A MODERN APPROACH TO GNN MODEL SELECTION FOR NODE PROPERTY PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

In the graph machine learning field, graph neural networks (GNNs) have become a standard class of models. However, most studies that use such models for their experiments overlook several important aspects of model selection. First, they often do not pay much attention to how GNN backbone is composed and instead use some fixed, arbitrarily chosen sequence of operations. Second, they normally do not consider message-passing mechanism as an option for architecture tuning and choose not to include it in the search space, while treating GNNs with different aggregation functions as completely independent methods. Lastly, they may not invest enough resources in GNN hyperparameter tuning, while using quite simple optimization techniques such as random search over a coarse grid of values. We believe these issues can notably limit the potential of GNNs as a class of models, reduce their predictive performance, and ultimately diminish their practical utility. In this work, we propose to make a step beyond the existing practices in GNN model selection and show how to construct a modern GNN baseline that better accounts for the given graph data and prediction task. Specifically, we investigate how the joint optimization of architecture backbone, its message-passing mechanism, and other hyperparameters over a complex configuration space can extend and outperform conventional GNN tuning approaches. By incorporating each mentioned aspect of model selection into a single tuning procedure and using a standard Bayesian optimization technique, we can achieve notably higher predictive performance in node property prediction than by using many other methods proposed in prior works on GNN model design and implemented in graph deep learning packages. The introduction of a flexible generalized aggregation function further improves the performance and provides more evidence that various prediction tasks may require finer tuning of the message-passing behavior. Our study provides a new perspective on what model selection in modern GNNs should account for and how their architecture can be optimized to obtain stronger results in practice.

1 INTRODUCTION

Impact of GNN model selection on prediction quality In the graph machine learning field, graph neural networks (GNNs) have become a very competitive and promising class of methods that allows us to solve various prediction tasks on graph-structured data from different domains. Over the past few years, a vast number of studies investigated the properties (Xu et al., 2019; Brody et al., 2022; Park et al., 2025; Arnaiz-Rodriguez & Errica, 2026; Kormann et al., 2026) and design principles (Xu et al., 2018; You et al., 2020; Chen et al., 2022; Luo et al., 2024) of GNN models, so the community has reached a common understanding of what a typical GNN model can look like, how the layers in its backbone can be arranged, and within what ranges its hyperparameters can be set. In particular, it has become clear that most node property prediction tasks can be solved with a relatively small number of message-passing layers and relatively low dimensionality of latent space. This allows for working with rather compact models that require quite modest computational resources compared to frontier models in vision or language domains and enables a more rigorous model selection for a given prediction task. At the same time, several recent works provided the first concrete results indicating that the design of GNN architecture and the choice of its hyperparameters can highly depend on the given graph data and prediction task. For example, Luo et al. (2024) found that by using quite a simple GNN backbone layout, including the standard deep learning operations such

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

as residual connections (He et al., 2016), normalizations (Ba et al., 2016), and dropout (Srivastava et al., 2014) into the search space, and tuning other basic hyperparameters such as the model size and learning rate, it is possible to drastically improve the performance of GNN models for node property prediction. However, another revealing finding in this work was that different prediction tasks may require very different architecture design: residual connections and normalizations can be very useful on some datasets and harmful on others, and the preferable values of the remaining hyperparameters also vary significantly. These observations suggest that careful model selection for GNNs is often not only possible but also necessary if our concern is to achieve the best prediction quality.

Important aspects of GNN model selection To solve node property prediction tasks across various real-world applications, GNNs have to deal with the two main sources of information: node features, which describe each sample independently of the others, and graph structure, which brings a valuable signal about the connections between different samples. Thus, two important questions need to be addressed in GNN model selection: how to transform the information across different message-passing rounds and how to aggregate it from neighboring nodes within a single round. One can isolate the two corresponding components in GNN architecture design. The first component is a GNN backbone layout, where we need to decide how to organize a single backbone block, what operations to include and in which order to place them. The second component is a message-passing mechanism, where we need to choose how to construct the messages from neighboring nodes and combine them to update the representations of central nodes. Besides these two components, there are many other basic hyperparameters that may significantly affect the predictive performance and should be selected just as carefully: for example, general model capacity expressed by the depth and width of GNN architecture, or optimization settings that affect the overall GNN training dynamics. Together with the mentioned components, they induce a very complex search space that has to be explored adaptively for the given graph data and prediction task.

Limitations of existing approaches to GNN model selection In the graph machine learning literature, there is a certain number of works that raised the problem of model selection for GNNs and devoted some attention to the mentioned components. The first notable works in the field (Kipf & Welling, 2017; Hamilton et al., 2017; Veličković et al., 2018; Xu et al., 2019) were primarily focused on the design of message-passing mechanism that would bring universally best performance and treated GNNs using different underlying aggregation methods as completely independent counterparts. Such an approach could indeed yield visible results, since the rest of their architecture, including its backbone layout and other basic hyperparameters, could be not properly tuned, which allowed a more expressive or simply heavier message-passing function to cover for a significant portion of the overall predictive performance. In later works (Xu et al., 2018; Chen et al., 2020; Li et al., 2020; Chen et al., 2022), the attention in the field shifted to directly improving backbone design by combining representations from different message-passing rounds and choosing the order between normalization, activation, and message-passing layers in a single GNN block. These studies found that backbone layout can have a notable impact on predictive performance, but they did not attempt to select it specifically for the given task and were rather exploring which backbone layout yields higher quality on average. Another line of works transferred the techniques from neural architecture search (NAS) to the problem of GNN model selection: they explored their own GNN architecture spaces with reinforcement learning methods (Gao et al., 2020; Zhou et al., 2022), composed super-networks that simultaneously incorporated many different architectures and extracted the sub-network with the highest weights (Huan et al., 2021; Cai et al., 2021), or evolved entire populations of GNN architectures (Shi et al., 2022; Chen et al., 2021). However, these numerous studies on graph NAS either completely avoided tuning other basic hyperparameters, or conducted this only after selecting the primary model architecture, or used a rather coarse grid of values. Moreover, some subsequent works (Nunes & Pappa, 2020; Qin et al., 2022) found that such advanced NAS approaches, despite their significantly higher implementation complexity, cannot reliably outperform random search over the same configuration spaces, which questions their necessity in practice. More recent works on effective GNN design (You et al., 2020) and thorough GNN evaluation (Luo et al., 2024) also proposed to include residual connections, normalizations, and activations into the joint search space as distinct hyperparameters and adjust them specifically for a given prediction task. However, they did not investigate diverse backbone layout options and did not consider the modern message-passing functions as an option for hyperparameter tuning. Moreover, they either ran an exhaustive grid search over a small number of configurations or used random search.

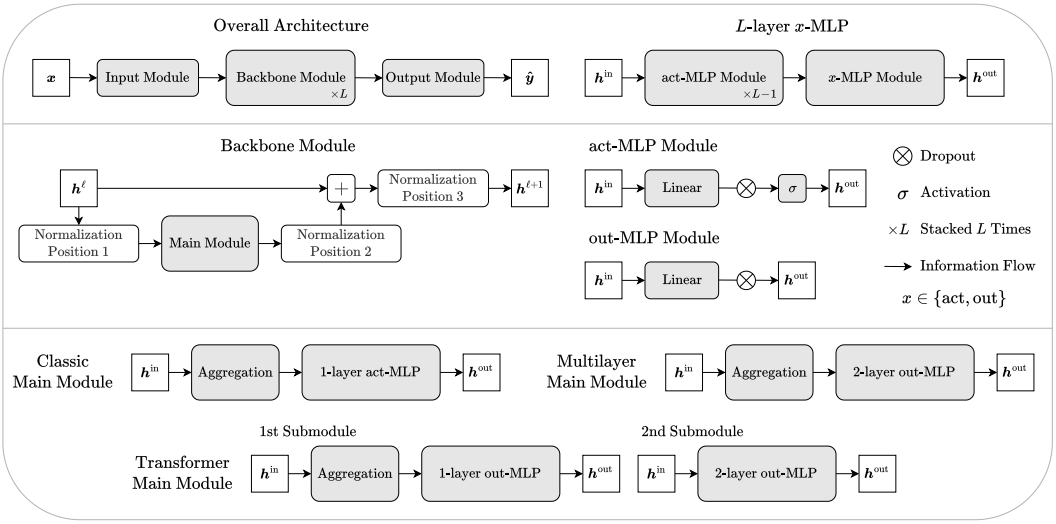

Figure 1: The proposed mGNN architecture. Refer to Section 2 for details.

Our contribution Our review suggests that previous works mostly treated GNN backbone layout and message-passing mechanism in isolation from each other and from the remaining hyperparameters: some of them searched over aggregation functions while keeping the backbone layout and the training setup fixed, others explored GNN backbone design while relying on a single predefined aggregation function, and many selected the model architecture first and only then adjusted the basic hyperparameters over a coarse grid of values. To the best of our knowledge, none of these works represented backbone layout, message-passing mechanism, and other basic hyperparameters as equally important design options within a single search space and tuned them jointly for the given graph data and prediction task. We argue that these issues can notably limit the potential of GNNs as a class of models and reduce their predictive performance. In this work, we propose to make a step beyond the existing practices in GNN model selection and show how to construct a modern GNN baseline that better accounts for the given graph data and prediction task. Specifically, we formulate GNN model selection as a single tuning procedure that explores a joint search space of backbone layout, message-passing mechanism, and other basic hyperparameters with a standard Bayesian optimization technique (Section 2). Our ablation study and analysis of tuning traces show that tuning these components is notably more effective than fixing any of them, since their preferable choice varies across prediction tasks, and that Bayesian optimization is a reliable way to explore such a search space (Sections 4 and 5). We further introduce a generalized aggregation function that can reproduce several standard message-passing operations or any intermediate behavior between them (Section 6), and show that the resulting models outperform many GNN baselines and model selection approaches under an equal tuning budget (Section 7).

2 mGNN: A MODERN APPROACH TO GNN MODEL SELECTION

In this section, we introduce mGNN (modern GNN), an approach to GNN model selection for node property prediction that can serve as a strong baseline and a worthy alternative for conventional tuning approaches. It builds on a GNN architecture with a flexible modular design and runs an adaptive tuning procedure to explore the associated search space on the given graph data and prediction task.

mGNN architecture and hyperparameters First, we provide an overview of the mGNN architecture, as depicted on Figure 1 (*Top row, left*). It starts with an input module, which is represented by an L_{in}-layer out-MLP (the output is not activated) and transforms an input vector of node features $\boldsymbol{x} \in \mathbb{R}^f$ to an initial hidden representation $\boldsymbol{h}^0 \in \mathbb{R}^d$, where d is the hidden dimension of the model. Next, there is a backbone, which stacks L_{main} identical backbone modules visualized in Figure 1 (*Middle row, left*). Each of them consists of a main module wrapped with a residual connection and some combination of normalization functions that can be placed on 3 different positions. This technically results in 2^3 normalization configurations, but we consider only those of them that provide higher training stability and predictive performance in practice (see Appendix C). The reason for

using several normalization configurations is that different main modules and aggregation functions may favor different normalization patterns, which should be served by the corresponding categorical hyperparameter. Finally, there is an output module, which is represented by an L_{out}-layer out-MLP and transforms the final representation $\boldsymbol{h}^L \in \mathbb{R}^d$ to predictions $\hat{\boldsymbol{y}} \in \mathbb{R}^k$, where k equals the number of classes for multiclass classification and 1 for regression and binary classification. We treat the numbers of layers L_{in}, L_{main}, L_{out} and the hidden dimension d as numerical hyperparameters and include them in the search space. Since mGNN includes numerous MLP modules with dropout and activations, we also include input dropout rate p_{in} and backbone dropout rate p_{main} as numerical hyperparameters, and activation function σ as a categorical one.

Further, we consider three types of the main module inspired by various design practices from prior graph machine learning studies, which are visualized on Figure 1 (*Bottom row*). The first is Classic architecture, which represents the GCN model design (Kipf & Welling, 2017) adopted in many subsequent works and consists of an aggregation function followed by a linear layer, dropout, and activation function. The second is Multilayer architecture, which follows the GNN design used by Platonov et al. (2023) and consists of an aggregation function followed by a 2-layer MLP. The last is Transformer architecture, which is based on the original Transformer model (Vaswani et al., 2017) and consists of two submodules, each wrapped with skip-connections and normalizations separately: an aggregation function followed by a linear layer, and a 2-layer MLP. The main module type is included in the search space as a separate categorical hyperparameter. Throughout this work, we refer to any specific combination of the main module type and the surrounding normalization placement as backbone layout. In our study, we consider 4 normalization configurations for Classic architecture, 4 for Multilayer, and 2 for Transformer, which results in 10 possible options of backbone layout.

Besides backbone layout, mGNN has several hyperparameters that control the message-passing mechanism. The first one is the aggregation function that serves as the base message-passing operation. Here, we choose a very straightforward path and consider the most widely used options in the graph machine learning literature: standard mean aggregation used by Hamilton et al. (2017); symmetrically normalized mean aggregation introduced by Kipf & Welling (2017); GAT by Veličković et al. (2018) as the most famous variant of graph-based attention; and standard local self-attention that is often referred to as Local Graph Transformer (LGT, Platonov et al., 2023). Furthermore, several studies (Platonov et al., 2023; Luo et al., 2024) examined a mechanism in which the initial representation of a central node is transformed separately outside of the aggregation and then added to its result. We refer to this operation as identity aggregation and include a corresponding boolean hyperparameter. Lastly, another boolean hyperparameter controls the presence of self-loops, which allow each node to take part in aggregation together with its immediate neighbors. We also define reasonable distributions for other basic hyperparameters that affect the GNN training dynamics, such as learning rate and weight decay (see Appendix C for the complete search spaces).

mGNN tuning procedure The last question is how to explore the introduced search space with numerous categorical and numerical hyperparameters. Many works in the field run grid search over a small number of configurations or use random sampling over a larger search space, and only some recent studies treat model tuning as a blackbox optimization problem and apply Bayesian optimization techniques to searcher for better configurations more efficiently. We believe this approach can be especially suitable for GNN model selection, so we choose it to jointly tune the hyperparameters of model architecture and training setup. Specifically, we use Tree-structured Parzen Estimator (TPE, Bergstra et al., 2011) implemented in Optuna package (Akiba et al., 2019) as an off-the-shelf solution that allows us to adaptively sample configurations from the induced search space.

3 Experimental setup

Evaluation protocol Each of the considered model selection approaches runs model tuning with a fixed budget on the number of trials using its own sampling strategy and then evaluates the best found configuration with 5 different random seeds. Configurations are compared by their performance on the validation subset, measured by accuracy for multiclass classification, average precision for binary classification, and R^2 score for regression. By default, the tuning budget is set to 100 trials, of which the first 25 are spent on startup sampling, and the reported results are obtained on the test subset.

Graph datasets and prediction tasks In our study, we use 25 graph datasets that include 15 classic datasets: `cora`, `citeseer`, `pubmed` (Sen et al., 2008; Yang et al., 2016), `coauthor-cs`,

coauthor-physics, amazon-computers, amazon-photo (Shchur et al., 2018), lastfm-asia, facebook (Rozemberczki et al., 2021), flickr (Zeng et al., 2020), wiki-cs (Mernyei & Cangea, 2020), roman-empire, amazon-ratings, amazon-ratings-full, questions (Platonov et al., 2023); and 10 datasets from the GraphLand benchmark (Bazhenov et al., 2025): tolokers-2, city-reviews, artnet-exp, hm-categories, hm-prices, avazu-ctr, city-roads-M, city-roads-L, twitch-views, and artnet-views. Taken together, this collection covers 15 multiclass classification, 4 binary classification, and 6 regression tasks. On GraphLand datasets, we use the released RL (random low) split, and on classic datasets, we use the same predefined random split with 10%, 10%, and 80% of nodes used for training, validation, and testing, respectively. We consider the GraphLand datasets especially interesting for our study: unlike the classic datasets, where node features are usually homogeneous bag-of-words counters or embeddings, they describe nodes with rich heterogeneous features that mix numerical and categorical attributes collected from real-world industrial applications. Even after the standard preprocessing, which includes quantile normalization of numerical features and one-hot encoding of categorical ones, such inputs still have to be properly transformed by the model itself, and this is exactly the purpose of GNN backbone. Thus, we consider the choice of backbone layout to be especially important on these datasets, which is supported by our results in Section 5 and Section 7.

Aggregated metrics To compare the considered approaches across many datasets at once, we use four aggregate measures. *Fair Rank* (FR) is a variant of rank that accounts for the variance of the evaluation results. On each dataset, we sort the methods by their mean test score, assign rank 1 to the leading method, and keep the current rank for every next method whose mean score stays within one standard deviation of the current leader; otherwise, the method receives the next rank and becomes the new reference. The resulting ranks are averaged over datasets, and lower values are better. *Normalized Score* (NS) is obtained by min–max normalizing the mean scores of the compared methods within each dataset and averaging them over datasets, and higher values are better. *Elo Rating* converts the seed-level results into pairwise comparisons, weights all datasets equally, and fits a Bradley–Terry model following the implementation from Erickson et al. (2025); the reference method of each comparison has the rating of 1000, and higher values are better. *Relative Improvement* (RI) is a per-dataset measure that we use in the ablation study to show the effect of fixing a particular component: it equals $100 \times (s/s_{\text{tuned}} - 1)$, where s and s_{tuned} are the mean scores of the considered method and its fully tuned counterpart. All these measures are computed within a particular pool of compared methods and datasets, so their values are not transferable between different comparisons. For FR and NS, we also report the standard deviation across datasets, and for Elo Rating, the standard deviation obtained by refitting the model on bootstrap samples of datasets.

4 Ablating the main distinctive mGNN components

In this section, we investigate how proper tuning of backbone layout and message-passing mechanism affects the predictive performance of mGNN. For each of these components, we fix the values of its associated hyperparameters and run independent tuning on the resulting reduced search space.

To ablate backbone layout, we take three representative configurations: Classic main module with normalization on position 3, Multilayer main module with normalization on position 1, and Transformer main module with normalization on positions 1 and 2. As can be seen from Figure 2 (*Top section*), tuning backbone layout brings notable performance gains compared to these fixed options. None of them is universally suitable: each of them matches or outperforms the tuned counterpart on a noticeable share of the considered datasets, but also loses several percent of predictive performance in the worst case. Thus, the observed performance gain is largely caused by a form of amortization that helps avoid particularly unfavorable options for the given task.

To ablate the message-passing mechanism, we compare four base options for aggregation function: SAGE (Hamilton et al., 2017), GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), LGT (Platonov et al., 2023). As shown in Figure 2 (*Middle section*), tuning the aggregation function does not yield a massive performance gap compared to the individual options, but it still delivers the best aggregated performance, while allowing a *fourfold reduction* in the number of tuning trials compared to tuning each option separately. Here we observe the same pattern as with backbone layout: every fixed aggregation function is competitive on a substantial part of the datasets, but noticeably falls behind on the remaining ones, so selecting it for the given prediction task allows the

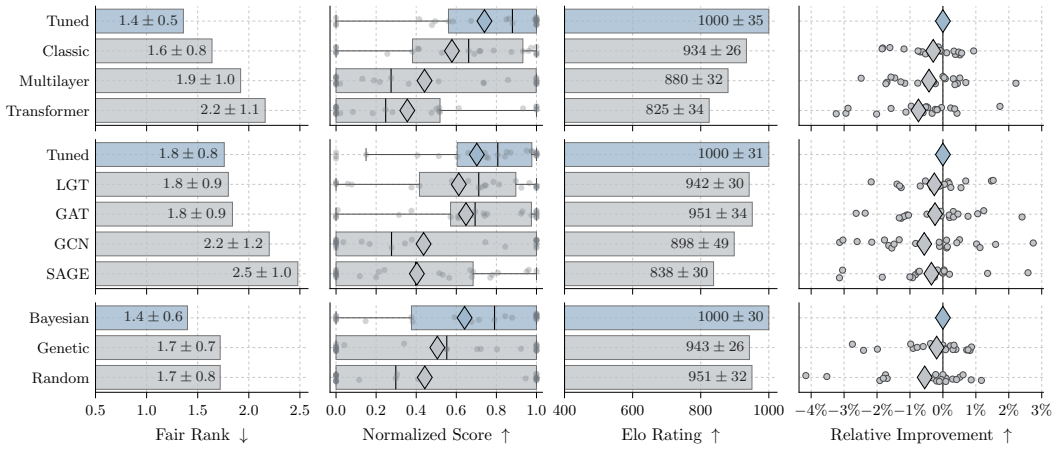

Figure 2: Ablation study for the mGNN components: backbone layout, message-passing mechanism, and Bayesian optimization. Refer to Section 3 for the definitions of aggregated metrics.

model to stay close to the best available option. We view this as a valuable practical finding that can help significantly reduce the computational cost of GNN model selection.

Finally, we replace the TPE sampler in our tuning procedure with either standard random sampler or genetic NSGAII sampler (Deb et al., 2002), keeping the search space and the tuning budget exactly the same. As shown in Figure 2 (*Bottom section*), TPE achieves the best aggregated results, while random and genetic samplers perform similarly to each other and fall behind it by a noticeable margin. We conclude that, on such a complex search space and within the same tuning budget, Bayesian optimization provides a more reliable model selection procedure and can serve as a suitable off-the-shelf solution for constructing a strong GNN baseline. The exact values of the aggregated metrics, computed separately for the two groups of graph datasets, are provided in Appendix D.

5 Analyzing the importance of mGNN tuning options

In this section, we look more closely at the tuning traces produced by our procedure to understand which options actually drive the difference in predictive performance. For each dataset, we compute the fANOVA importance (Hutter et al., 2014) of every tuning option on the completed trials of the corresponding mGNN tuning study and normalize the obtained values to sum to one. The results are provided on Figure 3, where the options are grouped into five components: backbone layout, message-passing mechanism, training setup, model capacity, and other hyperparameters. Note that this analysis is purely descriptive: it characterizes the configurations visited by our tuning procedure and should not be read as a universal, causal measure of importance. In particular, fANOVA estimates only first-order effects and does not account for the interactions between different options.

The first observation is that the learning rate is the single most important option, which is completely expected and agrees with the common practice of tuning it in the first place. However, the options associated with GNN architecture design also bring a significant portion of importance: normalization placement is the second most important option overall, and together, backbone layout and message-passing mechanism account for a share comparable to the entire training setup, exceeding it on almost a half of the datasets. The second observation is that these values vary substantially across datasets, which again indicates that the relevant aspects of model selection depend on the given task: on some datasets the outcome is almost entirely determined by the training setup, while on others backbone layout alone accounts for more than 40% of the total importance. This variation is also visible between the two groups of datasets compared on Figure 5 in Appendix B: on GraphLand datasets, backbone layout receives almost twice the importance it has on classic ones, while the training setup becomes relatively less influential, which agrees with our expectations from Section 3.

Finally, we check how these importance values translate into the actual predictive performance. On the datasets where the ablated components receive a high importance share, fixing a single option can cost from one to several percent of predictive performance even though the rest of the search space is tuned as before, and the most notable decreases reach -5.50% for the Multilayer main module

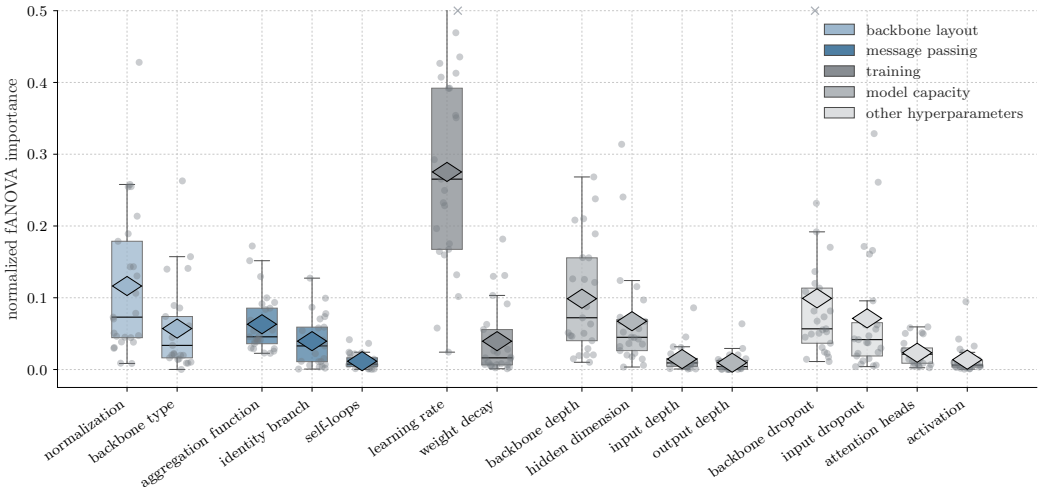

Figure 3: Importance of mGNN tuning options estimated with fANOVA and normalized to sum to one within a dataset. Each box shows the distribution over datasets, points correspond to individual datasets, and rhombuses mark the mean values. The options are grouped and colored by component. The values exceeding the axis limit are drawn as crosses on its boundary.

type on `city-roads-L`, -7.58% for the SAGE aggregation function on `twitch-views`, and -8.58% for the LGT aggregation function on `avazu-ctr` (see Table 1 in Appendix B). These examples show that the aggregated gains reported in Section 4 are not a result of averaging over many small differences, but rather a consequence of avoiding notable failures on the datasets where the corresponding component especially matters.

6 Enhancing mGNN with generalized aggregation function

In the mGNN architecture described in Section 2, the aggregation function is represented by a categorical hyperparameter with several predefined options. This is a natural first step, but it also has an evident limitation: the tuning procedure can only choose one of a few discrete values, while the aggregation functions behind them can be seen as particular cases of a more general computation. Thus, we take our idea one step further and replace this categorical hyperparameter with a single generalized aggregation function that has a small number of continuous knobs, so the tuning procedure can place the model either close to one of the known aggregation functions or somewhere in between them. This function is based on local self-attention and extends it with two additional hyperparameters and one optional modification. For each node v with the neighborhood $\mathcal{N}(v)$, it computes

$$e_{uv} = \frac{\boldsymbol{q}_v^\top \boldsymbol{k}_u}{\sqrt{d_{\text{head}}}}, \quad \alpha_{uv} = \frac{\exp(e_{uv}/\tau)}{\exp(s) + \sum_{w \in \mathcal{N}(v)} \exp(e_{wv}/\tau)}, \quad \boldsymbol{h}_v = \sum_{u \in \mathcal{N}(v)} \gamma_{uv}\, \alpha_{uv}\, \boldsymbol{v}_u,$$

where $\boldsymbol{q}_v$, $\boldsymbol{k}_u$, and $\boldsymbol{v}_u$ are the query, key, and value projections of node representations, d_{head} is the dimension of a single attention head, and the results of all heads are concatenated. The first knob is the temperature τ, which controls how sharp the attention distribution is. The second knob is the sink s, which adds a constant term to the softmax denominator and allows the node to leave a part of the attention mass unassigned, passing the neighborhood signal only partially. The last knob is an optional factor γ_{uv}, which equals 1 if disabled and $\sqrt{d_v/d_u}$ with $d_x = |\mathcal{N}(x)|$ otherwise.

Despite its simplicity, this parametrization covers a rather diverse set of message-passing behaviors. Depending on the position of the knobs, the function can approach mean aggregation following SAGE, symmetrically normalized mean aggregation used in GCN, sum aggregation, ordinary local self-attention used in LGT, or hard selection of the single most relevant neighbor. The intermediate positions produce the behavior between these regimes that cannot be expressed by any of the standard aggregation functions considered before: for example, attention that suppresses the neighborhood signal. We summarize these regimes in Table 2 and explain them in more detail in Appendix B. We note that this family is built around dot-product attention and degree-based normalization, so it does

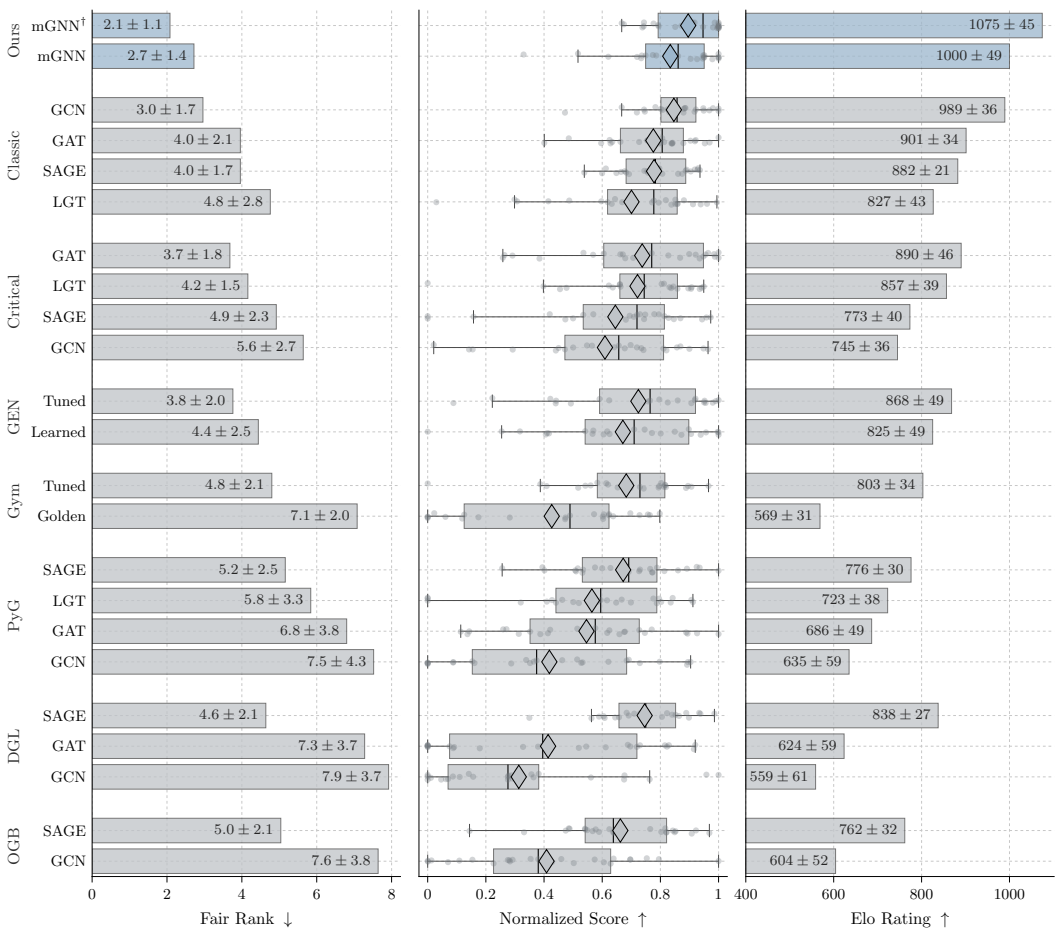

Figure 4: Comparison of the proposed mGNN and mGNN† with other GNN baselines and implementations. Refer to Section 3 for the definitions of aggregated metrics.

not include GAT as a special case. In our model selection procedure, temperature and sink are treated as numerical hyperparameters and degree adjustment as a boolean one. They are included in the search space and jointly tuned with the other hyperparameters (the corresponding ranges are provided in Appendix C). We refer to the resulting model as mGNN† and evaluate it in Section 7.

7 Comparison with other GNN baselines

In this section, we compare our approach with other GNN baselines and implementations introduced in previous graph machine learning studies. Besides mGNN and mGNN†, we consider 21 methods from 7 different sources. The first group consists of the reference implementations shipped with the popular graph deep learning packages: PyTorch Geometric (PyG, Fey & Lenssen, 2019), Deep Graph Library (DGL, Wang et al., 2019), and Open Graph Benchmark (OGB, Hu et al., 2020). For these methods, we tune only the basic hyperparameters, as it is usually done when they are taken as baselines. The second group consists of the model families from the works that specifically studied GNN design and evaluation: the *Classic* architectures by Luo et al. (2024) and the *Critical* baselines by Platonov et al. (2023), both of which additionally tune several architectural hyperparameters. The third group covers two variants of the model selection approach from the GraphGym work (You et al., 2020): the condensed, golden set of GNN architectures from their design space and Bayesian optimization over the same space. The last group is represented by the Generalized Aggregation Networks (GEN, Li et al., 2020), with their inverse temperature either learned at the training stage or selected at the tuning stage. All the compared methods follow the evaluation protocol from Section 3. Note that the provided budget for the number of trials in our study is more generous than in the original works behind the Classic and Critical baseline groups: the former selected its configurations

from a small predefined grid of values, and the latter tuned only the number of message-passing layers, keeping the remaining hyperparameters fixed across all models and datasets. Thus, in our comparison, these model families are tuned with Bayesian optimization over notably broader search spaces than in the studies that introduced them.

The results are provided on Figure 4, where mGNN serves as the reference method. mGNN† achieves the best values of all three aggregated metrics, and mGNN is the runner-up by two of them, slightly falling behind the strong GCN baseline of Luo et al. (2024) in terms of normalized score. This indicates that the proposed model selection procedure is competitive with the considered alternatives, while the generalized aggregation function brings further improvement. However, this advantage is distributed very unevenly across the graph datasets (the aggregated metrics for each group and the results on every individual dataset are reported in Appendix D). On classic datasets, the results of the leading methods are quite close to each other, which is not surprising given how long these datasets have been studied. On GraphLand datasets, both our models outperform every considered baseline by a large margin. This agrees with our observations in Section 3 and Section 5: on these datasets, predictive performance depends more heavily on how node features are transformed inside the backbone, which is exactly the part of GNN architecture that the compared baselines keep fixed.

Another important observation concerns the methods that share the same aggregation function but come from different sources: their results differ dramatically, often more than the results of different aggregation functions within one source. For example, the GCN models from the five providing sources occupy almost the entire range of the obtained Elo ratings. Thus, the name of an aggregation function says very little about the result one should expect from a GNN model, since the surrounding backbone layout, training setup, and tuning protocol contribute at least as much. This is particularly evident for the two GraphGym variants: replacing their golden set of configurations with Bayesian optimization over the same design space changes the outcome substantially. We consider this an additional argument for treating GNN model selection as a part of the method rather than a technical detail of the experimental setup.

Finally, no considered method wins on every dataset, and our approach is not an exception: the detailed results in Appendix D show that several baselines remain competitive on a notable part of the benchmark. Our claim is that, under an equal budget and an equal evaluation protocol, jointly tuning backbone layout, message-passing mechanism, and other basic hyperparameters gives a more reliable way to obtain a strong GNN baseline for the given prediction task.

8 Conclusion & Future Work

In this work, we make a step beyond the existing practices in GNN model selection and show how to construct a modern GNN baseline by including backbone layout, message-passing mechanism, and other basic hyperparameters into a single search space and exploring it with a standard Bayesian optimization technique. Tuning these components is notably more effective than fixing any of them, as their incorrect choice can cost several percent of predictive performance. The resulting models outperform many baselines from prior works and graph deep learning packages, especially on the datasets with rich heterogeneous node features, and the generalized aggregation function further improves the prediction quality, which indicates that different prediction tasks may require finer tuning of the message-passing behavior. We discuss the limitations of our study in Appendix A.

Our results also suggest several recommendations for future work in the field. Researchers working on new graph-based aggregation functions can include them in a search space similar to ours and check whether the tuning procedure prefers them to the other available options; those working on new backbone layouts can evaluate them within the same framework, since an improvement obtained under a properly invested tuning budget is strong evidence that the finding is reliable in practice; those working on new blackbox optimization methods can apply them to GNN model selection, where the preferable configurations vary across prediction tasks and make the landscape of this tuning problem an interesting object of study. Finally, an important direction is how to reduce the cost of the model selection procedure. It requires many tuning trials with many training steps, and a single step can be quite expensive on large and dense graphs, especially when message passing is based on self-attention. However, a recent work (Fomina et al., 2026) on hardware-efficient implementations of graph-based operations targets exactly this problem and achieves notable improvements in speed and memory consumption, so we believe the computational cost will gradually decrease.

AI USE STATEMENT

We used generative AI tools to help write and debug the code for running experiments and producing figures and tables, to search and check related literature, and to draft and edit parts of the text. All code was reviewed and tested by the authors, all references were checked against the original sources, and all text was reviewed and revised by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies model selection for graph neural networks on publicly available graph datasets. We do not collect new data and do not conduct studies with human subjects. We are not aware of ethical concerns specific to this work beyond the general risks of applying graph machine learning models in practice.

REPRODUCIBILITY STATEMENT

We release the code, the configurations of all experiments, and the reports with all obtained results at this repository. The evaluation protocol, datasets, preprocessing, and data splits are described in Section 3, the complete hyperparameter search spaces are provided in Appendix C, and the results on every dataset are provided in Appendix D. The repository contains the scripts that generate the configurations of every experiment and a single script that runs tuning, evaluation, and ensembling for a given configuration.

REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2623–2631, 2019. 4

Adrian Arnaiz-Rodriguez and Federico Errica. Oversmoothing," oversquashing", heterophily, long-range, and more: Demystifying common beliefs in graph machine learning. In *International Conference on Learning Representations*, volume 2026, pp. 62884–62910, 2026. 1

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 2

Gleb Bazhenov, Oleg Platonov, and Liudmila Prokhorenkova. GraphLand: Evaluating graph machine learning models on diverse industrial data. *Advances in Neural Information Processing Systems*, 2025. 5

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems*, 24, 2011. 4

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022. 1

Shaofei Cai, Liang Li, Jincan Deng, Beichen Zhang, Zheng-Jun Zha, Li Su, and Qingming Huang. Rethinking graph neural architecture search from message-passing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6653–6662. IEEE, 2021. 2

Jiamin Chen, Jianliang Gao, Yibo Chen, Moctard Babatounde Oloulade, Tengfei Lyu, and Zhao Li. GraphPAS: Parallel architecture search for graph neural networks. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2182–2186, 2021. 2

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pp. 1725–1735, 2020. 2

Tianlong Chen, Kaixiong Zhou, Keyu Duan, Wenqing Zheng, Peihao Wang, Xia Hu, and Zhangyang Wang. Bag of tricks for training deeper graph neural networks: A comprehensive benchmark study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):2769–2781, 2022. 1, 2

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2): 182–197, 2002. 6

Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Mutalik Desai, David Salinas, and Frank Hutter. TabArena: A living benchmark for machine learning on tabular data. *arXiv preprint arXiv:2506.16791*, 2025. 5

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with PyTorch Geometric. In *Representation Learning on Graphs and Manifolds Workshop at The Seventh International Conference on Learning Representations*, 2019. 8

Daria Fomina, Daniil Krasylnikov, Alexey Boykov, Andrey Dolgovyazov, Vyacheslav Zhdanovskiy, and Fedor Velikonivtsev. On efficient scaling of GNNs via IO-aware layers implementations. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=w6JVbu7VFo. 9, 14

Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. Graph neural architecture search. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 2020. 2

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30, 2017. 2, 4, 5

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778. IEEE, 2016. 2

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in Neural Information Processing Systems*, 33:22118–22133, 2020. 8

Zhao Huan, Yao Quanming, and TU Weiwei. Search to aggregate neighborhood for graph neural network. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pp. 552–563. IEEE, 2021. 2

Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *International Conference on Machine Learning*, pp. 754–762. PMLR, 2014. 6

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. 2, 4, 5

Niklas Kormann, Benjamin Doerr, and Johannes F Lutzeyer. Position: Don't be afraid of oversmoothing and over-squashing. *arXiv preprint arXiv:2601.07419*, 2026. 1

Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. DeeperGCN: All you need to train deeper GCNs. *arXiv preprint arXiv:2006.07739*, 2020. 2, 8

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in Neural Information Processing Systems*, 31, 2018. 14

Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Classic GNNs are strong baselines: Reassessing GNNs for node classification. *Advances in Neural Information Processing Systems*, 37:97650–97669, 2024. 1, 2, 4, 8, 9

Péter Mernyei and Cătălina Cangea. Wiki-CS: A Wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*, 2020. 5

Matheus Nunes and Gisele L Pappa. Neural architecture search in graph neural networks. In *Brazilian conference on intelligent systems*, pp. 302–317. Springer, 2020. 2

MoonJeong Park, Sunghyun Choi, Jaeseung Heo, Eunhyeok Park, and Dongwoo Kim. The over-smoothing fallacy: A misguided narrative in GNN research. *arXiv preprint arXiv:2506.04653*, 2025. 1

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *International Conference on Learning Representations*, 2023. 4, 5, 8

Yasha Pushak and Holger Hoos. AutoML loss landscapes. *ACM Transactions on Evolutionary Learning and Optimization*, 2(3):1–30, 2022. 14

Yijian Qin, Ziwei Zhang, Xin Wang, Zeyang Zhang, and Wenwu Zhu. NAS-Bench-Graph: Benchmarking graph neural architecture search. *Advances in neural information processing systems*, 35: 54–69, 2022. 2

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2), 2021. 5

Lennart Schneider, Lennart Schäpermeier, Raphael Patrick Prager, Bernd Bischl, Heike Trautmann, and Pascal Kerschke. HPO $\times$ ELA: Investigating hyperparameter optimization landscapes by means of exploratory landscape analysis. In *Parallel Problem Solving from Nature – PPSN XVII*, Lecture Notes in Computer Science, pp. 575–589. Springer, 2022. 14

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–93, 2008. 4

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. In *Relational Representation Learning Workshop at NeurIPS*, 2018. 5

Min Shi, Yufei Tang, Xingquan Zhu, Yu Huang, David Wilson, Yuan Zhuang, and Jianxun Liu. Genetic-GNN: Evolutionary architecture search for graph neural networks. *Knowledge-based systems*, 247:108752, 2022. 2

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. 2

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017. 4

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. 2, 4, 5

Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. In *Representation Learning on Graphs and Manifolds Workshop at The Seventh International Conference on Learning Representations*, 2019. 8

Colin White, Sam Nolen, and Yash Savani. Exploring the loss landscape in neural architecture search. In *Uncertainty in Artificial Intelligence*, pp. 654–664. PMLR, 2021. 14

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, pp. 5453–5462. pmlr, 2018. 1, 2

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. 1, 2

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International Conference on Machine Learning*, pp. 40–48, 2016. 4

Jiaxuan You, Zhitao Ying, and Jure Leskovec. Design space for graph neural networks. *Advances in Neural Information Processing Systems*, 33:17009–17021, 2020. 1, 2, 8

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graph-SAINT: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2020. 5

Kaixiong Zhou, Xiao Huang, Qingquan Song, Rui Chen, and Xia Hu. Auto-GNN: Neural architecture search of graph neural networks. *Frontiers in big Data*, 5:1029307, 2022. 2

A LIMITATIONS

Our study is only one particular example of how a modern approach to GNN model selection can be implemented, and each of its components can be extended or replaced. We consider three types of the main module and several normalization placements, but they just represent the design patterns common in the literature, and other backbone layouts can be added to the search space in the same way. The same holds for the message-passing mechanism, where many options are left outside of our search space, and for the configuration sampler, which can be replaced with a more advanced blackbox optimization method.

Next, we do not investigate the landscape of validation performance at the tuning stage, similar to the research on neural network loss landscapes (Li et al., 2018). Such landscapes have already been studied for hyperparameter optimization problems (Pushak & Hoos, 2022; Schneider et al., 2022) and neural architecture search (White et al., 2021), and it would be interesting to see how the landscape of GNN model selection compares to them. We find this direction particularly promising, since samplers that exploit the nature of this landscape could make the whole procedure considerably more efficient.

Further, all our conclusions are obtained under a fixed tuning budget of 100 trials, of which the first 25 are spent on startup sampling. This budget is the same for all the compared methods, which makes the comparison fair, but we do not study how the results change with the budget, and the relative standing of the compared methods and samplers may differ under a smaller or a larger one.

Finally, the proposed tuning procedure requires a noticeable amount of computational resources: it conducts many tuning trials with many training steps, and a single step can be quite expensive on large and dense graphs, especially when message passing is based on self-attention. Yet, some recent works on hardware-efficient implementations of graph-based operations (Fomina et al., 2026) can be viewed as a promising way of reducing this cost.

B ADDITIONAL ANALYSIS

Component importance on dataset groups In this section, we complement the analysis of tuning traces from Section 5 and compare the obtained importance values between the two groups of graph datasets. In particular, we sum the fANOVA importances of the tuning options within each component and report the resulting distributions in Figure 5.

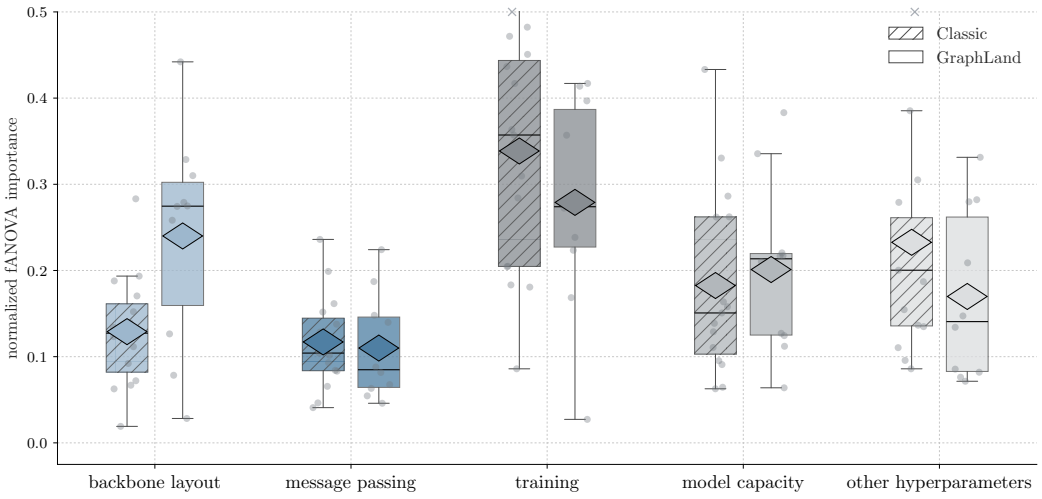

Figure 5: Importance of the mGNN components grouped on classic and GraphLand datasets. Backbone layout accounts for a notably larger share of importance on GraphLand datasets, while training setup takes a somewhat larger share on classic ones. The component importances sum the fANOVA importances of the corresponding tuning options. Each box shows the distribution over the datasets of the group, points correspond to individual datasets, and the rhombus marks the mean value. The values exceeding the axis limit are placed on its boundary and drawn as crosses.

Fixed options on datasets with high component importance To connect the importance values from Section 5 with the actual predictive performance, we take every fixed option from our ablation study in Section 4 and select a dataset on which the corresponding component receives a high importance share, requiring all the selected datasets to be distinct. For each of these cases, we report the result of the independently retuned experiment with this option kept fixed in Table 1.

Table 1: Predictive performance lost by fixing a single option on a dataset where the corresponding component is important. For every fixed option, we report one of the most pronounced cases. Importance share is the part of the total fANOVA importance that falls on the component.

Family	Dataset	Importance Share	Fixed Option	Relative Improvement ↑
backbone layout	`hm-categories`	25.8%	Classic	−1.83%
	`city-roads-L`	44.2%	Multilayer	−5.50%
	`hm-prices`	27.5%	Transformer	−2.94%
message passing	`twitch-views`	14.0%	SAGE	−7.58%
	`artnet-views`	18.7%	GCN	−5.35%
	`amazon-ratings-full`	19.9%	GAT	−1.33%
	`avazu-ctr`	22.4%	LGT	−8.58%

Regimes of generalized aggregation function Here, we explain in more detail how the generalized aggregation function introduced in Section 6 reproduces the message-passing behaviors listed in Table 2. When the temperature grows, $\tau \to \infty$, all terms $\exp(e_{uv}/\tau)$ tend to 1, and the attention weights become $\alpha_{uv} = 1/(\exp(s) + d_v)$, which no longer depend on the neighbor u. If the sink is small, $s \to -\infty$, this gives $\alpha_{uv} = 1/d_v$, which corresponds to mean aggregation, and with the degree adjustment enabled, the resulting coefficient becomes $\gamma_{uv}\alpha_{uv} = 1/\sqrt{d_u d_v}$, which corresponds to the symmetric normalization used in GCN. If the sink is large, $s \to +\infty$, the coefficient approaches $\exp(-s)$ for all neighbors, so the output becomes the sum of the neighboring messages up to a constant scale, which is absorbed by the following linear layer. For a moderate sink, the coefficient stays close to $1/d_v$ on the nodes with $d_v \gg \exp(s)$ and becomes notably smaller on the nodes with few neighbors, which results in mean aggregation weakened on such nodes. With $\tau = 1$ and $s \to -\infty$, the function coincides with ordinary local self-attention used in LGT, while a moderate sink allows this attention to suppress the neighborhood signal. Finally, when the temperature decreases, $\tau \to 0$, the softmax concentrates on the neighbor with the largest score e_{uv} and selects it. In practice, all these regimes are only approached, since our search space restricts the temperature to $\tau \in [10^{-1.5}, 10^{1.5}]$ and the sink to $s \in [-5, 5]$.

Table 2: Behavior of the proposed generalized aggregation function for different positions of its knobs. The boundary regimes are only approached, as our search space restricts the temperature to $\tau \in [10^{-1.5}, 10^{1.5}]$ and the sink to $s \in [-5, 5]$. Intermediate positions produce the behavior between the listed regimes.

Temperature τ	Sink s	Degree adjustment	Resulting behavior
$\tau \to \infty$	$s \to -\infty$	no	mean aggregation (SAGE)
$\tau \to \infty$	$s \to -\infty$	yes	symmetrically normalized mean aggregation (GCN)
$\tau \to \infty$	$s \to +\infty$	no	sum aggregation, up to a constant scale
$\tau \to \infty$	moderate	no	mean aggregation weakened on nodes with few neighbors
$\tau = 1$	$s \to -\infty$	no	local self-attention (LGT)
$\tau = 1$	moderate	no	attention that can suppress the neighborhood signal
$\tau \to 0$	$s \to -\infty$	no	hard selection of the single most relevant neighbor

C Hyperparameter search spaces

In this section, we provide the complete hyperparameter search spaces for all the GNN baselines and implementations considered in our study. We use the following notation for distributions:

- Int[`low`, `high`, `step`] — integer hyperparameter sampled from a uniform distribution from `low` to `high` with step `step`.

- Float[low, high, step] — real hyperparameter sampled from a uniform distribution from low to high with step step.
- LogUniform[low, high] — real hyperparameter sampled from a log-uniform distribution from low to high.
- Cat[a, b, c, ...] — categorical hyperparameter sampled from a uniform distribution on unordered values a, b, c, ...

The search spaces are reported in Tables 3 to 9.

Table 3: Hyperparameter search space for PyG baselines. This family treats GNNs with GCN, GAT, SAGE, and LGT aggregation functions as separate methods.

Parameter	Distribution
Learning rate λ	LogUniform[3e-5, 1e-2]
Weight decay	LogUniform[1e-4, 1]
Backbone layers L	Int[1, 10, 1]
Hidden dimension d	Int[96, 768, 32]
Backbone dropout p_{main}	Float[0, 0.5, 0.05]
Attention heads H (only for GAT and LGT)	Cat[2, 4, 8]

Table 4: Hyperparameter search space for DGL baselines. This family treats GNNs with GCN, GAT, and SAGE aggregation functions as separate methods.

Parameter	Distribution
Learning rate λ	LogUniform[3e-5, 1e-2]
Weight decay	LogUniform[1e-4, 1]
Backbone layers L	Int[1, 10, 1]
Hidden dimension d	Int[96, 768, 32]
Backbone dropout p_{main}	Float[0, 0.5, 0.05]
Attention heads H (only for GAT)	Cat[2, 4, 8]

Table 5: Hyperparameter search space for OGB baselines. This family treats GNNs with GCN and SAGE aggregation functions as separate methods.

Parameter	Distribution
Learning rate λ	LogUniform[3e-5, 1e-2]
Weight decay	LogUniform[1e-4, 1]
Backbone layers L	Int[1, 10, 1]
Hidden dimension d	Int[96, 768, 32]
Backbone dropout p_{main}	Float[0, 0.5, 0.05]

Table 6: Hyperparameter search space for Critical baselines. This family treats GNNs with GCN, GAT, SAGE, and LGT aggregation functions as separate methods.

Parameter	Distribution
Learning rate λ	LogUniform[3e-5, 1e-2]
Weight decay	LogUniform[1e-4, 1]
Backbone layers L	Int[1, 10, 1]
Hidden dimension d	Int[96, 768, 32]
Backbone dropout p_{main}	Float[0, 0.5, 0.05]
FFN expansion r_{FFN}	Cat[1, 2, 4]
Normalization	Cat[LayerNorm, BatchNorm, none]
Attention heads H (only for GAT and LGT)	Cat[2, 4, 8]
Separate self/message paths (only for GAT and LGT)	Cat[false, true]

Table 7: Hyperparameter search space for Classic baselines. This family treats GNNs with GCN, GAT, SAGE, and LGT aggregation functions as separate methods.

Parameter	Distribution
Learning rate λ	LogUniform[3e-5, 1e-2]
Weight decay	LogUniform[1e-4, 1]
Backbone layers L	Int[1, 10, 1]
Hidden dimension d	Int[96, 768, 32]
Backbone dropout p_{main}	Float[0, 0.5, 0.05]
Input projection	Cat[false, true]
Residual connections	Cat[false, true]
Normalization	Cat[none, LayerNorm, BatchNorm]
Attention heads H (only for GAT and LGT)	Cat[2, 4, 8]

Table 8: Hyperparameter search space for mGNN models. *For Transformer backbone type, `none` and `post` resolve to `post`, whereas `pre` and `pre-inter` resolve to `pre-inter`.

Parameter	Distribution
Learning rate λ	LogUniform[3e-5, 1e-2]
Weight decay	LogUniform[1e-4, 1]
Backbone type	Cat[Classic, Multilayer, Transformer]
Normalization position*	Cat[none, pre, post, pre-inter]
Aggregation function	Cat[SAGE, GCN, GAT, LGT]
Identity branch	Cat[false, true]
Self-loops	Cat[false, true]
Backbone layers L	Int[1, 10, 1]
Hidden dimension d	Int[96, 768, 32]
Activation	Cat[GELU, ReLU]
Backbone dropout p_{main}	Float[0, 0.5, 0.05]
Attention heads H	Cat[1, 2, 4, 8]
Input layers L_{in}	Int[1, 2, 1]
Input dropout p_{in}	Float[0, 0.5, 0.05]
Output layers L_{out}	Int[1, 2, 1]

Table 9: Hyperparameter search space for mGNN† models. *For Transformer backbone type, `none` and `post` resolve to `post`, whereas `pre` and `pre-inter` resolve to `pre-inter`.

Parameter	Distribution
Learning rate λ	LogUniform[3e-5, 1e-2]
Weight decay	LogUniform[1e-4, 1]
Backbone type	Cat[Classic, Multilayer, Transformer]
Normalization position*	Cat[none, pre, post, pre-inter]
Identity branch	Cat[false, true]
Self-loops	Cat[false, true]
Backbone layers L	Int[1, 10, 1]
Hidden dimension d	Int[96, 768, 32]
Activation	Cat[GELU, ReLU]
Backbone dropout p_{main}	Float[0, 0.5, 0.05]
Attention sink s	Float[−5, 5, 1]
Log-temperature $\log_{10}\tau$	Float[−1.5, 1.5, 0.1]
Attention heads H	Cat[1, 2, 4, 8]
Degree adjustment	Cat[false, true]
Input layers L_{in}	Int[1, 2, 1]
Input dropout p_{in}	Float[0, 0.5, 0.05]
Output layers L_{out}	Int[1, 2, 1]

D ADDITIONAL EXPERIMENTAL RESULTS

In this section, we provide the detailed results of our experiments. Specifically, Tables 10 to 12 report the aggregated metrics separately for the two groups of graph datasets, and Tables 13 to 37 report the predictive performance of every considered method on each individual dataset.

Table 10: Aggregate test performance by dataset group. *Overall* contains 25 datasets available for every method. Metrics are recomputed within each displayed dataset group. Family blocks follow the fixed displayed order; within each family, rows are sorted by ascending overall Fair Rank. mGNN is the Elo anchor at 1000. The best displayed value is colored blue; the runner-up is underlined.

		Fair Rank ↓			Normalized Score ↑			Elo Rating ↑		
Source	Method	GraphLand	Classic	Overall	GraphLand	Classic	Overall	GraphLand	Classic	Overall
Ours	mGNN†	1.80	2.27	2.08	0.963	0.850	0.896	1090	1072	1075
	mGNN	2.50	2.87	2.72	0.916	0.779	0.834	1000	1000	1000
Classic	GCN	3.70	2.47	2.96	0.852	0.843	0.846	836	1065	989
	GAT	4.70	3.47	3.96	0.817	0.748	0.776	789	948	901
	SAGE	4.70	3.47	3.96	0.790	0.770	0.778	733	947	882
	LGT	6.10	3.87	4.76	0.645	0.737	0.701	611	920	827
Critical	GAT	3.10	4.07	3.68	0.866	0.652	0.737	909	874	890
	LGT	4.00	4.27	4.16	0.836	0.645	0.721	819	863	857
	SAGE	5.40	4.60	4.92	0.744	0.580	0.645	637	821	773
	GCN	5.90	5.47	5.64	0.722	0.534	0.609	626	781	745
GEN	Tuned	3.30	4.07	3.76	0.838	0.648	0.724	873	856	868
	Learned	4.30	4.53	4.44	0.783	0.597	0.671	805	820	825
Gym	Tuned	6.00	4.00	4.80	0.733	0.649	0.683	631	872	803
	Golden	7.80	6.60	7.08	0.603	0.309	0.426	440	580	569
PyG	SAGE	5.80	4.73	5.16	0.723	0.638	0.672	616	836	776
	LGT	7.70	4.60	5.84	0.493	0.612	0.564	501	807	723
	GAT	9.40	5.07	6.80	0.454	0.607	0.546	272	839	686
	GCN	11.20	5.07	7.52	0.182	0.576	0.418	39	822	635
DGL	SAGE	6.10	3.67	4.64	0.724	0.760	0.746	615	937	838
	GAT	8.80	6.27	7.28	0.431	0.403	0.414	334	724	624
	GCN	10.20	6.40	7.92	0.248	0.357	0.313	198	677	559
OGB	SAGE	5.10	5.00	5.04	0.777	0.586	0.662	694	773	762
	GCN	10.60	5.67	7.64	0.231	0.527	0.409	130	760	604

Table 11: Aggregate controlled backbone layout ablation by dataset group (GraphLand $n = 10$, Classic $n = 15$, Overall $n = 25$). Metrics are recomputed within each group. Rows are sorted by ascending overall Fair Rank. Every fixed choice receives an independent 100-trial search over the remaining parameters. Elo is anchored to the tuned row at 1000. The best displayed value is colored blue; the runner-up is underlined.

	Fair Rank ↓			Normalized Score ↑			Elo Rating ↑		
BL	GraphLand	Classic	Overall	GraphLand	Classic	Overall	GraphLand	Classic	Overall
Tuned	1.30	1.40	1.36	0.893	0.639	0.740	1000	1000	1000
Classic	1.80	1.53	1.64	0.629	0.544	0.578	842	978	934
Multilayer	2.40	1.60	1.92	0.258	0.564	0.442	683	981	880
Transformer	2.60	1.87	2.16	0.271	0.412	0.355	644	914	825

Table 12: Aggregate controlled message passing ablation by dataset group (GraphLand $n = 10$, Classic $n = 15$, Overall $n = 25$). Metrics are recomputed within each group. Rows are sorted by ascending overall Fair Rank. Every fixed choice receives an independent 100-trial search over the remaining parameters. Elo is anchored to the tuned row at 1000. The best displayed value is colored blue; the runner-up is underlined.

	Fair Rank ↓			Normalized Score ↑			Elo Rating ↑		
MP	GraphLand	Classic	Overall	GraphLand	Classic	Overall	GraphLand	Classic	Overall
Tuned	1.80	1.73	1.76	0.754	0.668	0.702	1000	1000	1000
LGT	2.10	1.60	1.80	0.608	0.616	0.612	874	982	942
GAT	1.80	1.87	1.84	0.733	0.592	0.648	947	952	951
GCN	2.50	2.00	2.20	0.399	0.462	0.437	846	927	898
SAGE	3.20	2.00	2.48	0.187	0.548	0.403	681	923	838

Table 13: Full results on `tolokers-2`. *(Left)* Controlled backbone layout ablation and *(Middle)* controlled message passing ablation (Section 4). *(Right)* Baseline comparison (Section 7). Values are test score mean ± standard deviation over five evaluation seeds, multiplied by 100. Within every table, the best result is colored blue and the runner-up is underlined, both up to the standard deviation of the leader.

Tuned	54.73 ± 0.65
Classic	54.79 ± 0.47
Multilayer	53.80 ± 0.34
Transformer	55.67 ± 0.20

Tuned	54.73 ± 0.65
SAGE	54.72 ± 0.50
GCN	55.29 ± 0.24
GAT	57.21 ± 0.43
LGT	58.07 ± 0.37

Ours	mGNN†	55.47 ± 0.76
	mGNN	54.73 ± 0.65
Classic	GCN	54.68 ± 0.40
	GAT	57.62 ± 0.85
	SAGE	54.82 ± 0.46
	LGT	56.94 ± 0.27
Critical	GAT	58.94 ± 0.91
	LGT	56.20 ± 0.32
	SAGE	53.80 ± 0.62
	GCN	53.52 ± 0.61
GEN	Tuned	56.58 ± 0.82
	Learned	56.80 ± 0.39
Gym	Tuned	57.06 ± 0.59
	Golden	54.97 ± 1.18
PyG	SAGE	55.12 ± 0.31
	LGT	55.63 ± 0.37
	GAT	46.48 ± 2.13
	GCN	43.60 ± 0.92
DGL	SAGE	55.67 ± 0.65
	GAT	50.86 ± 0.55
	GCN	48.12 ± 0.54
OGB	SAGE	55.94 ± 0.33
	GCN	42.11 ± 0.78

Table 14: Full results on `city-reviews`. *(Left)* Controlled backbone layout ablation and *(Middle)* controlled message passing ablation (Section 4). *(Right)* Baseline comparison (Section 7). Values are test score mean ± standard deviation over five evaluation seeds, multiplied by 100. Within every table, the best result is colored blue and the runner-up is underlined, both up to the standard deviation of the leader.

Tuned	78.66 ± 0.07
Classic	78.62 ± 0.06
Multilayer	78.06 ± 0.28
Transformer	78.11 ± 0.14

Tuned	78.66 ± 0.07
SAGE	78.64 ± 0.07
GCN	78.75 ± 0.05
GAT	78.62 ± 0.12
LGT	78.41 ± 0.14

Ours	mGNN†	78.76 ± 0.07
	mGNN	78.66 ± 0.07
Classic	GCN	78.53 ± 0.08
	GAT	78.40 ± 0.10
	SAGE	78.40 ± 0.14
	LGT	78.38 ± 0.27
Critical	GAT	78.42 ± 0.18
	LGT	78.03 ± 0.13
	SAGE	77.54 ± 0.15
	GCN	76.59 ± 0.35
GEN	Tuned	78.40 ± 0.16
	Learned	78.35 ± 0.38
Gym	Tuned	77.87 ± 0.10
	Golden	76.91 ± 0.24
PyG	SAGE	77.84 ± 0.13
	LGT	78.07 ± 0.10
	GAT	75.79 ± 0.09
	GCN	74.43 ± 0.10
DGL	SAGE	78.32 ± 0.03
	GAT	74.81 ± 0.12
	GCN	74.70 ± 0.27
OGB	SAGE	78.12 ± 0.06
	GCN	74.90 ± 0.19

Table 15: Full results on `artnet-exp`. *(Left)* Controlled backbone layout ablation and *(Middle)* controlled message passing ablation (Section 4). *(Right)* Baseline comparison (Section 7). Values are test score mean ± standard deviation over five evaluation seeds, multiplied by 100. Within every table, the best result is colored blue and the runner-up is underlined, both up to the standard deviation of the leader.

Tuned	49.07 ± 0.36
Classic	48.49 ± 0.37
Multilayer	48.35 ± 0.55
Transformer	48.08 ± 0.71

Tuned	49.07 ± 0.36
SAGE	47.52 ± 0.52
GCN	47.79 ± 0.15
GAT	49.58 ± 0.29
LGT	48.38 ± 0.23

Ours	mGNN†	49.77 ± 0.55
	mGNN	49.07 ± 0.36
Classic	GCN	48.15 ± 0.29
	GAT	48.25 ± 0.39
	SAGE	46.42 ± 0.38
	LGT	48.04 ± 0.45
Critical	GAT	49.43 ± 0.19
	LGT	49.15 ± 0.67
	SAGE	48.13 ± 0.58
	GCN	45.46 ± 0.74
GEN	Tuned	47.20 ± 0.36
	Learned	46.12 ± 0.21
Gym	Tuned	45.80 ± 0.51
	Golden	41.90 ± 1.49
PyG	SAGE	45.07 ± 0.68
	LGT	46.59 ± 0.12
	GAT	45.14 ± 0.18
	GCN	42.97 ± 0.20
DGL	SAGE	45.85 ± 0.29
	GAT	40.23 ± 0.24
	GCN	40.68 ± 0.28
OGB	SAGE	45.40 ± 0.18
	GCN	42.86 ± 0.88

Table 16: Full results on `hm-categories`. *(Left)* Controlled backbone layout ablation and *(Middle)* controlled message passing ablation (Section 4). *(Right)* Baseline comparison (Section 7). Values are test score mean ± standard deviation over five evaluation seeds, multiplied by 100. Within every table, the best result is colored blue and the runner-up is underlined, both up to the standard deviation of the leader.

Tuned	75.07 ± 0.27
Classic	73.70 ± 0.15
Multilayer	$\underline{74.20 \pm 0.38}$
Transformer	71.76 ± 0.37

Tuned	75.07 ± 0.27
SAGE	70.11 ± 0.43
GCN	68.73 ± 0.52
GAT	75.22 ± 0.60
LGT	$\underline{74.56 \pm 0.24}$

Ours	mGNN†	72.52 ± 0.69
	mGNN	75.07 ± 0.27
Classic	GCN	70.00 ± 0.66
	GAT	70.05 ± 0.89
	SAGE	69.09 ± 0.52
	LGT	53.39 ± 0.36
Critical	GAT	$\underline{73.46 \pm 0.27}$
	LGT	72.15 ± 0.71
	SAGE	64.03 ± 1.87
	GCN	69.23 ± 0.12
GEN	Tuned	$\underline{73.70 \pm 0.51}$
	Learned	73.17 ± 0.60
Gym	Tuned	69.29 ± 0.20
	Golden	63.41 ± 0.39
PyG	SAGE	69.31 ± 0.35
	LGT	44.16 ± 0.17
	GAT	61.97 ± 0.17
	GCN	60.05 ± 0.61
DGL	SAGE	70.14 ± 0.40
	GAT	72.60 ± 0.68
	GCN	65.08 ± 0.20
OGB	SAGE	68.98 ± 0.74
	GCN	61.91 ± 1.79

Table 17: Full results on `hm-prices`. *(Left)* Controlled backbone layout ablation and *(Middle)* controlled message passing ablation (Section 4). *(Right)* Baseline comparison (Section 7). Values are test score mean ± standard deviation over five evaluation seeds, multiplied by 100. Within every table, the best result is colored blue and the runner-up is underlined, both up to the standard deviation of the leader.

Tuned	74.17 ± 0.70
Classic	74.55 ± 0.29
Multilayer	74.55 ± 0.30
Transformer	71.99 ± 0.58

Tuned	74.17 ± 0.70
SAGE	73.55 ± 0.40
GCN	72.95 ± 0.27
GAT	75.95 ± 0.27
LGT	74.68 ± 0.62

Ours	mGNN†	75.31 ± 0.52
	mGNN	74.17 ± 0.70
Classic	GCN	71.68 ± 0.50
	GAT	67.17 ± 0.10
	SAGE	73.39 ± 0.32
	LGT	67.95 ± 0.28
Critical	GAT	75.43 ± 0.22
	LGT	73.89 ± 0.47
	SAGE	75.02 ± 0.09
	GCN	71.25 ± 0.57
GEN	Tuned	75.66 ± 0.12
	Learned	76.00 ± 0.38
Gym	Tuned	72.20 ± 0.18
	Golden	66.18 ± 0.28
PyG	SAGE	73.64 ± 0.38
	LGT	61.35 ± 0.06
	GAT	66.14 ± 0.61
	GCN	39.84 ± 1.34
DGL	SAGE	71.08 ± 0.24
	GAT	64.71 ± 0.24
	GCN	44.95 ± 0.04
OGB	SAGE	73.19 ± 0.30
	GCN	42.37 ± 1.31

Table 18: Full results on `avazu-ctr`. *(Left)* Controlled backbone layout ablation and *(Middle)* controlled message passing ablation (Section 4). *(Right)* Baseline comparison (Section 7). Values are test score mean ± standard deviation over five evaluation seeds, multiplied by 100. Within every table, the best result is colored blue and the runner-up is underlined, both up to the standard deviation of the leader.

Tuned	32.56 ± 0.32
Classic	32.48 ± 0.10
Multilayer	30.37 ± 1.56
Transformer	31.51 ± 0.15

Tuned	32.56 ± 0.32
SAGE	32.32 ± 0.16
GCN	33.09 ± 0.10
GAT	31.80 ± 0.13
LGT	29.77 ± 0.87

Ours	mGNN†	32.51 ± 0.04
	mGNN	32.56 ± 0.32
Classic	GCN	30.98 ± 0.33
	GAT	30.53 ± 0.31
	SAGE	31.53 ± 0.19
	LGT	30.09 ± 0.20
Critical	GAT	32.43 ± 0.36
	LGT	31.44 ± 0.57
	SAGE	32.11 ± 0.49
	GCN	31.48 ± 0.16
GEN	Tuned	32.04 ± 0.53
	Learned	32.53 ± 0.26
Gym	Tuned	30.33 ± 0.05
	Golden	28.26 ± 0.61
PyG	SAGE	31.38 ± 0.15
	LGT	21.72 ± 2.46
	GAT	28.87 ± 0.19
	GCN	23.38 ± 0.21
DGL	SAGE	29.86 ± 0.28
	GAT	30.56 ± 0.21
	GCN	23.42 ± 0.37
OGB	SAGE	31.56 ± 0.20
	GCN	23.39 ± 0.11

Table 19: Full results on `city-roads-M`. *(Left)* Controlled backbone layout ablation and *(Middle)* controlled message passing ablation (Section 4). *(Right)* Baseline comparison (Section 7). Values are test score mean ± standard deviation over five evaluation seeds, multiplied by 100. Within every table, the best result is colored blue and the runner-up is underlined, both up to the standard deviation of the leader.

Tuned	60.44 ± 0.47
Classic	59.47 ± 0.27
Multilayer	59.49 ± 0.52
Transformer	60.07 ± 0.28

Tuned	60.44 ± 0.47
SAGE	59.33 ± 0.10
GCN	59.37 ± 0.18
GAT	59.68 ± 0.32
LGT	60.12 ± 0.27

Ours	mGNN†	61.97 ± 0.14
	mGNN	60.44 ± 0.47
Classic	GCN	60.14 ± 0.16
	GAT	60.22 ± 0.47
	SAGE	60.15 ± 0.58
	LGT	59.25 ± 0.26
Critical	GAT	59.68 ± 0.42
	LGT	59.55 ± 2.02
	SAGE	59.69 ± 0.54
	GCN	59.41 ± 0.40
GEN	Tuned	58.34 ± 0.50
	Learned	57.81 ± 1.31
Gym	Tuned	58.70 ± 0.49
	Golden	59.15 ± 0.35
PyG	SAGE	60.02 ± 0.19
	LGT	60.51 ± 0.55
	GAT	57.84 ± 0.71
	GCN	55.45 ± 0.11
DGL	SAGE	58.85 ± 0.68
	GAT	57.54 ± 0.19
	GCN	54.83 ± 0.09
OGB	SAGE	60.89 ± 0.15
	GCN	56.45 ± 0.17

Table 20: Full results on `city-roads-L`. *(Left)* Controlled backbone layout ablation and *(Middle)* controlled message passing ablation (Section 4). *(Right)* Baseline comparison (Section 7). Values are test score mean ± standard deviation over five evaluation seeds, multiplied by 100. Within every table, the best result is colored blue and the runner-up is underlined, both up to the standard deviation of the leader.

Tuned	56.23 ± 0.27
Classic	55.84 ± 0.33
Multilayer	53.14 ± 0.60
Transformer	54.61 ± 0.22

Tuned	56.23 ± 0.27
SAGE	54.52 ± 0.40
GCN	55.02 ± 0.12
GAT	55.62 ± 0.32
LGT	56.32 ± 0.28

Ours	mGNN†	55.65 ± 0.14
	mGNN	56.23 ± 0.27
Classic	GCN	55.34 ± 0.20
	GAT	55.19 ± 0.40
	SAGE	54.73 ± 0.25
	LGT	54.71 ± 0.27
Critical	GAT	53.75 ± 0.49
	LGT	54.12 ± 0.29
	SAGE	53.31 ± 0.41
	GCN	52.06 ± 0.78
GEN	Tuned	55.37 ± 0.72
	Learned	51.97 ± 3.09
Gym	Tuned	52.49 ± 0.29
	Golden	52.32 ± 0.35
PyG	SAGE	53.77 ± 0.41
	LGT	53.94 ± 0.67
	GAT	49.90 ± 0.09
	GCN	47.18 ± 0.08
DGL	SAGE	54.64 ± 0.16
	GAT	46.26 ± 0.39
	GCN	45.44 ± 0.30
OGB	SAGE	54.29 ± 0.37
	GCN	48.59 ± 0.12

Table 21: Full results on `twitch-views`. *(Left)* Controlled backbone layout ablation and *(Middle)* controlled message passing ablation (Section 4). *(Right)* Baseline comparison (Section 7). Values are test score mean ± standard deviation over five evaluation seeds, multiplied by 100. Within every table, the best result is colored blue and the runner-up is underlined, both up to the standard deviation of the leader.

Tuned	77.28 ± 0.13
Classic	77.61 ± 0.09
Multilayer	77.10 ± 0.24
Transformer	76.81 ± 0.05

Tuned	77.28 ± 0.13
SAGE	71.42 ± 0.25
GCN	78.15 ± 0.03
GAT	76.37 ± 0.19
LGT	76.30 ± 0.58

Ours	mGNN†	78.49 ± 0.07
	mGNN	77.28 ± 0.13
Classic	GCN	78.02 ± 0.13
	GAT	73.63 ± 1.22
	SAGE	73.52 ± 0.12
	LGT	54.77 ± 3.10
Critical	GAT	75.26 ± 0.30
	LGT	76.22 ± 0.08
	SAGE	71.37 ± 0.53
	GCN	77.60 ± 0.06
GEN	Tuned	74.22 ± 0.35
	Learned	74.81 ± 0.77
Gym	Tuned	72.19 ± 1.59
	Golden	71.07 ± 0.93
PyG	SAGE	72.74 ± 0.39
	LGT	54.05 ± 0.49
	GAT	67.23 ± 0.67
	GCN	67.06 ± 0.84
DGL	SAGE	71.89 ± 1.85
	GAT	71.89 ± 0.22
	GCN	72.70 ± 0.21
OGB	SAGE	72.74 ± 0.29
	GCN	68.85 ± 0.11

Table 22: Full results on `artnet-views`. *(Left)* Controlled backbone layout ablation and *(Middle)* controlled message passing ablation (Section 4). *(Right)* Baseline comparison (Section 7). Values are test score mean ± standard deviation over five evaluation seeds, multiplied by 100. Within every table, the best result is colored blue and the runner-up is underlined, both up to the standard deviation of the leader.

Tuned	59.51 ± 1.60
Classic	56.19 ± 0.52
Multilayer	$\underline{56.82 \pm 0.34}$
Transformer	56.26 ± 0.58

Tuned	59.51 ± 1.60
SAGE	$\underline{54.39 \pm 0.25}$
GCN	56.33 ± 3.70
GAT	57.94 ± 1.76
LGT	58.21 ± 0.33

Ours	mGNN†	62.26 ± 0.09
	mGNN	$\underline{59.51 \pm 1.60}$
Classic	GCN	$\underline{58.87 \pm 0.32}$
	GAT	56.40 ± 0.26
	SAGE	55.05 ± 0.08
	LGT	54.24 ± 1.24
Critical	GAT	55.00 ± 0.26
	LGT	56.58 ± 1.58
	SAGE	54.44 ± 0.20
	GCN	57.53 ± 0.31
GEN	Tuned	$\underline{58.17 \pm 0.34}$
	Learned	56.40 ± 0.21
Gym	Tuned	56.32 ± 0.41
	Golden	56.37 ± 0.73
PyG	SAGE	50.63 ± 0.25
	LGT	53.52 ± 4.78
	GAT	48.41 ± 1.59
	GCN	46.63 ± 1.08
DGL	SAGE	52.09 ± 0.16
	GAT	48.05 ± 1.16
	GCN	50.94 ± 0.21
OGB	SAGE	54.21 ± 0.16
	GCN	46.65 ± 0.56

Table 23: Full results on `cora`. *(Left)* Controlled backbone layout ablation and *(Middle)* controlled message passing ablation (Section 4). *(Right)* Baseline comparison (Section 7). Values are test score mean ± standard deviation over five evaluation seeds, multiplied by 100. Within every table, the best result is colored blue and the runner-up is underlined, both up to the standard deviation of the leader.

Tuned	82.07 ± 0.60
Classic	81.56 ± 0.74
Multilayer	80.65 ± 0.73
Transformer	82.26 ± 0.74

Tuned	82.07 ± 0.60
SAGE	81.31 ± 0.66
GCN	81.98 ± 0.85
GAT	82.32 ± 0.64
LGT	80.99 ± 0.94

Ours	mGNN†	81.89 ± 0.49
	mGNN	82.07 ± 0.60
Classic	GCN	82.32 ± 0.75
	GAT	81.84 ± 1.20
	SAGE	81.53 ± 0.54
	LGT	81.20 ± 0.27
Critical	GAT	81.06 ± 0.26
	LGT	81.51 ± 0.52
	SAGE	80.89 ± 0.72
	GCN	80.48 ± 0.74
GEN	Tuned	78.71 ± 1.22
	Learned	80.29 ± 0.58
Gym	Tuned	78.27 ± 1.63
	Golden	78.86 ± 1.15
PyG	SAGE	81.66 ± 0.56
	LGT	81.70 ± 0.26
	GAT	83.17 ± 0.26
	GCN	82.70 ± 0.62
DGL	SAGE	81.25 ± 0.69
	GAT	81.27 ± 0.51
	GCN	81.02 ± 0.32
OGB	SAGE	80.66 ± 0.77
	GCN	80.37 ± 0.91

Table 24: Full results on `citeseer`. *(Left)* Controlled backbone layout ablation and *(Middle)* controlled message passing ablation (Section 4). *(Right)* Baseline comparison (Section 7). Values are test score mean ± standard deviation over five evaluation seeds, multiplied by 100. Within every table, the best result is colored blue and the runner-up is underlined, both up to the standard deviation of the leader.

Tuned	71.33 ± 1.11
Classic	71.60 ± 0.85
Multilayer	70.99 ± 0.74
Transformer	71.10 ± 0.47

Tuned	71.33 ± 1.11
SAGE	71.36 ± 0.40
GCN	70.39 ± 0.51
GAT	71.29 ± 0.59
LGT	71.18 ± 0.69

Ours	mGNN†	70.75 ± 0.41
	mGNN	71.33 ± 1.11
Classic	GCN	71.45 ± 0.16
	GAT	70.36 ± 0.44
	SAGE	71.67 ± 0.72
	LGT	71.82 ± 0.39
Critical	GAT	68.40 ± 0.81
	LGT	70.16 ± 1.30
	SAGE	71.10 ± 0.68
	GCN	69.88 ± 0.54
GEN	Tuned	72.02 ± 0.96
	Learned	69.90 ± 1.28
Gym	Tuned	71.11 ± 0.47
	Golden	67.10 ± 0.92
PyG	SAGE	69.07 ± 0.23
	LGT	69.37 ± 0.96
	GAT	71.47 ± 0.26
	GCN	68.89 ± 1.22
DGL	SAGE	70.46 ± 0.84
	GAT	69.78 ± 0.35
	GCN	67.64 ± 0.48
OGB	SAGE	69.79 ± 0.57
	GCN	70.19 ± 0.84

Table 25: Full results on `pubmed`. *(Left)* Controlled backbone layout ablation and *(Middle)* controlled message passing ablation (Section 4). *(Right)* Baseline comparison (Section 7). Values are test score mean ± standard deviation over five evaluation seeds, multiplied by 100. Within every table, the best result is colored blue and the runner-up is underlined, both up to the standard deviation of the leader.

Tuned	86.65 ± 0.10
Classic	86.53 ± 0.12
Multilayer	86.31 ± 0.19
Transformer	86.22 ± 0.24

Tuned	86.65 ± 0.10
SAGE	86.03 ± 0.17
GCN	86.55 ± 0.05
GAT	86.45 ± 0.34
LGT	86.39 ± 0.20

Ours	mGNN†	86.85 ± 0.06
	mGNN	86.65 ± 0.10
Classic	GCN	85.61 ± 0.22
	GAT	85.35 ± 0.32
	SAGE	85.66 ± 0.21
	LGT	86.08 ± 0.09
Critical	GAT	86.20 ± 0.30
	LGT	86.32 ± 0.08
	SAGE	86.02 ± 0.20
	GCN	83.70 ± 0.09
GEN	Tuned	85.88 ± 0.25
	Learned	86.11 ± 0.27
Gym	Tuned	86.17 ± 0.20
	Golden	85.37 ± 0.28
PyG	SAGE	85.12 ± 0.25
	LGT	85.42 ± 0.14
	GAT	84.57 ± 0.22
	GCN	85.67 ± 0.20
DGL	SAGE	85.86 ± 0.41
	GAT	83.13 ± 0.09
	GCN	83.39 ± 0.06
OGB	SAGE	84.89 ± 0.16
	GCN	84.54 ± 0.27

Table 26: Full results on `coauthor-cs`. *(Left)* Controlled backbone layout ablation and *(Middle)* controlled message passing ablation (Section 4). *(Right)* Baseline comparison (Section 7). Values are test score mean ± standard deviation over five evaluation seeds, multiplied by 100. Within every table, the best result is colored blue and the runner-up is underlined, both up to the standard deviation of the leader.

Tuned	$\underline{94.03 \pm 0.08}$
Classic	94.35 ± 0.03
Multilayer	94.33 ± 0.09
Transformer	94.43 ± 0.14

Tuned	94.03 ± 0.08
SAGE	$\underline{94.22 \pm 0.08}$
GCN	94.50 ± 0.04
GAT	94.21 ± 0.10
LGT	$\underline{94.21 \pm 0.11}$

Ours	mGNN†	94.05 ± 0.06
	mGNN	94.03 ± 0.08
Classic	GCN	$\underline{94.25 \pm 0.05}$
	GAT	93.69 ± 0.14
	SAGE	93.61 ± 0.14
	LGT	93.80 ± 0.08
Critical	GAT	94.14 ± 0.10
	LGT	93.82 ± 0.13
	SAGE	93.91 ± 0.15
	GCN	93.87 ± 0.10
GEN	Tuned	93.66 ± 0.09
	Learned	94.36 ± 0.08
Gym	Tuned	93.73 ± 0.16
	Golden	93.52 ± 0.19
PyG	SAGE	92.69 ± 0.15
	LGT	93.79 ± 0.10
	GAT	92.12 ± 0.63
	GCN	92.50 ± 0.04
DGL	SAGE	93.50 ± 0.12
	GAT	90.91 ± 0.21
	GCN	90.94 ± 0.08
OGB	SAGE	92.77 ± 0.07
	GCN	92.49 ± 0.08

Table 27: Full results on `coauthor-physics`. *(Left)* Controlled backbone layout ablation and *(Middle)* controlled message passing ablation (Section 4). *(Right)* Baseline comparison (Section 7). Values are test score mean ± standard deviation over five evaluation seeds, multiplied by 100. Within every table, the best result is colored blue and the runner-up is underlined, both up to the standard deviation of the leader.

Tuned	96.45 ± 0.09
Classic	96.37 ± 0.04
Multilayer	96.45 ± 0.08
Transformer	96.41 ± 0.03

Tuned	$\underline{96.45 \pm 0.09}$
SAGE	96.15 ± 0.06
GCN	96.58 ± 0.04
GAT	96.15 ± 0.06
LGT	96.17 ± 0.04

Ours	mGNN†	96.31 ± 0.08
	mGNN	$\underline{96.45 \pm 0.09}$
Classic	GCN	96.53 ± 0.04
	GAT	96.00 ± 0.05
	SAGE	96.18 ± 0.03
	LGT	96.11 ± 0.13
Critical	GAT	96.17 ± 0.07
	LGT	96.16 ± 0.06
	SAGE	96.23 ± 0.05
	GCN	95.72 ± 0.06
GEN	Tuned	95.98 ± 0.10
	Learned	96.11 ± 0.06
Gym	Tuned	95.67 ± 0.22
	Golden	95.71 ± 0.14
PyG	SAGE	95.65 ± 0.03
	LGT	95.67 ± 0.12
	GAT	95.28 ± 0.29
	GCN	95.59 ± 0.08
DGL	SAGE	96.03 ± 0.08
	GAT	95.08 ± 0.06
	GCN	95.50 ± 0.03
OGB	SAGE	95.56 ± 0.07
	GCN	95.59 ± 0.06

Table 28: Full results on `amazon-computers`. *(Left)* Controlled backbone layout ablation and *(Middle)* controlled message passing ablation (Section 4). *(Right)* Baseline comparison (Section 7). Values are test score mean ± standard deviation over five evaluation seeds, multiplied by 100. Within every table, the best result is colored blue and the runner-up is underlined, both up to the standard deviation of the leader.

Tuned	90.31 ± 0.31
Classic	90.01 ± 0.11
Multilayer	90.05 ± 0.14
Transformer	90.10 ± 0.16

Tuned	90.31 ± 0.31
SAGE	90.16 ± 0.27
GCN	$\underline{89.85 \pm 0.31}$
GAT	90.11 ± 0.17
LGT	90.27 ± 0.27

Ours	mGNN†	90.18 ± 0.22
	mGNN	90.31 ± 0.31
Classic	GCN	90.77 ± 0.16
	GAT	$\underline{90.45 \pm 0.24}$
	SAGE	89.91 ± 0.30
	LGT	89.89 ± 0.09
Critical	GAT	89.87 ± 0.66
	LGT	90.07 ± 0.15
	SAGE	88.11 ± 0.55
	GCN	89.86 ± 0.25
GEN	Tuned	89.29 ± 0.14
	Learned	89.55 ± 0.33
Gym	Tuned	89.75 ± 0.37
	Golden	88.17 ± 0.21
PyG	SAGE	$\underline{90.57 \pm 0.18}$
	LGT	89.84 ± 0.50
	GAT	$\underline{90.58 \pm 0.27}$
	GCN	90.05 ± 0.25
DGL	SAGE	89.72 ± 0.31
	GAT	$\underline{90.32 \pm 0.19}$
	GCN	89.91 ± 0.33
OGB	SAGE	89.81 ± 0.35
	GCN	89.81 ± 0.08

Table 29: Full results on `amazon-photo`. *(Left)* Controlled backbone layout ablation and *(Middle)* controlled message passing ablation (Section 4). *(Right)* Baseline comparison (Section 7). Values are test score mean ± standard deviation over five evaluation seeds, multiplied by 100. Within every table, the best result is colored blue and the runner-up is underlined, both up to the standard deviation of the leader.

Tuned	$\underline{94.01 \pm 0.23}$
Classic	94.46 ± 0.50
Multilayer	94.66 ± 0.46
Transformer	94.35 ± 0.17

Tuned	$\underline{94.01 \pm 0.23}$
SAGE	$\underline{94.05 \pm 0.24}$
GCN	$\underline{94.11 \pm 0.11}$
GAT	94.31 ± 0.19
LGT	94.24 ± 0.29

Ours	mGNN†	94.43 ± 0.52
	mGNN	94.01 ± 0.23
Classic	GCN	94.17 ± 0.12
	GAT	93.58 ± 0.26
	SAGE	94.14 ± 0.18
	LGT	94.42 ± 0.11
Critical	GAT	94.20 ± 0.22
	LGT	$\underline{93.80 \pm 0.38}$
	SAGE	94.08 ± 0.23
	GCN	94.02 ± 0.50
GEN	Tuned	$\underline{93.87 \pm 0.18}$
	Learned	94.19 ± 0.25
Gym	Tuned	$\underline{93.81 \pm 0.29}$
	Golden	93.22 ± 0.40
PyG	SAGE	94.11 ± 0.12
	LGT	94.23 ± 0.19
	GAT	93.55 ± 0.14
	GCN	92.91 ± 0.22
DGL	SAGE	94.40 ± 0.23
	GAT	92.13 ± 0.40
	GCN	92.33 ± 0.14
OGB	SAGE	93.46 ± 0.18
	GCN	92.78 ± 0.32

Table 30: Full results on `lastfm-asia`. *(Left)* Controlled backbone layout ablation and *(Middle)* controlled message passing ablation (Section 4). *(Right)* Baseline comparison (Section 7). Values are test score mean ± standard deviation over five evaluation seeds, multiplied by 100. Within every table, the best result is colored blue and the runner-up is underlined, both up to the standard deviation of the leader.

Tuned	84.10 ± 0.26
Classic	83.65 ± 0.21
Multilayer	84.47 ± 0.37
Transformer	83.15 ± 0.30

Tuned	84.10 ± 0.26
SAGE	85.24 ± 0.14
GCN	82.79 ± 0.27
GAT	85.14 ± 0.19
LGT	85.33 ± 0.09

Ours	mGNN†	85.10 ± 0.43
	mGNN	84.10 ± 0.26
Classic	GCN	85.13 ± 0.14
	GAT	85.61 ± 0.20
	SAGE	84.78 ± 0.14
	LGT	85.02 ± 0.09
Critical	GAT	82.65 ± 0.64
	LGT	83.20 ± 0.23
	SAGE	82.24 ± 0.26
	GCN	83.50 ± 0.13
GEN	Tuned	83.30 ± 0.21
	Learned	81.61 ± 0.52
Gym	Tuned	83.16 ± 0.21
	Golden	82.74 ± 0.55
PyG	SAGE	84.12 ± 0.21
	LGT	83.95 ± 0.30
	GAT	85.18 ± 0.12
	GCN	85.18 ± 0.26
DGL	SAGE	84.90 ± 0.49
	GAT	84.40 ± 0.58
	GCN	82.78 ± 0.64
OGB	SAGE	84.98 ± 0.19
	GCN	84.79 ± 0.18

Table 31: Full results on `facebook`. *(Left)* Controlled backbone layout ablation and *(Middle)* controlled message passing ablation (Section 4). *(Right)* Baseline comparison (Section 7). Values are test score mean ± standard deviation over five evaluation seeds, multiplied by 100. Within every table, the best result is colored blue and the runner-up is underlined, both up to the standard deviation of the leader.

Tuned	93.35 ± 0.12
Classic	93.12 ± 0.21
Multilayer	93.41 ± 0.21
Transformer	92.92 ± 0.16

Tuned	93.35 ± 0.12
SAGE	93.01 ± 0.08
GCN	92.57 ± 0.18
GAT	93.16 ± 0.23
LGT	93.06 ± 0.18

Ours	mGNN†	92.86 ± 0.23
	mGNN	93.35 ± 0.12
Classic	GCN	93.08 ± 0.07
	GAT	93.15 ± 0.22
	SAGE	92.72 ± 0.24
	LGT	93.17 ± 0.19
Critical	GAT	93.43 ± 0.15
	LGT	93.10 ± 0.14
	SAGE	92.50 ± 0.27
	GCN	92.94 ± 0.31
GEN	Tuned	93.02 ± 0.21
	Learned	92.23 ± 0.21
Gym	Tuned	92.90 ± 0.10
	Golden	91.66 ± 0.26
PyG	SAGE	92.60 ± 0.12
	LGT	92.58 ± 0.20
	GAT	92.86 ± 0.15
	GCN	92.60 ± 0.22
DGL	SAGE	92.89 ± 0.20
	GAT	92.24 ± 0.10
	GCN	92.15 ± 0.17
OGB	SAGE	92.67 ± 0.19
	GCN	92.55 ± 0.10

Table 32: Full results on `flickr`. *(Left)* Controlled backbone layout ablation and *(Middle)* controlled message passing ablation (Section 4). *(Right)* Baseline comparison (Section 7). Values are test score mean ± standard deviation over five evaluation seeds, multiplied by 100. Within every table, the best result is colored blue and the runner-up is underlined, both up to the standard deviation of the leader.

Tuned	53.00 ± 0.24
Classic	52.02 ± 0.20
Multilayer	51.68 ± 0.25
Transformer	52.91 ± 0.24

Tuned	53.00 ± 0.24
SAGE	52.86 ± 0.25
GCN	51.34 ± 0.06
GAT	53.24 ± 0.23
LGT	53.29 ± 0.22

Ours	mGNN†	53.19 ± 0.23
	mGNN	53.00 ± 0.24
Classic	GCN	53.40 ± 0.16
	GAT	53.20 ± 0.30
	SAGE	52.59 ± 0.16
	LGT	51.93 ± 0.52
Critical	GAT	51.51 ± 0.32
	LGT	52.06 ± 0.66
	SAGE	50.54 ± 1.20
	GCN	51.02 ± 1.75
GEN	Tuned	51.28 ± 0.57
	Learned	51.39 ± 0.29
Gym	Tuned	52.27 ± 0.37
	Golden	50.75 ± 0.25
PyG	SAGE	52.46 ± 0.33
	LGT	51.61 ± 0.35
	GAT	52.63 ± 0.10
	GCN	53.53 ± 0.31
DGL	SAGE	52.69 ± 0.12
	GAT	52.94 ± 0.32
	GCN	53.88 ± 0.21
OGB	SAGE	52.51 ± 0.31
	GCN	53.05 ± 0.14

Table 33: Full results on `roman-empire`. *(Left)* Controlled backbone layout ablation and *(Middle)* controlled message passing ablation (Section 4). *(Right)* Baseline comparison (Section 7). Values are test score mean ± standard deviation over five evaluation seeds, multiplied by 100. Within every table, the best result is colored blue and the runner-up is underlined, both up to the standard deviation of the leader.

Tuned	82.33 ± 0.28
Classic	82.03 ± 0.26
Multilayer	81.28 ± 0.16
Transformer	81.54 ± 0.21

Tuned	82.33 ± 0.28
SAGE	81.50 ± 0.17
GCN	79.83 ± 0.20
GAT	81.47 ± 0.41
LGT	82.24 ± 0.30

Ours	mGNN†	81.74 ± 0.41
	mGNN	82.33 ± 0.28
Classic	GCN	80.08 ± 0.18
	GAT	80.24 ± 0.42
	SAGE	79.51 ± 0.35
	LGT	79.62 ± 0.54
Critical	GAT	80.37 ± 0.21
	LGT	80.08 ± 0.77
	SAGE	79.83 ± 0.39
	GCN	70.63 ± 0.22
GEN	Tuned	81.45 ± 0.30
	Learned	80.42 ± 0.35
Gym	Tuned	80.81 ± 0.43
	Golden	73.42 ± 0.31
PyG	SAGE	70.42 ± 0.41
	LGT	72.45 ± 0.34
	GAT	44.54 ± 1.68
	GCN	38.24 ± 0.70
DGL	SAGE	79.58 ± 0.36
	GAT	55.68 ± 0.98
	GCN	50.49 ± 0.31
OGB	SAGE	74.26 ± 0.18
	GCN	38.81 ± 0.30

Table 34: Full results on `amazon-ratings`. *(Left)* Controlled backbone layout ablation and *(Middle)* controlled message passing ablation (Section 4). *(Right)* Baseline comparison (Section 7). Values are test score mean ± standard deviation over five evaluation seeds, multiplied by 100. Within every table, the best result is colored blue and the runner-up is underlined, both up to the standard deviation of the leader.

Tuned	40.97 ± 0.54
Classic	41.36 ± 0.66
Multilayer	41.88 ± 0.67
Transformer	40.33 ± 0.57

Tuned	40.97 ± 0.54
SAGE	42.03 ± 0.26
GCN	42.10 ± 0.59
GAT	40.78 ± 0.46
LGT	40.88 ± 0.56

Ours	mGNN†	41.93 ± 0.24
	mGNN	40.97 ± 0.54
Classic	GCN	42.08 ± 0.87
	GAT	41.41 ± 0.48
	SAGE	42.56 ± 0.26
	LGT	41.87 ± 0.35
Critical	GAT	41.13 ± 0.32
	LGT	40.04 ± 0.39
	SAGE	42.16 ± 0.48
	GCN	42.73 ± 0.28
GEN	Tuned	41.29 ± 0.49
	Learned	42.15 ± 0.23
Gym	Tuned	41.63 ± 0.56
	Golden	40.39 ± 1.01
PyG	SAGE	41.95 ± 0.29
	LGT	41.46 ± 0.26
	GAT	42.22 ± 0.68
	GCN	41.80 ± 0.62
DGL	SAGE	42.56 ± 0.41
	GAT	42.37 ± 0.51
	GCN	41.07 ± 0.48
OGB	SAGE	41.81 ± 0.53
	GCN	42.88 ± 0.36

Table 35: Full results on `amazon-ratings-full`. *(Left)* Controlled backbone layout ablation and *(Middle)* controlled message passing ablation (Section 4). *(Right)* Baseline comparison (Section 7). Values are test score mean ± standard deviation over five evaluation seeds, multiplied by 100. Within every table, the best result is colored blue and the runner-up is underlined, both up to the standard deviation of the leader.

Tuned	$\underline{39.59 \pm 0.05}$
Classic	39.29 ± 0.32
Multilayer	39.74 ± 0.10
Transformer	39.52 ± 0.11

Tuned	$\underline{39.59 \pm 0.05}$
SAGE	39.44 ± 0.06
GCN	39.76 ± 0.16
GAT	39.07 ± 0.31
LGT	39.61 ± 0.11

Ours	mGNN†	40.28 ± 0.19
	mGNN	39.59 ± 0.05
Classic	GCN	$\underline{40.09 \pm 0.05}$
	GAT	39.89 ± 0.11
	SAGE	39.53 ± 0.13
	LGT	39.38 ± 0.05
Critical	GAT	39.47 ± 0.18
	LGT	39.00 ± 0.14
	SAGE	39.23 ± 0.06
	GCN	39.65 ± 0.06
GEN	Tuned	39.28 ± 0.12
	Learned	38.83 ± 0.21
Gym	Tuned	39.54 ± 0.09
	Golden	37.83 ± 0.21
PyG	SAGE	39.30 ± 0.17
	LGT	39.21 ± 0.38
	GAT	39.66 ± 0.12
	GCN	39.78 ± 0.10
DGL	SAGE	39.67 ± 0.08
	GAT	$\underline{40.07 \pm 0.08}$
	GCN	40.18 ± 0.05
OGB	SAGE	39.39 ± 0.15
	GCN	39.54 ± 0.34

Table 36: Full results on `questions`. *(Left)* Controlled backbone layout ablation and *(Middle)* controlled message passing ablation (Section 4). *(Right)* Baseline comparison (Section 7). Values are test score mean ± standard deviation over five evaluation seeds, multiplied by 100. Within every table, the best result is colored blue and the runner-up is underlined, both up to the standard deviation of the leader.

Tuned	$\underline{15.57 \pm 0.78}$
Classic	17.58 ± 0.58
Multilayer	17.34 ± 0.74
Transformer	17.64 ± 0.46

Tuned	$\underline{15.57 \pm 0.78}$
SAGE	16.45 ± 0.34
GCN	17.28 ± 0.92
GAT	12.95 ± 0.84
LGT	$\underline{15.81 \pm 0.68}$

Ours	mGNN†	19.55 ± 0.33
	mGNN	15.57 ± 0.78
Classic	GCN	15.20 ± 0.54
	GAT	14.61 ± 0.54
	SAGE	17.70 ± 0.52
	LGT	13.84 ± 0.79
Critical	GAT	16.30 ± 0.24
	LGT	17.30 ± 0.34
	SAGE	14.77 ± 1.25
	GCN	11.48 ± 0.26
GEN	Tuned	$\underline{19.14 \pm 0.38}$
	Learned	17.67 ± 0.44
Gym	Tuned	17.93 ± 1.15
	Golden	15.34 ± 1.53
PyG	SAGE	15.52 ± 0.89
	LGT	14.83 ± 1.40
	GAT	13.54 ± 1.03
	GCN	14.40 ± 0.46
DGL	SAGE	17.16 ± 0.13
	GAT	11.31 ± 0.78
	GCN	14.45 ± 0.39
OGB	SAGE	12.49 ± 0.75
	GCN	13.62 ± 0.80

Table 37: Full results on `wiki-cs`. *(Left)* Controlled backbone layout ablation and *(Middle)* controlled message passing ablation (Section 4). *(Right)* Baseline comparison (Section 7). Values are test score mean ± standard deviation over five evaluation seeds, multiplied by 100. Within every table, the best result is colored blue and the runner-up is underlined, both up to the standard deviation of the leader.

Tuned	81.88 ± 0.12
Classic	82.35 ± 0.30
Multilayer	81.91 ± 0.35
Transformer	81.46 ± 0.44

Tuned	81.88 ± 0.12
SAGE	81.62 ± 0.41
GCN	80.67 ± 0.42
GAT	82.51 ± 0.15
LGT	81.81 ± 0.29

Ours	mGNN†	81.91 ± 0.34
	mGNN	81.88 ± 0.12
Classic	GCN	82.12 ± 0.25
	GAT	82.65 ± 0.26
	SAGE	82.39 ± 0.15
	LGT	82.29 ± 0.23
Critical	GAT	81.79 ± 0.25
	LGT	81.65 ± 0.20
	SAGE	81.92 ± 0.16
	GCN	80.45 ± 1.60
GEN	Tuned	82.44 ± 0.31
	Learned	81.34 ± 0.23
Gym	Tuned	82.44 ± 0.23
	Golden	81.04 ± 0.10
PyG	SAGE	82.75 ± 0.09
	LGT	82.44 ± 0.18
	GAT	81.70 ± 0.36
	GCN	81.75 ± 0.14
DGL	SAGE	82.53 ± 0.18
	GAT	80.08 ± 0.25
	GCN	79.50 ± 0.12
OGB	SAGE	82.65 ± 0.20
	GCN	81.76 ± 0.21