 ± 0.035	1.501 ± 0.114	0.034 ± 0.042	0.302 ± 0.121
0.98	1.152 ± 0.048	1.343 ± 0.115	0.006 ± 0.042	0.191 ± 0.090
0.99	1.207 ± 0.030	1.303 ± 0.089	0.013 ± 0.046	0.096 ± 0.110

5.5.5 Input variable optimization

With the best settings for the `TPESampler` determined and the minimal reasonable hyperparameter search space found for each optimization target, the influence of the input variable selection (IVS) was investigated. As with the BDTs, the incentive to remove unnecessary observables from the set of input variables used to train the MLP is the reduction of possible modeling uncertainties. Additionally, as briefly discussed in Sec. 4.2.6, excluding uninformative input variables can help to improve the MLP’s generalization performance by reducing the number of local minima of the loss function.

For the IVS implemented in the optimization framework, which was presented in detail in Sec. 4.2.6, several settings needed to be specified.

As the time to train a single MLP was on the order of 15 to 30 minutes and required only a single CPU core due to the relatively low network sizes and small training dataset, the `retrain`-method was chosen. In contrast to the alternative `shuffle`-method that can only assess the importance of each input for a given fully trained model, the `retrain`-method allows to evaluate the importance of an observable for solving the given problem. This advantage can be understood by imagining a discriminative variable a that can be constructed from the remaining input variables. If this construction is easy enough to be discovered by the MLP given the available training dataset, a is not necessary to solve the given problem. Removing it from

the dataset will therefore not reduce the performance of the MLP and the **retrain**-method would thus exclude it from the set of input variables. Since the original MLP trained on the full set of inputs is however likely to use the information provided by a for the classification, the **shuffle**-method would find it to be important for the MLPs performance and thus not remove it.

Due to the relatively small size of the training dataset, significant fluctuations between different training runs were observed. In order to avoid premature termination of the IVS as explained in Sec. 4.2.6, the relative changes of the target metric after dropping a variable were calculated with respect to the baseline models trained using the full set of input variables.

Lastly, it had to be decided whether the mean or the median of the five relative changes should be used to quantify the importance of each input variable. Since neither option has a clear advantage over the other, both options needed to be investigated. Since the **retrain**-method was used and thus a performance improvement of the MLPs over the course of the IVS was expected, the IVS was terminated when removing any additional input variable resulted in worse performance than the baseline (by setting $\delta_{\max} = 0$, see Sec. 4.2.6).

For each combination of optimization target (binary cross-entropy loss and LLR) and method to quantify the importance of an input variable (mean and median), an optimization consisting of only the pre-optimization step and the subsequent input variable selection was performed. Since it is expected that, when using the IVS as part of a full optimization run, less CPU time will be allocated for the pre-optimization than for the main optimization, only 1000 trials were sampled instead of the 2000 trials that were used for most of the previously presented optimizations. For all optimization runs, the metric set as the optimization target was also used for the IVS. The cross-validation models whose hyperparameters were selected using the fit function evaluation method were used as the baseline. The relevant parts of the configuration file for the optimization using the binary cross-entropy loss and the mean relative change can be found in Appendix F.4. For the other optimizations, only the options `var_metric`, `var_metric_op` and `use_median_for_averages` needed to be adjusted.

In Tab. 5.22, the optimized sets of input variables obtained for each optimization are collected. In all cases, at least half of the input variables could be removed from the dataset before the performance of the resulting MLPs got worse than the baseline. Moreover, while the results are not identical, many input variables are present in all four optimized sets, showing that important variables are consistently detected.

Fig. 5.13 shows the relative changes of the target metric over the course of the IVS for all four optimizations. In all four trends, a clear initial improvement of the target metric for the first few iterations is visible, which is especially evident for the optimizations using the LLR target. After this initial improvement, the performance of the MLPs stays constant for several iterations before degrading to the level of the baseline near the end of the IVS. This shows that, while the current implementation of the IVS was able to remove half of the available

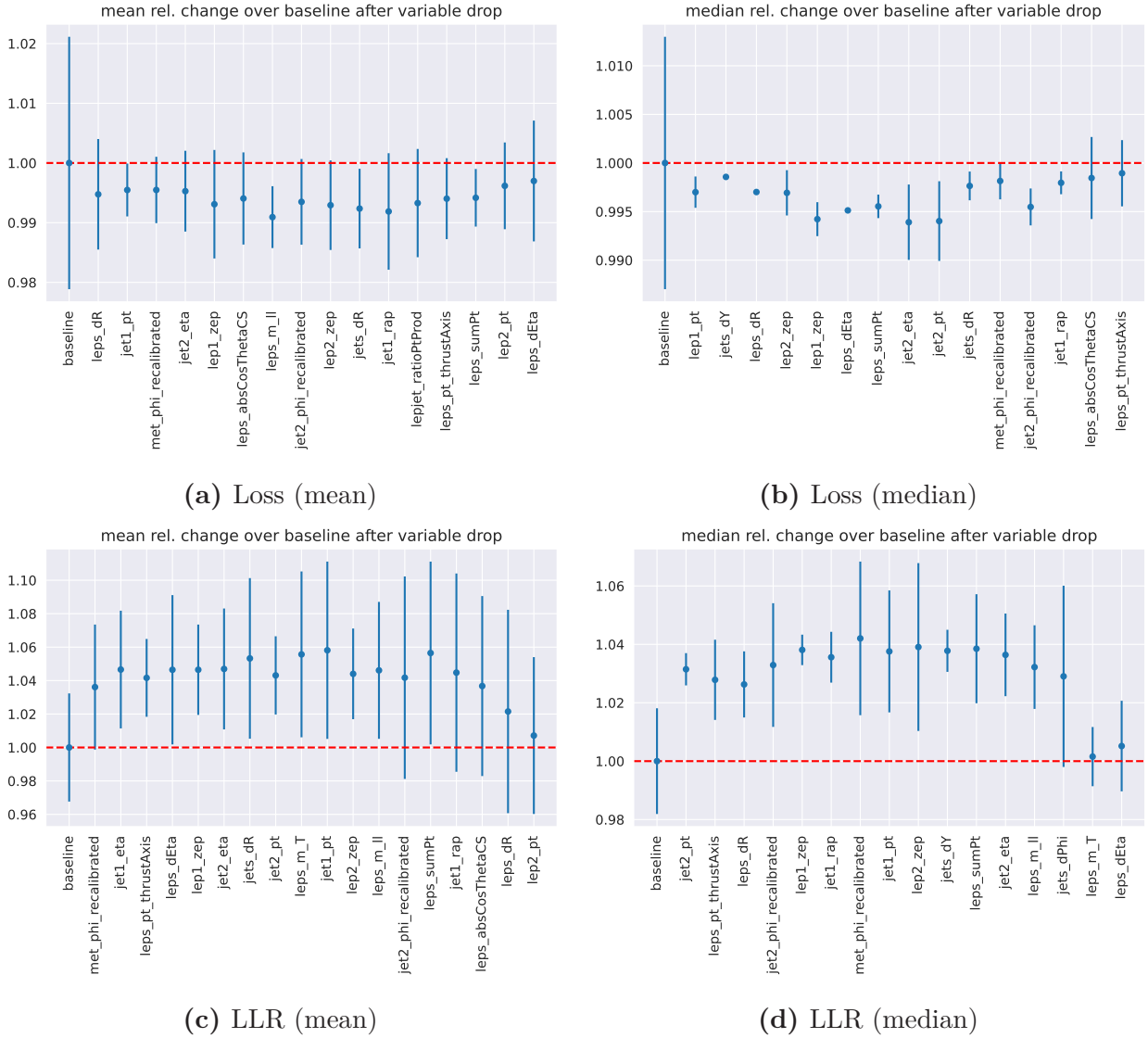


Figure 5.13: Evolution of the mean (left) and median (right) relative changes of the binary cross-entropy loss (top) and the Asimov log-likelihood ratio (bottom) over the course of the input variable selection during the four corresponding optimizations. The variable names on the x-axes denote the input variable that was dropped at the end of each iteration, with **baseline** corresponding to the full set of inputs. For the mean relative changes, the error bars correspond to the standard deviation of the relative changes for the five cross-validation models; for the median relative changes, the median absolute deviation was used.

Table 5.22: Optimized list of input variables obtained from the four performed optimizations. Two methods to obtain the average relative change (mean and median) were tested for each optimization target (binary cross-entropy and Asimov log-likelihood ratio). The entries are sorted by the frequency of occurrence of an observable in the four optimized input variable sets.

Cross-entropy loss		Log-likelihood ratio	
Mean	Median	Mean	Median
Y^{j_2}	Y^{j_2}	Y^{j_2}	Y^{j_2}
$\phi_{\text{recal}}^{j_1}$	$\phi_{\text{recal}}^{j_1}$	$\phi_{\text{recal}}^{j_1}$	$\phi_{\text{recal}}^{j_1}$
η^{ℓ_1}	η^{ℓ_1}	η^{ℓ_1}	η^{ℓ_1}
η^{ℓ_2}	η^{ℓ_2}	η^{ℓ_2}	η^{ℓ_2}
$\phi_{\text{recal}}^{\ell_2}$	$\phi_{\text{recal}}^{\ell_2}$	$\phi_{\text{recal}}^{\ell_2}$	$\phi_{\text{recal}}^{\ell_2}$
$\min(\Delta R_{j\ell})$	$\min(\Delta R_{j\ell})$	$\min(\Delta R_{j\ell})$	$\min(\Delta R_{j\ell})$
E_T^{miss}	E_T^{miss}	E_T^{miss}	E_T^{miss}
m_{o1}	m_{o1}	m_{o1}	m_{o1}
η^{j_1}	η^{j_1}	$p_T^{\ell_1}$	η^{j_1}
$p_T^{\ell_1}$	$\Delta\Phi_{jj}$	$\Delta\Phi_{jj}$	$p_T^{\ell_1}$
$\Delta\Phi_{jj}$	R_{Pt}	R_{Pt}	R_{Pt}
ΔY_{jj}	$p_T^{\ell_2}$	ΔY_{jj}	$p_T^{\ell_2}$
m_T^{WW}	m_T^{WW}	—	ΔR_{jj}
$p_T^{j_2}$	$p_T^{j_1}$	—	$ \cos(\vartheta_{CS}^{\ell\ell}) $
—	$m_{\ell\ell}$	—	—

input variables from the dataset without degradation of the target metric, terminating the IVS earlier has the potential to improve the performance of the MLPs. It is also visible that the option to use the best seen metric value as the reference for the relative changes would have terminated the IVS after the first or second iterations due to random fluctuations, missing the goal of removing a significant fraction of the input variables from the dataset. No significant differences were found between the trends of the mean and median relative changes, thus both methods were used for the final results presented in the Sec. 5.5.6.

An obvious improvement to the current implementation of the IVS would be to revert to the iteration of the best average relative change once the IVS has terminated. This, together with an early-stopping-like termination condition to reduce the runtime of the IVS, has already been implemented and is currently being tested. While the optimizations presented here have not yet been repeated with the added functionality, the influence of choosing the iteration of best average relative change over the last iteration on the performance of the final MLPs can still be investigated. This was done by manually creating the sets of input variables that would be obtained by reverting to the best iterations for each of the four performed optimizations. They are displayed in Tab. 5.23 and were used together with the sets of input variables corresponding to the last iteration of the IVS to obtain the final results presented in the Sec. 5.5.6.

Table 5.23: List of input variables corresponding to the IVS-iteration of the best average relative change for the four performed optimizations. Two methods to obtain the average relative change (mean and median) were tested for each optimization target (binary cross-entropy and Asimov log-likelihood ratio). The entries are sorted by the frequency of occurrence of an observable in the four optimized input variable sets.

Cross-entropy loss		Log-likelihood ratio	
Mean	Median	Mean	Median
Y^{j_2}	Y^{j_2}	Y^{j_2}	Y^{j_2}
$\phi_{\text{recal}}^{j_1}$	$\phi_{\text{recal}}^{j_1}$	$\phi_{\text{recal}}^{j_1}$	$\phi_{\text{recal}}^{j_1}$
η^{ℓ_1}	η^{ℓ_1}	η^{ℓ_1}	η^{ℓ_1}
η^{ℓ_2}	η^{ℓ_2}	η^{ℓ_2}	η^{ℓ_2}
$p_T^{\ell_1}$	$p_T^{\ell_1}$	$p_T^{\ell_1}$	$p_T^{\ell_1}$
$p_T^{\ell_2}$	$p_T^{\ell_2}$	$p_T^{\ell_2}$	$p_T^{\ell_2}$
$\phi_{\text{recal}}^{\ell_2}$	$\phi_{\text{recal}}^{\ell_2}$	$\phi_{\text{recal}}^{\ell_2}$	$\phi_{\text{recal}}^{\ell_2}$
$\Delta\Phi_{jj}$	$\Delta\Phi_{jj}$	$\Delta\Phi_{jj}$	$\Delta\Phi_{jj}$
$\min(\Delta R_{j\ell})$	$\min(\Delta R_{j\ell})$	$\min(\Delta R_{j\ell})$	$\min(\Delta R_{j\ell})$
R_{Pt}	R_{Pt}	R_{Pt}	R_{Pt}
E_T^{miss}	E_T^{miss}	E_T^{miss}	E_T^{miss}
m_{o1}	m_{o1}	m_{o1}	m_{o1}
Y^{j_1}	Y^{j_1}	Y^{j_1}	η^{j_1}
η^{j_1}	η^{j_1}	$\phi_{\text{recal}}^{j_2}$	ΔY_{jj}
$\phi_{\text{recal}}^{j_2}$	$\phi_{\text{recal}}^{j_2}$	ΔY_{jj}	ΔR_{jj}
ΔY_{jj}	ΔR_{jj}	$m_{\ell\ell}$	$m_{\ell\ell}$
ΔR_{jj}	$m_{\ell\ell}$	$p_{T,\ell\ell}$	$p_{T,\ell\ell}$
$p_{T,\ell\ell}$	$ \cos(\vartheta_{CS}^{\ell\ell}) $	z^{ℓ_2}	z^{ℓ_2}
z^{ℓ_2}	m_T^{WW}	$ \cos(\vartheta_{CS}^{\ell\ell}) $	$ \cos(\vartheta_{CS}^{\ell\ell}) $
m_T^{WW}	$p_T^{j_2}$	$\Delta R_{\ell\ell}$	m_T^{WW}
$p_T^{j_2}$	$a_{T,\ell\ell}$	—	$\Delta\eta_{\ell\ell}$
$\Delta\eta_{\ell\ell}$	$\phi_{\text{recal}}^{\text{MET}}$	—	$p_T^{j_1}$
$a_{T,\ell\ell}$	—	—	η^{j_2}

5.5.6 Results

For each of the eight optimized sets of input variables presented in the previous section, an optimization consisting of a main optimization step with 3000 trials and a subsequent fine-tuning step with Population Based Training using 1000 trials was performed. All optimizations utilized the settings of the `TPESampler` found to be optimal in Sec. 5.5.2 and sampled the minimal reasonable hyperparameter search space derived in Sec. 5.5.3 corresponding to the chosen optimization target. As before, the values shown in Tab. 5.14 were set for hyperparameters not included in the search space. The relevant parts of the configuration file for the optimization using the binary cross-entropy optimization target and the set of input variables corresponding to the final iteration of the IVS using the mean relative change can be found in Appendix F.5. For the remaining optimizations, the `input_vars`, `monitor_name`, `monitor_op`

and `custom_metrics` options were adjusted accordingly.

Both steps of each optimization were evaluated using the corresponding five cross-validation models, i.e. a total of ten MLPs per optimization. For the main optimization step, those models got chose, whose hyperparameters were selected with the fit function evaluation method. For the reason explained before in Sec. 4.2.5, the cross-validation models of the evaluation method based on the best seen metric value were used for the PBT step. As before, the stacked output distributions for each MLP on its corresponding validation dataset was calculated and scaled by a factor of 5 to reach the same integrated luminosity as the full dataset. For each output distribution, a maximum likelihood fit was performed to extract the expected significance of the LL polarization component, following the fit strategy detailed in Sec. 5.2. The final result for each optimization step is given by the mean of the five significances while the standard deviation is used a measure of the uncertainty.

All obtained results are collected in Tab. 5.24. Overall, a performance improvement can be seen when using the manually built sets of input variables shown in Tab. 5.23 corresponding to the best iteration of the IVS (best IVS-iteration) over the automatically created counterparts in Tab 5.22 (last IVS-iteration). This improvement is especially evident for the optimizations using the LLR target, as already observed in the evolutions of the average relative changes (see Fig. 5.13) in the previous section. No conclusive evidence was found on a consistent advantage of either the mean or the median to calculate the average relative changes, however the best results for each of the two optimization targets were obtained using the best IVS-iteration with median relative changes. Interestingly, for all but these two optimizations, the MLPs obtained from the PBT step performed worse than the MLPs produced during the main optimization step. For the optimizations based on the best IVS-iteration with median relative changes however, the PBT models outperformed all other MLPs of the same optimization target.

The highest expected significance of the $W_L^\pm W_L^\pm jj$ -EW polarization component was obtained using the MLPs produced by the Population Based Training step of the optimization using the Asimov log-likelihood optimization target and the sets of input variables corresponding to the best iteration of the IVS with median relative changes. Since all hyperparameters but the number of hidden layers and number of neurons per hidden layer were varied over the course of the training, only the fixed values of `num_layers` = 5 and `units` = 98 optimized during the main optimization step are easily accessible⁵. Using these MLPs, an expected significance of 1.40 ± 0.05 standard deviations was achieved, corresponding to an 11% improvement over the results obtained with the final BDTs (see Sec. 5.4.4). The output distributions of the MLP corresponding to the $k = 0$ cross-validation splitting are presented in Fig. 5.14.

⁵The changes applied to each hyperparameter during the perturbation are saved in a `pbt_policy`-file for each trial to allow replaying a PBT run. However, a parser to convert this `policy`-file to a legible format has yet to be implemented.

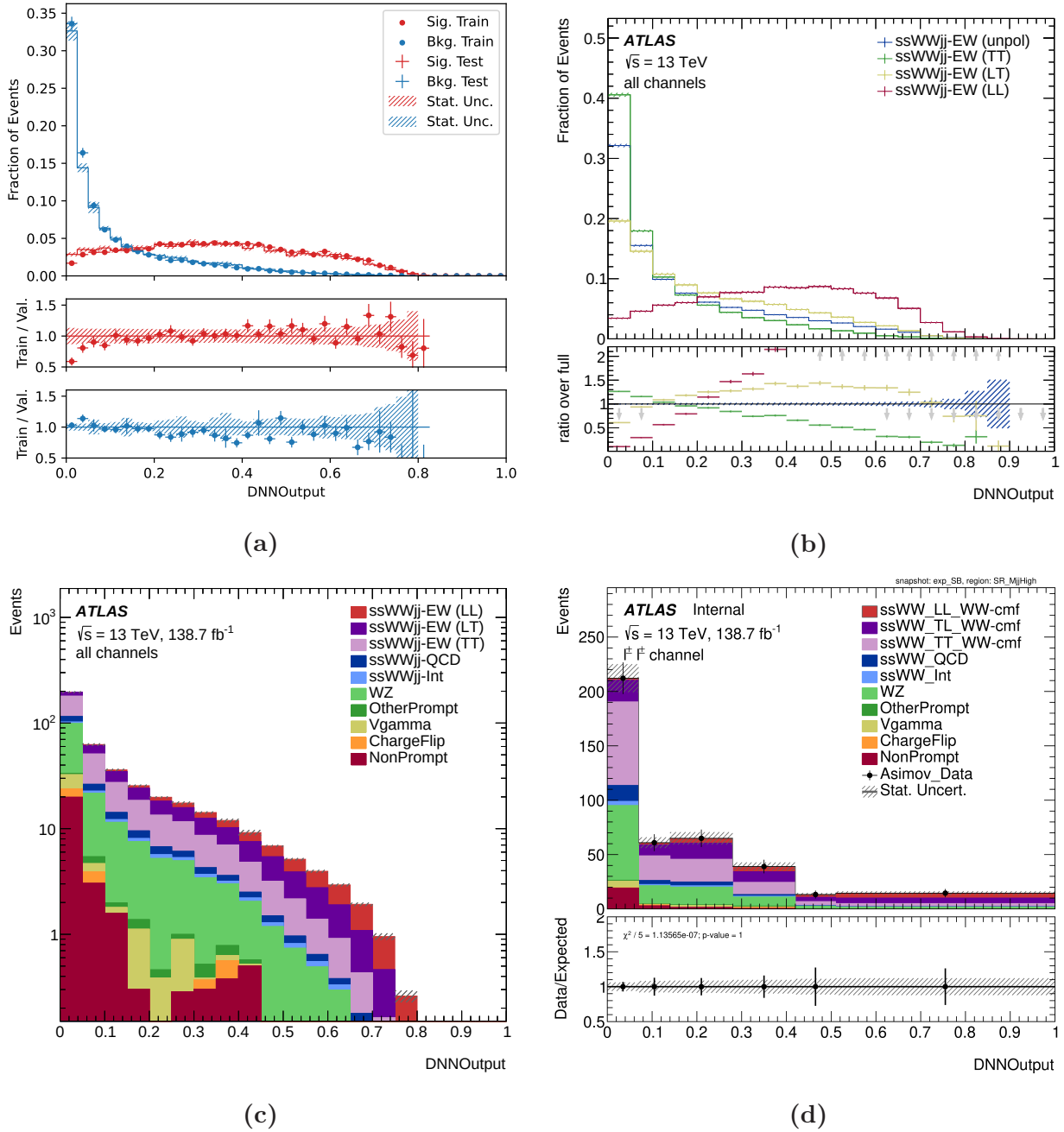


Figure 5.14: (a) Normalized signal and background distributions on the training and validation dataset, (b) the normalized output distributions for the three polarization components and (c) the stacked output distribution for all contributing processes using the full dataset and (d) the binning optimized stacked output distribution using the validation dataset in the $W^\pm W^\pm jj$ -EW signal region for the MLP obtained from the hyperparameter optimization with search space Nr. 3 (see Tab. 5.15) corresponding to the $k = 0$ cross-validation event splitting. For figure (a), the contribution of the LL polarization component of the $W^\pm W^\pm jj$ -EW process is used for the signal distribution while the background distribution is given by the sum of the remaining polarizations and background processes. Only statistical uncertainties are shown in all plots.

Table 5.24: Expected discovery significances of the $W_L^\pm W_L^\pm jj$ -EW polarization component using the MLPs obtained from different optimizations. The results are sorted according the metric used as the target of the optimization, the iteration of the input variable selection (IVS) the used set of input variables belongs to, the method used to calculate the average relative improvements during the IVS and the step of the optimization that produced the MLPs. The best obtained significances per optimization target are marked in bold.

Optimization target	IVS-iteration	Average rel. change	Step	exp. significance [σ]
Loss	Last	Mean	Main	1.340 ± 0.057
			PBT	1.329 ± 0.077
		Median	Main	1.373 ± 0.043
			PBT	1.363 ± 0.074
	Best	Mean	Main	1.370 ± 0.073
			PBT	1.318 ± 0.065
		Median	Main	1.364 ± 0.037
			PBT	1.389 ± 0.042
LLR	Last	Mean	Main	1.360 ± 0.013
			PBT	1.323 ± 0.083
		Median	Main	1.332 ± 0.080
			PBT	1.325 ± 0.074
	Best	Mean	Main	1.382 ± 0.040
			PBT	1.356 ± 0.082
		Median	Main	1.376 ± 0.070
			PBT	1.400 ± 0.048

5.6 Outlook for the High-Luminosity LHC

Even with the best performing neural networks obtained within this thesis, the expected discovery significance of the $W_L^\pm W_L^\pm jj$ -EW polarization component for Run 2 of the LHC is strongly limited by the low expected event yield. Given the 139 fb^{-1} of integrated luminosity recorded by the ATLAS detector during Run 2, the achieved significance of 1.40 ± 0.05 standard deviations with only statistical uncertainties unsurprisingly falls short of the evidence threshold of 3σ , let alone the 5σ required to claim observation.

For the High-Luminosity LHC that is currently planned to start data taking in 2029, an integrated luminosity of 3000 fb^{-1} is targeted [77, 78]. In order to investigate how much this increased integrated luminosity can improve the sensitivity of the measurement, the available dataset was scaled by a factor of $3000/139$. Other differences like the increased center of mass energy of 14 TeV, the higher average pile-up of 200 [77] or the planned upgrade of the ATLAS detector during LS3 are not considered.

In a previous study on the prospects of measuring the LL polarization component at the HL-LHC [79], an expected discovery significance of 1.8σ was obtained from a simultaneous binned maximum likelihood fit to the distribution of the azimuthal distance of the tagging jets $\Delta\Phi_{jj}$ in two m_{jj} regions. Their results for different integrated luminosities are shown in Fig. 5.15. Since no systematic uncertainties were included in the fits performed within this

thesis, the significance of 3σ with only statistical uncertainties for 3000 fb^{-1} shown in Fig. 5.15 is used for the following comparisons.

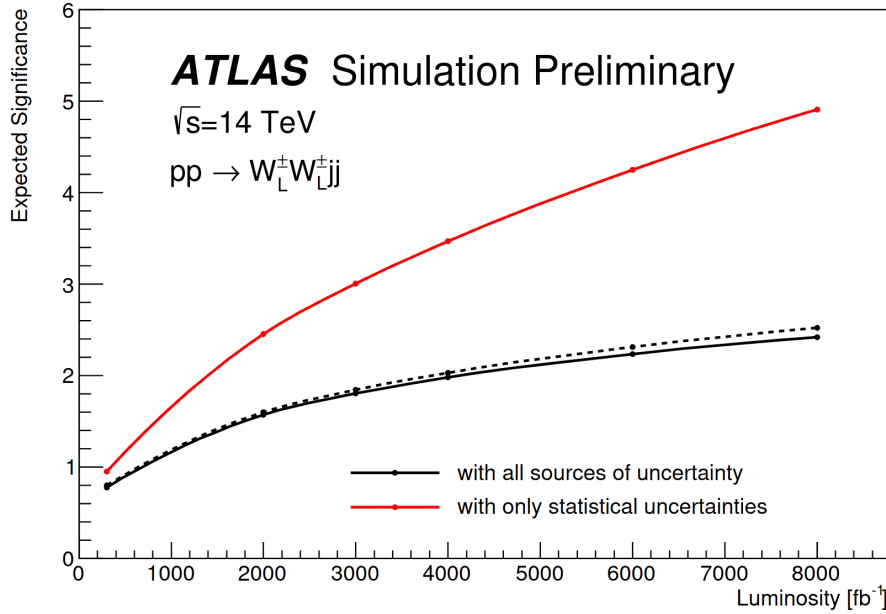


Figure 5.15: Expected discovery significance of the $W_L^\pm W_L^\pm jj$ -EW polarization component for different integrated luminosities obtained by the study performed in Ref. [79].

As a first comparison, a maximum likelihood fit to the $\Delta\Phi_{jj}$ distribution obtained using the scaled dataset was performed. Since no mention of a nuisance parameter for the normalization of the LT and TT polarization contributions is made in Ref. [79], it was assumed that they were fixed to the theory predictions. Thus, while applying the binning optimization detailed in Sec. 5.2, the unconstrained nuisance parameter controlling the sum of the polarization components was removed. From this fit, a significance of 4.97σ was obtained. It is assumed that this difference to the value seen in Fig. 5.15 is caused by the higher pile-up of 200 that was used for the Monte Carlo simulations in Ref. [79], resulting in a larger number of events containing charge misidentification and jets faking electrons. Due to the lower pile-up included in the Monte Carlo samples used in this thesis, the obtained significances are likely slight overestimations of the true sensitivity expected for the high pile-up environment of the HL-LHC. For this analysis, a more conservative fit procedure is used. By including the normalization of the polarization sum as an unconstrained nuisance parameter, the lack of knowledge of the polarization fractions is directly included in the fit. This approach was used for all fits presented in this thesis. Applying the same fit strategy to the scaled distribution of $\Delta\Phi_{jj}$ yields a significantly reduced significance of 1.49σ . Since in Sec. 5.2, $\Delta\Phi_{jj}$ was not found to be the most sensitive observable for the given fitting strategy, the same fit was also performed to the scaled distribution of m_{o1} (early-projected massless transverse inv. mass of the dilepton system and E_T^{miss}). As expected, the use of this observable resulted in a higher discovery significance of 3.19σ .

In order to assess the expected sensitivity of the measurement when using MLPs, the scaled output distributions of the final five cross-validation models presented in the previous section were calculated on their respective validation datasets. For each output distribution, a maximum likelihood fit was performed, following the same fitting strategy as before. From these fits, a mean expected significance of $(6.94 \pm 0.28)\sigma$ was obtained. This is in agreement with the ≈ 2 -fold sensitivity improvement over the most sensitive individual observable that was seen before.

Considering the presumed overestimation of the significance and the lack of systematic uncertainties, the 5σ observation level would likely be just out of reach of the current neural networks and fitting procedure. Still, the work presented in this thesis represents an important step towards the goal of observing the $W_L^\pm W_L^\pm jj$ -EW polarization component and, given the time that is available until the end of the HL-LHC, further development will likely be able to close the remaining gap. A few ideas on possible improvements will be presented in the next section.

6 Summary and Outlook

In this thesis, a framework for automated hyperparameter optimization and input variable selection of artificial neural networks is presented. Designed to support arbitrary Keras models for supervised learning tasks, this framework can be applied to a large range of problems in particle physics analyses. During the first of three consecutive optimization steps, the set of input variables is reduced with a backwards elimination approach. Using the reduced variable set, the neural network’s hyperparameters are optimized with a Tree Parzen Estimator based search algorithm in the second and fine-tuned with Population Based Training in the third optimization step. The flexible definition of metrics based on the model output and corresponding target labels allows the use of a variety of optimization targets to quantify the network’s performance. The use of overfitting conditions that are evaluated as part of the Early Stopping provides control over the generalization gap of the neural networks throughout all phases of the optimization. Built on top of the distributed computing library Ray, the optimization can naturally be extended across multiple computing nodes, making efficient use of available high-performance computing resources to improve parallelization and reduce turn-around times.

As part of the ATLAS study of polarization in the pair production of same-charged $W^\pm$ -bosons, the optimization framework is applied to binary event classification using multilayer perceptrons. The MLPs are to be used as the final classifier in the analysis and are trained on Monte Carlo simulated events to separate the longitudinal-longitudinal (LL) polarization component from the other polarizations and background processes. The available Monte Carlo simulations correspond to the full Run 2 dataset at a center of mass energy $\sqrt{s} = 13$ TeV and an integrated luminosity of 139 fb^{-1} .

Two metrics are compared as the target of the optimization: the binary cross-entropy loss and the Asimov log-likelihood ratio, both evaluated on the validation dataset. For each optimization target, the smallest reasonable hyperparameter search space is determined by performing multiple optimizations and iteratively including additional hyperparameters. Using these search spaces, the influence of the input variable selection is investigated. It is shown that, with the current implementation of the backwards elimination, more than half of the 30 available input variables can be removed before the performance of the neural network starts to degrade. Additionally, the benefit of a planned extension of the input variable selection is demonstrated by manually building the corresponding set of input variables. The overall

best performing MLPs are obtained using this manually created input variable set and the Asimov log-likelihood ratio as the optimization target. Maximum likelihood fits to Asimov data including only statistical uncertainties as well as the normalization of the sum of the three polarization components as an unconstrained nuisance parameter are performed. From these fits, an expected discovery significance of the LL polarization component of 1.40σ is obtained. The results are compared to the most sensitive single observable and a gradient-boosted decision tree, for which a significance of 0.69σ and 1.26σ , respectively, was obtained.

In a short study on the influence of reusing the training dataset for the maximum likelihood fit, clear mismodeling is observed, leading to a significant overestimation of the observed significance. It is shown that, while this mismodeling can be greatly reduced using overfitting conditions, the generalization performance of the MLPs is also significantly impaired.

In order to provide an outlook to the High-Luminosity LHC, the available dataset is scaled to an integrated luminosity of 3000 fb^{-1} . The increased center of mass energy $\sqrt{s} = 14 \text{ TeV}$ and the higher expected pileup are not considered. Applying the same fit strategy as before, an expected discovery significance of 6.94σ is obtained. Thus, considering the likely continued development until the end of the HL-LHC, the discovery of the LL polarization component seems probable.

One of the most promising improvements for future studies is the use of two MLPs for the event classification. With one MLP trained to separate $W^\pm W^\pm jj$ -EW from the background processes, and a second MLP trained to separate LL from the other polarization components, a two-dimensional histogram can be filled with the output-scores of both MLPs. Within the analysis group, this has been shown to greatly improve the constraint on the normalization of the polarization sum, and thus increase the significance, for the single-polarized case (LL+LT vs. TT). This approach is especially appealing in the light of the recent request to increase the size of the polarization datasets by a factor of ten, allowing the training of a much larger neural network for the classification of the polarizations. Additional improvements could be obtained by modifying the Asimov log-likelihood ratio to include the normalization of the polarization sum or, better still, by constructing a loss function to directly perform the gradient descent on the expected significance. Finally, changes to the neural network architecture are worth investigating. This could range from simple changes like allowing different number of neurons per hidden layer or using multiple output neurons to perform multiclass classification to the use of other types of neural networks. For example, Message Passing Graph Neural Networks have been shown to outperform multilayer perceptrons for event classification [80] and could also be tuned using the presented optimization framework.

A Mathematical definitions

Pauli matrices:

$$\sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (\text{A.1})$$

Gamma matrices (Dirac representation):

$$\gamma_0 = \begin{pmatrix} \mathbb{1}_{2 \times 2} & 0 \\ 0 & -\mathbb{1}_{2 \times 2} \end{pmatrix} \quad \gamma_i = \begin{pmatrix} 0 & \sigma_i \\ \sigma_i & 0 \end{pmatrix} \quad \gamma_5 = i\gamma_0\gamma_1\gamma_2\gamma_3 = \begin{pmatrix} 0 & \mathbb{1}_{2 \times 2} \\ \mathbb{1}_{2 \times 2} & 0 \end{pmatrix} \quad (\text{A.2})$$

B Monte Carlo samples

Table B.1: List of Monte Carlo samples used in this thesis

Process	DSID	p-tag
$W^\pm W^\pm$ – EW polarization		
MGPy8_ssWWjj_lep_EW6_LO	506202	p4252
MGPy8_ssWWjj_lep_TT_EW6_WWcmf_LO	506201	p4252
MGPy8_ssWWjj_lep_LL_EW6_WWcmf_LO	506204	p4252
MGPy8_ssWWjj_lep_TL_EW6_WWcmf_LO	506206	p4252
$W^\pm W^\pm$ – EW		
MGH7EG_LO_EW6_ssWWjj	500989	p4252
$W^\pm W^\pm$ – Int		
MGH7EG_LO_INT_ssWWjj	500991	p4252
$w^\pm w^\pm$ – QCD		
MGH7EG_LO_EW4_ssWWjj	500990	p4252
$W^\pm Z$		
MGPy8EG_NNPDF30NLO_A14NNPDF23LO_lvljjEW6_OFMinus	364739	p3975
MGPy8EG_NNPDF30NLO_A14NNPDF23LO_lvljjEW6_OFPlus	364740	p3975
MGPy8EG_NNPDF30NLO_A14NNPDF23LO_lvljjEW6_SFMinus	364741	p3975
MGPy8EG_NNPDF30NLO_A14NNPDF23LO_lvljjEW6_SFPlus	364742	p3975
Sherpa_222_NNPDF30NNLO_lllv	364253	p3975
OtherPrompt		
MadGraphPythia8EvtGen_A14_tZ_4fl_tchan_noAllHad	410560	p3975
PhPy8EG_A14_tchan_BW50_lept_top	410658	p4432
PhPy8EG_A14_tchan_BW50_lept_antitop	410659	p4432
PowhegPythia8EvtGen_A14_Wt_DR_dilepton_top	410648	p4252
PowhegPythia8EvtGen_A14_Wt_DR_dilepton_antitop	410649	p4252
PowhegPythia8EvtGen_A14_singletop_schan_lept_top	410644	p4434
PowhegPythia8EvtGen_A14_singletop_schan_lept_antitop	410645	p4434
aMcAtNloPythia8EvtGen_MEN30NLO_A14N23LO_ttee	410218	p3975
aMcAtNloPythia8EvtGen_MEN30NLO_A14N23LO_ttmumu	410219	p3975
aMcAtNloPythia8EvtGen_MEN30NLO_A14N23LO_tttatautau	410220	p3975
Vgamma		
Sh_2211_eegamma	700399	p4250
Sh_2211_enugamma	700403	p4250
Sh_2211_mumugamma	700398	p4250
Sh_2211_munugamma	700402	p4250
Sh_2211_tautaugamma	700400	p4250
Sh_2211_taubugamma	700404	p4250
ChargeFlip		
MGPy8EG_N30NLO_Zee_Ht0_70_CVetoBVeto	363147	p3975
MGPy8EG_N30NLO_Zee_Ht0_70_CFilterBVeto	363148	p3975
MGPy8EG_N30NLO_Zee_Ht0_70_BFilter	363149	p3975
MGPy8EG_N30NLO_Zee_Ht70_140_CVetoBVeto	363150	p3975
MGPy8EG_N30NLO_Zee_Ht70_140_CFilterBVeto	363151	p3975
MGPy8EG_N30NLO_Zee_Ht70_140_BFilter	363152	p3975
MGPy8EG_N30NLO_Zee_Ht140_280_CVetoBVeto	363153	p3975
MGPy8EG_N30NLO_Zee_Ht140_280_CFilterBVeto	363154	p3975
MGPy8EG_N30NLO_Zee_Ht140_280_BFilter	363155	p3975
MGPy8EG_N30NLO_Zee_Ht280_500_CVetoBVeto	363156	p3975
MGPy8EG_N30NLO_Zee_Ht280_500_CFilterBVeto	363157	p3975
MGPy8EG_N30NLO_Zee_Ht280_500_BFilter	363158	p3975
MGPy8EG_N30NLO_Zee_Ht500_700_CVetoBVeto	363159	p3975
MGPy8EG_N30NLO_Zee_Ht500_700_CFilterBVeto	363160	p3975
MGPy8EG_N30NLO_Zee_Ht500_700_BFilter	363161	p3975
MGPy8EG_N30NLO_Zee_Ht700_1000_CVetoBVeto	363162	p3975
MGPy8EG_N30NLO_Zee_Ht700_1000_CFilterBVeto	363163	p3975
MGPy8EG_N30NLO_Zee_Ht700_1000_BFilter	363164	p3975
MGPy8EG_N30NLO_Zee_Ht1000_2000_CVetoBVeto	363165	p3975
MGPy8EG_N30NLO_Zee_Ht1000_2000_CFilterBVeto	363166	p3975
MGPy8EG_N30NLO_Zee_Ht1000_2000_BFilter	363167	p3975
MGPy8EG_N30NLO_Zee_Ht2000_E_CMS_CVetoBVeto	363168	p3975
MGPy8EG_N30NLO_Zee_Ht2000_E_CMS_CFilterBVeto	363169	p3975
MGPy8EG_N30NLO_Zee_Ht2000_E_CMS_BFilter	363170	p3975

NonPrompt		
MGP _y 8EG_N30NLO_Wenu_Ht0_70_CVetoBVeto	363600	p3975
MGP _y 8EG_N30NLO_Wenu_Ht0_70_CFilterBVeto	363601	p3975
MGP _y 8EG_N30NLO_Wenu_Ht0_70_BFilter	363602	p3975
MGP _y 8EG_N30NLO_Wenu_Ht70_140_CVetoBVeto	363603	p3975
MGP _y 8EG_N30NLO_Wenu_Ht70_140_CFilterBVeto	363604	p3975
MGP _y 8EG_N30NLO_Wenu_Ht70_140_BFilter	363605	p3975
MGP _y 8EG_N30NLO_Wenu_Ht140_280_CVetoBVeto	363606	p3975
MGP _y 8EG_N30NLO_Wenu_Ht140_280_CFilterBVeto	363607	p3975
MGP _y 8EG_N30NLO_Wenu_Ht140_280_BFilter	363608	p3975
MGP _y 8EG_N30NLO_Wenu_Ht280_500_CVetoBVeto	363609	p3975
MGP _y 8EG_N30NLO_Wenu_Ht280_500_CFilterBVeto	363610	p3975
MGP _y 8EG_N30NLO_Wenu_Ht280_500_BFilter	363611	p3975
MGP _y 8EG_N30NLO_Wenu_Ht500_700_CVetoBVeto	363612	p3975
MGP _y 8EG_N30NLO_Wenu_Ht500_700_CFilterBVeto	363613	p3975
MGP _y 8EG_N30NLO_Wenu_Ht500_700_BFilter	363614	p3975
MGP _y 8EG_N30NLO_Wenu_Ht700_1000_CVetoBVeto	363615	p3975
MGP _y 8EG_N30NLO_Wenu_Ht700_1000_CFilterBVeto	363616	p3975
MGP _y 8EG_N30NLO_Wenu_Ht700_1000_BFilter	363617	p3975
MGP _y 8EG_N30NLO_Wenu_Ht1000_2000_CVetoBVeto	363618	p3975
MGP _y 8EG_N30NLO_Wenu_Ht1000_2000_CFilterBVeto	363619	p3975
MGP _y 8EG_N30NLO_Wenu_Ht1000_2000_BFilter	363620	p3975
MGP _y 8EG_N30NLO_Wenu_Ht2000_E_CMS_CVetoBVeto	363621	p3975
MGP _y 8EG_N30NLO_Wenu_Ht2000_E_CMS_CFilterBVeto	363622	p3975
MGP _y 8EG_N30NLO_Wenu_Ht2000_E_CMS_BFilter	363623	p3975
MGP _y 8EG_N30NLO_Wmunu_Ht0_70_CVetoBVeto	363624	p3975
MGP _y 8EG_N30NLO_Wmunu_Ht0_70_CFilterBVeto	363625	p3975
MGP _y 8EG_N30NLO_Wmunu_Ht0_70_BFilter	363626	p3975
MGP _y 8EG_N30NLO_Wmunu_Ht70_140_CVetoBVeto	363627	p3975
MGP _y 8EG_N30NLO_Wmunu_Ht70_140_CFilterBVeto	363628	p3975
MGP _y 8EG_N30NLO_Wmunu_Ht70_140_BFilter	363629	p3975
MGP _y 8EG_N30NLO_Wmunu_Ht140_280_CVetoBVeto	363630	p3975
MGP _y 8EG_N30NLO_Wmunu_Ht140_280_CFilterBVeto	363631	p3975
MGP _y 8EG_N30NLO_Wmunu_Ht140_280_BFilter	363632	p3975
MGP _y 8EG_N30NLO_Wmunu_Ht280_500_CVetoBVeto	363633	p3975
MGP _y 8EG_N30NLO_Wmunu_Ht280_500_CFilterBVeto	363634	p3975
MGP _y 8EG_N30NLO_Wmunu_Ht280_500_BFilter	363635	p3975
MGP _y 8EG_N30NLO_Wmunu_Ht500_700_CVetoBVeto	363636	p3975
MGP _y 8EG_N30NLO_Wmunu_Ht500_700_CFilterBVeto	363637	p3975
MGP _y 8EG_N30NLO_Wmunu_Ht500_700_BFilter	363638	p3975
MGP _y 8EG_N30NLO_Wmunu_Ht700_1000_CVetoBVeto	363639	p3975
MGP _y 8EG_N30NLO_Wmunu_Ht700_1000_CFilterBVeto	363640	p3975
MGP _y 8EG_N30NLO_Wmunu_Ht700_1000_BFilter	363641	p3975
MGP _y 8EG_N30NLO_Wmunu_Ht1000_2000_CVetoBVeto	363642	p3975
MGP _y 8EG_N30NLO_Wmunu_Ht1000_2000_CFilterBVeto	363643	p3975
MGP _y 8EG_N30NLO_Wmunu_Ht1000_2000_BFilter	363644	p3975
MGP _y 8EG_N30NLO_Wmunu_Ht2000_E_CMS_CVetoBVeto	363645	p3975
MGP _y 8EG_N30NLO_Wmunu_Ht2000_E_CMS_CFilterBVeto	363646	p3975
MGP _y 8EG_N30NLO_Wmunu_Ht2000_E_CMS_BFilter	363647	p3975
MGP _y 8EG_N30NLO_Wtaunu_Ht0_70_CVetoBVeto	363648	p3975
MGP _y 8EG_N30NLO_Wtaunu_Ht0_70_CFilterBVeto	363649	p3975
MGP _y 8EG_N30NLO_Wtaunu_Ht0_70_BFilter	363650	p3975
MGP _y 8EG_N30NLO_Wtaunu_Ht70_140_CVetoBVeto	363651	p3975
MGP _y 8EG_N30NLO_Wtaunu_Ht70_140_CFilterBVeto	363652	p3975
MGP _y 8EG_N30NLO_Wtaunu_Ht70_140_BFilter	363653	p3975
MGP _y 8EG_N30NLO_Wtaunu_Ht140_280_CVetoBVeto	363654	p3975
MGP _y 8EG_N30NLO_Wtaunu_Ht140_280_CFilterBVeto	363655	p3975
MGP _y 8EG_N30NLO_Wtaunu_Ht140_280_BFilter	363656	p3975
MGP _y 8EG_N30NLO_Wtaunu_Ht280_500_CVetoBVeto	363657	p3975
MGP _y 8EG_N30NLO_Wtaunu_Ht280_500_CFilterBVeto	363658	p3975
MGP _y 8EG_N30NLO_Wtaunu_Ht280_500_BFilter	363659	p3975
MGP _y 8EG_N30NLO_Wtaunu_Ht500_700_CVetoBVeto	363660	p3975
MGP _y 8EG_N30NLO_Wtaunu_Ht500_700_CFilterBVeto	363661	p3975
MGP _y 8EG_N30NLO_Wtaunu_Ht500_700_BFilter	363662	p3975
MGP _y 8EG_N30NLO_Wtaunu_Ht700_1000_CVetoBVeto	363663	p3975
MGP _y 8EG_N30NLO_Wtaunu_Ht700_1000_CFilterBVeto	363664	p3975
MGP _y 8EG_N30NLO_Wtaunu_Ht700_1000_BFilter	363665	p3975
MGP _y 8EG_N30NLO_Wtaunu_Ht1000_2000_CVetoBVeto	363666	p3975
MGP _y 8EG_N30NLO_Wtaunu_Ht1000_2000_CFilterBVeto	363667	p3975
MGP _y 8EG_N30NLO_Wtaunu_Ht1000_2000_BFilter	363668	p3975
MGP _y 8EG_N30NLO_Wtaunu_Ht2000_E_CMS_CVetoBVeto	363669	p3975
MGP _y 8EG_N30NLO_Wtaunu_Ht2000_E_CMS_CFilterBVeto	363670	p3975
MGP _y 8EG_N30NLO_Wtaunu_Ht2000_E_CMS_BFilter	363671	p3975
PhPy8EG_A14_ttbar_hdamp258p75_nonallhad	410470	p3975
PhPy8EG_A14_ttbar_hdamp258p75_allhad	410471	p3975

C Comparison plots for the full, unpolarized and polarized samples

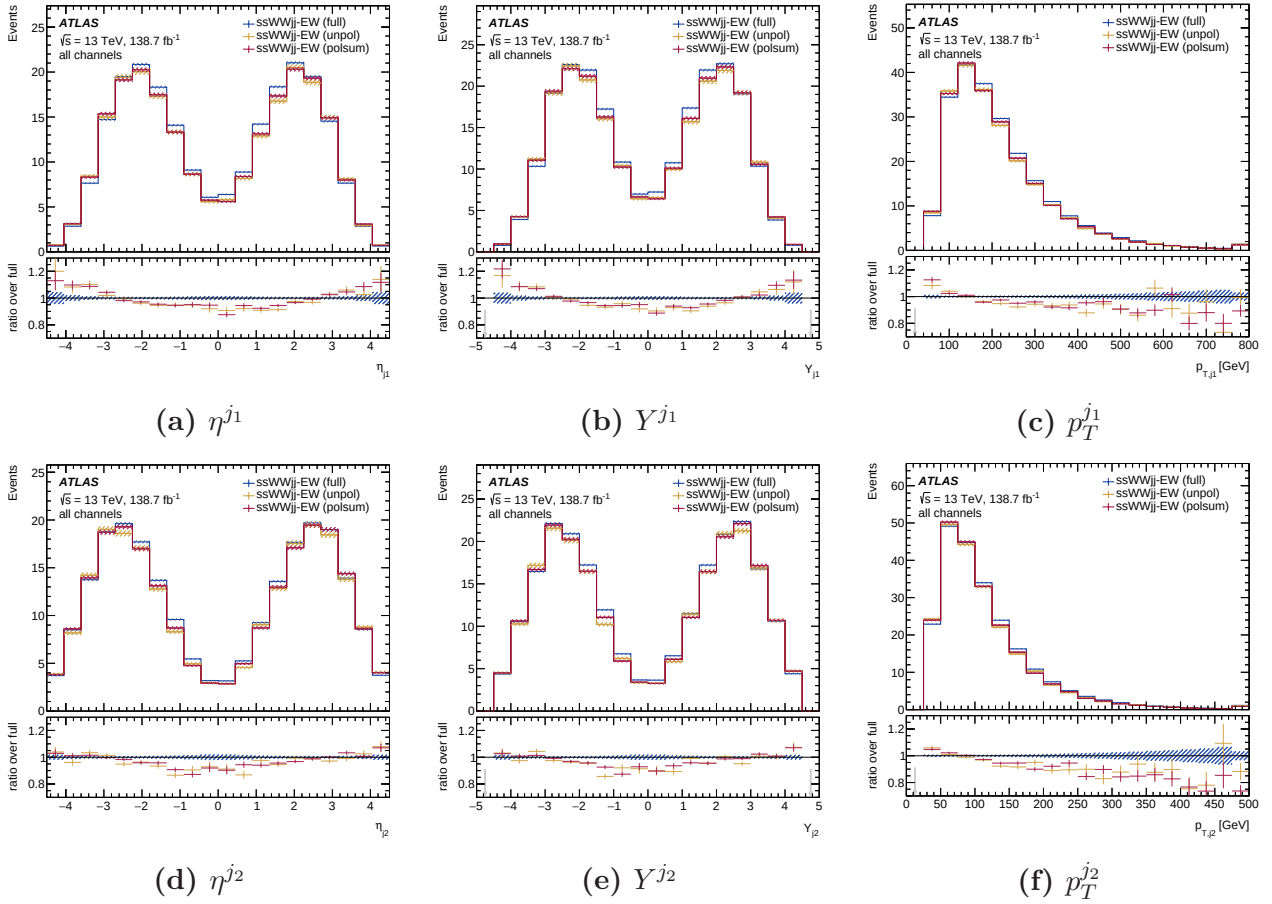


Figure C.1: Distributions of the pseudo-rapidity η , the rapidity Y and the transverse momentum p_T of the leading (a-c) and subleading (d-f) jet for the full, unpolarized and polarized samples.

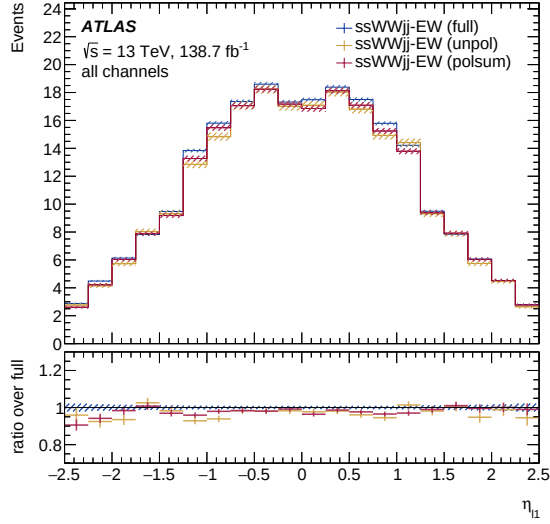
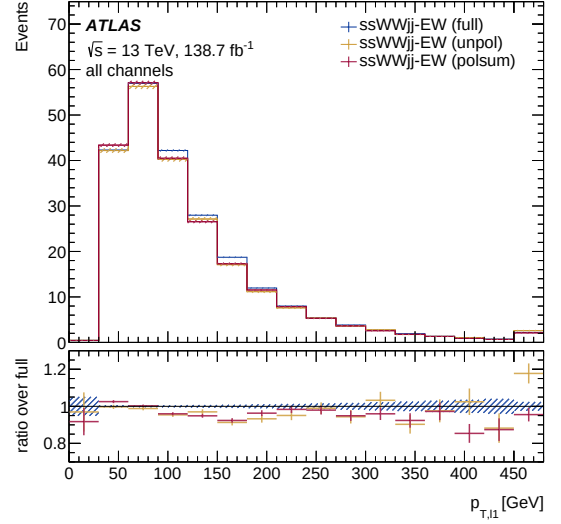
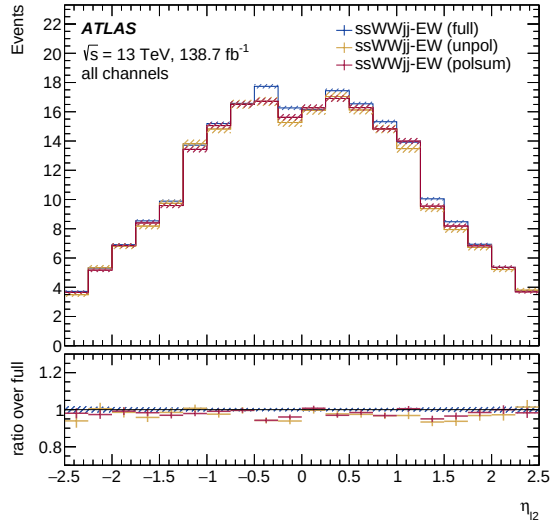
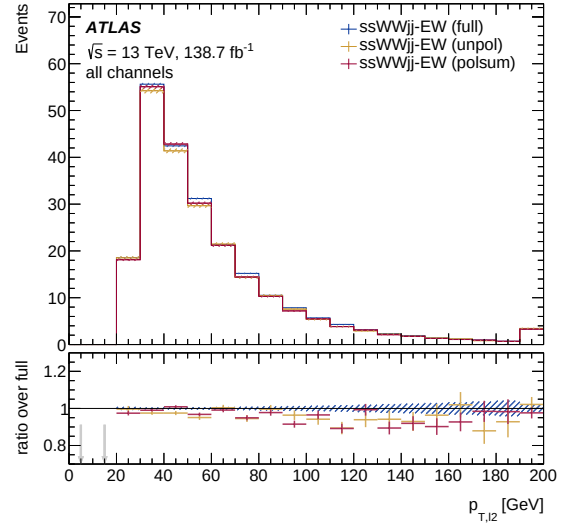
(a) η^{ℓ_1} (b) $p_T^{\ell_1}$ (c) η^{ℓ_2} (d) $p_T^{\ell_2}$

Figure C.2: Distributions of the pseudo-rapidity η and transverse momentum p_T of the leading (a-b) and subleading (c-d) lepton for the full, unpolarized and polarized samples.

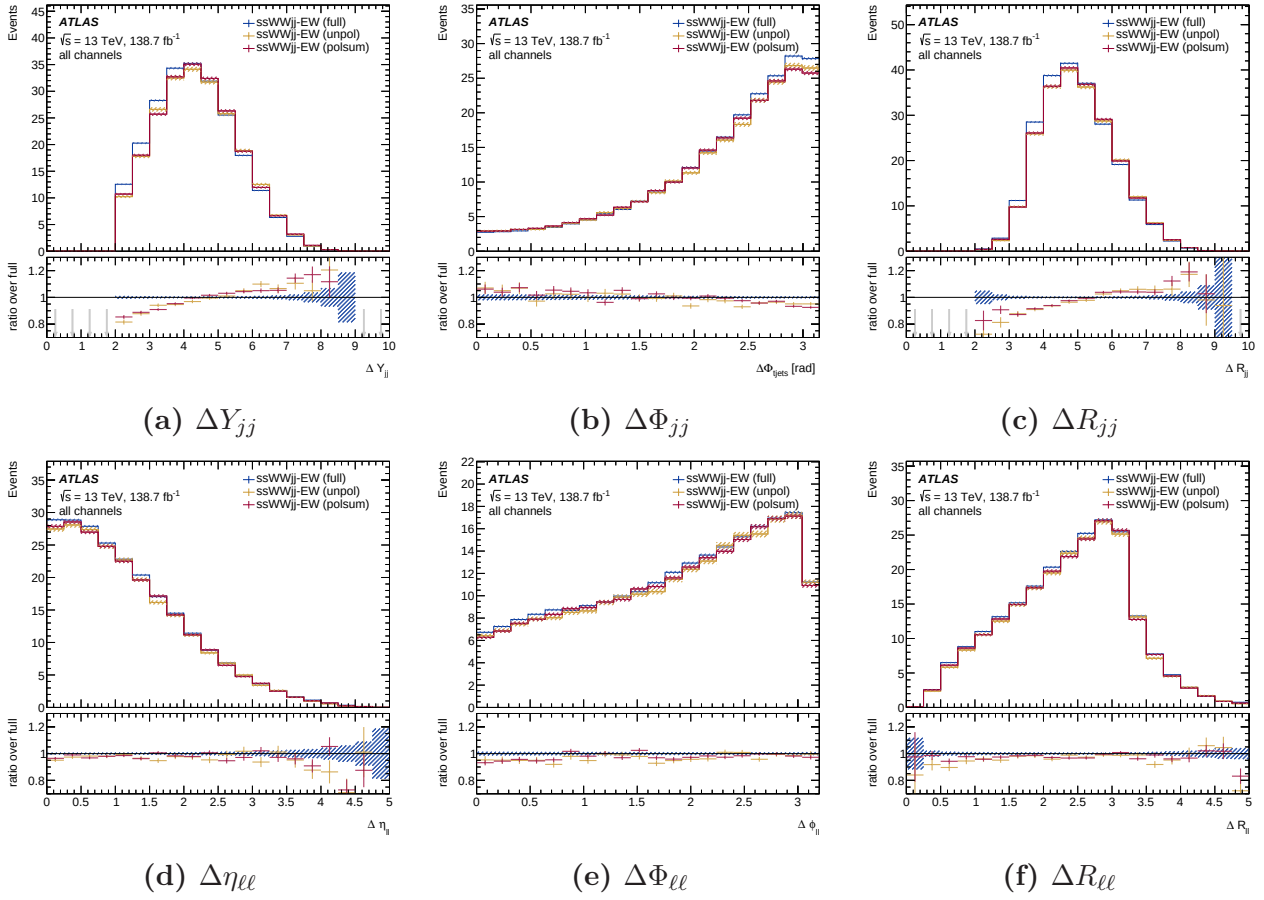


Figure C.3: Distributions of the rapidity separation ΔY , azimuthal distance $\Delta \Phi$ and angular distance ΔR of the tagging jets (a-c) and pseudo-rapidity separation $\Delta \eta$, azimuthal distance $\Delta \Phi$ and angular distance ΔR of the two leptons (d-f) for the full, unpolarized and polarized samples.

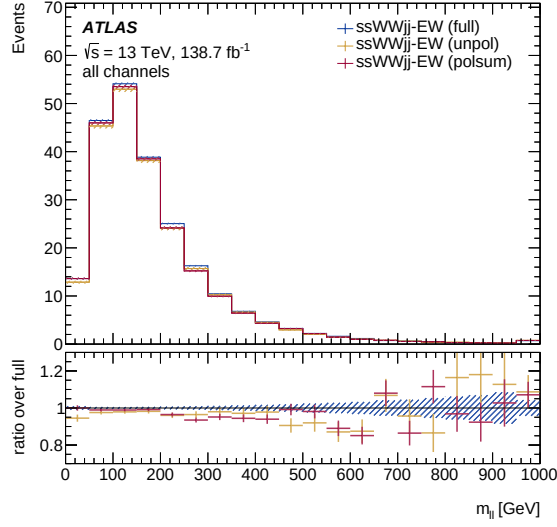
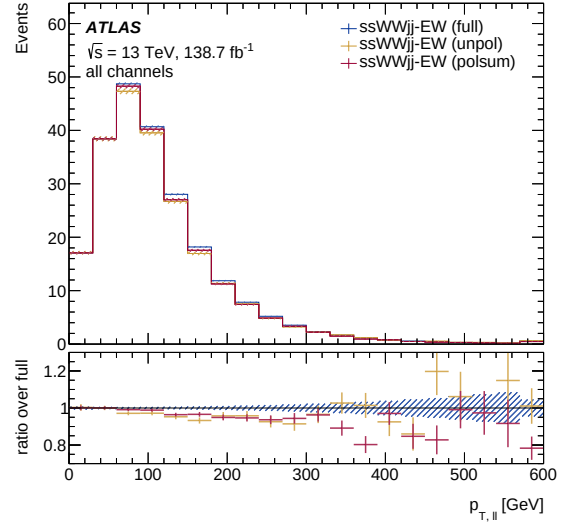
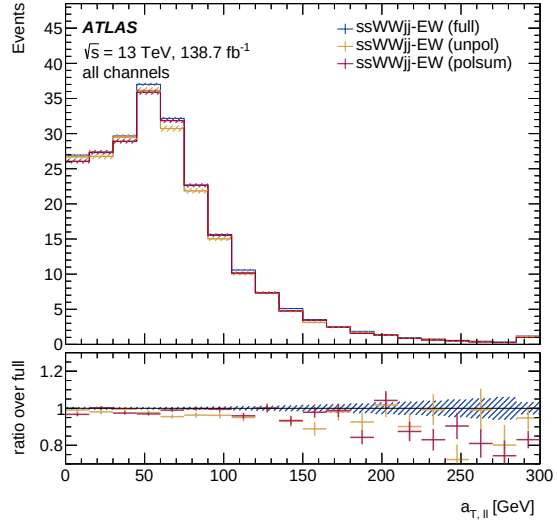
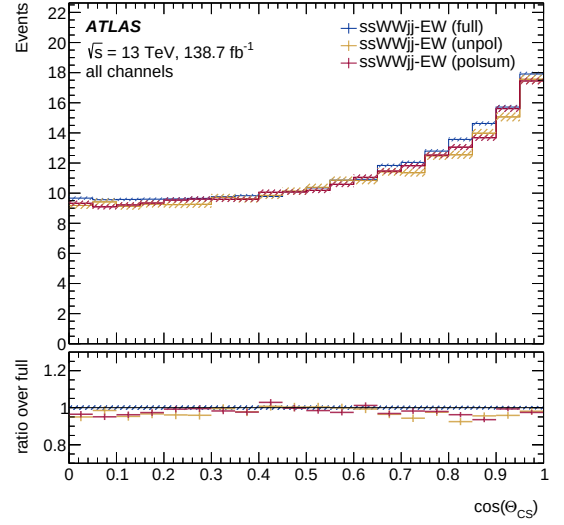
(a) $m_{\ell\ell}$ (b) $p_{T,\ell\ell}$ (c) $a_{T,\ell\ell}$ (d) $|\cos(\vartheta_{CS}^{\ell\ell})|$

Figure C.4: Distributions of (a) the invariant mass of the dilepton system, (b) the transverse momentum of the dilepton system, (c) transverse momentum of the dilepton system relative to its thrust axis and (d) the absolute value of the cosine of the angle between the two leptons in the Collins-Soper reference frame for the full, unpolarized and polarized samples.

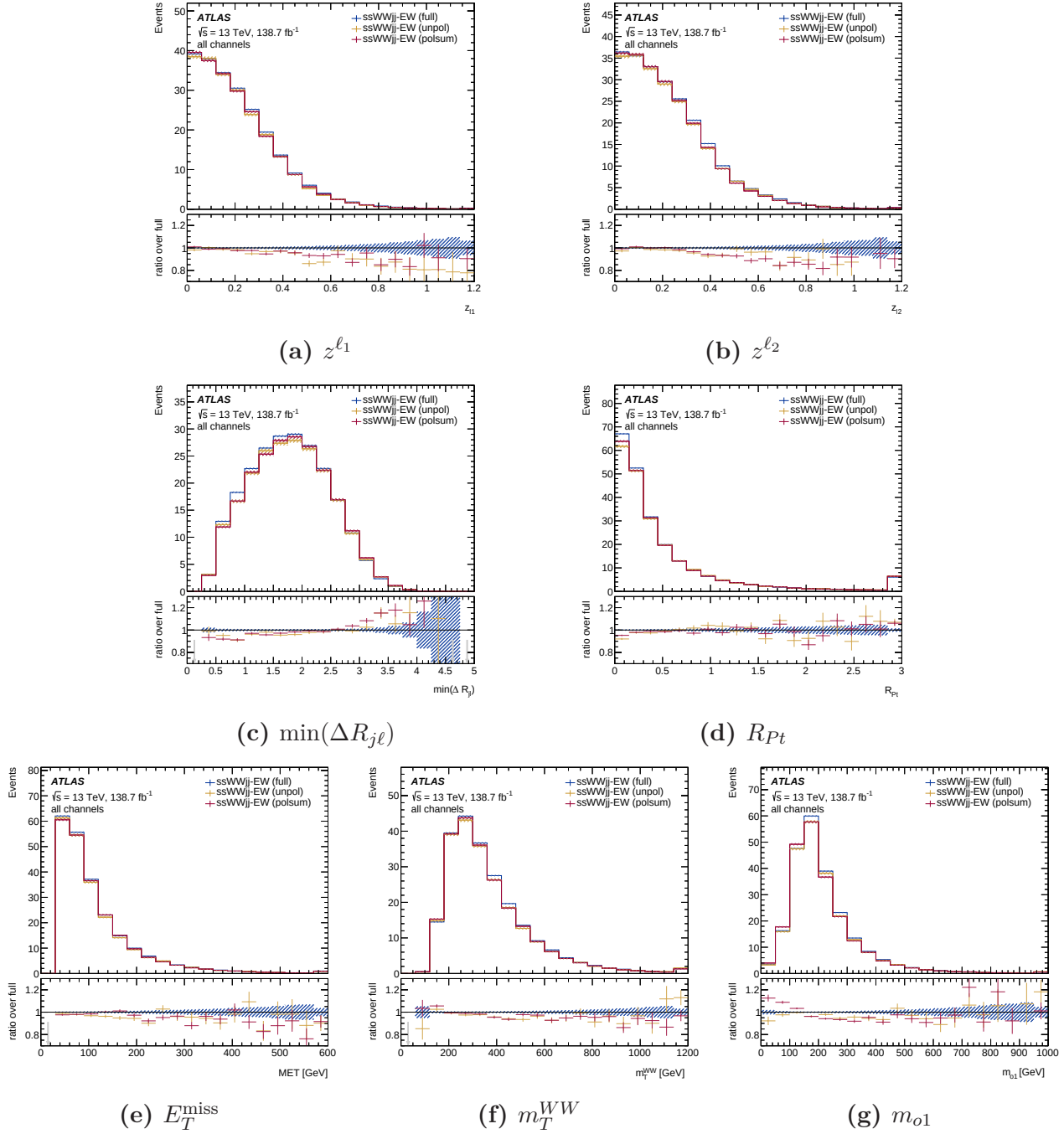


Figure C.5: Distributions of the Zeppenfeld variable of the leading (a) and subleading (b) lepton, (c) the minimal angular distance between the leptons and the tagging jets, (d) the ratio of the products of the transverse momenta of the leptons and the tagging jets, (e) the missing transverse energy, (f) the transverse mass of the dilepton system and the missing transverse energy and (g) the early-projected massless transverse invariant mass of the dilepton system and the missing transverse energy for the full, unpolarized and polarized samples.

D Distributions of polarization sensitive observables

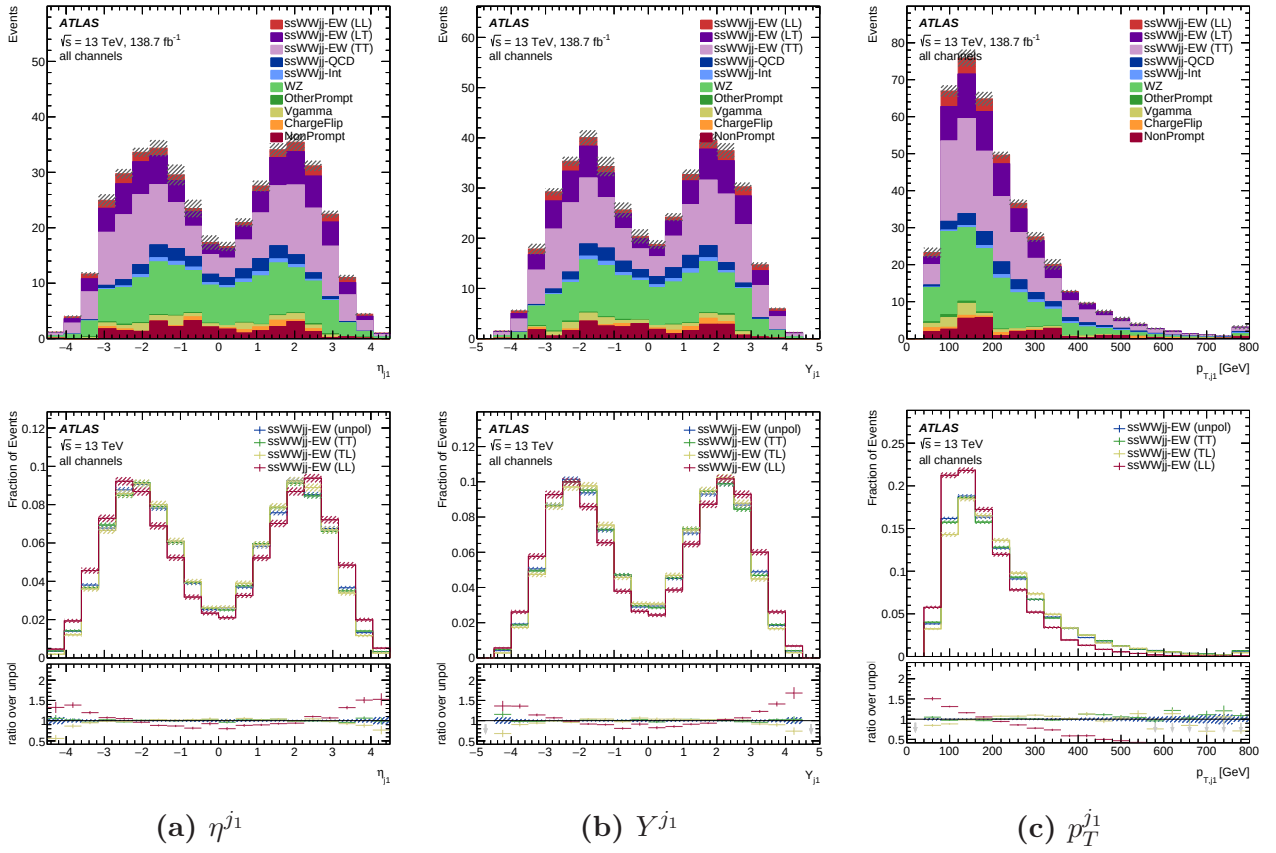


Figure D.1: Distributions of (a) the pseudo-rapidity, (b) the rapidity and (c) the transverse momentum of the leading jet for different polarization components of the $W^\pm W^\pm jj$ -EW process.

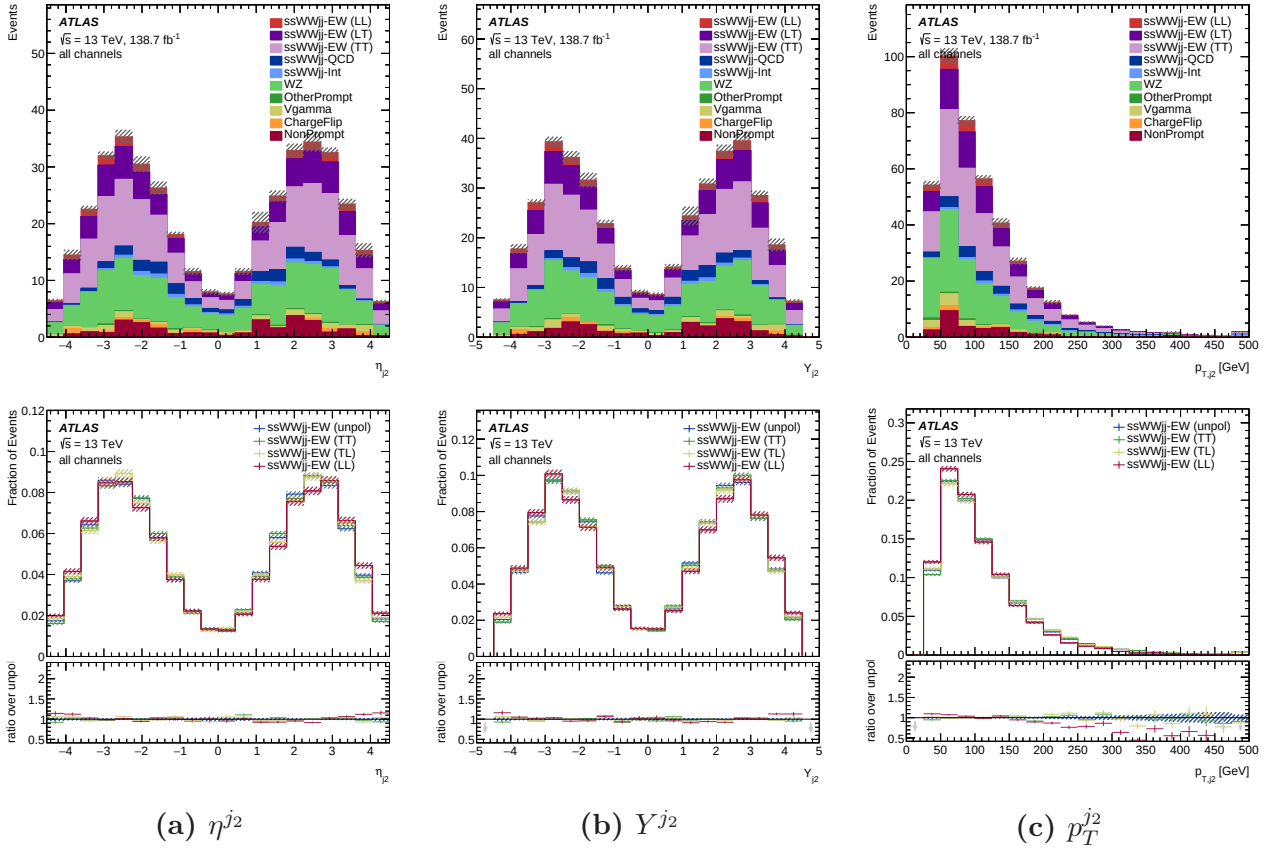


Figure D.2: Distributions of (a) the pseudo-rapidity, (b) the rapidity and (c) the transverse momentum of the subleading jet for different polarization components of the $W^\pm W^\pm jj$ -EW process.

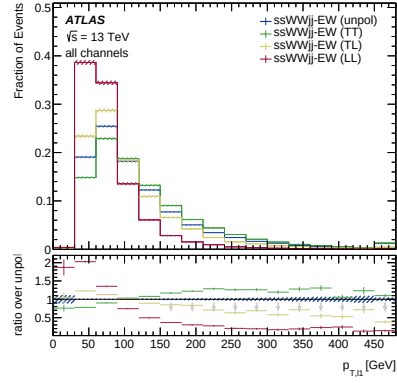
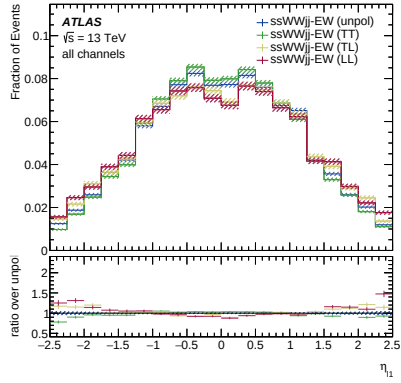
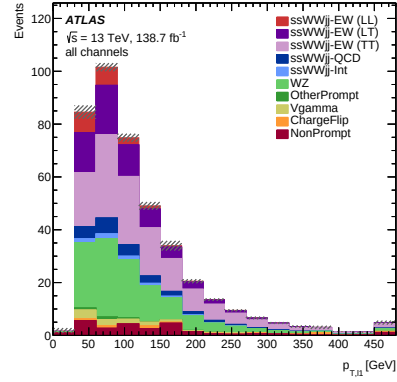
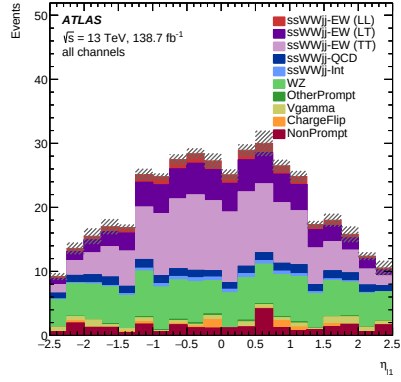
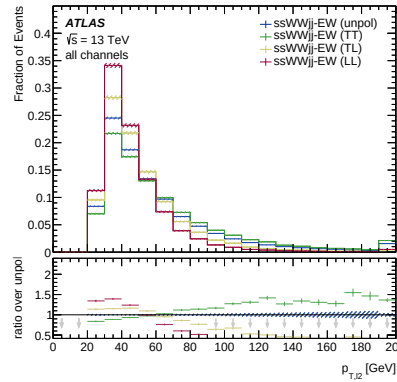
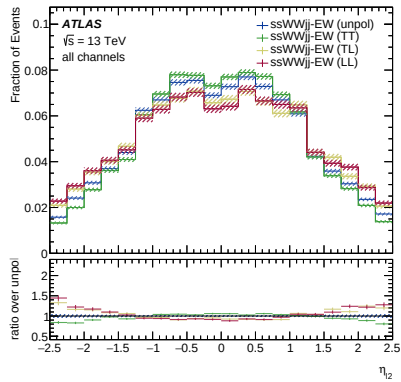
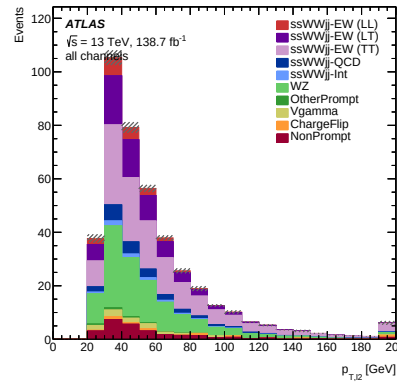
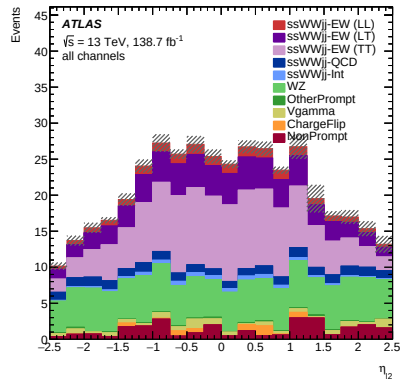
(a) η^{ℓ_1} (b) $p_T^{\ell_1}$ (c) η^{ℓ_2} (d) $p_T^{\ell_2}$

Figure D.3: Distributions of the pseudo-rapidity η and the transverse momentum p_T of the leading (a-b) and subleading (c-d) lepton for different polarization components of the $W^\pm W^\pm jj$ -EW process.

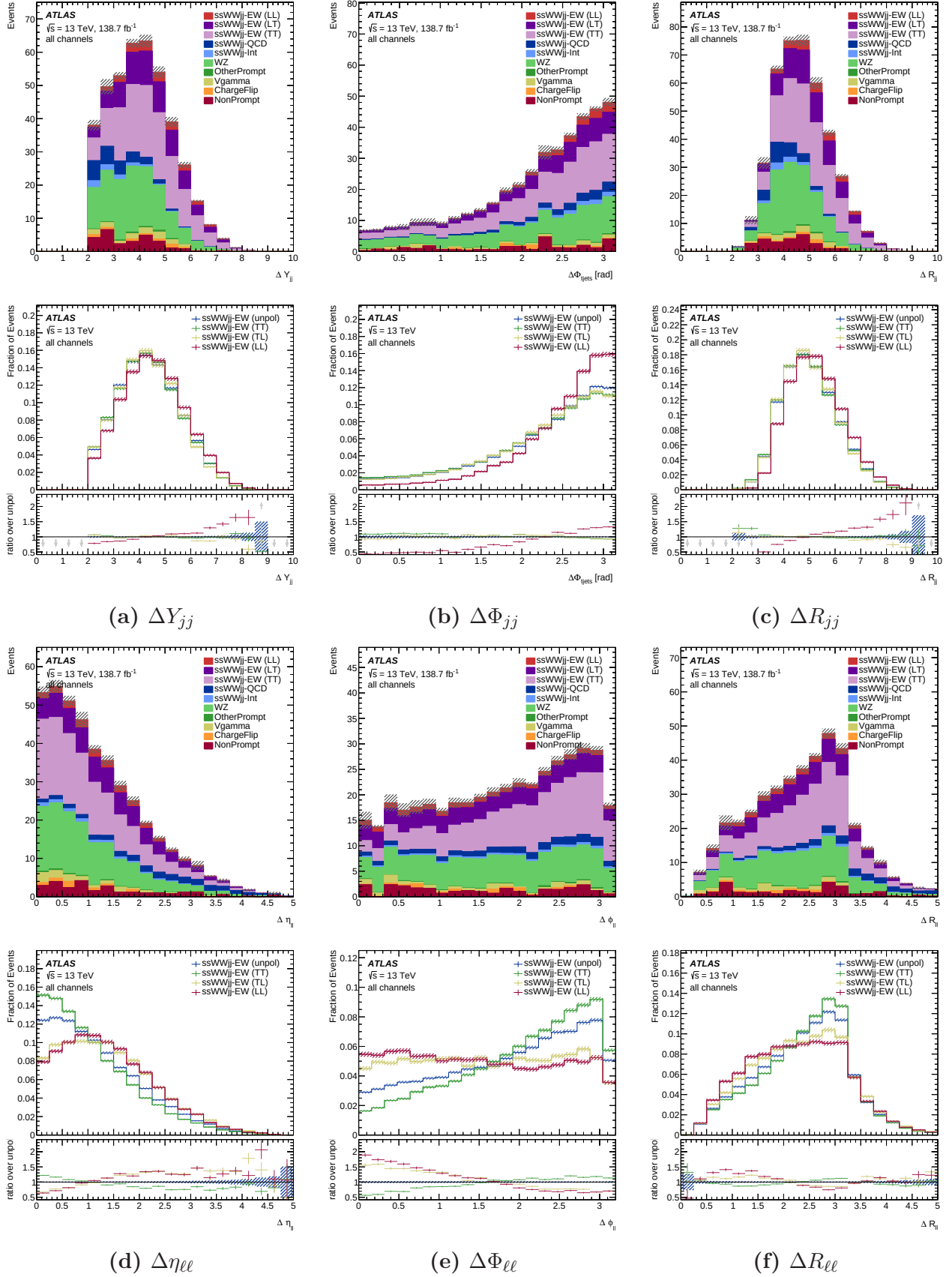


Figure D.4: Distributions of the rapidity separation ΔY , azimuthal distance $\Delta \Phi$ and angular distance ΔR of the tagging jets (a-c) and pseudo-rapidity separation $\Delta \eta$, azimuthal distance $\Delta \Phi$ and angular distance ΔR of the two leptons (d-f) for different polarization components of the $W^\pm W^\pm jj$ -EW process.

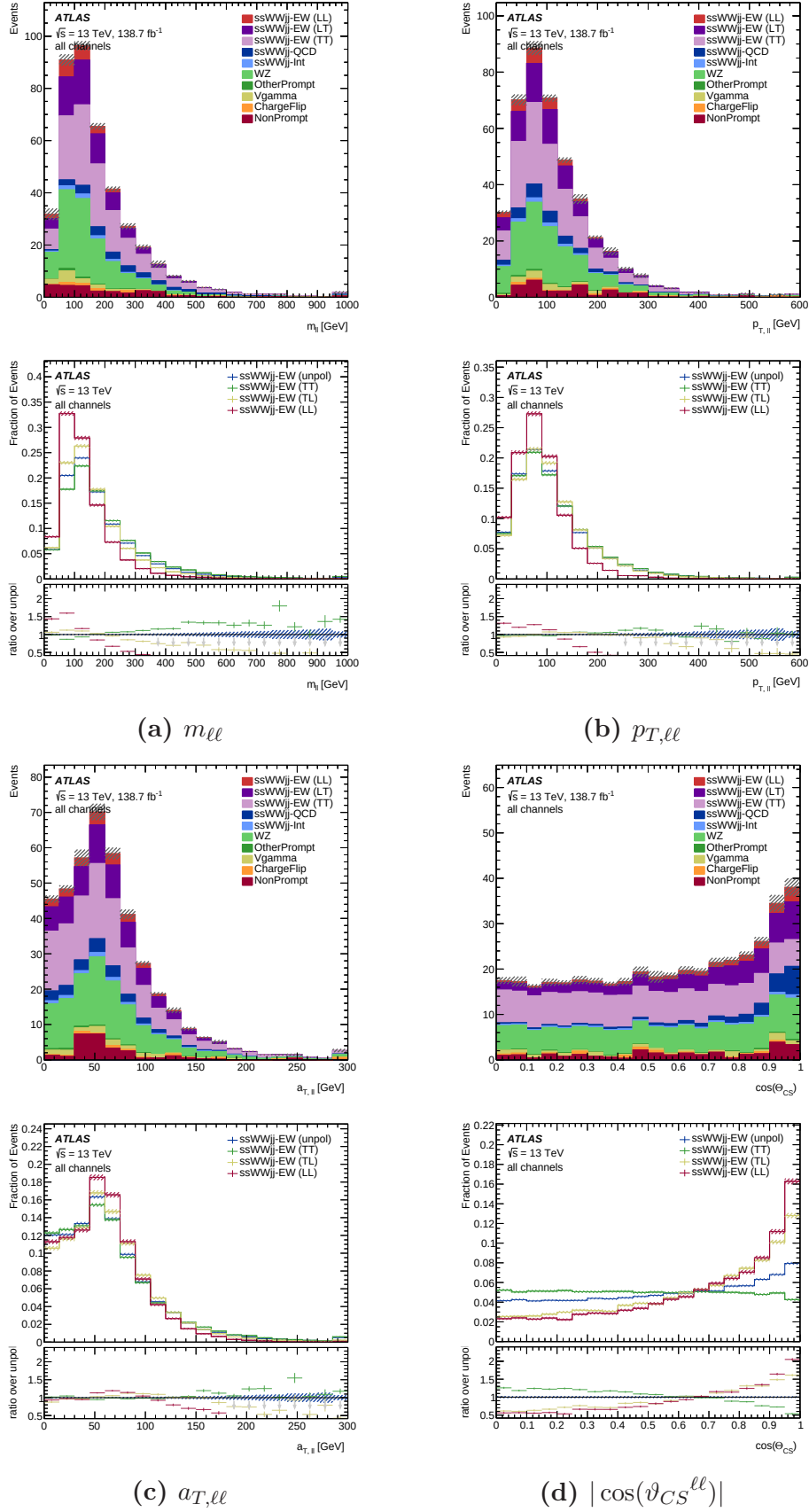


Figure D.5: Distributions of (a) the invariant mass of the dilepton system, (b) the transverse momentum of the dilepton system, (c) transverse momentum of the dilepton system relative to its thrust axis and (d) the absolute value of the cosine of the angle between the two leptons in the Collins-Soper reference frame for different polarization components of the $W^\pm W^\pm jj$ -EW process.

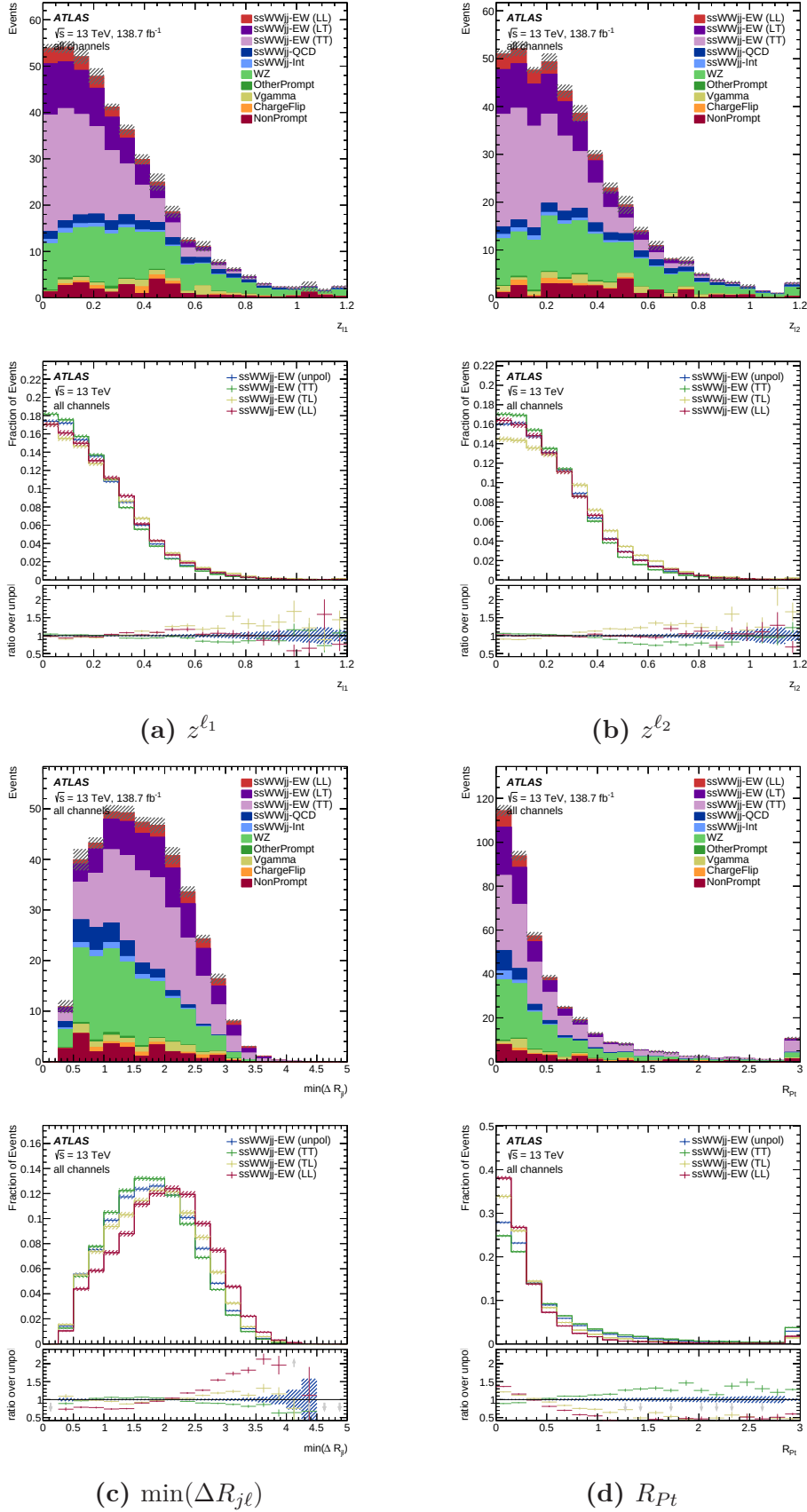


Figure D.6: Distributions of the Zeppenfeld variable of the leading (a) and subleading (b) lepton, (c) the minimal angular distance between the leptons and the tagging jets and (d) the ratio of the products of the transverse momenta of the leptons and the tagging jets for different polarization components of the $W^\pm W^\pm jj$ -EW process.

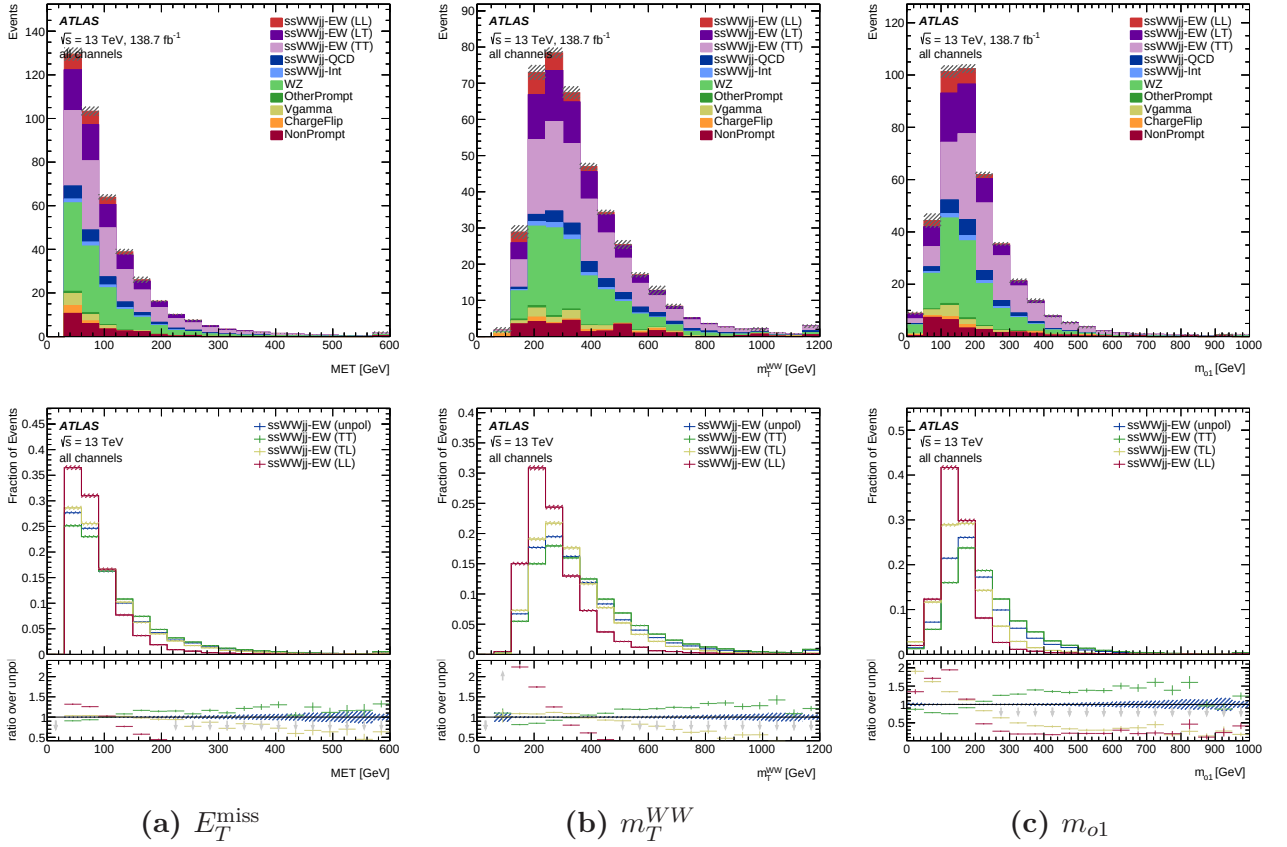


Figure D.7: Distributions of (a) the missing transverse energy, (b) the transverse mass of the dilepton system and the missing transverse energy and (c) the early-projected massless transverse invariant mass of the dilepton system and the missing transverse energy for different polarization components of the $W^\pm W^\pm jj$ -EW process.

E Preprocessed signal and background distributions of BDT input variables

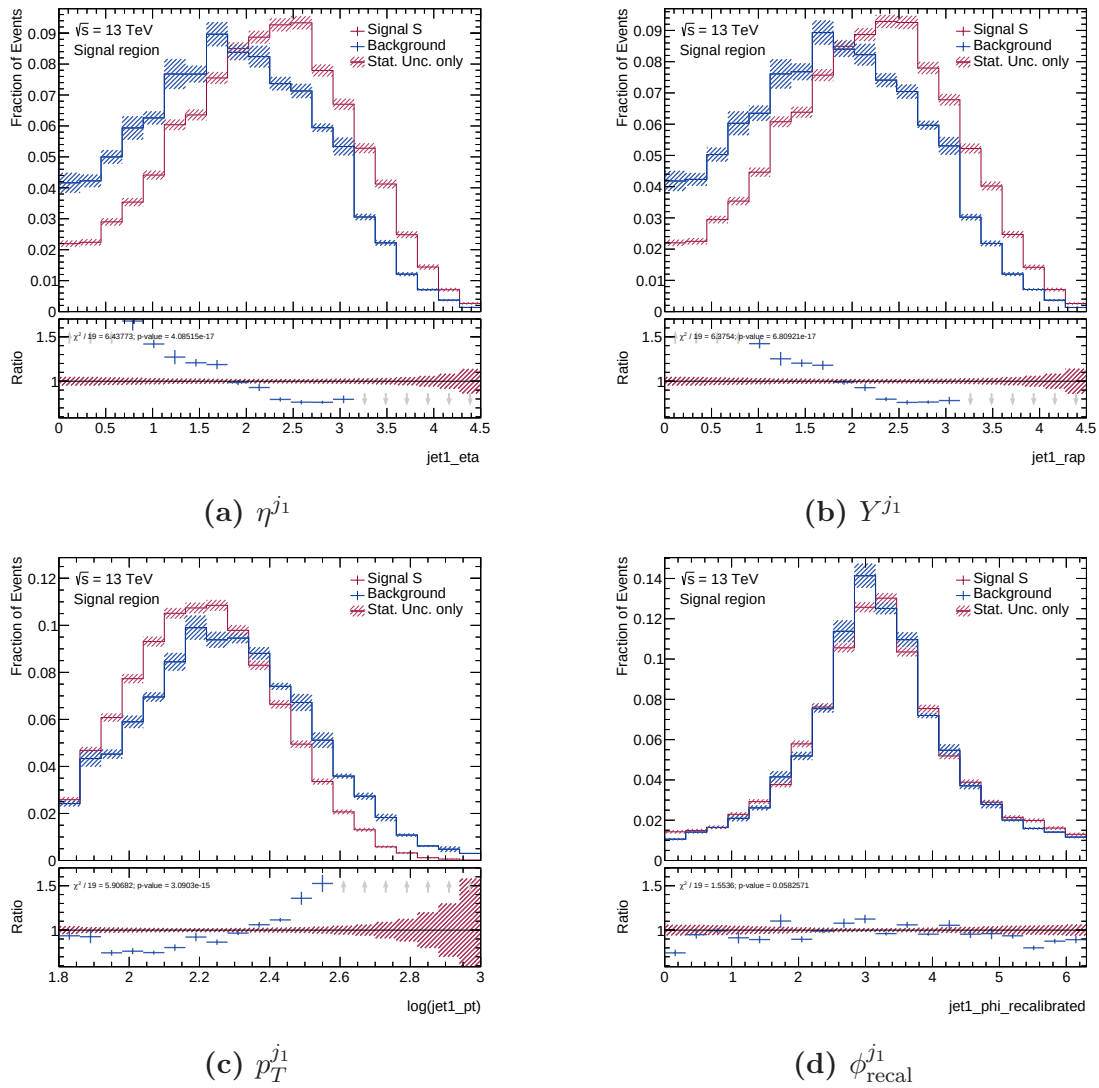


Figure E.1: Preprocessed signal and background distributions of (a) the pseudo-rapidity, (b) the rapidity Y , (c) the transverse momentum p_T and (d) the redefined azimuth angle of the leading jet.

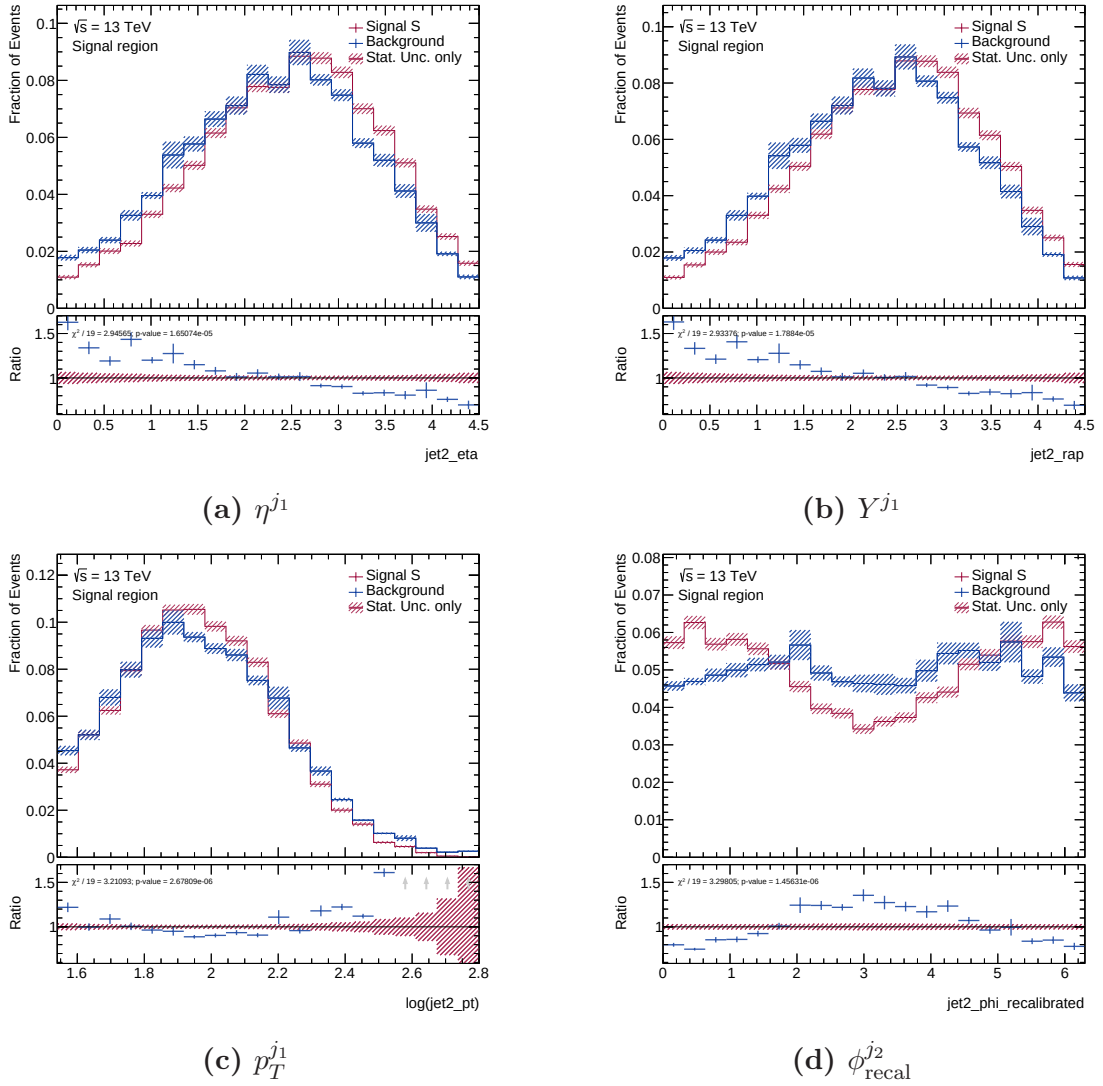


Figure E.2: Preprocessed signal and background distributions of (a) the pseudo-rapidity, (b) the rapidity Y , (c) the transverse momentum p_T and (d) the redefined azimuth angle of the subleading jet.

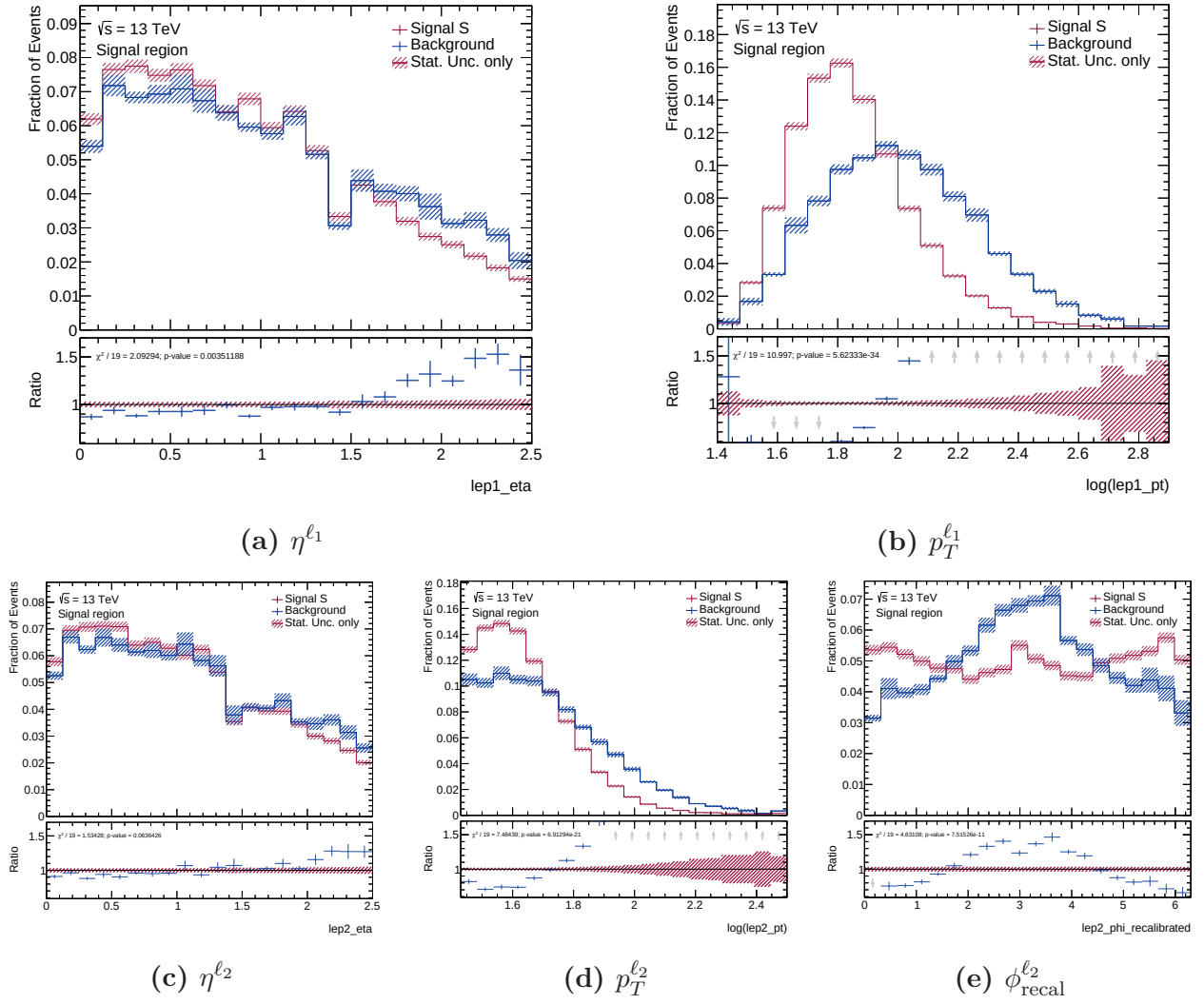


Figure E.3: Preprocessed signal and background distributions of (a) the pseudo-rapidity and (b) the transverse momentum of the leading lepton and (c) the pseudo-rapidity, (d) the transverse momentum and (e) the redefined azimuth angle of the subleading lepton.

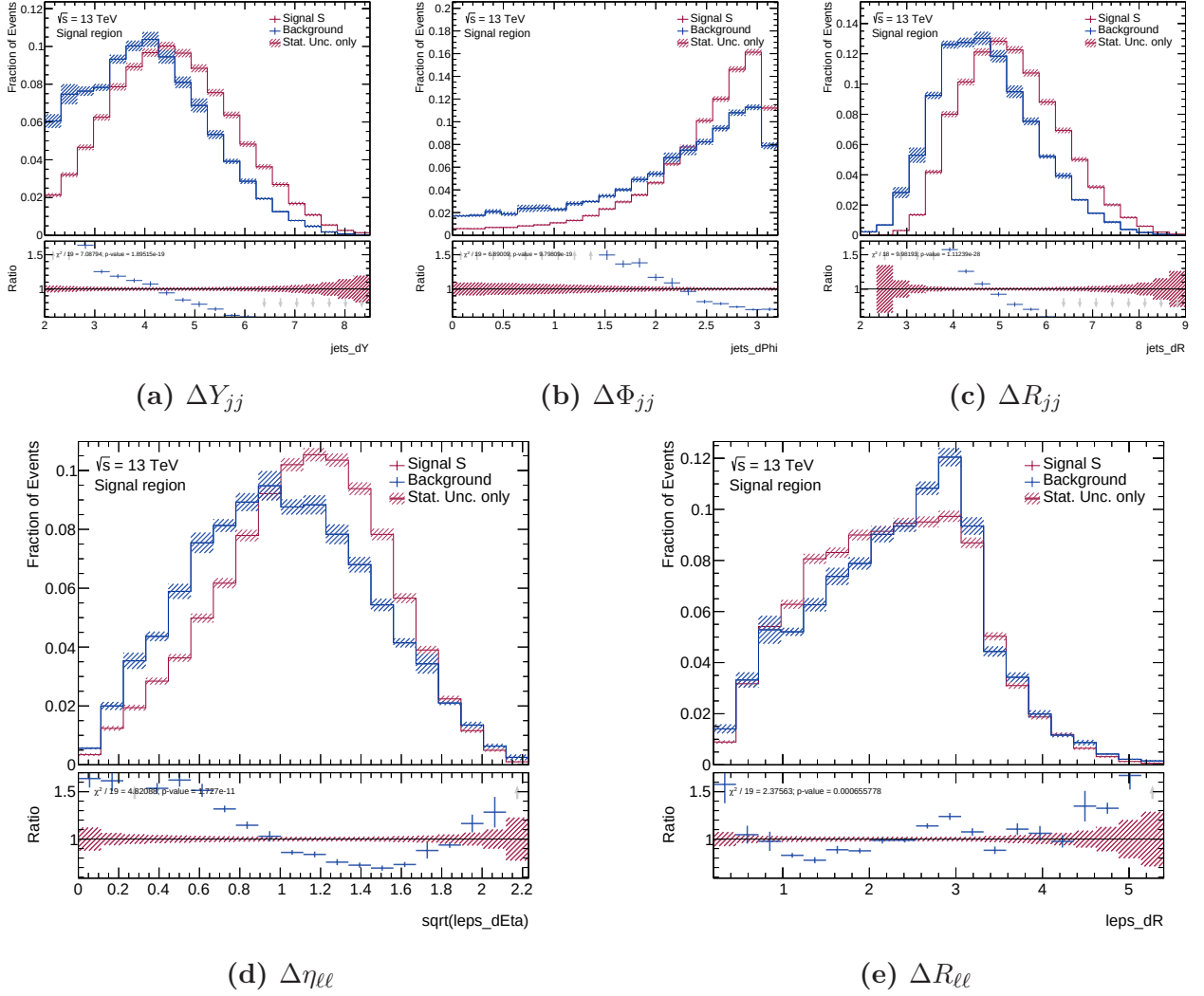


Figure E.4: Preprocessed signal and background distributions of (a) the rapidity separation, (b) azimuthal distance and (c) angular distance of the tagging jets and (d) the pseudo-rapidity separation and (e) angular distance of the two leptons.

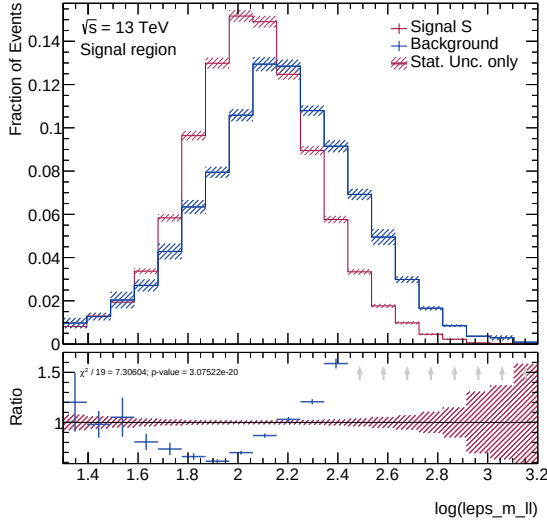
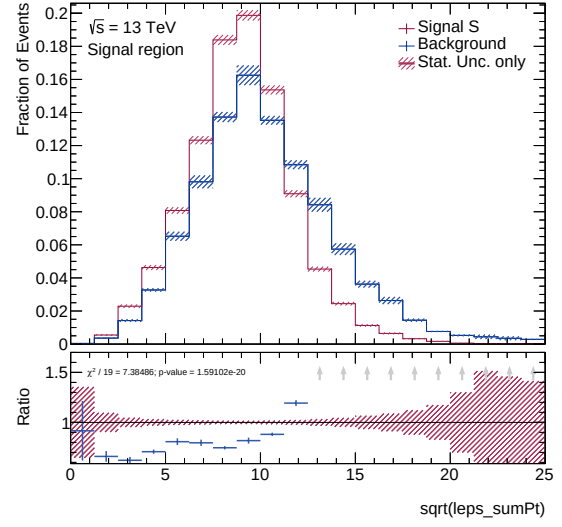
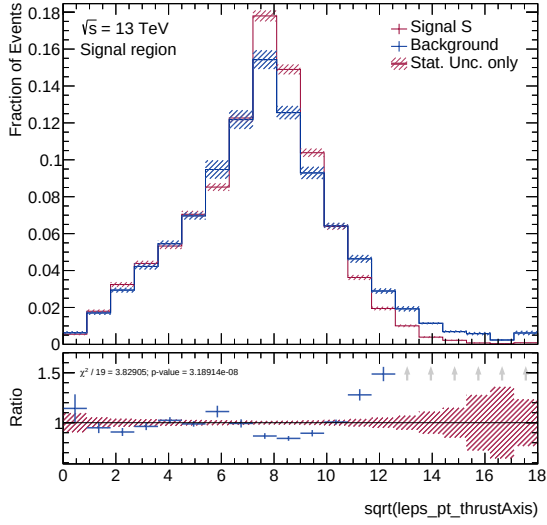
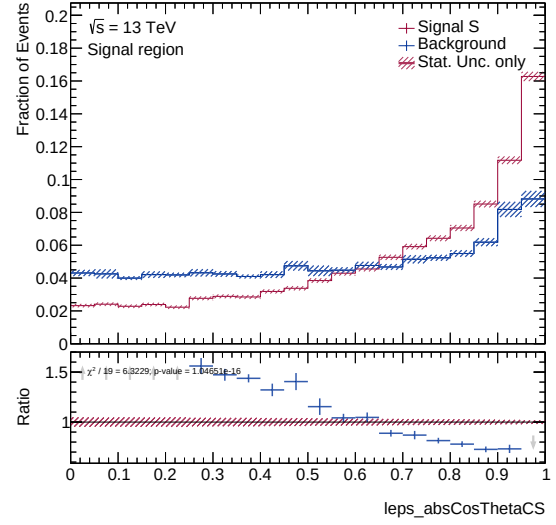
(a) $m_{\ell\ell}$ (b) $p_{T,\ell\ell}$ (c) $a_{T,\ell\ell}$ (d) $|\cos(\vartheta_{CS}^{\ell\ell})|$

Figure E.5: Preprocessed signal and background distributions of (a) the invariant mass of the dilepton system, (b) the transverse momentum of the dilepton system, (c) transverse momentum of the dilepton system relative to its thrust axis and (d) the absolute value of the cosine of the angle between the two leptons in the Collins-Soper reference frame.

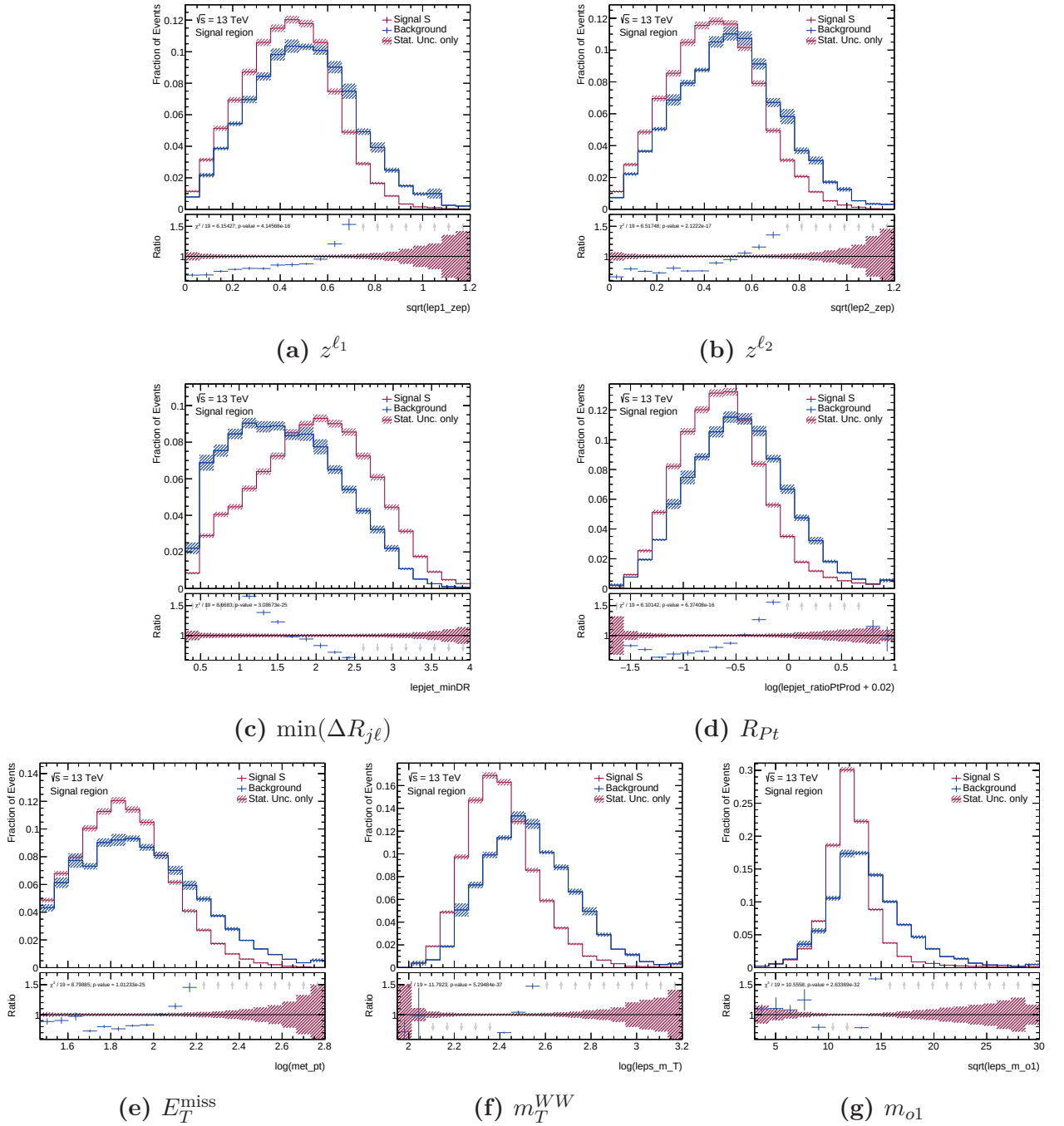


Figure E.6: Preprocessed signal and background distributions of the Zeppenfeld variable of the leading (a) and subleading (b) lepton, (c) the minimal angular distance between the leptons and the tagging jets, (d) the ratio of the products of the transverse momenta of the leptons and the tagging jets, (e) the missing transverse energy, (f) the transverse mass of the dilepton system and the missing transverse energy and (g) the early-projected massless transverse invariant mass of the dilepton system and the missing transverse energy.

F Configuration of the optimization framework

F.1 Example configuration file for the optimizations determining the optimal settings of the TPESampler and the reporting of results to Tune

```
import numpy as np
import KerasTools
import FiguresOfMerit

# input data and splittings
use_custom_splitting = True
custom_splitting_C = 1
test_fraction = 0.2
validation_fraction = 0.2
max_num_events = 25000

# Initial input variables
input_vars = ['jet1_pt', 'jet1_phi_recalibrated', 'jet1_eta',
              'jet2_pt', 'jet2_phi_recalibrated', 'jet2_eta',
              'lep1_pt', 'lep1_eta',
              'lep2_pt', 'lep2_phi_recalibrated', 'lep2_eta',
              'met_pt', 'met_phi_recalibrated',
              'jet1_rap', 'jet2_rap',
              'lep1_zep', 'lep2_zep',
              'jets_dY', 'jets_dPhi', 'jets_dR',
              'leps_dEta', 'leps_dR', 'leps_m_ll', 'leps_absCosThetaCS',
              'leps_pt_thrustAxis', 'leps_m_T', 'leps_sumPt', 'leps_m_o1',
              'lepjet_ratioPtProd', 'lepjet_minDR']

# input scaling
inputs_scaling_dict = {
    "jet1_pt": (lambda x: np.log10(x)),
    "jet1_eta": (lambda x: x),
    "jet1_phi_recalibrated": (lambda x: x),
```

```

    "jet2_pt": (lambda x: np.log10(x)),
    "jet2_eta": (lambda x: x),
    "jet2_phi_recalibrated": (lambda x: x),
    "lep1_pt": (lambda x: np.log10(x)),
    "lep1_eta": (lambda x: x),
    "lep2_pt": (lambda x: np.log10(x)),
    "lep2_eta": (lambda x: x),
    "lep2_phi_recalibrated": (lambda x: x),
    "met_pt": (lambda x: np.log10(x)),
    "met_phi_recalibrated": (lambda x: x),
    "jet1_rap": (lambda x: x),
    "jet2_rap": (lambda x: x),
    "lep1_zep": (lambda x: np.sqrt(x)),
    "lep2_zep": (lambda x: np.sqrt(x)),
    "jets_dY": (lambda x: x),
    "jets_dPhi": (lambda x: x),
    "jets_dR": (lambda x: x),
    "leps_dEta": (lambda x: np.sqrt(x)),
    "leps_dR": (lambda x: x),
    "leps_m_ll": (lambda x: np.log10(x)),
    "leps_sumPt": (lambda x: np.sqrt(x)),
    "leps_absCosThetaCS": (lambda x: x),
    "leps_pt_thrustAxis": (lambda x: np.sqrt(x)),
    "lepjet_ratioPtProd": (lambda x: np.log10(x + 0.02)),
    "lepjet_minDR": (lambda x: x),
    "leps_m_T": (lambda x: np.sqrt(x)),
    "leps_m_o1": (lambda x: np.sqrt(x)),
}

# general settings for the optimization
monitor_name = 'val_bce_loss'
monitor_op = 'min'
optimize_on_best_value = True
restore_on_best_checkpoint = True
max_epochs = 1000
early_stopping_patience = 10
overtraining_patience = 6

# settings for the evaluation
fit_min_R_squared = 0.9
check_overtraining_with_fit = True
use_early_stopping_for_crossvalidation = True
use_OT_conditions_for_crossvalidation = True

# settings for the initial optimization phase and the variable optimization
perform_variable_opt = False

```

```
num_samples_variableOpt = 400
var_metric = 'loss'
var_metric_op = 'min'
max_rel_change = 0.0
num_shuffles = 5
use_best_for_threshold = True

# settings for Optuna and the Asynchronous HyperBand scheduler (ASHA)
use_optuna_opt = True
num_samples_optuna = 10000
ASHA_grace_period = 10
ASHA_max_t = 75
ASHA_reduction_factor = 2
use_multivariate_TPE = True

# PBT specific settings
use_population_based_training = False
use_fit_results_for_PBT = False
num_samples_PBT = 400
perturbation_interval = 4
burn_in_period = 6

# hyperparameter search space
search_space = {
    "num_layers": ('uniform', 3, 6, 1),
    "units": ('log', 4, 128, 1),
    "activation": 'swish',
    "kernel_initializer": 'auto',
    "bias_initializer": 'auto',
    "l1_lambda": ('log', 1e-18, 1e-1),
    "l2_lambda": ('log', 1e-18, 1e-1),
    "dropout": ('uniform', 0.0, 0.2),
    "batch_size": ('log', 4, 256, 1),
    "learning_rate": ('log', 1e-5, 1e-2),
    "Adam_beta_1": ('uniform', 1e-4, 0.99),
    "one_minus_Adam_beta_2": ('log', 1e-5, 0.9999),
    "Adam_epsilon": ('log', 1e-10, 1.0),
    "loss_function": 'BinaryCrossentropy',
    "loss_signal_weight": ('log', 0.1, 10.0),
}

optimize_units_per_layer = False

# metrics
custom_metrics = [
    ('bce_loss',
     KerasTools.WeightedBinaryCrossentropy(signal_weight=1.0, only_numpy=True).calc_loss)]
```

```
custom_Keras_metrics = []
custom_weighted_Keras_metrics = []
composite_metrics = []

# overtraining conditions
overtraining_conditions = []
```

F.2 Changes applied to the configuration file for the determination of the optimal hyperparameter search space

Optimization target: binary cross-entropy

```
# input data and splittings
max_num_events = np.inf

# general settings for the optimization
optimize_on_best_value = False
restore_on_best_checkpoint = False

# settings for Optuna and the Asynchronous HyperBand scheduler (ASHA)
num_samples_optuna = 2000
use_multivariate_TPE = False
```

Optimization target: Asimov log-likelihood ratio

```
# general settings for the optimization
monitor_name = 'val_LogLikelihoodRatio'
monitor_op = 'max'

# metrics
custom_metrics = [
    ('LogLikelihoodRatio', FiguresOfMerit.build_FoM(name='LogLikelihoodRatio',
        exp_sig=20.002, exp_bkg=394.450, min_events_per_bin=10.)),
]
```

F.3 Definition of the overfitting conditions

Optimization target: binary cross-entropy

```
# settings for the evaluation
use_OT_conditions_for_crossvalidation = False

# overtraining conditions
overtraining_conditions = [
    ('bce_loss train / val', ('train_bce_loss', 'val_bce_loss'),
     lambda train_loss, val_loss: np.divide(train_loss, val_loss) < 0.95)
]
```

Optimization target: Asimov log-likelihood ratio

```
# settings for the evaluation
use_OT_conditions_for_crossvalidation = False

# overtraining conditions
overtraining_conditions = [
    ('LogLikelihoodRatio val / train', ('train_LogLikelihoodRatio', 'val_LogLikelihoodRatio'),
     lambda train_LogLikelihoodRatio, val_LogLikelihoodRatio:
         np.divide(val_LogLikelihoodRatio, train_LogLikelihoodRatio) < 0.95),
]
```

F.4 Changes applied to the configuration file for the investigation of the input variable selection

```
# settings for the initial optimization phase and the variable optimization
perform_variable_opt = True
use_fit_results_for_varOpt = True
variable_optimization_mode = 'retrain'
num_samples_variableOpt = 1000
var_metric = 'loss'
var_metric_op = 'min'
max_rel_change = 0.0
use_best_for_threshold = False
use_median_for_averages = False

# settings for Optuna and the Asynchronous HyperBand scheduler (ASHA)
use_optuna_opt = False
```

F.5 Changes applied to the configuration file for the final results using the optimized input variable sets

```
# initial input variables
input_vars = ['jet1_pt', 'jet1_phi_recalibrated', 'jet1_eta',
              'lep1_eta', 'lep2_pt', 'lep2_phi_recalibrated',
              'lep2_eta', 'met_pt', 'jet2_rap', 'jets_dPhi',
              'leps_m_ll', 'leps_m_T', 'leps_m_o1',
              'lepjet_ratioPtProd', 'lepjet_minDR']

# general settings for the optimization
monitor_name = 'val_bce_loss'
monitor_op = 'min'

# settings for the initial optimization phase and the variable optimization
perform_variable_opt = False

# settings for Optuna and the Asynchronous HyperBand scheduler (ASHA)
use_optuna_opt = True
num_samples_optuna = 3000

# PBT specific settings
use_population_based_training = True
```


List of Figures

2.1	Example Feynman diagrams of gauge boson self-interactions	5
2.2	Feynman diagrams of the interactions between the Higgs and the massive electroweak gauge bosons.	7
2.3	Leading order Feynman diagrams of the interaction of two same-charged $W^\pm$ -bosons.	8
2.4	Feynman diagram of the $W^\pm W^\pm$ vector boson scattering at leading order at the LHC	9
2.5	Examples of non-VBS Feynman diagrams of order $\mathcal{O}(\alpha_{EW}^6 \alpha_S^0)$ contributing to the $qq \rightarrow \ell_1^\pm \nu_{\ell_1} \ell_2^\pm \nu_{\ell_2} jj$ process	10
2.6	Examples of Feynman diagrams contributing to the $W^\pm W^\pm jj$ -QCD process	10
2.7	Scattering cross-section for all combinations of longitudinally polarized massive gauge boson pairs, $V_L V_L \rightarrow V_L V_L$, as a function of the center of mass energy	12
3.1	Schematic depiction of the CERN accelerator complex	14
3.2	Cut-away view of the ATLAS detector	16
3.3	Example of a binary decision tree	19
3.4	Example of a multilayer perceptron	24
3.5	Schematic depiction of NN training	27
3.6	Generalization and training error as a function of model capacity	29
3.7	Visualization of over- and underfitting	29
3.8	Visualization of Early Stopping	30
4.1	Comparison of grid search and random search	33
4.2	Visualization of different sampling algorithms	34
4.3	Visualization of Bayesian optimization with a Gaussian process	36
4.4	Visualization of Population Based Training	37
4.5	Distribution of input variable before and after scaling	41
4.6	Optimization overview and progress plots	49
4.7	Trail selection based on the best metric value and best fit function value	50
5.1	Most polarization sensitive observables	64
5.2	Comparison of the full, the unpolarized and sum of the polarized samples	67
5.3	Normalized signal and background distribution of $p_T^{\ell_1}$ before and after preprocessing	71
5.4	Influence of the number of trees and the maximum depth on the BDT's performance	76

5.5	Influence of the shrinkage and the minimum node size on the BDT's performance	77
5.6	Change of the BDT's performance over the course of the input variable optimizations	79
5.7	Output distributions of the final BDT	81
5.8	Minimum validation loss as a function of the total number of epochs for four different optimization settings	86
5.9	Output distributions of the MLP resulting from the optimizations to determine the best hyperparameter search space (loss optimization target)	90
5.10	Output distributions of the MLP resulting from the optimizations to determine the best hyperparameter search space (Asimov log-likelihood ratio optimization target)	94
5.11	Visualization of the influence of the generalization gap on the observed significance	95
5.13	Evolution of the average relative changes of the target metric over the course of the input variable selection for different optimization settings	102
5.14	Output distributions of the MLP resulting from the optimizations to determine the best hyperparameter search space (Asimov log-likelihood ratio optimization target)	106
5.15	Expected discovery significance of the $W_L^\pm W_L^\pm jj$ -EW polarization component for different integrated luminosities obtained by the study performed in Ref. [79]. . .	108
C.1	Distribution of single-jet observables for the full, unpolarized and polarized samples	116
C.2	Distribution of single-lepton observables for the full, unpolarized and polarized samples	117
C.3	Distributions of angular distances of the tagging jets and the two leptons for the full, unpolarized and polarized samples	118
C.4	Distributions of dilepton observables for the full, unpolarized and polarized samples	119
C.5	Distributions of lepton, jet and E_T^{miss} observables for the full, unpolarized and polarized samples	120
D.1	Distribution of observables of the leading jet for different polarizations	121
D.2	Distribution of observables of the subleading jet for different polarizations	122
D.3	Distribution of single-lepton observables for different polarizations	123
D.4	Distributions of angular distances of the tagging jets and the two leptons for different polarizations	124
D.5	Distributions of dilepton observables for different polarizations	125
D.6	Distributions of lepton and jet observables for different polarizations	126
D.7	Distributions of lepton and E_T^{miss} for different polarizations	127
E.1	Preprocessed signal and background distributions of observables of the leading jet	128
E.2	Preprocessed signal and background distributions of observables of the subleading jet	129
E.3	Preprocessed signal and background distributions of single-lepton observables . . .	130
E.4	Preprocessed signal and background distributions of angular distances of the tagging jets and the two leptons	131

E.5	Preprocessed signal and background distributions of dilepton observables	132
E.6	Preprocessed signal and background distributions of lepton, jet and E_T^{miss} observables	133

List of Tables

5.1	Muon and Electron selection criteria	57
5.2	Event selection defining the signal region of the $W^\pm W^\pm jj$ -EW process	59
5.3	Estimated event yields of the $W^\pm W^\pm jj$ -EW and relevant background processes .	60
5.4	List of polarization sensitive observables and corresponding descriptions.	62
5.5	Significance of the $W_L^\pm W_L^\pm$ contribution on different observables	63
5.6	Comparison of the expected event yields of the $W^\pm W^\pm jj$ -EW process	65
5.7	Ranking and preprocessing for BDT input variables	72
5.8	Set of hyperparameters used during the first step of the BDT optimization.	73
5.9	Search space of the BDT hyperparameter optimization	74
5.10	Optimized set of BDT hyperparameters	75
5.11	Optimized list of input variables of the BDT	78
5.12	Expected discovery significances of the LL polarization component using output of the five final BDTs	80
5.13	Hyperparameter search space used for the determination of the best settings for the TPESampler and the reporting of results to Tune.	85
5.14	Default values for hyperparameters not included in the hyperparameter search space.	87
5.15	Overview of the optimizations performed to determine the minimal reasonable hyperparameter search space (binary cross-entropy loss)	88
5.16	Optimized values of the hyperparameters when using the binary cross-entropy optimization target and the full set of input variables	89
5.17	Overview of the optimizations performed to determine the minimal reasonable hyperparameter search space (Asimov log-likelihood ratio)	92
5.18	Optimized values of the hyperparameters when using the Asimov log-likelihood optimization target and the full set of input variables	92
5.19	Expected event yields using the validation datasets corresponding to the k cross- validation splittings	97
5.20	Influence of the threshold c of the overfitting condition on the fit results for the binary cross-entropy loss optimization target	99
5.21	Influence of the threshold c of the overfitting condition on the fit results for the Asimov log-likelihood ratio optimization target	100
5.22	Optimized sets of MLP input variables	103

5.23	Manually built sets of MLP input variables	104
5.24	Expected discovery significances of the LL polarization component using the MLPs obtained from the final optimization runs	107
B.1	List of Monte Carlo samples	114

Bibliography

- [1] The ATLAS Collaboration, “Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC,” Phys.Lett. B716 (2012) 1-29 **716**, 1 (2012), arXiv:1207.7214 [hep-ex].
- [2] The CMS Collaboration, “Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC,” Phys. Lett. B 716 (2012) 30 **716**, 30 (2012), arXiv:1207.7235 [hep-ex].
- [3] The CMS Collaboration, “Observation of Electroweak Production of Same-Sign W Boson Pairs in the Two Jet and Two Same-Sign Lepton Final State in Proton Proton Collisions at $\sqrt{s} = 13$ TeV,” Physical Review Letters **120**, 081801 (2018).
- [4] The ATLAS Collaboration, “Observation of Electroweak Production of a Same-Sign W Boson Pair in Association with Two Jets in pp Collisions at $\sqrt{s} = 13$ TeV with the ATLAS Detector,” Physical Review Letters **123**, 161801 (2019).
- [5] The CMS Collaboration, “Measurements of production cross sections of polarized same-sign W boson pairs in association with two jets in proton-proton collisions at $\sqrt{s} = 13$ TeV,” Phys. Lett. B 812 (2020) 136018, doi:10.1016/j.physletb.2020.136018 (2020), arXiv:2009.09429 [hep-ex].
- [6] S. Todt, “The $W^{\pm}W^{\pm}$ Scattering Process – Theoretical Studies, Observation, Cross-section Measurement, and EFT Reinterpretation at $\sqrt{s} = 13$ TeV with the ATLAS Detector,” Presented 02 Dec 2022, PhD thesis (2022), <https://cds.cern.ch/record/2845282>.
- [7] M. Hoppe, “Implementation and phenomenology of polarized cross sections in the simulation of vector boson production with the SHERPA event generator,” MA thesis (2023).
- [8] M. V. Stange, “Data-driven estimation of the non-prompt background in same charged $W^{\pm}W^{\pm}$ scattering within the ATLAS experiment,” Presented 05 Nov 2020 (2020), <https://cds.cern.ch/record/2746597>.
- [9] A. S. Tegetmeier, “Multivariate Analysis of Polarized Vector Boson Scattering of a W and Z Boson with the ATLAS Detector,” Presented 2022, MA thesis (2022), <https://cds.cern.ch/record/2842600>.

- [10] C. S. Wu et al., “Experimental Test of Parity Conservation in Beta Decay,” *Physical Review* **105**, 1413 (1957).
- [11] S. L. Glashow, “Partial-symmetries of weak interactions,” *Nuclear Physics* **22**, 579 (1961).
- [12] A. Salam, “Elementary Particle Theory,” Almqvist and Wiksell, (1968).
- [13] S. Weinberg, “A Model of Leptons,” *Physical Review Letters* **19**, 1264 (1967).
- [14] F. Englert and R. Brout, “Broken Symmetry and the Mass of Gauge Vector Mesons,” *Physical Review Letters* **13**, 321 (1964).
- [15] P. W. Higgs, “Broken Symmetries and the Masses of Gauge Bosons,” *Physical Review Letters* **13**, 508 (1964).
- [16] G. S. Guralnik, C. R. Hagen, and T. W. B. Kibble, “Global Conservation Laws and Massless Particles,” *Physical Review Letters* **13**, 585 (1964).
- [17] C. Gumpert, “Measurement of Electroweak Gauge Boson Scattering in the Channel $pp \rightarrow W^\pm W^\pm jj$ with the ATLAS Detector at the Large Hadron Collider,” Presented 27 Feb 2015, PhD thesis (2014), <https://cds.cern.ch/record/2003240>.
- [18] C. Bittrich, “Evidence for Scattering of Electroweak Gauge Bosons in the $W^\pm Z$ Channel with the ATLAS Detector at the Large Hadron Collider,” Presented 14 Jul 2020, PhD thesis (2020), <https://cds.cern.ch/record/2719126>.
- [19] C. Bittrich, “Study of Polarization Fractions in the Scattering of Massive Gauge Bosons $W^\pm Z \rightarrow W^\pm Z$ with the ATLAS Detector at the Large Hadron Collider,” Presented 2015, MA thesis (2015), <https://cds.cern.ch/record/2014124>.
- [20] L. R. Evans and P. Bryant, “LHC Machine,” *JINST* **3**, This report is an abridged version of the LHC Design Report (CERN-2004-003), S08001 (2008), <https://cds.cern.ch/record/1129806>.
- [21] The ATLAS Collaboration et al., “The ATLAS Experiment at the CERN Large Hadron Collider,” *Journal of Instrumentation* **3**, S08003 (2008).
- [22] The CMS Collaboration et al., “The CMS experiment at the CERN LHC,” *Journal of Instrumentation* **3**, S08004 (2008).
- [23] The ALICE Collaboration et al., “The ALICE experiment at the CERN LHC,” *Journal of Instrumentation* **3**, S08002 (2008).
- [24] The LHCb Collaboration et al., “The LHCb Detector at the LHC,” *Journal of Instrumentation* **3**, S08005 (2008).
- [25] E. Mobs, “The CERN accelerator complex in 2019. Complexe des accélérateurs du CERN en 2019,” *General Photo* (2019), <https://cds.cern.ch/record/2684277>.
- [26] The ATLAS collaboration, “ATLAS data quality operations and performance for 2015–2018 data-taking,” *Journal of Instrumentation* **15**, P04003 (2020).

- [27] A. Hoecker et al., “TMVA - Toolkit for Multivariate Data Analysis,” (2007), arXiv:physics/0703039v5.
- [28] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning* (Springer New York, 2009).
- [29] F. Chollet, *Deep Learning with Python* (Manning Publications, 2017).
- [30] S. Thais, *Machine Learning in ATLAS*, presented at SYNPA 2017, https://indico.cern.ch/event/673207/contributions/2755302/attachments/1560980/2457386/synpa_2017.pdf.
- [31] U. Lämmel and J. Cleve, *Künstliche Intelligenz* (Carl Hanser Verlag GmbH & Co. KG, 2008).
- [32] S. Ruder, “An overview of gradient descent optimization algorithms,” (2016), arXiv:1609.04747v2 [cs.LG].
- [33] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016), <http://www.deeplearningbook.org>.
- [34] M. V. Stange, “Schematic depiction of a Multilayer Perceptron,” (unpublished).
- [35] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, “Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark,” (2021), arXiv:2109.14545v3 [cs.LG].
- [36] D. Bertoin et al., “Numerical influence of ReLU’(0) on backpropagation,” (2021), arXiv:2106.12915v2 [cs.LG].
- [37] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” (2014), arXiv:1412.6980v9 [cs.LG].
- [38] F. Pedregosa et al., “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research* **12**, 2825 (2011).
- [39] N. Srivastava et al., “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *Journal of Machine Learning Research* **15**, 1929 (2014), <http://jmlr.org/papers/v15/srivastava14a.html>.
- [40] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” 281 (2012).
- [41] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated Machine Learning : Methods, Systems, Challenges, Methods, systems, challenges* (Springer, 2019), p. 233.
- [42] J. Bergstra et al., “Algorithms for Hyper-Parameter Optimization,” in *Advances in Neural Information Processing Systems*, Vol. 24, edited by J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger (2011), <https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf>.

- [43] J. Bergstra, D. Yamins, and D. Cox, “Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures,” in Proceedings of the 30th International Conference on Machine Learning, Vol. 28, edited by S. Dasgupta and D. McAllester, Proceedings of Machine Learning Research 1 (17–19 Jun 2013), pp. 115–123, <https://proceedings.mlr.press/v28/bergstra13.html>.
- [44] T. Akiba et al., “Optuna: A Next-generation Hyperparameter Optimization Framework,” doi:10.48550/ARXIV.1907.10902 (2019), arXiv:1907.10902 [cs.LG].
- [45] M. Jaderberg et al., “Population Based Training of Neural Networks,” (2017), arXiv:1711.09846 [cs.LG].
- [46] R. Liaw et al., “Tune: A Research Platform for Distributed Model Selection and Training,” doi:10.48550/ARXIV.1807.05118 (2018), arXiv:1807.05118 [cs.LG].
- [47] F. Chollet et al., *Keras*, <https://keras.io>.
- [48] Y. A. LeCun et al., “Efficient BackProp,” in *Lecture Notes in Computer Science* (Springer Berlin Heidelberg, 2012), pp. 9–48.
- [49] K. He, X. Zhang, S. Ren, and J. Sun, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,” doi:10.48550/ARXIV.1502.01852 (2015), arXiv:1502.01852 [cs.CV].
- [50] S. Hayou, A. Doucet, and J. Rousseau, “On the Selection of Initialization and Activation Function for Deep Neural Networks,” doi:10.48550/ARXIV.1805.08266 (2018), arXiv:1805.08266 [stat.ML].
- [51] L. Li et al., “A System for Massively Parallel Hyperparameter Tuning,” Conference on Machine Learning and Systems 2020, doi:10.48550/ARXIV.1810.05934 (2018), arXiv:1810.05934 [cs.LG].
- [52] P. Ramachandran, B. Zoph, and Q. V. Le, “Searching for Activation Functions,” (2017), arXiv:1710.05941 [cs.NE].
- [53] D. Misra, “Mish: A Self Regularized Non-Monotonic Activation Function,” (2019), arXiv:1908.08681 [cs.LG].
- [54] G. Klambauer et al., “Self-Normalizing Neural Networks,” Advances in Neural Information Processing Systems 30 (NIPS 2017), doi:10.48550/ARXIV.1706.02515 (2017), arXiv:1706.02515 [cs.LG].
- [55] M. Tavakoli, F. Agostinelli, and P. Baldi, “SPLASH: Learnable Activation Functions for Improving Accuracy and Adversarial Robustness,” doi:10.48550/ARXIV.2006.08947 (2020), arXiv:2006.08947 [cs.LG].

- [56] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in Proceedings of the thirteenth international conference on artificial intelligence and statistics, Vol. 9, edited by Y. W. Teh and M. Titterton, Proceedings of Machine Learning Research (13–15 May 2010), pp. 249–256, <https://proceedings.mlr.press/v9/glorot10a.html>.
- [57] J. Pennington, S. S. Schoenholz, and S. Ganguli, “Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice,” doi:10.48550/ARXIV.1711.04735 (2017), arXiv:1711.04735 [cs.LG].
- [58] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” doi:10.48550/ARXIV.1502.03167 (2015), arXiv:1502.03167 [cs.LG].
- [59] R. May, G. Dandy, and H. Maier, “Review of Input Variable Selection Methods for Artificial Neural Networks,” in *Artificial neural networks - methodological advances and biomedical applications* (InTech, Apr. 2011).
- [60] M. Queitsch-Maitland et al., *Support note for measurement of fiducial and differential $W^\pm W^\pm jj$ production cross-sections at $\sqrt{s} = 13$ TeV using the full Run-2 dataset*, tech. rep. (CERN, Geneva, 2021), <https://cds.cern.ch/record/2790652>.
- [61] The ATLAS Collaboration, “Jet reconstruction and performance using particle flow with the ATLAS Detector,” Eur. Phys. J. C **77** (2017) 466 **77**, doi:10.1140/epjc/s10052-017-5031-2 (2017), arXiv:1703.10485 [hep-ex].
- [62] M. Cacciari, G. P. Salam, and G. Soyez, “The anti- k_t jet clustering algorithm,” JHEP **04**, 063 (2008), arXiv:0802.1189 [hep-ph].
- [63] M. Cacciari, G. P. Salam, and G. Soyez, “FastJet user manual,” Eur. Phys. J. C **72**, 1896 (2012), arXiv:1111.6097 [hep-ph].
- [64] The ATLAS Collaboration, “Performance of pile-up mitigation techniques for jets in pp collisions at $\sqrt{s} = 8$ TeV using the ATLAS detector,” Eur. Phys. J. C **76**, 581 (2016), arXiv:1510.03823 [hep-ex].
- [65] D. B. Franzosi et al., “Automated Predictions from Polarized Matrix Elements,” J. High Energ. Phys. **2020**, 82 (2020) **2020**, doi:10.1007/jhep04(2020)082 (2019), arXiv:1912.01725 [hep-ph].
- [66] T. Sjöstrand et al., “An introduction to pythia 8.2,” Computer Physics Communications **191**, 159 (2014), arXiv:1410.3012 [hep-ph].
- [67] J. Alwall et al., “The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations,” JHEP07(2014)079 **2014**, doi:10.1007/jhep07(2014)079 (2014), arXiv:1405.0301 [hep-ph].

- [68] J. Bellm et al., “Herwig 7.0 / herwig++ 3.0 release note,” The European Physical Journal C **76**, doi:10.1140/epjc/s10052-016-4018-8 (2015), arXiv:1512.01178 [hep-ph].
- [69] The ATLAS Collaboration, *Formulae for Estimating Significance*, tech. rep. (CERN, Geneva, 2020), <https://cds.cern.ch/record/2736148>.
- [70] J. Alwall et al., “MadGraph 5 : Going Beyond,” Journal of High Energy Physics **2011**, doi:10.1007/jhep06(2011)128 (2011), arXiv:1106.0522 [hep-ph].
- [71] P. Baldi, P. Sadowski, and D. Whiteson, “Searching for Exotic Particles in High-Energy Physics with Deep Learning,” Nature Communications **5**, doi:10.1038/ncomms5308 (2014), arXiv:1402.4735 [hep-ph].
- [72] S. Falkner, A. Klein, and F. Hutter, “BOHB: Robust and Efficient Hyperparameter Optimization at Scale,” doi:10.48550/ARXIV.1807.01774 (2018), arXiv:1807.01774 [cs.LG].
- [73] D. H. Guest and lwttn contributors, *Lightweight Trained Neural Network Version 2.13*, Apr. 2022, <https://github.com/lwttn/lwttn>.
- [74] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs),” doi:10.48550/ARXIV.1511.07289 (2015), arXiv:1511.07289 [cs.LG].
- [75] A. Elwood and D. Krücker, “Direct optimisation of the discovery significance when training neural networks to search for new physics in particle colliders,” (2018), arXiv:1806.00322 [hep-ex].
- [76] K. Schubert, “Application of the Profile-Likelihood Method to the Combination of Cross Section Measurements,” MA thesis (2012), https://iktp.tu-dresden.de/IKTP/pub/12/bachelor_KS.pdf.
- [77] O. Aberle et al., *High-Luminosity Large Hadron Collider (HL-LHC): Technical design report*, tech. rep. (Geneva, 2020), <https://cds.cern.ch/record/2749422>.
- [78] *The High Luminosity LHC Project*, visited on 4.4.2023, <https://hilumilhc.web.cern.ch/content/hl-lhc-project>.
- [79] The ATLAS Collaboration, *Prospects for the measurement of the $W^\pm W^\pm$ scattering cross section and extraction of the longitudinal scattering component in pp collisions at the High-Luminosity LHC with the ATLAS experiment*, tech. rep. (CERN, Geneva, 2018), <https://cds.cern.ch/record/2652447>.
- [80] M. Abdughani et al., “Probing stop pair production at the LHC with graph neural networks,” Journal of High Energy Physics **2019**, doi:10.1007/jhep08(2019)055 (2018), arXiv:1807.09088 [hep-ph].

Danksagung

Ich möchte mich an dieser Stelle bei all denjenigen bedanken, die mich während meines Studiums begleitet haben und ohne deren Unterstützung diese Arbeit nicht zustande gekommen wäre. Zunächst geht ein großer Dank an meinen Betreuer Dr. Frank Siegert für die Möglichkeit, im Rahmen meiner Masterarbeit an solch einem interessanten und vielversprechenden Thema zu arbeiten und meine Ergebnisse bei ATLAS-D in Siegen und der DPG-Frühjahrstagung in Dresden vorstellen zu können. Des Weiteren möchte ich mich bei Max Stange dafür bedanken, dass er mir während des letzten Jahres mit Rat und Tat zur Seite stand und mit seiner fröhlichen Art stets für gute Laune sorgen konnte.

I would also like to thank the entire VBS group. When I started working on my thesis a year ago, I was welcomed by such a friendly and open group of people that I never thought existed in a working environment as competitive as research. Thank you for all the interesting and funny conversations at the BBQs or the “Nachsitzen”.

Additionally, I would like to give a big thank you to Joany Manjarres for sparking my interest in particle physics analyses. Without her hands-on lecture, I would have probably pursued a very different path.

Für das Korrekturlesen meiner Arbeit bedanke ich mich bei Dirk Bachmann, Stefanie Todt und Max Stange.

Ein besonderes Dankeschön geht an meine Eltern Katrin und Dirk Bachmann und meine Großeltern Irene Rapp und Ursula und Herbert Bachmann dafür, dass ihr immer für mich da wart, mich unterstützt und mein Interesse für Naturwissenschaft, Technik und Computer von Anfang an gefördert habt. Ihr habt mich zu dem Menschen gemacht, der ich heute bin, und ich danke euch dafür von ganzem Herzen!

Zu guter Letzt möchte ich mich bei meiner Freundin Mareen Hoppe dafür bedanken, dass du immer ein offenes Ohr für meine Probleme hast, auch dann, wenn das Problem zum zehnten Mal das gleiche ist. Danke dafür, dass du mich immer motivierst, an mich glaubst, und jederzeit für mich da bist. Ohne deine stetige Unterstützung hätte ich es nicht bis hierhin geschafft. Und danke dafür, dass du immer für Diskussionen zur Verfügung standest, auch dann, wenn du der Ansicht warst, nicht helfen zu können; dein Blickwinkel hat so oft den zündenden Funken zur Lösung eines Problems beigetragen.

Erklärung

Hiermit erkläre ich, dass ich diese Arbeit im Rahmen der Betreuung am Institut für Kern- und Teilchenphysik ohne unzulässige Hilfe Dritter verfasst und alle Quellen als solche gekennzeichnet habe.

Erik Bachmann

Dresden, April 2023