ETP-KA/2022-19

Karlsruhe Institute of Technology

Feasibility and Reliability Studies of Graph Neural Networks for Multivariate $t\bar{t}$+X Event Classification at the CMS Experiment at CERN

Master Thesis

Yee-Ying Christina Cung

At the Department of Physics
Institute of Experimental Particle Physics

Reviewer:	Prof. Dr. Ulrich Husemann
Second Reviewer:	PD Dr. Roger Wolf
Advisor:	Emanuel Pfeffer

December 06, 2022

This thesis has been accepted by the first reviewer of the master thesis.

Karlsruhe, December 06, 2022

.....

(**Prof. Dr. Ulrich Husemann**)

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, December 06, 2022

.....

(**Yee-Ying Christina Cung**)

Contents

1 Introduction **1**

2 High Energy Physics at the CMS Experiment at CERN **3**
2.1 Standard Model of Particle Physics 3
2.2 Large Hadron Collider 7
2.3 The CMS Detector 9
2.4 Kinematic Quantities 11
2.5 Physics Processes 13
2.6 Jet Reconstruction and b Tagging 14

3 Machine Learning Algorithms **17**
3.1 Deep Neural Networks 17
3.2 Graph Neural Networks 20
3.2.1 Graph Theory 20
3.2.2 Graph Data and Tasks 22
3.2.3 Message Passing Neural Networks 23
3.2.4 Graph Network Formalism 26
3.3 Explainable AI 29
3.3.1 GNNExplainer 29
3.3.2 Taylor Coefficient Analysis 32
3.3.3 Comparison 33

4 Multivariate Event Classification with GNNs **35**
4.1 Reproducibility and Data 35
4.2 Architecture and Hyperparameters 37
4.3 Binary and Multiclass Classification 38
4.4 Further Optimization Approaches 43
4.5 Applying Preclassified Category Flags to Input Features 46

5 Modeling of the Dependency of a GNN-Based Event Classification on the Goodness of the Jet Assignment **51**
5.1 Modeling Strategies 51
5.2 Validation 52

6 In-Depth Analysis of GNNs by Applying Explainable AI Methods **55**
6.1 Identifying the Decision Basis of Models Trained with Different Information Levels 56
6.1.1 Category Importance and Category Specific Feature Importance . . 56
6.1.2 Feature Importance 57
6.1.3 Conclusion 61
6.2 Evolution of the Feature Importance in AddB-LTB Modeling 61

7 Benchmark Study on Equivalent GNNs and DNNs **67**
7.1 Comparability Challenges and Solutions 67
7.2 Comparison of Models with an Equivalent Architecture 69
7.2.1 Model Performance and Training Stability 73
7.2.2 Convergence Speed and Degrees of Freedom 79
7.3 Comparison of Models with a Similar Number of Degrees of Freedom 82
7.3.1 DNNs with a Tuned Number of Degrees of Freedom 82
7.3.2 DNNs with a Restricted Number of Degrees of Freedom 83
7.3.3 GNNs with an Expanded Number of Degrees of Freedom 83
7.4 In-Depth Analysis of the Best-Performing Models 83

8 Summary and Outlook **91**

Bibliography **93**

Appendix **99**
A Distribution of Observables 100
B Decision Basis for Outlier Criterion (b) 101
C Normalized Performance Rates of NLP-GGSNN 102
D Properties of the Manipulated Data Sets 103
E Derivation of $\Delta r^*_{\max}$ 104
F Supplementary Information to Chapter 6 105
G Supplementary Information to Chapter 7 119

1 Introduction

Ten years have passed since the announcement of the world-wide attention-grabbing discovery of the Higgs boson at the ATLAS and CMS experiments at CERN in 2012 [1, 2]. While the Higgs boson is known as the last puzzle piece of the Standard Model, the theory that successfully explains phenomena ranging over several orders of magnitude with great precision, the structure of the universe is still not fully understood. In fact, many phenomena observed in nature are not even covered by the Standard Model, leading to the development of new theories and the search for new physics beyond the Standard Model (BSM) for answering these pending questions. For this reason, as well as for measuring the Higgs sector with even greater precision, the Large Hadron Collider (LHC) at CERN has recently resumed operations after a three-year break for maintenance and upgrades.

Top quark-antiquark pair production in association with a bottom quark-antiquark pair ($t\bar{t} + b\bar{b}$) plays a central role in this thesis. On the one hand, this process is interesting per se as large uncertainties, caused by the very different energy scales associated with $t\bar{t}$ and $b\bar{b}$, are attached to its mathematical description and simulation [3]. On the other hand, the $t\bar{t} + b\bar{b}$ process is an irreducible background to the $t\bar{t}H(b\bar{b})$ process, in which $t\bar{t}$ and a Higgs boson ($t\bar{t}$ + H) are produced in association, followed by the subsequent decay of the Higgs boson into a bottom quark-antiquark pair ($H \to b\bar{b}$). Equally, the $t\bar{t}Z(b\bar{b})$ process ($t\bar{t}$ + Z, $Z \to b\bar{b}$) forms an irreducible background to both processes. It is crucial to separate these processes with high efficiency as the $t\bar{t}H(b\bar{b})$ process is – thanks to the large top quark mass – predestined for probing the coupling strength of the Yukawa coupling, which in turn serves as an important consistency check for both the Standard Model and BSM models [4].

As the physics processes under scrutiny are more naturally representable as graphs, graph neural networks (GNNs), a novel machine learning algorithm dedicated to processing graph data, are presumably a promising approach for multivariate $t\bar{t}$+X event classification. The goal of this thesis is to examine the feasibility of GNNs for this classification task as well as to probe their reliability with explainable AI methods. Additionally, a detailed comparison with a machine learning technique already well-established in high energy physics (HEP), deep neural networks (DNNs), is conducted.

After introducing the theoretical and experimental foundations of this thesis in Chapters 2 and 3, the general feasibility and reliability of graph neural networks for multivariate $t\bar{t}$+X event classification are examined in Chapters 4-6. The studies are subsequently concluded with a comparison of GNNs and DNNs in Chapter 7. Lastly, a summary of the results and an outlook is given in Chapter 8.

2 High Energy Physics at the CMS Experiment at CERN

The theoretical basis of particle physics is the Standard Model of Particle Physics (SM), a theory that describes the matter in the universe and its interactions. Albeit the Standard Model is commonly known to be the best experimentally verified theory in the world, it covers neither gravitation nor various other phenomena that have already been observed, leading to the development of new theories beyond the Standard Model (BSM). In order to further probe the consistency of the Standard Model as well as search for BSM physics in a controlled environment, powerful colliders, such as the Large Hadron Collider (LHC) at the European Organization for Nuclear Research (CERN), and reliable detectors are inevitable. Since the colliders operate at highest energies, particle physics is often referred to as high energy physics (HEP). Consequently, in order to comprehend the studies conducted with (simulated) HEP data in the scope of this thesis (cf. Chapters 4-7), knowledge of the mathematical background of the Standard Model and of the LHC is necessary and are therefore briefly introduced in Section 2.1 and Section 2.2. Subsequently, the CMS detector is described in Section 2.3, before several important kinematic quantities for describing HEP data as well as the physics process under scrutiny in this thesis and the reconstruction of jets, including b tagging, are presented in Sections 2.4-2.6.

2.1 Standard Model of Particle Physics

The Standard Model is a Quantum Field Theory (QFT) and forms the theoretical fundament of today's comprehension of matter and its electromagnetic, weak and strong interactions. This theory enables a quantitative explanation of observed natural phenomena over a range of eleven orders of magnitude of energy, although the SM only comprises of 17 fundamental particles (when neglecting color quantum numbers and antiparticles). Fundamental particles are objects with no further inner structure. They either possess half integer or integer spin in units of $\hbar$, resulting in fundamentally different behavior. Accordingly, the particles are further differentiated in fermions and bosons on the basis of their spin. The former form the matter in our universe whereas the latter are the mediators of the aforementioned interactions between matter. [7]

Fermions are further distinguished in leptons and quarks. As presented in Table 2.1, there are six leptons in total, whereby half of them, the electron (e), the muon (μ) and the

Table 2.1: **Properties of fundamental fermions.** Except for the electron neutrino mass, which corresponds to the value of the latest finding in Ref. [5], all values are taken from Ref. [6]. The uncertainties for the electron and muon mass are omitted for the sake of readability as they are more than seven orders of magnitudes smaller than the respective mass. Incidentally, up, down and strange quarks are also referred to as light flavors, due to their small mass.

fermion name	type	mass	electric charge	interaction
up (u)	up-type quark	$2.16^{+0.49}_{-0.26}$ MeV	2/3	all
charm (c)	up-type quark	(1.27 ± 0.02) GeV	2/3	all
top (t)	up-type quark	(172.69 ± 0.30) GeV	2/3	all
down (d)	down-type quark	$4.67^{+0.48}_{-0.17}$ MeV	−1/3	all
strange (s)	down-type quark	$93.4^{+8.6}_{-3.4}$ MeV	−1/3	all
bottom (b)	down-type quark	$4.18^{+0.03}_{-0.02}$ GeV	−1/3	all
electron (e)	lepton	0.511 MeV	−1	electroweak
muon (μ)	lepton	105.66 MeV	−1	electroweak
tau (τ)	lepton	(1776.86 ± 0.12) MeV	−1	electroweak
electron neutrino (ν_e)	lepton	< 0.8 eV	0	weak
muon neutrino (ν_μ)	lepton	< 0.19 MeV	0	weak
tau neutrino (ν_τ)	lepton	< 18.2 MeV	0	weak

tau (τ), carry an electric charge of -1 in units of e, while the associated neutrinos (ν_e, ν_μ, ν_τ) are electrically neutral. Equally, there are six quark flavors. The quarks up (u), charm (c), top (t) possess an electric charge of 2/3 and the quarks down (d), strange (s), bottom (b) an electric charge of $-1/3$. In addition to the electric charge, a color charge (red, green, blue) is assigned to each quark.
Furthermore, the fermions can be grouped in three generations in ascending order with respect to their mass, whereby each generation consists of an up-type quark, a down-type quark, a charged lepton and the corresponding neutrino. Only the first generation (u, d, e, ν_e) is known to form stable matter while the particles of the remaining generations eventually decay to particles from the first generation. It should be noted that an antiparticle of each fermion exists, leading to 48 fermions in total. [8]

The following explanations follow Ref. [7] and Ref. [9] if not stated otherwise.
For describing the dynamics of the particles in the Standard Model or, more generally speaking, in QFT, the Lagrangian formalism is used. In this formalism, the Lagrangian density $\mathcal{L}(\phi(x), \partial_\mu \phi(x))$, where x denotes space-time coordinates and ∂_μ corresponds to the covariant derivative, plays a central role as it fully describes the state of the field $\phi(x)$. In particular, it turns out that all aforementioned interactions and their mediators (gauge bosons) are a direct consequence of requiring the Lagrangian density to be invariant under certain continuous local gauge transformations.
The Lagrangian density for free fermions is, for instance, given by

$$\mathcal{L} = \overline{\psi}(i\gamma^\mu \partial_\mu - m)\psi \,, \tag{2.1}$$

where m corresponds to the fermion mass, $\psi(x)$ is a spinor field, $\overline{\psi}(x) = \psi^\dagger(x)\gamma^0$ the spinor adjoint and γ^μ denote the Dirac matrices. This Lagrangian density is invariant under global $U(1)$ gauge transformation $\psi(x) \to \psi'(x) = e^{i\theta}\psi(x)$, but not under local $U(1)$ symmetry operations ($\theta = \theta(x)$) unless a gauge field $A_\mu(x)$ and the covariant derivative

$\partial_\mu \rightarrow D_\mu = \partial_\mu + iqA_\mu(x)$ are introduced. Thereby, the gauge field $A_\mu(x)$ emerges as the photon field from electrodynamics and it turns out that the variable q coming alongside the photon field can be identified as the electric charge. As known already since 1918, each continuous symmetry is accompanied with a conserved quantity (Noether's theorem) [10]. For $U(1)$ this is indeed the electric charge.
The full quantum electrodynamic (QED) Lagrangian density is

$$\mathcal{L}_{\text{QED}} = \overline{\psi}(i\gamma^\mu D_\mu - m)\psi - \frac{1}{4}F_{\mu\nu}F^{\mu\nu} \tag{2.2}$$

$$= \overline{\psi}(i\gamma^\mu \partial_\mu - m)\psi - q(\overline{\psi}\gamma^\mu\psi)A_\mu - \frac{1}{4}F_{\mu\nu}F^{\mu\nu}\,, \tag{2.3}$$

where $F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu$ is the field strength tensor. Solely the last two terms are new in comparison to Equation 2.1. The third term is required for covering the dynamics of $A_\mu(x)$ while the second term describes the interaction between a fermion and the photon field coupled through the electric charge. Thus, it can indeed be seen that the electromagnetic interaction is a direct consequence of conservation of local gauge invariance. In particular, a mass term for the mediator would break gauge invariance, i.e., the mediator particle has to be massless, which is the case for the photon and results in an infinite range of the electromagnetic force. The infinite range neither holds for the strong nor the weak force.

For deriving the strong and the weak interaction, non-Abelian symmetry groups, i.e., groups that are not commutative, need to be introduced and potentially lead to more sophisticated fields. The strong force is described by Quantum Chromodynamics (QCD) and arises from the $SU(3)_C$ symmetry in color space (hence the subscript C), which builds upon eight generators $\tau^a = \frac{1}{2}\lambda^a$, with $a = 1..8$ and λ^a being the Gell-Mann matrices. Accordingly, there are eight fields A^a_μ in total, i.e., eight particles mediating the strong force. These mediators are called gluons and each of them possesses a color charge (mixture of red, green, blue and antired, antigreen and antiblue), which is, equally to the electric charge, a conserved quantity. The QCD Lagrangian density is given as

$$\mathcal{L}_{\text{QCD}} = \sum_q \left(\overline{\psi}_{q,i}(i\not{\partial} - m)\psi_{q,i} - g_{\text{S}}(\overline{\psi}_i\gamma^\mu\tau^a_{ij}\psi_j)A^a_\mu\right) - \frac{1}{4}F^a_{\mu\nu}F^{a,\mu\nu}\,, \tag{2.4}$$

with the color indices $i, j = 1, 2, 3$ and g_{S} denoting the strong coupling constant. The sum runs over all quark flavors q and the field strength tensor $F^a_{\mu\nu} = \partial_\mu A^a_\nu - \partial_\nu A^a_\mu + g_{\text{S}} f^{abc} A^b_\mu A^c_\nu$. The second term in the equation for the field strength tensor does not appear in QED as the structure constants f^{abc} equal zero for Abelian symmetry groups. This however enables self-interactions between the mediators (incorporated in the third term in Equation 2.4). Analogous to QED, the second term in Equation 2.4 represents the interaction between color charge carrying fermions (quarks) and gluon fields. A mass term for gluons would break the gauge invariance in QCD as well, resulting in massless gluons equally to photons. However, a peculiarity is still incorporated in QCD. With increasing energy/decreasing distances, the strength of the strong coupling constant decreases, while the opposite applies to decreasing energy/increasing distances. The consequences are asymptotic freedom, i.e., quarks and gluons are quasi free at high energies, and confinement, i.e., only color neutral objects exist. Accordingly, as of a certain distance between fundamental particles with color charge, it is energetically more favorable to create new particles while conserving color charge, called hadrons. Eventually, this hadronization process results in collimated bunches of hadrons forming a cone (jet).

The weak and electromagnetic force can be unified into the electroweak force (Glashow-Weinberg-Salam theory for high energies [11–13]), which stems from $SU(2)_L \times U(1)_Y$

symmetry on a theoretical level. The former gauge group comes with three generators $\tau^a = \frac{1}{2}\sigma^a$, where $a = 1, 2, 3$ and σ^a are the Pauli matrices, and therewith three gauge fields W^a_μ. As seen for QED already, the $U(1)_Y$ gauge group results in one postulated gauge boson, denoted B_μ. The observed quantities in this combined gauge group are the third component of the weak isospin I_3 and the hypercharge Y, which are connected through the Gell-Mann-Nishijima relation $I_3 = Y/2 - q$.
What is special about the weak force in particular is that it is confined to fermions with certain chirality. These are left-handed components of particles and right-handed components of antiparticles, hence the subscript L for $SU(2)_L$, which can be projected out of spinors with the projection operator $P_{\mathrm{R/L}} = (1 \pm \gamma^5)/2$, with $\gamma^5 \equiv i\gamma^0\gamma^1\gamma^2\gamma^3$. Eventually, this characteristic of the weak interaction leads to parity violation, i.e., no invariance of physics processes under point reflection, and violation of the combination of charge conjugation and parity symmetry (CP violation).
Furthermore, the physically observable mediators ($W^\pm_\mu$, Z_μ and A_μ) do not directly correspond to the postulated gauge fields, unlike for QCD or QED, but they are found to be linear combinations of the postulated gauge fields.
Moreover, the W and Z bosons are experimentally proven to be massive [14–16], which is however not straightforward obtainable from the conservation of the $SU(2)_L \times U(1)_Y$ symmetry. Even the mass of weakly interacting leptons poses a problem to the gauge invariance. Fortunately, the introduction of spontaneous symmetry breaking in the scope of the Brout-Englert-Higgs mechanism [17–19] solves this. In this concept, the Lagrangian density still possesses the $SU(2)_L \times U(1)_Y$ symmetry but the energy ground state of the physical system does not. This is induced with an additional complex scalar field (Higgs field)

$$\phi = \begin{pmatrix} \phi_+ \\ \phi_0 \end{pmatrix} \quad , \quad \phi^\dagger = \begin{pmatrix} \phi_+^* & \phi_0^* \end{pmatrix} \equiv \begin{pmatrix} \phi_- & \phi_0^* \end{pmatrix} , \tag{2.5}$$

which is an weak isospin doublet and obeys the $SU(2)_L \times U(1)_Y$ symmetry, and the corresponding Lagrangian density

$$\mathcal{L}_{\mathrm{Higgs}} = (D_\mu\phi)^\dagger(D^\mu\phi) - V(\phi) \tag{2.6}$$

$$= (D_\mu\phi)^\dagger(D^\mu\phi) - \mu^2\phi^\dagger\phi - \lambda\left(\phi^\dagger\phi\right)^2 . \tag{2.7}$$

As the components in ϕ are complex, four degrees of freedom are introduced with the Higgs field and the index of these components corresponds to their electric charge. For the parameters μ and λ, it holds $\mu^2 < 0$ and $\lambda > 0$ since otherwise the ground state of the potential is not degenerate and thus no spontaneous symmetry breaking is inducible. It can be derived that these parameters are related to each other via $v = \sqrt{-\mu^2/(2\lambda)}$, which is referred to as the non-zero vacuum expectation value of the Higgs field ϕ, i.e., the value of ϕ for which the Higgs potential is minimal (ground state). Choosing a specific ground state of the degenerate ground states, e.g., $\phi = \begin{pmatrix} 0 & v \end{pmatrix}^\mathsf{T}$ leads to the desired spontaneous symmetry breaking of $SU(2)_L \times U(1)_Y$. The consequence of a radial excitation of the Higgs field ϕ can be seen with a Taylor expansion around this ground state

$$\phi = \begin{pmatrix} 0 \\ v + \frac{H}{\sqrt{2}} \end{pmatrix} . \tag{2.8}$$

It results in a new field H, called the Higgs boson field. The postulated neutral scalar boson with the mass $m_H = \sqrt{4\lambda v^2}$ was eventually found in 2012 at CERN [1, 2] after almost 50 years of search and is known as Higgs boson.
The remaining degrees of freedom of ϕ are absorbed in W^+_μ, W^-_μ and Z_μ, which allows

Table 2.2: **Properties of fundamental bosons.** Both the mediator of the strong force and of the electromagnetic force are massless. The $W^{\pm}$ bosons are the only gauge bosons with charge and the Higgs boson is the only scalar boson (spin-0) in the Standard Model. Moreover, the latter is the only boson that does not serve as mediator for a force. The values are taken from Ref. [6].

boson name	mass	electric charge	spin	mediating
gluon (g)	—	0	1	strong force
photon (γ)	—	0	1	electromagnetic force
Z^0 boson	$(80.377 \pm 0.012)\,\mathrm{GeV}$	0	1	weak force
$W^{\pm}$ boson	$(91.1876 \pm 0.0021)\,\mathrm{GeV}$	± 1	1	weak force
Higgs boson (H)	$(125.25 \pm 0.17)\,\mathrm{GeV}$	0	0	—

them to be massive while the photon remains massless. The masses of the W and Z boson relate to v as follows

$$m_{\mathrm{W}} = \frac{g}{2}v \quad , \quad m_{\mathrm{Z}} = \frac{\sqrt{g^2+g'^2}}{2}v\,. \tag{2.9}$$

The fermion mass m_f issue is, on the other hand, solved with the Yukawa coupling between the fermion and the Higgs field

$$\mathcal{L}_{\mathrm{Yukawa}} = -y_f\left[\left(\overline{\psi}_{\mathrm{R}}\phi^{\dagger}\psi_{\mathrm{L}}\right) - \left(\overline{\psi}_{\mathrm{L}}\phi\psi_{\mathrm{R}}\right)\right] \tag{2.10}$$

with the coupling constant $y_f = m_f/v$.

In summary, all interactions (except for gravitation) arise from the local $SU(3)_C \times SU(2)_L \times U(1)_Y$ gauge symmetries to which the Standard Model is internally subject. The properties of the corresponding mediators and of the Higgs boson are summarized in Table 2.2.

2.2 Large Hadron Collider

As the name implies, the Large Hadron Collider is a hadron accelerator and collider at CERN. It is, until this day, the most powerful particle accelerator with respect to the provided center-of-mass energy $\sqrt{s}$ [20]. The LHC consists of two accelerator rings with a circumference of 26.7 km, which are utilized for either colliding protons or heavy (Pb) ions. There are four interaction regions, each with a dedicated experiment, as depicted in Fig. 2.1. These experiments are ALICE [21], which is dedicated to heavy ions, LHCb [22], which is mainly focused on B hadrons, as well as the high luminosity, multi-purpose experiments ATLAS [23] and CMS [24]. The latter is further presented in Section 2.3, as the data used in this thesis is simulated in correspondence to real data taken at the CMS experiment. [25]

A key property of colliders is called instantaneous luminosity

$$L \propto f\frac{N_{\mathrm{b}}^2 n_{\mathrm{b}}}{4\pi\sigma^2}\,, \tag{2.11}$$

with σ denoting the geometric cross section of the beams (with an assumed Gaussian profile), f denoting the revolution frequency of the bunches and N_{b} and n_{b} being the

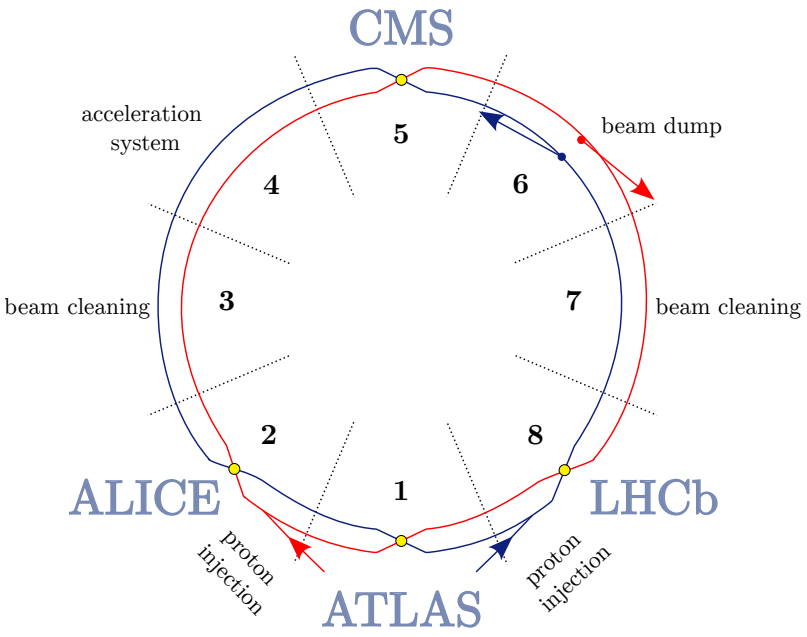


Figure 2.1: **Top view of a schematic representation of the LHC.** The LHC can be divided into eight octants. The clockwise proton beam (red) and the counterclockwise proton beam (blue) are inserted in the second and eighth octant of the LHC, respectively. The two beams are caused to collide at four points (yellow) which coincided with the position of the four experiments ATLAS, ALICE, CMS and LHCb. Figure adapted from Ref. [26].

number of particles in a bunch and the number of bunches, respectively. The reason for its importance is that it has a direct effect on

$$\frac{\mathrm{d}N_{\mathrm{event}}}{\mathrm{d}t} = L\sigma_{\mathrm{process}}\,, \tag{2.12}$$

the generated number of events per second. [25] The only other factor in Equation 2.12 is the cross section $\sigma_{\mathrm{process}}$ for the particular physics processes, which is a measure of the probability of the occurrence of a certain physics process and as such obviously constrained by physics and non-tunable.

As presented in Fig. 2.2, the data-taking at the LHC started in 2010 and operated for three years until 2013. During this first data taking period (LHC Run 1), an integrated luminosity of $L_{\mathrm{int}} = 25\,\mathrm{fb}^{-1}$ was reached at $\sqrt{s} = 7,8\,\mathrm{TeV}$ in proton-proton collisions. The integrated luminosity is thereby given as $L_{\mathrm{int}} = \int L\mathrm{d}t$ and serves as measure of the amount of generated data over a certain period of time. Among various physics highlights of this run, see Ref. [27], the discovery of the Higgs boson, simultaneously at CMS and ATLAS, stands out in particular. Although, this main goal of the LHC, which has been pursued since day one, is successfully achieved, the accelerator is not yet obsolete. For instance, despite of the experimentally verified predictive power of the SM in various experiments, there are, even apart from the gravitational force, many pending questions that are not covered by the SM. It fails to elucidate dark matter, which makes up 96 % of the matter content of the universe, or the baryon asymmetry, i.e., the matter-antimatter imbalance in the universe, to just name a few issues, and it is strived to answer them in the subsequent LHC runs. [27]
Thanks to various upgrades of the LHC and detectors during the first Long Shutdown (LS1) phase, the center-of-mass energy was doubled ($\sqrt{s} = 13\,\mathrm{TeV}$) for LHC Run 2 (2015–2018)

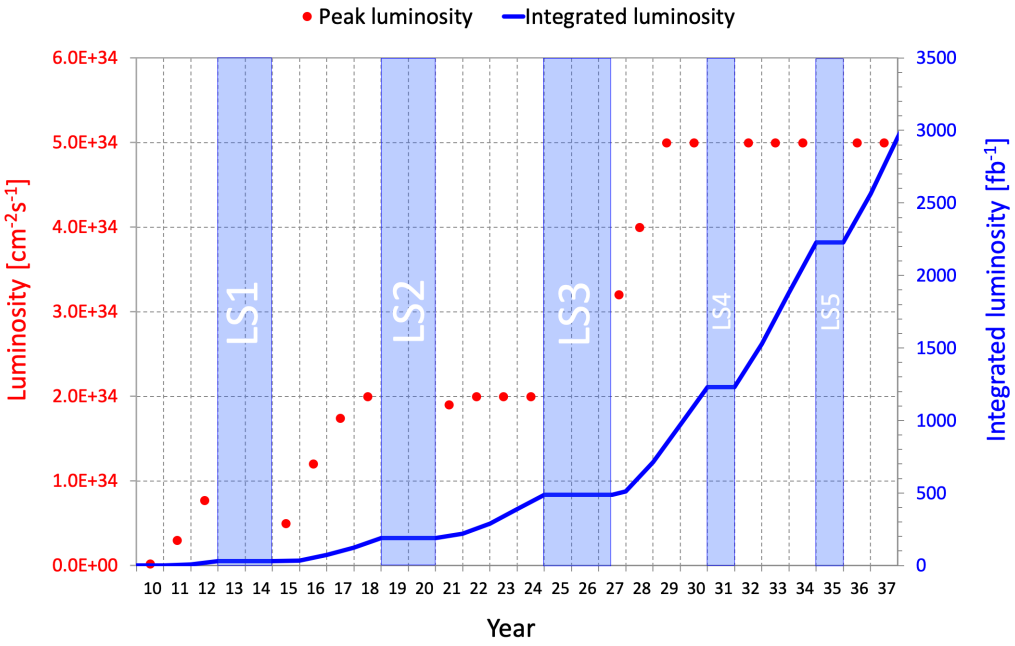


Figure 2.2: **Schedule of the LHC with the corresponding forecasts of the nominal instantaneous luminosity and integrated luminosity for each run.** So far, two data taking periods and two upgrade phases have been successfully completed. Unlike scheduled, Run 3 started in 2022 and not in 2021, partly due to the coronavirus pandemic [20]. It is aimed to achieve significantly higher luminosities (HL-LHC) in the upcoming data taking periods (after LS3). Figure taken from Ref. [31].

and the originally planned peak luminosity of $L = 10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$ [25] at the LHC has been exceeded in 2016 [28]. At the end of this period, the integrated luminosity in proton-proton collisions was in total about $160\,\mathrm{fb}^{-1}$ [28]. Due to this amount of data, e.g., the properties of the Higgs boson could be measured with greater precision on the one hand. On the other hand, also the exclusion limits on physics beyond the Standard Model could be further constrained [29]. After a second Long Shutdown with upgrades, the current data taking period (LHC Run 3, $\sqrt{s} = 13.6\,\mathrm{TeV}$) started in 2022 [30]. Yet, three further runs with a further upgraded LHC (High Luminosity LHC, HL-LHC) are already scheduled [20].

2.3 The CMS Detector

In the following, the Compact Muon Solenoid (CMS) detector will be elucidated on the basis of Ref. [24] if not stated otherwise.

The CMS detector is designed in a way that enables the detection of all types of particles (multi-purpose detector), i.e., it has to match some prerequisites like having a large hermetic geometric coverage, high energy and momentum resolution for photons, leptons, in particular muons, as well as hadrons, efficient online event filtering (trigger) and many more. The result is a detector composed of several layers of different sub-detectors arranged cylindrically around the interaction point and are designed in such a compact way that it is still (financially) feasible to fit the entire tracker system and calorimeters inside the magnet coil (Fig. 2.3). In total, the CMS detector only possesses a diameter of 14.6 m and a length of 21.6 m.

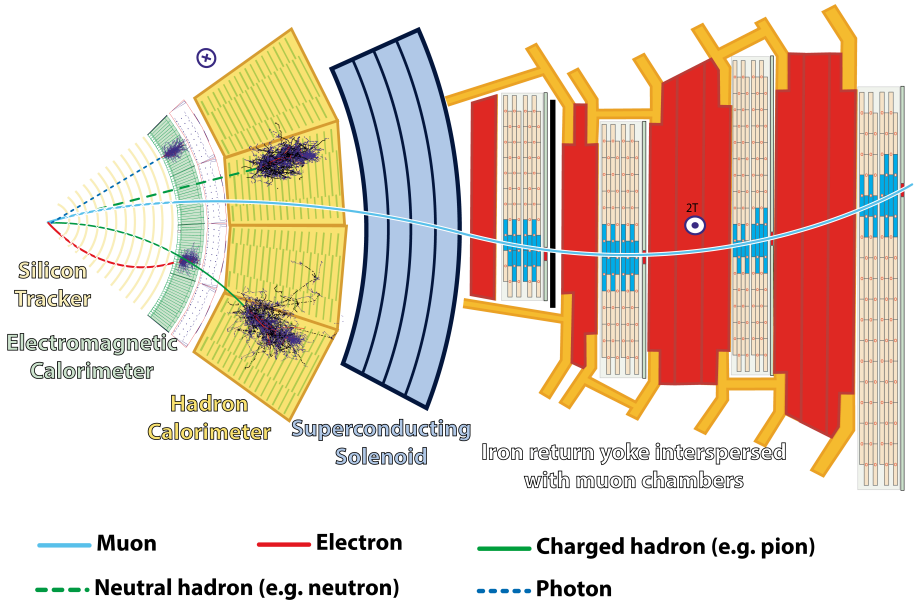


Figure 2.3: **Schematic representation of a slice of the CMS detector.** The CMS detector consists of four different sub-detectors. These are from the innermost to the outmost sub-detector, the silicon tracker, the electromagnetic and hadron calorimeters as well as the muon system. Additionally, exemplary traces of different particles through the CMS detector as well as their interaction with the sub-detectors are depicted. Figure taken from Ref. [32].

The **inner tracking system** is the innermost sub-detector and dedicated to vertex and track reconstruction of charged particles. It comprises several layers of pixel as well as silicon microstrip detectors in order to ensure a high granularity. This enables a precise measurement of the particle trajectories and the origin of particle tracks, which is indispensable due to the high luminosities reached at the LHC.
As the whole detector is penetrated by a homogeneous, almost 4-T magnetic field, provided by a superconducting solenoid, the tracks of charged particles are bent. Given

$$p_{\mathrm{T}} \propto q \cdot B \cdot R\,, \tag{2.13}$$

with q denoting the electric charge, B being the strength of the magnetic field and R the radius of the track curvature, the bent trajectories can be exploited for determining the transverse momentum p_{T} of the produced charged particles.

Subsequently, the **energy measuring systems** in the form of the electromagnetic calorimeter (ECAL) and the hadron calorimeter (HCAL) are positioned.
As a homogeneous calorimeter, the ECAL uses the same material ($PbWO_4$ crystals) for emitting scintillation light and absorbing the incident particles. With the scintillation light, it is then possible to measure the energy of electrons, positrons and photons. The ECAL energy resolution follows

$$\left(\frac{\sigma_E}{E}\right)^2 = \left(\frac{a}{\sqrt{E}}\right)^2 + \left(\frac{b}{E}\right)^2 + c\,, \tag{2.14}$$

whereby a, b and c are inherent calorimeter parameters that need to be determined with a beam test, for instance. The first term is known as the stochastic term, e.g., caused by statistical fluctuations, and the second term arises due to noise. With the energy

independent third term, for example, calibration, errors or energy leakage are taken into account.
On the contrary, the HCAL, surrounding the ECAL, is a sampling calorimeter. It is mainly dedicated to measuring the energy of hadrons, which have a longer interaction length than electrons, for example. The HCAL consists of alternating absorber (brass) and plastic scintillator layers, leading to lower achievable energy resolutions than with the ECAL. For that reason, the Particle Flow algorithm is introduced. With that, a higher energy resolution can be realized, as the algorithm fully exploits the advantageous conditions prevailing at CMS (large magnetic field, compact design, i.e., fewer dead spots, and each per se highly optimized sub-detector) in the best possible way by combining the measured information from all sub-detectors to determine the energy of particles (and track reconstruction) instead of solely relying on the HCAL for charged hadron [33].

The **muon system** as outmost sub-detector completes the CMS detector. It is composed of three different types of gaseous ionization detectors and has the purpose to exclusively measure both the momentum and charge of muons, as the only remaining particles that are able to pass through the entire construction without being absorbed are muons and the non-detectable neutrinos. Equally to the inner trackers, the momentum is measured via the bending radius of the muon tracks in a magnetic field. Hence, the muon system is embedded in the iron return yoke of the superconducting solenoid. Among all sub-detectors, the muon system is probably the most important tool for BSM searches as muons are comparatively easy to detect since they are not subject to strong interaction and suffer less from radiative losses than electrons.

An efficient **online trigger system** is also essential in order to cope with the vast amount of data produced at the LHC. Its goal is to select only the potentially interesting events produced out of approximately 10^9 proton-proton collisions per second at $L = 10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$ for storage and further analysis. The trigger system at the CMS consists of a hardware-based Level-1 (L1) trigger and a High-Level Trigger (HLT), which corresponds to a large computing farm. In the L1 trigger, only information provided by the calorimeters and the muon system serves as basis for the initial event selection. As the subsequent HLT then only needs to process a data rate of around 100 kHz, more sophisticated selection criteria, involving calculations with consideration of the complete detector information, are applicable. Eventually, only the much more feasible data rate of approximately 1 kHz [34] remain for offline processing.

2.4 Kinematic Quantities

For describing the direction and the kinematic quantities of the particles produced at LHC, it is reasonable to introduce a coordinate system whose origin is placed in the nominal interaction point, which coincides with the center of the CMS experiment. The corresponding x-axis is thereby directed towards the center of the LHC ring. The y-axis is perpendicular to it and points upwards and the z-axis coincides with the axis of the counterclockwise rotating beam in order to form a right-handed coordinate system. Being a cylindrical detector, using cylindrical coordinates (r, ϕ, θ) facilitates the description. The radial coordinate r lies in the plane spanned by x and y. The azimuthal angle ϕ corresponds to the angular distance from the x-axis in this plane and the polar angle θ is the angular distance from the z-axis in the (r, z)-plane. [24]

The **pseudorapidity**

$$\eta = -\ln \tan\left(\frac{\theta}{2}\right) \tag{2.15}$$

is however more commonly used for describing the direction of an object than the polar angle. In the limit of momentum p being significantly larger than the mass M of the corresponding object, which is ubiquitous in HEP, it is a good approximation of the rapidity

$$y = \frac{1}{2} \ln\left(\frac{E + p_z}{E - p_z}\right) , \tag{2.16}$$

which is a measure of the velocity of the particular object along the beam axis. The variable E in the equation denotes the energy of the particular object.

The **spatial distance**

$$\Delta R_{ij} = \sqrt{(\eta_i - \eta_j)^2 + (\phi_i - \phi_j)^2} \tag{2.17}$$

between two objects i and j is constructed on the basis of the pseudorapidity and azimuthal angle.

The **transverse momentum**

$$p_\mathrm{T} = \sqrt{p_x^2 + p_y^2} \tag{2.18}$$

is another important quantity. It is transverse with respect to the beam axis and as such also invariant under Lorentz transformations along the beam axis. Its importance is attributed to the fact that protons are not fundamental particles, i.e., actually not the proton as a whole but rather its substructures (partons) participate in the collisions. Thus, the actual momenta of the colliding partons along the beam axis are unknown.

As mentioned in Section 2.3, neutrinos (or neutral, at most weakly interacting, particles that are not described in the SM) are not detectable as they hardly interact with matter. Neutrinos do, however, possess momentum, which in turn also cannot be detected. The corresponding **missing transverse momentum** $\not{E}_\mathrm{T}$, which is also referred to as missing transverse energy (MET), can be reconstructed via the vectorial sum over the transverse momenta of all detected particles

$$\not{E}_\mathrm{T} = \left| -\sum_i \mathbf{p}_{\mathrm{T},i} \right| , \tag{2.19}$$

given there are no uninstrumental or ineffective areas in the detector through which particles can escape.

An **invariant mass** M_inv can be assigned to each system of objects. For that, it is necessary to sum over the four momenta of all objects. The invariant mass then reads as

$$M = \sqrt{E^2 - p_\mathrm{T}^2 - p_z^2} , \tag{2.20}$$

with E, p_T and p_z denoting the energy, the transverse momentum and the momentum along the beam axis of the particular system of objects, respectively. In this manner, a mass can also be assigned to jets, which is however referred to as jet mass and not as invariant mass.

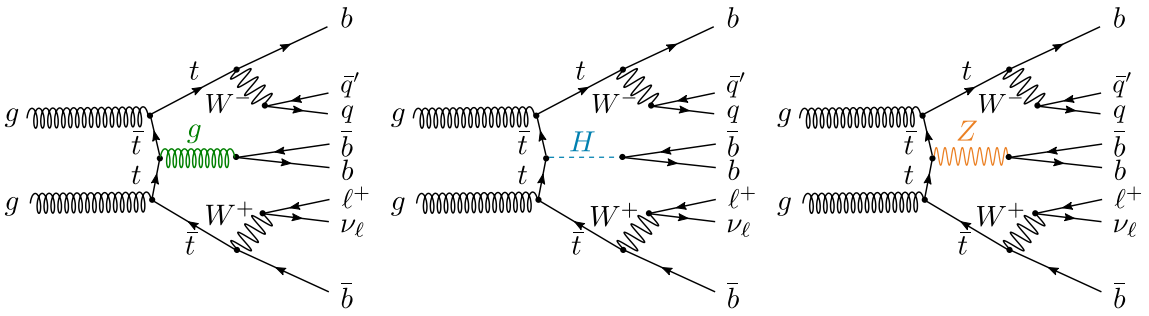


Figure 2.4: **Exemplary LO Feynman diagrams of $t\bar{t} + b\bar{b}$ (a), $t\bar{t}H(b\bar{b})$ (b) and $t\bar{t}Z(b\bar{b})$ (c) in the t-channel.** All three processes share the exact same final state objects and hence are irreducible backgrounds to each other. The only difference in their event topology is highlighted in color, namely the particle from which b jets that do not directly stem from a top decay originate.

2.5 Physics Processes

Being the heaviest fermions in the Standard Model and only possessing a lifetime of $5 \cdot 10^{-25}$ s, top quarks behave significantly different than other quarks. For instance, no bound-states can be formed with these short-lived quarks ($t\bar{t}$-quarkonium) and instead of producing jets, as other strongly interacting particles do due to confinement (cf. Section 2.1), they decay into a W boson and a b quark with a branching ratio $\mathcal{B}$ of $(99.7 \pm 1.5)\,\%$. Hence, it is only reasonable to classify top decays with respect to the decay products of the W boson. For a $t\bar{t}$ pair, three decaying scenarios exist with different associated branching ratios. In the all-hadronic channel ($\mathcal{B} = 45.7\,\%$), both W bosons decay into $q\bar{q}'$ pairs, which in turn produce jets. In the single-lepton channel ($\mathcal{B} = 43.8\,\%$), only one W boson decays hadronically and the other W boson decays into a charged lepton and a neutrino. Accordingly, both W bosons decay leptonically in the dilepton channel ($\mathcal{B} = 10.5\,\%$). [6]

The physics focus of this thesis is on the single-lepton channel of $t\bar{t} + b\bar{b}$ processes as it combines both a large branching ratio and a clear signature (due to the charged lepton). Moreover, it suffers less from large QCD backgrounds. An exemplary Feynman diagram at leading order (LO) of QCD perturbation theory for this process is depicted in Fig. 2.4a. In total, six jets, one charged lepton and one neutrino are produced at LO. In this thesis, the neutrino counts as a final state object as well, leading to eight final state objects at LO. The b jets in the event name thereby refers to the b jets that stem from gluon decay. As they are not regarded as part of the $t\bar{t}$ system, these jets are considered additional jets.

The motivation behind studying $t\bar{t} + b\bar{b}$ is that there are still big uncertainties in mathematically describing and therefore also simulating this process. This traces back to the large top quark mass, which has the consequence that the production of $t\bar{t}$ and the associated b jets in $t\bar{t} + b\bar{b}$ events take place on completely different energy scales [3]. On the other hand, the $t\bar{t} + b\bar{b}$ process is of outstanding importance from a broader perspective as well since it is an irreducible background to the $t\bar{t}H(b\bar{b})$ process (Fig. 2.4b), in which $t\bar{t}$ pairs are produced in association with a Higgs boson ($t\bar{t}$ + H) combined with a subsequent decay of the Higgs boson in a $b\bar{b}$ pair ($H \to b\bar{b}$). The $t\bar{t}H(b\bar{b})$ process is, in turn, a promising process for measuring the coupling strength of the Yukawa coupling to the top quark y_t as $y_t \sim m_t$ holds (cf. Section 2.1), i.e., the large top quark mass is, in this case, indeed advantageous and leads to a coupling strength of order unity [6]. This allows both the consistency of the Standard Model and BSM models predicting a different coupling strength to be probed [4]. But also the $t\bar{t}Z(b\bar{b})$ process ($t\bar{t}$ + Z, $Z \to b\bar{b}$) is an irreducible background

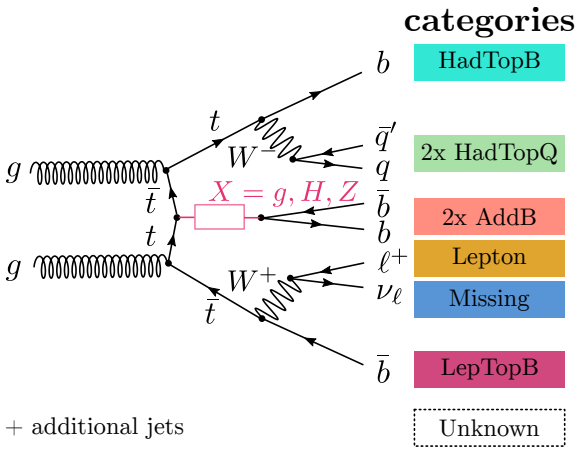


Figure 2.5: **Category assignment.** In total, seven different categories are introduced. All jets that are neither part of the $t\bar{t}$ system nor an additional b jet, i.e., the b jets from gluon, Higgs or Z boson decay, are assigned to the category Unknown.

to $t\bar{t} + b\bar{b}$ and $t\bar{t}H(b\bar{b})$ processes (Fig. 2.4c). In order to separate these $t\bar{t}$+X events, mainly graph neural networks are deployed in this thesis. In a binary classification, $t\bar{t} + b\bar{b}$ is treated as signal events while $t\bar{t}H(b\bar{b})$ and $t\bar{t}Z(b\bar{b})$ are summarized as background events.

To facilitate the differentiation between the final state objects, seven categories are introduced. As showcased in Fig. 2.5, b jets that are produced in association with the hadronically/leptonically decaying W bosons correspond to the categories HadTopB and LepTopB, respectively. The jets emerging from the hadronically decaying W boson are assigned to the category HadTopQ and the categories Lepton and Missing are assigned to the decay products of the leptonically decaying W boson. Jets that are not included at leading order, are aggregated in the category Unknown. While, strictly speaking, only b jets that stem from a radiated gluon decay are real additional b jets, the b jets resulting from the decay of the Higgs or Z boson, are equally assigned to the category AddB in this thesis.

2.6 Jet Reconstruction and b Tagging

It already becomes evident from the Feynman diagrams of the presented $t\bar{t}$+X events (cf. Fig. 2.4) that b jets play a crucial role for the studies performed in this thesis and hence it is important to identify them correctly.

At the LHC, jets are generally reconstructed via the anti-k_T algorithm ($k_T := p_T$) [35], which is both infrared and collinear safe. That is, the reconstruction of the jet is neither influenced by the radiation of low-energy gluons (soft gluons) nor gluons radiated at small angles. The distance measure for a pair of objects i, j

$$d_{ij} = \min\left(\frac{1}{p_{T,i}^2}, \frac{1}{p_{T,j}^2}\right) \frac{\Delta R_{ij}^2}{R^2}, \tag{2.21}$$

with R being a radius parameter (usually $R = 0.4$), and the distance measure between an object i and the beam B

$$d_{iB} = p_{T,i}^{-2} \tag{2.22}$$

form the core of this algorithm. The algorithm sequentially combines in each iteration the pair of objects (particles or pseudoparticles) with the smallest d_{ij} to a new pseudoparticle until $d_{ij} < d_{i\mathrm{B}}$ does not hold anymore. All particles within the same pseudoparticle are constituents of the same reconstructed jet.

For identifying b jets out of the reconstructed jets, the DeepJet b tagging algorithm, which is based on a deep neural network (DNN), is utilized [36]. The DNN exploits, for instance, properties of the jet constituents as well as secondary vertices related information for its predictions. Especially the latter is relevant for b tagging as secondary vertices usually stem from the exceptionally long lifetime of b hadrons in the order of picoseconds [6]. This results in their production and decay points not coinciding. The DNN response for a jet, which ranges from zero to one, is referred to as b tag and the threshold (working point) at which a jet is regarded as b jet depends on the accepted rate of confusing light flavor jets with actual b jets. The working point for which 10 %, 1 % and 0.1 % of the light flavor jets are misidentified as b jets are referred to as loose, medium and tight working points, respectively. [36]
Here, the medium working point is chosen, which corresponds to a DNN response value of 0.304 for data taken in the year 2017, i.e., only jets with a b tag exceeding this value are identified as b jets.

3 Machine Learning Algorithms

Thanks to technological advances more sophisticated detectors can be constructed and as ever-increasing computation power can be used for analyzing the big HEP data detected by these detectors. In order to cope with this vast amount of data, often machine learning techniques are employed. While, at the beginning, rather simple methods were applied to HEP data, there is now rather a trend towards using more complex, i.e., deeper, machine learning algorithms such as deep neural networks. In general, machine learning algorithms are extremely versatile and can be used for various HEP tasks, including reconstruction of tracks in detectors, event classification, jet tagging, triggering and many more. [37, 38]
The machine learning algorithms used in this thesis are both deep neural networks and the novel, yet increasingly popular, graph neural networks, introduced in Section 3.1 and Section 3.2. In Section 3.3, two explainable AI methods, which are useful for gaining a deeper understanding of the behavior of neural networks, are introduced.

3.1 Deep Neural Networks

The following explanations are based on Refs. [39–41] if not stated otherwise.

An artificial neural network (ANN) comprises a set of neurons N, organized in layers and connected via edges E with each other. The first and the last layer of an ANN is referred to as input layer I and output layer O, respectively. Layers that are in between the input and output layer are called hidden layers. Thus, an ANN can be fully described via the tuple (I, N, O, E). Figure 3.1 depicts a generic ANN consisting of the input layer, one hidden layer and the output layer.
This thesis is confined to feed-forward neural networks, meaning that only neural networks which pass the data strictly from left-to-right (input layer $\rightarrow$ optional hidden layer(s) $\rightarrow$ output layer) in the forward pass are considered. Accordingly, loops or, for instance, intra-layer connections are prohibited. Moreover, the ANNs are fully-connected, i.e., each node of a layer is connected to all other nodes of the subsequent layer and each connection is associated with a (trainable) scalar, the weight.

The input layer can be mathematically described with a vector (input vector)

$$\mathbf{x} = \begin{pmatrix} x_1 & x_2 & \dots & x_m \end{pmatrix}^\intercal \tag{3.1}$$

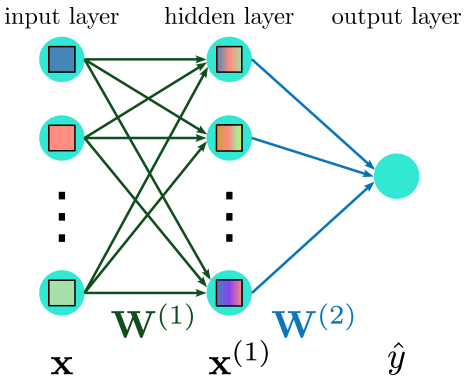


Figure 3.1: **Generic ANN consisting of one input, one hidden and one output layer.** Bias nodes are not depicted for the sake of simplicity. The single-colored rectangles exemplify the input features $\mathbf{x}$ and the multi-colored rectangles indicate the change of these feature values after the linear transformation of the inputs (cf. Equation 3.3).

whereas the weights of each layer l can be described with a matrix (weight matrix)

$$\mathbf{W}^{(l)} = \begin{pmatrix} w_{11}^{(l)} & w_{12}^{(l)} & \dots & w_{1d'}^{(l)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{d1}^{(l)} & w_{d2}^{(l)} & \dots & w_{dd'}^{(l)} \end{pmatrix} . \tag{3.2}$$

Thereby, $x_i, i \in \{1, 2, \dots, m\}$ denotes the features provided to the neural network and d and d' denote the number of nodes in a layer l and the number of nodes in the previous layer $l-1$, respectively. The computation performed in each non-input layer l corresponds to

$$\mathbf{x}^{(l)} = f(\mathbf{W}^{(l)}\mathbf{x}^{(l-1)} + \mathbf{b}^{(l)}) , \tag{3.3}$$

where $\mathbf{x}^{(l-1)}$ equals $\mathbf{x}$ and d equals m for $l = 1$ and f denotes an activation function applied elementwise on its argument. Optionally, a so-called bias $\mathbf{b} \in \mathbb{R}^d$ can be additionally applied to each neuron in each hidden and output layer. What is special about the bias is that the associated value is always the constant 1. The purpose of the bias is to shift the activation function along the x-axis. The activation function used after each hidden layer in this thesis is ReLU [42, 43]

$$f_{\text{ReLU}}(\mathbf{x}^{(l)}) = \max(0, \mathbf{x}^{(l)}) \tag{3.4}$$

as it is, despite of its simplicity, still often the function of choice in state-of-the-art deep learning architectures [44]. On the contrary, SIGMOID and its generalized version SOFTMAX

$$\hat{\mathbf{y}} := f_{\text{SOFTMAX}}(\mathbf{x}^{(l)}) = \frac{\exp(\mathbf{x}^{(l)})}{\sum_{k=1}^{K} \exp(\mathbf{x}^{(l)})_k} , \tag{3.5}$$

are applied after the output layer for binary and multiclass classification, respectively, since this allows a probability interpretation of the neural network response (logits). However, it should be noted that only one neuron is used in the output layer for the binary classification instead of multiple neurons. Hence, a value closer to zero indicates a rather background-like event and a value closer to one rather a signal-like event.

Consequently, an ANN is nothing else than a function composition $\Phi : \mathbb{R}^m \to \mathbb{R}^o$ mapping the input vector of the length m to an output vector of the length o. The target is

to find the set of weights, so that the ANN is most suitable for solving the given task (training), i.e., approximating the underlying true function f as best as possible. However, only P examples of the true function f are known and given as input to the ANN. In supervised learning the target of the ANN's output is known beforehand. Thus, these examples (training set) correspond to a set of tuples $\{(\mathbf{x}_1, \mathbf{t}_1), \ldots, (\mathbf{x}_P, \mathbf{t}_P)\}$, where $\mathbf{t}_i$, $i \in \{1, 2, \ldots, P\}$ denotes the targeted output of the ANN for the i-th example. In order to train the ANN on the given tasks, i.e., output the desired targets, a loss function $\mathcal{L}$ measuring the deviation between the outputs $\hat{\mathbf{y}}$ and the targets as well as an optimization algorithm for finding the (global) minimum of $\mathcal{L}$ are required. If the latter is found, then the model is converged.
A widely used loss function for classification tasks is called CATEGORICAL CROSS-ENTROPY, which is called BINARY CROSS-ENTROPY when applied to binary classification tasks and which can be additionally scaled with a factor λ. The scaled loss of an example i is calculated as

$$\mathcal{L}_i = -\lambda_i \sum_{j=1}^{o} t_{ij} \log(\hat{y}_{ij}) \,. \tag{3.6}$$

In this thesis, the factor λ corresponds to the inverse of the sample size of the class of the processed example.
A simple approach for optimizing the weights on the basis of the calculated loss is gradient descent. This algorithm updates the weights through

$$w_{dd'}^{(l)\prime} \leftarrow w_{dd'}^{(l)} - \gamma \frac{\partial \mathcal{L}}{\partial w_{dd'}^{(l)}} \,, \tag{3.7}$$

with γ being the learning rate. This is a hyperparameter that is chosen prior to the training and controls how much the weights are updated at once. Due to the necessity for calculating the gradients of $\mathcal{L}$, the loss function must be continuous and differentiable. Incidentally, it is rather computationally inefficient to update the weights only after having propagated the whole training set. It is faster to only calculate an estimation of the true gradient of $\mathcal{L}$ by updating the weights after the propagation of a mini-batch, i.e., already after the propagation of a sub-group of the full training samples. Accordingly, this approach is named stochastic gradient descent. However, this method can be even further improved by replacing the constant learning rate by an adaptive learning rate during training. This more sophisticated approach is used in the form of ADAM in this thesis, as ADAM appears to outperform other adaptive learning methods such as ADAGRAD [45]. After having propagated the entire training set through the ANN and having updated the weights, one training step is completed.

Since the training set has a limited size and the presence of noise is inevitable, there is the possibility that the neural network starts to memorize this unwanted noise during training instead of learning the underlying concept of the data (overfitting). Overfitting leads to the consequence that a ANN performs well on the training data but will perform poorly on unseen data. A possible solution for this issue is EARLY-STOPPING, for example. Here, the data is divided into three subsets, a training, validation and test set. In order to monitor when overfitting begins, the neural network is evaluated on the validation set after each training step. As soon as the monitored metric, such as the validation loss or the validation accuracy, on the validation set has not improved for a predefined number of epochs, the training is stopped. Afterwards, the neural network can be evaluated on the test set in order to get the final performance metrics of the trained model. [46]
Apart from EARLY-STOPPING, there are many regularization techniques for addressing the

overfitting issue such as L2 REGULARIZATION or DROPOUT. The first approach modifies the loss function by adding a term to it penalizing big weights. The latter influences the learning process by randomly turning a fraction of hidden nodes in the neural network temporarily off and therewith each neuron needs to focus on robust features. With DROPOUT it is basically possible to train several neural networks at once. The model inference step, where the dropout is omitted, profits from that since due the indirect averaging of the model responses, it is expected that the differently pronounced overfitting in each model is neutralized, leading to a higher model performance. [41]

Depending on the source, different definitions apply to when an ANN is considered as a deep neural network. This thesis follows the definition of the authors in Refs. [40, 41], who consider a neural network with at least two hidden layers as a deep neural network.
Using more hidden layers should support the neural network at learning more abstract concepts of the data. There is empirical evidence that multi-layer neural networks are more powerful than single-layer neural networks, although a single-layer neural network can theoretically provide the same results as a multi-layer neural network. This is shown in Ref. [41], for instance.
Since the number of trainable parameters increases with each additional hidden layer, an effective algorithm for optimizing the trainable weights is required. The backpropagation algorithm fulfills this as, in this algorithm, the gradients are calculated backwards for each layer starting from the output layer until the input layer is reached (backward pass). Thus, the desired gradients can all be simultaneously calculated with only one backward pass (after the forward pass) instead of calculating each gradient separately.

3.2 Graph Neural Networks

Deep neural networks are only applicable to data represented as vectors. In some domains, vectors are however not the best choice for representing the data. Instead, graphs turned out to be a more natural way for representing many real-world data. Presumably due to the increasing importance of graph data, graph neural networks (GNNs), which are particularly designed for processing this kind of data, gained more popularity in recent years. Moreover, as stated in Ref. [38], there are many examples showcasing that GNNs outperform other methods. [38, 47]

3.2.1 Graph Theory

The subsequent overview over the graph theory follows Refs. [48, 49] if not stated otherwise.

A graph G is defined as a tuple (V, E) of a set of vertices $V = V(G)$ and a set of edges $E = E(G)$. Vertices are also commonly referred to as nodes. Due to the ambiguity with the nodes in an ANN, the term vertex/vertices is predominantly used in this thesis when referring to nodes in a graph. The vertex set is finite and non-empty whereas the edge set may be empty and otherwise consists of pairs $e = (u, v), u, v \in V$. If a connection, i.e., an edge, between the vertices u and v exists, u and v are called adjacent to one another and they are incident with the associated edge. In general, graphs having multiple edges between the same pair of vertices (skeins) is not precluded. Nevertheless, such graphs are not considered here. Being undirected and, if E is not empty, complete are the only constraints on the graphs examined in this thesis. This means that the edges must not possess any direction and each vertex is connected with all other vertices. Loops, i.e., edges $e = (u, v)$ with $u = v \in V$, are in fact allowed, yet not necessary for a graph to be

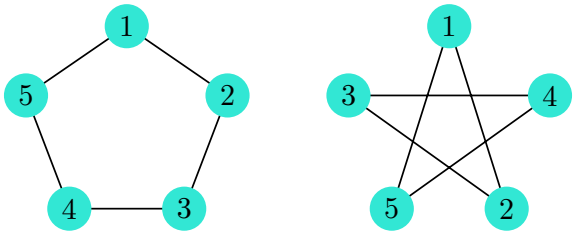


Figure 3.2: **Different representations of the same graph.** Despite the different representations, the left and the right graph are considered the same graph since the mathematical expression for both is the exact same. Figure adapted from Ref. [48].

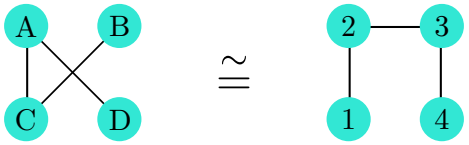


Figure 3.3: **Example of an isomorphic graph.** Figure adapted from Ref. [49].

considered as complete.

The vertices and edges of G are usually illustrated as a graph diagram, which consists of dots as well as lines connecting the particular adjacent dots. However, it should be noted that many representations can exist for the exact same graph, as shown in Fig. 3.2. These graphs are considered equal since their vertex set and edge set are equal, independently of permutations of the elements in the sets. Given this definition, two graphs $G = (V, E)$ and $G' = (V', E')$ with, for instance, solely different naming of their vertices are not considered equal anymore. But, if it is possible to define a one-to-one correspondence $g : V \to V'$ for the two graphs in question with

$$(u, v) \in E \Leftrightarrow (g(u), g(v)) \in E' \,, \tag{3.8}$$

then G and G' are referred to as isomorphic (cf. Fig. 3.3).

Although the graph diagrams are visually appealing for humans, they are not a handy format for computers. For the latter, it is more useful to consecutively number the vertices of G and represent a graph via an adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, where $n = |V|$. The elements a_{ij} of $\mathbf{A}$ are either one or zero depending on whether the vertices $i, j \in \{1, 2, \ldots, n\}$ are adjacent or not. Since only undirected graphs are considered here, the adjacency matrix is symmetric without any exceptions. The adjacency matrix for the complete graphs depicted in Fig. 3.2 is

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix} . \tag{3.9}$$

If $\mathbf{A}$ is sparse, adjacency matrices are not an efficient way to store the graph anymore. For a comparison of different storing methods, see Ref. [50], for instance.

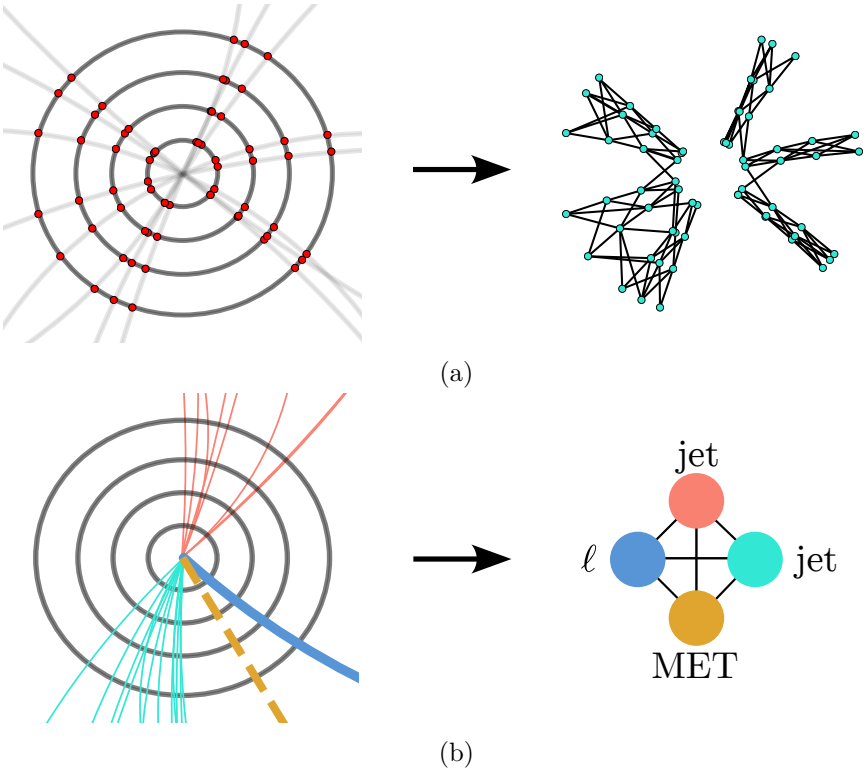


Figure 3.4: **Examples of HEP data represented as graphs.** A possible transformation of detector hits into a graph representation is depicted in (a). In (b), a transformation of detected physics objects of an illustrative event into a complete graph is shown. Figure adapted from Ref. [38].

3.2.2 Graph Data and Tasks

The following descriptions are based on Refs. [38, 47] if not noted otherwise.

Graphs are an equally powerful and natural data representation for many real-world applications in engineering or science, especially including HEP. Daily-life examples of data suitable for being represented in a graph structure are social relationships or traffic on roads. Examples of HEP data represented as graphs are depicted in Fig. 3.4.

One advantage of dealing with graph data is the possibility to assign three different types of attributes to it, node-level-attributes

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_n \end{pmatrix}^\intercal , \tag{3.10}$$

edge-level-attributes

$$\tilde{\mathbf{A}} = \begin{pmatrix} \tilde{\mathbf{a}}_{11} & \dots & \tilde{\mathbf{a}}_{1n} \\ \vdots & \ddots & \vdots \\ \tilde{\mathbf{a}}_{n1} & \dots & \tilde{\mathbf{a}}_{nn} \end{pmatrix} \tag{3.11}$$

and global-level attributes $\mathbf{g}$ (cf. Fig. 3.5). The edge-level-attributes are denoted $\tilde{\mathbf{A}}$ to show the close relationship to the adjacency matrix $\mathbf{A}$, as it is already sufficient to replace the entries of $\mathbf{A}$ by the corresponding edge-level-attributes.
Due to the rich information preserved in these attributed graphs, there are in general many tasks that can be accomplished with GNNs. These include node classification (also

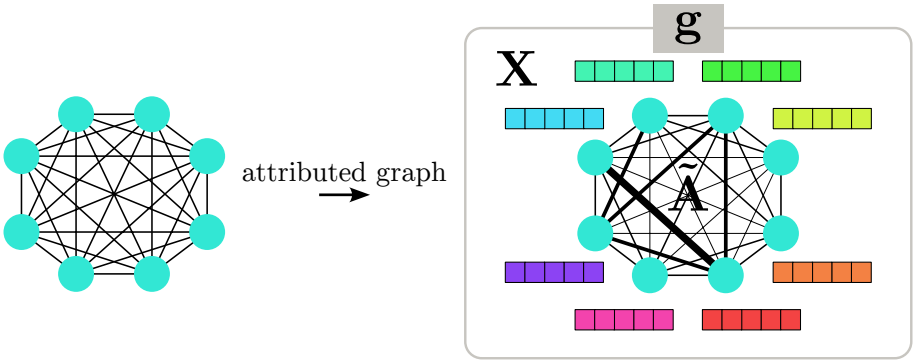


Figure 3.5: **Visualization of an attributed graph.** An attributed graph can comprise vertex attributes $\mathbf{X}$, edge attributes $\tilde{\mathbf{A}}$ as well as global attributes $\mathbf{g}$. In this thesis the colored rectangles represent the attributes of the vertices and the thickness of the edges indicates the different edge attributes.

node-level-prediction, (NLP)), relational prediction (also edge-level-prediction, (ELP)) or graph classification (also graph-level-prediction, (GLP)), just to name a few. The goal of ELP is to infer information about the connections between a set of vertices in a given, partially connected graph. A possible application of that is the reconstruction of tracks in a detector, as shown in Fig. 3.6a. By contrast, in NLP, a graph with partially labeled and partially unlabeled vertices is given. The goal is to predict the label of the unlabeled vertices based on the given graph, i.e., the relational information to the other, labeled, vertices in the graph (cf. Fig. 3.6b). In GLP, the focus of interest is to predict a label for the graph as a whole. An exemplary application of GLP is shown in Fig. 3.6c, where the class of event to which the graph belong needs to be determined (event classification). However, it should be stated that these three tasks are not associated with the same conventional machine learning categories. Strictly speaking, NLP is rather regarded as semi-supervised learning since there are no unlabeled data points in supervised learning and ELP is both supervised and unsupervised learning. Only GLP can be considered a real supervised learning task. Moreover, it is assured in the latter task that each data point, i.e., each graph in this case, is independent and identically distributed (i.i.d.). The i.i.d. assumption, which is usually demanded in machine learning models, as this otherwise complicates the training and harms the generalization abilities of trained models, does not apply to NLP. Quite the contrary, the relationships between the vertices in a graph is intentionally exploited in NLP. In fact, this is often the main factor for the success of GNN-based approaches for node classification.

3.2.3 Message Passing Neural Networks

This section is as well based on Refs. [38, 47] if not noted otherwise.

Graph neural networks are graph-to-graph functions, meaning they take a graph as input and update it either way but without changing the vertex and edge structure of the original input graph. Throughout the years, several types of GNNs have been evolved for various problem domains [52]. The key feature of many of these GNNs is called message passing between adjacent vertices in the graph and they are embraced in the Message Passing Neural Network (MPNN) framework, introduced in Ref. [53].
The message passing is used for updating the features associated with the vertices in a graph based on the current features associated with the vertices (embedding) as well as the aggregated information (messages) from the particular graph neighborhoods. The

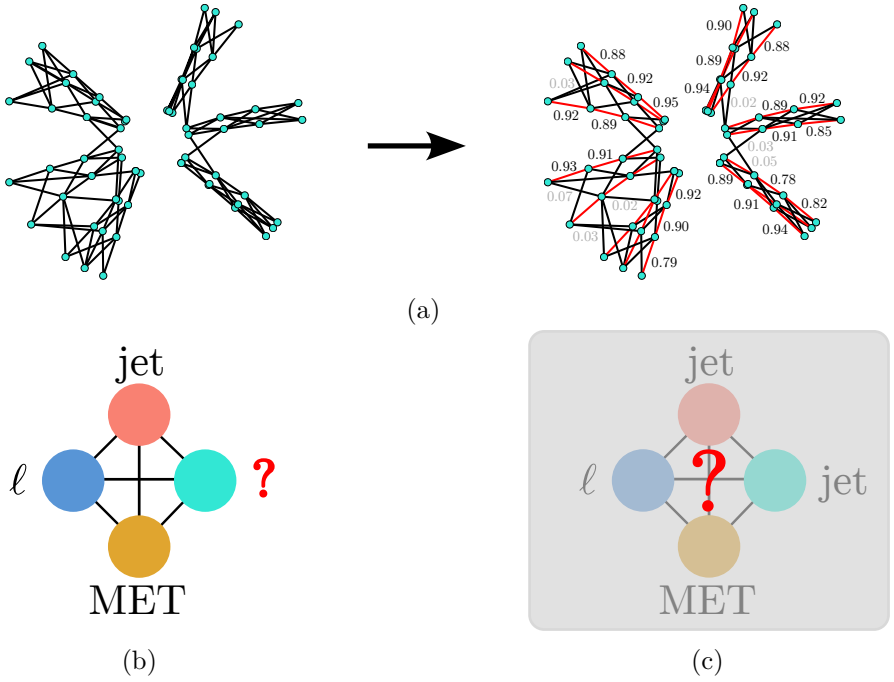


Figure 3.6: **Different tasks performed on the graphs shown in Fig. 3.4.** In (a), track reconstruction using edge-level-prediction is depicted. Edges with high prediction scores are seen as part of a track (red). The graphs at the bottom represent an event and can be used for jet tagging via node-level-prediction (b) or for classifying the event as a whole via graph-level-prediction (c). Figures adapted from Ref. [38] and Ref. [51].

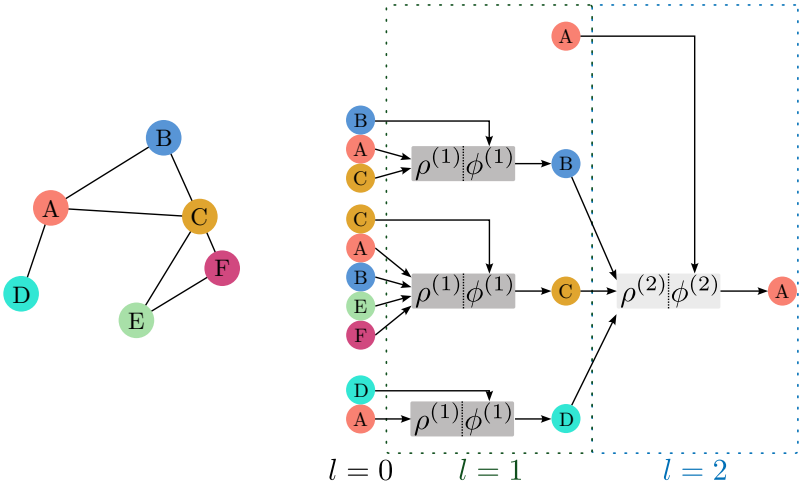


Figure 3.7: **Message passing.** The graph on which the message passing step $l = 2$ for its vertex A is based is shown on the left. The message passing step is depicted in an unfolded manner on the right, which is also called the computation graph of vertex A. For clarity reasons, the colored circles denote the corresponding vertex attributes that need to be passed to the particular aggregation functions ρ instead of the colored rectangles in previous figures. Figure adapted from Ref. [47].

graph neighborhood of the vertex u is the subgraph being formed when only the edges associated with u including the vertices that are adjacent to u are considered. Accordingly, the updated hidden embedding $\mathbf{x}_u^{(l)}$ of u after the l-th message passing iteration can be expressed by

$$\mathbf{x}_u^{(l)} = \phi^{(l)}\left(\mathbf{x}_u^{(l-1)}, \rho^{(l)}\left(\left\{\mathbf{x}_u^{(l-1)}, \forall v \in \mathcal{N}(u)\right\}\right)\right) \tag{3.12}$$

$$= \phi^{(l)}\left(\mathbf{x}_u^{(l-1)}, \mathbf{m}^{(l)}\right), \tag{3.13}$$

where ϕ and ρ represent arbitrary differentiable update and aggregation functions, respectively, $\mathcal{N}(u)$ denotes the graph neighborhood of u and $\mathbf{m}$ represents the messages. The update function is shared across all vertices in the graph, which is referred to as parameter sharing. It should also be noted that the aggregation function is a set function, i.e., it must be a function whose output is independent of the order of the set elements in its input. This is important as otherwise the aforementioned permutation invariance of graphs (cf. Section 3.2.1) cannot be preserved.

The message passing step is performed L times. At iteration $l = 1$, $\mathbf{x}_u^{(l-1)}$ equals the initial embeddings, i.e., the input features/vertex attributes $\mathbf{x}_u$ of u. Each iteration produces another updated hidden embedding, in the same way as each further hidden layer in an ANN produces a new updated hidden output $\mathbf{x}^{(l)}$ (cf. Equation 3.3). Therefore, each message passing iteration can be regarded as another layer of the GNN. This is especially visible when the aggregated information in the l-th iteration for u is drawn in an unfolded manner, as visualized in Fig. 3.7 for vertex A and $l = 2$.

Moreover, the aggregation step is of particular importance since by aggregating the information of the neighborhood the graph structure is taken into consideration, which is the main reason for dealing with graph data. The idea behind the iterations is that each vertex u in the graph is able to collect information from vertices being one-hop further away from u than the previous furthest vertices. That is, the vertex receptive field of u

increases and it can "receive" a greater part of the graph with each iteration.

The type defining factor for the different GNNs within the MPNN framework only lies in the different implementations of the update and/or aggregation function in Equation 3.12.

Graph Convolutional Neural Networks (GCN), firstly introduced in Ref. [54], are one of the most common and effective GNNs. They use a symmetrically normalized sum as aggregation function and redundantize the update function by introducing (self-)loops, i.e., they do not only aggregate the information of the neighbors of the vertex u but the embedding of u as well. Accordingly, Equation 3.12 can now be expressed by

$$\mathbf{x}_u^{(l)} = f\left(\mathbf{W}^{(l)} \sum_{v \in \mathcal{N}(u) \cup \{u\}} \frac{\mathbf{x}_v^{(l-1)}}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}} + \mathbf{b}^{(l)}\right), \tag{3.14}$$

with $\mathbf{W}^{(l)} \in \mathbb{R}^{d \times d'}$ denoting trainable parameter matrices, i.e., an ANN, $\mathbf{b}^{(l)} \in \mathbb{R}^d$ denoting the added bias and $|\mathcal{N}(u)|, |\mathcal{N}(v)|$ denoting the degree of u and v, respectively, i.e., the number of adjacent vertices. As in Equation 3.3, f is an arbitrary activation function applied elementwise to its argument.
However, using (self-)loops inevitably leads to the disadvantage that information coming from the neighborhood of u is not distinguishable anymore from the information coming directly from u. On the one hand, this harms the expressivity of the model and on the other hand, this facilitates an unwanted over-smoothing. Over-smoothing addresses the problem that the hidden embeddings of the vertices in a graph resemble each other more the more message passing iterations are performed. Thereby, information about the graph structure in the neighborhood of u are lost. Since the number of iterations can be understood as the number of GNN layers, over-smoothing makes it difficult to build deeper GNNs.

Instead of using ANNs in the message passing as for the GCNs, update functions from recurrent neural network (RNN) architectures can also be used. For a detailed explanations of RNNs in general, refer to Ref. [55], for instance.
The update function used in Gated Graph (Sequence) Neural Networks (GGSNN), introduced in Ref. [56], is the Gated Recurrent Unit (GRU) cell (see Ref. [57]). Thus, Equation 3.12 becomes

$$\mathbf{x}_u^{(l)} = \mathrm{GRU}(\mathbf{x}_u^{(l-1)}, \mathbf{m}^{(l)}). \tag{3.15}$$

Indeed this type of neural networks comes with a much higher number of trainable parameters in comparison to a GCN of the same depth due to its more sophisticated update function, but utilizing the GRU cell facilitates building models with ten or more layers. GGSNNs are less prone to over-smoothing, which is an advantage over GCNs.

3.2.4 Graph Network Formalism

Additionally to Refs. [38, 47], the following explanations also follows Ref. [52] if not noted otherwise.

The majority of the GNNs only concentrate on vertex attributes in message passing. However, there is no reason for not aggregating and updating edge-level or graph-level information as well. The Graph Network (GN) formalism, introduced in Ref. [52], provides this but also captures GNNs that do not belong to the MPNN framework. Thus, the GN

formalism is an even more general framework for defining GNNs and at the same time even extending various GNN approaches. The core computation unit in this formalism is called GN block. It consists of three sub-blocks, an edge block (e), a vertex block (v) and a global block (g). Every sub-block contains an update function ϕ (often an ANN), which outputs the new embedding for the edges, the vertices and the entire graph. The vertex and the graph block additionally contain aggregation functions ρ, which are often element-wise sums or means since they are permutation invariant. In the case of ϕ, its index simply reflects the sub-block in which it is used whereas the index of ρ and $\mathbf{m}$ indicates with respect to which object (edge, vertex, graph) the computation is performed and to which sub-block this result will subsequently flow. In the Graph Network formalism the message passing step, defined in Equation 3.12, extends to

$$\tilde{\mathbf{a}}_{uv}^{(l)} = \phi_{\mathrm{e}}^{(l)}\left(\tilde{\mathbf{a}}_{uv}^{(l-1)}, \mathbf{x}_u^{(l-1)}, \mathbf{x}_v^{(l-1)}, \mathbf{g}^{(l-1)}\right) \tag{3.16}$$

$$\mathbf{m}_{\mathrm{edge}\to\mathrm{v}}^{(l)} = \rho_{\mathrm{edge}\to\mathrm{v}}^{(l)}\left(\left\{\tilde{\mathbf{a}}_{uv}^{(l)}, \forall v \in \mathcal{N}(u)\right\}\right) \tag{3.17}$$

$$\mathbf{x}_u^{(l)} = \phi_{\mathrm{v}}^{(l)}\left(\mathbf{m}_{\mathrm{edge}\to\mathrm{v}}^{(l)}, \mathbf{x}_u^{(l-1)}, \mathbf{g}^{(l-1)}\right) \tag{3.18}$$

$$\mathbf{m}_{\mathrm{edge}\to\mathrm{g}}^{(l)} = \rho_{\mathrm{edge}\to\mathrm{g}}^{(l)}\left(\left\{\tilde{\mathbf{a}}_{uv}^{(l)}, \forall u, v \in E\right\}\right) \tag{3.19}$$

$$\mathbf{g}^{(l)} = \phi_{\mathrm{g}}^{(l)}\left(\mathbf{m}_{\mathrm{edge}\to\mathrm{g}}^{(l)}, \mathbf{m}_{\mathrm{vertex}\to\mathrm{g}}^{(l)}, \mathbf{g}^{(l-1)}\right) \tag{3.20}$$

$$\mathbf{m}_{\mathrm{vertex}\to\mathrm{g}}^{(l)} = \rho_{\mathrm{vertex}\to\mathrm{g}}^{(l)}\left(\left\{\mathbf{x}_u^{(l)}, \forall u \in V\right\}\right), \tag{3.21}$$

where $\tilde{\mathbf{a}}_{uv}^{(l)}$ is the hidden embedding for the edge $(u, v) \in E$, $v \in \mathcal{N}(u)$ and $\mathbf{g}^{(l)}$ is the hidden embedding for the entire graph, both calculated at iteration l. Figure 3.8a shows a flow chart of a possible order in which Equations 3.16-3.21 are calculated at each iteration l:

(1) ϕ_{e}: Update the hidden edge embeddings considering the current hidden edge embeddings and the current hidden vertex embeddings of the incident vertices as well as the current hidden global embeddings.

(2) $\rho_{\mathrm{edge}\to\mathrm{v}}$: Calculate for each vertex in the graph the aggregation of the updated hidden edge embeddings of the respective incident edges.

(3) ϕ_{v}: Update the hidden vertex embeddings considering the results from (2), the current hidden vertex embeddings and the current hidden global embeddings.

(4) $\rho_{\mathrm{edge}\to\mathrm{g}}$, $\rho_{\mathrm{vertex}\to\mathrm{g}}$: Aggregate over all updated hidden vertex embeddings as well as over all updated hidden edge embeddings.

(5) ϕ_{g}: Update the hidden global embeddings based on the current hidden global embeddings and the results from (4).

As already mentioned, this is only one out of many possible orders for calculating the GN block's output. In fact, some of the GNN architectures captured within this GN framework will only be obtained if rearranging or removing of functions in the edge, vertex or global block is allowed. A representation of the aforementioned GCN in the GN framework is depicted in Fig. 3.8b. An expression of Equation 3.14 in the GN formalism can be found in Ref. [38].

However, despite having processed all iterations of message passings, still no final predicted class $\mathbf{y}$ for a given task can be made, since GNNs only output graphs. Therefore, a readout function R, such as a neural network, is required, which has the purpose to calculate the final feature vector for the particular graphs. [53] The complete GNN flow is depicted in Fig. 3.9.

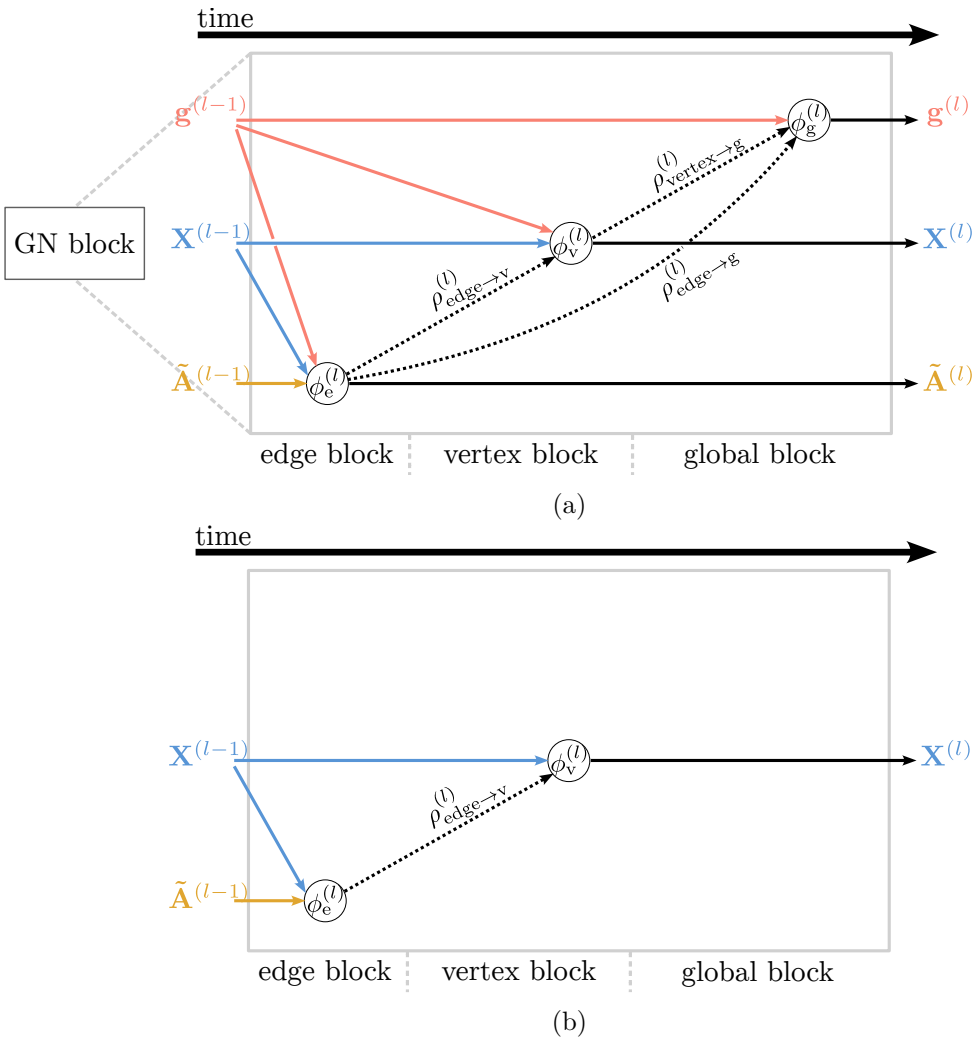


Figure 3.8: **GN blocks.** The input to a complete GN block (a) are the global attributes $\mathbf{g}^{(l-1)}$, the vertex attributes $\mathbf{X}^{(l-1)}$ and the edge attributes $\tilde{\mathbf{A}}^{(l-1)}$ associated with the graph in question. All these attributes are updated in a sequential manner, yet their update order is not fixed. In this flow chart, the edge block is performed first, the vertex block second and lastly the global block. The updated attributes are denoted $\mathbf{g}^{(l)}$, $\mathbf{X}^{(l)}$ and $\tilde{\mathbf{A}}^{(l)}$. On the contrary, a GN block used in a GCN (b) only requires the computations made in the edge block and the vertex block. Moreover, only the vertex attributes are updated. Figures adapted from Ref. [38]

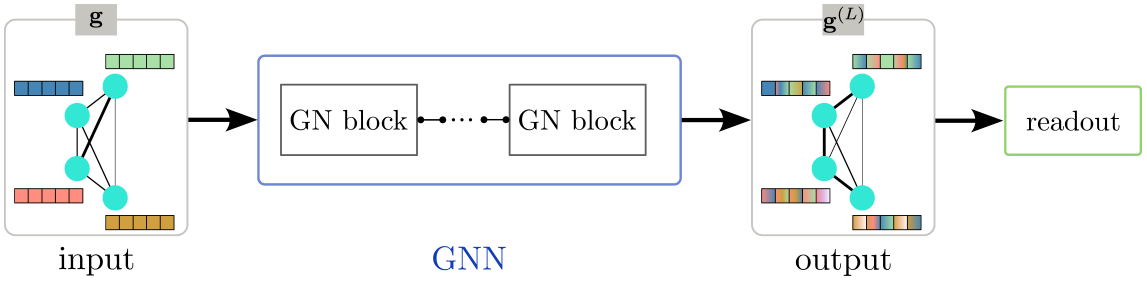


Figure 3.9: **GNN flow**. The forward pass in a GNN consists of two parts, the GNN, which consists of L GN blocks, and a readout [53] since the GNN per se is a graph-to-graph function. Figure adapted from Ref. [58].

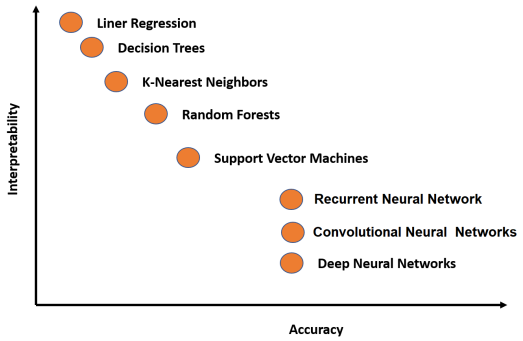


Figure 3.10: **Schematic comparison of model accuracy and model interpretability.** There is a tendency that less interpretable, i.e., more complex models, show a greater prediction accuracy than simpler models such as linear regression or decision trees. Being just another variant of neural networks, GNNs are expected to have a similar relationship between accuracy and interpretability as the neural networks shown in the diagram. Figure taken from Ref. [61].

3.3 Explainable AI

As stated in Section 3.1, empirically, neural networks with more hidden layers are more performant than single-layer, less deep neural networks. In fact, deep learning methods outperform traditional machine learning methods in many research tasks but are at the same time naturally more complex than decision trees (cf. Fig. 3.10), for example. Thus, while the reasons behind decisions of decision trees can be understood by simply looking at the decision tree itself (cf. Fig. 3.11) and are therefore considered as interpretable models, this neither holds for DNNs nor GNNs. They are hence considered not fully trustworthy, black-box models. Of course, it is inevitable to understand the decisions of a model in order to draw correct conclusions from data and, for instance, not mistaking noise with new physics, on the one hand. On the other hand, using uninterpretable methods in critical domains such as medical domains could even be life-threatening. Consequently, the demand for explainable AI (xAI), which comprises post hoc techniques with the purpose to explain the predictions of uninterpretable methods, arouse and is ever-increasing. It should be noted that in this thesis, unlike in many other papers, e.g., Ref. [37] or Ref. [59], the definitions of interpretability and explainability are not treated as interchangeable terms and follow the definitions of Ref. [60]. A standardized notion does unfortunately not yet exist. [59, 60]

There are already various approaches developed for explaining deep learning methods. In the following, the GNNExplainer (Section 3.3.1), a method for exclusively explaining GNNs, and the Taylor coefficient analysis (Section 3.3.2) will be introduced as two out of many approaches. Both explain the predictions of uninterpretable methods by extracting the importance of the input features for the model prediction but are otherwise fundamentally different methods and will be compared in Section 3.3.3.

Since global attributes are not applied to graphs in this thesis, global attributes are not considered anymore in the following explanations and visualizations.

3.3.1 GNNExplainer

The GNNExplainer (GNNX), introduced in Ref. [63], is a post hoc, model-agnostic method, which is capable of providing explanations for predictions of GNNs performing NLP or

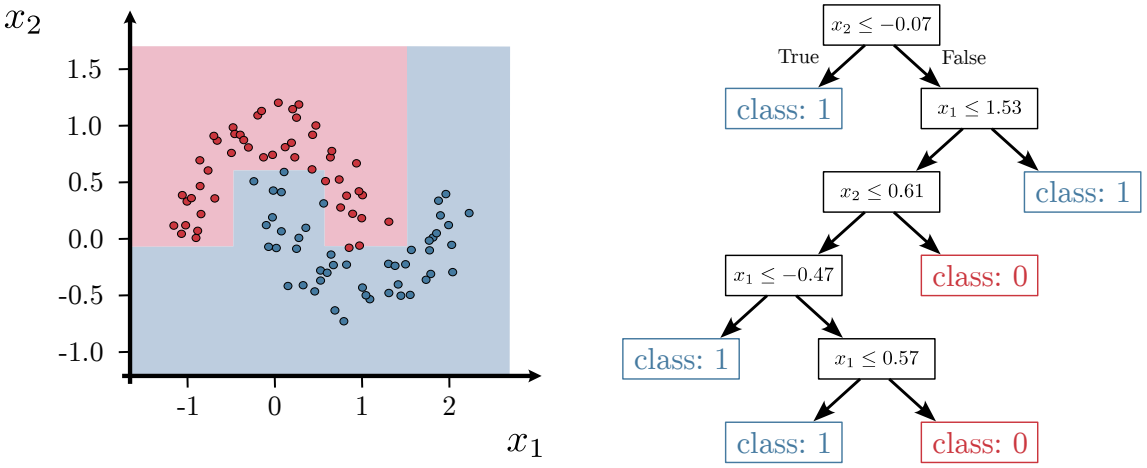


Figure 3.11: **Decision tree.** On the left-hand side, data points that shall be classified are drawn in feature space including the decision boundary that is determined via training of a decision tree. On the right-hand side, the corresponding decision tree is depicted. New data points can be classified by evaluating the decision tree from top (root) to bottom until a leaf (red or blue box) is reached. Thus, the reason behind each prediction can be easily comprehended by looking at the decision rules in the decision tree. Figure adapted from Ref. [62].

GLP. However, there are differences in terms of the mathematical foundation of the GNNExplainer being applied to explaining predictions on node-level or graph-level. In the following, only the mathematical background of the latter is further explained, based on Ref. [63], since this thesis mainly explores graph-level-prediction.

A prediction $\hat{\mathbf{y}}$ of a GNN model Φ only depends on the information derived from the computation graph G_C and the vertex attributes $\mathbf{X}_\mathrm{C}$ of the vertices captured in G_C. The computation graph comprises all the vertices and edges involved in computing the hidden embedding of a vertex u in the graph in question. In the case of an incomplete graph, the computation graph varies between different vertices in the same graph and does not correspond to the input graph. But, since in GLP the whole graph is taken into account for a prediction, G_C always equals the input graph G and its adjacency matrix $\mathbf{A}_\mathrm{C}$ equals $\mathbf{A}$. The idea behind the GNNExplainer is that only a small fraction of edges and vertex attributes of an input graph have a high impact on a prediction $\hat{\mathbf{y}}$. Thus, the goal of the GNNExplainer is to extract the small subgraph $G_\mathrm{S} \subseteq G_\mathrm{C}$ and the small subset of vertex attributes $\mathbf{X}_\mathrm{S}$ playing a crucial role for the GNN's prediction and therefore being considered the explanation for $\hat{\mathbf{y}}$.
Using the mutual information (MI) between the predicted label distribution Y and the explanation $(G_\mathrm{S}, \mathbf{X}_\mathrm{S})$, the task can be formulated as the optimization problem

$$\max_{G_\mathrm{S}, \mathbf{X}_\mathrm{S}} \left(\mathrm{MI}\left(Y, (G_\mathrm{S}, \mathbf{X}_\mathrm{S})\right)\right) . \tag{3.22}$$

The mutual information is defined as the difference between the entropy of the computation graph $H(Y)$ and the entropy of the subgraph $H(Y|G = G_\mathrm{S}, X = \mathbf{X}_\mathrm{S})$

$$\mathrm{MI}\left(Y, (G_\mathrm{S}, \mathbf{X}_\mathrm{S})\right) = H(Y) - H(Y|G = G_\mathrm{S}, X = \mathbf{X}_\mathrm{S}) . \tag{3.23}$$

Since the GNNExplainer is applied to a model post-training, i.e., Φ is constant, the first term in Equation 3.23 is a constant. Consequently, maximizing MI can be simplified to

minimizing

$$H(Y|G = G_{\mathrm{S}}, X = \mathbf{X}_{\mathrm{S}}) = -\mathbb{E}_{(Y|G_{\mathrm{S}}, \mathbf{X}_{\mathrm{S}})}\big[\log P_{\Phi}(Y|G = G_{\mathrm{S}}, X = \mathbf{X}_{\mathrm{S}})\big]\,, \tag{3.24}$$

where $P_{\Phi}(\cdot)$ denotes the probability of the particular event in question being associated with a specific class. Accordingly, $P_{\Phi}(\cdot)$ equates the SIGMOID or SOFTMAX transformed logits $\hat{\mathbf{y}}$ of GNNs for the binary or multiclass classification (cf. Equation 3.5), respectively. However, Equation 3.24 is not tractable due to the existence of exponentially many subgraphs G_{S}. The GNNExplainer overcomes this issue by introducing an edge mask $\mathbf{M}$ and a feature mask $\mathbf{F}$ that act as filters on the adjacency matrix and vertex attributes, respectively. In other words, it is possible to control the influence of all entries in the adjacency matrix and the vertex attributes on the final prediction of the model by varying the values in the masks. These masks are randomly initialized at the beginning and are adjusted through training. The training of the masks should however not be confused with the training of the model per se, which is already completed before the GNNExplainer is applied. Training the masks figuratively means that in each training step different edges and vertex attributes are suppressed, i.e., several subgraphs and subset of vertex attributes can be tested in that manner until the optimal subgraph and subset of vertex attributes are found for explaining the prediction. The loss function $\mathcal{L}$ for this training, which is optimized via gradient descent, is described by Equation 3.24. If the explanation for the predicted label y and not the exact prediction $\hat{\mathbf{y}}$ (output of the GNN) is the main focus of interest, it can be simplified to the cross-entropy between the originally predicted class y from all possible classes K for an event and the prediction value $P_{\Phi}(\cdot)$ of the subgraph for the same event

$$\mathcal{L} \coloneqq -\sum_{k=1}^{K} \mathbb{1}_{y=k} \log P_{\Phi}\left(Y = y|G = \mathbf{A}_{\mathrm{C}} \odot \sigma(\mathbf{M}), X = \mathbf{X}_{\mathrm{S}} \odot \sigma(\mathbf{F})\right)\,, \tag{3.25}$$

with $\sigma(\cdot)$ denoting the SIGMOID function, which is applied to keep the range of the mask entries between zero and one, and $\odot$ denoting an elementwise multiplication. For a detailed derivation of Equation 3.25, see Ref. [63].
Moreover, regularization terms such as the sum over the elements in the masks or the elementwise entropy of the masks are added to Equation 3.25 in order to ensure a small size of the explanation on the one hand and to favor more discrete masks on the other hand.

The subgraph and subset of vertex attributes that are responsible for the prediction of the GNN are consequently

$$G_{\mathrm{S}} = \mathbf{A}_{\mathrm{C}} \odot \sigma(\mathbf{M}) \tag{3.26}$$

$$\mathbf{X}_{\mathrm{S}} = \mathbf{X}_{\mathrm{S}} \odot \sigma(\mathbf{F})\,. \tag{3.27}$$

The sigmoid-transformed edge and sigmoid-transformed feature mask are the ones actually capturing the information about the magnitude of the influence (and therefore the importance) of the particular edges and vertex attributes for the GNN response. The larger the values in the entries, the higher the impact of the corresponding vertex and feature. Therefore, the masks are regarded as the actual explanation provided by the GNNExplainer for the predictions in this thesis and not $(G_{\mathrm{S}}, \mathbf{X}_{\mathrm{S}})$.

The GNNExplainer is able to perform different independent tasks for explaining both NLP and GLP depending on the dimensions of the matrixes with which the feature mask $\mathbf{F}$ is originally initialized. If $\mathbf{F} \in \mathbb{R}^{1\times m}$, $\mathbf{F} \in \mathbb{R}^{n\times m}$ or $\mathbf{F} \in \mathbb{R}^{n\times 1}$, the GNNExplainer delivers the importance of vertex attributes, the importance of each vertex attribute of each vertex and the importance of each vertex per se for the prediction, respectively (cf. Fig. 3.12). Conversely, the dimension of the edge mask $\mathbf{M}$ remains $\mathbb{R}^{n\times n}$ at all times.

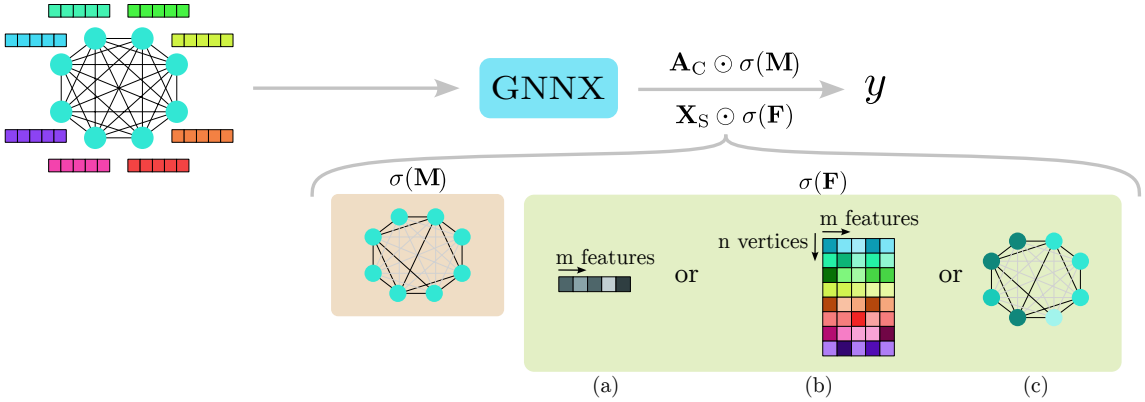


Figure 3.12: **GNNExplainer flow.** The GNNExplainer explains the predicted class y for an input graph by determining an edge mask $\mathbf{M}$ and a feature mask $\mathbf{F}$ through training. These masks determine which part of the subgraph ($\mathbf{A}_C \odot \sigma(\mathbf{M})$) and which node features ($\mathbf{X}_S \odot \sigma(\mathbf{F})$) need to be considered for getting y. Thereby, the edge mask captures the influence of the edges on the prediction, which is illustrated through the different thicknesses of the edges of the graphs and depending on the dimensionality of $\mathbf{F}$, either the importance of the vertex attributes (a), the importance of each vertex attribute of each vertex (b) or the importance of each vertex (c) of the input graph for the prediction are delivered by the GNNExplainer. This is visualized by the different color intensities of the sub-rectangles and the vertices.

3.3.2 Taylor Coefficient Analysis

The Taylor expansion of a function is a powerful tool for approximating and thus simplifying complicated problems, as shown in Ref. [64], for instance. As already stated in Section 3.1, a neural network model Φ is essentially an analytic function, consisting of a great number of parameters, mapping the input vector to an output vector. Consequently, as proposed in Ref. [65] and further examined in Ref. [66], the Taylor expansion is applicable to approximating Φ around the expansion points, given that Φ is differentiable at the expansion points, which is guaranteed here since the applied activation functions are all differentiable [39]. In this context of application, the data points from the test set serve as expansion points $\mathbf{z} \in \mathbb{R}^m$.
The multidimensional Taylor expansion T, see, for example, Ref. [67], for Φ in its input features $\mathbf{x} \in \mathbb{R}^m$ at the expansion points $\mathbf{z}$ can be generally expressed by

$$\begin{aligned} T_\Phi(x_1,\dots,x_m) &= \sum_{n_1=0}^{\infty} \cdots \sum_{n_m=0}^{\infty} \left(\frac{\partial^{\sum_{i=1}^m n_i} \Phi(z_1,\dots,z_m)}{\partial x_1^{n_1} \cdots \partial x_m^{n_m}} \right) \frac{\Pi_{i=1}^m (x_i - z_i)^{n_i}}{\Pi_{i=1}^m n_i!} \\ &= \Phi(z_1,\dots,z_m) + \sum_{j=1}^{m} \frac{\partial \Phi(z_1,\dots,z_m)}{\partial x_j}(x_j - z_j) \\ &\quad + \frac{1}{2!} \sum_{j=1}^{m} \sum_{k=1}^{m} \frac{\partial^2 \Phi(z_1,\dots,z_m)}{\partial x_j \partial x_k}(x_j - z_j)(x_k - z_k) + \text{higher-orders} \\ &=: t_0 + \sum_{j=1}^{m} t_{x_j}(x_j - z_j) + \frac{1}{2!} \sum_{j=1}^{m} \sum_{k=1}^{m} t_{x_j x_k}(x_j - z_j)(x_k - z_k) + \text{h.-o.}\,, \end{aligned} \tag{3.28}$$

where $t_\alpha, \alpha \in \{0, x_j, x_j x_k, \dots\}$ denotes the Taylor coefficients. Except for the zeroth-order Taylor coefficient t_0, all Taylor coefficients correspond to the derivative in a summand and the subscript of the Taylor coefficients indicates which input feature(s) a particular

Table 3.1: **Comparison of the explanation abilities of GNNX and TCA.** If the particular xAI method is able to explain the given aspect, it is marked with ✔ and otherwise with ✗ in the table. Since the GNNExplainer is theoretically able to explain the importance of edges but do not take edge attributes into account in calculating the explanation, this ability is put in parentheses.

explained aspects	GNNExplainer	Taylor coefficient analysis
importance of edges/relational information	(✔)	✗
importance of vertices	✔	✗
importance of vertex attributes	✔	✔
importance of relations of vertex attributes	✗	✔
importance of vertex attributes per vertex	✔	✗

Taylor coefficient contains information about. Since t_0 does not contain any input feature related information, the authors in Ref. [65] utilize Taylor coefficients as of the first-order for extracting the impact of the input features on the prediction of a neural network. This method, called Taylor coefficient analysis, exploits that the influence of the bare input features on the prediction, which is calculated via Φ, is captured in the first-order features $\{t_{x_j}, \forall j \in \{1, 2, \ldots, m\}\}$ and the influence of relations among the input features is contained in the higher-order features.

3.3.3 Comparison

The GNNExplainer and the Taylor coefficient analysis are two different approaches for shedding light on black-box models in (deep) machine learning. While the GNNExplainer itself, or rather the particular mask, needs to be trained, the Taylor coefficient is based on a well-understood mathematical foundation. Therefore, the explanations provided by the GNNExplainer are not necessarily deterministic or stable and are moreover dependent on the chosen hyperparameters. However, since the GNNX is specially designed for explaining GNNs, it is capable of explaining the importance of relational information, i.e., edges, as well as the importance of vertices or the importance of features per vertex, also named vertex specific feature importance in the following, for the GNN response, unlike the TCA, as summarized in Table 3.1. Yet, the GNNX still does not take edge attributes into consideration when determining the edge mask. In Chapter 6, these two xAI methods are applied to the GNNs that are studied in this thesis by which it is possible to probe the actual plausibility of the delivered explanations.
The advantage of the TCA is that it is applicable to both GNNs and DNNs and can give further insight into the importance of relations across input features for the model predictions when higher-orders of the Taylor expansion are taken into account. This versatility of TCA is exploited in Section 7.4 for a more detailed comparison of DNNs and GNNs.

4 Multivariate Event Classification with GNNs

As described in Section 2.5, the physics goal of this thesis is to classify $t\bar{t} + b\bar{b}$, $t\bar{t}H(b\bar{b})$ and $t\bar{t}Z(b\bar{b})$ events in the single-lepton channel of $t\bar{t}$ decays in both a binary and a multiclass manner. For the binary case, $t\bar{t} + b\bar{b}$ is treated as signal process and $t\bar{t}H(b\bar{b})$ and $t\bar{t}Z(b\bar{b})$ as background processes. In order to accomplish these tasks, graph neural networks performing graph-level-predictions are the method of choice for this thesis. In a former study [68], GNNs were only examined for identifying additional b jets in $t\bar{t} + b\bar{b}$ events, which is a node-level-prediction task. This thesis builds on this work and the goal of this first chapter is thus to probe the general feasibility of GNNs for classification of $t\bar{t}$+X events, i.e., conduct a proof of concept study[1].
In Section 4.1, the data used are presented and remarks on the reproducibility of the results are given. Afterwards, the model architecture and hyperparameters used for this study are introduced in Section 4.2. The resulting binary and multiclass classification models are subsequently compared in Section 4.3 and in Section 4.4 further optimization approaches are examined on the models. As generator-level information, i.e., ground-truth, is used in a part of the trainings, a method for circumventing this is introduced in Section 4.5. The latter section also contains a performance comparison of these models with models from the previous sections trained with generator-level information.

4.1 Reproducibility and Data

All models that are evaluated in this thesis are implemented in PyTorch Geometric (PyG) [70] v2.0.3, a PyTorch-based library [71] designated to developing graph neural networks, and are trained on an NVIDIA GeForce GTX 1080 Ti graphic card. Albeit the same machine is deployed and seeds are set wherever possible, the trained models are not fully reproducible. This traces back to internal PyTorch functions behaving non-deterministically when applied to CUDA tensors and for which no deterministic alternative is provided yet. A list of functions to which this applies can be found in Ref. [72]. It is also not feasible to circumvent this by completely refraining from using GPUs and only utilize CPUs for training the neural networks due to the time frame of this thesis. Yet, since each training is repeated ten times, still robust results are receivable and well-grounded

[1] In a parallel study [69], GNNs were applied to the separation of $t\bar{t}$+X events as well, yet in the dileptonic channel and with a different GNN architecture.

conclusions can be drawn accordingly.

For accomplishing the multivariate $t\bar{t}$+X event classification task with GNNs, simulated data provided by the CMS collaboration is applied to training the models. Originally, this simulated data is obtained through Monte Carlo event generators, which are POWHEG + PYTHIA8 [73, 74] being set up with the CMS PYTHIA8 tune 5 (CP5) [75]. The generated data are here all simulated in a way that reflects real data taken under the experimental conditions of the CMS experiment in 2017. What is special about the $t\bar{t} + b\bar{b}$ events is that, due to the small number of events, both simulations based upon a 4-flavor scheme (4FS) and simulations that are based on a 5-flavor scheme (5FS) description of the b quarks in QCD are used without further differentiation [76]. Using both simulation approaches jointly for training machine learning models is possible as it does not have an impact on the model performance, according to a study conducted in Ref. [77].
Since it is known beforehand that the events of interests consist of at least six jets with at least four b-tagged jets ($\geq$ 6 jets, $\geq$ 4 b-tagged jets phase space), this selection criterion is applied to the simulated data. Furthermore, each of the additional b jets must contain one and only one b hadron on generator-level.
As the jets are not yet labeled, i.e., not yet assigned to their particular category, this has to be performed manually by comparing the flavor of the jets (light flavor, c, b) to the particle from which the jets originate (unassigned, additional particle, t, $\bar{t}$, H, Z, W, hadronically decaying W). Nevertheless, it can occur that not all six jet categories, i.e., HadTopB, 2x HadTopQ, 2x AddB and LepTopB (cf. Section 2.5), are incorporated in an event. When this happens to the category AddB then the particular event is discarded, otherwise the missing categories HadTopB, LepTopB or 2x HadTopQ are assigned to the remaining jets in the event in descending p_T order. This auxiliary assignment approach is justified by physics since it is expected that a big fraction of the energy that is contained in the large mass of top quarks is transferred to momentum when decaying into a b quark (and a W boson) due to the small mass of b quarks.
With all these restrictions, there are 189131 events remaining in total, of which 53477 belong to $t\bar{t} + b\bar{b}$, 102464 to $t\bar{t}H(b\bar{b})$ and 33190 to $t\bar{t}Z(b\bar{b})$ events. These events are then split into training, validation and test set in the ratio 60/20/20.

Graph neural networks are deployed for this classification task since the physics processes to be classified can be naturally described as graphs. Each final state object simply forms a vertex in the graph and then each vertex is connected to every other vertex, leading to complete graphs, as illustrated in Fig. 4.1. Thereby, only kinematic observables and the b tag value of jets serve as vertex attributes

$$\mathbf{x}_i = \begin{pmatrix} p_\mathrm{T} & \phi & \eta & M & E & \text{b tag} \end{pmatrix}^\intercal \tag{4.1}$$

for each vertex i in a graph and ΔR is added as edge weight to the graph. For leptons and neutrinos, not all of this information exists. In that case, the particular feature is set to zero. These features are selected since promising results are therewith achieved in Ref. [68] in node-level-prediction. The distribution of the selected vertex attributes can be found in Fig. A.1.

In order to facilitate the training process, each input feature x_i is transformed to a similar scale (feature scaling). For vertex attributes, this is archived via standardization

$$x_i' = \frac{x_i - \mu_i}{\sigma_i} \tag{4.2}$$

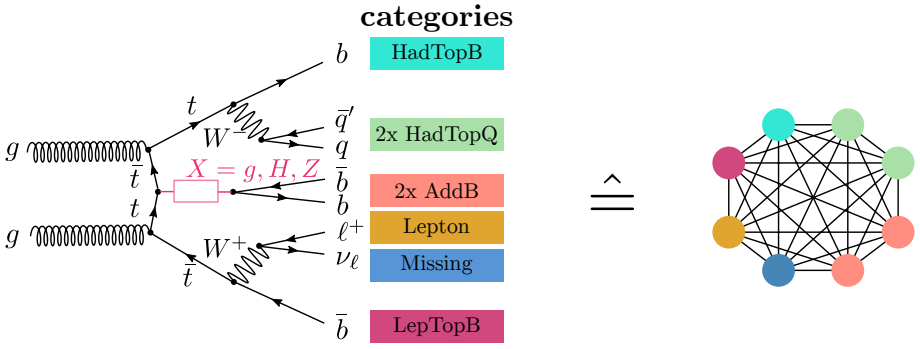


Figure 4.1: **Schematic depiction of the translation of the physics processes under scrutiny to a graph representation.** At leading order, the graph representation only consists of eight vertices corresponding to the eight final states showcased in the Feynman diagram.

whereas normalization

$$x_i' = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)} \tag{4.3}$$

is applied to edge weights. The variables μ_i and σ_i correspond to the mean and standard deviation (square root of the unbiased sample variance) of the distribution of the particular feature x_i. Using normalization instead of standardization for edge weights ensures that these values share a similar scale with the vertex attributes but are still non-negative. The disadvantage of this technique is that its scaling depends on the minimal and maximal values of the distribution and is hence not suitable for distributions with outliers. For this reason, normalization is used for vertex attributes.

4.2 Architecture and Hyperparameters

Since Gated Graph Sequence Neural Networks (GGSNN) have shown promising results in identifying additional b jets in Ref. [68] and these jets are expected to play a crucial role in binary and multiclass classification of $\mathrm{t\bar{t}} + \mathrm{b\bar{b}}$, $\mathrm{t\bar{t}H(b\bar{b})}$ and $\mathrm{t\bar{t}Z(b\bar{b})}$, this type of graph neural networks is deployed for the graph-level-prediction task at hand as well. Equally, the majority of the selected hyperparameters, summarized in Table. 4.1, are inspired by the findings in Ref. [68]. A hyperparameter for which this does not completely hold is the metric that is monitored in EARLY-STOPPING and that is also relevant for adaption of the learning rate. Originally, both EARLY-STOPPING and the adaption of the learning rate depend on the true positive rate (TPR) of a model, which is however not reasonable as it may prohibit the convergence of a training. The true positive rate is the fraction of events being correctly classified as signal (true positive) by a model with respect to the total number of signal events, i.e., positives. Consequently, also a poorly performing model that classifies each event as signal possesses a perfect TPR and deceives the training process. Hence, the validation loss is probed as monitored metric alongside the TPR.

It might be surprising that the input dimension can be chosen to be 24 although only five attributes are associated to each vertex. This is possible as the excess nodes are automatically filled with zeros, i.e., are zero padded. Incidentally, as the dimension of the embeddings of the vertices are not or rather cannot be changed throughout the 18 GGSNN layers, the number of hidden nodes in each hidden layer (HL) n_{hidden} equals the number of input nodes n_{in}. The corresponding schematic representation of the model architecture is depicted in Fig. 4.2. A global pooling method, summarizing the adjusted

Table 4.1: **Selected hyperparameters.** Parameters that are not further specified in the table correspond to the particular default values in PyG. The classification threshold used to calculate the TPR is 0.5 in the binary case, i.e., all events for which the model predicts a score higher than 0.5 are classified as signal, otherwise as background. For calculating the TPR in multiclass classification, no classification threshold needs to be set since the output node with the highest prediction score determines the predicted class for an event.

hyperparameter	setting
$n_{\text{in}}/n_{\text{hidden}}$	24
n_{HL}	18
n_{out} (of readout)	1 (binary), 3 (multiclass)
bias	true
aggregation function ρ (Eq. 3.21)	mean
global pooling method	mean
maximum number of epochs	200
EARLY-STOPPING	Δepoch = 15, ΔTPR = 0.01 or Δepoch = 15, Δloss = 0.001
mini-batch size	200
optimizer	Adam ($\gamma = 0.01$)
activation function (in output layer)	sigmoid (binary), softmax (multiclass) (Eq. 3.5)
loss function	binary/categorical cross-entropy (Eq. 3.6)
number of repetitions	10

hidden embeddings of the vertices, is inserted in between the last GGSNN layer and the readout since in GLP, a classification based on the graph as a whole is desired. With these configurations, the models deployed for binary and multiclass classification consist of 13993 and 14043 trainable parameters N_{TP}, respectively. This also corresponds to the number of degrees of freedom (DOF) of the models.

4.3 Binary and Multiclass Classification

The ROC-AUC value is a common measure for determining the performance of machine learning models. Accordingly, this is chosen as performance measure of the trained models in this thesis. The ROC-AUC value corresponds to the area under the receiver operating characteristic (ROC) curve, which is the graphical representation of the relationship between TPR and the false positive rate (FPR) of the model under scrutiny for every possible classification threshold. The false positive rate is defined in an analogous manner to TPR and the classification threshold determines the prediction score from which an event is labeled as signal. The range of the ROC-AUC value is [0.5, 1.0], where 1.0 corresponds to a perfect classifier and 0.5 to a model without any discriminative power, often referred to as random estimator. For further details on ROC analysis refer to Ref. [78], for instance. A multiclass classification model distinguishes more than two classes and thus the ROC analysis does not seem applicable at first sight. However, there are several approaches to handle this. Here, it is decided to calculate the ROC curve separately for each class whereat the particular class is treated as signal that is classified against all remaining classes in the dataset ("rest") at the same time. This means the multiclass classification model is simplified to several binary classifiers in order to be able to calculate the ROC curve and the corresponding ROC-AUC value. The ROC curve describing the overall multiclass classification model is finally obtained by averaging over the ROC curves calculated

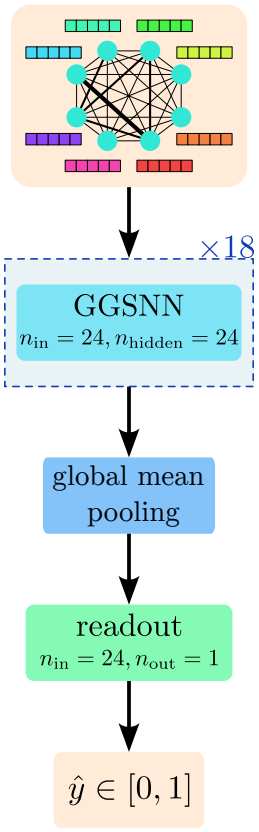


Figure 4.2: **Schematic representation of the GGSNN model for the binary case, including readout.** The vertex attributes of the input graph are updated through the 18-layer GGSNN, yet without changing the dimensions of these attributes. Since event classification shall be performed, i.e., a graph-level-prediction task, a global mean pooling layer is required for transforming the updated graph into a vector representation, which serves as input to the subsequent readout. The readout is a conventional neural network without hidden layers and has the purpose to transform the vector representation into a single value, the prediction $\hat{y}$ for an event, in the binary case. In case of multiclass classification, three output nodes are utilized, i.e., the prediction is not a scalar but a vector $\hat{\mathbf{y}}$, whereby the output node with the highest output value determine the class of the particular event.

separately for each class. This curve is referred to as macro-average ROC curve.

The ROC curves and macro-averaged ROC curves of the models under scrutiny averaged over all ten repetitions of the particular trainings are depicted in Fig. 4.3. Models trained on the same monitored metric in the training process are placed in the same column (left: TPR, right: loss) and models performing the same task are in the same row (top: binary classification, bottom: multiclass classification). For binary classification models, the ROC curve of each repetition is additionally displayed in the plots. It becomes evident that the performance spread of the models trained under TPR monitoring is larger than when trained under loss monitoring, which indicates a comparatively more stable training of the latter and is expected in accordance to the explanation given in Section 4.2. Moreover, one repetition of the training sticks out in Fig. 4.3a as it possesses a perceivably lower performance than the other repetitions and even lies outside the standard deviation of the mean ROC curve. When looking into that model in detail, it becomes apparent that this is indeed caused by a deceptive high true positive rate of 91.92 % while at the same time the true negative rate (TNR), i.e., the fraction of events being correctly classified as background (true negative) by a model with respect to the total number of background events, is only 9.96 %. Hence, it is justified to treat this repetition as an outlier and discard it from further analysis. In order to ease identification of further outliers, the following objective criteria are introduced

(a) ROC-AUC = 0.5 or

(b) ROC-AUC $\notin \langle \text{ROC-AUC} \rangle \pm 1.5 \cdot \sigma^{\text{pre}}_{\text{ROC-AUC}}$ and $\Delta\sigma > 0.0025$,

where $\Delta\sigma = \sigma^{\text{pre}}_{\text{ROC-AUC}} - \sigma^{\text{post}}_{\text{ROC-AUC}}$ holds. All models with a performance beyond ± 1.5 times the standard deviation (square root of the unbiased sample variance) of the mean ROC-AUC $\langle \text{ROC-AUC} \rangle$ are considered as potential outliers. If the removal of all potential outliers results in a spread $\sigma^{\text{post}}_{\text{ROC-AUC}}$ that is at least 0.0025 smaller than the previous spread $\sigma^{\text{pre}}_{\text{ROC-AUC}}$, the potential outliers are regarded as real outliers. These thresholds have been empirically proven to be reasonable (cf. Appendix B). Only models that pass this outlier check, i.e., do not fulfill the outlier criteria, are further considered. This applies to every trained model considered in this thesis.

After application of the outlier criteria, also two outliers in the multiclass classification models trained under monitoring of the TPR are identified and discarded. One of the outliers performed too poorly and one too well, leading to a decreased mean ROC-AUC value post-removal. The mean ROC curves post-application of the outlier criteria are shown in Fig. 4.4. Despite of the removal of outliers, the mean performance of the models trained under monitoring of TPR are worse than when trained under monitoring of the loss regardless of the task. While for binary classification the difference in model performance is (-3.8 ± 1.3) %, it is only (-0.5 ± 0.6) % for multiclass classification. It should be noted too that the performance of the binary model trained under monitoring of TPR is worse than the equivalent multiclass classification model. For the loss as monitored metric, this is vice versa. That is, the training for multiclass classification is similarly stable regardless of the actual monitored metric. This makes sense as a perfect TPR of 1 cannot simply be obtained by assigning every event to the same class in the multiclass classification case. However, the slight superior performance of multiclass classifiers trained under monitoring of the loss as opposed to monitoring of the TPR indicates that it is generally beneficial to monitor the loss. Consequently, only this concept will be further pursued for both tasks. Nevertheless, it should not be neglected that the performance of both binary and multiclass classifiers are not even 45 % better than a random estimator and thus, still leaving a lot of room for improvements.

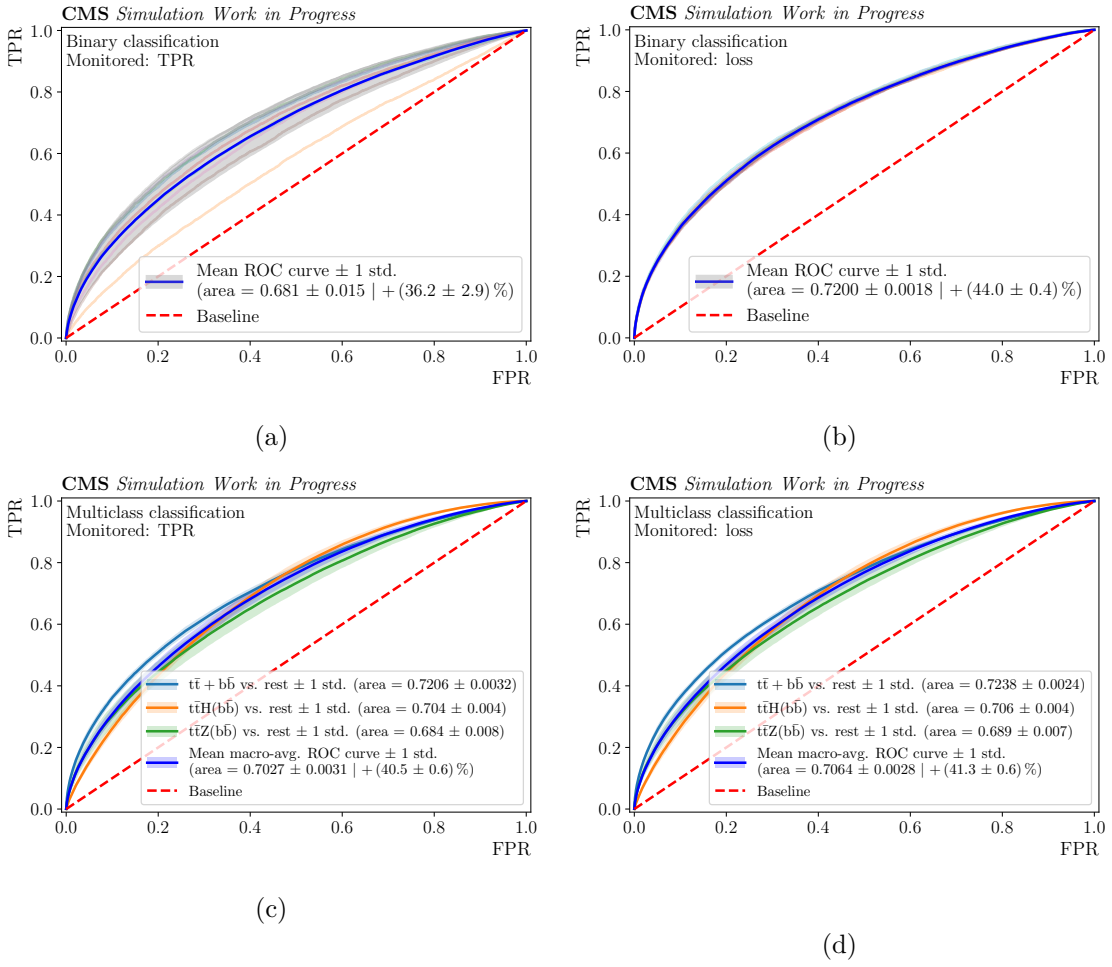
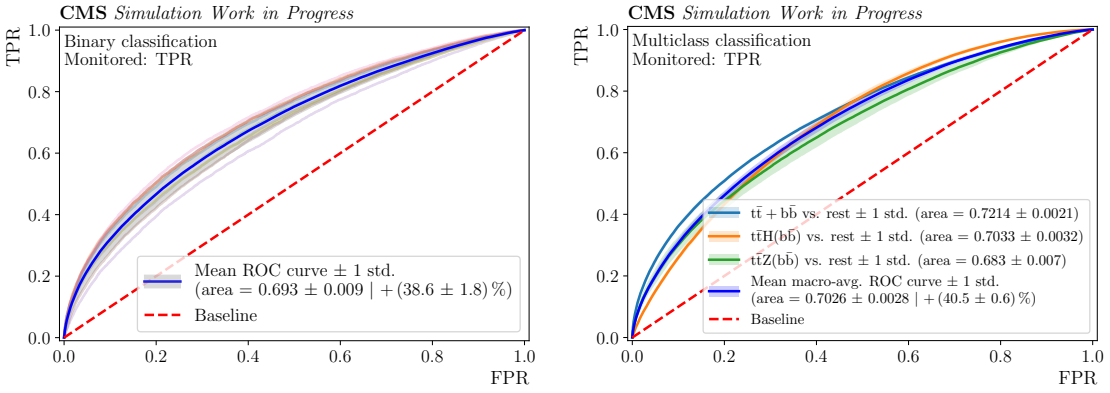


Figure 4.3: **Mean ROC curves pre-removal of outliers.** The top row showcases the mean performance of the binary classifiers trained under monitoring of TPR (a) and loss (b). In addition to the mean ROC curve (dark blue) and its spread (gray band, square root of the unbiased sample variance), which are calculated on the basis of the ROC-AUC values obtained in the ten realized repetitions of a training, the ROC curve of each repetition is displayed in both plots. At the bottom row, the mean performance of the multiclass classification model trained under monitoring of TPR (c) and loss (d) is depicted. The percentage in the legend corresponds to the model performance improvement with respect to a random estimator. For both classification tasks, the spread of the trainings under monitoring of TPR is larger than under monitoring of the loss. Equally, the mean performance of models trained under the latter condition is superior.



(a) (b)

Figure 4.4: **Mean ROC curves post-removal of outliers.** The outlier-adjusted mean ROC curve of the models trained under monitoring of the TPR are depicted for the binary case and the multiclass case in (a) and (b), respectively. In (a) the ROC curve of each repetition is again additionally shown. The percentage in the legend corresponds to the model performance improvement with respect to a random estimator. While the mean performance of the binary classifier has improved after removing the outlier, the contrary occurs to the multiclass classifier as in this case also one too well performing model is discarded, additionally to a too poorly performing model.

From a physics point of view, the additional b jets are the most useful final state objects for classifying the events. Hence, an intuitive approach is to directly provide the information to which category a vertex in the graph data belongs in the form of flags as additional input feature to the trainings. These category flags are discrete features and only contain the values 1 or 0, representing whether a vertex belongs to the particular category or not. Although it is expected to be sufficient to add the AddB flag to the vertex attributes (cf. Equation 4.1), also a feature set where not only the AddB flag but all possible category flags are added is deployed. The naming scheme the feature sets follow is introduced in Fig. 4.5. When, for instance, the feature set `AddB+LTB` is deployed, the vertex attributes of a vertex assigned to the category LepTopB pre-standardization could look like

$$\mathbf{x} = \Big(p_\mathrm{T} = 41.32 \quad \phi = -1.93 \quad \eta = 1.69 \quad M = 8.94 \quad \ldots \quad \mathrm{AddB} = 0 \quad \mathrm{LTB} = 1\Big)^\intercal \,. \tag{4.4}$$

The category flag information is of course not inherent in real, detected data, i.e., on reconstruction-level. Consequently, for instance, another classifier, such as a GNN performing node-level-prediction on the data (cf. Section 4.5), is required for determining this information. For now, generator-level information is used for constructing the category flags. The dependency of the event classification on the jet classification is further analyzed in Chapter 5.

A ranking of all investigated models with respect to their mean performance is illustrated in Fig. 4.6. As expected, introducing AddB flag as additional vertex attribute is very beneficial for the training of both binary and multiclass classifiers. In fact, the model performance can be increased by $(25.5 \pm 1.6)\,\%$ and $(24.6 \pm 0.5)\,\%$, respectively. When adding the flags of the remaining categories to the feature set then in both cases the model performance can be slightly further increased. Therefore it is decided to pursue the further optimization approaches in Section 4.4 with the feature set `extended`. Moreover, due to

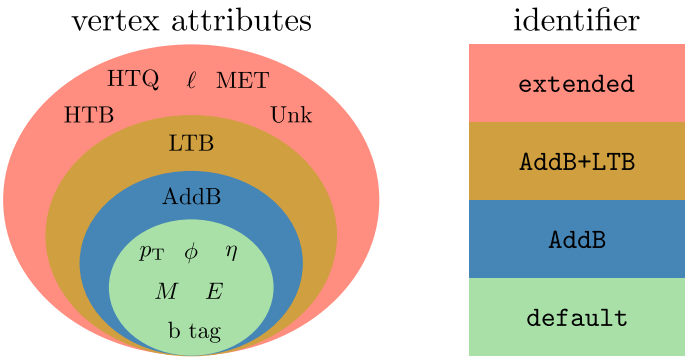


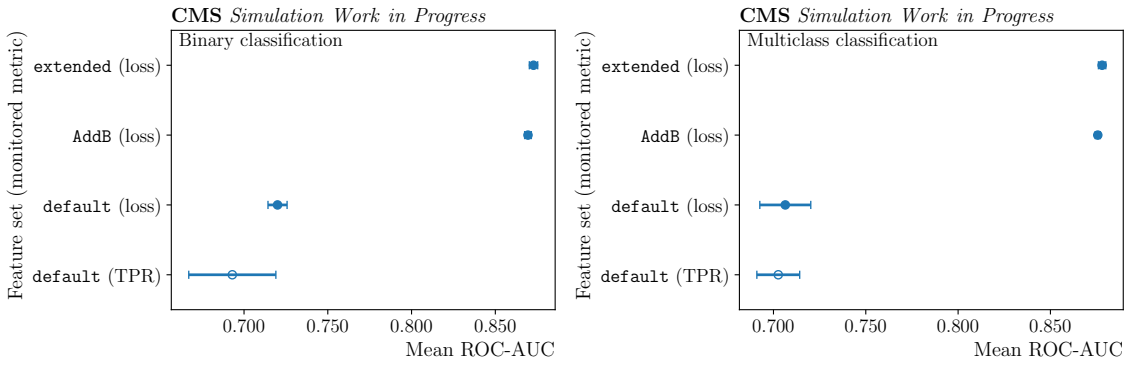
Figure 4.5: **Feature set naming scheme.** The abbreviations AddB, LTB, HTB, HTQ, MET, Unk and ℓ stand for the described category flags and contain the information whether a vertex belongs to the category AddB, LepTopB, HadTopB, HadTopQ, Missing, Unknown or Lepton, respectively. It will be clear from context whether AddB refers to the category flag or the category per se.

the apparent similar impacts of the realized modifications on the binary and the multiclass classifiers, the following studies are applied to the binary case only. It is assumed that the further findings can be adopted to the multiclass case.

4.4 Further Optimization Approaches

So far, only the vertex attributes are modified. However, it is expected to be able to push the model performance even further by using M_{inv} as edge weight instead of ΔR since especially the invariant mass of the additional b jets should give a direct indication of the class of an event. In addition, also two modifications of the original model architecture (cf. Fig. 4.2) are probed. The idea behind these modifications is to deploy both M_{inv} and ΔR as weight for the edges in graphs and profit from both in training and inference. For that, the GGSNN block from before is split into two sub-blocks. In one sub-block the messages are weighted with ΔR only and in the other with M_{inv}. In Fig. 4.7a, these sub-blocks are arranged sequentially. In order to maintain the same total number of GNN layers in the model as before, the two GNN sub-blocks are each only half as deep as before. In total, this model possesses 17593 trainable parameters. The second approach is depicted in Fig. 4.7b. The corresponding sub-blocks are arranged in a parallel manner, leading to 27962 trainable parameters. These two models are referred to as respectively $\text{GGSNN}_{\text{seq}}$ and $\text{GGSNN}_{\text{para}}$ in the following.

As shown in Fig. 4.8, all modifications on the model architecture indeed outperform the conventional GGSNN, used up till now, though the performance gain only ranges from $(0.29 \pm 0.14)\,\%$ to $(0.74 \pm 0.10)\,\%$. On the other hand, the spread of the two best-performing models is approximately only half as big, i.e., their training are, in comparison, much more stable, which is a non-negligible factor, for instance, in terms of the reliability of their predictions.
Furthermore, in agreement with physical intuition, the introduction of M_{inv} as edge weight turns out to be advantageous for the classification task. With this adjustment alone, the model performance is only improved by $(0.70 \pm 0.10)\,\%$ but it should not be overlooked that this improvement equals to an increase in the ROC-AUC value of 0.0061, which is noticeably larger than the spread of the corresponding data points in Fig. 4.8. This is even more remarkable when considering that no other modification on the model architecture can outperform this model, despite of having up to approximately twice as many DOF for adapting to the classification task. Indeed the $\text{GGSNN}_{\text{seq}}$ with $M_{\text{inv}} \to \Delta R$ as edge



(a) (b)

Figure 4.6: **Mean performance ranking of models trained with various feature sets and under different monitored metrics.** The mean ROC-AUC of the best model in the binary case (a) and in the multiclass case (b) are 0.8728 ± 0.0008 and 0.8780 ± 0.0004. Data points with unfilled markers indicate that at least one model of ten repetitions of the particular training has a performance fulfilling the outlier criteria defined in Section 4.3 and the identified outlier(s) is/are therefore discarded. All data points possess an error bar representing the performance spread (square root of the unbiased sample variance), which is calculated on the basis of the ROC-AUC values of all not discarded models (at most ten models, one for each of the ten realized repetitions of training). If an error bar is not visible, then this is due to the fact that the spread is too narrow to be displayed.

weights is ranked higher but the performance increase is only $(0.05 \pm 0.06)\,\%$, i.e., lies within the standard deviation and is hence negligible. That emphasizes again the discriminating power of M_{inv} as edge weight. Also the fact that reversing the order of the deployed edge weights from $\Delta R \to M_{\text{inv}}$ to $M_{\text{inv}} \to \Delta R$ in the sub-blocks in $\text{GGSNN}_{\text{seq}}$ already leads to a performance gain of $(0.41 \pm 0.12)\,\%$ underlines this statement. Apparently, weighting the messages in the message passing step with the invariant mass, supports the model better in extracting higher-level information of the data than using ΔR in the first sub-block. The subsequent sub-block with ΔR as edge weight seems to profit from that extracted information as well. Presumably for that reason, this model is ranked higher than the model with M_{inv} as edge weight only.

When all suggested optimization methods are applied, including the usage of loss as monitored metric during training, the extension of the feature set `default` by category flags and deploying a model architecture using M_{inv} and ΔR as edge weight in that order in subsequent GGSNN blocks, the training process is stabilized on the one hand. On the other hand, the classification ability of the model can be improved by $(26.9 \pm 1.3)\,\%$ to $(0.8793 \pm 0.0004)\,\%$ in comparison to the GLP attempt presented first or by $(75.860 \pm 0.008)\,\%$ with respect to a random estimator. However, it should be noted that so far no regularization methods are applied to the GNN trainings. This would presumably further promote the model performance but unfortunately could not be tested within the time frame of this thesis. It should not be overlooked that the model performances measured are only achieved by deploying generator-level information for constructing the category flags in the input features. The determination of the category flags is nothing else than performing jet identification on the data set prior to the training of the event classifier. This can be fulfilled with, e.g., the help of another classifier, which of course will not perform flawlessly. An exemplary approach for determining the category flags is shown in the next section.

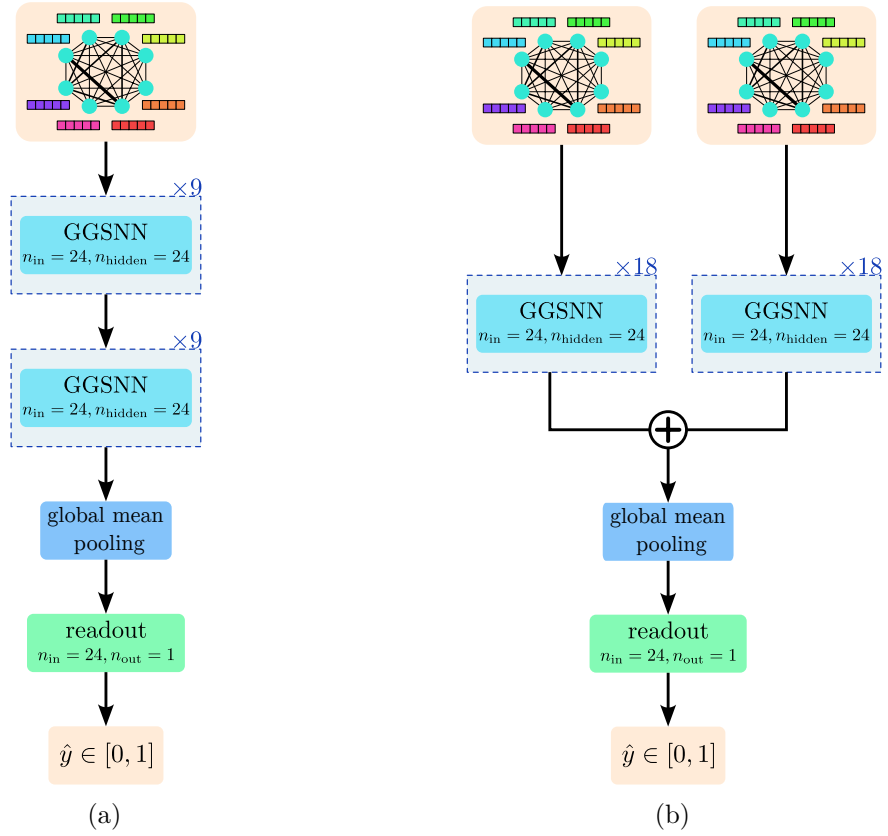


Figure 4.7: **Model architecture variations.** In (a) the GGSNN blocks are arranged in a sequential manner whereas in (b) they are arranged in parallel. The remaining architecture is the same as in the conventional GGSNN, shown in Fig. 4.2.

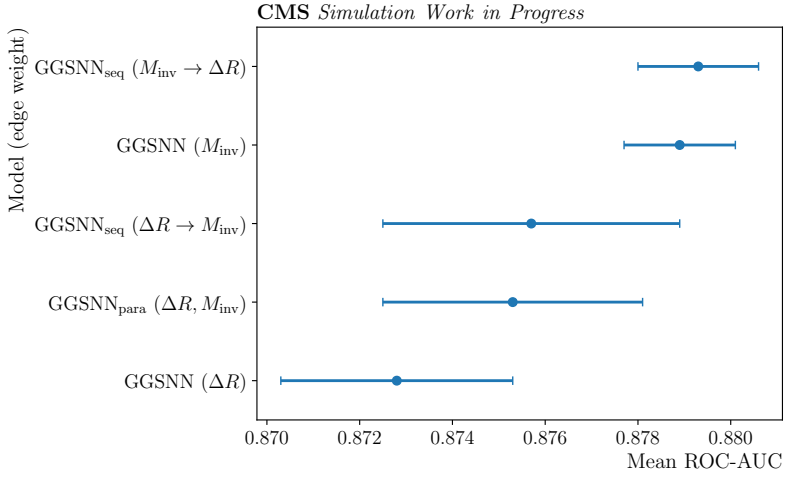


Figure 4.8: **Mean performance ranking of modified models, including the best-performing binary classifier from Fig. 4.6a for reference.** The mean ROC-AUC of the model ranked the highest is 0.8793 ± 0.0004. The error bars represent the performance spread (square root of the unbiased sample variance), which is calculated on the basis of the ROC-AUC values obtained in the ten realized repetitions of a training. The arrow in some labels on the y-axis indicate the order in which the particular edge weights are used in the GGSNN sub-blocks in the sequential approach.

4.5 Applying Preclassified Category Flags to Input Features

Jet identification and assignment in multijet final states have been the focus of interest of many previous works, such as in Ref. [68] and Ref. [77]. In particular, it has been demonstrated in Ref. [68] that GGSNNs outperform DNNs on identifying both additional b jets in an event (2/2 rate) by more than 6 % and on identifying at least one additional b jets in an event (1/2 rate) by more than 1 %. Moreover, the GGSNN does not correctly identify any of the two additional b jets (0/1 rate) in only about 9 % instead of about 16 % of the cases. Accordingly, a GGSNN is selected to be the machine learning technique for preclassifying AddB flags, which are needed as input for training the actual GGSNN performing the event classification. To avoid confusion, the former model is named NLP-GGSNN in the following. Since the data used in Ref. [68] does not fully match the data used in this thesis, it is here refrained from utilizing the hyperparameters of the best-performing model from Ref. [68]. Instead, the exact same hyperparameters as for training a binary event classifier (cf. Table 4.1), whose selection was inspired by the findings in Ref. [68], are applied to the training of the NLP-GGSNN. The only exceptions are that the global mean pooling layer is dispensed with this node-level-prediction task, TPR is again used as monitored metric and the loss function is weighted as described in Ref. [68]. Furthermore, the feature set `default` is deployed and the vertices of each graph to which the two highest NLP-GGSNN output values correspond are identified as additional b jets in the end. Hence, also no classification threshold needs to be determined for calculating the true positive rate. As opposed to the GNNs of the previous sections, all ten repetitions of the training of NLP-GGSNN is processed on CPUs (AMD Ryzen 9 3900X 12-Core Processor) instead of the aforementioned graphics card, yet on the same machine. The underlying reason for this switch is the reproducibility of the results (cf. Section 4.1). Since the NLP-GGSNN response will be further utilized as input to the actual event classification model, the AddB flag predictions are demanded to be deterministic/reproducible, equally to the remaining features used for the training.
With this setup for the NLP-GGSNN, a mean ROC-AUC value of 0.93523 ± 0.00031 is achieved, which is even (6.36 ± 0.06) % superior than the performance of the best-performing GNN for event classification from Section 4.4. The mean TNR $\langle \text{TNR} \rangle$, mean TPR $\langle \text{TPR} \rangle$, mean confusion rate $\langle \text{CR} \rangle$, which corresponds to the frequency that the model mistakes another jet category for the category AddB, as well as the 2/2, 1/2 and 0/2 rates of the NLP-GGSNN evaluated on the test set are summarized in Table 4.2. Although the NLP-GGSNN is trained and evaluated on all classes without further differentiation, the respective values can be extracted for each class separately. This is added to the same table and is relevant for the modeling strategies presented in Chapter 5. It becomes evident that the GNN confuses jets belonging to the categories LepTopB and HadTopB far more often than other categories with additional b jets throughout all classes. The same observation is made with DNNs in Ref. [77]. This probably traces back to the similar b tag values of these jets. The fact that the categories Lepton and Missing are never confused as additional b jet underlines this assumption.

For determining the AddB flags that are provided as input to the event classification model, the best-performing NLP-GGSNN model (TPR = 70.88 %, TNR = 91.42 %) of ten realized NLP-GGSNN trainings is evaluated on the whole data set, i.e., also on data that is used in its training process. This is inevitable due to the restricted number of available samples. However, thanks to EARLY-STOPPING, an overfitting of the trained model on this data is prevented and therefore this unconventional data usage is justified in this case.

In general, two ways of replacing generator-level information in the feature sets by the NLP-GGSNN response are conceivable. Either the actual prediction scores $\hat{\mathbf{y}}$ are used,

Table 4.2: **Performance properties of NLP-GGSNN.** The mean values in this table are calculated over ten realized repetitions of the training. All values are given in %. The subscript of CR designates the category that the NLP-GGSNN confuses with an additional b jet. Due to rounding errors, the sum of $\langle \mathrm{TPR} \rangle$ and $\langle \mathrm{CR} \rangle$ per class is not necessarily 100 %. The same applies to the sum of the 2/2, 1/2 and 0/2 rates per class.

class	$\langle \mathrm{TNR} \rangle$	$\langle \mathrm{TPR} \rangle$	$\langle \mathrm{CR}_{\mathrm{HadTopB}} \rangle$	$\langle \mathrm{CR}_{\mathrm{LepTopB}} \rangle$	$\langle \mathrm{CR}_{\mathrm{HadTopQ}} \rangle$	$\langle \mathrm{CR}_{\mathrm{Unknown}} \rangle$	$\frac{\langle \mathrm{CR}_{\mathrm{Lepton}} \rangle}{\langle \mathrm{CR}_{\mathrm{Missing}} \rangle}$	2/2 rate	1/2 rate	0/2 rate
$\mathrm{t\bar{t}H(b\bar{b})}$	90.69 ± 0.049	68.50 ± 0.17	11.620 ± 0.081	16.139 ± 0.085	2.673 ± 0.025	1.0701 ± 0.0098	0.0 ± 0.0	0.430 ± 0.30	0.51 ± 0.27	0.060 ± 0.053
$\mathrm{t\bar{t}Z(b\bar{b})}$	93.25 ± 0.022	77.247 ± 0.073	7.413 ± 0.039	12.053 ± 0.051	2.466 ± 0.029	0.8209 ± 0.0073	0.0 ± 0.0	0.594 ± 0.12	0.36 ± 0.12	0.049 ± 0.050
$\mathrm{t\bar{t}} + \mathrm{b\bar{b}}$	91.47 ± 0.016	70.695 ± 0.054	10.000 ± 0.044	14.325 ± 0.052	3.082 ± 0.025	1.898 ± 0.016	0.0 ± 0.0	0.4931 ± 0.072	0.428 ± 0.085	0.079 ± 0.064
total	91.36 ± 0.026	70.683 ± 0.089	10.410 ± 0.040	14.896 ± 0.049	2.751 ± 0.023	1.2592 ± 0.0089	0.0 ± 0.0	0.477 ± 0.16	0.46 ± 0.14	0.063 ± 0.030

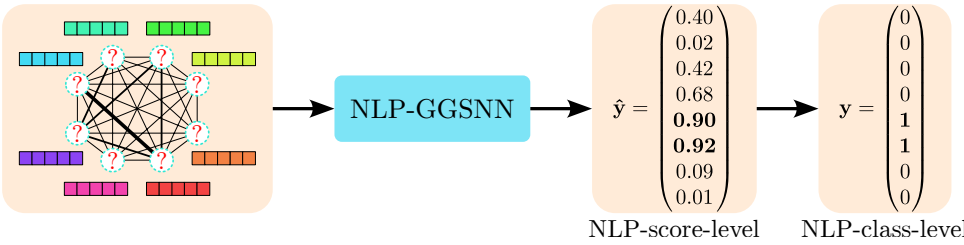


Figure 4.9: **Illustrative representation of different information levels.** Values in the NLP-score-level are continuous whereas the values in the NLP-class-level are discrete.

Table 4.3: **True positive rates achieved with the joint b tag/p_{T} approach.**

category	TPR (%)
HadTopB	65.61
HadTopQ	79.04
LepTopB	52.26
Unknown	62.24
Lepton/Missing	100.00

leading to continuous AddB flags, or the discrete nature of the flags shall be maintained by inserting the classification $\mathbf{y}$ derived from $\hat{\mathbf{y}}$ instead (Fig. 4.9). The former is referred to as NLP-score-level information and the latter as NLP-class-level information in the following. Both information levels are tested in this section and it is clear from the subscript of the particular feature set whether the AddB flag is continuous or discrete. Since the classifier is not flawless and hence its prediction is not equally certain for each event, it is expected that providing this information in the form of continuous AddB flags to the event classification model is fruitful.

The development of a multiclass classification NLP-GGSNN would unfortunately exceed the time frame of this thesis. Therefore, the remaining category flags are instead determined by considering the b tag value of the particular jets in combination with their transverse momentum. In this joint b tag/p_{T} approach, only jets with a b tag value greater than the freely chosen value 0.277 are treated as candidates for the categories HadTopB or LepTopB whereat the jet with the highest p_{T} value (hardest jet) is assigned to the HadTopB category and the second hardest jet to LepTopB. The hardest and second hardest remaining jets are then associated with the category HadTopQ. On the contrary, leptons and neutrinos (category: Missing) can be straightaway identified with 100 % certainty via their mass and η value. The category flags constructed in this way contain of course only discrete values. Despite of this rather unsophisticated approach, clearly more than 50 % of the jets in the whole data set are successfully assigned to the particular jet categories in the majority of the cases, as specified in Table 4.3. This only does not hold for the LepTopB jet assignment. Nevertheless, since it has been shown in Section 4.3 that the highest impact on the performance of the event classification model mainly comes from the additional b jets, it is expected that the poor assignment rate of LepTopB will not noticeably distort the results in the following.

Both the worst- as well as the best-performing model architecture from the ranking in Fig. 4.8 are trained with feature sets with category flags assigned in the described manner instead of generator-level information. The ranking of the performance of these event classifiers can be seen in Fig. 4.10. In correspondence to the expectation, models trained

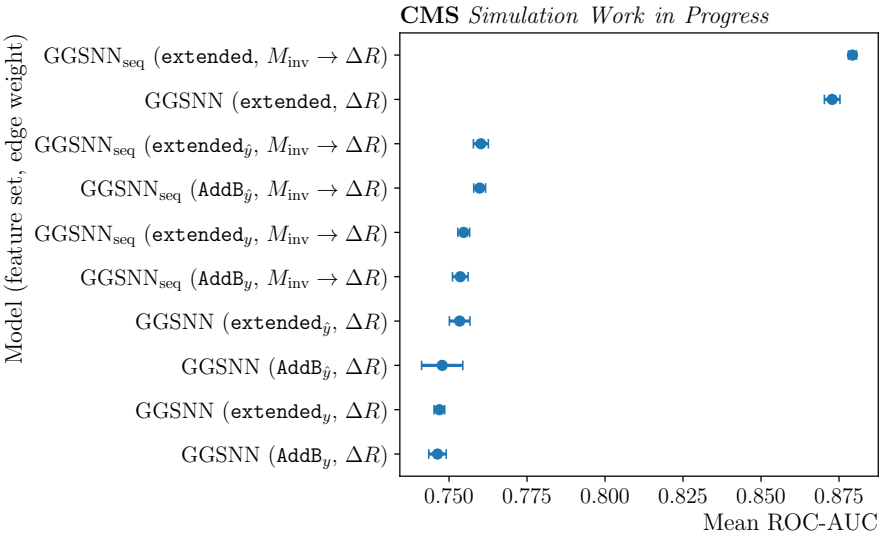


Figure 4.10: **Mean performance ranking of the models trained without generator-level information, including the best- and worst-performing model from Fig. 4.8 for reference.** The mean ROC-AUC value of the best-performing model trained without using generator-level information is 0.7602 ± 0.0008. The error bars represent the performance spread (square root of the unbiased sample variance), which is calculated on the basis of the ROC-AUC values obtained in the ten realized repetitions of a training. The arrow in some labels on the y-axis indicate the order in which the particular edge weights are used in the GGSNN sub-blocks in the sequential approach.

with continuous AddB flags outperform congruent models that only differ in the use of discrete AddB flags. Indeed the model performance declines by $(-14.38 \pm 0.06)\,\%$ on average with respect to the best ranked model, yet it is still $(5.58 \pm 0.29)\,\%$ superior than a model trained on the feature set `default` only (cf. Fig. 4.3b), i.e., without using category flags at all in the feature set. It is especially noticeable that the observations made in the previous sections also apply to the models trained without generator-level information. For instance, the best model architecture stays GGSNN$_{\text{seq}}$ throughout all feature sets considered and also using `extended` instead of `AddB` in otherwise identical models leads to a slight model performance improvement without any exceptions, like observed before. This is in particular interesting when considering that a rather basic approach for determining the non-AddB categories is deployed but the information contained in these category flags are, despite of that, still beneficial for the classifier.

As the performance of the GNNs trained with generator-level information can reach up to be about 76 % superior than a random estimator, when trained with `extended`, it can be generally concluded that the proof of concept is indeed successful, i.e., GNNs are indeed suitable for $\mathrm{t\bar{t}}$+X event classification. A benchmark with respect to the performance and other aspects of GNNs and DNNs is, for instance, conducted in Chapter 7. Nevertheless, it is interesting and reasonable to investigate how much the event classification task is actually dependent on the quality of the category flags provided to the training, for example, to be able to assess whether it is more worthwhile to invest resources in developing a better preclassifier or in further optimizing the event classifier per se (cf. Chapter 5) in the future.

5 Modeling of the Dependency of a GNN-Based Event Classification on the Goodness of the Jet Assignment

In Section 4.3, it was found to be in particular beneficial to extend the vertex attributes, which mainly contain kinematic observables, by category flags for improving the performance of the GNN on $t\bar{t}$+X event classification. These category flags are (originally) a discrete quantity that indicate the category (cf. Section 2.5) to which a vertex in a graph structure belongs but this information is of course not directly accessible at reconstruction-level and need to be determined. In Section 4.5, a GNN performing node-level-prediction in combination with a joint b tag/p_{T} approach are utilized for determining the category flags. Since this preclassification process is not flawless, the performance of the event classifier trained with category flags preclassified in this way is significantly poorer than the performance of an event classifier trained with category flags that are constructed using generator-level information. For example, in order to be able to estimate in a cost-benefit calculation whether it is more desirable to optimize the preclassification process or to pursue a completely different approach to increase the event classification performance, it is worth attempting to model the dependency of the GNN-based event classifier on the jet category assignment, which is part of its input features. In addition, the modeling could provide interesting insights into how the neural network works, like as of which quality level the category flags only play a subordinate role for the performance of the event classification model. This is the case when, for instance, the model performance does not noticeably decrease anymore despite of providing category flags with increasingly poor quality to its training. In reverse, this means that the model successfully adapts to the data and relies on other features instead of category flags for its prediction.
In Section 5.1, two strategies for modeling the dependency are presented. These modeling strategies are subsequently probed for their validity in Section 5.2.

5.1 Modeling Strategies

The idea behind the modeling strategies is that starting from the ground-truth of the category flags, i.e., category flags derived from generator-level information, the category flags of an increasingly larger fraction of the events in the data set are intentionally manipulated. Consequently, by training GNNs with simulated data containing different

fractions of manipulated events, the dependency of the event classifier on the goodness of the pre-assignment of the category flags can be modeled. However, as the results in the previous chapter suggest that the model performance mainly depends on the AddB flag (compared to all other category flags), the modeling can be simplified from modeling the dependency on the goodness of the jet assignment to modeling in particular the additional b jet assignment correctly.

Two strategies are developed in this thesis for modeling the dependency. In the first, rather naive modeling approach (AddB-LTB modeling) only the AddB flag value and the LepTopB flag value of the additional b jet and the jet assigned to the LepTopB category are exchanged in the manipulated events. That is, the actual additional b jet is labeled as a LepTopB jet and vice versa. To avoid any bias in choosing which of the two additional b jets in an event shall be manipulated as LepTopB jet, this is randomized. However the fact that there are only half as many LepTopB jets than additional b jets in an event and hence in the whole dataset, leads to the consequence that when 100 % of the data is manipulated with this approach, still 50 % of the total additional b jets in the data set contain the correct AddB flag whereas 100 % of the LepTopB jets possess a manipulated LepTopB flag. The motivation behind the AddB-LTB modeling is that the trained with NLP-GGSNN confuses these two jet categories the most often in its predictions and hence it could already be sufficient to only manipulate the events in this regard.
On the contrary, all jets are taken into account for manipulating the events in the second, more sophisticated approach (AddB-X modeling). Here, the AddB flag value of the additional b jet in an event is exchanged with the flag value of one of the remaining categories (except for AddB) in correspondence to the (normalized) class specific mean confusion rates of the NLP-GGSNN (cf. Table 4.2). That means that in this approach the confusion rate determines how likely a specific jet category is intentionally mislabeled as additional b jet and vice versa in an event. Moreover, the probability whether both additional b jets in an event shall be manipulated or only one is based on the (normalized) 1/2 and 0/2 rates from Table 4.2. For better comprehension, the normalized rates are included in the appendix (cf. Table C.1).

5.2 Validation

To check the validity of the presented strategies for modeling the real dependency of the event classifier on the quality of the preclassified data, ten data sets with a manipulated proportion of the events from 10 % to 100 % in increments of 10 % are generated for each of the two modeling strategies. To better distinguish the different data sets, the manipulated portion of events in a feature set is encoded in the subscript of the name of the respective feature set. The corresponding fraction of manipulated objects per category in these data sets is specified in Table D.1.
The manipulated data sets are used for training GNNs following the best model architecture so far, i.e., $\text{GGSNN}_{\text{seq}}$ in combination with the usage of M_{inv} and ΔR as edge weights in that order in the sub-blocks. Since the TPR of the NLP-GGSNN used for determining the AddB flags is 71.76 %, i.e., 28.24 % of the additional b jets are assigned to the wrong category, it is expected that the performance of the event classifiers trained with NLP-score-level and NLP-class-level information lie within the performance of models trained with data sets containing 50 % and 60 % of manipulated events, respectively. This expectation applies to both modeling strategies.

The performance of the models trained with data sets manipulated as suggested by AddB-LTB modeling is summarized in Fig. 5.1. Interestingly, the worst-performing event classifier

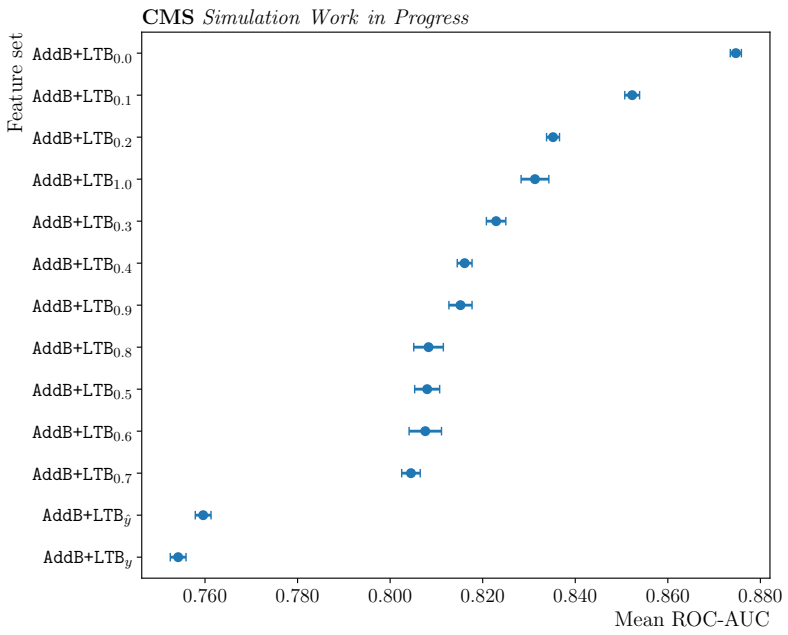


Figure 5.1: **Mean performance ranking of models trained with data sets manipulated according to AddB-LTB modeling, including models trained with actual preclassified category flags for reference.** In addition to the fact that the performance of the two models trained with actual preclassified category flags are both far behind the modeled performances, also models trained with a higher fraction of manipulated events are ranked higher than models trained with better category flags quality. Both is not expected and desired. The error bars represent the performance spread (square root of the unbiased sample variance), which is calculated on the basis of the ROC-AUC values obtained in the ten realized repetitions of a training.

is not the model trained with fully manipulated events but with 70 % manipulated events. The former is even the fourth best model in the ranking. Equally, models trained with $\texttt{AddB+LTB}_{0.8}$ and $\texttt{AddB+LTB}_{0.9}$ are ranked higher than trainings with a smaller portion of manipulated events, which is however only counterintuitive at first sight. As only the values of the AddB flags and LTB flags are exchanged in this naive modeling approach, at a certain point probably no further confusion is added to the data set. Yet, this turning point is certainly not at a fraction of 50 % manipulated events in the data set as there are twice as many additional b jets than LepTopB jets in an event. The GNNs might have adapted to that, which can be verified by unveiling these black-box models with xAI methods (cf. Section 6.2). Accordingly, it is hardly surprising that this modeling strategy turns out to be too naive for properly capturing the real dependency of the event classifier on the preclassification. Both models trained with $\texttt{AddB+LTB}_{y}$ and $\texttt{AddB+LTB}_{\hat{y}}$ have a performance about 6 % lower than the worst-performing model trained with a manipulated data set.

The models trained with data manipulated according to AddB-X modeling behave more as expected in comparison to the previous modeling strategy (cf. Fig. 5.2). The higher the manipulated portion of events in the data the worse the resulting performance of the model trained with these data without any exceptions. Equally, it can be observed that the spread of the model performance is larger for models that are provided with lower quality data for training, i.e., their training stability decreases. Apart from that, it is noticeable that also the difference in mean performance of the models decreases with decreasing data quality. In other words, beyond a certain proportion of manipulated events, the impact of the quality of the AddB flags on the model performance diminishes as the models have

presumably learnt to rely less on this obscure information and focus on other features instead, which is desirable since this shows that they indeed adapt to the provided data. The most important finding is clearly that the mean performance between models trained with either $\texttt{AddB}_{\hat{y}}$ or $\texttt{extended}_{\hat{y}}$ exactly lie in between the predicted performance range of models trained with 50 % and 60 % manipulated events, respectively. The mean performance of models trained with either $\texttt{AddB}_{y}$ or $\texttt{extended}_{y}$ are outside of this range, yet are still in correspondence with the performance of models trained with $\texttt{AddB}_{0.6}$. Hence, on the one hand, it is shown that it is well-justified to only focus on manipulating the AddB flags in the data sets correctly and not necessary to invest resources in finding a more sophisticated modeling strategy that captures the correct manipulation of the non-AddB category flags as well. On the other hand, the validity of the AddB-X modeling strategy for capturing the real dependency of the event classifier performance on the provided quality of the AddB flags is successfully demonstrated. Moreover, thanks to the AddB-X modeling, it can be derived that increasing the TPR of NLP-GGSNN by just 0.17 % leads to an expected performance gain of the event classifier by about 2 %, i.e., it is indeed worth it to develop a better preclassifier in the future. A possible candidate for this could be a multi-task model that learns to identify the additional b jets in the events and classify the respective events at the same time as opposed to the sequential approach used so far with two separate sequential trainings, one for identifying the additional b jets and one for event classification. However, a subsequent further optimization of the preclassifier is probably not reasonable, since a further 2 % increase in performance of the event classifier requires an improvement of the TPR of the preclassifier by approximately 6 %.

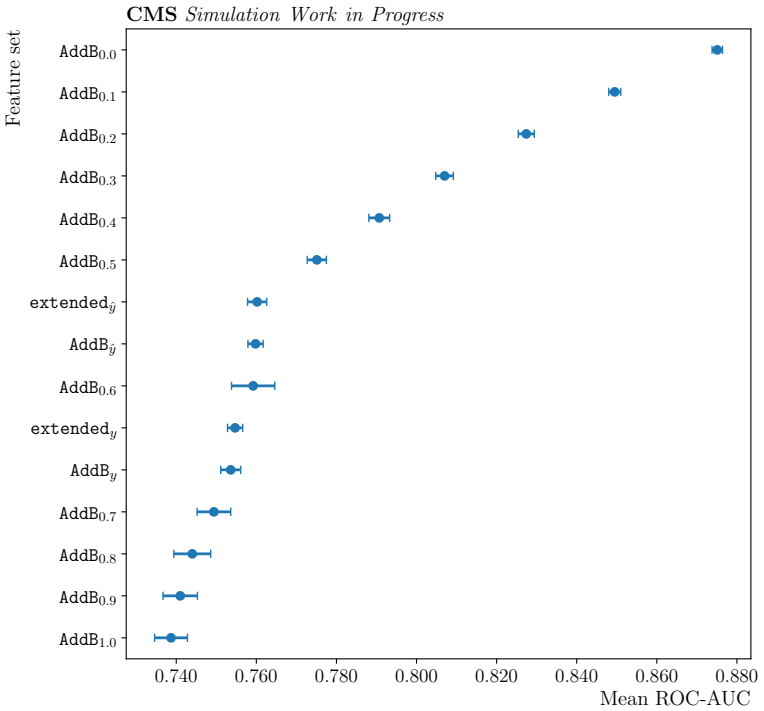


Figure 5.2: **Mean performance ranking of models trained with data sets manipulated according to AddB-X modeling, including models trained with actual preclassified category flags for reference.** As desired and expected, the higher the proportion of manipulated events used for training, the worse the particular model performance. Moreover, the performance of the models trained with NLP-score-level and NLP-class-level information are ranked exactly within the predicted range. The error bars represent the performance spread (square root of the unbiased sample variance), which is calculated on the basis of the ROC-AUC values obtained in the ten realized repetitions of a training.

6 In-Depth Analysis of GNNs by Applying Explainable AI Methods

The focus of this chapter is on examining the underlying decision basis for the predictions of a selection of models from previous chapters by applying explainable AI methods to them. The explainable AI methods considered in this chapter are the GNNExplainer (GNNX, cf. Section 3.3.1) and the Taylor coefficient analysis (TCA, cf. Section 3.3.2). As opposed to TCA, GNNX requires training, in which the edge and feature masks that serve as explanation for a model are adjusted. However, since the GNNExplainer does not take edge features into account and it is revealed in Section 4.4 that edge weights indeed have a noticeable impact on the model performance, it is chosen to only train and consider the feature mask in the following. For feasibility reasons, the default choice in PyG of using only 100 attempts for adjusting the feature mask remains unchanged. Both the calculations of the TCA and the GNNX trainings are carried out on CPUs (cf. Section 4.5) in order to guarantee reproducibility of these explanations. As the spread of the model performance of each configuration is comparatively narrow, it is decided to analyze the best-performing model of ten realized repetitions only.

Both xAI methods only provide single-instance explanations, i.e., they are only capable of explaining one single prediction of the trained model under scrutiny at a time. Thus, in order to get an idea on how the model generally extracts its information from the inputs, the trained model is evaluated on the test set of size $S = 37827$ events and for each prediction the explanation is calculated separately by the xAI methods. The overall explanation for the model then simply corresponds to the arithmetic mean over the explanations or, since Taylor coefficients can also be negative, rather the arithmetic mean over absolute values. Incidentally, as the Taylor coefficient analysis outputs are of the same dimension as the NN inputs, i.e., of size $\mathbb{R}^{n\times m}$ for each explained graph instance, the arithmetic mean corresponds for TCA applied to GNNs to averaging over the total number of vertices in the test set instead of over S. As a reminder, n is the number of vertices in the graph under scrutiny and m denotes the number of attributes assigned to each vertex.

The goal is to gain, with the explanations, a further understanding of the achieved model performance in previous chapters and therewith probing their reliability as well as leveraging the optimization of the event classification model (Section 6.1). Furthermore, the reasons

for the observed apparent peculiar behaving models in AddB-LTB modeling shall be unraveled (Section 6.2). At the same time, the plausibility of the explanations provided by GNNX and TCA is inspected in each section.

6.1 Identifying the Decision Basis of Models Trained with Different Information Levels

As observed in Section 4.5, GNNs trained with NLP-score-level (NLP-score-level GNN) or NLP-class-level (NLP-class-level GNN) information perform about 14 % worse than GNNs trained with generator-level information (generator-level GNN). Hence, it is interesting to investigate whether GNNs trained with generator-level information indeed identify the same features and vertices as important for its classification as expected from a physics perspective. Moreover, it can be investigated in this manner whether more underlying explanations exist for the performance drain of the poorer performing GNNs, except for the worse data quality of the category flags. The xAI methods are applied to the best-performing $\text{GGSNN}_{\text{seq}}$ model (according to the ranking in Fig. 4.10) trained with the feature set `extended`, $\texttt{extended}_y$ and $\texttt{extended}_{\hat{y}}$, respectively.

For facilitating the comparison of the rankings in the following, the measure $\langle \Delta r \rangle$ is proposed in this thesis. It is obtained by calculating the difference between the ranks of each object q in ranking A and ranking B and subsequently taking the arithmetic mean over the absolute rank differences. Hence, it holds

$$\langle \Delta r \rangle = \frac{1}{M} \sum_{i=1}^{M} \Delta r(q_i) \tag{6.1}$$

$$= \frac{1}{M} \sum_{i=1}^{M} \left| r_A(q_i) - r_B(q_i) \right|, \tag{6.2}$$

where M is the number of objects q in a ranking. The range of this measure is $\left[0, \Delta r^*_{\text{max}}\right]$ with Δr^*_{max} being $\frac{1}{M} \cdot \frac{M^2-1}{2}$ if M is odd and $\frac{1}{M} \cdot \frac{M^2}{2}$ otherwise. A detailed derivation of this range is included in Appendix E. The closer the value of this metric is to zero, the more alike are the two compared rankings. Alternatively, the conformity between two rankings can be expressed as $1 - \langle \Delta r \rangle / \Delta r^*_{\text{max}}$.

6.1.1 Category Importance and Category Specific Feature Importance

With GNNX, there is the possibility to calculate the importance of vertices or the vertex specific feature importance for the GNN response, for instance. As the impact of the vertices belonging to the same category are summarized in an average in this thesis, strictly speaking, the category importance and the category specific feature importance are examined in the following.

The impact of the different categories on the response of the GNNs under scrutiny according to GNNX is showcased in Fig. 6.1. It becomes evident that the most important category by far is AddB, regardless of the information level of the category flags provided to the training. This fully matches physics expectations. The importance of the other categories for the GNNs are, on the contrary, rather obscure as the three GNNs under scrutiny do not agree in their rankings anymore. But at the same time, it should not be overseen that the difference in the actual calculated importance of these categories are not as distinct

as between AddB and the second highest ranked category in all three rankings. Hence, it can be derived from that observation that there is no clear importance hierarchy in the remaining categories for trained GNNs. This is presumably physically reasonable as well. When, for instance, looking at the distribution of the transverse momentum of the b jets originating from the top quark decays (HadTopB and LepTopB), no discriminating power for the $t\bar{t}$+X events can be spotted (cf. Fig. F.1). The same holds for the invariant mass of the $t\bar{t}$ system (cf. Fig. F.2). Of course, these are only three out of various numbers of conceivable observables and multivariate methods possibly are able to extract relevant information also from these observables for classifying the $t\bar{t}$+X events. Hence, further studies are required.

Since AddB turns out to be the most important category, it is also worth looking at the feature importance hierarchy of this very category (cf. Fig. 6.2). It is striking that these GNNX-based rankings are almost identical for all three GNNs. The conformity between them is above 90 %. In all three cases, the AddB flag, the transverse momenta and the b tag of the final state objects are the three highest ranked features. At the same time, all remaining category flags are positioned in the lower half of the ranking without any exceptions. This is no surprise considering these flag values are zero at all times for vertices representing additional b jets.

For the sake of completeness, the rankings containing the category specific feature importance for the remaining categories are included in the appendix (cf. Figs. F.3-F.8). Like for AddB vertices, the most important category flag in these ranking is the one denoting the actual category of the particular vertex while the other category flags are only of minor importance for the GNN response. This solely do not apply to the feature importance rankings of the category LepTopB for the NLP-score-level GNN, i.e., the category that the NLP-GGSNN confuses with additional b jets the most often. Instead, the AddB flag is the most important category flag and the LTB flags is only the second most important category flag. This observation is regarded as the final proof of the validity of the assumption made in Section 4.5, i.e., providing continuous AddB flags, indicating the uncertainty of the preclassifier, is indeed fruitful information for the event classification model. As a matter of fact, in that ranking, the AddB flag is even the most important feature for the NLP-score-level GNN.

6.1.2 Feature Importance

The feature importance rankings for the generator-level, the NLP-score-level and the NLP-class-level GNN, calculated by GNNX, are depicted in Fig. 6.3. Also from a global perspective, the AddB flag is the most important feature for each GNN albeit the additional b jets are not correctly assigned by the NLP-GGSNN in about 30 % of the cases. In fact, this is as well reflected in the ranking. It is noticeable that only for the generator-level GNN the AddB flag is the most important flag by far. For the other two GNNs, the differences in the impact of the features are not as distinct. Yet, it is unexpected that the LTB flag is only one of the least impactful features for the two GNNs as well, as the NLP-GGSNN confuses additional b jets with jets assigned to this category the most often. About 14 % actual additional b jets are misidentified as LepTopB jets to be precise (cf. Table 4.2). Apart from the AddB flag, the transverse momentum and the b tag of the final state objects are identified as relevant information for accomplishing the classification task, while ϕ is a rather not valuable observable throughout all GNNs. This also corresponds to physics expectations since b jets radiated by top quarks are highly more probable to achieve greater transverse momentum than additional b jets due to the comparatively large top quark mass. Hence, this feature supports the neural network in finding the correct additional b jets and therewith assists the event classification.

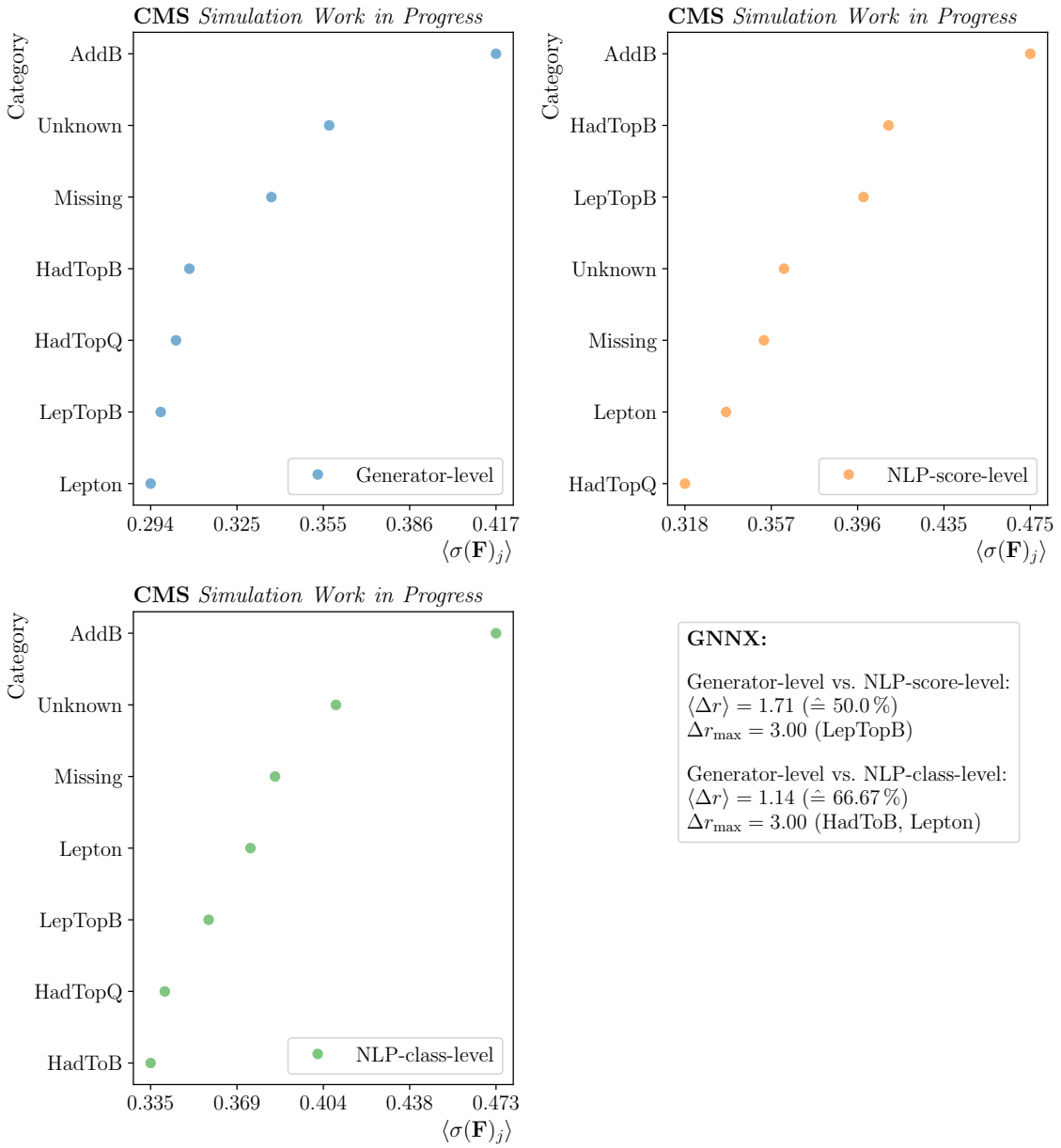


Figure 6.1: **Ranking of the GNNX-based category importance for the generator-level (top left), NLP-score-level (top right) and NLP-class-level (bottom left) GNN response.** As expected, the most important category by far is AddB throughout all GNNs. The percentage in the parentheses corresponds to the conformity $1 - \langle \Delta r \rangle / \Delta r^*_{\text{max}}$ of the compared rankings. The variable Δr_{max} denotes the highest number of rank difference between the compared rankings and should not be confused with Δr^*_{max}, which denotes the maximum mean number of absolute rank differences between the compared rankings. The objects to which this applies are specified in the subsequent parentheses.

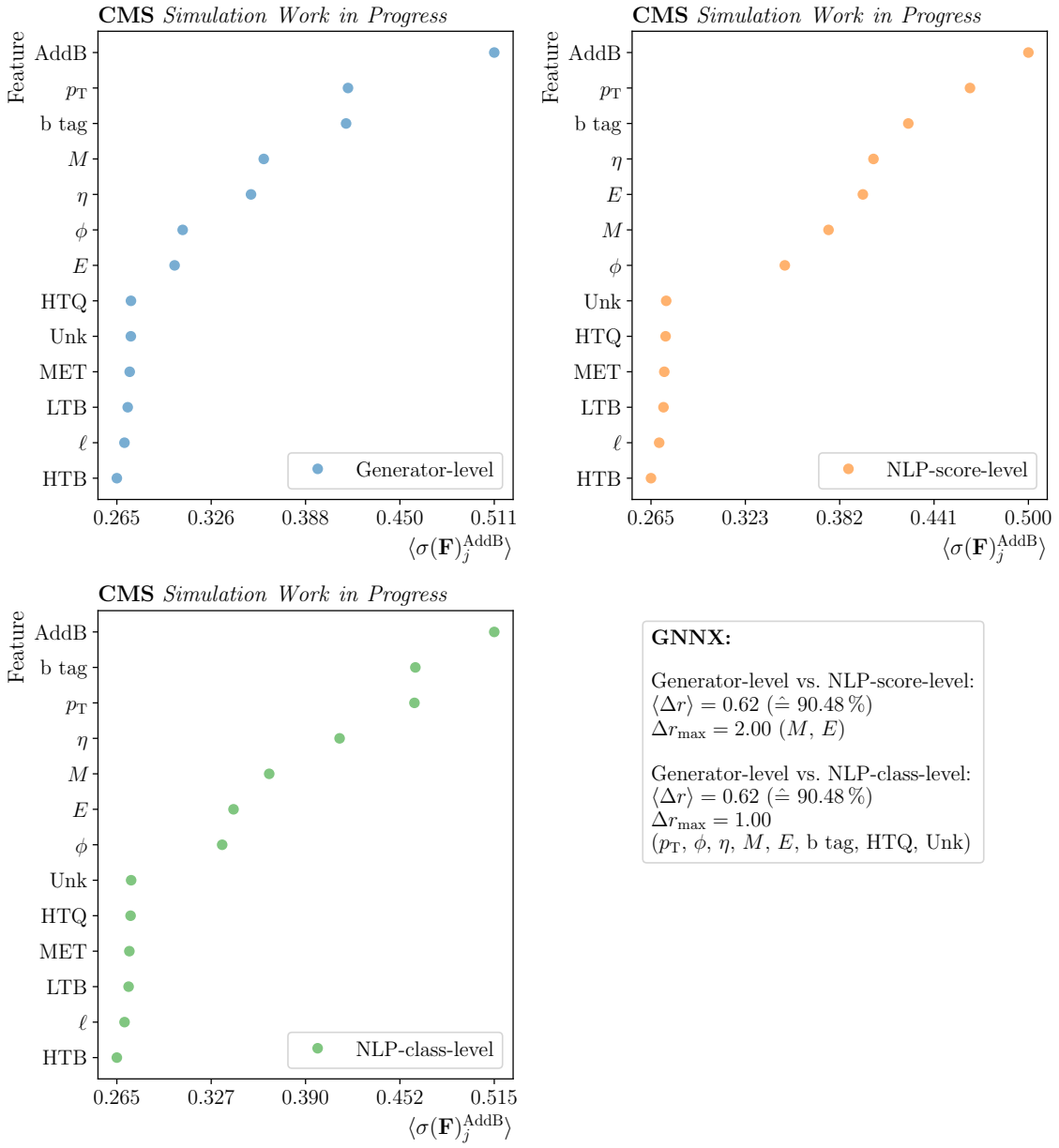


Figure 6.2: **Ranking of the GNNX-based AddB specific feature importance for the generator-level (top left), NLP-score-level (top right) and NLP-class-level (bottom left) GNN response.** Across all three rankings, AddB flags, p_T and the b tag are identified as the most important features of AddB vertices for the GNN response. All other category flags occupy the lower positions in the rankings without exception. The percentage in the parentheses corresponds to the conformity $1 - \langle\Delta r\rangle/\Delta r^*_{\text{max}}$ of the compared rankings. The variable Δr_{max} denotes the highest number of rank difference between the compared rankings and should not be confused with Δr^*_{max}, which denotes the maximum mean number of absolute rank differences between the compared rankings. The objects to which this applies are specified in the subsequent parentheses.

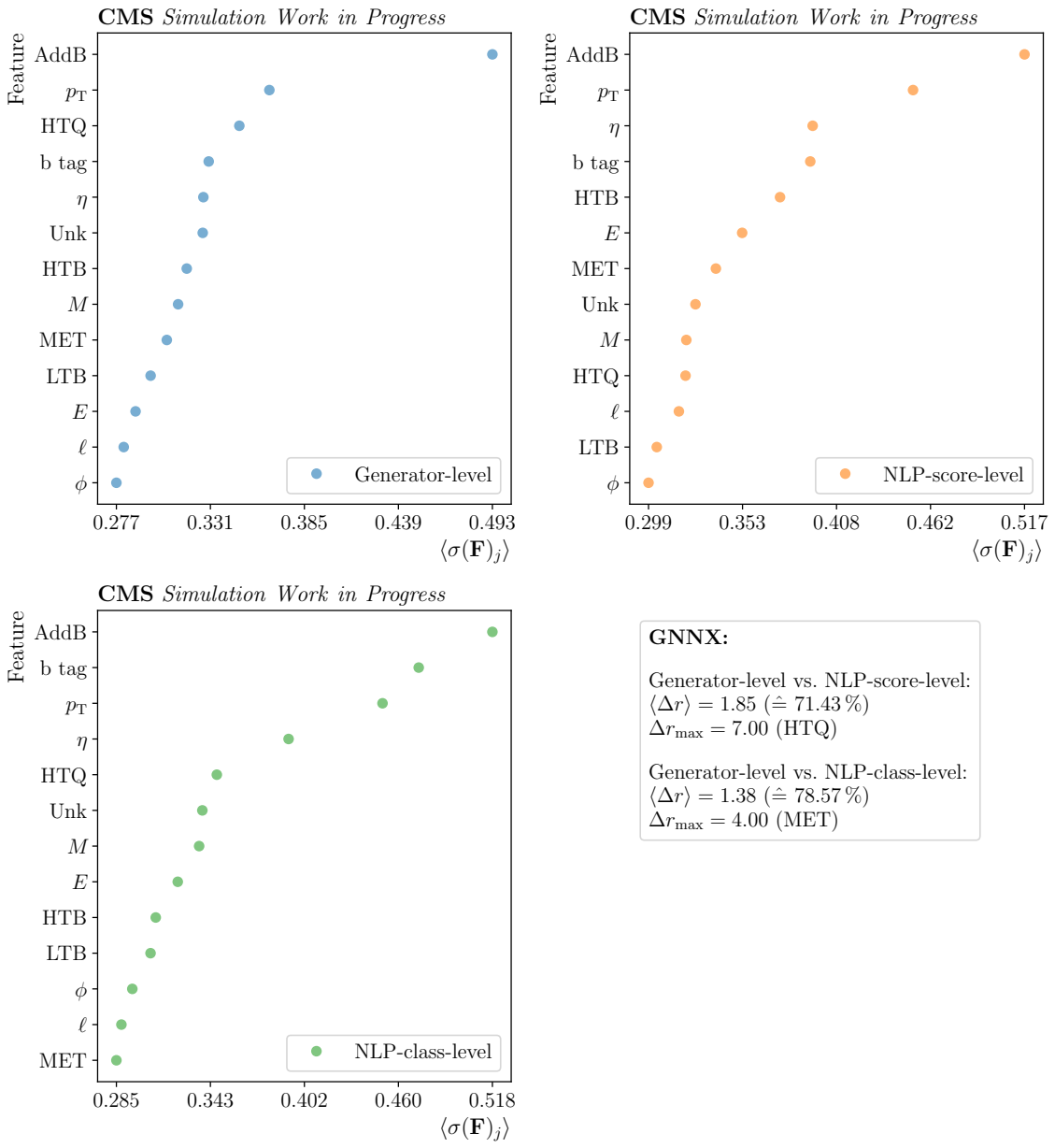


Figure 6.3: **Ranking of the GNNX-based feature importance for the generator-level (top left), NLP-score-level (top right) and NLP-class-level (bottom left) GNN response.** The information captured in ϕ forms the smallest part of the decision basis and the information contained in AddB flags forms most of the decision basis for all GNNs. The difference in importance between the AddB flag and the second highest ranked feature is however for both GNNs trained on actual preclassified flags not as distinct as for the generator-level GNN. The percentage in the parentheses corresponds to the conformity $1 - \langle\Delta r\rangle / \Delta r^*_{\max}$ of the compared rankings. The variable $\Delta r_{\max}$ denotes the highest number of rank difference between the compared rankings and should not be confused with $\Delta r^*_{\max}$, which denotes the maximum mean number of absolute rank differences between the compared rankings. The objects to which this applies are specified in the subsequent parentheses.

The first-order TCA (cf. Fig. 6.4) identifies, like GNNX, the AddB flag as the most relevant category flag by far. Except for the NLP-class-level GNN, it is at the same time the most important feature for the predictions of the GNNs here as well. Yet, it must be noticed that the TCA sees category flags in general as rather impactful to the prediction throughout all GNNs, while the predictions are less dependent on the b tag value according to the first-order TCA. However, the b tag value is among the four most relevant features in the GNNX-based ranking in Fig. 6.3. At this point, it cannot fully be judged yet whether the explanations provided by the first-order TCA or GNNX are more reasonable. As on the one hand, a performance gain can in fact be achieved when the feature set `extended`, containing all category flags, is deployed for training instead of only `AddB` (cf. Chapter 4). But on the other hand, the improvement is below 1 %. Thus, both rankings of the importance of the category flags are reasonable. The rank of the b tag value expected from a physics point of view is also ambiguous. It is conceivable that the b tag value is only of minor importance for a classification between $t\bar{t} + b\bar{b}$, $t\bar{t}H(b\bar{b})$ and $t\bar{t}Z(b\bar{b})$ samples as each of them contain b jets. Yet, at the same time, it is possibly a helpful feature for identifying additional b jets, which are crucial for the event classification, if the AddB flags are not correctly assigned. Within the three GNNs under scrutiny, the NLP-class-level GNN is the worst-performing model and should therefore also be the one trained with the worst AddB flag quality. The second-order TCA, which reveals the impact of relations across input features to neural network responses, supports this assumption (cf. Fig. F.9). Indeed, for that particular GNN, the b tag value is a more relevant feature than for the remaining two GNNs according to both the first-order TCA and the corresponding GNNX ranking. Though, both xAI methods agree on p_T being the most valuable kinematic jet feature and as well agree on the rather minor importance of ϕ for the GNN response, regardless of the information level with which the GNN is trained. As described above, this is also expected in terms of physics. Consequently, although the identified dependencies for GNN responses are not really congruent between the xAI methods – the conformity between both GNNX- and TCA-based rankings is for all GNNs only 54.76 % (cf. Fig. F.10) – both xAI methods appear to deliver plausible explanations so far.

6.1.3 Conclusion

In the majority of the cases, the GNNX-based rankings of GNNs trained without generator-level information are quite similar to the GNNX-based ranking of the generator-level GNN according to the $\langle \Delta r \rangle$ measure. To be precise, the conformity is above 80 % in two-thirds of the GNNX-based rankings. In first-order TCA rankings, the conformity between the rankings for the GNNs trained on different information levels is at about 70 %. Thus, according to both xAI methods both the NLP-score-level and NLP-class-level GNN indeed have learnt to extract similar high-level features from the provided data as the generator-level GNN or, to put it the other way around, xAI reveals that all GNNs under scrutiny behave as expected and hence are reliable. Accordingly, it is presumably more reasonable and recommended to enhance the quality of the provided AddB flags by developing a superior preclassifier (as also concluded in Section 5.2 through AddB-X modeling) or construct new, more discriminating features for optimizing the event classifier than investing resources in finding a more suitable GNN architecture for achieving better performance results, for example.

6.2 Evolution of the Feature Importance in AddB-LTB Modeling

In Section 5.2, it is observed that some GNNs trained with data sets in which beyond 70 % of the events are manipulated according to the AddB-LTB modeling have a peculiarly

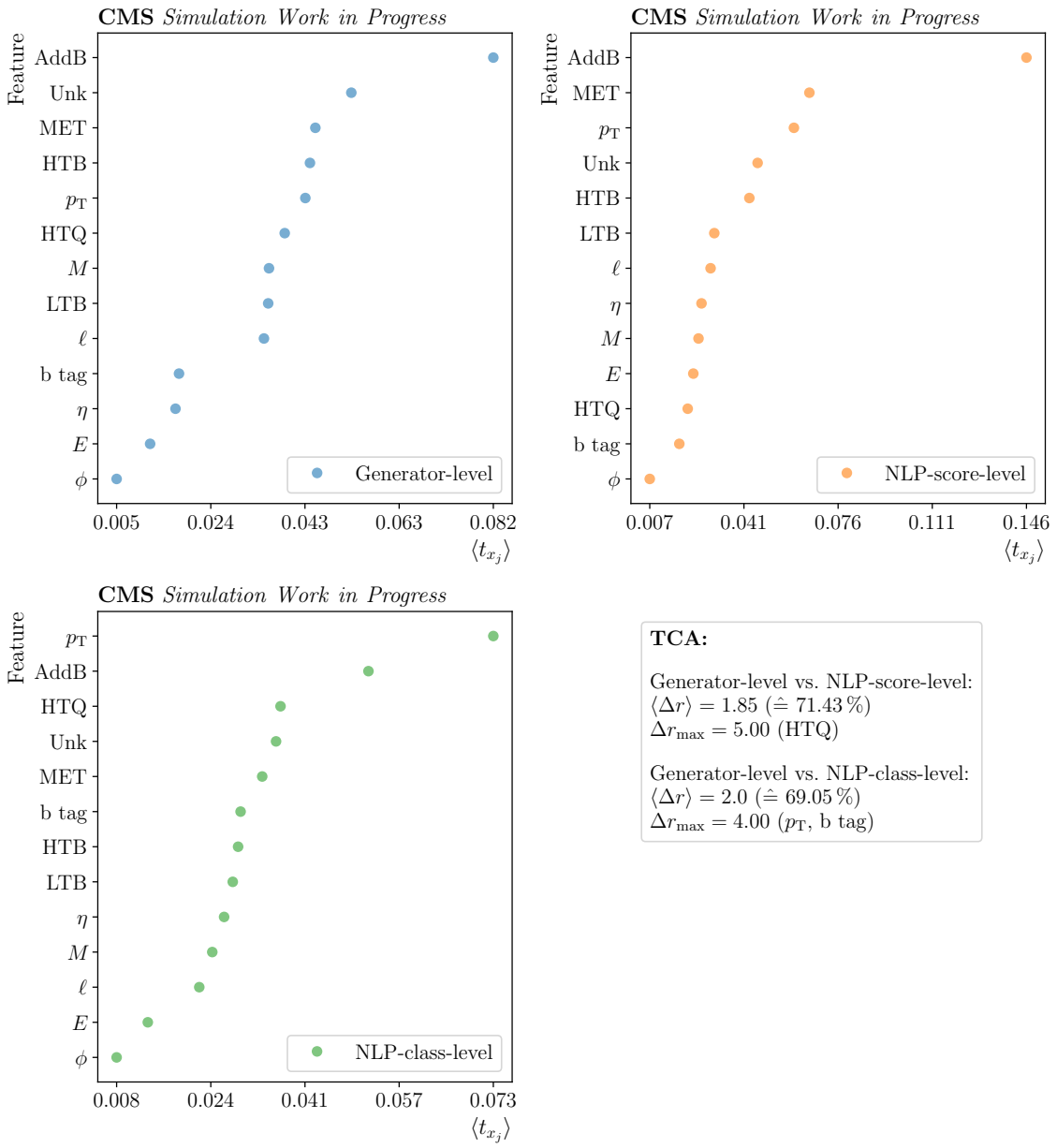


Figure 6.4: **First-order Taylor coefficient ranking of the generator-level GNN (top left), NLP-score-level GNN (top right) and NLP-class-level GNN (bottom left).** The top positions of the rankings are mainly covered by category flags. Among all of them, the AddB flag turns out to be the most important category flag by far throughout all three rankings. The most important kinematic jet feature is p_T and the least important feature is ϕ. This also holds for all GNNs under scrutiny. The percentage in the parentheses corresponds to the conformity $1 - \langle \Delta r \rangle / \Delta r^*_{\max}$ of the compared rankings. The variable $\Delta r_{\max}$ denotes the highest number of rank difference between the compared rankings and should not be confused with $\Delta r^*_{\max}$, which denotes the maximum mean number of absolute rank differences between the compared rankings. The objects to which this applies are specified in the subsequent parentheses.

high mean ROC-AUC value. In fact, these GNNs were superior to GNNs trained with a smaller portion of manipulated events, i.e., with better data quality. This appears to be counterintuitive at first sight but could probably be traced back to the fact that beyond a certain fraction of manipulated events, no further confusion or manipulation is added to the data set, since the concept inherent in AddB-LTB modeling is simply to exchange the values of AddB and LTB flags. In order to verify this, all eleven models that are trained with different fractions of manipulated events are further investigated with the help of xAI. The xAI methods are expected to reveal that with increasing fraction of manipulated events, the importance of the AddB flag is decreasing while the importance of the LTB flag should simultaneously increase.

Like in the previous section, the xAI methods are only applied to the best-performing model of ten repetitions, yet instead of evaluating the trained best-performing model on the full test set of size S, for feasibility reasons, only 5 % of the test set are provided to GNNX for training the feature mask in this section. This has the advantage that the duration for obtaining the feature importance with GNNX reduces from approximately $\mathcal{O}(10\,\mathrm{h})$ to $\mathcal{O}(30\,\mathrm{min})$. For comparison, the duration required for calculating the Taylor coefficients up to the second-order is only about 15 min. Accordingly, the TCA is still performed on the entire test set. Empirically, the reduction of the samples used for evaluating the trained models appears to have basically no impact on the actual ranks of the objects in GNNX-based rankings, as shown in the appendix (cf. Fig. F.11-F.13) for the models considered in Section 6.1. Therefore, this approach is justified.

Figure 6.5 depicts the normalized explanations provided by GNNX and by the first-order TCA in dependence of the deployed fraction of manipulated events in the GNN inputs. In this manner, the evolution of the basis for the GNN response with respect to the provided data quality can be examined. It becomes evident that the GNNX explanations are not as distinct as the ones delivered by TCA. Nevertheless, both xAI methods agree on ϕ being the least important feature regardless of the data quality. Yet, the first-order TCA and GNNX are not equally powerful in explaining the peculiar behavior of the aforementioned GNNs. The first-order Taylor coefficient analysis demonstrates clearly that the AddB flag is the by far most important feature for GNNs trained with a manipulated fraction of events below 70 %. At the same time, it shows that the impact of the LTB flag on the GNN response increases, as expected, with increasing manipulated fraction of events until it becomes clearly the most important feature as of a fraction of 80 % manipulated events. This is indeed plausible as, for instance, for a completely AddB-LTB manipulated data set, only 50 % of the additional b jets can be identified via the actual AddB flag while, due to the flag swap, 100 % of the LTB flags points to an additional b jet.
At a manipulated fraction of 70 %, the importance of the AddB and the LTB flag is about the same, i.e., the corresponding GNN is probably confused the most by the manipulated data set among all GNNs. The comparison of the fraction of additional b jets identifiable via the AddB flag and the fraction of additional b jets identifiable via the LTB flag (cf. Fig. F.14) reveals indeed that the information content of AddB and LTB flags is indeed the most similar for the data set with 70 % manipulated events. In data sets with a fraction of manipulated events higher than 70 %, more additional b jets are identifiable via the LTB flag than via the AddB flag. The observation that a GNN trained with $\texttt{AddB+LTB}_{0.7}$ is the most confused perfectly corresponds to the observation that this GNN is the worst-performing GNN in this scope as well. In contrast to that, none of these findings is discovered by GNNX. The GNNExplainer only gives a light indication of the increasing importance of the LTB flag but, according to GNNX, the AddB flag contains

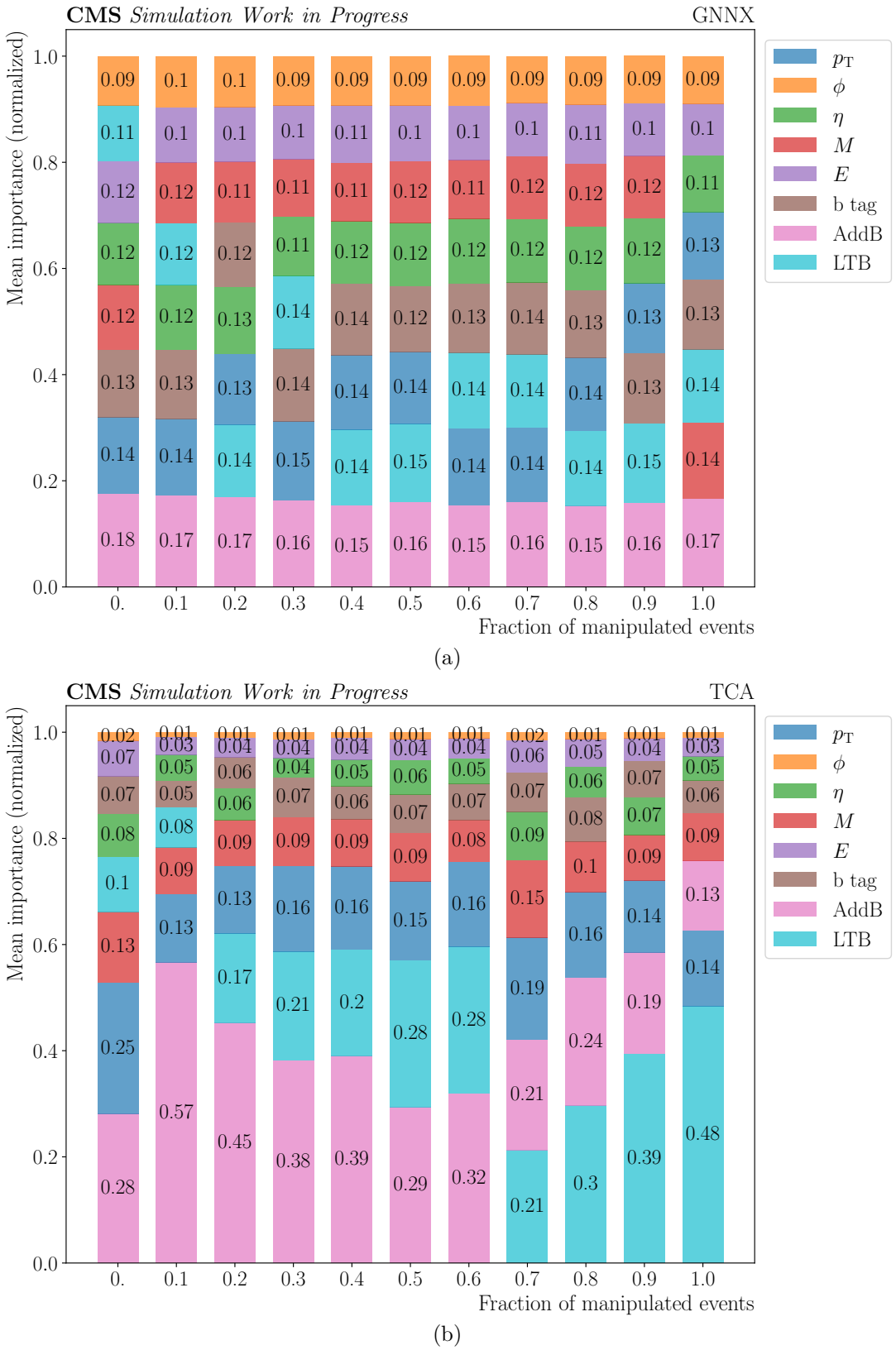


Figure 6.5: **GNNX-based (a) and TCA-based (b) feature importance in dependence of the proportion of AddB-LTB manipulated events in the deployed data sets.** Each bar is normalized to a total height of one. The explanations provided by GNNX are not as distinct as the ones obtained with TCA. Nevertheless, both xAI methods successfully identify ϕ as the least important feature regardless of the fraction of manipulated events in the deployed data sets. Yet, only TCA reveals that there is a change in focus from AddB flags as the most important feature to LTB flags as of a manipulated fraction of 70 %.

the most valuable information for the GNN response throughout all data sets. But this is clearly not possible due to the exchange of the values in the LTB and AddB flags. Thus, the GNNExplainer fails in explaining the performance of the worst-performing model as well as in explaining the apparent superior classification ability of models trained with a manipulated fraction of above 70 %, which is not only successfully explained by the TCA but can also be derived from Fig. F.14.

Consequently, at least for this application, the Taylor coefficient analysis seems to deliver more plausible explanations than GNNX, while requiring a significantly lower time for calculating the explanations as well. Only the Taylor coefficient analysis is able to successfully shed light on the behavior of the, at first sight, peculiarly behaving GNNs and verifies the assumption made at the beginning that this is caused by the fact that no further confusion is added to the data set as of a certain fraction of manipulated events.

7 Benchmark Study on Equivalent GNNs and DNNs

Many papers, such as Refs. [79, 80], have been praising the performance gain achieved with different variants of graph neural networks in contrast to other approaches, boosting the popularity of graph neural networks in this way. However, at the time this thesis is being written, no published paper could be found that deals with a detailed comparison of GNNs and DNNs, which are simpler in architecture and already well-established in HEP (e.g. DeepJet), on fair premises. A comparison on fair premises, between equivalent GNNs and DNNs, e.g., in terms of their number of hidden layers or the number of degrees of freedom, is indispensable as otherwise it is not possible to probe whether performance increases achieved with GNNs are due to their intrinsic properties or not. If not, it should be in principle possible to achieve the same performance with other machine learning techniques, like DNNs, by using more suitable hyperparameters, novel regularization methods, etc.
A comparison of equivalent GNNs and DNNs is accomplished in this chapter in the scope of the $t\bar{t}$+X event classification. At first, the comparability challenges between DNNs and GNNs are addressed and solutions are proposed in Section 7.1 before models with equivalent architectures are benchmarked against each other in Section 7.2. In Section 7.3, models with similar numbers of degrees of freedom are being exclusively focused on. Section 7.4 completes this chapter with an in-depth analysis of the best models according to the findings in the preceding sections.

7.1 Comparability Challenges and Solutions

Deep neural networks and graph neural networks are two different types of neural networks that are specialized in operating on vectors or graph structures (cf. Chapter 3), respectively. Being variants of neural networks, the same loss function, activation functions and optimizer can be employed for the training of both DNNs and GNNs. However, this does not hold for other aspects of these two types of neural networks such as the hyperparameter selection or the data-related information provided for training due to the intrinsic properties of DNNs and GNNs and therewith leading to comparability issues. Parameter sharing is, for instance, an intrinsic property of GNNs (cf. Section 3.2.3). This cannot be applied to DNNs or rather its usage would be quite against the natural information processing of DNNs, leading to the necessity of DNNs for higher numbers of trainable parameters for performing the same task as GNNs. A bigger obstacle poses the fact that DNNs only deal with vectors

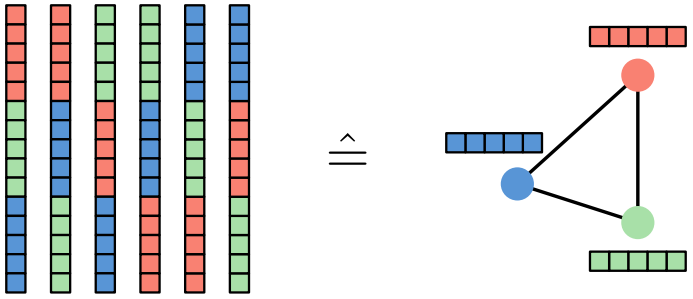


Figure 7.1: **Consequence of the lack of permutation invariance in DNN inputs.** Due to the lack of permutation invariance in DNN inputs, DNNs have to be trained on all $n!$ permutations of the n final state objects in an event. For the case of three final state objects with five associated features each, the form of all six permutations on which the DNN need to be trained is illustrated on the left. The corresponding graph representation, which would serve as input for GNNs and do not possess any intrinsic order, is depicted on the right.

as input, i.e., the matrices used as input in GNNs have to be "flattened", yet leading to the further disparity that no differentiation between vertex or edge attributes is possible anymore as well as to a larger DNN input layer. Also the lack of permutation invariance in the inputs of DNNs hinders an accurate comparison with GNNs. Since the final states objects of $\mathrm{t\bar{t}}$+X events do not possess any intrinsic order, i.e., no specific number can be assigned to them, the DNNs either have to be trained on all permutations of the final state objects (cf. Fig. 7.1), as realized in Ref. [77] for finding additional b jets in $\mathrm{t\bar{t}}$ events, or, alternatively, the category flags, originally used in the training of GNNs on a voluntary basis (cf. Section 4.3), must be determined. With this information, it is then possible to assign the attributes, captured in graphs, to the same position in the input vector for every event.

The disadvantage of the first solution would be a more time-consuming training whereas the latter approach provides an artificial order of the final state objects to the DNN training and is moreover incompatible with node-level prediction. However, since the category flags serve as features in the GNN trainings anyway, the latter approach is selected for this comparison. Another intrinsic property of DNNs is their fixed input size. In particular for the $\mathrm{t\bar{t}}$+X event classification task, this is a difficulty since the number of final state objects varies in each event due to additional unknown jets coming from higher-order effects or misreconstruction. In order to circumvent this limitation, the input size n_{in} is fixed to the maximum number of objects $N_{\text{obj}}^{\text{max}}$ that appears in the data, which is 17, times the number of features in the feature set. Thus, the input size of the DNNs for the feature set `extended` (cf. Fig. 4.5) would be $17 \cdot 13 = 221$ and adding edge weights of a complete graph to the input vector would even lead to an input size of $17 \cdot 13 + 17 \cdot 16 = 493$. In cases with smaller numbers of final state objects in an event than $N_{\text{obj}}^{\text{max}}$, zero padding is introduced, meaning that the empty entries in the input vector are filled with zeros. Figure 7.2 showcases the composition of an exemplary input vector for the training of DNNs. The increased input size of DNNs comes with the unfortunate side effect that either the same number of hidden layers and hidden nodes as of the GNN or the same number of trainable parameters of the compared models can be assured. However, both can be crucial aspects for the performance of models and therewith for defining whether two models are considered alike. If the number of hidden layers differs, then one model can potentially extract more abstract concepts of the data, as stated in Section 3.1, and might profit from these information when performing the classification. At the same time, a higher number

of trainable parameters gives a model a wider scope and more freedom to adapt on the task at hand. On the other hand, this increases the risk of overfitting [41]. Therefore, the comparison is split in two parts in the following, where only models with the same number of hidden layers and hidden nodes in these layers are compared in Section 7.2 and only models with similar numbers of trainable parameters are compared in Section 7.3.

In order to isolate the comparison from other possible model intrinsic properties, only fully-connected feed-forward neural networks without any further modifications, such as skip connections, are utilized and a rather basic GNN type shall be deployed. In addition, all regularization methods are deliberately dropped for the sake of keeping the contemplated models as simple as possible.
The GCN from Ref. [54] and presented in Section 3.2.3 seems to be predestined for this task since it only uses simple neural networks as update functions ϕ and a sum as aggregation function ρ. However, according to pretests performed with this GNN variant, it turned out to be rather not suitable for the $t\bar{t}$+X event classification task. Instead, a related GNN variant, named GraphConv in PyG and which was first introduced in Ref. [81], is applied. GraphConv equally utilizes neural networks and a sum for ϕ and ρ, respectively, in the most basic variant. The difference lies in their different treatment of aggregated information from the neighborhood and the embedding of the vertex to be updated. This is achieved by using two separate neural networks for updating the particular information before a summation over the output of both neural networks is performed in the end. Consequently, Equation 3.14 becomes

$$\mathbf{x}_u^{(l)} = f\left(\mathbf{W}_1^{(l)}\mathbf{x}_u^{(l-1)} + \mathbf{b}_1^{(l)} + \sum_{v\in\mathcal{N}(u)} \mathbf{W}_2^{(l)}\mathbf{x}_v^{(l-1)} + \mathbf{b}_2^{(l)}\right) \tag{7.1}$$

for GraphConv. For a detailed explanation of all variables, see Chapter 3.

7.2 Comparison of Models with an Equivalent Architecture

In this section, only models with the exact same number of hidden layers n_{HL}, with the exact same number of hidden nodes n_{hidden} in the particular hidden layers and that are trained with equivalent feature sets are defined as equivalent or comparable. For the sake of simplicity, feature sets that are seen as equivalent in this thesis possess the same names for DNNs and GNNs. Though, it must be kept in mind that, as opposed to GNNs, due to the nature of DNNs, no differentiation between non-relational and relational information, i.e., vertex and edge attributes, can be made, and the input vector is zero padded. The name of the feature sets matches the following naming scheme

$$\texttt{base\_feature\_set}(\text{relational\_information})\,.$$

The base feature set corresponds to the vertex attributes that were used in the previous chapters, e.g., `default` or `extended`. By optionally adding relational information to the input (for DNNs in the input vector and for GNNs in the form of edge weights), variations of the base feature set are created. For M_{inv} as additional information, the feature set is named `default`$[M_{\text{inv}}]$ or `extended`$[M_{\text{inv}}]$, for example. If all variations of the base feature set shall be addressed, then the parentheses are omitted and the superscript $*$ is added to the name of the base feature set.

As shown in Table 7.1, models with one or two hidden layer(s), denoted as $\text{DNN}_{\text{1HL}}/\text{GNN}_{\text{1HL}}$ and $\text{DNN}_{\text{2HL}}/\text{GNN}_{\text{2HL}}$, respectively, are examined in this comparison. The hidden layers

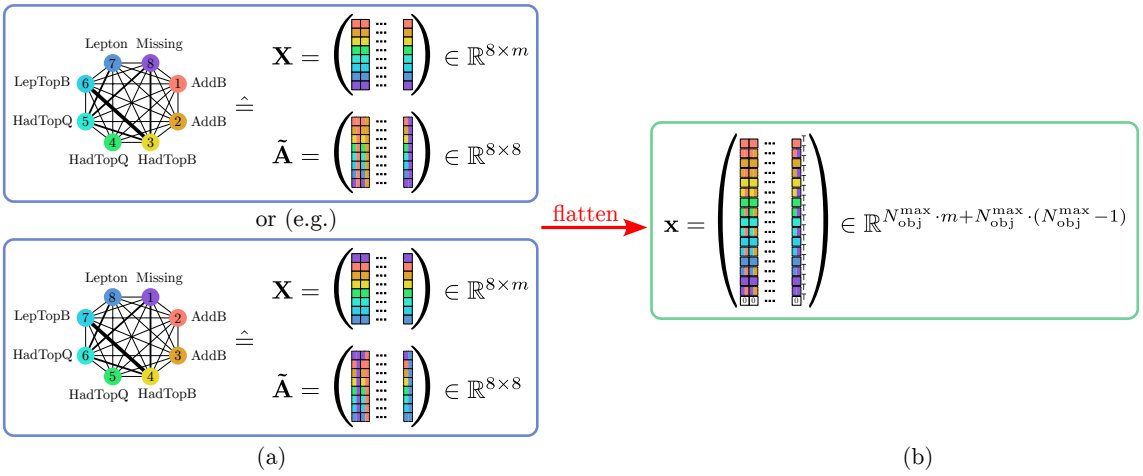


Figure 7.2: **Exemplary generation of the input vector for DNNs for an event with eight final state objects.** In (a), an exemplary graph representation of an event with eight final states and the corresponding matrices $\mathbf{X}$ and $\tilde{\mathbf{A}}$, describing the m attributes per vertex and the edge weights, respectively, are depicted. The rectangles in $\tilde{\mathbf{A}}$ are bicolored for visualizing the start vertex (left color) and the end vertex of the incident edge (right color). Since the graph is undirected, $\tilde{\mathbf{A}}$ is symmetric. As a graph does not possess an intrinsic order, permutations of the rows and columns of $\mathbf{X}$ and $\tilde{\mathbf{A}}$ are also valid matrices for describing the depicted graph mathematically. This does not hold for the positions of the vertex and edge features in the input vector $\mathbf{x}$, shown in (b). The "flattening" of both depicted possible matrices results in the same input vector for training DNNs. Thereby, the vertex attributes and edge features belonging to the respective vertex are always concatenated directly in succession. Due to the symmetric nature of $\tilde{\mathbf{A}}$, each edge feature appears twice in the input vector and despite of having only eight final state objects in the exemplary event, the input vector of DNNs is as big as for an event with 17 final state objects since that is the maximum number of final state objects in the data set. Thus, zeros have to be incorporated into the input vector.

Table 7.1: **Selected hyperparameters for GNNs and DNNs.** The variations of the base feature sets for both DNNs and GNNs include M_{inv}, ΔR or ΔR^{-1} as relational information. In the case of the GNNs, feature sets with zero, one or random values as relational information, i.e., as edge weights, are introduced, additionally. Except for n_{in}, which for DNNs depends on the feature set, the DNNs and GNNs share the same hyperparameters. Parameters that are not further specified in the table correspond to the default values in PyG.

hyperparameter	DNN	GNN
n_{in} (feature set)	102 (`default`) 221 (`extended`) 374 (`default`*) 493 (`extended`*)	13 (`extended`*)
N_{HL}	$\{1, 2\}$	
N_{hidden}	$\{13, 26, 39\}^{n_{\text{HL}} \in N_{\text{HL}}}$ (cf. Eq. 7.2)	
n_{out} (of readout)	1	
bias	true	
aggregation function ρ (Eq. 3.21)	sum	
global pooling method	mean	
maximum number of epochs	200	
EARLY-STOPPING	$\Delta\text{epochs} = 15$, $\Delta\text{loss} = 0.001$	
mini-batch size	200	
optimizer	ADAM ($\gamma = 0.01$)	
activation function (in hidden layers)	ReLU (Eq. 3.4)	
activation function (in output layer)	SIGMOID (Eq. 3.5)	
loss function	BINARY CROSS-ENTROPY (Eq. 3.6)	
number of repetitions	10	

consist of 13, 26 or 39 hidden notes. The number of hidden nodes n_{hidden} in two-layer models is

$$\begin{aligned} n_{\text{hidden}} \in N_{\text{hidden}} &= \{13, 26, 39\}^{n_{\text{HL}}=2} \\ &= \{13, 26, 39\} \times \{13, 26, 39\} \\ &= \{(13, 13), (13, 26), \dots\}\,. \end{aligned} \tag{7.2}$$

That is, all permutations of the given number of hidden nodes are deployed. The values selected as number of hidden nodes are arbitrarily chosen and correspond to multiples of the number of attributes per vertex used in `extended`, which is 13. The models are trained with different feature sets. Like in the previous studies in this thesis, the edges of the graph data are associated with physically motivated values such as M_{inv} or ΔR. Additionally, ΔR^{-1} as edge weight is tested because, from a physics point of view, it should be more reasonable to weight messages between final state objects stronger that are closer to each another. Furthermore, some GNN specific properties shall be probed in a more isolated way and therewith enhance the comparability between DNNs and GNNs even further. The approach for that is to train the GNNs with different graph connectivity schemes. (cf. Fig. 7.3). With the choice of setting every edge weight to zero, the message passing in GNNs is entirely prohibited. Using random edge weights, on the contrary, shall resemble the first hidden layer in fully-connected DNNs where a linear transformation with all input features on the basis of (in the first epoch of the training) randomly initialized weights is performed. Since graph representations were selected because they can describe the relations between objects in $\mathrm{t\bar{t}} + \mathrm{b\bar{b}}$, $\mathrm{t\bar{t}H(b\bar{b})}$ and $\mathrm{t\bar{t}Z(b\bar{b})}$ events in a more natural way, it is expected that the performance of GNNs trained with graphs with physically motivated edge weights are superior.
Since the aforementioned DNN weights are adjusted throughout the training, this serves as motivation for training the edge weights in the graph data via an additional neural network (addNN). These GNNs are denoted tGNNs in the following whereas the simple GNNs operating (as before) without addNN are denoted sGNNs. The interplay between updating edge weights and vertex attributes takes place in the way described in Section 3.2.4 and depicted in Fig. 3.8b. That is, the input of addNN ($\widehat{=}\ \phi_{\text{e}}^{(l)}$ in Equation 3.16) for updating the weight of an edge with the incident vertices u and v at iteration l comprises the current edge weight itself $\mathbf{x}_{uv}^{(l-1)}$ and the current embeddings of the incident vertices $\mathbf{x}_{u}^{(l-1)}$, $\mathbf{x}_{v}^{(l-1)}$. The subsequent hidden layer has as many hidden nodes as the corresponding update function $\phi_{\text{v}}^{(l)}$ and the output layer consists of one output node. Thereby, ReLU serves as activation function for both hidden and output nodes. In total, 120 GNNs are taken into account in this section (cf. Table G.1).

It is expected that DNNs will not perform as poorly as GNNs (cf. Section 4.3) when trained with `default`. The reason is that the additional information preserved in the category flags in `extended` is already implicitly stored in the (fixed) position of the features in the input vector due to the lack of permutation invariance in the inputs of DNNs. For example, in the feature set `extended` the first thirteen elements of the input vector always contain the vertex attributes of the harder additional b jet. Thus, for DNNs, category flags capture redundant information and by omitting them the number of input nodes and therewith the number of DOF of the DNNs under scrutiny could be strongly decreased. However, since DNNs and GNNs then do not receive the same explicit information for training, it is arguably whether the feature sets `default`* provided to DNNs are really similar to the feature sets `extended`* provided to GNNs. To avoid this discrepancy, these models are not regarded as comparable. On the other hand, in order to prove the assumption that the category flags do not contain highly relevant information for DNNs, DNNs trained with `default`* are still considered, leading to eight different feature sets for training DNNs.

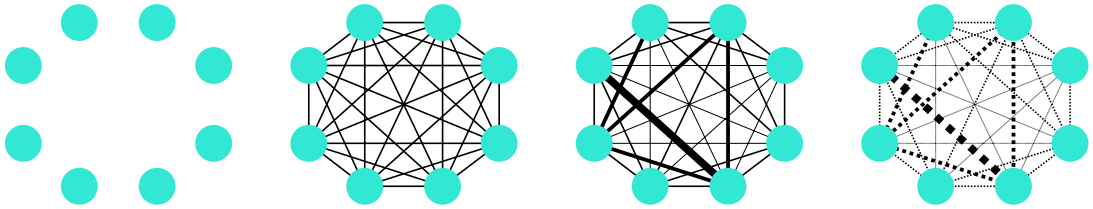


Figure 7.3: **Graph connectivity schemes.** Since the edge weights are all set to zero in (a), the corresponding graph does not possess any edges at all, meaning message passing is inaccessible for the GNN training. On the contrary, the graphs depicted in (b), (c) and (d) are complete. While the edge weights in (b) are all chosen to be one, in (c) and (d) the messages exchanged between vertices are differently weighted. The edge weights tested in this comparison are both physically motivated values (M_{inv}, ΔR, ΔR^{-1}) and random values. The dotted edges in (d) shall indicate that the same edge weights as in (c) are used at first but they are adapted in the course of the training. Thus, the aforementioned edge weights only serve as initialization in this case.

In total, this comparison comprises 48 (96 including the DNNs trained with `default`*) different DNNs (cf. Table G.1).
Together with the GNNs, there are in total 168 (216) models or rather 1680 (2160) trained neural networks for benchmarking against each other since each training is repeated ten times in order to be able to draw statistically more sound conclusions.

7.2.1 Model Performance and Training Stability

In Fig. 7.4, the model performance of **GNNs consisting of one hidden layer** in dependence of the feature set used for training and the selected N_{hidden} is depicted. At first glance, it becomes apparent that data points of the same color, indicating the same feature set being used for training, form groups. That means using different edge weights, i.e., different graph connectivity schemes, has a greater impact on the model performance than the variation of the number of hidden nodes and therewith the number of DOF of the particular models. Equally to the results collected in Section 4.4, selecting M_{inv} as edge weight is superior to ΔR. However, it is against the initial physics intuition that utilizing ΔR^{-1} as edge weight is not more beneficial than ΔR. On the other hand, it is striking that using physically non-meaningful, random, edge weights or fixing every edge weight to one lead to similar results. However, completely prohibiting message passing (edge weight $= 0$) results in the worst-performing models by far, with mean ROC-AUC values at around 0.655. The corresponding models can therefore be regarded as random estimators and thus, it becomes evident that the message passing is indeed a crucial aspect to data processing in GNNs.
When the training of edge weights is allowed, the performance of GNNs trained with edge weight ΔR, ΔR^{-1} or random values can be further increased in all except for one case in comparison to equivalent GNNs. However, as hinted by the comparatively large standard deviation and the occurrence of outliers, training such GNNs seems to be not stable. That even leads to the consequence that the average performance improvement gained through training the edges weighted with ΔR^{-1} as edge weight is $(0.0 \pm 1.2)\,\%$, i.e., nonexistent.
Only the best-performing GNNs in the ranking in Fig. 7.4 appear to be completely unaffected by this approach, i.e., no further enhancement of their performance is achievable. A possible explanation for this observation could be that the invariant mass is in fact already the best possible choice for edge weights for the binary $\mathrm{t\bar{t}}$+X event classification. This is

consistent with the observation that when GNNs are trained with graphs with random edge weights, i.e., on graphs with the least meaningful edge weight initialization, the largest improvements in model performance are obtained.

All these findings basically also hold for **GNNs consisting of two hidden layers** and therefore, it is renounced to investigate deeper GNNs in this section. This is in particular visible in Fig. 7.5a, where the mean performance improvement of one-layer and two-layer GNNs in dependence of the deployed edge weight is shown. In both cases, the baseline is $\mathrm{GNN}_{\mathrm{1HL}}$ and $\mathrm{GNN}_{\mathrm{2HL}}$, respectively, trained with graphs without edges. Not only the ranking of the different edge weights between $\mathrm{GNN}_{\mathrm{1HL}}$ and $\mathrm{GNN}_{\mathrm{2HL}}$ in terms of gained model performance matches but also the actual performance improvement through the particular edge weights is similar between these two hidden layer configurations. The tendency whether the model performance is improved when tGNNs are used is, according to Fig. 7.5b, except for the edge weight ΔR, also consistent between models with one and two hidden layer(s). The big discrepancy in ΔR is probably caused by the not stable training of tGNNs. In fact, five out of nine tGNNs trained with edges weighted with ΔR show an outstanding high spread compared to sGNNs trained with the same feature set but still having a standard deviation that is just low enough for passing the outlier criterion (b). It is however interesting that for GNNs with two hidden layers, unlike for GNNs with one hidden layer, outliers occur in trainings with fixed edge weights as well. This could be traced back to the increased number of trainable parameters to be adjusted and a therewith more elaborate training.

Like for GNNs, **DNNs** perform better on the given classification task when relational information is included in the feature set. The choice of relational information, which is originally used as edge weights for training GNNs, dominates the model performance of DNNs as well. Neither the particular number of hidden nodes nor the choice of the base feature set can outperform this effect, as shown in Fig. 7.6. When only looking at the models trained with feature sets with M_{inv} as relational information, it appears that using `extended` as base feature set is beneficial for the model performance of DNNs. However, this is not as clear anymore when considering the other feature set variants. In fact, the worst-performing DNN in the ranking is also trained with the base feature set `extended`. That is, in some cases, using `extended` is even counterproductive, presumably, since it leads to a more complex DNN model, due to the associated increased input size, but at the same time no additional information is provided to the training as the category flags only contain redundant information. Thus, the assumption that the category flag information does not need to be provided to the DNN explicitly tends to be correct. In general, these observations can also be transferred to $\mathrm{DNN}_{\mathrm{2HL}}$. It should be noted that the strict separation of the models trained with the same relational information in groups in the mean performance ranking (regardless of the base feature set being used) has been softened a little in the case of two-layer DNNs.

Interestingly, between **DNNs and GNNs**, the hierarchy of the relational information in terms of the corresponding resulting model performance is exactly the same. So, the best expansion of the base feature set is in descending order M_{inv}, ΔR and ΔR^{-1}. However, for DNNs, completely omitting relational information does not result in random estimators as the worst-performing models still possess a mean ROC-AUC of 0.7622 ± 0.0029 and 0.7856 ± 0.0078 for DNNs with one and two hidden layer(s), respectively. In Fig. 7.7, the mean model performance improvement induced by the different added relational information and separated by n_{HL} and the base feature set used for training is depicted. In all cases,

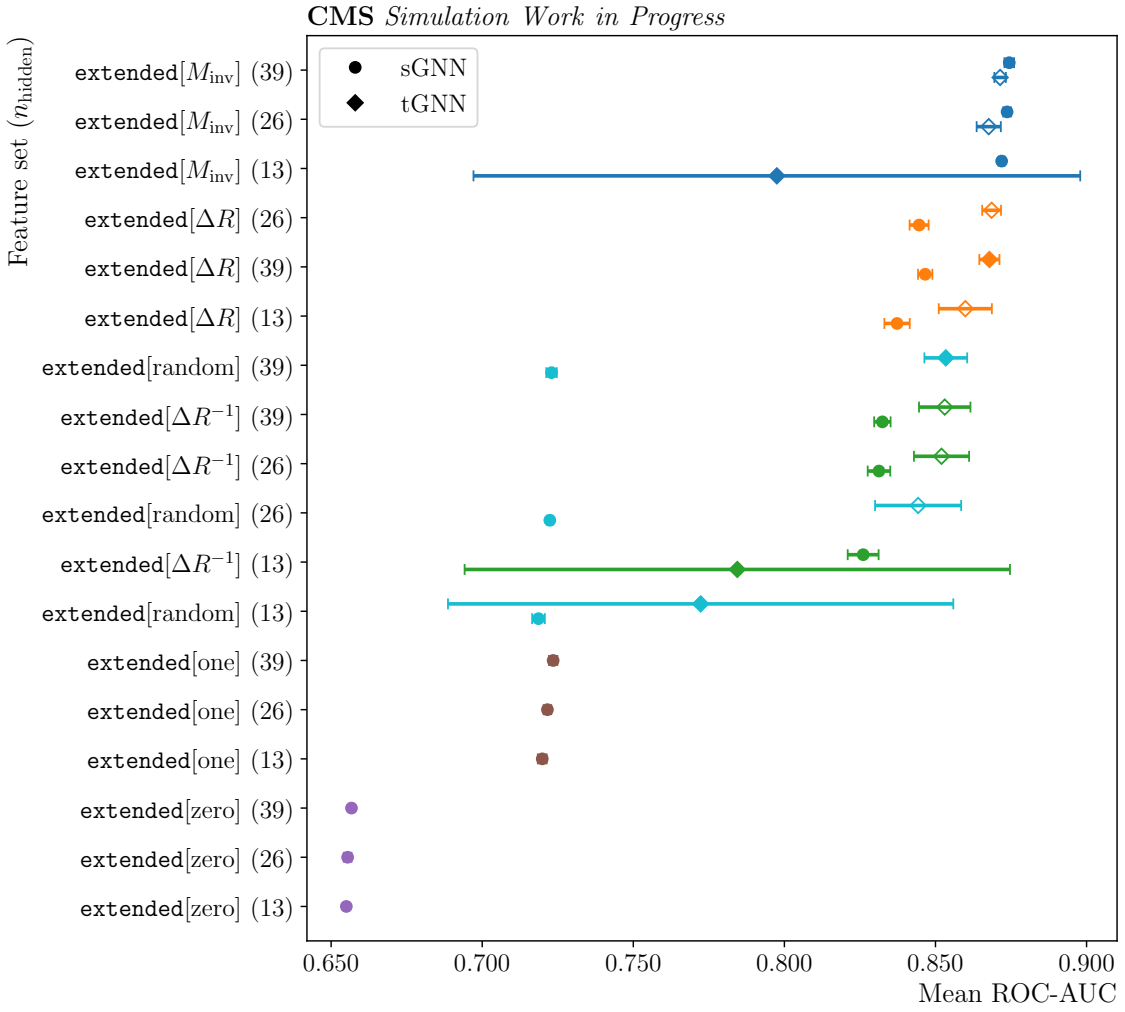
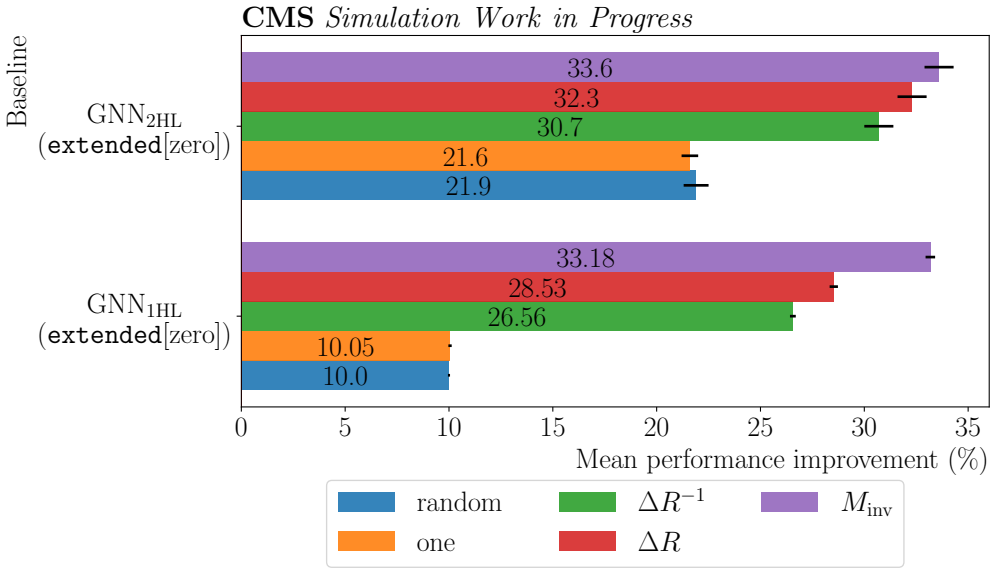
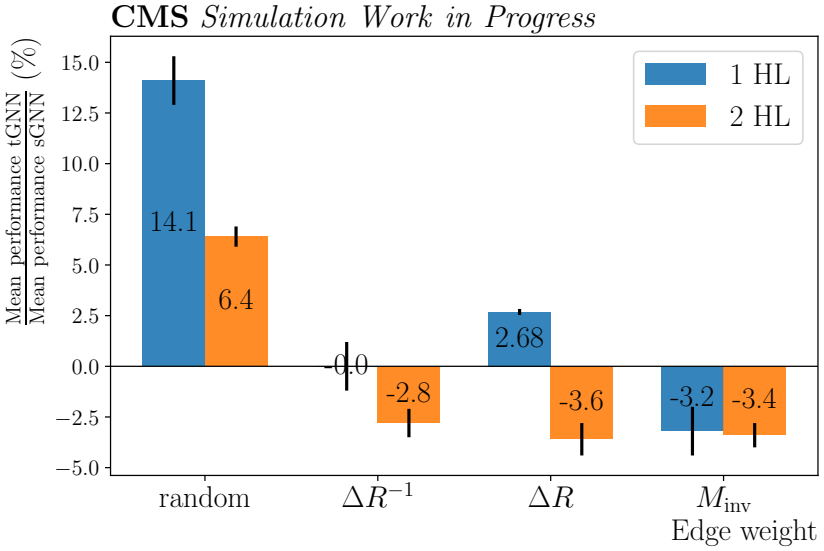


Figure 7.4: **Mean performance ranking of GNN_{1HL}.** The colors correspond to the feature set used for training. The data points are color grouped. Data points with unfilled markers indicate that at least one model of ten repetitions of the particular training has a performance fulfilling the outlier criteria defined in Section 4.3 and the identified outlier(s) is/are therefore discarded. All data points possess an error bar representing the performance spread (square root of the unbiased sample variance), which is calculated on the basis of the ROC-AUC values of all not discarded models (at most ten models, one for each of the ten realized repetitions of a training). In order to enhance the visibility of the error bars of sGNNs and tGNNs, which share the same label on the y-axis, the corresponding data points are shifted in a way that they are stacked in descending order of the corresponding mean ROC-AUC value from top to bottom. If an error bar is not visible, then this is due to the fact that the spread is too narrow to be displayed. Especially, the spread of the performance of tGNNs is exceptionally higher than the spread of other models.



(a)



(b)

Figure 7.5: **Mean performance improvement of GNNs.** In (a), the mean performance improvement of one-layer and two-layer GNNs with respect to `extended`[zero] is depicted separately for each edge weight variant. The ranking of the edge weights with the highest positive impact on the model performance is the exact same for one-layer and two-layer GNNs. In (b) the mean performance gain with respect to the deployment of edge weight training is shown. If randomly initialized edge weights are additionally adjusted through the training, another huge performance improvement for one-layer and two-layer GNNs can be achieved. This does not hold for the remaining edge weight variants, where the model performance is often even decreased. In both plots, the error bars correspond to the uncertainty calculated on the basis of the unbiased sample variance.

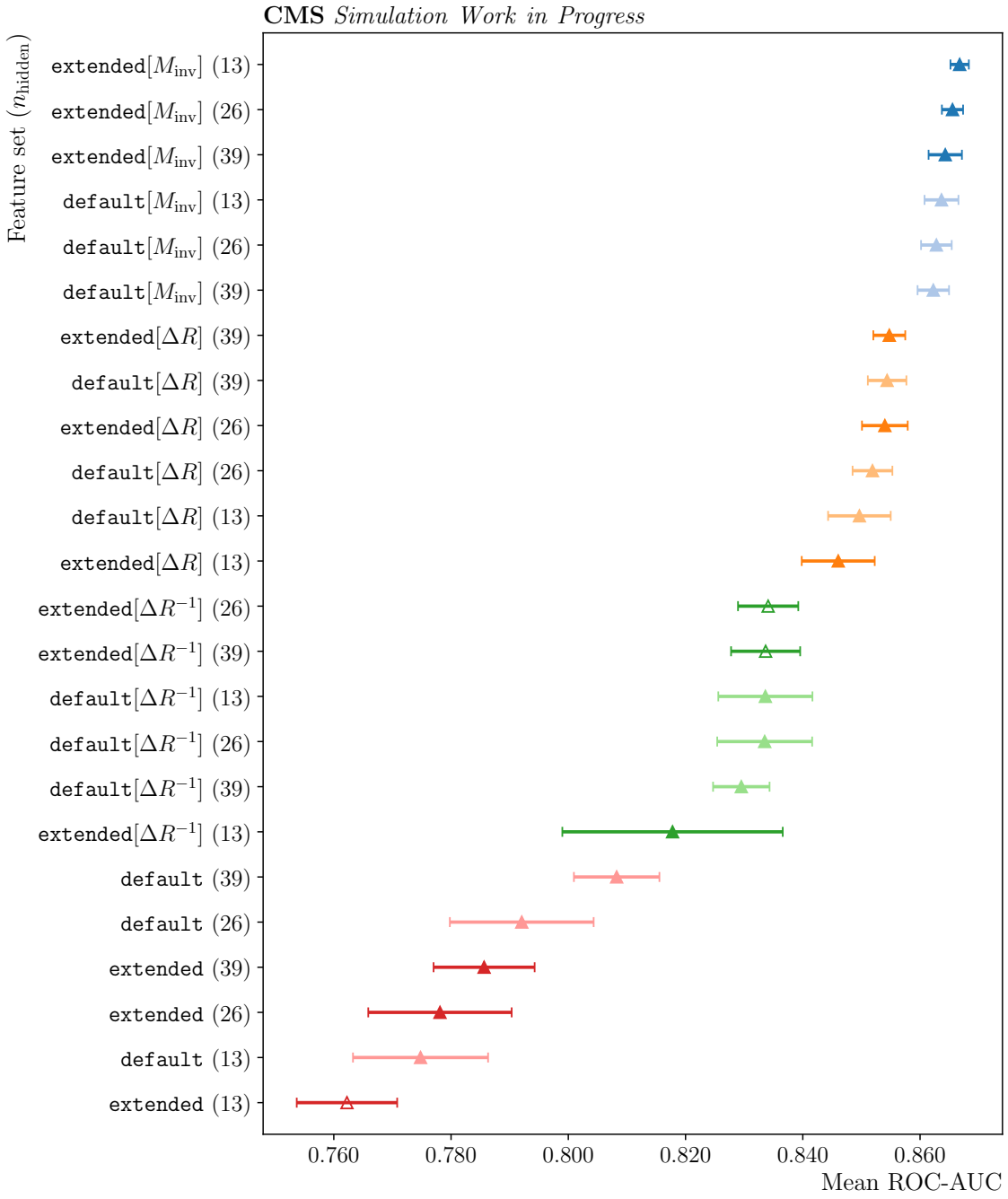


Figure 7.6: **Mean performance ranking of DNN_{1HL}.** The colors correspond to the particular feature set used for training. It is striking that the data points of the DNNs trained with the same relational information form non-overlapping groups with respect to the mean ROC-AUC value. The edge weights ΔR^{-1}, ΔR and M_{inv} as expansions of the input vector lead, in that order, to better performing DNNs. Except for feature sets with M_{inv} as relational information, there is no clear performance hierarchy between models that are only distinguishable through the base feature set. Data points with unfilled markers indicate that at least one model of ten repetitions of the particular training has a performance fulfilling the outlier criteria defined in Section 4.3 and the identified outlier(s) is/are therefore discarded. All data points possess an error bar representing the performance spread (square root of the unbiased sample variance), which is calculated on the basis of the ROC-AUC values of all not discarded models (at most ten models, one for each of the ten realized repetitions of a training).

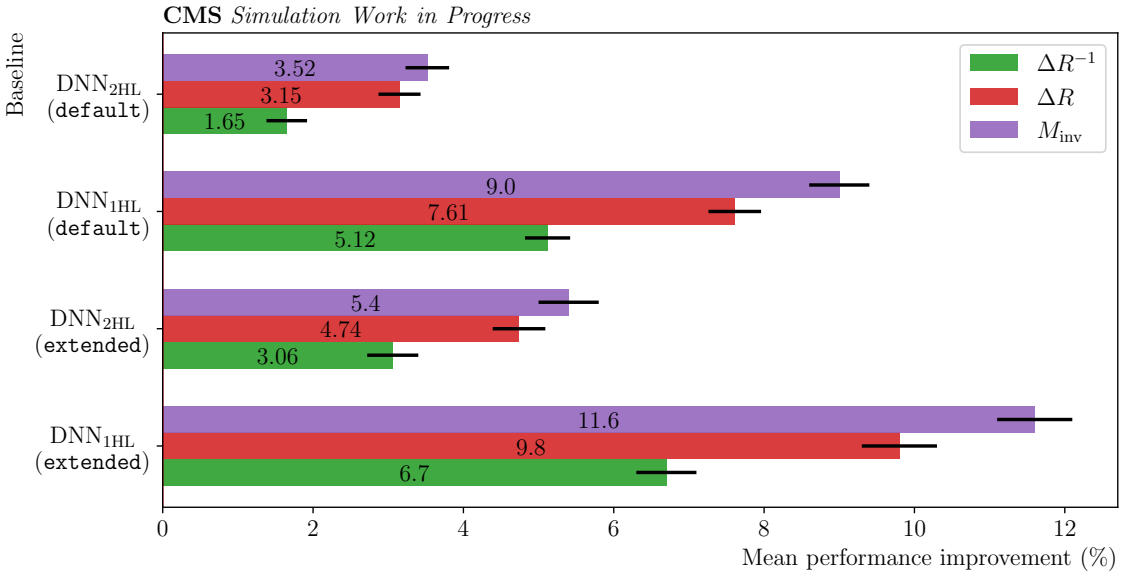


Figure 7.7: **Mean performance improvement of DNNs separated by the combinations of base feature set and relational information deployed.** The base on which the mean performance improvements are calculated is a one-layer or two-layer DNN trained on the base feature set specified on the y-axis. Accordingly, four different groups of DNNs can be distinguished and in all cases the expansion of the base feature set by M_{inv} provides the largest, by ΔR the second largest and by ΔR^{-1} the third largest performance improvement within each group. The error bars correspond to the uncertainty calculated on the basis of the unbiased sample variance.

one-layer and two-layer DNNs trained without relational information serve as baseline. The performance gain with added relational information seems to be not as large as for GNNs. That appears to indicate that the embedding of relational information in the graph structure or rather the meaningfully weighted message passing is of outstanding importance to the learning process. Yet, it must not be neglected that the baseline performance of GNNs trained without relational information is much lower.
How close the model performance between equivalent DNNs and GNNs really is, can be seen in Fig. 7.8 for one-layer models or in Fig. G.1 for two-layer models. The difference in model performance between equivalent DNNs and sGNNs is on average only $(0.02 \pm 0.10)\,\%$ and $(0.75 \pm 0.04)\,\%$ for models with one and two hidden layer(s), respectively. However, it becomes evident that in general the performance spread of DNNs is larger than the performance spread of GNNs (at least if the edge weights of the GNNs are not additionally trained) and therewith their underlying training is less stable. The cause for the larger standard deviation could be that the corresponding models have a much higher number of DOF than sGNNs, such as the best-performing GNNs. Especially the best-performing GNNs have a such narrow spread that their error bar is not even visible in the plot.
Though sGNNs having the feature set `extended`$[M_{\text{inv}}]$ as input outperform the equivalent DNNs, this does not apply to the other variations of `extended` in all except for two cases. However, DNNs trained with `default` or `extended`, thus without any relational information at all, are superior to sGNNs trained with graphs without any physically meaningful graph structure, i.e., graphs with random, one or zero as edge weights. Hence, this demonstrates that GNNs are not by all means a more powerful neural network type than DNNs. Instead, only GNNs trained with a thoroughly selected graph structure are of increased value. On the other hand, the results also reveal that two of three tGNNs using `extended`[random] as input outperform the mentioned DNNs. Thus, perhaps in cases in which no prior knowledge of the underlying relationships is available, it might be indicated to deploy such tGNNs instead of DNNs.
It is worth mentioning that two-layer sGNNs trained with either `extended`$[\Delta R]$ or `extended`$[\Delta R^{-1}]$ also outperform the equivalent DNNs and moreover, not all DNNs trained without relational information are superior to GNNs trained with graphs without any physically meaningful graph structure in this case. However, these minor differences in the results of models with one hidden layer and models with two hidden layers do not have an impact on the conclusions.

The best-performing models in this section are summarized in Table 7.2. It should be taken into account that increasing the number of hidden layers, i.e., deploying deeper neural networks, is beneficial for both DNNs and GNNs. Yet, using the greatest number of hidden nodes that is allowed in this comparison does not necessarily result in the best-performing models.

7.2.2 Convergence Speed and Degrees of Freedom

The model performance and the training stability are not the only aspects to be considered when comparing neural networks. For instance, a simple model, e.g., in terms of necessary DOF (N_{TP}) for accomplishing a given task, is preferred over a more complicated one. But in order to account for the fact that due to the zero padding in the input vector for DNNs only a small fraction of the actual number of trainable parameters is relevant in theory, an effective number of trainable parameters $N_{\text{TP}}^{\text{eff}}$ is additionally calculated. For that the average number of final state objects in an event in the data set, which is 10.80, is used as basis for calculating $N_{\text{TP}}^{\text{eff}}$ instead of $N_{\text{obj}}^{\text{max}}$.

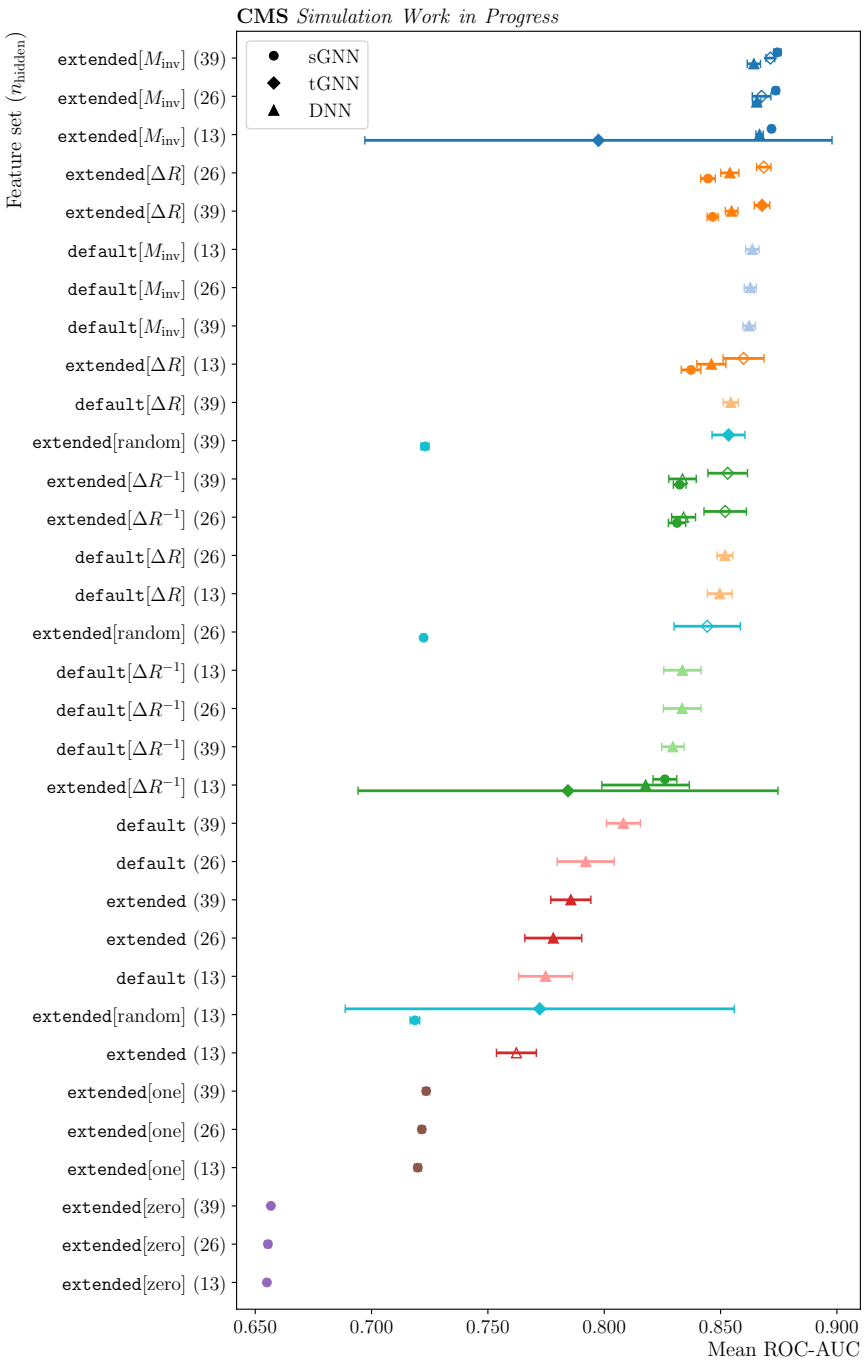


Figure 7.8: **Combined mean performance of GNN_{1HL} and DNN_{1HL}.** The colors correspond to the particular feature set used for training. Although now different NN types are displayed together, the color grouping of the data points is still clearly recognizable. Apart from that, the small performance difference of equivalent DNNs, sGNNs as well as tGNNs can be observed. Data points with unfilled markers indicate that at least one model of ten repetitions of the particular training has a performance fulfilling the outlier criteria defined in Section 4.3 and the identified outlier(s) is/are therefore discarded. All data points possess an error bar representing the performance spread (square root of the unbiased sample variance), which is calculated on the basis of the ROC-AUC values of all not discarded models (at most ten models, one for each of the ten realized repetitions of a training). Whenever models share the same label on the y-axis, the corresponding data points are shifted in a way that they are stacked in descending order of the corresponding mean ROC-AUC value from top to bottom to enhance the visibility of the error bars. If an error bar is not visible, then the spread is too narrow to be displayed.

Table 7.2: **Architecture and performance of the best-performing models.** The variables n_{hidden} and N_{TP} correspond to the number of nodes in the hidden layer(s) and the number of trainable parameters in the particular model, respectively. The effective number of trainable parameters is denoted $N_{\text{TP}}^{\text{eff}}$ and is calculated on the basis of the average number of final state objects in an event (10.80) (cf. Section 7.2.2).

NN type	n_{hidden}	N_{TP}	$N_{\text{TP}}^{\text{eff}}$	$\langle$ROC-AUC$\rangle$	identifier
sGNN	(39)	1093	—	0.87441 ± 0.00051	$\text{GNN}_{1\text{HL}}^{*}$
	$(26, 26)$	2107	—	0.87860 ± 0.00035	$\text{GNN}_{2\text{HL}}^{*}$
DNN	(13)	6436	2405	0.86676 ± 0.00050	$\text{DNN}_{1\text{HL}}^{*}$
	$(13, 26)$	6813	2782	0.87198 ± 0.00044	$\text{DNN}_{2\text{HL}}^{*}$

Also the training and inference duration of models could be an indispensable factor when it comes to deciding which type of neural networks should be deployed for a task at hand. Since the models were trained on hardware that was not exclusively used for processing the trainings, the measured training duration of the models is not guaranteed to be unbiased and it is therefore only of reduced expressive power. Instead, the best validation epoch or rather its inverse, defined as convergence speed in this thesis, is chosen to be the alternative "time" measure of the training of neural networks. Though the convergence speed does not necessarily correlate with the actual training duration, empirically, this is the case and is therefore seen as an adequate replacement for the training duration. In Table 7.3, the average difference in the mean convergence speed $\langle\Delta\text{speed}\rangle$, the average difference in the mean number of trainable parameters $\langle\Delta N_{\text{TP}}\rangle$ as well as the average difference in the mean number of effective trainable parameters $\langle\Delta N_{\text{TP}}^{\text{eff}}\rangle$ of one-layer and two-layer sGNNs and tGNNs with respect to equivalent DNNs are depicted. As an example, the average difference in the mean convergence speed is obtained by:

(1) Calculating the mean convergence speed by averaging over all not discarded repetitions of each training.

(2) Calculating the difference in the mean convergence speed between equivalent GNNs and DNNs.

(3) Calculating the average of these differences for each combination of neural network type (sGNN, tGNN and DNN) and number of hidden layers separately.

In the same manner, $\langle\Delta N_{\text{TP}}\rangle$ and $\langle\Delta N_{\text{TP}}^{\text{eff}}\rangle$ are obtained. The only difference is that the (effective) number of trainable parameters and its mean over all not discarded repetitions is obviously exactly the same.

Only $\text{sGNN}_{1\text{HL}}$ are converging slower on average than equivalent one-layer DNNs, which is in particular surprising when considering that between these models there is the biggest discrepancy in terms of the number of DOF among all compared models. The naive assumption would be that convergence speed correlates with the number of trainable parameters that need to be adjusted during training. Apparently, this correlation is not realized in this case. In general, it can be summarized that GNNs converge faster than equivalent DNNs. Moreover, they also appear to be more effective in the usage of the DOF than DNNs (even when only the effective number of trainable parameters is taken into account).

Table 7.3: **Average difference in the mean convergence speed and number of DOF between equivalent models.** The values are calculated with respect to model B in the table, as given in the second column. Positive $\langle\Delta\text{speed}\rangle$ values correspond to a greater convergence speed of model A in comparison to model B.

model A	model B	$\langle\Delta\text{speed}\rangle$ (%)	$\langle\Delta N_{\text{TP}}\rangle$ (%)	$\langle\Delta N_{\text{TP}}^{\text{eff}}\rangle$ (%)
sGNN_{1HL}	DNN_{1HL}	-20.1 ± 3.3	-94.33	-84.84
tGNN_{1HL}	DNN_{1HL}	26 ± 13	-88.47	-69.13
sGNN_{2HL}	DNN_{2HL}	4.1 ± 2.5	-84.49	-62.10
tGNN_{2HL}	DNN_{2HL}	31 ± 4	-68.36	-22.66

7.3 Comparison of Models with a Similar Number of Degrees of Freedom

Since the DNNs considered in Section 7.2 consists of up to approximately 95 % more DOF than GNNs, the following further questions arise naturally:

(1) Can DNNs outperform $\text{GNN}_{\text{2HL}}^{*}$ if only the number of degrees of freedom is tuned? (Section 7.3.1)

(2) How well do DNNs perform if their number of degrees of freedom is restricted to $N_{\text{TP}}(\text{GNN}_{\text{2HL}}^{*}) = 2107$? (Section 7.3.2)

(3) How well do GNNs perform if their number of degrees of freedom is expanded to $N_{\text{TP}}(\text{DNN}_{\text{2HL}}^{*}) = 6813$ or $N_{\text{TP}}^{\text{eff}}(\text{DNN}_{\text{2HL}}^{*}) = 2782$? (Section 7.3.3)

In this comparison, only `extended`$[M_{\text{inv}}]$ is given as input to the DNNs and GNNs because this provided the best-performing neural networks in the previous comparison. In correspondence to the observations made in Section 7.2.1, rather deeper neural networks but not necessarily networks with a bigger n_{hidden} shall be trained for answering question (1). Accordingly, it is decided to allow DNNs to consist of three hidden layers and $N_{\text{hidden}} = \{6, 13, 26\}^3$. For elaborating on questions (2) and (3), models with one up to four hidden layer(s) are trained with $N_{\text{hidden}} = \{5, 6, \ldots, 50\}^{n_{\text{HL}}}$ and $N_{\text{hidden}} = \{2, 3, \ldots, 12\}^{n_{\text{HL}}}$ being scanned. Yet, in order to keep the numbers of models that need to be trained for question (2) and question (3) feasible, for each hidden layer only the model(s) with N_{TP} closest to the particular target value are further considered. In total, 71 different models, i.e., 710 networks, are evaluated. The remaining hyperparameters are not varied in this comparison and follow the same selection as shown in Table 7.1.

7.3.1 DNNs with a Tuned Number of Degrees of Freedom

In contrast to the results in Section 7.2.1, having more hidden layers appears not to be beneficial for the performance of DNNs. In fact, only one out of 27 DNNs with a three-layer architecture is ranked higher than $\text{DNN}_{\text{2HL}}^{*}$ in terms of model performance. But it also turns out that the mean ROC-AUC of the best three-layer DNN is 0 % higher than its of $\text{DNN}_{\text{2HL}}^{*}$ and therewith an improvement with three-layer architectures is nonexistent. At the same time, it is however noticeable in Fig. G.2 that in principle all DNNs with three hidden layers show a larger spread than $\text{DNN}_{\text{2HL}}^{*}$. At first sight, it seems reasonable to assume that the lack of improvement despite applying a deeper DNN is promoted by the lack of regularization methods with its stabilizing effect on the training. On the other hand, when looking at the progression of the loss during training and validation averaged over ten repetitions (cf. Fig. G.3), no big spread in the loss curve progression is observable and

hence this assumption is disproven. Consequently, the only remaining conclusion appears to be that DNNs indeed cannot outperform GNNs when simply their depths and their numbers of trainable parameters are tuned.

7.3.2 DNNs with a Restricted Number of Degrees of Freedom

Since DOF are taken away from the model for properly learning the task at hand, its generalization abilities are more stipulated. As the DNNs do not seem to overfit yet in the previous analysis, i.e., they are already generalizing as well as possible, it is expected that the restriction complicates the training. Furthermore, the performance of DNNs with N_{TP} (or $N_{\text{TP}}^{\text{eff}}$) being restricted to $N_{\text{TP}}(\text{GNN}_{\text{2HL}}^{*}) = 2107$ should decrease. These DNNs are named restrDNNs in the following. The results, shown in Fig. G.4, supports this expectation. Indeed, around 70 % of the trained restrDNNs contain outliers and especially the lower ranked models still show a striking high spread, although outliers were already removed. It is only surprising that nevertheless, on average, the mean ROC-AUC value of restrDNNs is only (1.540 ± 0.027) % worse than its of $\text{DNN}_{\text{2HL}}^{*}$ although the N_{TP} and $N_{\text{TP}}^{\text{eff}}$ of restrDNNs are on average respectively around 56 % and 57 % lower than its of $\text{DNN}_{\text{2HL}}^{*}$. On the other hand, this is consistent with the observations made in Section 7.3.1, where an increase of the numbre of DOF equally did not affect the model performance much, i.e., the originally provided DOF were already sufficient for accomplishing the given task. Moreover, like already seen in Section 7.2.2, it is once more shown that the convergence speed is independent of N_{TP} since the convergence speed of restrDNNs, where $N_{\text{TP}} \approx N_{\text{TP}}(\text{GNN}_{\text{2HL}}^{*})$, is (24 ± 6) % slower than the convergence speed of $\text{GNN}_{\text{2HL}}^{*}$. Due to that it is not recommended to choose restrDNNs over GNNs. Because of the less stable training of restrDNNs in general, it is also not reasonable to intentionally restrict the number of DOF of DNNs despite of the comparable performance.

7.3.3 GNNs with an Expanded Number of Degrees of Freedom

In correspondence to the explanation and observations in the previous section, it is expected that GNNs with a higher number of N_{TP} should perform slightly better but not necessarily converge slower than $\text{GNN}_{\text{2HL}}^{*}$. Both is indeed the case. In total, five models with N_{TP} expanded to roughly $N_{\text{TP}}^{\text{eff}}(\text{DNN}_{\text{2HL}}^{*})$ or $N_{\text{TP}}(\text{DNN}_{\text{2HL}}^{*})$ are ranked higher than $\text{GNN}_{\text{2HL}}^{*}$ (cf. Fig. G.5), whereby the performance is increased by a maximum of (0.14 ± 0.06) %. These models also converge (24 ± 10) % faster than $\text{GNN}_{\text{2HL}}^{*}$ on average. The convergence speed of the GNNs with $N_{\text{TP}} \approx N_{\text{TP}}(\text{DNN}_{\text{2HL}}^{*})$ is even (64 ± 9) % faster than the convergence speed of $\text{DNN}_{\text{2HL}}^{*}$. This proves once again that GNNs indeed converge faster than DNNs. In general, the effect that can be observed when expanding the number of DOF of GNNs is greater than when tuning (cf. Section 7.3.1) or restricting (cf. Section 7.3.2) the number of DOF of DNNs, yet negligible in terms of performance gain, especially considering how much more complex (regarding depth and number of DOF) these expanded GNNs are. Thus, it is reasonable to choose expanded GNNs over simpler GNNs with less depth when the convergence speed is a crucial aspect for the particular task at hand, for example.

7.4 In-Depth Analysis of the Best-Performing Models

The best-performing models of both comparisons, summarized in Table 7.4, shall be further analyzed in this section. At first, the mean prediction scores of all best models for each event in the test set are compared, so that it becomes evident whether the models perform similarly on classifying the same events. For that, two-dimensional histograms with 50 equidistant bins in each dimension are generated. If the mean prediction scores of the two models under scrutiny are the exact same, then the histogram equals the angle bisector.

Table 7.4: **Architecture and performance of the best-performing models of both comparisons.**

NN type	n_{hidden}	N_{TP}	$N_{\text{TP}}^{\text{eff}}$	$\langle$ROC-AUC$\rangle$	identifier
sGNN	(39)	1093	—	0.87441 ± 0.00051	$\text{GNN}_{\text{1HL}}^{*}$
	(26, 26)	2107	—	0.87860 ± 0.00035	$\text{GNN}_{\text{2HL}}^{*}$
	(25, 40, 50)	6816	—	0.87980 ± 0.00034	expGNN*
DNN	(13)	6436	2405	0.86676 ± 0.00050	$\text{DNN}_{\text{1HL}}^{*}$
	(13, 26)	6813	2782	0.87198 ± 0.00044	$\text{DNN}_{\text{2HL}}^{*}$
	(13, 13, 6)	6695	2664	0.87201 ± 0.00085	$\text{DNN}_{\text{3HL}}^{*}$
	(4, 8, 7, 3)	2107	867	0.86892 ± 0.00090	restrDNN*
	(11, 3, 3, 7)	5518	2107	0.86941 ± 0.00062	$\text{restrDNN}_{\text{eff}}^{*}$

In some plots, a plateau appears for mean prediction scores near zero. That is in particular prevailing for any comparisons of restrDNN* with the remaining best models (cf. Fig. 7.9). The plateau indicates that restrDNN* fails to notice subtle differences in non-$\text{t}\bar{\text{t}} + \text{b}\bar{\text{b}}$ samples, probably caused by the restricted number of DOF, and therefore only manages to predict the same score for the particular events, unlike the other models in the comparisons. The plateau is larger when restrDNN* is compared to GNNs than to DNNs. This is a universal feature since it can be observed in general that mean prediction scores between same neural network types are more alike than between different neural network types, i.e., that the corresponding histogram is distributed narrower along the bisecting line in that cases (cf. Fig. G.6). The particular correlation coefficient ρ also reflects this phenomenon. When different neural network types are compared usually $\rho < 0.980$ holds. Moreover, GNNs appear to be more certain in their prediction for non-$\text{t}\bar{\text{t}} + \text{b}\bar{\text{b}}$ samples since their mean prediction scores are closer to zero than DNNs for such events. Thus, despite of the quite similar model performance of GNNs and DNNs in terms of ROC-AUC values, it is recommended to rather use GNNs when the correct classification of non-$\text{t}\bar{\text{t}} + \text{b}\bar{\text{b}}$ samples is more crucial than the correct identification of $\text{t}\bar{\text{t}} + \text{b}\bar{\text{b}}$ events.

In order to investigate further whether the strengths of the best-performing models lie in different areas, radial plots are introduced. In radial plots, it is possible to compare several aspects of a model at once. The aspects of interest in Fig. 7.10 are the mean convergence speed, the mean loss on the test set, which serves as measure of the generalization ability of the models, the performance of the models expressed by the mean ROC-AUC and the complexity of the model (N_{TP}, $N_{\text{TP}}^{\text{eff}}$).
While the ranking of the models regarding the generalization ability corresponds to the performance of DNNs and GNNs as well, this does not hold for the other aspects. For both neural network types, the biggest discrepancy between the different models are the convergence speed and the number of (effective) trainable parameters. The overall best GNN and DNN appear to be $\text{GNN}_{\text{2HL}}^{*}$ and restrDNN*, respectively, as they perform well in all aspects compared to the other models.

Lastly, a first- and second-order Taylor coefficient analysis (TCA, cf. Section 3.3.2) is executed for shedding light on whether all best models profit in the same way from the information provided during the training. The advantage of the TCA is that it can be applied to both DNNs and GNNs. However, since a Taylor coefficient is calculated for each input feature, for DNNs, there are 493 values for the feature set `extended`$[M_{\text{inv}}]$, which

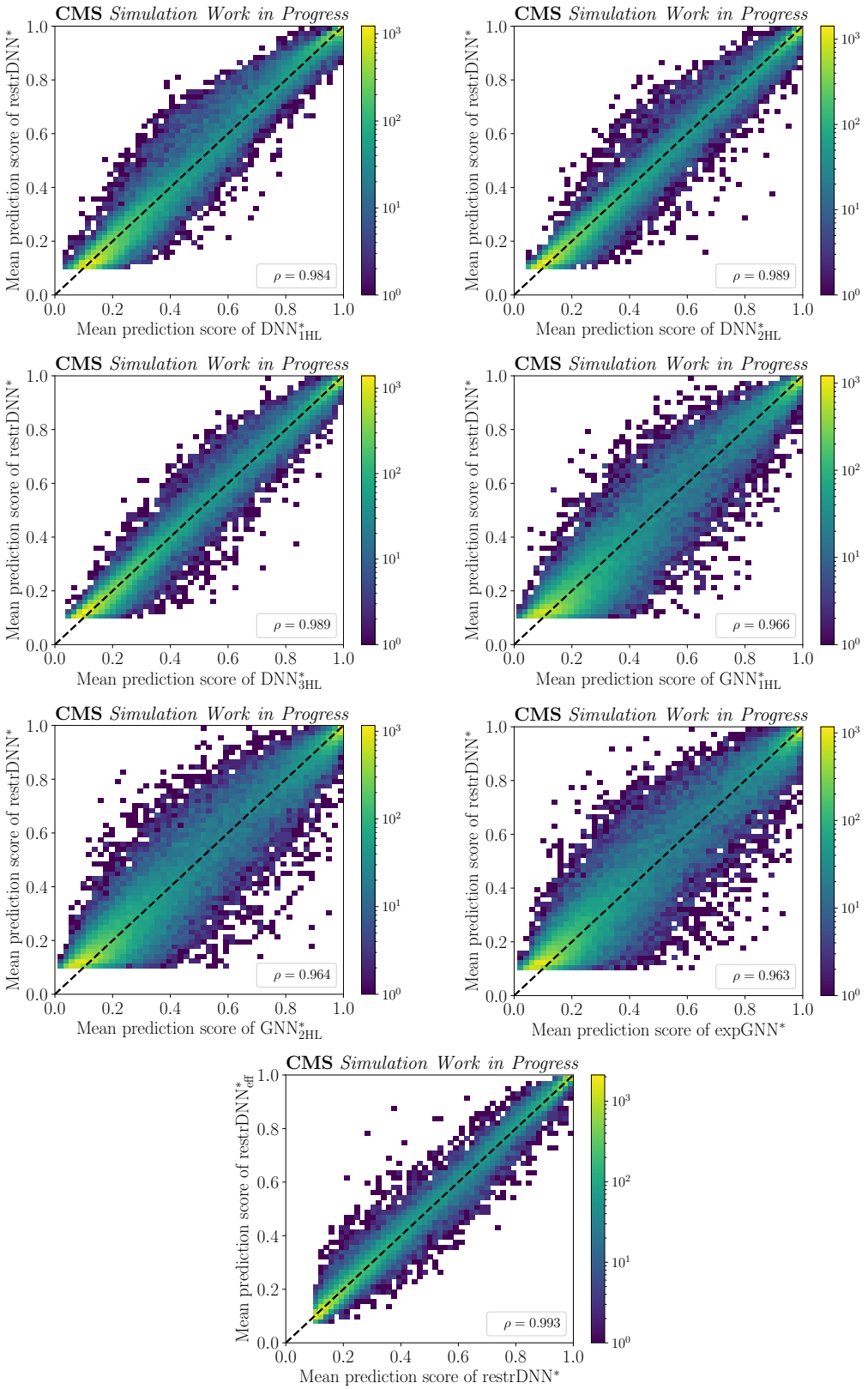


Figure 7.9: **Comparison of the mean prediction scores of restrDNN* to the prediction of any other best-performing model from Table 7.4.** Fifty equidistant bins are chosen along each axis. The closer the particular correlation coefficient is to one, the more alike the mean prediction scores of the compared models are. Visually, this translates to a narrower distribution of the histogram along the angle bisector (dashed line). In every plot shown, a plateau arises for mean prediction scores near zero. This is more pronounced when the response of restrDNN* are compared to GNNs instead of other DNNs.

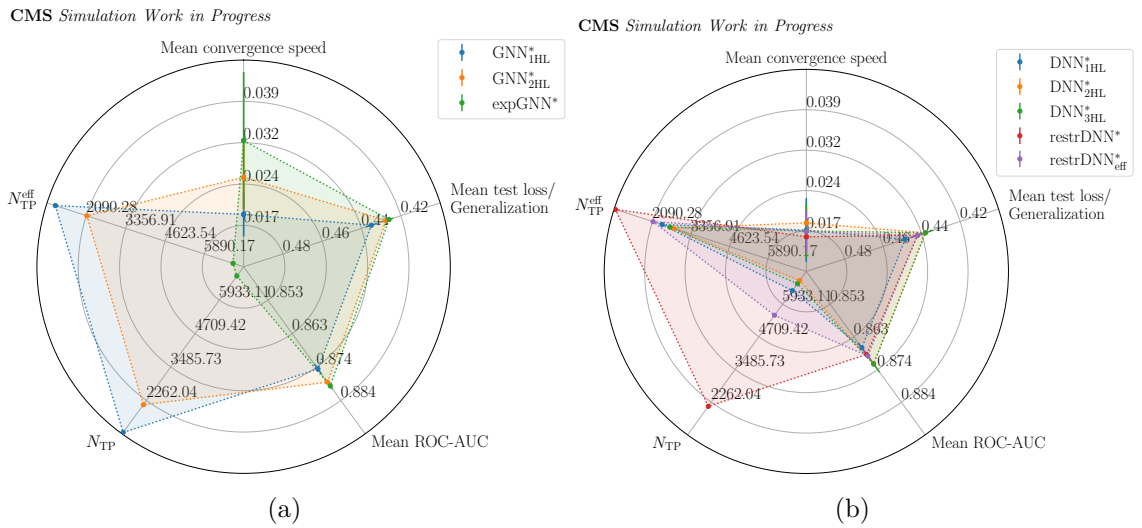


Figure 7.10: **Comparison of various aspects of neural networks.** The mean convergence speed, the generalization ability, the model performance and the model complexity of the best-performing GNNs (a) and the best-performing DNNs (b) are compared at once. The larger the area covered by a model in this diagram, the better its all-round ability is. For that the axes corresponding to the generalization and the DOF of a model are inverted. The error bars represent the spread (square root of the unbiased sample variance) of the particular quantities and are calculated on the basis of the corresponding quantities obtained in the ten realized repetitions of a training.

is not handy to deal with and complicates the comparison to the Taylor coefficients of GNNs, for which only 13 Taylor coefficients are calculated. As solution, it is proposed to calculate "global" Taylor coefficients, i.e., to average over all Taylor coefficients belonging to the same observable. Taking the invariant mass as an example, 272 individual invariant mass features M_{inv}^{ij}, one for each permutation of a final state objects i and j exist. The corresponding Taylor coefficients $t^{(s)}_{M_{\text{inv}}^{ij}}$, calculated for each event s in the test set of size S, are reduced to one global Taylor coefficient

$$t^{(s)}_{M_{\text{inv}}} = \frac{1}{N_{\text{obj}}^{\max} \cdot (N_{\text{obj}}^{\max} - 1)} \sum_{i=1}^{N_{\text{obj}}^{\max}} \sum_{j \neq i} \left| t^{(s)}_{M_{\text{inv}}^{ij}} \right| . \tag{7.3}$$

Finally, only 14 Taylor coefficients remain. In addition, however, only the Taylor coefficients of non-padded features of an event are considered in the averaging process, because the results are expected to be diluted otherwise. In order to extract the general decision basis of the model, as before (see introduction of Chapter 6), the arithmetic mean over all events in the test set is computed subsequently.

As indicated by the $\langle \Delta r \rangle$ values, the first-order TCA ranking of the features within GNNs and within DNNs in Figs. 7.11 and 7.12, respectively, are similar. All best GNN models have in common that the most important category flag is the AddB flag, the most important kinematic information is p_{T} and the least important feature is ϕ, which is all reasonable from a physics point of view. As observed in Section 6.1 already, each category flag appears to be of larger importance for the GNN response, especially for $\text{GNN}^*_{\text{1HL}}$, than an actual physics observable.
For DNNs, the most important category flag is the AddB flag and the least important

feature is ϕ, like for GNNs. Yet, the most important feature for DNNs is indeed M_{inv} by far, i.e., the information that is encoded in the graph structure for GNNs and has proven to be the most impactful aspect to the GNNs' performance (cf. Section 7.2.1). Apparently, this relational information is of such outstanding importance that even DNNs learn to focus on this information as well. Accordingly, this can be seen as the cause for the similar performances of DNNs and GNNs when trained on the same feature set. The biggest discrepancy between the first-order TCA rankings of DNNs and GNNs is the positioning of category flags. For DNNs, they are all situated in the lower part of the ranking. This is however not peculiar and is predicted in Section 7.2 already where it is assumed that category flags only contain redundant information for DNNs due to the fixed positions of the features in the input vector. Hence, with this observation, the expectation can be regarded as fully confirmed.
The second-order Taylor coefficients, which capture relations between input features, reflects as well that $\text{GNN}^*_{\text{1HL}}$ only learns to focus on correlations of all category flags with basically similar importance in general. In contrast to that, both $\text{GNN}^*_{\text{2HL}}$ and expGNN^* manage to learn that it is already sufficient to only look at correlations involving the AddB flag (cf. Fig. G.7). This improvement cannot be observed in the second-order Taylor coefficient plots for DNNs (cf. Fig. G.8). For all inspected DNNs basically no correlations between features except for M_{inv} with itself are relevant for their classification. Interestingly, this observation can only be made when feature sets including relational information are provided to the DNN training. Therefore, it is assumed that this effect is not directly caused by the usage of global feature and rather traces back to a possible incomparability of Taylor coefficents of relational information and vertex attributes. Relational information is normalized while vertex attributes are standardized (cf. Section 4.1). Thus, their distributions differ in the mean and standard deviation, which could affect the comparability of the Taylor coefficients since their calculation is based on gradients. Furthermore, also the fact that any relational information occurs twice in the input vector for DNNs, as $M^{ij}_{\text{inv}} = M^{ji}_{\text{inv}}$ (cf. Fig. 7.2), possibly influence the results, especially since second-order Taylor coefficients extract the relation between the input features. Further studies are required to verify these assumptions.

An aspect that is neglected so far is the data preprocessing effort required for generating the data set for training DNNs and GNNs, since no 100 % objective measure can be applied. From a subjective point of view, it is easier to generate the data set for training GNNs on $\text{t}\bar{\text{t}}$+X event classification than for DNNs since for the latter, e.g., each final state object in an event needs to be permuted so that their order matches with every other event in the data set. For this, the category flags have to be predetermined (if no generator-level information is used). Otherwise, the DNNs need to be trained on all permutations of the final state objects in each event, which is time-consuming as well. On the contrary, for GNNs, only the edges between the final state objects need to be constructed, which is a straightforward process since only complete graphs are used so far for the $\text{t}\bar{\text{t}}$+X event classification. Of course, for other tasks with, for instance, a greater number of objects (track reconstruction, for example) more time needs to be invested in evaluating the best graph structure and for actually processing the graph construction. For $\text{t}\bar{\text{t}}$+X event classification, it can be concluded that it is beneficial to choose GNNs over DNNs, especially due to their strengths in terms of convergence speed, their more effective usage of the provided DOF, their more stable training and the lower data preprocessing effort.

In summary, DNNs and GNNs show comparable performance on the binary $\text{t}\bar{\text{t}} + \text{b}\bar{\text{b}}$ vs. $\text{t}\bar{\text{t}}\text{H}(\text{b}\bar{\text{b}})/\text{t}\bar{\text{t}}\text{Z}(\text{b}\bar{\text{b}})$ event classification task, presumably because they have learnt to focus

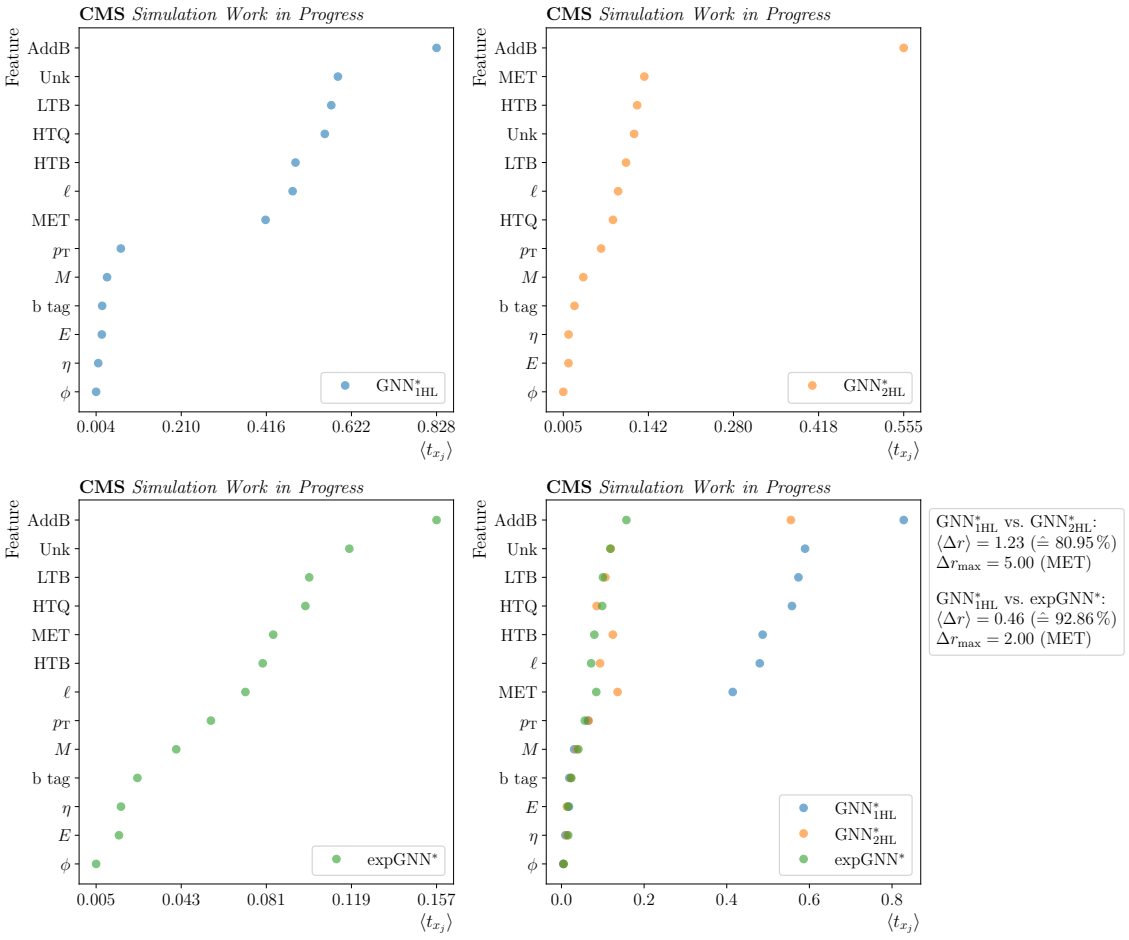


Figure 7.11: **First-order Taylor coefficient ranking of the best-performing GNNs.** It is striking that all three GNNs have a similar focus on the same features. The most important features are the category flags, followed by p_{T}, which is the most important kinematic jet feature. The least important feature for training is ϕ, which is reasonable from a physics point of view. At the bottom, on the right, the ranking for all three GNNs are combined. The percentage in the parentheses corresponds to the conformity $1 - \langle \Delta r \rangle / \Delta r^*_{\mathrm{max}}$ of the compared rankings. The variable Δr_{max} denotes the highest number of rank difference between the compared rankings and should not be confused with $\Delta r^*_{\mathrm{max}}$, which denotes the maximum mean number of absolute rank differences between the compared rankings. The objects to which this applies are specified in the subsequent parentheses.

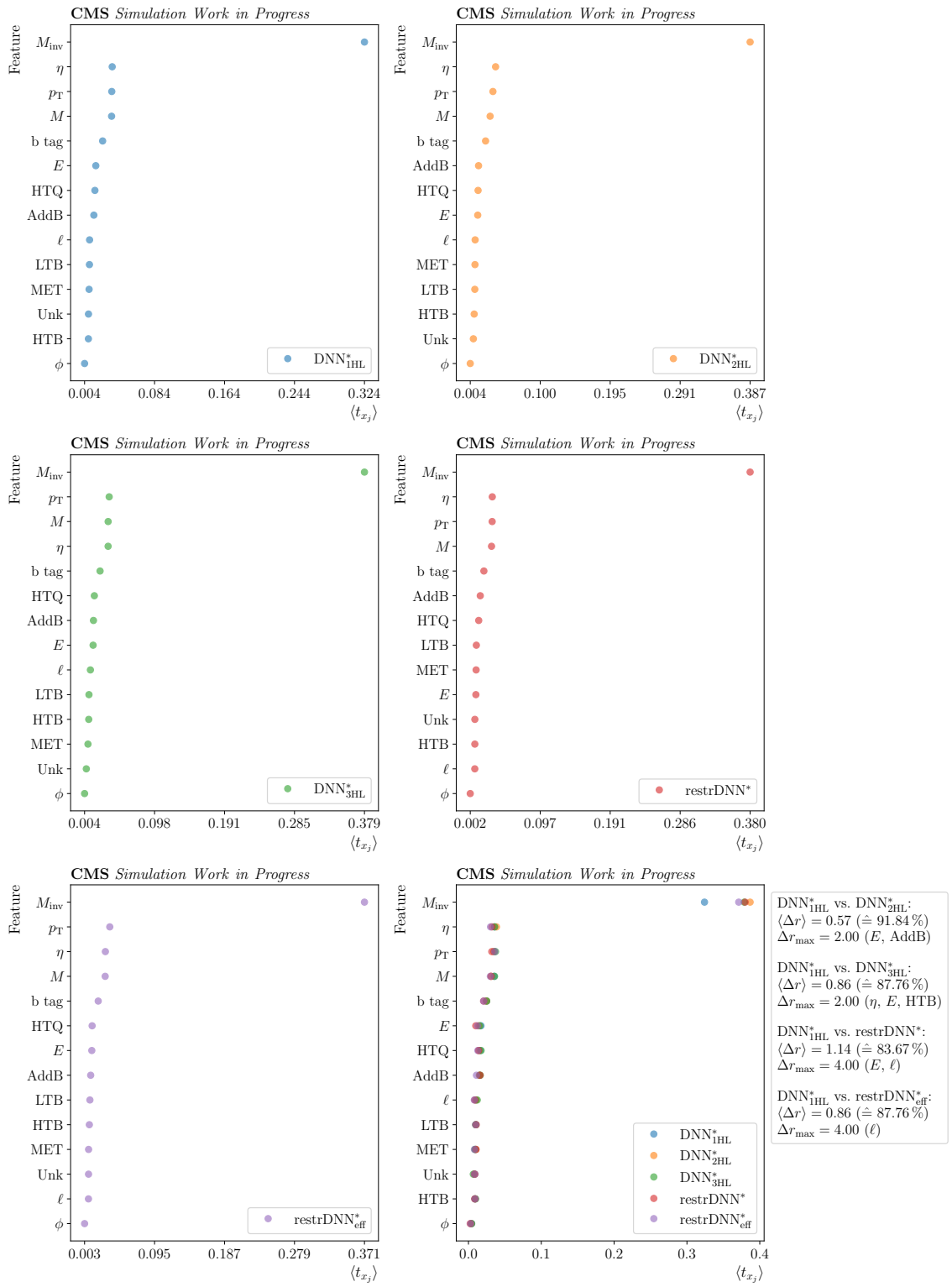


Figure 7.12: **First-order Taylor coefficient ranking of the best-performing DNNs.** For all DNNs under scrutiny, M_{inv} is the by far most important feature. Like for GNNs, ϕ is the least important feature for accomplishing the classification. However, unlike GNNs, the category flags are all ranked in the lower half of these plots. At the bottom, on the right, the ranking for all five DNNs are combined. The percentage in the parentheses corresponds to the conformity $1 - \langle \Delta r \rangle / \Delta r^*_{\text{max}}$ of the compared rankings. The variable Δr_{max} denotes the highest number of rank difference between the compared rankings and should not be confused with Δr^*_{max}. The objects to which this applies are specified in the subsequent parentheses. Instead of dealing with all 493 Taylor coefficients, 14 global Taylor coefficients are calculated and ranked. Moreover, only Taylor coefficients corresponding to non-padded values in an event are considered.

on the same features, as the Taylor coefficient analysis reveals. The biggest discrepancy between these two neural network types clearly lies in the convergence speed, the number of DOF necessary for performing the task and the data preprocessing effort. Regarding all three aspects, the best-performing DNNs are weaker than the best-performing GNNs. Moreover, GNNs tend to be more confident in classifying non-$t\bar{t} + b\bar{b}$ samples than DNNs and their training is more stable.

8 Summary and Outlook

The physics focus of this thesis is on classifying $t\bar{t} + b\bar{b}$, $t\bar{t}H(b\bar{b})$ and $t\bar{t}Z(b\bar{b})$ processes in proton-proton collisions at the LHC. Since these $t\bar{t}$+X processes can be more naturally represented by graphs rather than, for instance, by vectors, graph neural networks (GNNs), designed for processing exactly this data type, appear to be the best choice from a theoretical point of view. Accordingly, the main goal of this thesis is to verify whether this in fact applies to this special scope. For this purpose, the general feasibility of GNNs for this task is probed first. Subsequently, explainable AI (xAI) methods are applied to the models for unraveling the underlying decision basis of these black-box models and therewith estimating the reliability of the GNN responses. Lastly, comparisons between GNNs and deep neural networks (DNNs), which are already well-established in high energy physics (HEP), are performed under fair conditions to finally assess whether these novel GNNs are indeed more preferable for multivariate $t\bar{t}$+X event classification.

In the feasibility study, Gated Graph Sequence Neural Networks (GGSNN) are deployed for classifying $t\bar{t}$+X events in a binary ($t\bar{t} + b\bar{b}$ vs. $t\bar{t}H(b\bar{b})/t\bar{t}Z(b\bar{b})$) as well as a multiclass manner. It becomes evident that, unlike for GNN-based additional b jet assignment, it is insufficient to only provide a few kinematic observables and the b tag as vertex attributes to the GNN training. The corresponding models are only at most 45 % better than random estimators. The most impactful adjustment is to extend the vertex attributes by category flags that specify for each final state object its corresponding category. For example, for jets, information about the particle with which they were produced in association is provided in this manner. In accordance with physics expectations, it is found that indicating which final state objects are additional b jets (AddB flag) is very beneficial for the $t\bar{t}$+X event classification. In fact, when a GNN-based classifier (NLP-GGSNN) with a true positive rate (TPR) of about 71 % is utilized for identifying these jets and a joint b tag/p_T approach is used for assigning the remaining categories, the original model performance improves by about 6 % in the binary case. Yet, with a flawless preclassifier even an improvement by roughly another 16 % to a mean ROC-AUC of 0.8793 would be in theory realizable. With respect to a random estimator that equals to a performance gain of about 76 %. Hence, it has been successfully shown that GNNs are – even without hyperparameter optimization – indeed a very suitable technique for $t\bar{t}$+X event classification.

To lay a solid foundation for estimating whether it is more worthwhile to concentrate on enhancing the preclassifier or rather the event classification model in future optimizations, the dependency of the event classification model on the goodness of the additional b jet assignment is modeled. Of two proposed modeling strategies, the AddB-X modeling strategy, which exploits several performance related key figures of the NLP-GGSNN, emerges as a valid approach for capturing the real dependency of the event classifier performance on the preclassifier. Therewith, it is demonstrated that increasing the TPR of NLP-GGSNN by just 0.17 % will presumably result in an about 2 % better performing event classifier. Thus, in future developments, it is recommended to target the optimization of the preclassifier first.

With the help of two xAI methods, the GNNExplainer (GNNX) and the Taylor coefficient analysis (TCA), the trustworthiness of the best-performing models is successfully probed. Both xAI methods identify, in consistence with previous observations and physics expectations, the AddB flag by far as the most impactful feature on the GNN response and p_{T} as the most valuable kinematic jet information to the GNN response in general. However, the first-order TCA and GNNX disagree in particular about the influence of the remaining category flags and the b tag value on the GNN responses. Yet, no clear physics statements can be made for this either. A further study reveals that the explanations provided by the first-order TCA are more plausible. Moreover, its computations are even significantly less time consuming.

Ultimately, two comparisons between GNNs (GraphConv) and DNNs with mutually exclusive focuses and a total of 287 models are conducted. For both types of neural networks, the type of the provided relational information has generally by far the greatest impact on the model performance compared to the numbers of hidden layers or the number of degrees of freedom (DOF) of the model, for instance. In particular, it becomes evident that the difference in performance of the best-performing equivalent GNNs and DNNs on the binary $\mathrm{t\bar{t}}$+X event classification task is less than 1 %. This could be traced back to the fact that, according to the TCA, both focus on similar features. It should however not go unmentioned that it is not realizable to overcome this performance gap between GNNs and DNNs by simply tuning the number of DOF and the depth of DNNs. Furthermore, deploying DNNs for $\mathrm{t\bar{t}}$+X event classification comes with a number of disadvantages. They show a diminished training stability, a slower convergence speed and require a higher data preprocessing effort. At the same time, they are also less effective in using the provided DOF and seem to be less certain regarding classifying non-$\mathrm{t\bar{t}} + \mathrm{b\bar{b}}$ events than GNNs.

In summary, it has been successfully verified to the full extent that GNNs are not only reliable and suitable for $\mathrm{t\bar{t}}$+X event classification. In fact, it is also more beneficial to deploy GNNs rather than DNNs for this very task despite their comparable peak performance.

Since the main focus of this thesis is not on mere performance optimization but rather on gaining a deeper comprehension on bare GNNs, their feasibility and reliability for multivariate $\mathrm{t\bar{t}}$+X event classification, neither regularization methods are deployed nor a hyperparameter optimization is conducted. Yet, it is expected that both will noticeably increase the performance of the event classification model. Especially, the usage of regularization methods is recommended for further stabilizing the training in future studies. Furthermore, it is presumably beneficial to develop a multi-task network that is simultaneously trained on both additional b jet assignment and $\mathrm{t\bar{t}}$+X event classification as this end-to-end model would come with the advantage of being easier to be retrained, optimized and distributed, unlike the current sequential approach.

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". In: *Physics Letters B* 716(1) (2012), pp. 1–29. DOI: 10.1016/j.physletb.2012.08.020.

[2] The CMS Collaboration. "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC". In: *Physics Letters B* 716(1) (2012), pp. 30–61. DOI: 10.1016/j.physletb.2012.08.021.

[3] The CMS collaboration. "Measurement of the cross section for $t\bar{t}$ production with additional jets and b jets in pp collisions at $\sqrt{s} = 13\,\text{TeV}$". In: *Journal of High Energy Physics (JHEP)* 2020(125) (2020). DOI: 10.1007/JHEP07(2020)125.

[4] The CMS collaboration. *Measurement of* $t\bar{t}H$ *production in the* $H \to b\bar{b}$ *decay channel in* $41.5\,\text{fb}^{-1}$ *of proton-proton collision data at* $\sqrt{s} = 13\,\text{TeV}$. Tech. rep. Geneva: CERN, 2019. URL: https://cds.cern.ch/record/2675023.

[5] The KATRIN Collaboration. "Direct neutrino-mass measurement with sub-electronvolt sensitivity". In: *Nature Physics* 18 (2022), pp. 160–166. DOI: 10.1038/s41567-021-01463-1.

[6] R. L. Workman et al. (The Particle Data Group). "Review of Particle Physics". In: *Progress of Theoretical and Experimental Physics (PTEP)* 2022, 083C01 (2022). DOI: 10.1093/ptep/ptac097.

[7] R. Wolf. *The Higgs Boson Discovery at the Large Hadron Collider*. Ed. by G. Höhler et al. Vol. 264. Springer Tracts in Modern Physics (STMP). Springer Cham, 2015. ISBN: 978-3-319-18512-5. DOI: 10.1007/978-3-319-18512-5.

[8] D. J. Griffiths. *Introduction to Elementary Particles*. Second, Revised Edition. Physics textbook. Weinheim: Wiley-VCH, 2008. ISBN: 978-3-527-40601-2.

[9] M. D. Schwartz. *Quantum Field Theory and the Standard Model*. New York: Cambridge University Press, 2014. ISBN: 978-1-107-03473-0. DOI: 10.1017/9781139540940.

[10] E. Noether. "Invariante Variationsprobleme". In: *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse* 1918 (1918), pp. 235–257. URL: http://eudml.org/doc/59024.

[11] S. L. Glashow. "Partial-symmetries of weak interactions". In: *Nuclear Physics* 22(4) (1961), pp. 579–588. DOI: 10.1016/0029-5582(61)90469-2.

[12] S. Weinberg. "A Model of Leptons". In: *Physical Review Letters* 19(21) (1967), pp. 1264–1266. DOI: 10.1103/PhysRevLett.19.1264.

[13] A. Salam. "Weak and electromagnetic interactions". In: *Conf. Proc. C* 680519 (1968), pp. 367–377. DOI: 10.1142/9789812795915_0034.

[14] The UA1 Collaboration. "Experimental observation of isolated large transverse energy electrons with associated missing energy at $\sqrt{s} = 540$ GeV". In: *Physics Letters B* 122B(1) (1983), pp. 103–116. DOI: 10.1016/0370-2693(83)91177-2.

[15] The UA2 Collaboration. "Observation of single isolated electrons of high transverse momentum in events with missing transverse energy at the CERN $\bar{p}p$ collider". In: *Physics Letters B* 122B(5,6) (1983), pp. 476–485. DOI: 10.1016/0370-2693(83)91605-2.

[16] The UA1 Collaboration. "Experimental observation of lepton pairs of invariant mass around 95 GeV/c^2 at the CERN SPS collider". In: *Physics Letters B* 126B(5) (1983), pp. 398–410. DOI: 10.1016/0370-2693(83)90188-0.

[17] P. W. Higgs. "Broken symmetries, massless particles and gauge fields". In: *Physics Letters* 12(2) (1964), pp. 132–133. DOI: 10.1016/0031-9163(64)91136-9.

[18] P. W. Higgs. "Broken Symmetries and the Masses of Gauge Bosons". In: *Physical Review Letters* 13(16) (1964), pp. 508–509. DOI: 10.1103/PhysRevLett.13.508.

[19] F. Englert and R. Brout. "Broken Symmetry and the Mass of Gauge Vector Mesons". In: *Physical Review Letters* 13(9) (1964), pp. 321–323. DOI: 10.1103/PhysRevLett.13.321.

[20] I. B. Alonso et al., eds. *High-Luminosity Large Hadron Collider (HL-LHC). Technical design report*. Vol. 10. CERN Yellow Reports: Monographs. Geneva: CERN, 2020. ISBN: 978-92-9083-587-5. DOI: 10.23731/CYRM-2020-0010.

[21] The ALICE Collaboration. "The ALICE experiment at the CERN LHC". In: *Journal of Instrumentation (JINST)* 3 S08002 (2008). DOI: 10.1088/1748-0221/3/08/s08002.

[22] The LHCb Collaboration. "The LHCb Detector at the LHC". In: *Journal of Instrumentation (JINST)* 3 S08005 (2008). DOI: 10.1088/1748-0221/3/08/s08005.

[23] The ATLAS Collaboration. "The ATLAS Experiment at the CERN Large Hadron Collider". In: *Journal of Instrumentation (JINST)* 3 S08003 (2008). DOI: 10.1088/1748-0221/3/08/s08003.

[24] The CMS Collaboration. "The CMS experiment at the CERN LHC". In: *Journal of Instrumentation (JINST)* 3 S08004 (2008). DOI: 10.1088/1748-0221/3/08/s08004.

[25] L. Evans and P. Bryant. "LHC Machine". In: *Journal of Instrumentation (JINST)* 3 S08001 (2008). DOI: 10.1088/1748-0221/3/08/S08001.

[26] A. D. Rosso. "Particle Kickers". In: *CERN Bulletin* 24-25/2014 (2014). URL: http://cds.cern.ch/record/1708739/files/2014-24-25-E-web.pdf.

[27] D. d'Enterria. "CMS physics highlights in the LHC Run 1". In: *Proceedings of 53rd International Winter Meeting on Nuclear Physics*. Vol. 238. 2015. DOI: 10.22323/1.238.0027.

[28] B. Salvachua. "Overview of Proton-Proton Physics during Run 2". In: *Proceedings of the 2019 Evian Workshop on LHC Beam Operations. Session 1: Overview of Run 2*. Geneva: CERN, 2019, pp. 7–14. URL: https://cds.cern.ch/record/2750272.

[29] J. T. Boyd. "LHC Run-2 and future prospects". In: *Proceedings of the 2019 European School of High-Energy Physics*. Ed. by C. Duhr and M. Mulders. Vol. 5. 2021. DOI: 10.23730/CYRSP-2021-005.247.

[30] K. Bernhard-Novotny. *First Run 3 physics result by CMS*. 2022. URL: https://home.cern/news/news/physics/first-run-3-physics-result-cms. Accessed November 16, 2022.

[31] CERN. *LHC nominal lumi projection*. no date. URL: https://lhc-commissioning.web.cern.ch/schedule/images/LHC-nominal-lumi-projection.png. Accessed November 16, 2022.

[32] D. Barney. *CMS Detector Slice*. CMS Collection. 2016. URL: https://cds.cern.ch/record/2120661. Accessed November 18, 2022.

[33] A. Sirunyan et al. "Particle-flow reconstruction and global event description with the CMS detector". In: *Journal of Instrumentation (JINST)* 12 P10003 (2017). DOI: 10.1088/1748-0221/12/10/P10003.

[34] S. Donato. "CMS trigger performance". In: *6th International Conference on New Frontiers in Physics (ICNFP 2017)* 182(02037) (2018). DOI: 10.1051/epjconf/201818202037.

[35] M. Cacciari, G. P. Salam, and G. Soyez. "The anti-k_t jet clustering algorithm". In: *Journal of High Energy Physics (JHEP)* 2008(04) (2008). DOI: 10.1088/1126-6708/2008/04/063.

[36] The CMS collaboration. *Performance of the DeepJet b tagging algorithm using 41.9/fb of data from proton-proton collisions at 13TeV with Phase 1 CMS detector*. 2018. URL: https://cds.cern.ch/record/2646773.

[37] M. S. Neubauer and A. Roy. *Explainable AI for High Energy Physics*. 2022. arXiv: 2206.06632 [hep-ex].

[38] J. Shlomi, P. Battaglia, and J.-R. Vlimant. "Graph neural networks in particle physics". In: *Machine Learning: Science and Technology* 2(2) 021001 (2021). DOI: 10.1088/2632-2153/abbf9a.

[39] R. Rojas. *Neural Networks. A Systematic Introduction*. Berlin: Springer Berlin, Heidelberg, 1996. ISBN: 978-3-642-61068-4. DOI: 10.1007/978-3-642-61068-4.

[40] O. I. Abiodun et al. "State-of-the-art in artificial neural network applications: A survey". In: *Heliyon* 4(11) e00938 (2018). DOI: 10.1016/j.heliyon.2018.e00938.

[41] M. A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. URL: http://neuralnetworksanddeeplearning.com/.

[42] K. C. Fukushima. "Cognitron: A Self-organizing Multilayered Neural Network". In: *Biological Cybernetics* 20 (1975), pp. 121–136. DOI: 10.1007/BF00342633.

[43] V. Nair and G. E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th International Conference on Machine Learning (ICML)*. 2010. URL: https://dl.acm.org/doi/10.5555/3104322.3104425.

[44] C. E. Nwankpa et al. "Activation Functions: Comparison of Trends in Practice and Research for Deep Learning". In: *Proceedings of the 2nd International Conference on Computational Sciences and Technology (INCCST)*. 2021, pp. 124–133. URL: https://pure.strath.ac.uk/ws/portalfiles/portal/118946797/Nwankpa_etal_ICCST_2021_Activation_functions_comparison_of_trends_in_practice.pdf.

[45] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *Proceedings of the 3rd International Conference for Learning Representations (ICLR)*. 2015. arXiv: 1412.6980v9 [cs.LG].

[46] X. Ying. "An Overview of Overfitting and its Solutions". In: *Journal of Physics: Conference Series* 1168(2) 022022 (2019). DOI: 10.1088/1742-6596/1168/2/022022.

[47] W. L. Hamilton. *Graph Representation Learning*. Ed. by P. S. Ronald Brachman Francesca Rossi. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2020. ISBN: 978-1-68173-964-9. DOI: 10.2200/S01045ED1V01Y202009AIM046.

[48] P. Hartmann. *Mathematik für Informatiker. Ein praxisbezogenes Lehrbuch*. 7th ed. Springer Vieweg Wiesbaden, 2019. ISBN: 978-3-658-26524-3. DOI: 10.1007/978-3-658-26524-3.

[49] R. Trudeau. *Introduction to Graph Theory*. Dover Books on Mathematics. New York: Dover Publications Inc., 1993. ISBN: 978-0-486-67870-2.

[50] I. Stanimirović and M. Tasic. "Performance comparison of storage formats for sparse matrices". In: *Facta Universitatis. Series Mathematics and Informatics* 24 (2009), pp. 39–51. URL: http://facta.junis.ni.ac.rs/mai/mai24/fumi-24_39_51.pdf.

[51] A. M. Deiana et al. "Applications and Techniques for Fast Machine Learning in Science". In: *Frontiers in Big Data* 5:787421 (2022). DOI: 10.3389/fdata.2022.787421.

[52] P. W. Battaglia et al. "Relational inductive biases, deep learning, and graph networks". In: *Computing Research Repository (CoRR)* (2018). arXiv: 1806.01261v3 [cs.LG].

[53] J. Gilmer et al. "Neural Message Passing for Quantum Chemistry". In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. Vol. 70. JMLR.org, 2017, pp. 1263–1272. arXiv: 1704.01212v2 [cs.LG].

[54] T. N. Kipf and M. Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2017. arXiv: 1609.02907v4 [cs.LG].

[55] R. M. Schmidt. "Recurrent Neural Networks (RNNs): A gentle Introduction and Overview". In: *Computing Research Repository (CoRR)* (2019). arXiv: 1912.05911 [cs.LG].

[56] Y. Li et al. "Gated Graph Sequence Neural Networks". In: *International Conference for Learning Representations (ICLR)* (2017). arXiv: 1511.05493v4 [cs.LG].

[57] K. Cho et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179.

[58] J. Zhou et al. "Graph neural networks: A review of methods and applications". In: *AI Open* 1 (2020), pp. 57–81. DOI: 10.1016/j.aiopen.2021.01.001.

[59] F. Xu et al. "Explainable AI: A Brief Survey on History, Research Areas, Approaches and Challenges". In: *Natural Language Processing and Chinese Computing. 8th CCF International Conference, NLPCC 2019, Dunhuang, China, October 9–14, 2019, Proceedings, Part II*. Vol. LNAI 11839. Lecture Notes in Computer Science. Springer, Cham, 2019, pp. 563–574. DOI: 10.1007/978-3-030-32236-6_51.

[60] H. Yuan et al. "Explainability in Graph Neural Networks: A Taxonomic Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022). DOI: 10.1109/TPAMI.2022.3204236.

[61] Kowsari et al. "Text Classification Algorithms: A Survey". In: *Information* 10(4) (2019), p. 150. DOI: 10.3390/info10040150.

[62] M. Fratello et al. "Multi-View Ensemble Classification of Brain Connectivity Images for Neurodegeneration Type Discrimination". In: *Neuroinformatics* 15 (2017), pp. 199–213. DOI: 10.1007/s12021-017-9324-2.

[63] R. Ying et al. "GNNExplainer: Generating Explanations for Graph Neural Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Ed. by H. Wallach et al. Vol. 32. 2019. URL: https://proceedings.neurips.cc/paper/2019/file/d80b7040b773199015de6d3b4293c8ff-Paper.pdf.

[64] J. Fosso-Tande. *Applications of Taylor series*. 2013. URL: http://sces.phys.utk.edu/~moreo/mm08/fosso.pdf.

[65] S. Wunsch et al. "Identifying the Relevant Dependencies of the Neural Network Response on Characteristics of the Input Space". In: *Computing and Software for Big Science* 2(5) (2018). DOI: 10.1007/s41781-018-0012-1.

[66] G. S. Heine. "Illustration of the Neural Network Learning Process during Training". Bachelor thesis. Karlsruhe Institute of Technology (KIT), 2019. URL: https://publish.etp.kit.edu/record/21984.

[67] G. B. Arfken and H. J. Weber. *Mathematical methods for physicists*. 6th ed. Elsevier Academic Press, 2005. ISBN: 0-12-059876-0.

[68] T. Halenke. "Studien zu Graph Neural Networks in $t\bar{t}$ + $b\bar{b}$-Prozessen am CMS-Experiment". Bachelor thesis. Karlsruhe Institute of Technology (KIT), 2021. URL: https://publish.etp.kit.edu/record/22074.

[69] M. Erhart. "Simultaneous Cross Section Measurements of $t\bar{t}$ + X processes using Graph Neural Networks". Master thesis. Karlsruhe Institute of Technology (KIT), 2022.

[70] M. Fey and J. E. Lenssen. "Fast Graph Representation Learning with PyTorch Geometric". In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*. 2019. arXiv: 1903.02428v3 [cs.LG].

[71] A. Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS)*. Ed. by H. Wallach et al. 2019. URL: https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

[72] PyTorch Contributors. *PyTorch Documentation*. 2022. URL: https://pytorch.org/docs/stable/generated/torch.use_deterministic_algorithms.html. Accessed October 26, 2022.

[73] S. Alioli et al. "A general framework for implementing NLO calculations in shower Monte Carlo programs: the POWHEG BOX". In: *Journal of High Energy Physics (JHEP)* 2010(43) (2010). DOI: 10.1007/JHEP06(2010)043.

[74] T. Sjöstrand et al. "An introduction to PYTHIA 8.2". In: *Computer Physics Communications* 191 (2015), pp. 159–177. DOI: 10.1016/j.cpc.2015.01.024.

[75] The CMS Collaboration. "Extraction and validation of a new set of CMS PYTHIA8 tunes from underlying-event measurements". In: *The European Physical Journal C* 80(4) (2020). DOI: 10.1140/epjc/s10052-019-7499-4.

[76] F. Maltoni, G. Ridolfi, and M. Ubiali. "b-initiated processes at the LHC: a reappraisal". In: *Journal of High Energy Physics (JHEP)* 2012(22) (2012). DOI: 10.1007/JHEP07(2012)022.

[77] E. L. Pfeffer. "Studies on $t\bar{t}$+$b\bar{b}$ production at the CMS experiment". Master thesis. Karlsruhe Institute of Technology (KIT), 2021. URL: https://publish.etp.kit.edu/record/22082.

[78] T. Fawcett. "An introduction to ROC analysis". In: *Pattern Recognition Letters* 27(8) (2006), pp. 861–874. DOI: 10.1016/j.patrec.2005.10.010.

[79] I. Henrion et al. "Neural Message Passing for Jet Physics". In: *Deep Learning for Physical Sciences Workshop at the 31st Conference on Neural Information Processing Systems (NeurIPS)* (2017). URL: https://dl4physicalsciences.github.io/files/nips_dlps_2017_29.pdf.

[80] E. A. Moreno et al. “JEDI-net: a jet identification algorithm based on interaction networks”. In: *The European Physical Journal C* 80(58) (2020). DOI: 10.1140/epjc/s10052-020-7608-4.

[81] C. Morris et al. “Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33(01). 2019, pp. 4602–4609. DOI: 10.1609/aaai.v33i01.33014602.

Appendix

A Distribution of Observables

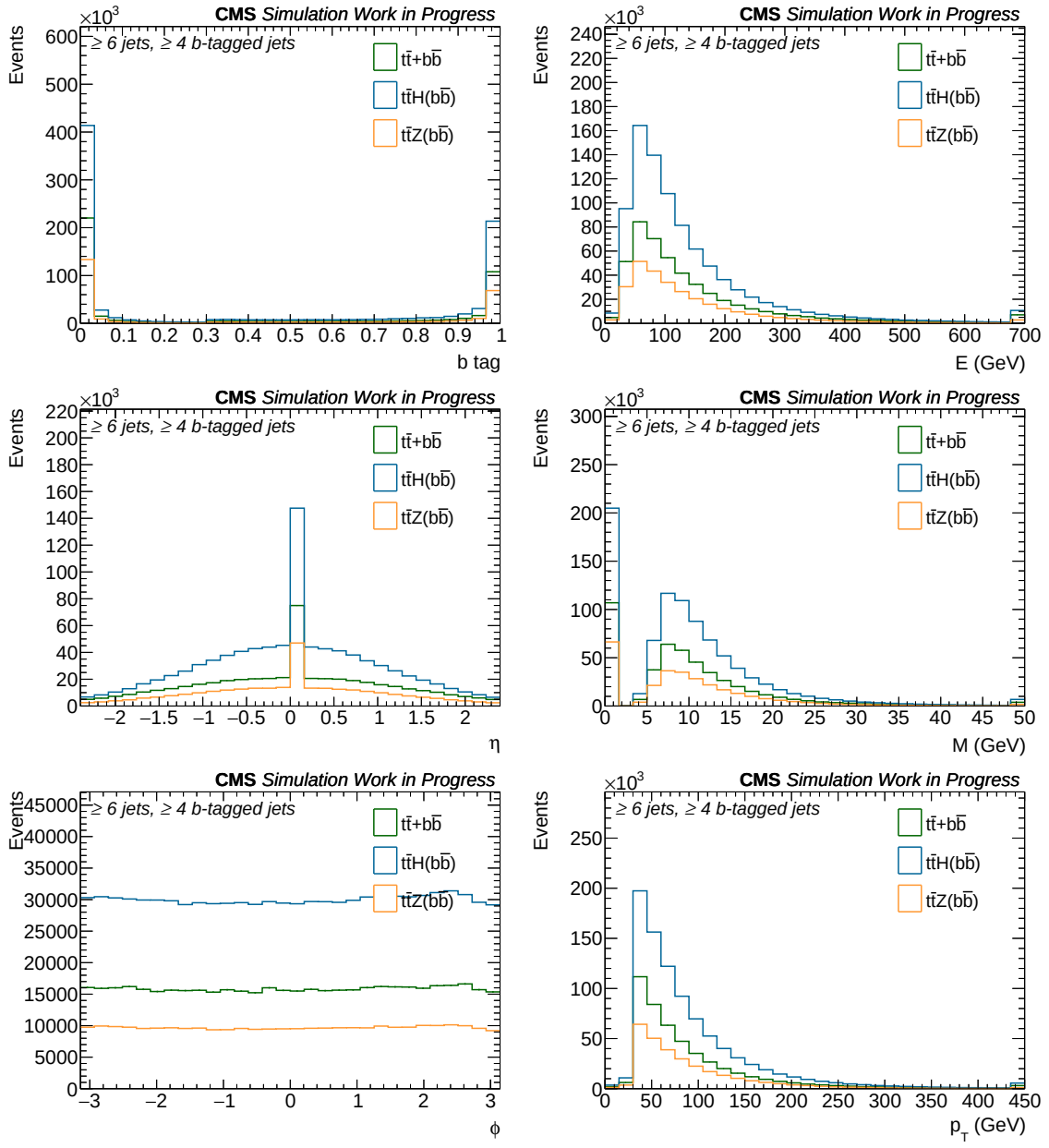


Figure A.1: **Distribution of the deployed vertex attributes pre-standardization.** For each histogram 30 equidistant bins are chosen. Not all of this information exists for leptons and neutrinos. The particular features are set to zero then.

B Decision Basis for Outlier Criterion (b)

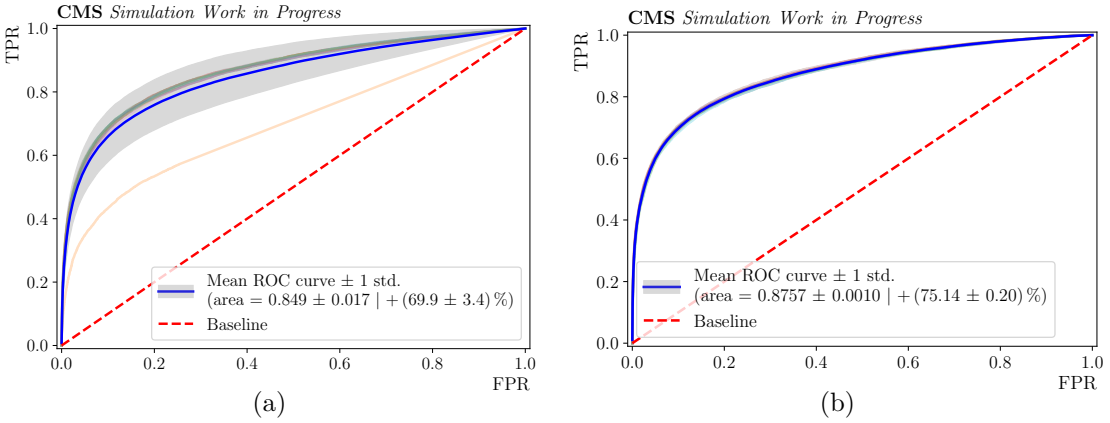


Figure B.1: **Exemplary ROC curves.** In addition to the mean ROC curve (dark blue) and its spread (gray band, square root of the unbiased sample variance), which are calculated on the basis of the ROC-AUC values obtained in the ten realized repetitions of a training, the ROC curve of each repetition is displayed in both plots. Without the second subcondition of criterion (b), the model corresponding to the orange ROC curve in (a) and the models corresponding to the light blue and brown curves in (b) would be discarded. With that subcondition, only the former is identified as an outlier.

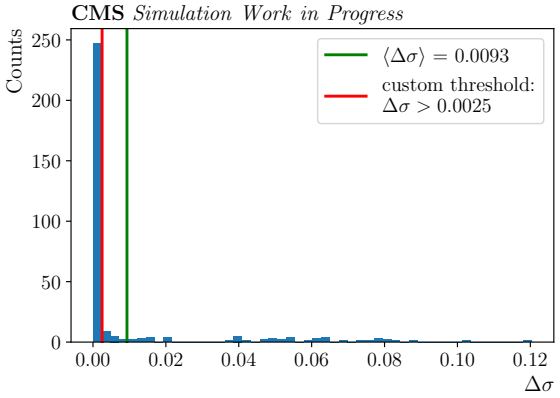


Figure B.2: **Decision basis for the second subcondition of criterion (b).** Histogram of the standard deviation (square root of the unbiased sample variance) difference pre- and post-removal of models with ROC-AUC values beyond the range of $\langle\text{ROC-AUC}\rangle \pm 1.5 \cdot \sigma^{\text{pre}}_{\text{ROC-AUC}}$. The histogram consists of 50 equidistant bins and contains the $\Delta\sigma$ values of all models analyzed in this thesis. With the selected threshold of $\Delta\sigma > 0.0025$, the majority of the falsely identified outliers (cf. Fig. B.1b) can be avoided.

Models with ROC-AUC values beyond the empirically chosen range of $\langle\text{ROC-AUC}\rangle \pm 1.5 \cdot \sigma^{\text{pre}}_{\text{ROC-AUC}}$ (first subcondition of criterion (b) of Section 4.3) shall be identified as outliers and discarded from further consideration. That applies, for instance, to the model corresponding to the orange ROC curve in Fig. B.1a. However, if a training has an exceptionally narrow performance spread, also models like the ones corresponding to the light blue and brown ROC curves in Fig. B.1b fall under the outlier category. This is clearly not desired and therefore, the additional subcondition $\Delta\sigma > 0.0025$ is introduced. The histogram in Fig B.2 forms the decision basis for this choice. As shown in the histogram, the majority of these unwanted cases can be avoided with this threshold value, while models like the one corresponding to the orange curve in Fig. B.1a are still identified as outliers.

C Normalized Performance Rates of NLP-GGSNN

Table C.1: **Normalized performance rates of NLP-GGSNN.** All values in the table are given in %. The subscript of CR designates the category that the NLP-GGSNN confuses with an additional b jet. Due to rounding errors, the sum of the mean confusion rates per class is not necessarily 100 %. The same applies to the sum of the 1/2 and 0/2 rates per class. These rates form the basis of the AddB-X modeling.

class	$\langle \mathrm{CR}_{\mathrm{HadTopB}} \rangle$	$\langle \mathrm{CR}_{\mathrm{LepTopB}} \rangle$	$\langle \mathrm{CR}_{\mathrm{HadTopQ}} \rangle$	$\langle \mathrm{CR}_{\mathrm{Unknown}} \rangle$	$\frac{\langle \mathrm{CR}_{\mathrm{Lepton}} \rangle}{\langle \mathrm{CR}_{\mathrm{Missing}} \rangle}$	1/2 rate	0/2 rate
$\mathrm{t\bar{t}H(b\bar{b})}$	36.89 ± 0.19	51.23 ± 0.19	8.49 ± 0.08	3.397 ± 0.033	0.0 ± 0.0	89 ± 10	11 ± 10
$\mathrm{t\bar{t}Z(b\bar{b})}$	32.58 ± 0.14	52.97 ± 0.16	10.84 ± 0.12	3.608 ± 0.033	0.0 ± 0.0	88 ± 11	12 ± 11
$\mathrm{t\bar{t}} + \mathrm{b\bar{b}}$	34.12 ± 0.12	48.88 ± 0.13	10.52 ± 0.08	6.48 ± 0.05	0.0 ± 0.0	84 ± 11	16 ± 11

D Properties of the Manipulated Data Sets

Table D.1: **Fraction of manipulated objects per category in the manipulated datasets.** All values in the table are given in %.

modeling strategy	fraction of manipulated events	fraction of manipulated objects in the categories						
		AddB	HadTopB	LepTopB	HadTopQ	Unknown	Lepton/Missing	
AddB-LTB	10	5.00	0.00	10.00	0.00	0.00	0.00	
	20	10.00	0.00	20.00	0.00	0.00	0.00	
	30	15.00	0.00	30.00	0.00	0.00	0.00	
	40	20.00	0.00	40.00	0.00	0.00	0.00	
	50	25.00	0.00	50.00	0.00	0.00	0.00	
	60	30.00	0.00	60.00	0.00	0.00	0.00	
	70	35.00	0.00	70.00	0.00	0.00	0.00	
	80	40.00	0.00	80.00	0.00	0.00	0.00	
	90	45.00	0.00	90.00	0.00	0.00	0.00	
	100	50.00	0.00	100.00	0.00	0.00	0.00	
AddB-X	10	5.60	4.10	5.62	0.61	0.34	0.00	
	20	11.21	8.17	11.31	1.21	0.66	0.00	
	30	16.80	12.24	16.97	1.79	1.00	0.00	
	40	22.45	16.44	22.63	2.38	1.34	0.00	
	50	28.07	20.62	28.19	2.98	1.70	0.00	
	60	33.69	24.74	33.85	3.58	2.03	0.00	
	70	39.32	28.97	39.41	4.18	2.39	0.00	
	80	44.91	33.14	45.00	4.76	2.71	0.00	
	90	50.53	37.31	50.60	5.35	3.07	0.00	
	100	56.13	41.49	56.18	5.94	3.40	0.00	

E Derivation of $\Delta r^*_{\mathbf{max}}$

Proof. Let M be the number of objects q in a finite ranking and

$$\langle \Delta r \rangle = \frac{1}{M} \sum_{i=1}^{M} \Delta r(q_i) \tag{8.1}$$

$$= \frac{1}{M} \sum_{i=1}^{M} \left| r_A(q_i) - r_B(q_i) \right| \tag{8.2}$$

the mean absolute difference in rank between a finite ranking A and a finite ranking B. Thereby, $r_A(q_i)$ and $r_B(q_i)$ denote the rank of q in the finite rankings $\{A, B\}$ and it is assumed that A and B are only comprised of the exact same objects. The maximum mean absolute difference in rank is obtained when the sum is maximal. This is the case if each summand in the sum is maximal. As this is dependent on every other summand in finite rankings, it is justified to start with finding the maximum of the first summand and then continue successively with the next summands. Obviously, if the highest and lowest ranked object in A are ranked in the exact opposite way in B, the first two summands are maximal and corresponds to $M - 1$ each. The subsequent two summands in Equation 8.1 are maximal and corresponds to $(M - 1) - 2$ each if the second highest ranked object and the next-to-last ranked object in A switch their positions in B and so on. This can be executed m times, where

$$m = \begin{cases} \frac{M}{2}\,, & \text{if } M \text{ is even.} \\ \frac{M-1}{2}\,, & \text{if } M \text{ is odd.} \end{cases} \tag{8.3}$$

For the maximum mean absolute rank difference, it holds

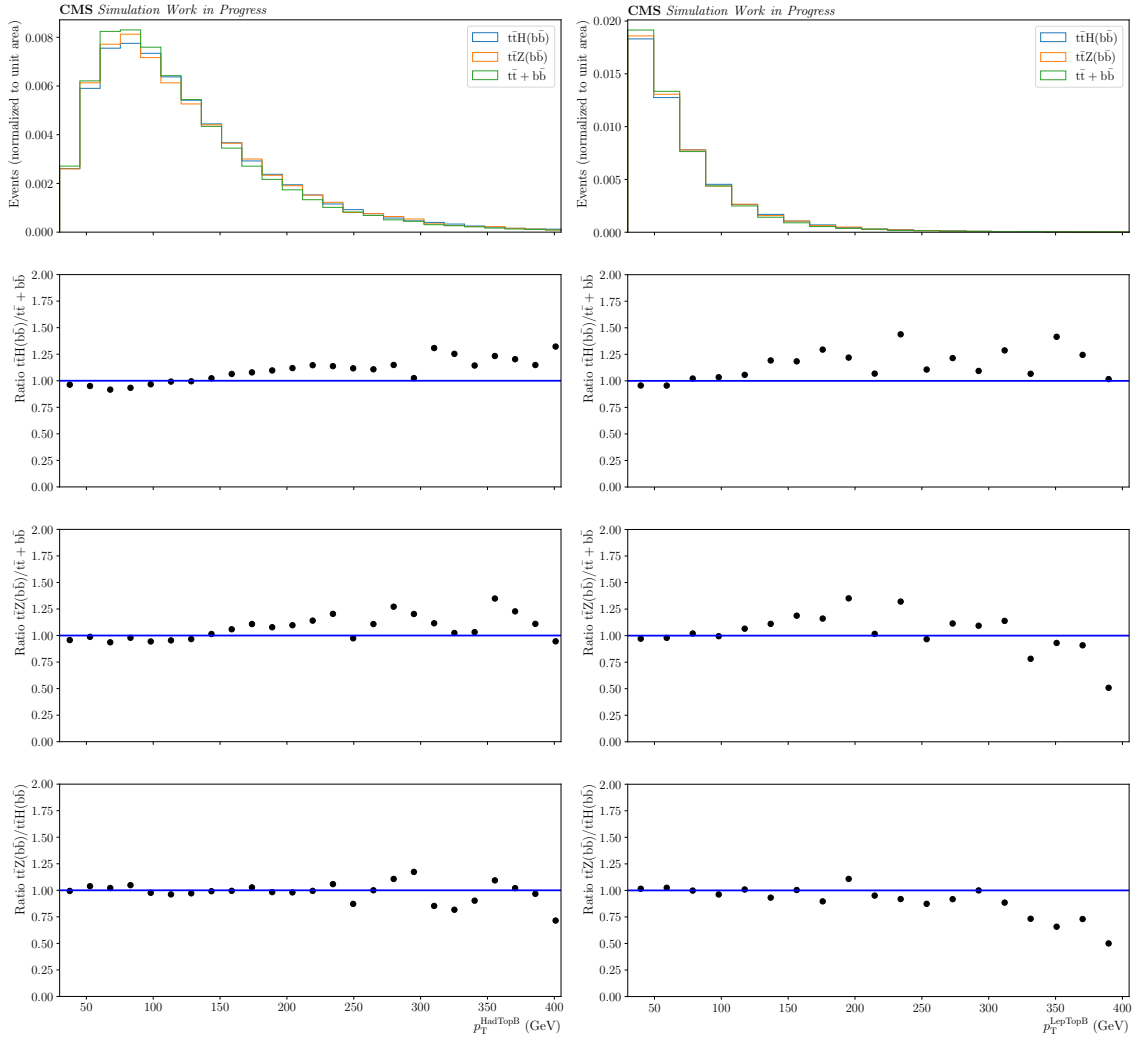
$$\Delta r^*_{\max} = 2 \cdot \frac{1}{M} \left((M-1) + ((M-1) - 2) + \cdots + \left(\left(M - \frac{m}{2} + 1 \right) - \frac{m}{2} \right) \right) \tag{8.4}$$

$$= \begin{cases} 2 \cdot \frac{1}{M} \sum\limits_{i=0}^{m} (2i+1)\,, & \text{if } M \text{ is even.} \\ 2 \cdot \frac{1}{M} \sum\limits_{i=0}^{m} 2i\,, & \text{if } M \text{ is odd.} \end{cases} \tag{8.5}$$

$$= \begin{cases} \frac{1}{M} \cdot \frac{M}{2}^2\,, & \text{if } M \text{ is even.} \\ \frac{1}{M} \cdot \frac{M^2-1}{2}\,, & \text{if } M \text{ is odd.} \end{cases} \tag{8.6}$$

□

F Supplementary Information to Chapter 6



(a) (b)

Figure F.1: **Distribution of the transverse momenta of HadTopB (a) and LepTopB jets (b).** All histograms consist of 100 equidistant bins and are individually normalized to unit area for an enhanced comparability. Both observables appear to only have minor discriminating power for $t\bar{t}$+X events.

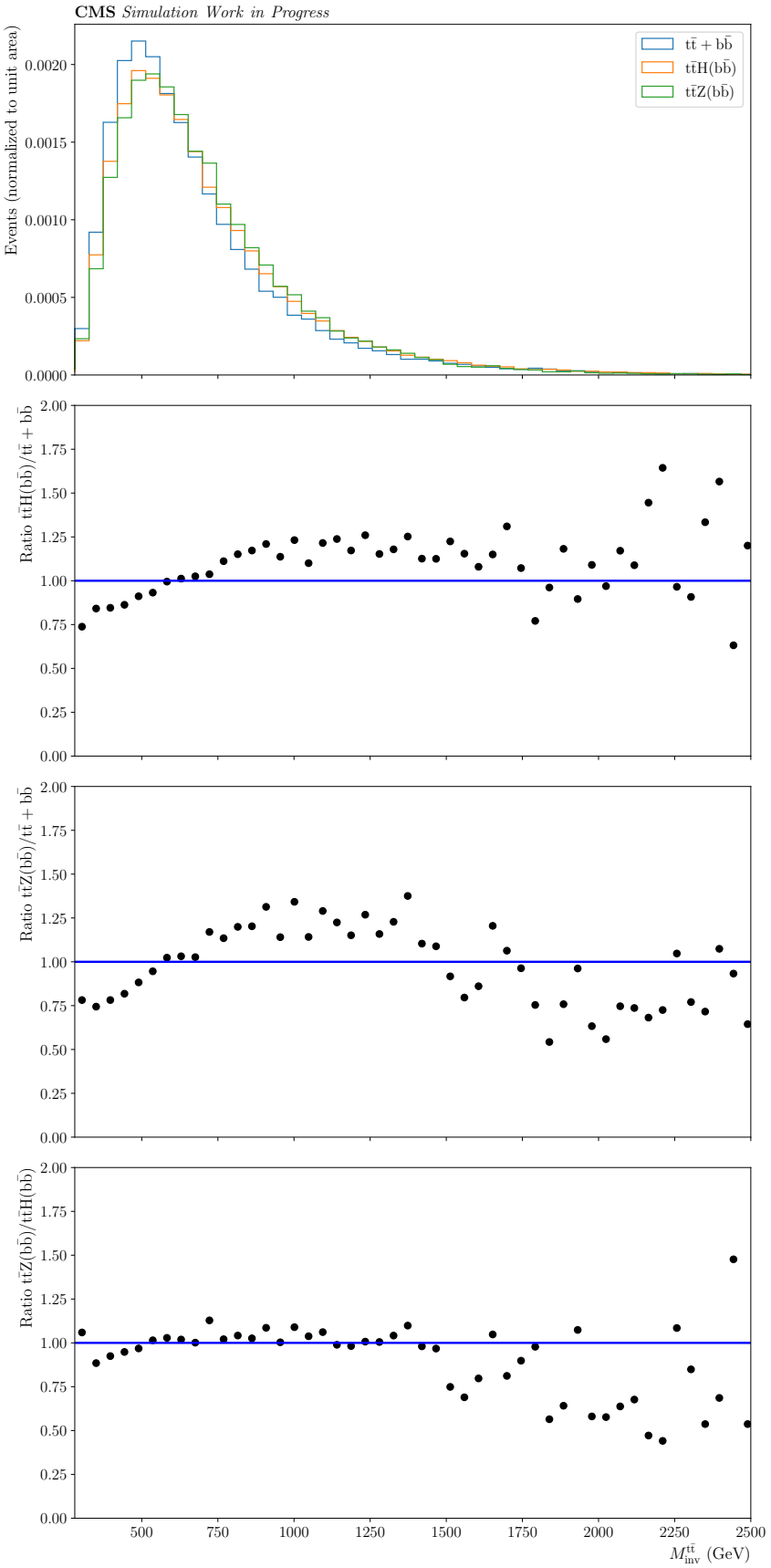


Figure F.2: **Distribution of the invariant mass of the $t\bar{t}$ system.** All histograms consist of 100 equidistant bins and are individually normalized to unit area for an enhanced comparability. The invariant mass of the $t\bar{t}$ system $M^{t\bar{t}}_{inv}$ does not show any discriminating power for $t\bar{t}$+X events.

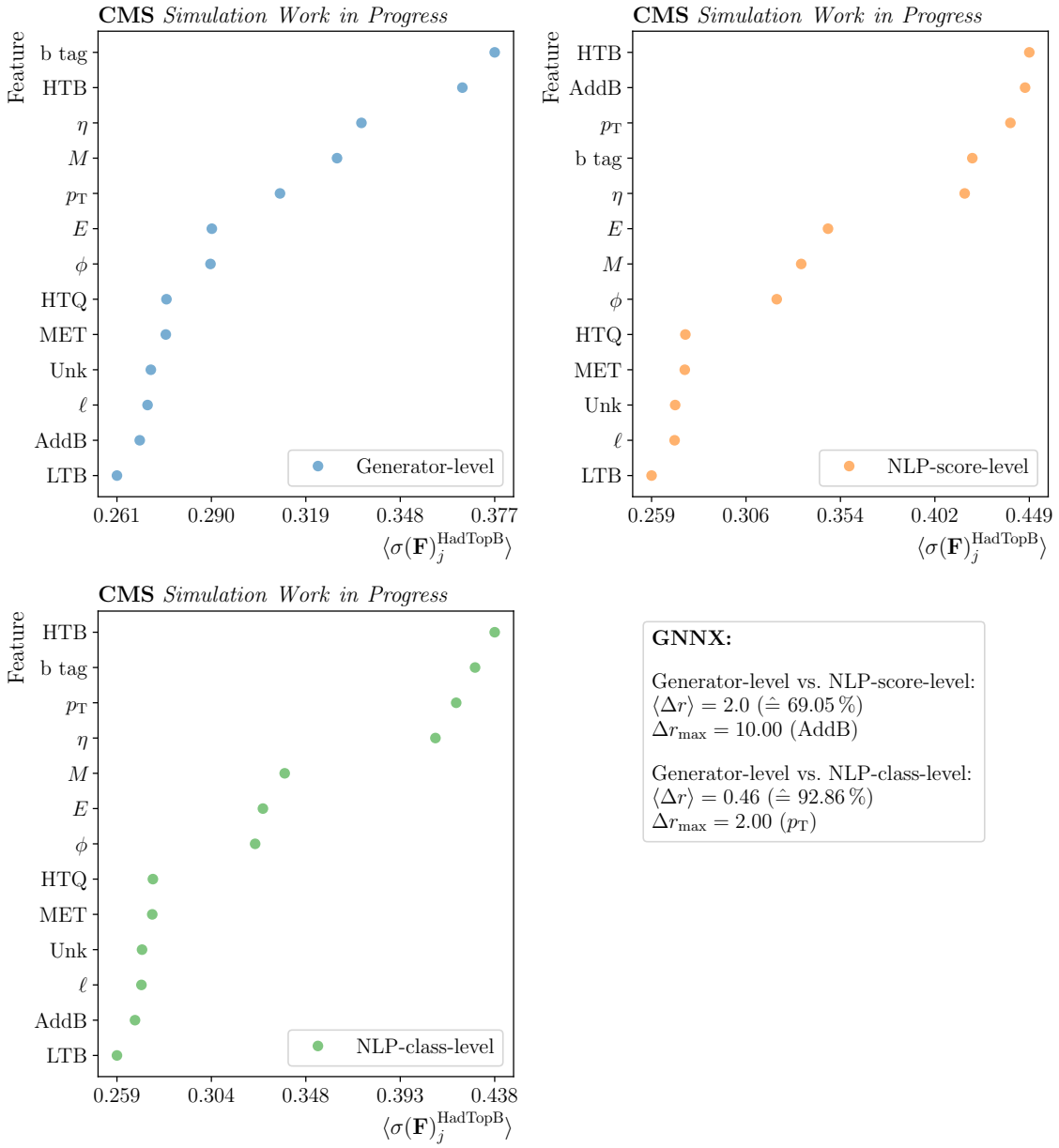


Figure F.3: **Ranking of the GNNX-based HadTopB specific feature importance for the generator-level (top left), NLP-score-level (top right) and NLP-class-level (bottom left) GNN response.** The most important category flag is the one corresponding to the actual category under scrutiny (HadTopB) across all GNNs. The remaining category flags are only of minor importance. The percentage in the parentheses corresponds to the conformity $1 - \langle\Delta r\rangle/\Delta r^*_{\text{max}}$ of the compared rankings. The variable Δr_{max} denotes the highest number of rank difference between the compared rankings and should not be confused with Δr^*_{max}, which denotes the maximum mean number of absolute rank differences between the compared rankings. The objects to which this applies are specified in the subsequent parentheses.

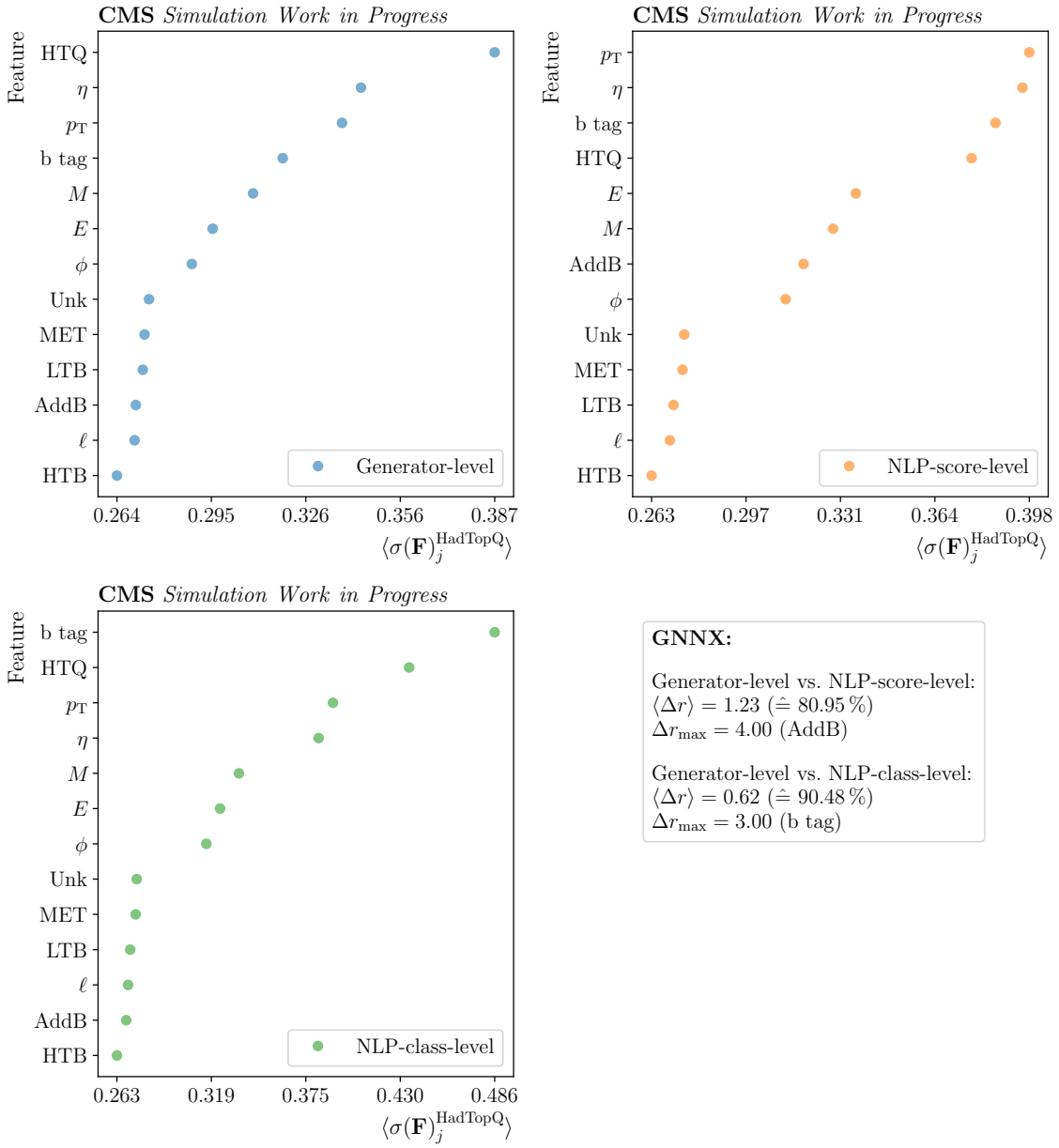


Figure F.4: **Ranking of the GNNX-based HadTopQ specific feature importance for the generator-level (top left), NLP-score-level (top right) and NLP-class-level (bottom left) GNN response.** The most important category flag is the one corresponding to the actual category under scrutiny (HadTopQ) across all GNNs. The remaining category flags are only of minor importance. The percentage in the parentheses corresponds to the conformity $1 - \langle \Delta r \rangle / \Delta r^*_{\text{max}}$ of the compared rankings. The variable Δr_{max} denotes the highest number of rank difference between the compared rankings and should not be confused with Δr^*_{max}, which denotes the maximum mean number of absolute rank differences between the compared rankings. The objects to which this applies are specified in the subsequent parentheses.

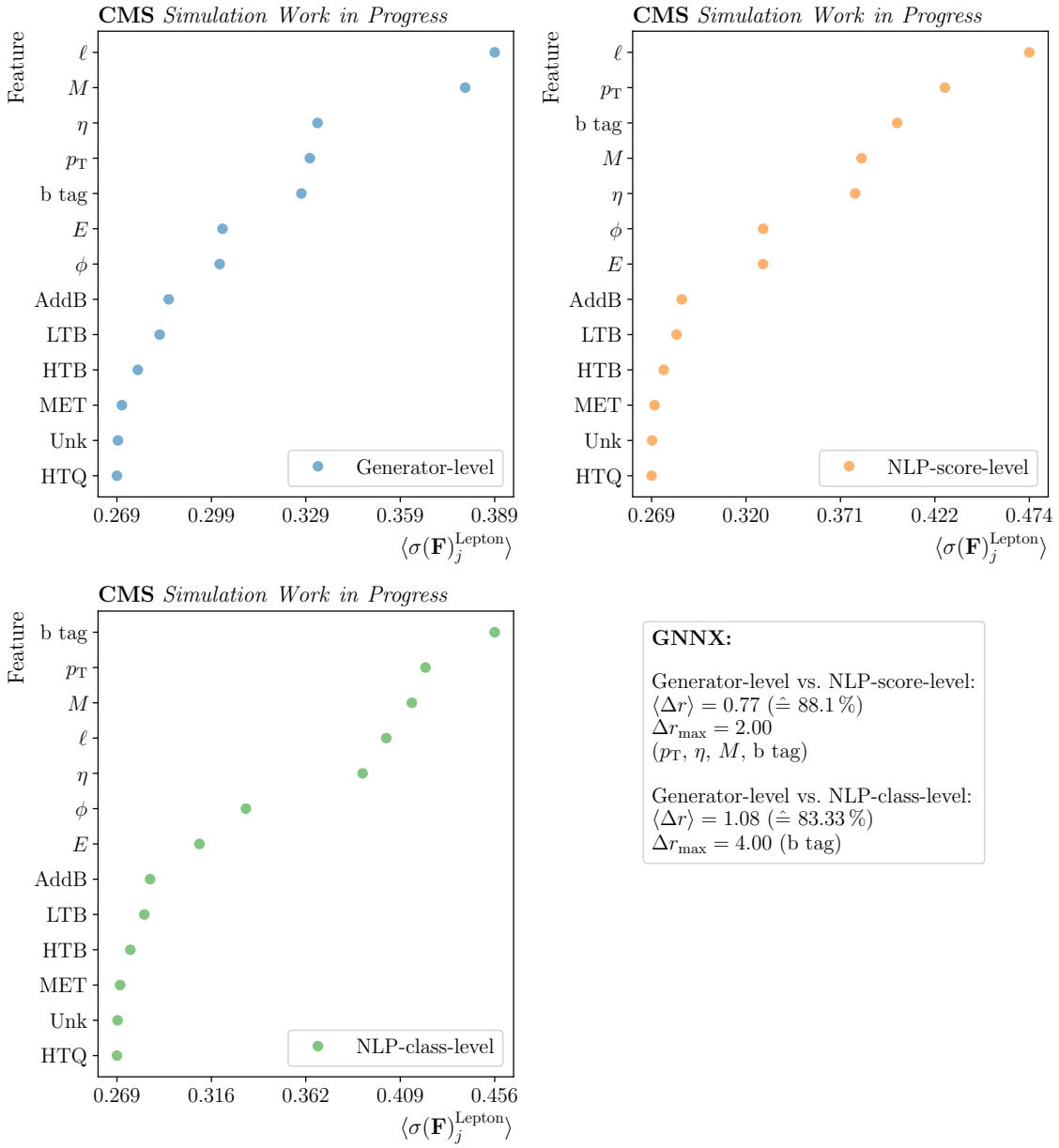


Figure F.5: **Ranking of the GNNX-based Lepton specific feature importance for the generator-level (top left), NLP-score-level (top right) and NLP-class-level (bottom left) GNN response.** The most important category flag is the one corresponding to the actual category under scrutiny (Lepton) across all GNNs. The remaining category flags are only of minor importance. The percentage in the parentheses corresponds to the conformity $1 - \langle \Delta r \rangle / \Delta r^*_{\text{max}}$ of the compared rankings. The variable Δr_{max} denotes the highest number of rank difference between the compared rankings and should not be confused with Δr^*_{max}, which denotes the maximum mean number of absolute rank differences between the compared rankings. The objects to which this applies are specified in the subsequent parentheses.

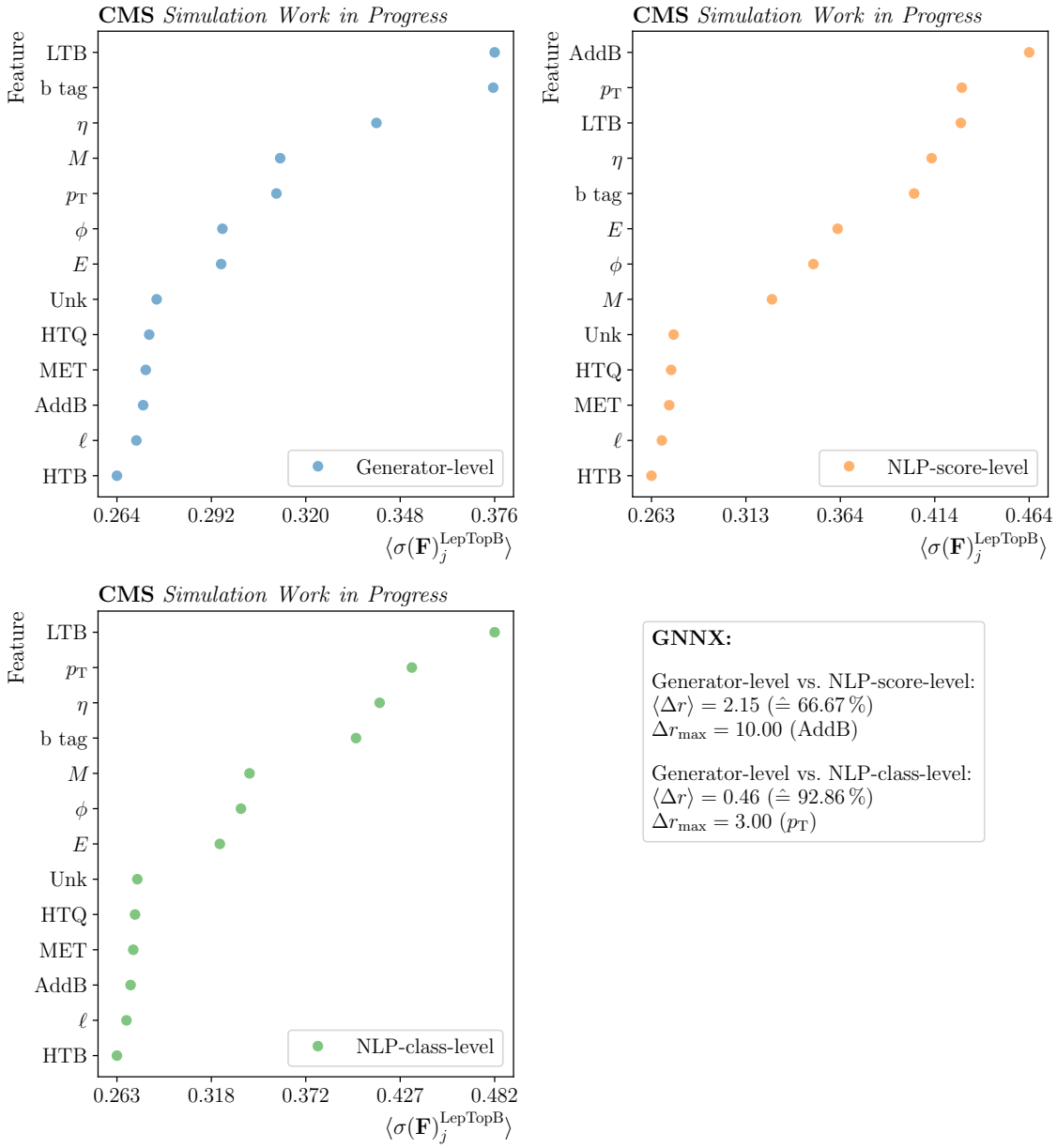


Figure F.6: **Ranking of the GNNX-based LepTopB specific feature importance for the generator-level (top left), NLP-score-level (top right) and NLP-class-level (bottom left) GNN response.** As a matter of fact, the AddB flag is the most important flag and feature in LepTopB vertices for the NLP-score-level GNN response. The remaining category flags are in all rankings only of minor importance. The percentage in the parentheses corresponds to the conformity $1 - \langle\Delta r\rangle / \Delta r^*_{\text{max}}$ of the compared rankings. The variable Δr_{max} denotes the highest number of rank difference between the compared rankings and should not be confused with Δr^*_{max}, which denotes the maximum mean number of absolute rank differences between the compared rankings. The objects to which this applies are specified in the subsequent parentheses.

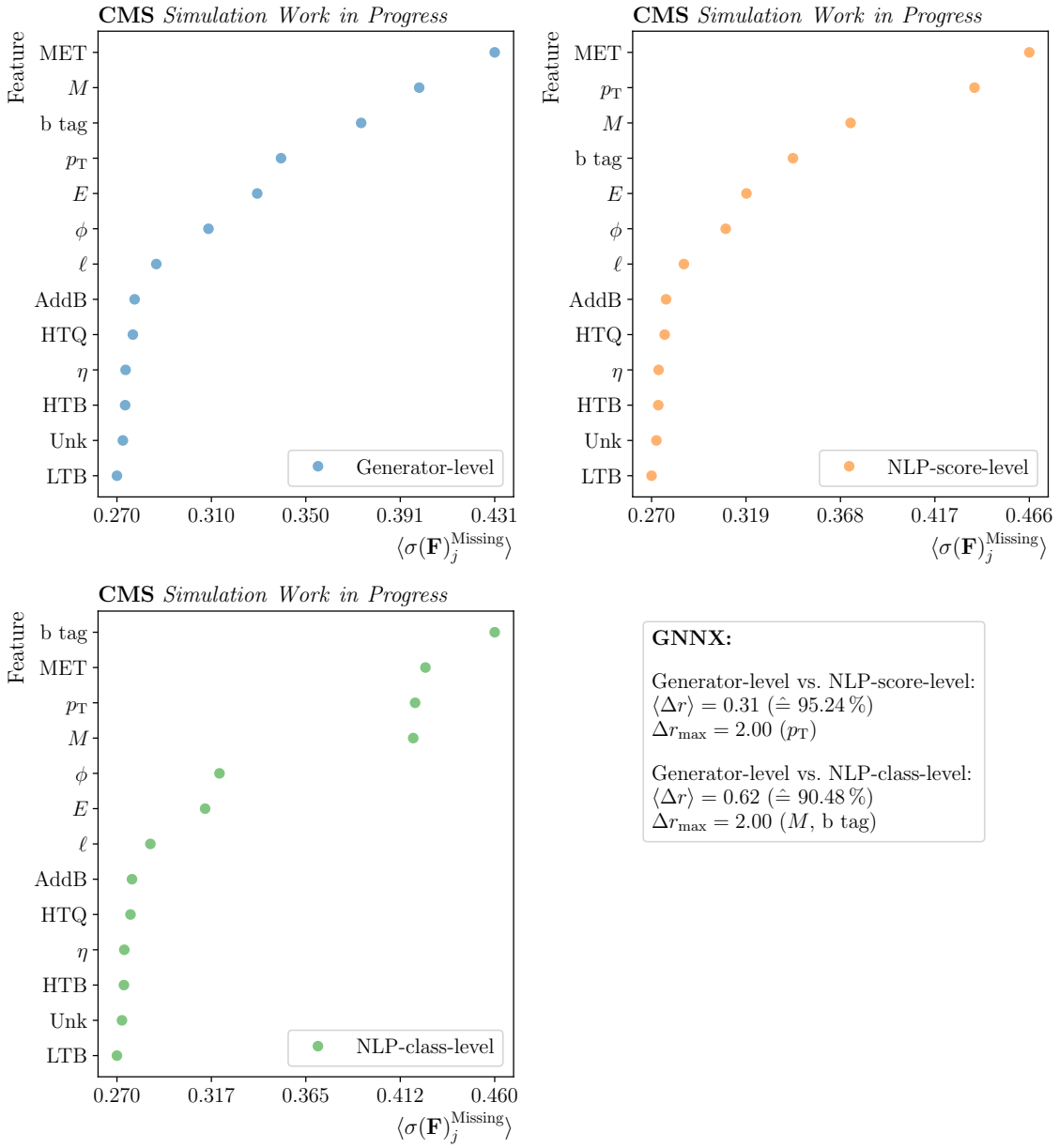


Figure F.7: **Ranking of the GNNX-based Missing specific feature importance for the generator-level (top left), NLP-score-level (top right) and NLP-class-level (bottom left) GNN response.** The most important category flag is the one corresponding to the actual category under scrutiny (Missing) across all GNNs. The remaining category flags are in all rankings only of minor importance. The percentage in the parentheses corresponds to the conformity $1 - \langle\Delta r\rangle / \Delta r^*_{\text{max}}$ of the compared rankings. The variable Δr_{max} denotes the highest number of rank difference between the compared rankings and should not be confused with Δr^*_{max}, which denotes the maximum mean number of absolute rank differences between the compared rankings. The objects to which this applies are specified in the subsequent parentheses.

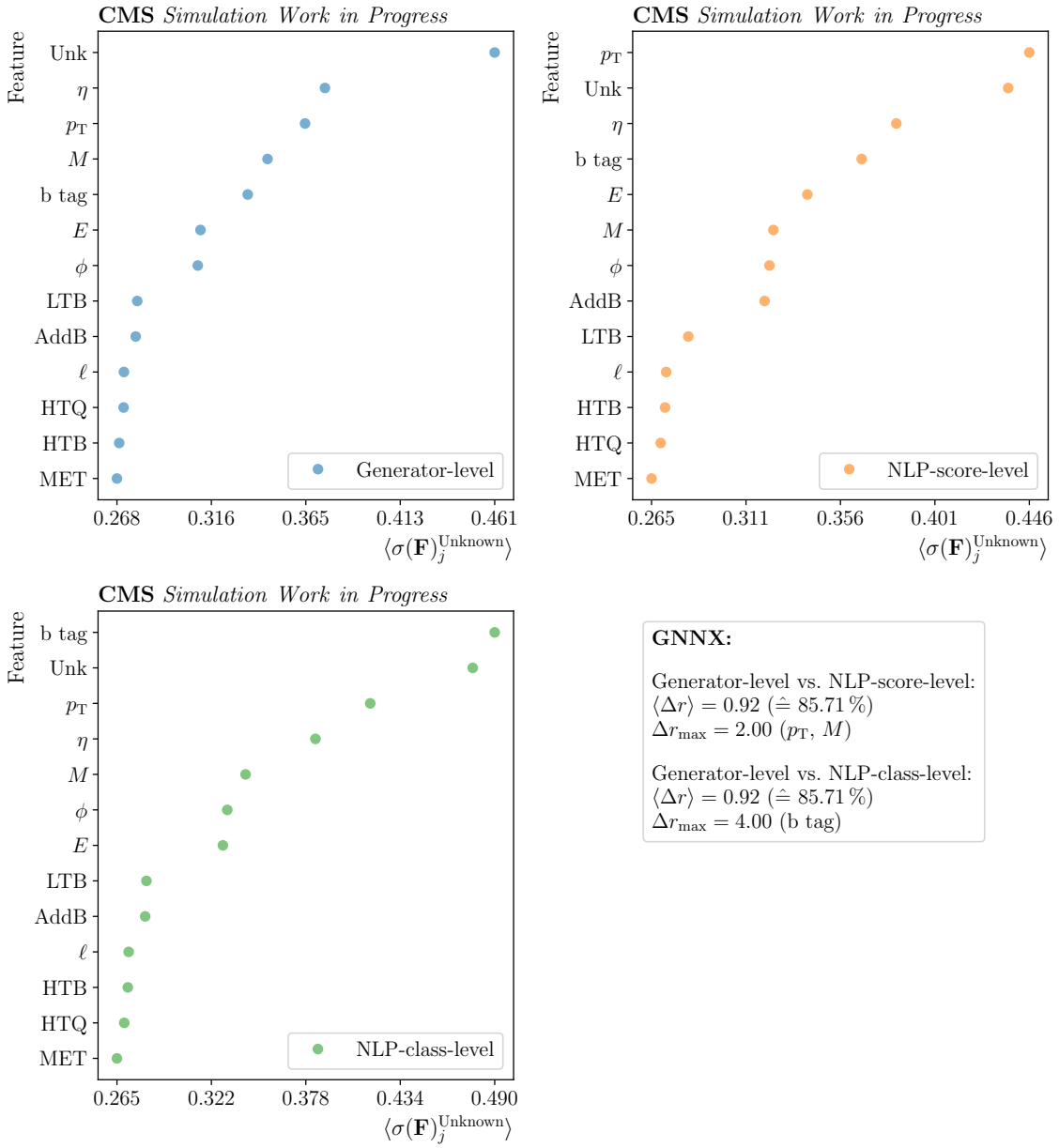


Figure F.8: **Ranking of the GNNX-based Unknown specific feature importance for the generator-level (top left), NLP-score-level (top right) and NLP-class-level (bottom left) GNN response.** The most important category flag is the one corresponding to the actual category under scrutiny (Unknown) across all GNNs. The remaining category flags are in all rankings only of minor importance. The percentage in the parentheses corresponds to the conformity $1 - \langle \Delta r \rangle / \Delta r^*_{\text{max}}$ of the compared rankings. The variable Δr_{max} denotes the highest number of rank difference between the compared rankings and should not be confused with Δr^*_{max}, which denotes the maximum mean number of absolute rank differences between the compared rankings. The objects to which this applies are specified in the subsequent parentheses.

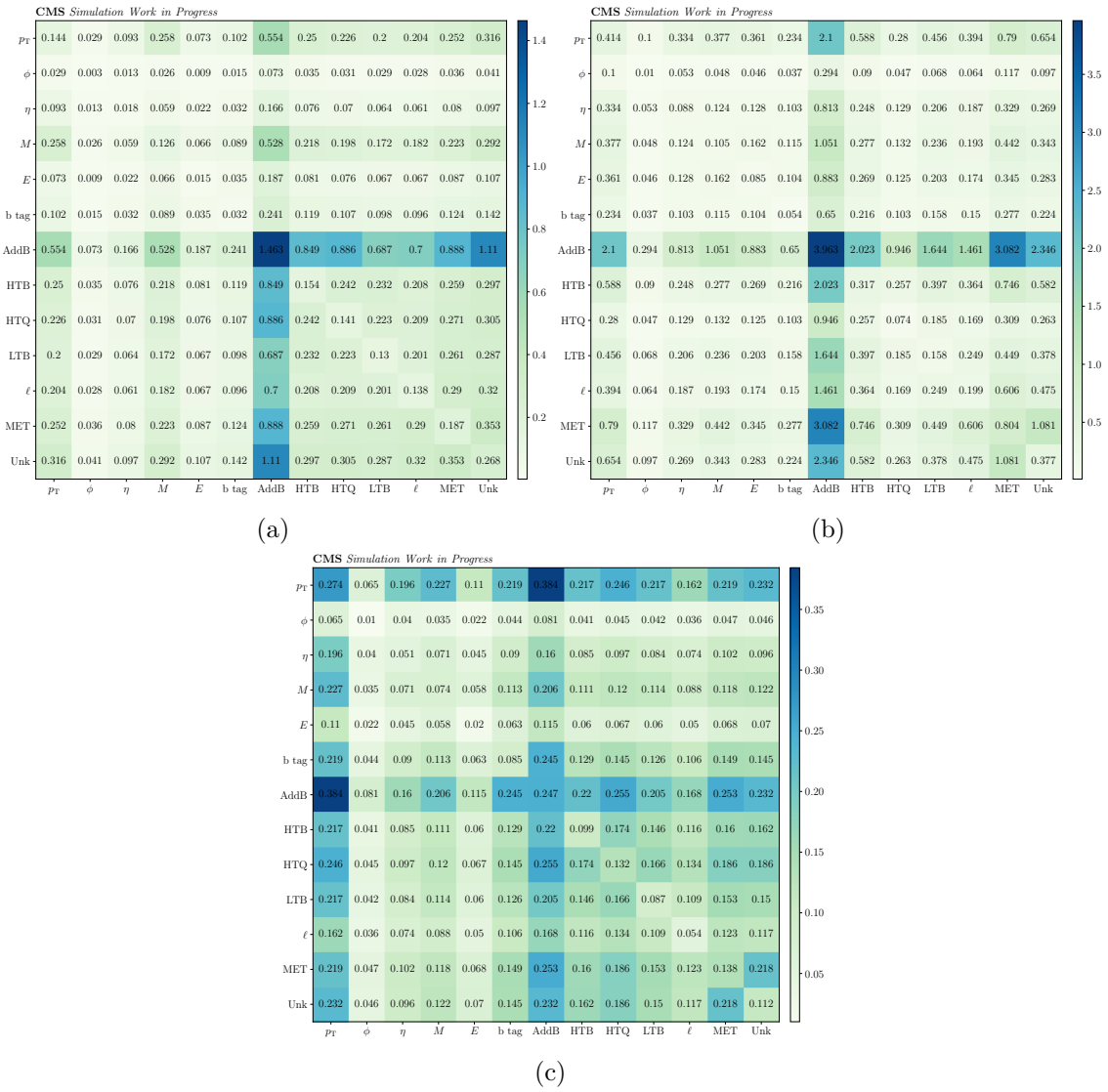


Figure F.9: **Second-order Taylor coefficients of the generator-level (a), NLP-score-level (b) and NLP-class-level GNN (c).** While the generator-level GNN is mainly focused on correlations between the AddB flag and the remaining flags (or rather it can rely on the correctness of the information captured in the provided AddB flags), the NLP-score-level GNN response and in particular the NLP-class-level GNN response are also dependent on information gained from correlations between other features, e.g., correlations between p_T with all other features. Hence, the AddB flags provided to the training of the NLP-class-level GNN are of the lowest quality of all information levels.

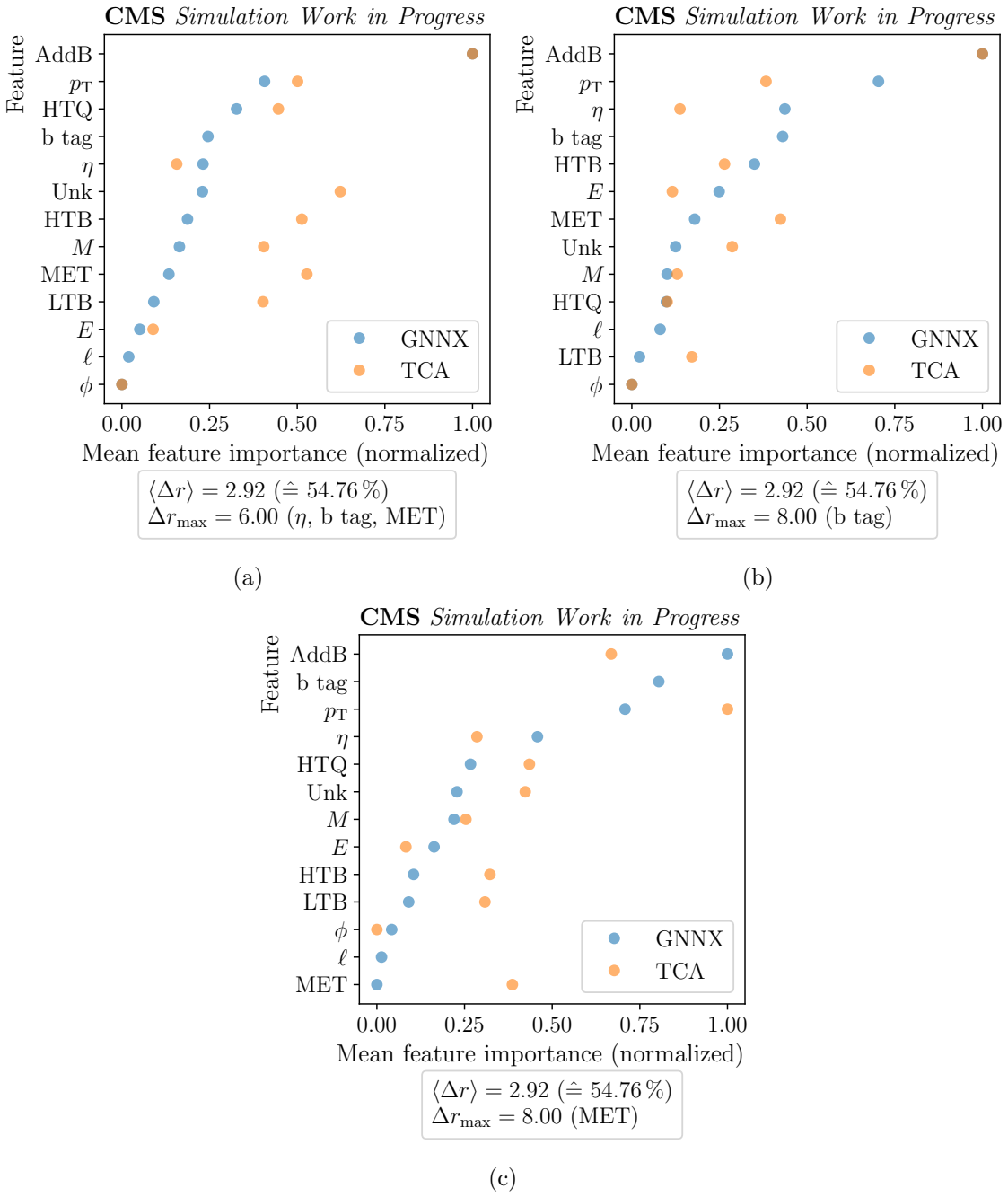


Figure F.10: **Combined GNNX- and TCA-based normalized feature importance ranking for the generator-level (a), NLP-score-level (b) and NLP-class-level (c) GNN.** The percentage in the parentheses corresponds to the conformity $1 - \langle \Delta r \rangle / \Delta r^*_{max}$ of the compared rankings. The variable Δr_{max} denotes the highest number of rank difference between the compared rankings and should not be confused with Δr^*_{max}, which denotes the maximum mean number of absolute rank differences between the compared rankings. The objects to which this applies are specified in the subsequent parentheses. The conformity between the GNNX- and TCA-based rankings is in all cases 54.76 %.

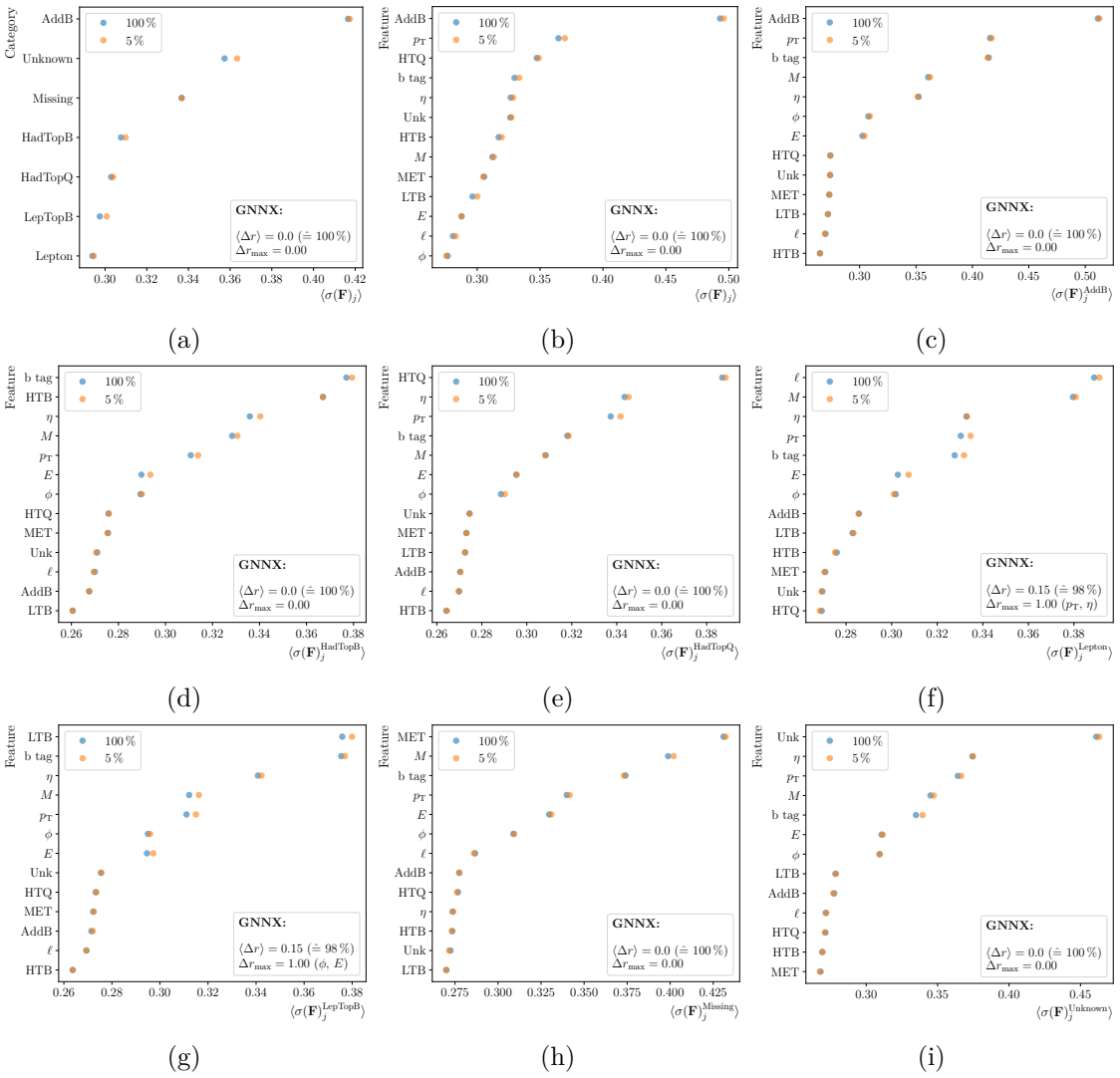


Figure F.11: **Comparison of GNNX-based rankings calculated with both 5 % and 100 % of the test samples for the generator-level GNN.** The percentage in the parentheses corresponds to the conformity $1 - \langle \Delta r \rangle / \Delta r^*_{\text{max}}$ of the compared rankings. The variable Δr_{max} denotes the highest number of rank difference between the compared rankings and should not be confused with Δr^*_{max}, which denotes the maximum mean number of absolute rank differences between the compared rankings. The objects to which this applies are specified in the subsequent parentheses. In the majority of the cases, the conformity between the rankings calculated on the basis of the whole test set and on only 5 % of the test set is 100 %. In the worst case, the conformity is still 98 %.

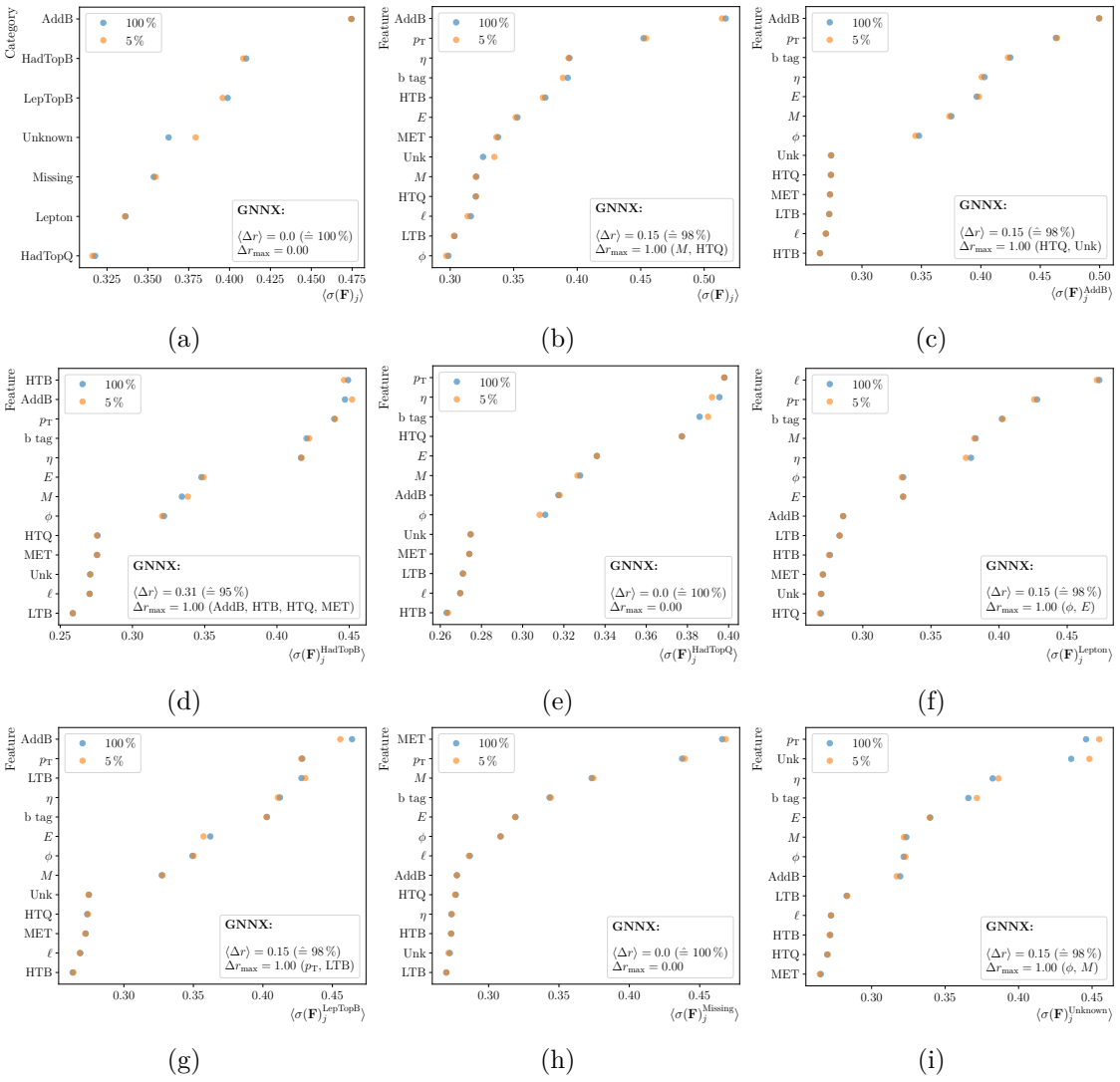


Figure F.12: **Comparison of GNNX-based rankings calculated with both 5 % and 100 % of the test samples for the NLP-score-level GNN.** The percentage in the parentheses corresponds to the conformity $1 - \langle \Delta r \rangle / \Delta r^*_{\max}$ of the compared rankings. The variable $\Delta r_{\max}$ denotes the highest number of rank difference between the compared rankings and should not be confused with $\Delta r^*_{\max}$, which denotes the maximum mean number of absolute rank differences between the compared rankings. The objects to which this applies are specified in the subsequent parentheses. The conformity between the rankings calculated on the basis of the whole test set and on only 5 % of the test set is 95 % at worst.

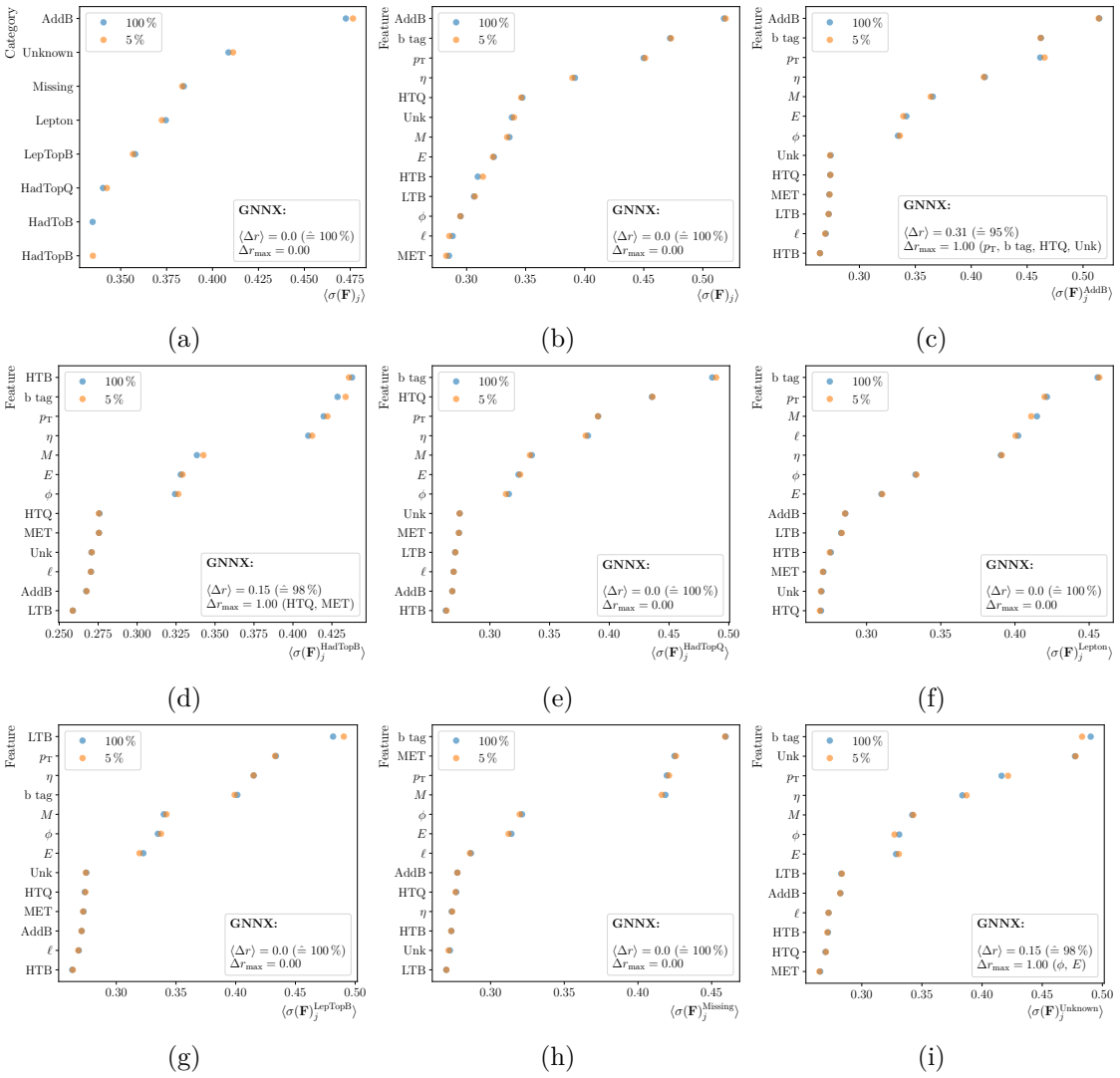


Figure F.13: **Comparison of GNNX-based rankings calculated with both 5 % and 100 % of the test samples for NLP-class-level GNN.** The percentage in the parentheses corresponds to the conformity $1 - \langle \Delta r \rangle / \Delta r^*_{\max}$ of the compared rankings. The variable $\Delta r_{\max}$ denotes the highest number of rank difference between the compared rankings and should not be confused with $\Delta r^*_{\max}$, which denotes the maximum mean number of absolute rank differences between the compared rankings. The objects to which this applies are specified in the subsequent parentheses. In the majority of the cases the conformity between the rankings calculated on the basis of the whole test set and on only 5 % of the test set is 98 % or 100 %. In the worst, case the conformity is still 95 %.

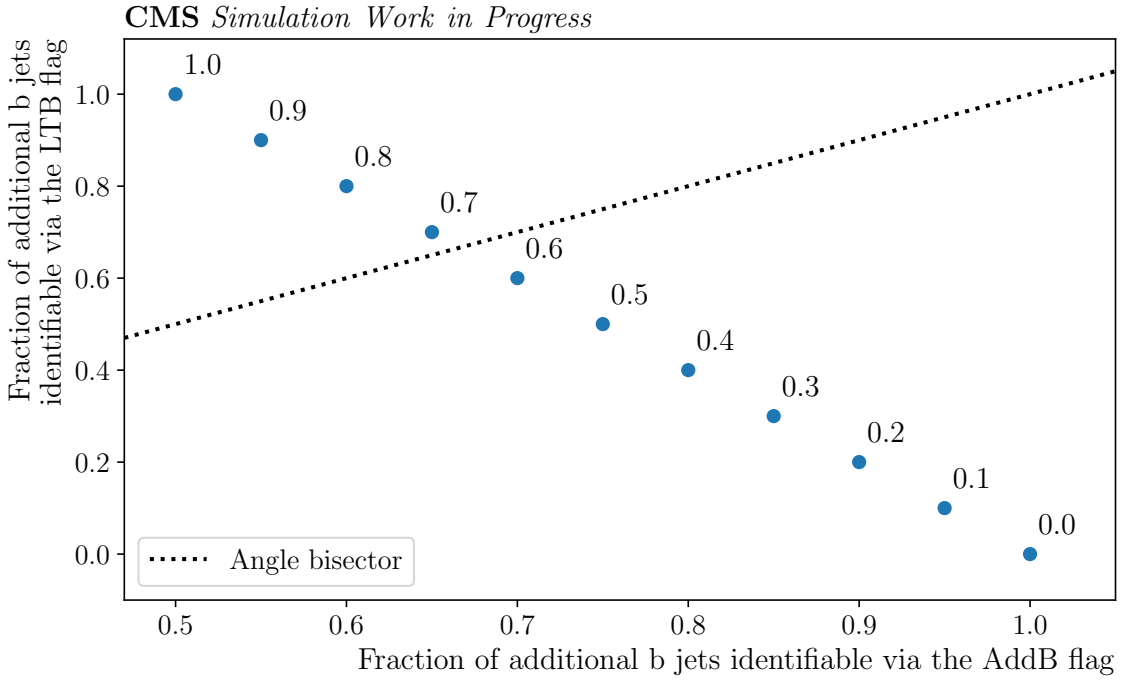
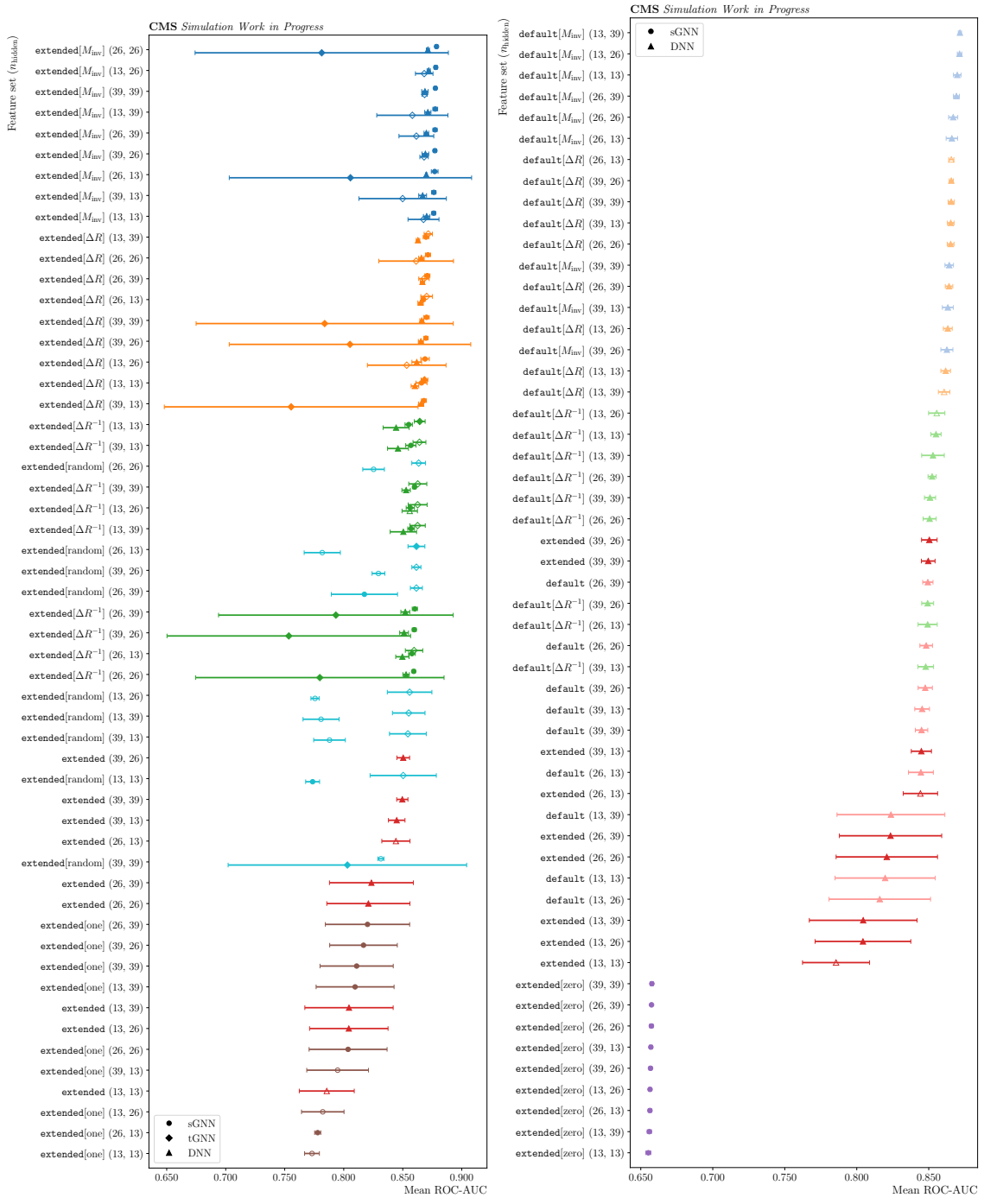


Figure F.14: **Fraction of additional b jets identifiable via the AddB flag vs. fraction of additional b jets identifiable via the LTB flag.** The fraction of additional b jets identifiable via the LTB flag corresponds – due to the flag swap in AddB-LTB modeling – to the fraction of manipulated objects in the category LepTopB whereas the fraction of additional b jets identifiable via the AddB flag equals to 1 − (the fraction of manipulated objects in the category AddB) (cf. Table. D.1). The figure displayed next to each data point denotes the fraction of manipulated events in the data set. It is clearly noticeable that the data point corresponding to a data set with 70 % manipulated events is the closest to the angle bisector, i.e., in this data set, the information content of the AddB flag and the LTB flag is the most similar among all data sets manipulated according to the AddB-LTB modeling. Accordingly, the GNN trained with this data set is confused the most and hence, its performance is the worst (cf. Fig. 5.1). In data set with a fraction of manipulated events higher than 70 %, more additional b jets are identifiable via the LTB flag than via the AddB flag so that the performance of GNNs trained with these data sets increases again.

G Supplementary Information to Chapter 7

Table G.1: **Architecture and feature sets of all models considered in Section 7.2.** Each neural network type (tGNN, sGNN, DNN) comes with twelve different combinations of hidden nodes and hidden layers. Feature sets used in the training of (all twelve configurations of) a neural network type are marked with ✔ in the table, otherwise with ✗. Consequently, there are in total 120 GNNs and 96 DNNs for benchmarking in Section 7.2.

	tGNN	sGNN	DNN
n_{hidden}	(13), (26), (39), $(13,13)$, $(13,26)$, $(13,39)$, $(26,13)$, $(26,26)$, $(26,39)$, $(39,13)$, $(39,26)$, $(39,39)$		
`default`	✗	✗	✔
`default`$[M_{\text{inv}}]$	✗	✗	✔
`default`$[\Delta R]$	✗	✗	✔
`default`$[\Delta R^{-1}]$	✗	✗	✔
`extended`	✗	✗	✔
`extended`[zero]	✗	✔	✗
`extended`[one]	✗	✔	✗
`extended`[random]	✔	✔	✗
`extended`$[M_{\text{inv}}]$	✔	✔	✔
`extended`$[\Delta R]$	✔	✔	✔
`extended`$[\Delta R^{-1}]$	✔	✔	✔



(a) (b)

Figure G.1: **Combined mean performance of GNN_{2HL} and DNN_{2HL}.** For reason of presentation the ranking is split in two. The mean performance of models trained with `extended`$[x]$, $x \in \{M_{\text{inv}}, \Delta R, \Delta R^{-1}, \text{random}, \text{one}\}$ are located in (a) and the remaining models in (b). Data points with unfilled markers indicate that at least one model is identified as an outlier (cf. outlier criteria in Section 4.3) and therefore discarded. All data points possess an error bar representing the performance spread (square root of the unbiased sample variance), which is calculated on the basis of the ROC-AUC values of all not discarded models (at most ten models, one for each of the ten realized repetitions of a training). Data points of models sharing the same label on the y-axis are shifted in a way that they are stacked in descending order of the corresponding mean ROC-AUC value from top to bottom to enhance the visibility of the error bars. The color grouping of the data points is still recognizable for two-layer models, yet not as distinct as for one-layer models, however, the performance of equivalent DNNs and sGNNs and tGNNs, respectively, remains similar.

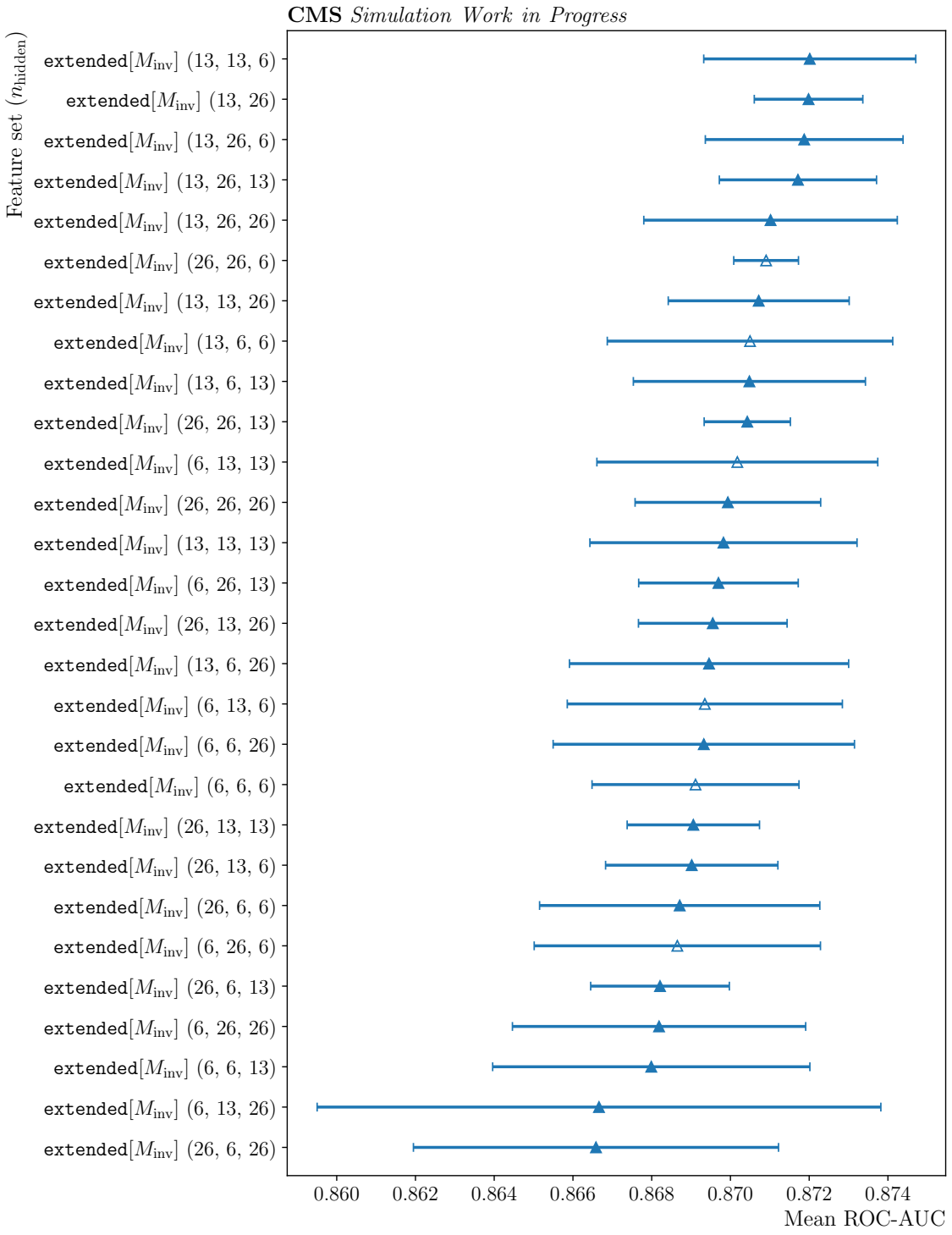
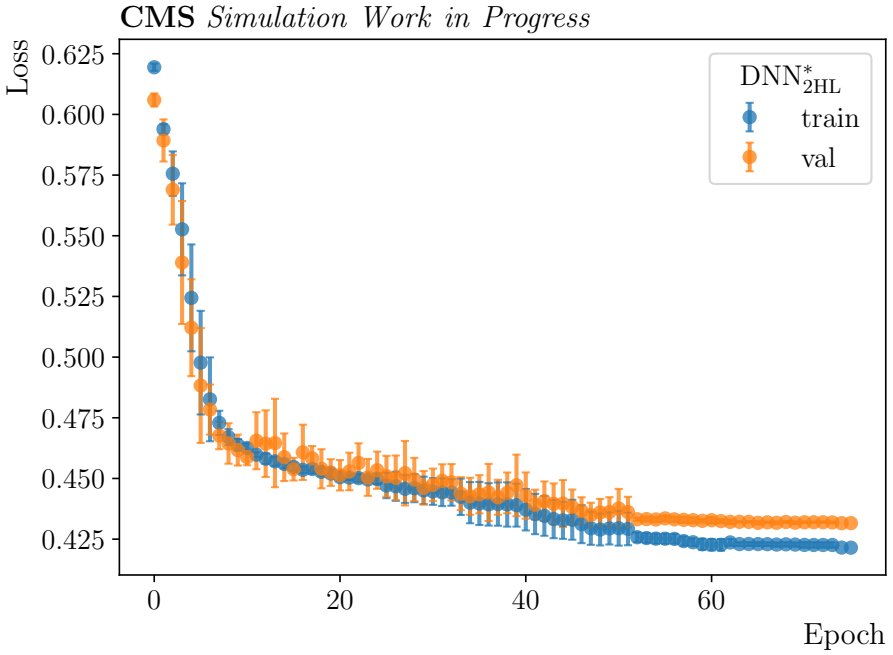
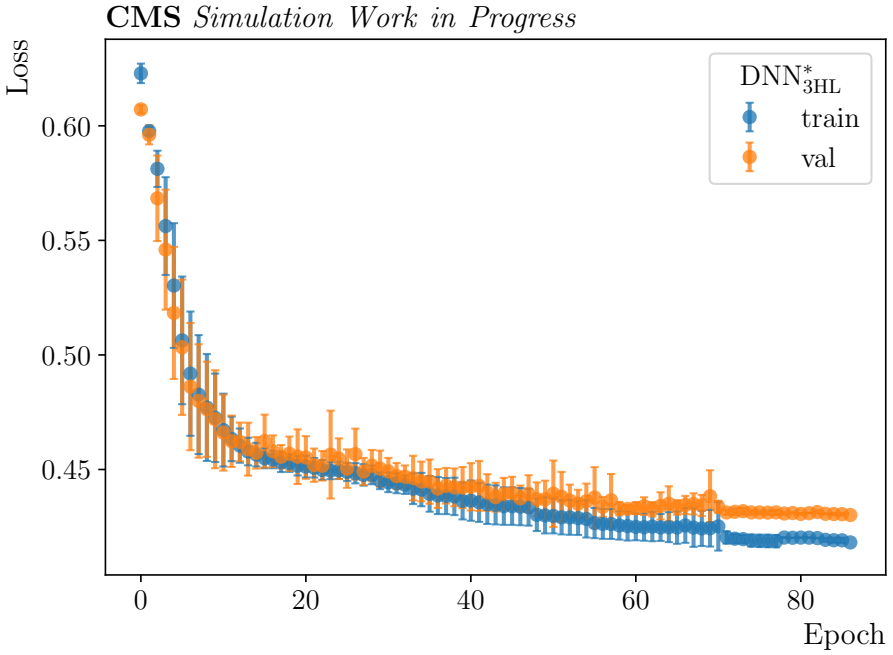


Figure G.2: **Mean performance ranking of DNN_{3HL}, including $\text{DNN}^{*}_{\text{2HL}}$.** One DNN_{3HL} is ranked higher than $\text{DNN}^{*}_{\text{2HL}}$ but the corresponding performance gain is negligible. Thus, in fact no further performance gain can be achieved by tuning the number of DOF. On the contrary, the spread of these DNNs appears to be several times larger than the spread of $\text{DNN}^{*}_{\text{2HL}}$. Data points with unfilled markers indicate that at least one model of ten repetitions of the particular training has a performance fulfilling the outlier criteria defined in Section 4.3 and the identified outlier(s) is/are therefore discarded. All data points possess an error bar representing the performance spread (square root of the unbiased sample variance), which is calculated on the basis of the ROC-AUC values of all not discarded models (at most ten models, one for each of the ten realized repetitions of a training).



(a)



(b)

Figure G.3: **Mean loss curves of $\mathrm{DNN}^{*}_{\mathbf{2HL}}$ and the highest ranked DNN in Fig. G.2 ($\mathbf{DNN}^{*}_{\mathbf{3HL}}$).** The loss on the train and validation set in dependence of the epoch and averaged over ten repetitions is showcased for both models. The error bars represent the spread (square root of the unbiased sample variance) calculated on the basis of the train and validation loss obtained for each epoch in the ten realized repetitions of the trainings.

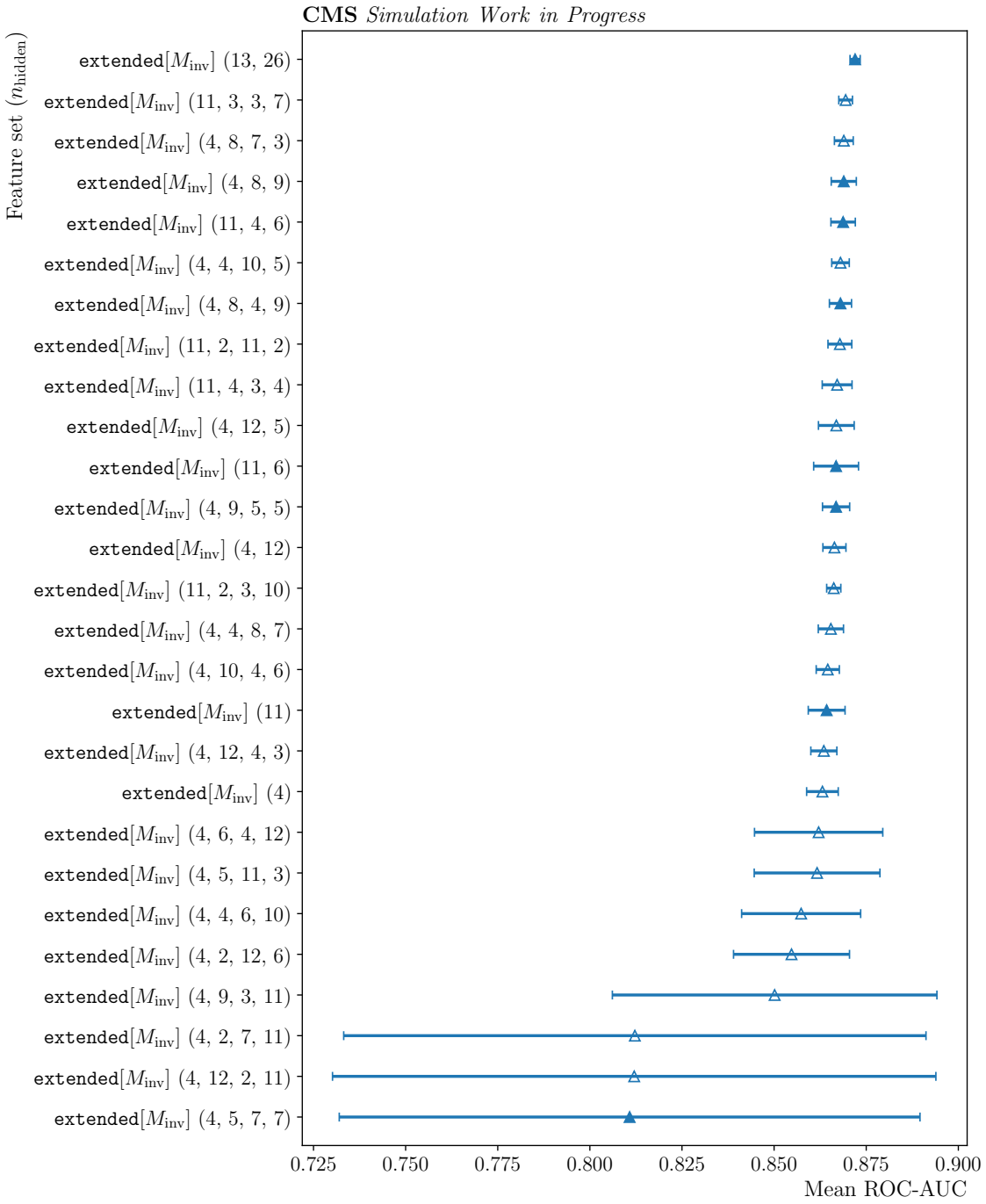


Figure G.4: **Mean performance ranking of DNNs with a restricted number of DOF, including $\text{DNN}^{*}_{\text{2HL}}$.** None of the DNNs with a restricted number of DOF outperforms $\text{DNN}^{*}_{\text{2HL}}$. Data points with unfilled markers indicate that at least one model of ten repetitions of the particular training has a performance fulfilling the outlier criteria defined in Section 4.3 and the identified outlier(s) is/are therefore discarded. All data points possess an error bar representing the performance spread (square root of the unbiased sample variance), which is calculated on the basis of the ROC-AUC values of all not discarded models (at most ten models, one for each of the ten realized repetitions of a training).

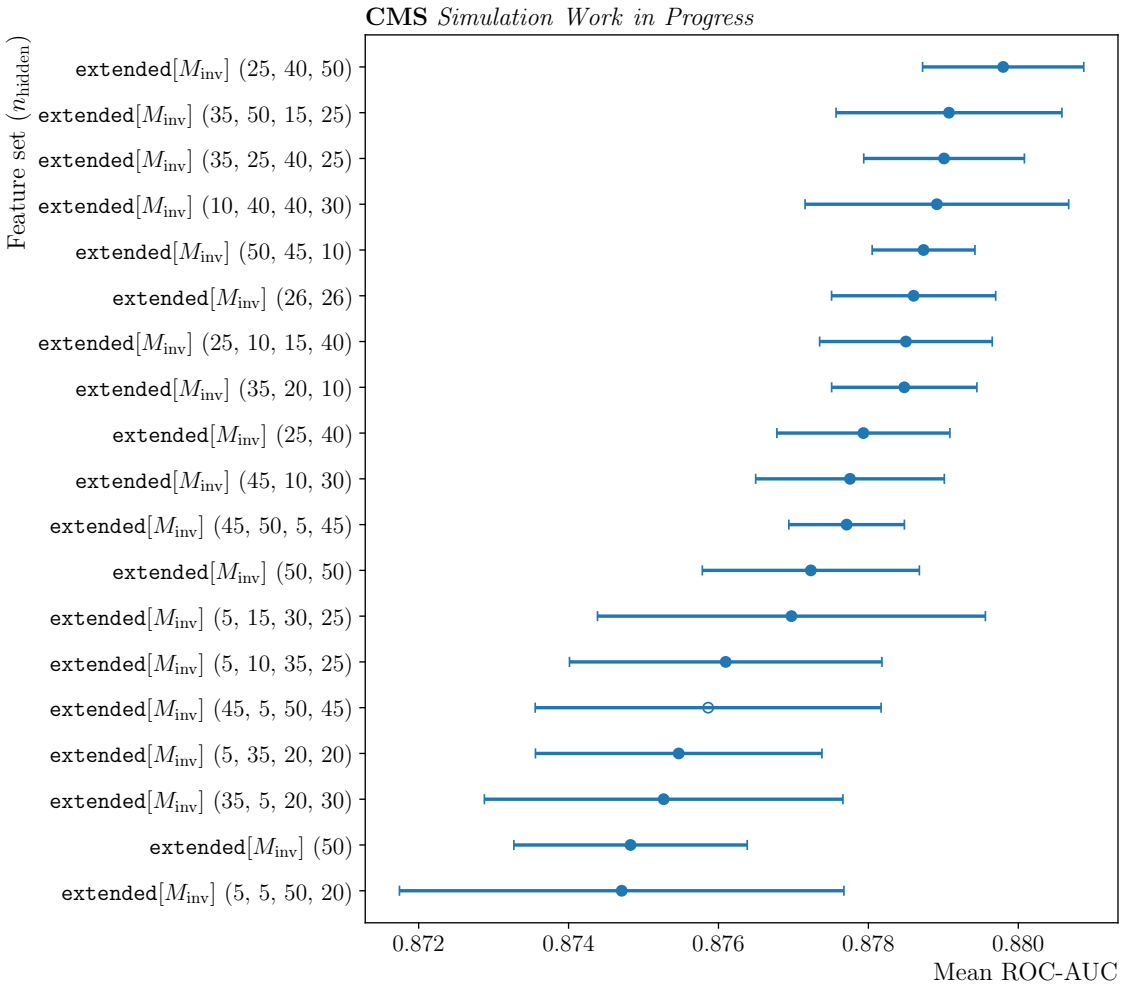


Figure G.5: **Mean performance of GNNs with an expanded number of DOF, including $\text{GNN}^{*}_{\text{2HL}}$.** Only five GNNs with an expanded number of DOF outperform $\text{GNN}^{*}_{\text{2HL}}$. Data points with unfilled markers indicate that at least one model of ten repetitions of the particular training has a performance fulfilling the outlier criteria defined in Section 4.3 and the identified outlier(s) is/are therefore discarded. All data points possess an error bar representing the performance spread (square root of the unbiased sample variance), which is calculated on the basis of the ROC-AUC values of all not discarded models (at most ten models, one for each of the ten realized repetitions of a training).

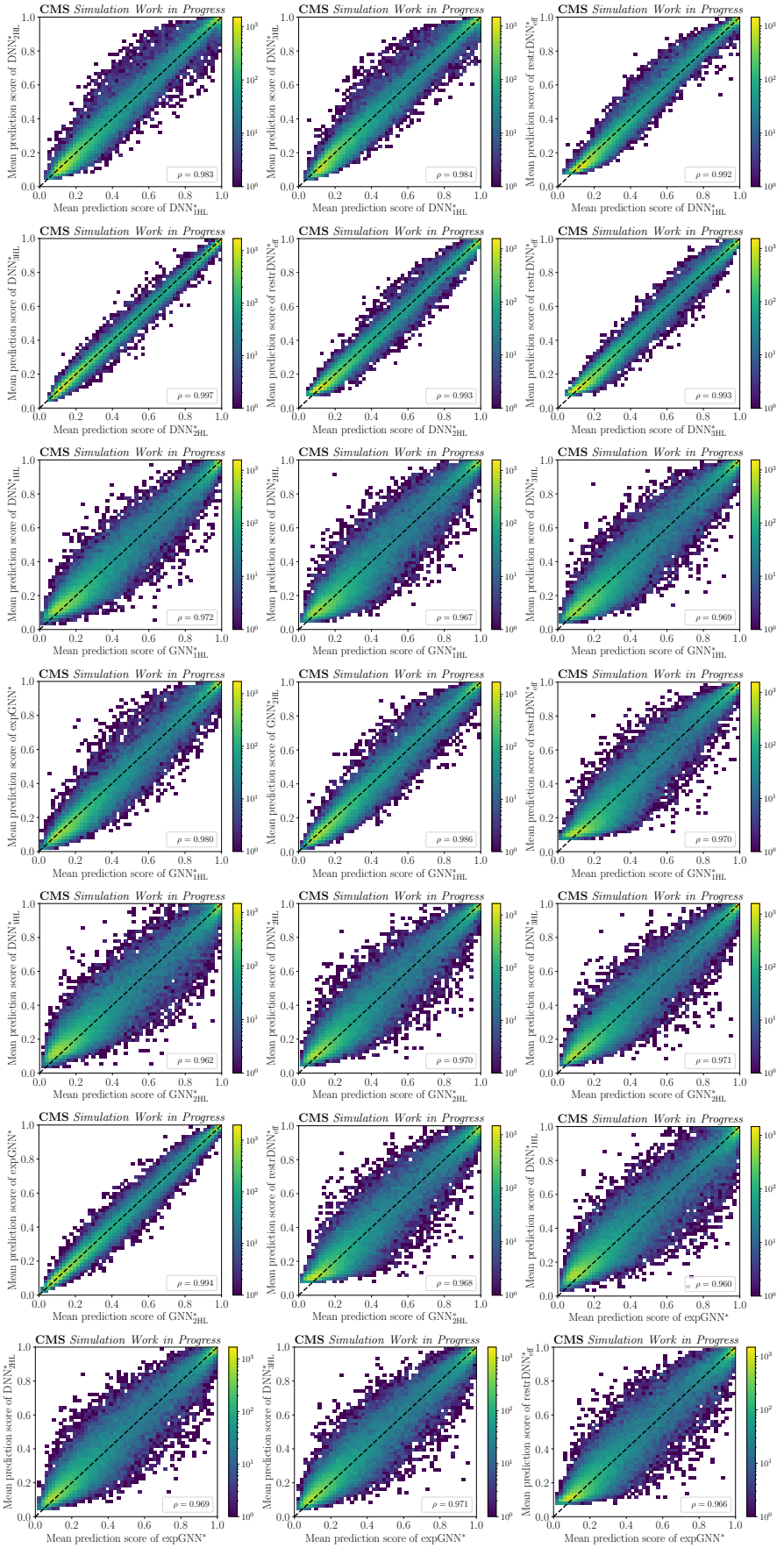


Figure G.6: **Comparison of the mean prediction scores of all best-performing models (excluding restrDNN*) evaluated on the test set.** The two-dimensional histogram of models of the same NN type possess a correlation coefficient ρ closer to one and GNNs appear to be more secure in classifying non-$t\bar{t} + b\bar{b}$ samples as their mean prediction scores for such events are closer to zero than DNNs. The dashed line shows the angle bisector.

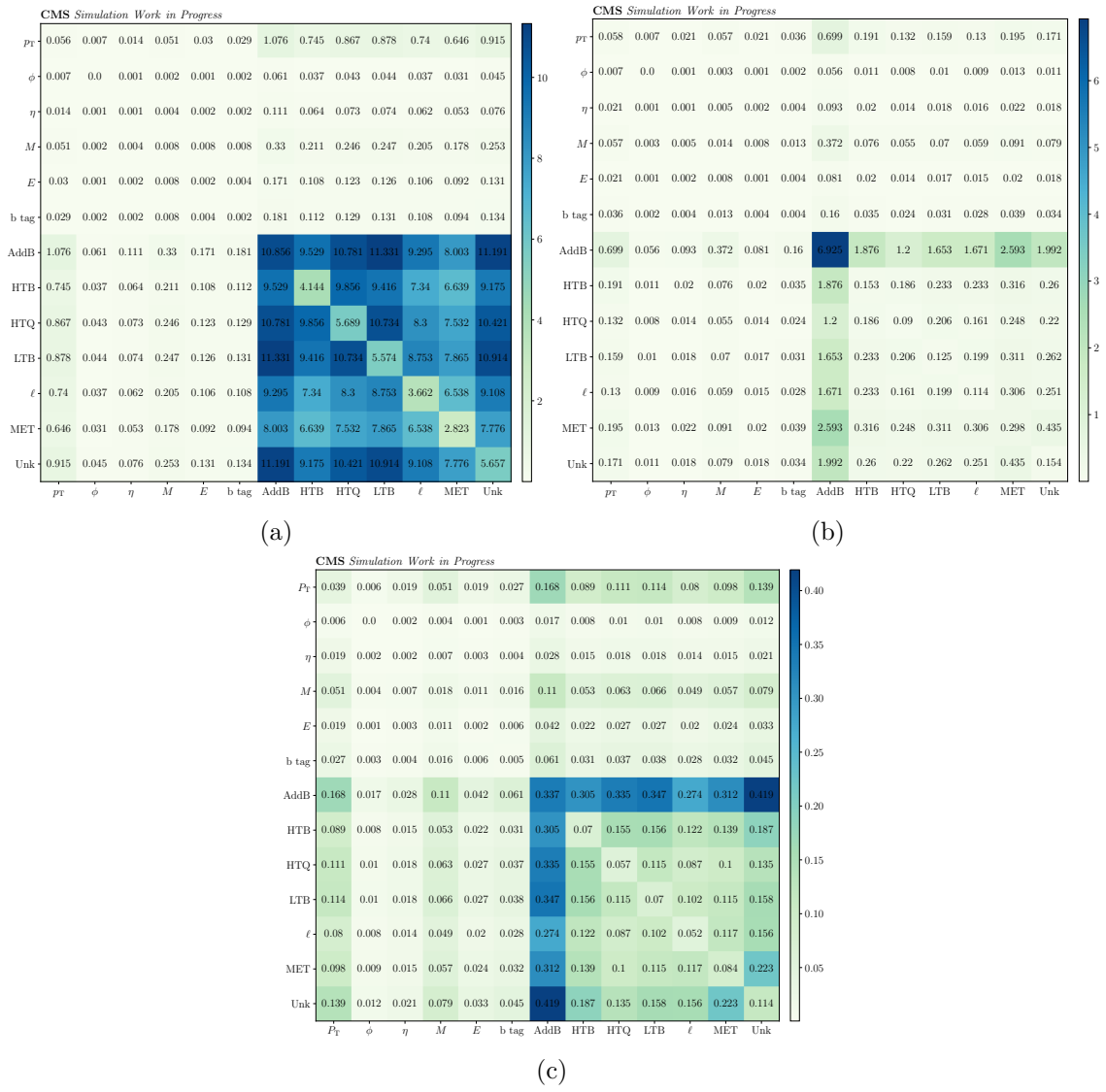


Figure G.7: **Second-order Taylor coefficients of the best-performing GNNs.** It can be observed that $\text{GNN}^*_{\text{1HL}}$ (a) focuses on all correlations between category flags and does not discover yet that it is sufficient to rely on correlations with the AddB flag as both $\text{GNN}^*_{\text{2HL}}$ (b) and expGNN* (c) have learnt to do.

Figure G.8: **Second-order Taylor coefficients of the best-performing DNNs.** Like for first-order Taylor coefficients of DNNs, only the global features, calculated without considering padded values, are taken into consideration. All DNNs ((a) $\text{DNN}^*_{\text{1HL}}$, (b) $\text{DNN}^*_{\text{2HL}}$, (c) $\text{DNN}^*_{\text{3HL}}$, (d) restrDNN^*, (e) $\text{restrDNN}^*_{\text{eff}}$) under investigation basically do not focus on any other correlations between the global features except M_{inv} with itself.