Computing and Software for Big Science (2024) 8:13
https://doi.org/10.1007/s41781-024-00122-3

REVIEW

A Case Study of Sending Graph Neural Networks Back to the Test Bench for Applications in High-Energy Particle Physics

Emanuel Pfeffer[1] · Michael Waßmer[1] · Yee-Ying Cung[1] · Roger Wolf[1] · Ulrich Husemann[1]

Received: 26 February 2024 / Accepted: 27 June 2024

Abstract
In high-energy particle collisions, the primary collision products usually decay further resulting in tree-like, hierarchical structures with a priori unknown multiplicity. At the stable-particle level all decay products of a collision form permutation invariant sets of final state objects. The analogy to mathematical graphs gives rise to the idea that graph neural networks (GNNs), which naturally resemble these properties, should be best-suited to address many tasks related to high-energy particle physics. In this paper we describe a benchmark test of a typical GNN against neural networks of the well-established deep fully connected feed-forward architecture. We aim at performing this comparison maximally unbiased in terms of nodes, hidden layers, or trainable parameters of the neural networks under study. As physics case we use the classification of the final state X produced in association with top quark–antiquark pairs in proton–proton collisions at the Large Hadron Collider at CERN, where X stands for a bottom quark–antiquark pair produced either non-resonantly or through the decay of an intermediately produced Z or Higgs boson.

Keywords Graph neural networks · Deep neural networks · High-energy particle physics · LHC

Introduction

The continuous rise and flourish of deep learning has significantly impacted also the community of high-energy particle physics, where modern algorithms of deep learning—mostly in the form of various neural network (NN) architectures—find applications as automation tools, for (multiclass) classification, parameter regression, or universal function approximation. The Large Hadron Collider (LHC) at CERN offers a unique test environment for such algorithms providing a large amount of independent identically distributed (i.i.d.) data from proton–proton (pp) collisions under well controlled laboratory conditions. These data feature a rich hierarchical structure, optimally suited for the application of all kinds of general methods of statistical data analysis. Moreover, the underlying physics laws and statistical models, which have emerged over many decades of research, are scrutinized to a level that allows the reliable estimation of particle properties with a relative accuracy ranging far below the per-mille level, in rare cases even below 10^{-10} [1]. This circumstance offers a toolbox for generating a large amount of perfectly known, complex, synthetic data, with a high relation to experimental observations, through the application of Monte Carlo (MC) methods [2, 3]. These data are usually obtained as samples from an intractable though well-known likelihood function $\mathcal{L}$. This setup provides a unique opportunity to thoroughly benchmark any kind of machine learning (ML) algorithms under complex, real-life laboratory conditions.

At the LHC, data analysts strive for the application of more and more sophisticated ML-models with more and more not further processed—and in this sense "raw"—input data. This strategy is fed, among others, by the belief that automated algorithms might find ways of extracting information of interest to the analyst, which are superior to selection

✉ Emanuel Pfeffer
emanuel.pfeffer@kit.edu

Michael Waßmer
michael.wassmer@kit.edu

Yee-Ying Cung
yeeying.cung@web.de

Roger Wolf
roger.wolf@kit.edu

Ulrich Husemann
ulrich.husemann@kit.edu

[1] Karlsruhe Institute of Technology, Institute of Experimental Particle Physics, Karlsruhe, Germany

Published online: 12 July 2024

strategies that are vulnerable to the bias of human prejudice, and at the same time are capable of taking correlations of observables in a high-dimensional feature space into account. On the other hand, ML-algorithms should not be forced to learn already known and well-established physics principles, like symmetries inherent to the presented task. While such information can only be insufficiently passed through the necessarily finite training samples, it can be intrinsically incorporated either into the loss functions used for training, or in the NN architectures.

At the large multi-purpose LHC experiments, ATLAS [4] and CMS [5], pp collisions at a center-of-mass energy of, e.g., 13 TeV result in the creation of thousands of collision products to be recorded by the experiments. Primary collision products might decay further resulting in tree-like, hierarchical structures with a priori unknown multiplicity. The collision process can be described in a factorized approach:

During the hard scattering process, the fundamental constituents of the protons, i.e., the quarks and gluons which are also collectively referred to as partons, interact via the fundamental interactions under investigation. We refer to the result of these interactions as the partonic final state. It cannot be observed directly in an experiment. Rather, each parton undergoes a series of theoretically well-known processes, setting in at lower energy scales, resulting in stable particles. The inverse problem usually subject of high-energy particle physics is to infer the presence and properties of the stable particles and eventually the partonic final state from their observable energy deposits in the detectors.

At the stable particle level all decay products of a collision form permutation invariant sets of final state objects, which may emerge from the collision in the form of collimated particle jets [6], forming well-suited proxies for strongly interacting final state partons, or individual, spatially isolated particles, like leptons. From the preparation of the collision's initial state and energy and momentum conservation, physicists may infer the presence of non- or weakly interacting particles, like neutrinos, in the collision's final state, through the principle of missing transverse energy (MET) [7]. A natural representation of this richly structured data is in the form of mathematical graphs $\mathcal{G}$, which are indeed also the basis of theoretical amplitude calculations of the quantum-mechanical wave function in the form of Feynman graphs [8].

Within the high-energy particle physics community, this observation has led to an increased interest in NNs based on mathematical graphs (GNNs) [9–13], where nodes are usually identified by particles and edges potentially by relations across particles. A comprehensive review can be found in Ref. [14]. In this paper, we describe a comparison of typical GNN architectures with NN models based on the deep fully connected feed-forward architecture (DNN), which has been studied intensely, in the past. The presented studies are performed in a complex environment representative for typical high-energy physics tasks. The goal of the studies is to reveal the underlying work mechanisms that might give advantage to GNNs over DNNs, if any, based on comparisons that are to best effort unbiased in terms of expressiveness and information provided to the NN models to solve a given task.

In the "Neural Network Task" section, we give an introduction to the task that serves as benchmark for this comparison. In the "Neural Networks Under Study" section, the architectures and training setups of the NNs under study are described. In the "Results" section, we present the results of the comparison. We conclude the paper with a summary in the "Summary" section.

Neural Network Task

Physics Processes

As benchmark for the comparison we use the classification of the final state X produced in association with top (t) quark–antiquark pairs ($t\bar{t}$) in pp collisions at the LHC, where X stands for a bottom (b) quark–antiquark pair ($b\bar{b}$) produced either through non-resonant gluon exchange or through the decay of a massive Z or Higgs (H) boson as intermediate particle, as discussed, e.g., in Ref. [15]. Under realistic conditions, the collision of interest might be overlaid by several tens of additional collisions, referred to as pileup. The complete detectable final state of a collision of interest, including pileup, is referred to as an event, whose feature vector $\mathbf{x}$ would be presented to the NN. An arbitrary number of such events may be generated synthetically by evaluating $\mathcal{L}$ of the full process via the MC method. In this study, we focus on the classification of the underlying hard process neglecting the effects of pileup.

The interest in the chosen classification task arose from studies of H production in association with $t\bar{t}$ in the subsequent $H \rightarrow b\bar{b}$ decay (ttH(bb)), for which $t\bar{t}$ associated Z boson production (ttZ) in the $Z \rightarrow b\bar{b}$ decay channel (ttZ(bb)) and non-resonant $b\bar{b}$ production in association with $t\bar{t}$ (ttbb) are important background processes. Exemplary Feynman diagrams of these processes in leading-order (LO) of perturbation theory are shown in Fig. 1. The decay products, resembled by the outgoing lines in the diagrams, shown in Fig. 1, represent the partonic final state of interest to this study, which is the same for all processes. Therefore, the processes can only be distinguished by the kinematic properties of the particles, in particular b quarks. This situation is complicated by the fact that the t quark also decays into b quarks radiating a quasi-real W boson with a branching fraction of nearly 100% [16]. The W boson subsequently decays either into quarks, which further on form jets in the

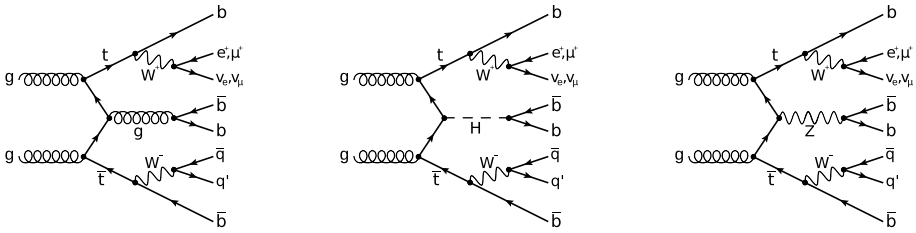

Fig. 1 Exemplary Feynman diagrams for the processes of interest to this study: (left) ttbb, (middle) ttH(bb), and (right) ttZ(bb)

detector, or leptons. For the presented study the semi-leptonic $t\bar{t}$ final state has been chosen, in which the W boson of one t decays into an electron or muon, further on referred to as ℓ, and a corresponding neutrino ν_ℓ. The other W boson decays into quarks. Due to the radiation of additional gluons and the splitting of gluons into quark–antiquark pairs additional colored particles and consequently jets might emerge from the process. This constellation implies a richly structured final state of an event with at least four b quark- and two predominantly light-quark-induced jets; an ℓ, which is spatially isolated from any other activity originating from the hard scattering process in the detector; and MET, due to the emitted ν_ℓ. The b quark-induced jets, referred to as b jets in the following, may be identified experimentally with a finite purity and efficiency, as, e.g., described in Ref. [17]; the methods of how to achieve this are not subject of this paper.

Sample Preparation

Samples for all processes in question have been generated synthetically from a corresponding likelihood $\mathcal{L}$ using the MC technique. The tools used for event generation are the matrix-element generator MadGraph5_aMC@NLO [18, 19] in version 2.9.9 interfaced with the Pythia event generator [20] in version 8.306 to map the partonic final state to the stable-particle level. All processes in consideration have been generated at LO in perturbative quantum chromodynamics (QCD), in the four-flavor scheme. The same setup has been used for the generation of all samples to avoid spurious differences due to the use of different generation tools.

All generated events have been passed through a simplified simulation of the CMS detector as configured during the LHC Run-2 data-taking period in the years 2016–2018, using the DELPHES simulation package [21]. For this study only reconstructed leptons, jets, and MET are considered. All detected and reconstructed final-state objects have been selected to fulfill a set of selection criteria typically used for the analysis of data collected by the CMS experiment, as summarized in Table 1. These selection criteria comprise the following observables:

- The transverse momentum p_T and pseudorapidity η of the reconstructed ℓ and jets.

Table 1 Selection requirements on the reconstructed final-state objects

Object	p_T (GeV)	$\lvert\eta\rvert$	$I_\mathrm{rel}^{\Delta R}$	β
Electron	≥ 25	< 2.5	< 0.12 ($\Delta R = 0.3$)	–
Muon	≥ 25	< 2.4	< 0.25 ($\Delta R = 0.4$)	–
Jet (anti-k_T, $R_0 = 0.4$ [22])	≥ 20	< 2.4	–	–
b jet	≥ 20	< 2.4	–	$\beta \geq 2$

The quantity $I_\mathrm{rel}^{\Delta R}$ corresponds to the scalar sum of energy deposits detected within the radius ΔR around ℓ in η-ϕ divided by the p_T of ℓ, as defined in the text. A lower value of $I_\mathrm{rel}^{\Delta R}$ implies less activity in the spacial vicinity of ℓ, indicating that ℓ originates from $\mathrm{W} \to \ell\nu_\ell$. The variable β refers to the working points of a specific b jet identification algorithm, as described in the text

- A variable $I_\mathrm{rel}^{\Delta R}$ that corresponds to the scalar sum of energy deposits E_i detected within the radius $\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2}$ around ℓ, divided by the p_T of ℓ, where $\Delta\eta$ refers to the difference between E_i and ℓ in η and $\Delta\phi$ to the corresponding difference in azimuthal angle ϕ, based on the coordinate system deployed by CMS [5]. For ℓ originating from $\mathrm{W} \to \ell\nu_\ell$ low values of $I_\mathrm{rel}^{\Delta R}$ are expected.
- The output of a specific b jet identification algorithm represented by the discrete observable $\beta \in \{0,1,2,3\}$ indicating whether an object has been identified as a b jet under a specific working point α ($\beta \geq \alpha$). The value of α represents ($\alpha = 1$) loose, ($\alpha = 2$) medium, and ($\alpha = 3$) tight selection criteria, corresponding to a rate of light quark or gluon jets (excluding c jets) wrongly identified as a b jet (false-positive rate), of approximately 10%, 1%, and 0.1% for b jet tagging efficiencies of 80–90%, 60–75%, and 40–60%, respectively.

All events have been selected to exhibit at least six and not more than eight jets, at least four of which are assumed to be correctly identified as b jets according to $\beta \geq 2$, and exactly one ℓ, matching all selection criteria. The selection of one ℓ and six jets, of which at least four are identified as b jets, is motivated by the partonic final state under study, as depicted in Fig. 1. The selection of up to two additional

Table 2 Requirements for the jet-class definition, according to the matching to the partonic final state, where b_{add} corresponds to a b quark not originating from a t decay, $b_{t_{had}}$ ($b_{t_{lep}}$) to a b quark originating from $t \to bW(qq')$ ($t \to bW(\ell\nu_\ell)$), and $q_{W_{had}}$ to a quark originating from $W \to qq'$

Class label	Assignment	Description
ADDB	$\Delta R(\text{jet}, b_{add}) < 0.4$	b jets not from t decays
HTB	$\Delta R(\text{jet}, b_{t_{had}}) < 0.4$[a]	b jet from t_{had}
LTB	$\Delta R(\text{jet}, b_{t_{lep}}) < 0.4$[a]	b jet from t_{lep}
HTQ	$\Delta R(\text{jet}, q_{W_{had}}) < 0.4$[a]	q jet from W
NA	No match	Additional jet

We note that the classes ADDB and HTQ comprise two jets. Jets not assigned to any other class are assigned to the NA class and sorted by decreasing p_T. If no jet is found to fulfill the corresponding ΔR criterion, the leading jet from the NA class is re-assigned to the classes HTB, HTL, HTQ, in that order

[a] If no assignment by ΔR is found, the leading jet from NA is assigned

jets, increases the chance that the complete partonic final state can be matched to the selected jets.

For each reconstructed jet the attempt is made to assign the initiating particle of the partonic final state, based on the distance ΔR between the jet and the corresponding parton. Only partonic final state objects with transverse momentum of $p_T > 20\,\text{GeV}$ and $|\eta| < 2.4$ are considered for this assignment. From the assignment five mutually exclusive jet classes are build, referring to the (ADDB) b quarks not originating from any t decay; the b quark originating from the (HTB) hadronic and (LTB) leptonic t decays, (HTQ) quarks originating from the hadronic W decay; and (NA) jets not assigned to the partonic final state.

The assignment of the partonic final state to the reconstructed jets may be incomplete, for a given event. Events for which no or only one jet is assigned to the ADDB class are discarded from the training and test samples. In all other cases, if the assignment by ΔR did not result in one jet of class LTB, one jet of class HTB, and two jets of class HTQ, the remaining not associated jets of class NA are ordered by decreasing p_T and the leading jets in p_T are re-assigned to these classes in the order of LTB, HTB, HTQ. A summary of all jet classes is given in Table 2

Task Definition

The NN models are supposed to perform a classification task, in which a given event should be assigned either to ttbb, ttH(bb), or ttZ(bb). The physics process from which the event was generated constitutes the ground truth information for this task.

To simplify the task, the reconstructed jets are assumed to perfectly match the initiating partons, whenever possible, through the parton association algorithm, as described in the "Sample Preparation" section. We follow this path to give the NNs optimal conditions, in this respect, to fulfill the task. We note that this information has a significant impact on the success of each algorithm to fulfill the task, and anticipate that in a realistic physics-driven application this ideal parton association would be replaced, e.g., by another, potentially also ML-estimated, output of a dedicated parton association algorithm with lower than ideal accuracy. The ADDB class contains jets which stem directly from the intermediate particle to distinguish the processes under study. The features of these jets are assumed to provide the most decisive contribution to the classification.

Training Setup

All NN models are subject to a supervised training. The generated samples used for this are split into a training, validation, and test sample, containing 60, 20, and 20% of the generated events, respectively. Event numbers, split by process and sample, are given in Table 3.

Each training is performed for an ensemble of ten statistically independent repetitions to obtain a rough estimate of the statistical spread of the trained models, due to random choices in the training setup. Performance measures of each model are reported as sample means μ, of which the corresponding uncertainty $\Delta\mu$ is estimated from the square root of the sample variance. All repetitions are based on the same training, validation, and test samples. Due to the large number of events in these samples, randomization through data shuffling is assumed not to change the conclusions of the studies significantly.

Each training is performed on CPUs through a distributed computing infrastructure, where each training is performed on a dedicated CPU. We ensure that each step affected by random choices is based on a different random seed. The library used to build the GNNs is PyTorch Geometric v2.0.3 [23] based on the PyTorch library [24]. The same library is used, to construct the corresponding DNNs under study. Further parameter choices of the training setups are given in the upper part of Table 4. They are the same for both NN architectures.

Table 3 Numbers of events for each process, in the training, validation, and test samples

Process	Training	Validation	Test	Sum
ttbb	41,650	13,883	13,803	69,336
ttH(bb)	99,695	33,329	33,229	166,253
ttZ(bb)	88,107	29,272	29,453	146,832

Table 4 Common parameter choices for the NN architectures under study and their training setup

Parameter	Setting(s)
Loss function	Binary cross-entropy
Optimizer	Adam [25] ($\gamma = 0.01$)
Mini-batch size	200
Maximum number of epochs	200
Early-stopping	Δepochs = 15, Δloss = 0.001
Use of weights and biases	Yes
Number of outputs	1 (binary)
Activation function (for hidden layers)	ReLu
Activation function (for output layer)	Sigmoid

Neural Networks Under Study

Architectures

For the GNN models, a graph representation of the final state is obtained from the reconstructed jets, ℓ, and MET in an event. Each of these objects is represented by a node i in $\mathcal{G}$. This set of nodes may be complemented by nodes for up to two more jets in the selection. Accordingly, $\mathcal{G}$ has 8–10 nodes. For each i, a vector $\mathbf{a}_i$ of attributes with $\dim(\mathbf{a}_i) = n$, $\forall i$ is defined, forming the embedding space. At initialization time, the $\mathbf{a}_i$ are initialized by the feature vectors $\mathbf{x}_i$ with $\dim(\mathbf{x}_i) = n_{\text{feat}}$, $\forall i$. The following studies also comprise configurations with $n > n_{\text{feat}}$. In these cases, attributes in the $\mathbf{a}_i$ without correspondence in $\mathbf{x}_i$ are set to zero (zero padding), at initialization time.

All nodes are connected with edges, resulting in a fully connected non-directed graph without self-loops. Relational information between nodes i and j may be assigned, in the form of edge weights ω_{ij}. For this purpose three physics motivated choices are made: (i) the invariant mass m; (ii) the distance ΔR; and (iii) the reciprocal distance ΔR^{-1} of the connected final-state objects. In addition, the cases of a (one) constant, (rnd) random, and (zero) no edge connection at all are studied, resulting in a total of six variants of edge connections. An illustration of a resulting graph for an event with eight nodes and no additional jets is given in Fig. 2.

The GNN algorithm to process the graph data is based on the layered GraphConv operation, as introduced and described in Ref. [26], using the sum over all i as aggregation function. After initialization, k GraphConv operations are applied, after which the resulting $\mathbf{a}_i$ are transformed into a single vector of length n, averaging over all i. A linear combination of the components of this vector, which is scaled to values between 1 (indicating ttbb as signal) and 0, for binary classification, eventually forms the output $\hat{y}$ of the GNN. A graphical illustration of this model is given in the lower part of Fig. 2.

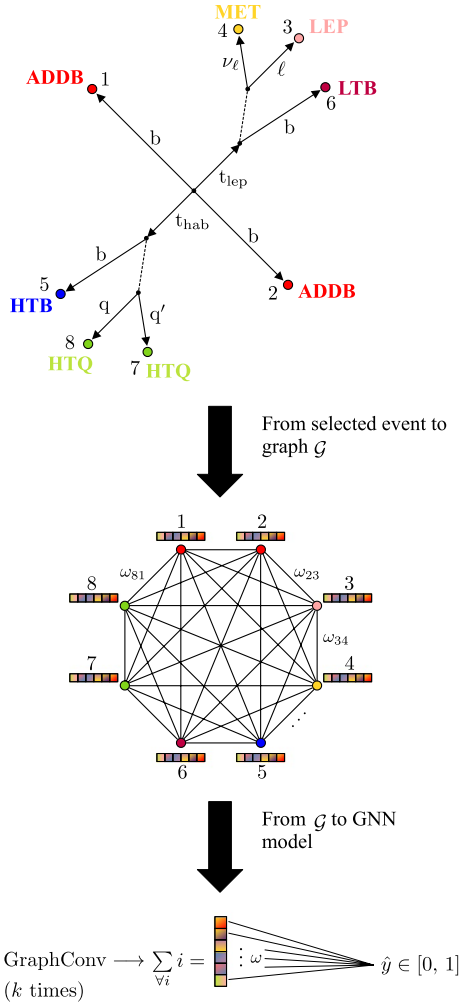

Fig. 2 Translation of an (upper part) selected event (without jets in the NA jet-class in this case) into a (middle part) graph $\mathcal{G}$ and finally into the (lower part) GNN model. For the indication of the partonic final state we do not distinguish particles from anti-particles. The individual object-classes are indicated by different colors. The nodes of $\mathcal{G}$ are labeled by $i = 1 \ldots 8$ and colored the same way as the object-classes. The boxes next to the nodes indicate the embedding space of the GNN model. The GNN output is indicated by $\hat{y}$

In a first study, the GNN are compared with corresponding DNN models with k hidden layers, containing n hidden nodes, each. The values of k, n, and the choice of weights, steering the exploitation of relational information by the GNN models are varied, resulting in 36 variants of parameter choices, as summarized in Table 5. We note that here, as in the following, n represents a tuple of length k. For a GNN this tuple indicates the dimension of the embedding space per GraphConv operation; for a DNN it represents the number of nodes per hidden layer. Other parameter choices related to the NN architectures under study or the setup of the NN training are made common and summarized in Table 4. Special care is taken to

Table 5 Parameters varied for the comparison of GNN with corresponding DNN models, where n corresponds to the dimension of the embedding space during a GraphConv operation (number of nodes in a hidden layer) and k to the number of GraphConv operations (hidden layers) in the GNN (DNN) case

Parameter	Setting(s)
k	1, 2
n	13, 26, 39
Edge weights (ω_{ij})	m, ΔR, ΔR^{-1}, one, rnd, zero

The choices of n are motivated by the size of the input vector n_{feat} to the GNN, as described in the "Presentation of the Feature Space" section. The choices of one, rnd, and zero for the use of edge information in GNN models are compared to DNN models without relational information between individual objects. These DNN models are indicated by the label none, in corresponding figures

compare the GNN with the corresponding DNN models on the same footing, especially in terms of information about the feature space presented to them, as discussed in the following section.

Presentation of the Feature Space

Primary features passed to the NNs are the invariant mass (m), energy (E), η, and ϕ of each reconstructed final state object. The reconstructed final state objects comprise ℓ, MET, at least four b jets, two additional jets, all of which are associated with the partonic final state, and potentially two more non-associated jets from the event selection. The primary features are complemented by β. The selection requires values of $\beta \geq 2$ for identified b jets. For ℓ and MET β is set to zero. For MET p_{T} corresponds to the absolute value of MET and ϕ ranges from $-\pi$ to π depending on the MET direction in the r-ϕ plane; the features E and m are set to zero.

We note that both NN architectures may profit from additional information, which is not passed explicitly through $\mathbf{x}_i$, but implicitly through the way the features are presented to the NNs. An obvious difference between the NN architectures arises from the fact that the GNN naturally supports processing of events with arbitrary object multiplicities. The $\mathbf{x}_i$ are transferred to the GNN through the $\mathbf{a}_i$, during initialization. During the GNN processing the information per i is aggregated over all nodes. In the DNN case such an aggregation step is absent. Instead, the $\mathbf{x}_i$ are concatenated into an enlarged feature vector $\mathbf{x}^{\text{DNN}}$ of length $10 \times n_{\text{feat}}$, comprising the $\mathbf{x}_i$ of the eight reconstructed objects, of which all jets have been associated to the partonic final state, plus potentially two additional selected jets. The order in which these objects are concatenated has been chosen to follow the association to the partonic final state. It is given by ADDB(1), ADDB(2), HTB, HTQ(1), HTQ(2), LTB, LEP, MET, NA(1), NA(2). In cases with more than one jet of a given class, e.g., like the ADDB class, the jets are sorted by their p_{T}. For events that contain fewer than two jets in addition to those that have been matched to the partonic final state, the corresponding entries in $\mathbf{x}^{\text{DNN}}$ are filled with zeros. We point out two subtleties, which are related to these choices:

One subtlety, in favor of the GNN, arises from incorporating relational information through ω_{ij}. This advantage is compensated for by appending equivalent information to $\mathbf{x}^{\text{DNN}}$. For up to ten selected objects in an event this results in up to 45 additional features. The choices of one, rnd, and zero for the use of edge information in GNN models are compared to DNN models without relational information between individual objects. These DNN models are indicated by the label none, in corresponding figures.

Another subtlety, related to the same fundamental difference, but this time in favor of the DNN, arises from the fact that through the concatenation of the $\mathbf{x}_i$ into $\mathbf{x}^{\text{DNN}}$, according to the association to the partonic final state, the DNN receives extra information through the positions of the $\mathbf{x}_i$ in $\mathbf{x}^{\text{DNN}}$ which is not accessible to the GNN. This advantage is compensated for by adding information about the association of the i to the partonic final state via one-hot encoding. For the five jet-classes defined in Table 2 plus one label (LEP) for ℓ and one label (MET) for MET this extension increases n_{feat} by seven, leading to the dimension of $\mathbf{x}_i$, $\forall i$ of $n_{\text{feat}} = 13$.

For $\mathbf{x}^{\text{DNN}}$ the $\mathbf{x}_i$ are concatenated for all i assuming two more jets in the NA class. For events with fewer than two jets in the NA class the foreseen features are initialized with zero. Together with the relational information between all potential objects in an event, this results in a dimension of $\mathbf{x}^{\text{DNN}}$ of 175 features, of which up to 47 features might potentially be filled with zeros.

In this configuration, the information about the association to the partonic final state is presented in the form of one-hot encoding to both NN architectures. To confirm to what extent the DNN may infer this information already from the position of the $\mathbf{x}_i$ within $\mathbf{x}^{\text{DNN}}$ we also investigate configurations of the DNNs without this information in the form of on-hot encoding, resulting in a reduced input vector $\mathbf{x}^{\text{DNN}}_{\text{red}}$ with dimension 105. All input features in use are listed in Table 6. All non-integer features are standardized to a mean of zero and a standard deviation of one.

Table 6 Input features used for the studies described in the text

Input feature	Continuous	Discrete	One-hot	Comment
m	✓	–	–	Primary features
E	✓	–	–	
p_{T}	✓	–	–	
ϕ	✓	–	–	
η	✓	–	–	
β	–	✓	–	
LEP	–	–	✓	Assoc. to ℓ
MET	–	–	✓	Assoc. to MET
ADDB	–	–	✓	Assoc. to additional b quarks
HTB	–	–	✓	Assoc. to $\mathrm{t}_{\mathrm{b_{had}}}$
LTB	–	–	✓	Assoc. to $\mathrm{t}_{\mathrm{b_{lep}}}$
HTQ	–	–	✓	Assoc. to $\mathrm{q}_{\mathrm{W_{had}}}$
NA	–	–	✓	Not associated

Columns 2–4 indicate whether a given feature is continuous, discrete, or presented via one-hot encoding. The given features form the feature vectors $\mathbf{x}_i$ per object i. For the DNNs the $\mathbf{x}_i$ for potentially ten selected objects are concatenated into an extended feature vector $\mathbf{x}^{\mathrm{DNN}}$, according to their association to the partonic final state; for events with fewer than ten objects the $\mathbf{x}_i$ of the missing objects are filled with zeros. In addition, relational information between all potential objects is added to $\mathbf{x}^{\mathrm{DNN}}$. In a reduced configuration, the one-hot encoded information about the association of the objects to the partonic final state is omitted to form a reduced input vector $\mathbf{x}^{\mathrm{DNN}}_{\mathrm{red}}$ to the DNNs

Results

To be able to draw fair conclusions from a comparison of different NN architectures (of potentially different complexity) special care has to be taken for this comparison to be based on the same ground. For this study we have focused on a common choice of non-tunable (hyper-) parameters, which are not subject to the NN training, as well as on an equal level of information primarily passed to the NNs, through training conditions and input features.

An inevitable difference remains in the organization and layering of hidden nodes, which when kept similar, may well lead to a different number of TPs and therefore a priori different expressiveness of the NN models. Vice versa, keeping the number of TPs similar, implies differences in the layering and organization of hidden nodes. Since differences of one or the other kind may not be overcome, both configurations, (i) similar layering of hidden nodes; and (ii) similar number of TPs, are studied. In any case, the enlarged size of $\mathbf{x}^{\mathrm{DNN}}$ ($\mathbf{x}^{\mathrm{DNN}}_{\mathrm{red}}$) with respect to $\mathbf{x}$ will give higher emphasis to the first DNN layer compared to the corresponding GNN architecture. In addition, the DNN architecture features the less complex pre-processing of the inputs, since it does not imply the creation of graphs. On the other hand, the potentially more constrained GNN may have advantages over the DNN architecture in terms of convergence properties of the training.

Comparison of Neural Networks with Similar Layering of Hidden Nodes

Neural Networks with One Layer of Hidden Nodes

A first comparison of GNN with corresponding DNN models, based on a similar layering of hidden nodes, is shown in Fig. 3.

The metric by which to judge the success of an NN to fulfill the task is chosen to be the mean of the ROC-AUC μ_{AUC} based on the training setup, as described in the "Training Setup" section. The results are presented for the variation of parameters as summarized in Table 5. For the presentation in Fig. 3, a simple architecture with one hidden layer for the DNN models and one GraphConv operation of the GNN models ($k = 1$) has been chosen. For the GNN architecture this implies that there is only one exchange of information across adjacent nodes of $\mathcal{G}$, i.e., each node receives information only of its nearest and not the next to nearest neighbor in $\mathcal{G}$. Since $\mathcal{G}$ has been chosen to be fully connected, there is no strict suppression of information that way, in the sense that each node i receives information from any other node $j \neq i$ in $\mathcal{G}$. The input features presented to each corresponding NN architecture are chosen, as described in the "Presentation of the Feature Space" section and summarized in Table 6.

The results for the GNN models are represented by circles, the results for the DNN models with $\mathbf{x}^{\mathrm{DNN}}$ ($\mathbf{x}^{\mathrm{DNN}}_{\mathrm{red}}$) as input vector by upward (downward) pointing triangles. The bars associated with the points indicate the uncertainty

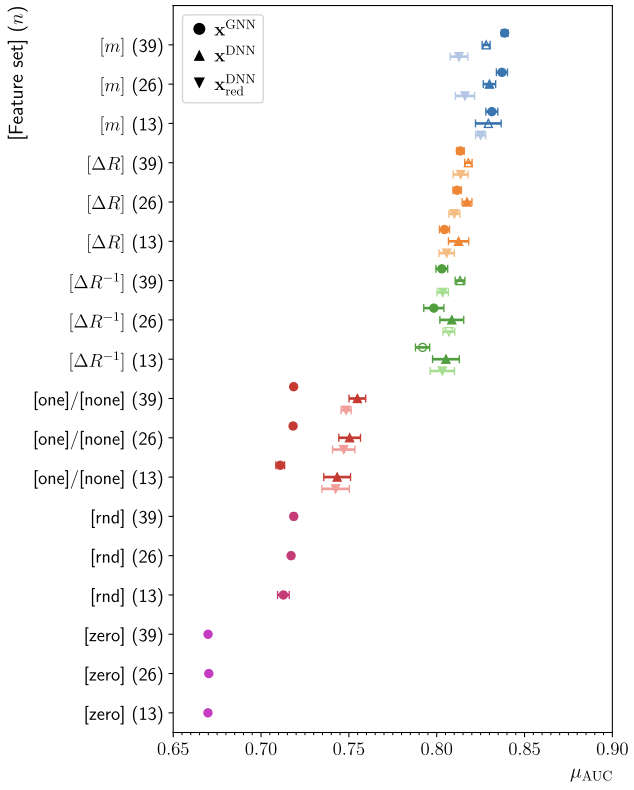

Fig. 3 Mean ROC-AUC μ_{AUC} as obtained for 18 different configurations of GNN and 24 corresponding configurations of DNN models with $k = 1$. The labels in brackets on the vertical axis indicate the use of relational information, as discussed in the "Presentation of the Feature Space" section, the numbers in parentheses correspond to the choices of n. The circles refer to GNN and the upward (downward) pointing triangles to DNN models with a default (reduced) set of input features $\mathbf{x}^{\mathrm{DNN}}$ ($\mathbf{x}^{\mathrm{DNN}}_{\mathrm{red}}$), as discussed in the "Presentation of the Feature Space" section. For better readability, markers of the same configuration are shifted vertically along the y-axis. The bars are obtained from the sample variance of an ensemble, as described in the "Training Setup" section. Those NN architectures which belong to the same choices of varying parameters are spatially grouped and shown in the same color. Open markers indicate that significant outliers of the corresponding distribution of ROC-AUC values have been removed from the calculation of μ_{AUC} and its variance, as described in the text

$\Delta\mu_{\mathrm{AUC}}$ in μ_{AUC} due to random choices in the training, as described in the "Training Setup" section. Open markers indicate that at least one training repetition in an ensemble has been removed as an outlier from the calculations of μ_{AUC} and $\Delta\mu_{\mathrm{AUC}}$. Candidates for outliers have been identified by their μ_{AUC} values exceeding $1.5\,\Delta\mu_{\mathrm{AUC}}$, as obtained from the full ensemble. An outlier candidate has been definitely removed and the ensemble size reduced accordingly, if doing so changed $\Delta\mu_{\mathrm{AUC}}$ by a value of at least 0.0025. Following this procedure, 54 outliers have been removed from a total of 1840, which corresponds to a rate of 2.9%. Split by NN architectures, it corresponds to 17 (37) removed outliers for GNN (DNN) models from a total of 800 (1040). In no case more than two outliers have been removed from the original ten training repetitions of any individual model.

On the x-axis of the figure the corresponding values of μ_{AUC}, ranging from 0.67 to 0.83, are shown. On the y-axis the individual NN models are labeled, such that brackets indicate, what relational information between final state objects has been used, and the values in parentheses indicate the choices of n.

We conclude that all training setups have succeeded in the sense that all NN models result in values of $\mu_{\mathrm{AUC}} > 0.5$. The worst separation of signal from background we observe for the GNN models for which no relational information is exploited, indicated by three groups of NN architectures shown in pink, purple and red colors in the lower part of the figure. It is noted that the feature set zero refers to the case where the node convolution in the GraphConv operation is forcefully suppressed, and deliberately no information across nodes is exchanged at all. We anticipate that this approach counteracts the whole GNN idea. We still keep this configuration as part of the study, to gauge the effect and importance of the GraphConv operation itself. Compared to the feature set zero, the feature sets one and rnd single out cases in which information exchange across nodes takes place, but no real relational information is associated with it. Instead, the embedding spaces of the individual nodes are just mixed without particular prevalence. We note that even when allowing node convolution the GNN architecture falls significantly behind the comparable DNN architecture, even with a reduced input vector $\mathbf{x}^{\mathrm{DNN}}_{\mathrm{red}}$, as long as no mindful relational information across adjacent nodes i and j is provided according to ω_{ij}. This is true for assigning (one) the same or (rnd) random weights to each edge, irrespective of the expressiveness of the NNs, indicated by n. The superiority of the DNN architecture in this case cannot be attributed to the additional information about the jet parton association, since this information is available to all NN architectures under study. In particular it is passed on to the GNNs, in the explicit form of one-hot encoding, which is not even the case for the DNNs with $\mathbf{x}^{\mathrm{DNN}}_{\mathrm{red}}$ as input vector. At this occasion, we note that the additional (and in fact in this case redundant) information of the parton association in the form of one-hot encoding to the DNN does not lead to a significant increase of μ_{AUC}, compared to the implicit knowledge already provided by the positional information in $\mathbf{x}^{\mathrm{DNN}}_{\mathrm{red}}$. Hence, if the way this information is presented to the DNN were of influence, this influence is not significant in the scope of our study. We also observe that $\Delta\mu_{\mathrm{AUC}}$ is considerably larger for the DNNs.

In conclusion, if the advantage of the GNN over the DNN architecture were that potentially excessive degrees

of freedom in the DNN architecture are replaced by built-in constraints, the GNN architecture appears too confined, until these constraints are introduced mindfully. In turn the additional degrees of freedom of the DNN architecture result in a larger spread $\Delta\mu_{\mathrm{AUC}}$ due to random choices in the training setup.

The upper part of Fig. 3 reveals that, as soon as a domain-knowledge motivated ranking of information exchange across neighboring nodes is introduced, the GNN architecture significantly gains in separation power. Also here, this gain comes with an increase in $\Delta\mu_{\mathrm{AUC}}$. The choices of (green) ΔR^{-1}, (orange) ΔR, and (blue) m as weights leads to an increase in separation power in the given order, where for each choice the values of μ_{AUC} can be grouped with a corresponding internal spread. On the other hand, a significant gain in μ_{AUC}, when increasing the expressiveness of an NN within a certain configuration group in terms of n, for the utilized NN models, is not observed.

We note that, as in the case of the positional encoding of parton association in $\mathbf{x}_{\mathrm{red}}^{\mathrm{DNN}}$, all choices of relational information are intrinsic to the training sample and implicitly accessible to all NNs through their feature vectors. For ΔR this is, e.g., the case through ϕ and η in the primary features of each reconstructed object. However, the information of $\Delta\phi$ and $\Delta\eta$ between pairs of reconstructed objects appears too subtle in the high-dimensional feature space, so that none of the chosen architectures could grasp it without the assistance of an accordingly conditioned representation of $\mathbf{x}$ and $\mathbf{x}^{\mathrm{DNN}}$, even from a training sample with more than 200,000 events.

We further note that when turning the edge-weights of the GNN structure into TPs we did not obtain a separation of signal from background better than the domain-knowledge supported use of m. At the same time we observed a significantly increased spread in the achieved separation power based on random choices of the training setup.

Our physics prior assigns more physical meaning to the choice of ΔR over ΔR^{-1}, since due to causality we expect a closer relation between objects with smaller than larger spatial distance in ΔR. The observation that both choices of relational information lead to nearly similar results in μ_{AUC} we explain by a special characteristic of NN-based classification tasks in the given setup. For the NN decision, downgrading information from further-away objects is equivalent to upgrading close-by objects. The fact that corresponding DNN architectures follow the trends of the GNN architectures, as long as equipped with the same information, supports the assumption that it is this additional relational information between objects, rather than the GNN-specific operation of mixing features across nodes that leads to the increase of μ_{AUC}.

We note that, consistently for all architectures, the highest values of μ_{AUC} are obtained with an energy-weighted distance measure like m, which again follows our prior physics intuition. It is noteworthy that in this case the DNN architecture with $\mathbf{x}_{\mathrm{red}}^{\mathrm{DNN}}$ seems to significantly lose in separation power, compared to the other architectures. In fact μ_{AUC} even decreases for increasing values of n. Also $\Delta\mu_{\mathrm{AUC}}$ appears consistently higher for all DNN compared to the GNN architectures. These observations may be interpreted as indications of the advantage of careful guidance of the NN training- and model-setup over just confronting a highly expressive NN architecture, represented by a large number of TPs, with an even excessively large training sample. This guidance may be provided through the choice and representation of input features, as well as through the choice of a more constrained NN architecture. We note that the highest value of μ_{AUC} with the smallest spread $\Delta\mu_{\mathrm{AUC}}$ is indeed obtained from the GNN architecture with highest expressiveness, given for $n = 39$.

Neural Networks with Two Layers of Hidden Nodes

Moving on to an NN architecture with $k = 2$ introduces the ambiguity of how to choose n for each individual layer. To prevent any kind of potential selection biases, all ways of allocating the tested values of $n = 13, 26, 39$ to the individual layers/embedding spaces are shown in Fig. 4.

For the case of completely suppressed relational information for the GNN architectures (zero) μ_{AUC} remains lowest and unaffected by the choice of k, as expected for a setup in which any information transfer through a GaphConv operation is deliberately suppressed. At the same time we generally observe that no superior choice of allocating n across layers can be pointed to, throughout all tested architecture configurations. Especially the gain of using a configuration with $n = (39, 39)$ for both hidden layers/GraphConv operations over a configuration with $n = (13, 13)$ within a given architecture appears marginal.

However, the increase in k consistently mitigates the previously observed, clearly inferior separation of signal from background, of the GNN compared to the corresponding DNN architectures, for randomly (rnd) and unweighted (one) relational information. The μ_{AUC} values of these groups of GNN models start to clearly supersede the μ_{AUC} values of the one-layered DNN models with unweighted relational information, labeled by none in Fig. 3, even slightly taking the lead over the two-layered DNNs of the same kind, in terms of μ_{AUC}.

We note that all two-layered NN architectures without use of relational information still result in lower values of μ_{AUC} than all tested one-layered NN models that profit from the use of relational information, as presented in Fig. 3. At the same time, they are subject to an increased spread $\Delta\mu_{\mathrm{AUC}}$ compared to their one-layered counterparts, in most cases.

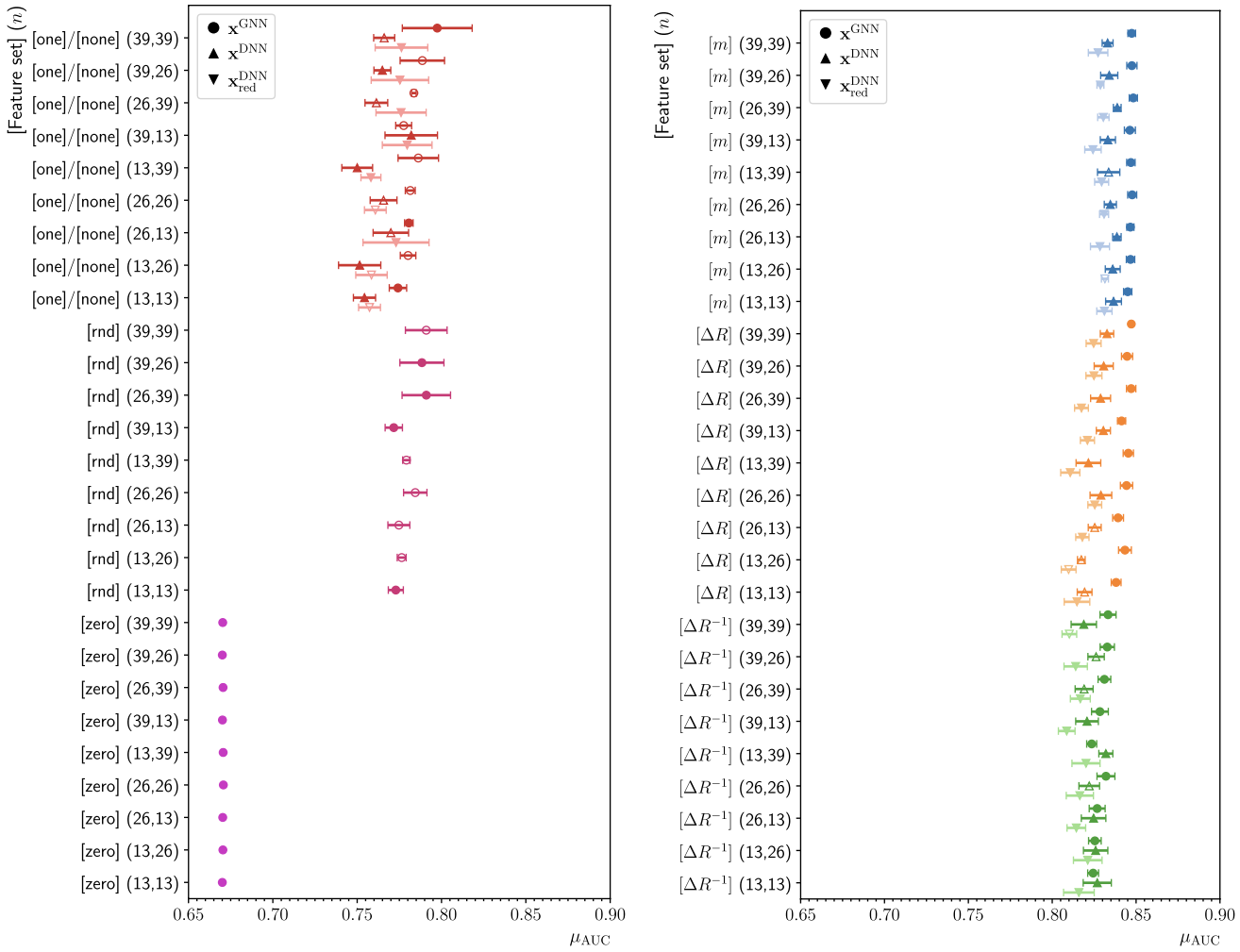

Fig. 4 Mean ROC-AUC μ_{AUC} as obtained for 54 different GNN and 72 corresponding DNN models with $k = 2$. The labels in brackets on the vertical axis indicate the use of relational information, as discussed in the "Presentation of the Feature Space" section, the numbers in parentheses correspond to n. The circles refer to GNN and the upward (downward) pointing triangles to DNN models with a default (reduced) set of input features $\mathbf{x}^{\text{DNN}}$ ($\mathbf{x}^{\text{DNN}}_{\text{red}}$), as discussed in the "Presentation of the Feature Space" section. For better readability, markers of the same configuration are shifted vertically along the y-axis. The bars are obtained from the sample variance of an ensemble, as described in the "Training Setup" section. NN architectures which belong to the same choices of varying parameters are spatially grouped and shown in the same color. Open markers indicate that significant outliers of the corresponding distribution of ROC-AUC values have been removed from the calculation of μ_{AUC} and its variance, as described in the text

In this sense, the wise choice of relational information outweighs the presumable advantage in expressiveness provided by $k = 2$, irrespective of the allocated values of n.

For the NN architectures including relational information, we observe no further, dramatic gains, with respect to their one-layered counterparts in μ_{AUC}, apart from a slight advantage of the GNN over the corresponding DNN architectures that seems to become more manifest. While this advantage is below the 1%-level it is still significant compared to $\Delta\mu_{\text{AUC}}$. A summary of the achieved values of μ_{AUC} for the one- and two-layered GNN models is shown in Fig. 5. An equivalent summary for the corresponding DNN models is shown in Fig. 6.

From the study we conclude that the external information of the ω_{ij} seems to give slight advantages to the GNNs with $k = 2$. We note that two subsequent GraphConv operations indeed convey more information than a DNN model with two hidden layers. Viewing ΔR^{-1}, ΔR, and m as distance measures, the first GraphConv operation conveys information about the nearest neighborhood of each i. The second GraphConv operation conveys information about the nearest neighborhood of the nearest neighbors, which is not the

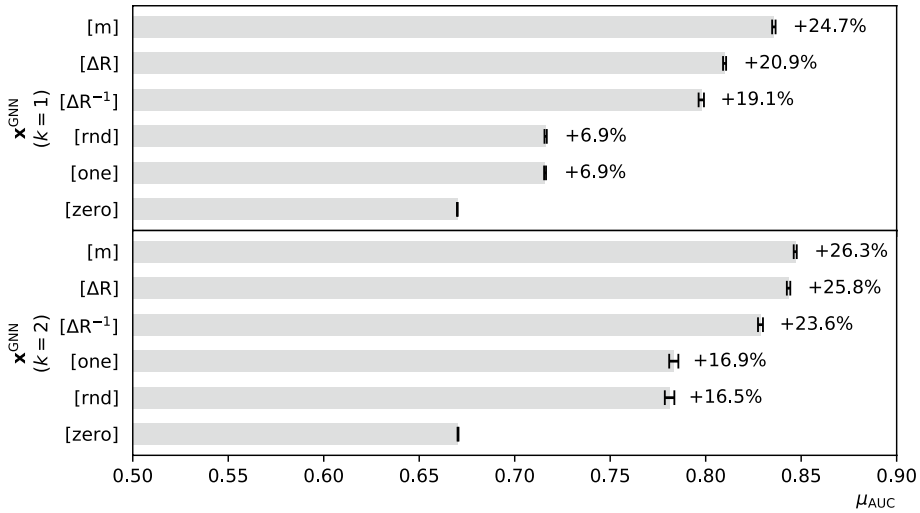

Fig. 5 Summary of the achieved values of μ_{AUC} for the GNN models with (upper half) one and (lower half) two GraphConv operations, with different use of relational information. For this summary, the associations of n with the highest values of μ_{AUC} in each group of GNN models have been used. The value of μ_{AUC} is displayed on the x-axis. Improvements relative to the least separating GNN with no relational information at all (zero) is given in numbers to the right of the bars. The use of relational information, as defined in Table 5, is indicated in brackets, on the y-axis

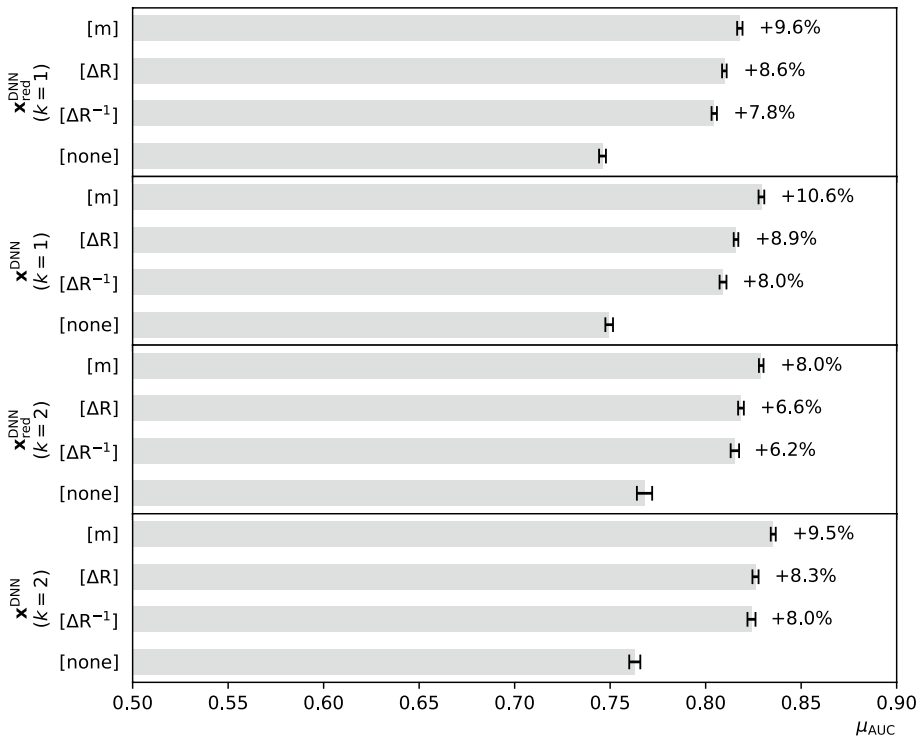

Fig. 6 Summary of the achieved values of μ_{AUC} for the DNN models with (upper half) one and (lower half) two hidden layers, with different use of relational information. For this summary, the associations of n with the highest values of μ_{AUC} in each group of DNN models have been used. The DNN configurations with $\mathbf{x}^{\mathrm{DNN}}$ and $\mathbf{x}^{\mathrm{DNN}}_{\mathrm{red}}$ are shown separately. The values of μ_{AUC} are displayed on the x-axis. Improvements relative to the least separating DNN with no relational information (none) is given in numbers to the right of the bars. The use of relational information, as defined in Table 5, is indicated in brackets, on the y-axis

same as in the case of the first operation. This information is indeed not primarily accessible to the DNN architectures, but it emerges from the definition of the GraphConv operation. Along this line, once again, we conclude that not the mixing of features across neighboring nodes i during the GraphConv operation, but the additional (implicit) information accessible to the GNN through this operation is the source for the slight gain in μ_{AUC}. This interpretation is supported by the observation that an increase in n does not lead to any significant improvements in μ_{AUC} despite the increase in expressiveness of the models. This also explains why the GNN models with constant (one) or randomly associated (rnd) weights suffer in their performance: Through both ways of assigning weights across the nodes i the information about any kind of distance measure across nodes, in what ever space, is omitted.

A summary of the GNN and DNN configurations with the highest values of μ_{AUC} is given in Table 7.

Table 7 Summary of n, number of TPs (N_{TP}), and μ_{AUC} of the (upper part) GNN and (lower part) DNN models with the highest results in μ_{AUC}

NN arch.	n	N_{TP} ($N_{\text{TP}}^{\text{eff}}$)	μ_{AUC}	Label
GNN	(39)	1093	0.8387±0.0006	$\text{GNN}_{k=1}$
	(26, 39)	2809	0.8484±0.0008	$\text{GNN}_{k=2}$
	(29, 28, 29, 29)	5829	0.8546±0.0006	$\text{GNN}^{\uparrow}$
DNN	(26)	5773 (4753)	0.8300±0.0011	$\text{DNN}_{k=1}$
	(26, 39)	6839 (5819)	0.8388±0.0007	$\text{DNN}_{k=2}$
	(13, 14, 14)	3294 (2784)	0.8400±0.0006	$\text{DNN}_{\text{eff}}^{\downarrow}$
	(11, 10, 11, 11)	2816 (2385)	0.8386±0.0007	$\text{DNN}^{\downarrow}$

For the DNN models a number of effective TPs $N_{\text{TP}}^{\text{eff}}$, as defined in the text, is also given in parentheses. Corresponding summaries are given for the cases of $k = 1$ and 2. In all cases m has been used as relational information between nodes/physics objects. Also shown are the configurations of three additional NN models discussed in the "Neural Networks with Comparable Numbers of Trainable Parameters" section: the $\text{GNN}^{\uparrow}$ model with N_{TP} comparable within 1% with $N_{\text{TP}}(\text{DNN}_{k=2})$ and the $\text{DNN}^{\downarrow}$ ($\text{DNN}_{\text{eff}}^{\downarrow}$) model with N_{TP} ($N_{\text{TP}}^{\text{eff}}$) comparable within 1% with $N_{\text{TP}}(\text{GNN}_{k=2})$

Neural Networks with Comparable Numbers of Trainable Parameters

As stated before, two principally different NN architectures lack comparability in the sense that it might be more natural to pick up certain information from the training sample through one or the other architecture. As long as it addresses an intrinsic property of one or the other architecture, such a difference is part of the benchmark comparison. If, on the other hand, such an inequality results from withholding primary information from one or the other architecture, or potentially inappropriate advantages in terms of expressiveness, an effort should be made to study and estimate the effect of it.

We have identified and noted such an inequality, in the beginning of the "Results" section, in terms of the number of TPs (N_{TP}), which turns out to be naturally higher for the DNN compared to the GNN architecture, due to the usually much larger input layer. We therefore complete our study by three additional setups, for which we drop the restrictions on the layered structure of the trained NN models in favor of comparable numbers of TPs.

As shown in Table 7, the GNN model with the highest value of μ_{AUC} (labeled as $\text{GNN}_{k=2}$) is based on $k = 2$ and $n = (26, 39)$, with $N_{\text{TP}}(\text{GNN}_{k=2}) = 2809$ and a value of $\mu_{\text{AUC}} = 0.8484 \pm 0.0008$. The DNN with the highest result of μ_{AUC} (labeled as $\text{DNN}_{k=2}$) is based on the same configuration in terms of k and n, with $N_{\text{TP}}(\text{DNN}_{k=2}) = 6839$ and a value of $\mu_{\text{AUC}} = 0.8388 \pm 0.0007$. One may argue that for the DNN model, not the full set of TPs is really actively contributing to the solution of the task, since part of the input space is regularly filled with zeros, e.g., if fewer than eight jets are selected in an event. Therefore, we estimate, in addition to N_{TP}, a number of effective TPs ($N_{\text{TP}}^{\text{eff}}$) from the product of N_{TP} with the average number of nonzero input nodes in $\mathbf{x}^{\text{DNN}}$, evaluated on the training dataset. This results in a value of $N_{\text{TP}}^{\text{eff}}(\text{DNN}_{k=2}) = 5819$.

In a first approach we survey varying DNN structures with N_{TP} ($N_{\text{TP}}^{\text{eff}}$) comparable to $N_{\text{TP}}(\text{GNN}_{k=2})$. We do this based on the following algorithm: We allow $k \leq 4$ and any number of nodes per hidden layer (n). Of all DNN configurations for which $N_{\text{TP}}(\text{GNN}_{k=2})$ is matched by N_{TP} ($N_{\text{TP}}^{\text{eff}}$) within a margin of 1%, the model with the smallest spread of n across layers is selected. If no DNN configuration with N_{TP} ($N_{\text{TP}}^{\text{eff}}$) within a 1% margin of the target value can be found the closest possible model is chosen. This situation occurs only in models with one hidden layer.

This procedure ensures a homogeneous structure of hidden layers. As a result, e.g., a DNN configuration with $k = 4$ and $n = (11, 10, 11, 11)$, with $N_{\text{TP}} = 2816$, further on referred to as $\text{DNN}^{\downarrow}$, is preferred over a model with $k = 4$ and $n = (10, 8, 14, 24)$, even though the latter results in an exact match with $N_{\text{TP}}(\text{GNN}_{k=2})$. The $\text{DNN}^{\downarrow}$ model achieves a value of $\mu_{\text{AUC}} = 0.8386 \pm 0.0007$. A second DNN configuration with $k = 3$ and $n = (13, 14, 14)$, with $N_{\text{TP}}^{\text{eff}} = 2784$ within the 1% margin of $N_{\text{TP}}(\text{GNN}_{k=2})$ is also considered and further on referred to as $\text{DNN}_{\text{eff}}^{\downarrow}$. This model achieves a value of $\mu_{\text{AUC}} = 0.8400 \pm 0.0006$. We observe that, although the $\text{DNN}^{\downarrow}$ ($\text{DNN}_{\text{eff}}^{\downarrow}$) model uses only 41% (48%) of N_{TP} ($N_{\text{TP}}^{\text{eff}}$) of $\text{DNN}_{k=2}$, this does not result in any significant loss in separation power, after training.

In a second approach we survey varying GNN structures with N_{TP} comparable to $N_{\text{TP}}^{\text{eff}}(\text{DNN}_{k=2})$. For this purpose we exploit the same algorithm as described above, resulting in a GNN with $k = 4$ and $n = (29, 28, 29, 29)$ with $N_{\text{TP}} = 5829$, further on referred to as $\text{GNN}^{\uparrow}$. This model achieves a value of $\mu_{\text{AUC}} = 0.8546 \pm 0.0006$. It reveals the highest value of μ_{AUC} across all tested models. The difference in μ_{AUC} with respect to other NN configurations

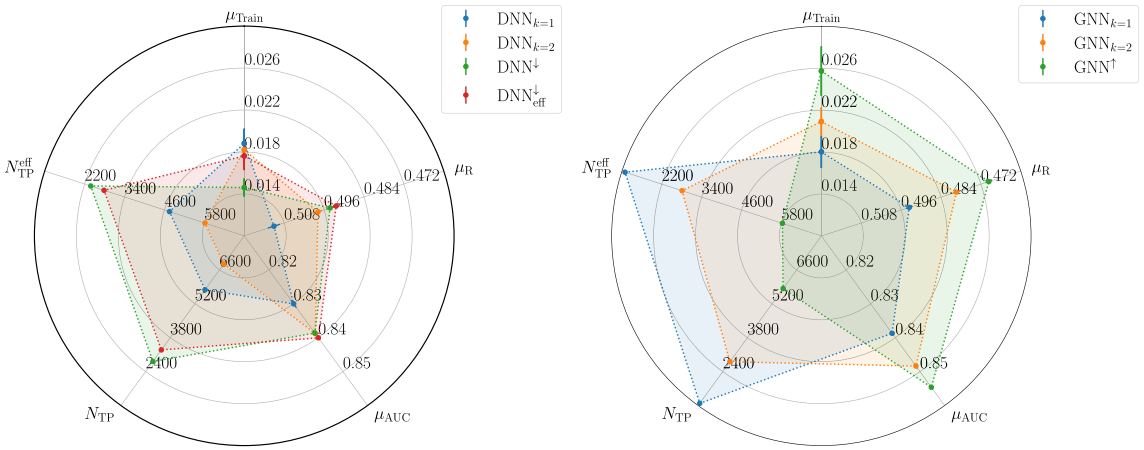

Fig. 7 Visualization of GNN and DNN models with the highest values in μ_{AUC} and $k = 1$ (GNN$_{k=1}$, DNN$_{k=1}$) and $k = 2$ (GNN$_{k=2}$, DNN$_{k=2}$). Also shown are the DNN models DNN$^{\downarrow}$ and DNN$^{\downarrow}_{\text{eff}}$ with a comparable number of (effective) TPs as the GNN$_{k=2}$ model, and the GNN model GNN$^{\uparrow}$ with a comparable number of effective TPs as the DNN$_{k=2}$ model. Five different metrics to evaluate the properties of an NN model are shown: N_{TP} ($N_{\text{TP}}^{\text{eff}}$), the mean convergence rate μ_{Train}, the mean empirical risk evaluated on the test sample μ_R, and μ_{AUC}. The spanned area in the figures indicates the capability of an NN model to fulfill the task

is only at the 1%-level, but it is still significant in terms of $\Delta\mu_{\text{AUC}}$.

From this finding, we conclude that the GNN model with the same expressiveness as a maximally comparable DNN must have intrinsic advantages over the DNN model in extracting additional information from the given, large training sample. For the benchmark setup in use, this advantage is small but significant in the scope of the study. It emerges after external augmentation with an energy-weighted distance measure like m between the input objects/nodes, and more clearly manifests itself in the study for $k > 1$. We anticipate that this gain originates from the hierarchically structured information about nearest neighbors and the nearest neighborhood of nearest neighbors of node i, when viewing the edge weights ω_{ij} as a distance measure. This information is an intrinsic property of the GNN model and not easily accessible through the more simplistic DNN structure. An increase of TPs of the simpler DNN structure does not compensate for this informational advantage. In this interpretation the gain of GNN$^{\uparrow}$ over all other configurations should mostly be attributed to the increase in k over the association of n per GraphConv operation. The parameter choices of the DNN$^{\downarrow}$, DNN$^{\downarrow}_{\text{eff}}$, and GNN$^{\uparrow}$ models and correspondingly achieved values of μ_{AUC} are also given in Table 7.

Convergence Behavior

In this study we have investigated the capacities of GNN and DNN models to fulfill their primary target, i.e., to provide the best possible solutions to the classification task defined in the "Task Definition" section. We anticipate that, in particular in practical life, the properties of an NN architecture may be evaluated in other terms, viz. the mean of the training speed μ_{Train}, which we evaluate as the inverse of the epoch with the highest value of the ROC-AUC on the validation dataset and the mean of the empirical risk obtained from the test dataset μ_R, which we take as a measure of the generalization property of the given NN model under study. To conclude our studies we provide a visualization of these properties and all other properties of the NN models that have been discussed throughout the paper so far, in Fig. 7. In Fig. 7 (left) μ_{Train}, μ_R, μ_{AUC}, N_{TP}, and $N_{\text{TP}}^{\text{eff}}$ for the DNN$_{k=1}$, DNN$_{k=2}$, DNN$^{\downarrow}$, and DNN$^{\downarrow}_{\text{eff}}$ models are shown, on five independent axes. The axes are defined such that values closer to the common origin of the figure are disfavored. This is in particular true for N_{TP}, $N_{\text{TP}}^{\text{eff}}$, and μ_R, where the values are given in descending order when moving away from the origin. In this sense a larger size of the correspondingly colored area indicates the larger capability of a given NN model to adequately solve the presented task. In Fig. 7 (right), the same quantities are shown for the GNN$_{k=1}$, GNN$_{k=2}$, and GNN$^{\uparrow}$ models. The axes ranges are kept the same to ease comparison between both architectures.

In terms of μ_{Train} the DNN models usually fall behind their corresponding GNN counterparts. This finding, as well as the observation that the GNN models usually achieve a comparable or slightly larger value of μ_{AUC}, after training, indicates the assumed effect of guiding the convergence by constraints, which are built-in to the GNN architecture. This property allows the GNN architecture not only to converge to a solution that is equally good or slightly better than the solution found by corresponding DNN architectures, but also to converge faster and typically with a smaller number of TPs. We note though that a clear correlation between μ_{Train} and N_{TP} ($N_{\text{TP}}^{\text{eff}}$) cannot

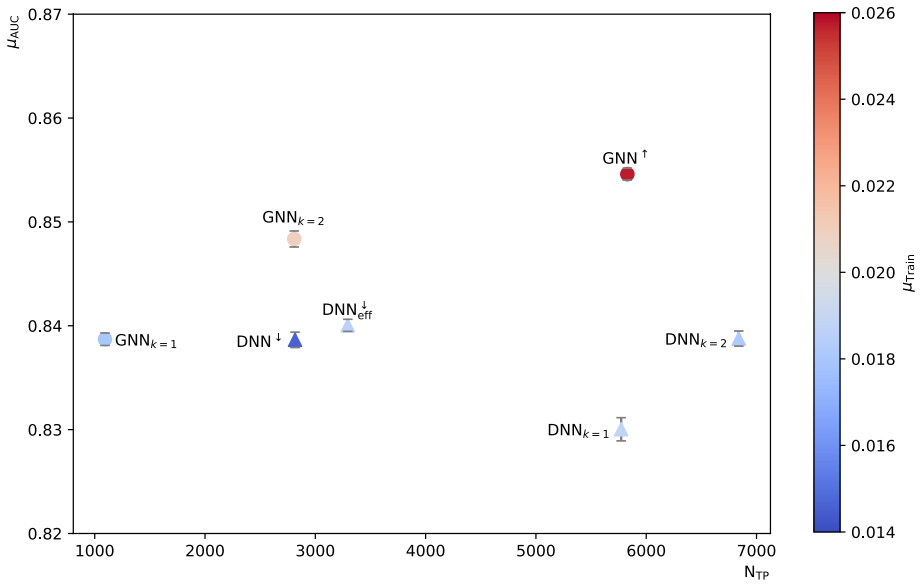

Fig. 8 Overview of the GNN and DNN models with $k = 1$ and 2, and the highest achieved values of μ_{AUC}, as well as a GNN model with an increased number of TPs ($GNN^{\uparrow}$) and a DNN model $DNN^{\downarrow}$ ($DNN^{\downarrow}_{eff}$) with a restricted (effective) number of TPs. The GNN models are indicated by circles and the DNN models by triangles. The models are shown in a three-dimensional space built from μ_{AUC}, N_{TP} and μ_{Train}. The bars in μ_{AUC} are obtained from the sample variance of a training ensemble, as described in the "Training Setup" section. The quantity μ_{Train}, indicated by the color code of the points corresponds to the inverse of the epoch with the highest value of the ROC-AUC on the validation dataset

be deduced from our study. We understand this situation such that a less expressive NN model, with fewer degrees of freedom for the minimization process, may well lead to a more pronounced landscape of the expected risk and thus reduced μ_{Train}. In addition we note that also μ_R for the DNN models falls behind, compared to their GNN counterparts. Here we observe the benefit of a regularizing effect of the built-in constraints, which correlates with reduced numbers of (effective) TPs. It is obvious that an NN model with more TPs reveals a higher vulnerability to specific properties of the training sample. A summary of all NN configurations that have been discussed in this section is shown in Fig. 8.

Summary

With this paper we have made an effort to put the comparison of graph neural network (GNN) and equivalent fully connected feed-forward neural network (DNN) architectures on a maximally fair ground. Under the laboratory conditions of a high-energy physics process of interest, at the CERN LHC, we have controlled the definition of the task, choice of non-tunable (hyper-)parameters of the models, which are not subject to the training, and amount of information primarily presented to the neural network models, through their input feature vectors.

Within the scope of the study we have demonstrated clear evidence that the presumable advantage of the more complex GNN over an equivalent DNN structure does not originate from an uncontrolled mixture of features in the embedding space of the graph nodes, but from the extra relational information between nodes, that we have added based on domain-knowledge. Without this extra knowledge the GNN models fall behind equivalent DNN models in terms of their capability to separate signal from background. Neither are the GNN models superior in terms of their separation power to equivalent DNN models, as long as these are equipped with the same information in the input space.

Both, the built-in permutation invariance and the circumstance that the GNN architecture a priori is not bound to a fixed number of nodes might be viewed as advantages. They might also have positive influence on the convergence behavior of the training. On the other hand it cannot be deduced that either of these properties significantly contributes to an increase, e.g., in the power to separate a given signal from background. Any advantage of the GNN over the DNN architecture that we observed in our studies could be traced back to the access to more information, which, when given to the other architecture lead to the same performance also for the other architecture.

The real advantage of the GNN over the DNN structure emerges as soon as more than one GraphConv operations are applied, during which the GNN structure naturally accesses more relational information between nodes than a DNN has access to. In the configurations investigated in our study this gain is tied to the use of relational information that can be interpreted as a distance measure to define

proximity between two nodes. Apart from that we observe significant advantages of the GNN over the DNN architecture in terms of convergence and generalizability that we attribute to a level of built-in implicit constraints to the GNN model resulting in a better ratio of accessible information over trainable parameters of the model. We anticipate that these advantages might be more pronounced the more hierarchical the training data are. We assume that this property is the basis for the success of GNN structures when applied to particle physics jets, which are highly hierarchical objects. In conclusion, we expect the highest gain of a GNN over a DNN structure for tasks based on hierarchically structured data, ideally based on a known distance measure.

Acknowledgements This research was supported by the German Federal Ministry of Education and Research (BMBF) under grant 05H21VKCCB.

Author Contributions E.P., M.W. and R.W wrote the main manuscript text. E.P. created Figs. 1, 3, 4, 5, 6, 7, and 8. R.W. created Fig. 2. Y.C. wrote a back-end script for Figs. 3, 4 and 7. All authors contributed to the manuscript and reviewed it.

Funding Open Access funding enabled and organized by Projekt DEAL. This study was funded by German Federal Ministry of Education and Research (BMBF) (Grant number 05H21VKCCB).

Data Availability No datasets were generated or analyzed during the current study.

Declarations

Competing Interests The authors declare no competing interests.

References

1. Aguillard D, Albahri T et al (2023) Measurement of the positive muon anomalous magnetic moment to 0.20 ppm. Phys Rev Lett 131(16):161802
2. Metropolis N, Rosenbluth A et al (2004) Equation of state calculations by fast computing machines. J Chem Phys 21(6):1087
3. Rosenbluth M (2003) Genesis of the Monte Carlo algorithm for statistical mechanics. AIP Conf Proc 690(1):22
4. Aad G, Abat E et al (2008) The ATLAS experiment at the CERN large Hadron Collider. JINST 3:S08003
5. Chatrchyan S, Hmayakyan G et al (2008) The CMS experiment at the CERN LHC. JINST 3:S08004
6. Salam G (2010) Towards jetography. Eur Phys J C67:637–686
7. Sirunyan A, Tumasyan A et al (2019) Performance of missing transverse momentum reconstruction in proton-proton collisions at $\sqrt{s} = 13$ TeV using the CMS detector. JINST 14(07):P07004
8. Feynman R (1949) The theory of positrons. Phys Rev 76:749–759
9. Scarselli F, Gori M et al (2009) The graph neural network model. IEEE Trans Neural Netw 20(1):61–80
10. Micheli A (2009) Neural network for graphs: a contextual constructive approach. IEEE Trans Neural Netw 20(3):498–511
11. Kipf T, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv e-prints, page arXiv:1609.02907
12. Hamilton W, Ying R et al (2017) Inductive representation learning on large graphs. In: Proceedings of the 31st international conference on neural information processing systems, NIPS'17, Red Hook, NY, USA. Curran Associates Inc, pp 1025–1035
13. Veličković P, Cucurull G et al (2018) Graph attention networks. International conference on learning representations.
14. Shlomi J, Battaglia P, Vlimant JR (2020) Graph neural networks in particle physics. Mach Learn Sci Technol 2(2):021001
15. Sirunyan A, Tumasyan A et al (2019) Search for tth production in the H→bb decay channel with leptonic tt decays in proton-proton collisions at $\sqrt{s} = 13$ TeV. JHEP 03:026
16. Workman R, Burkert V et al (2022) Review of particle physics. PTEP 2022:083C01
17. Sirunyan A, Tumasyan A et al (2018) Identification of heavy-flavour jets with the CMS detector in pp collisions at 13 TeV. JINST 13(05):P05011
18. Alwall J, Frederix R et al (2014) The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations. JHEP 07:079
19. Alwall J, Herquet M et al (2011) MadGraph 5: going beyond. JHEP 06:128
20. Sjöstrand T, Ask S et al (2015) An introduction to PYTHIA 8.2. Comput Phys Commun 191:159–177
21. de Favereau J, Delaere C et al (2014) DELPHES 3, a modular framework for fast simulation of a generic collider experiment. JHEP 02:057
22. Cacciari M, Salam G et al (2008) The anti-k^t jet clustering algorithm. JHEP 04:063
23. Fey M, Lenssen J (2019) Fast graph representation learning with pytorch geometric. arXiv e-prints, arXiv:1903.02428
24. Paszke A, Gross S et al (2019) Pytorch: an imperative style, high-performance deep learning library. arXiv e-prints, arXiv:1912.01703
25. Kingma D, Ba J (2017) Adam: a Method for Stochastic Optimization. arXiv e-prints, arXiv:1412.6980
26. Morris C, Ritzert M et al (2019) Weisfeiler and leman go neural: higher-order graph neural networks. Proc AAAI Conf Artif Intell 33(01):4602–4609