Benchmarking and Transfer Learning for Hyperparameter Optimization of Graph Neural Networks

Marek Dědič[1,2][a] and Michal Bělohlávek[1,2]
[1] *Czech Technical University in Prague, Trojanova 13, Prague, Czechia*
[2] *Cisco Systems, Prague, Czechia*
marek@dedic.eu, mbelohla@cisco.com

Keywords:
Graph Neural Networks, Hyperparameter Optimization, Meta-Learning

Abstract:
Graph Neural Networks (GNNs) rely critically on effective hyperparameter optimization (HPO), but comprehensive HPO benchmarks for graph learning tasks remain limited. This paper addresses this gap by presenting an extensive comparison of HPO strategies for GNNs, ranging from simple random search to sophisticated Sequential Model-Based Optimization (SMBO) techniques like Bayesian Optimization (BO) and Tree-structured Parzen Estimators (TPE). Furthermore, we explore meta-learning for transfer learning, investigating its potential to accelerate HPO on new tasks by leveraging knowledge from past runs. Our results provide a practical comparative guide and demonstrate the viability of using meta-learned knowledge to significantly accelerate GNN HPO.

1 INTRODUCTION

Machine learning (ML) has become a cornerstone of modern technology. Many ML applications naturally involve structured data—such as network flows, system call trees, or privilege graphs—that are most faithfully modeled as graphs. A critical, though often overlooked, aspect of deploying these models is hyperparameter optimization (HPO). The specific choices made for hyperparameters – such as learning rate, regularization strength, hidden dimensions, and sampling strategies – can cause significant performance swings.

1.1 Related Work

Classical HPO approaches range from simple, exhaustive grid search to more efficient, non-exhaustive sampling methods like random search, which was shown to be surprisingly effective, and Quasi-Monte Carlo (QMC) sampling [Bergstra and Bengio, 2012]. More sophisticated strategies leverage Sequential Model-based Optimization (SMBO). Prominent SMBO approaches include Bayesian Optimization (BO) and Tree-structured Parzen Estimators (TPE) [Bergstra et al., 2011]. These methods build a surrogate model of the objective function and use an acquisition function to intelligently select the next set of hyperparameters to evaluate, balancing exploration and exploitation.

The intersection of HPO and graph learning is a nascent but growing field, largely motivated by the high cost of GNN training and the complex interplay of their hyperparameters. Much of the effort has been concentrated under the umbrella of Neural Architecture Search (NAS) for GNNs. For example, GraphNAS [Gao et al., 2020] and AutoGNN [Zhou et al., 2022] employed reinforcement learning to search for optimal GNN architectures. Both of these works, however, primarily focus on discovering new architectures rather than tuning hyperparameters of existing ones and evaluate on a limited number of datasets.

Works specifically addressing HPO for GNNs are most often limited to specific domains, such as molecular property prediction [Yuan et al., 2021a, Yuan et al., 2021b, Yuan et al., 2021c, Ebadi et al., 2025] or mRNA

[a] https://orcid.org/0000-0003-1021-8428

degradation prediction [Vodilovska et al., 2023]. These studies often evaluate a narrow selection of datasets and do not offer a comprehensive comparison to other HPO methods, making it difficult to generalize their findings across different types of graph data and GNN architectures.

1.2 Contributions

Despite extensive benchmarking of HPO algorithms on traditional tabular, image and text tasks, existing works evaluating HPO methods for graph learning are limited in scope (see Section 1.1). They typically focus on a small number of graph datasets, usually in a single domain, a limited set of hyperparameters, or a limited set of HPO methods. This work addresses that gap by presenting the first systematic HPO benchmark on Graph Neural Networks (GNNs).

Additionally, we propose a novel approach to HPO on graph datasets specifically. This approach leverages the fact that (unlike most other dataset types) graph datasets have an extensive internal structure that can be characterized by various statistical properties (e.g., number of nodes, edges, degree distribution, homophily, etc.). These properties can provide valuable insights into the nature of the data and its underlying relationships. Our work aims to exploit these unique characteristics of graph datasets to enhance the HPO process. Specifically, we argue that these graph-specific properties can be used to inform and guide the HPO process more effectively than generic methods that do not consider the structure of the data.

The key contributions of this research are:

1. **Systematic HPO Benchmark on GNNs**: The study evaluates and analyzes the convergence speed and final score, of five popular HPO methods (grid search, random search, BO, TPE, and Sobol QMC) for tuning GraphSAGE across nine diverse graph datasets. This establishes a reproducible baseline for future work on GNN hyperparameter tuning.
2. **Meta-Learning for Hyperparameter Transfer**: The work evaluates the *Cross-RF* (i.e. cross-dataset random-forest) surrogate model that learns how dataset properties (meta-features) interact with hyperparameter performance. This model leverages past HPO runs on related graphs to warm-start the search on new datasets, demonstrating its ability to accelerate tuning and reduce the number of costly model evaluations required to reach peak performance.

In Section 1.1, we review related work on HPO methods and existing benchmarks, particularly in the context of graph learning. Section 2 provides necessary context of HPO more generally. Section 3 introduces the Cross-RF method for HPO for graph learning. Section 4 outlines our benchmarking methodology, including dataset selection, hyperparameter spaces, and evaluated methods and reports on the performance of SoTA HPO methods in the graph setting. In Section 5, we evaluate the Cross-RF method, comparing its performance to established HPO methods.Finally, Section 6 summarizes our findings and discusses future research directions.

2 PROBLEM DEFINITION

Most machine learning algorithms, including graph neural networks (GNNs), have hyperparameters that need to be set before training the model. The process of finding the optimal hyperparameter configuration is known as hyperparameter optimization (HPO). While previously manually tuning hyperparameters was common, it is now widely recognized that systematic HPO methods lead to better model performance and generalization. HPO is essentially a black-box optimization problem, moreover, it is a very expensive one, as each evaluation of a hyperparameter configuration requires training and validating a machine learning model.

In this work, we mainly consider HPO in the context of supervised learning tasks. We follow the notation and formalism of [Bischl et al., 2023]. Let $\mathcal{D}$ be a labelled dataset of pairs $(\boldsymbol{x}_i, y_i)$, where $\boldsymbol{x}_i \in \mathcal{X}$ are the input features and $y_i \in \mathcal{Y}$ are the corresponding labels. We assume that $\mathcal{D}$ is drawn as repeated i.i.d. samples from an unknown joint probability distribution $\mathrm{P}_{\mathcal{X}\mathcal{Y}}$. We may sometimes want to operate with the whole dataset at once, in which case we denote by $\boldsymbol{X}_{\mathcal{D}}$ the matrix of all input features and by $\boldsymbol{y}_{\mathcal{D}}$ the vector of all labels.

The goal of supervised learning is to learn a function $f : \mathcal{X} \to \mathbb{R}^g$ that maps input features to predictions (we consider g to be the number of classes for classification and $g = 1$ for regression) and generalizes well to unseen data. The function space to which a model f belongs is usually called the *hypothesis space* and is denoted $\mathscr{H}$. The process of obtaining such a model is configured by

some hyperparameters forming a hyperparameter configuration $\lambda \in \mathbf{\Lambda}$. Formally, the learning process can be described as an application of an *inducer* function $\mathscr{F} : \mathrm{D} \times \mathbf{\Lambda} \to \mathscr{H}$, which takes as input a dataset $\mathcal{D} \in \mathrm{D}$ and a hyperparameter configuration $\lambda \in \mathbf{\Lambda}$, and outputs a trained model $\hat{f} = \mathscr{F}(\mathcal{D}, \lambda)$ (we may sometimes denote this model $\hat{f}_\lambda$ to emphasize the role of the hyperparameter configuration in its training), where $\mathrm{D} = \bigcup_{n \in \mathbb{N}} (\mathcal{X} \times \mathcal{Y})^n$ is the set of all datasets.

After training a model $\hat{f}$, we need to evaluate its performance given new, unseen data. We define a general performance measure $\rho : \bigcup_{m \in \mathbb{N}} (\mathcal{Y}^m \times \mathbb{R}^{g \times m}) \to \mathbb{R}$ that takes as input the true labels and the model predictions for m samples and outputs a real-valued score (for example, accuracy for classification or mean squared error for regression).

The inducer $\mathscr{F}$ is usually configured by a set of hyperparameters $\lambda \in \mathbf{\Lambda}$. The goal of HPO is to find the optimal hyperparameter configuration λ^* that maximizes the performance measure ρ on unseen data. Usually, a HPO algorithm considers only a finite set of hyperparameter configurations $\tilde{\Lambda} \subset \mathbf{\Lambda}$ called a *search space* due to computational constraints. Most HPO algorithms work by iteratively selecting hyperparameter configurations from the search space, training models using these configurations, and evaluating their performance using the performance measure ρ. Based on the evaluation results, the HPO algorithm updates its strategy for selecting the next hyperparameter configurations to evaluate. This process allows such an HPO algorithm to be formalized as a tuner $\tau : \left(\mathcal{D}, \mathscr{F}, \tilde{\Lambda}, \rho\right) \to \hat{\lambda}$, which takes as input a dataset $\mathcal{D}$, an inducer $\mathscr{F}$, a search space $\tilde{\Lambda}$, and a performance measure ρ, and proposes an estimate $\hat{\lambda} \in \tilde{\Lambda}$ of the optimal hyperparameter configuration λ^*.

3 AN RF-BASED METHOD FOR GRAPH HPO

In this section, we introduce the metalearning Cross-RF method, a novel approach for HPO specifically designed for graph learning, building on our previous metamodel work [Procházka et al., 2023] and repurposing it for hyperparameter optimization.

Suppose we have a *graph property descriptor* function $\delta : \mathrm{D} \to \mathbb{R}^d$ that maps a graph dataset $\mathcal{D}$ to a vector of numerical properties $\boldsymbol{d}$ (e.g., number of nodes, number of edges, average degree, etc.). This function allows us to capture the characteristics of different graph datasets in a structured manner. Suppose we also have a *metamodel* $\mathcal{M}_\rho : \mathbb{R}^d \times \mathbf{\Lambda} \to \mathbb{R}$ that takes as input the properties of a graph dataset $\delta(\mathcal{D})$ (i.e. the metafeatures) and a hyperparameter configuration λ, and outputs an estimate of a performance metric ρ (e.g., accuracy, F1-score) of a model $\hat{f}_\lambda$ trained on $\mathcal{D}$ with the hyperparameter configuration λ. Using this, we can define the Cross-RF hyperparameter tuning method as follows:

$$\tau\left(\mathcal{D}, \mathscr{F}, \tilde{\Lambda}, \rho\right) = \arg\max_{\lambda \in \tilde{\Lambda}} \mathcal{M}_\rho\left(\delta\left(\mathcal{D}\right), \lambda\right)$$

The feasibility of the Cross-RF method relies on the construction of the metamodel $\mathcal{M}_\rho$ and on its computational complexity – namely, the metamodel must be fast enough so that evaluating it on every hyperparameter configuration from $\tilde{\Lambda}$ at every step of the optimization is not prohibitively costly.

To build the metamodel $\mathcal{M}_\rho$, we will use an off-the-shelf regression model (in our case, a random forest (RF) regressor, but other regression models could be used as well) trained on a meta-dataset $\mathcal{H}$. The meta-dataset $\mathcal{H}$ is initialized as $\mathcal{H} = \emptyset$ and constructed by collecting dataset properties, hyperparameter configurations, and corresponding performance metrics across the progress of the training:

$$\mathcal{H} = \left\{ \left(\delta\left(\mathcal{D}^{(1)}\right), \lambda^{(1)}, \rho(\boldsymbol{y}_{\mathcal{D}^{(1)}}, \hat{f}_{\lambda^{(1)}}(\boldsymbol{X}_{\mathcal{D}^{(1)}}))\right), \right.$$
$$\left. \left(\delta\left(\mathcal{D}^{(2)}\right), \lambda^{(2)}, \rho(\boldsymbol{y}_{\mathcal{D}^{(2)}}, \hat{f}_{\lambda^{(2)}}(\boldsymbol{X}_{\mathcal{D}^{(2)}}))\right), \dots \right\}$$

An observant reader may have already noticed that when we use this method to optimize hyperparameters for one learning task (and thus only one dataset), the dataset properties $\delta\left(\mathcal{D}^{(i)}\right)$ will all be identical and thus of no benefit to the metamodel. To fix this issue, we must employ a cross-dataset pre-training strategy, where we collect data from multiple different graph datasets to populate the starting meta-dataset $\mathcal{H}$. This way, the metamodel $\mathcal{M}_\rho$ can learn to generalize across different graph datasets based on their properties. The datasets used in the pre-training phase should ideally be diverse and representative of the types of graphs we expect to encounter in practice.

3.1 Dataset Property Descriptor

We characterize each dataset using a descriptor vector $\delta(\mathcal{D})$ consisting of properties categorized by their relation to the graph **structure** (topology), node **attributes**, and the specific **task** (labels):

- **General Structural Properties:** These capture the topology independent of labels, including basic statistics (node count, edge count, number of components, average node degree) and global assortativity.
- **Structure & Attributes:** We measure the interplay between topology and features via attribute similarity (average cosine similarity across edges).
- **Structure & Task (Label Dependencies):** The majority of properties quantify the relationship between the graph topology and node labels. This includes comprehensive homophily measures (edge, node, class, attribute, and adjusted homophily), node label informativeness, and label-dependent connectivity metrics (balanced and adjusted accuracy). We also calculate structural metrics specific to the positive class, such as the average positive node degree, the ratio of positive nodes with degree > 1 or > 2, and the relative presence of positive edges.
- **Attributes & Task:** To assess how well features separate classes, we compute the average positive attribute similarity and the cross-class similarities (positive-to-negative and negative-to-positive).

By incorporating these diverse properties into the descriptor function δ, we aim to capture a comprehensive representation of each graph dataset, which is essential for the effective training of the metamodel $\mathcal{M}_\rho$. For properties that depend on a positive class (e.g., "Average positive node degree"), we take the average over all possible reductions of the dataset to a binary classification task in a one-vs-others manner.

4 AN HPO INVESTIGATION ON GRAPH NEURAL NETWORKS

In this section, we present a benchmark study to evaluate the performance of different hyperparameter optimization (HPO) methods on Graph Neural Networks (GNNs). We focus on assessing the efficiency and effectiveness of various HPO techniques in optimizing GNN architectures for node classification tasks, which are the most common tasks and are the cornerstones of other techniques for e.g. graph classification as well.

4.1 Experimental Setup

We conduct our experiments on 9 standard graph datasets. The 3 "small" datasets were the datasets Cora and CiteSeer [Yang et al., 2016], which were used with the "full" train-test split, and the Squirrel dataset [Rozemberczki et al., 2021]. Five medium sized datasets were also used, the PubMed dataset [Yang et al., 2016], the DBLP dataset and the full version of the Cora dataset [Bojchevski and Günnemann, 2018], the Computers dataset [Shchur et al., 2019] and the Flickr dataset [Zeng et al., 2019]. Finally, one large dataset was used, the OGB ArXiv dataset [Hu et al., 2021].

We evaluate the following HPO methods:

- Grid Search,
- Random Search,
- Bayesian Optimization,
- Sobol Quasi-Monte Carlo,
- Tree-structured Parzen Estimator (TPE).

The methods were all implemented using the Optuna hyperparameter optimization framework [Akiba et al., 2019]. All algorithms run with the default settings, with 2 exceptions: (1) for Sobol, we use the scrambling method of [Matoušek, 1998] and (2) for TPE, we use the multivariate variant [Falkner et al., 2018].

4.2 Evaluation Routine

Each HPO method and dataset combination was evaluated independently 10 times to account for variability in performance due to random initialization and stochastic training processes. For each study, we used 2 stopping criteria to limit the number of trials:

- **Patience**: If no improvement in validation performance was observed for M consecutive trials, the optimization process was terminated early.
- **Wall-clock time**: Each HPO method was allocated a maximum wall-clock time budget of T hours to complete its trials.

The value M was set to 10, 20 and 30 for small, medium and large datasets respectively, while T was set to 4, 8 and 12 hours respectively.

4.3 Search Spaces

In our experiments, we used the well-studied and general framework of GraphSAGE [Hamilton et al., 2017] as the base GNN architecture for node classification tasks. We defined a search space encompassing 8 hyperparameters that are critical to the performance of GraphSAGE models. The search space included the ReLU and PReLU activation functions, Adam and AdamW optimizers, dropout rates of 0 and 0.5 and mean and max aggregation functions. Other hyperparameters and their ranges differ across datasets and are listed in Table 1. The ranges were chosen based on prior art on GNNs [Hamilton et al., 2017, Zeng et al., 2019, Zhu et al., 2020, Yang et al., 2023, den Boef et al., 2021, Li et al., 2022].

4.4 Results and Analysis

Table 2 lists the final F1 scores for each HPO method for each dataset, averaged over 10 independent runs. Figure 1 then shows the ranks-over-time of the benchmarked HPO methods over the progress of the optimization.

Three broad observations can be made from the results.

- SMBO methods (BO and TPE) outperform non-SMBO methods (Random, Grid, QMC) consistently across all datasets. Only on the Squirrel and DBLP datasets does any other method reach the second best performance, otherwise SMBO methods take the top two spots. However, between BO and TPE there is no clear winner, with both methods achieving the best performance on multiple datasets. It is also worth noting that the advantage of SMBO methods only becomes pronounced after 10 trials, which is the startup period needed to build an initial model of the search space.
- Simple methods like random search and grid search remain solid choices for HPO as they usually trail behind SMBO methods by only a small margin. The exceptions to this are the Computers and Flickr datasets, where the performance gap is significant.
- Sobol QMC does not fulfill its promise of better coverage of the search space, as it consistently ranks worse than SMBO methods and on smaller datasets also worse than random search.

5 EXPERIMENTAL EVALUATION OF THE CROSS-RF METHOD

In order to evaluate the proposed Cross-RF method, we conducted experiments on the same datasets as described in Section 4, using the same evaluation routine and search space. For better readability, we limit our comparison to only the two highest-performing reference methods from Section 4, which are Bayesian Optimisation (BO) and Tree Parzen Estimators (TPE).

5.1 Metamodel Architecture

Fot the metamodel, we used a scikit-learn Random Forest regressor with 100 trees and 30% of all features considered for each split. Several small implementation details are worth mentioning. First, in each step of the optimisation, all of the already evaluated configurations from $\mathcal{H}$ are excluded from the set of candidate configurations $\tilde{\Lambda}$, so that the same configuration is never needlessly re-evaluated. Second, all categorical features are one-hot encoded before being passed to the Random Forest model. Finally, in all our experiments, we used the F1-score as the performance metric ρ to be optimised.

For the construction of the metadataset $\mathcal{H}$ for a Cross-RF model trained on a dataset $\mathcal{D}$, we used the remaining 8 datasets from the benchmark, excluding $\mathcal{D}$.

5.2 Results and Analysis

Table 3 lists the final F1 scores for Cross-RF and the 2 reference methods for each dataset, averaged over 10 independent runs. We can see that Cross-RF achieves the best final performance on 4 out of the 9 datasets, while being second best on another 3 datasets. Only on the CiteSeer and CoraFull datasets does Cross-RF perform worse than both reference methods. Overall, Cross-RF achieves an average rank of 1.78, outperforming both BO (2.0) and TPE (2.22).

Table 1: Hyperparameters based on dataset sizes.

Dataset	Ep.	Lyrs.	HW	ES	L2	LR
Small datasets						
Cora	200	1–3	16–64	10	$0, 5 \times 10^{-4}, 5 \times 10^{-3}$	$10^{-2}, 5 \times 10^{-3}, 10^{-3}$
CiteSeer	200	1–3	16–64	10	$0, 5 \times 10^{-4}, 5 \times 10^{-3}$	$10^{-2}, 5 \times 10^{-3}, 10^{-3}$
Squirrel	500	1–3	32–128	20	$0, 5 \times 10^{-4}, 5 \times 10^{-3}$	$10^{-2}, 5 \times 10^{-3}, 10^{-3}$
Medium datasets						
PubMed	200	1–3	16–64	10	$0, 5 \times 10^{-4}, 5 \times 10^{-3}$	$10^{-2}, 5 \times 10^{-3}, 10^{-3}$
CoraFull	500	2–3	64–128	20	$0, 5 \times 10^{-5}, 5 \times 10^{-4}$	$10^{-2}, 5 \times 10^{-3}, 10^{-3}$
DBLP	500	2–3	64–128	20	$0, 5 \times 10^{-5}, 5 \times 10^{-4}$	$10^{-2}, 5 \times 10^{-3}, 10^{-3}$
Computers	500	2–3	64–128	20	$0, 5 \times 10^{-5}, 5 \times 10^{-4}$	$10^{-2}, 5 \times 10^{-3}, 10^{-3}$
Flickr	100	2–3	128–256	10	$0, 5 \times 10^{-4}$	$5 \times 10^{-3}, 10^{-3}$
Large datasets						
ArXiv	150	2–3	256–512	10	$0, 5 \times 10^{-4}$	$10^{-2}, 5 \times 10^{-3}$

Abbreviations: **Ep.**: Epochs, **Lyrs.**: Number of layers, **HW**: Hidden Layer Width, **ES**: Early Stopping patience, **L2**: L_2 Regularization, **LR**: Learning Rate.

Table 2: Final F1 scores for each HPO method and for each dataset, averaged over 10 independent runs. The best method is **bold** and the second best is underlined.

Dataset	Random	Grid	BO	TPE	QMC
Cora	0.8560	0.8491	**0.8747**	0.8659	0.8351
CiteSeer	0.7146	0.7046	0.7172	**0.7236**	0.7089
Squirrel	0.3659	0.3605	0.3544	**0.3755**	0.3549
PubMed	0.8515	0.8457	**0.8825**	0.8643	0.8596
CoraFull	0.6211	0.6371	0.6450	**0.6555**	0.6385
DBLP	0.8051	0.7996	**0.8118**	0.8085	0.8088
Computers	0.6973	0.5999	**0.8945**	0.8047	0.7428
Flickr	0.0864	0.0961	**0.1908**	0.1460	0.1069
ArXiv	0.3950	0.3796	0.3987	**0.4098**	0.3955

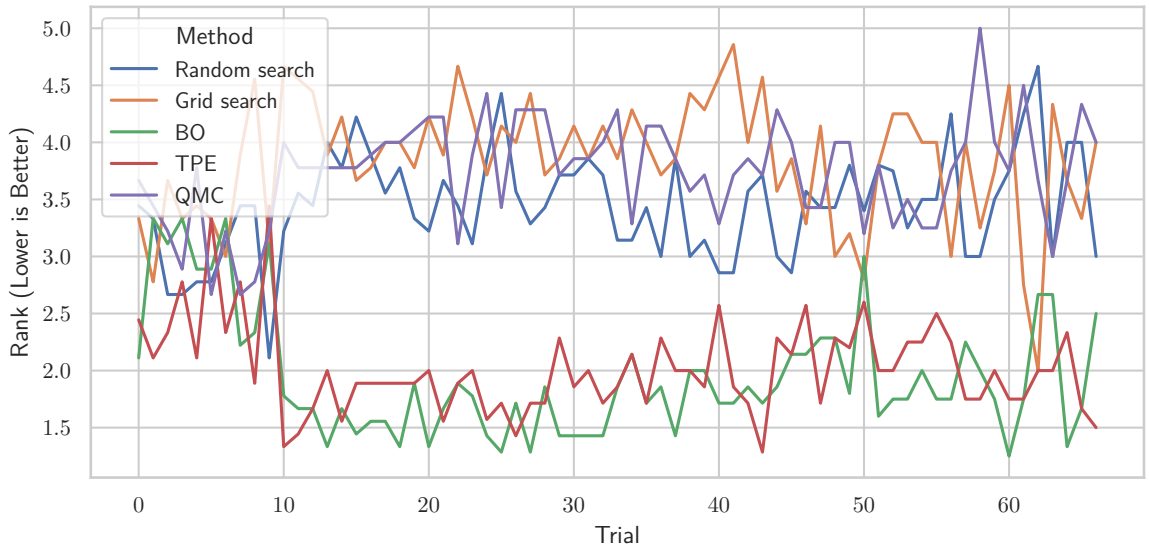

Figure 1: Ranks-over-time of the benchmarked HPO methods over the progress of the optimization, aggreagated over 10 independent runs for each of 9 datasets.

Table 3: Final F1 scores for each HPO method and for each dataset, averaged over 10 independent runs. The best method is **bold** and the second best is underlined.

Dataset	BO	TPE	Cross-RF
Cora	**0.8747**	0.8659	0.8727
CiteSeer	0.7172	**0.7236**	0.7095
Squirrel	0.3544	0.3755	**0.3769**
PubMed	**0.8825**	0.8643	0.8807
CoraFull	0.6450	**0.6555**	0.6426
DBLP	0.8118	0.8085	**0.8123**
Computers	0.8945	0.8047	**0.9023**
Flickr	0.1908	0.1460	**0.1956**
ArXiv	0.3987	**0.4098**	0.4094

6 CONCLUSION AND FUTURE WORK

In this work, we addressed the critical and often-overlooked challenge of hyperparameter optimization (HPO) for Graph Neural Networks (GNNs). While GNNs are increasingly powerful, their performance is highly sensitive to hyperparameter choices, and the computational cost of HPO remains a significant barrier to their effective deployment.

Our first contribution was to conduct a systematic benchmark of five classical HPO methods, evaluating their efficiency and effectiveness on a diverse set of graph-structured problems. The results clearly demonstrate that sophisticated Sequential Model-Based Optimization (SMBO) strategies, such as the Tree-structured Parzen Estimator and Bayesian Optimization, significantly outperform simpler methods like random and grid search. These findings provide concrete, evidence-based recommendations for practitioners, guiding them toward more efficient optimization strategies that yield superior model performance in fewer trials.

Our second contribution was the introduction and validation of a transfer-learning framework for GNN hyperparameter optimization. By leveraging metalearning, we demonstrated that knowledge from previously completed HPO tasks can be effectively transferred to new, unseen datasets. This approach successfully "warm-starts" the optimization process, substantially reducing the number of evaluations required to find a high-performing set of hyperparameters.

6.1 Future Work

Future work focuses on broadening the benchmark and enhancing the transfer learning framework. We plan to extend the evaluation to additional architectures (e.g. GAT), diverse metrics (e.g. accuracy, AUC-ROC), and newer, deep-learning-specific HPO methods. Additionally, we aim to improve Cross-RF performance by pre-training on synthetic datasets (e.g. Erdős–Rényi, stochastic block models), adopting metamodels suited for continuous spaces and incorporating the metamodel as a surrogate within an SMBO framework.

In summary, this work lays a solid foundation for more efficient and effective hyperparameter optimization in GNNs, with significant implications for their practical deployment in real-world applications. The proposed Cross-RF method is promising, however there remains ample opportunity for further research and refinement to fully realize its potential.

REFERENCES

[Akiba et al., 2019] Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631.

[Bergstra et al., 2011] Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc.

[Bergstra and Bengio, 2012] Bergstra, J. and Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13(10):281–305.

[Bischl et al., 2023] Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., Thomas, J., Ullmann, T., Becker, M., Boulesteix, A.-L., Deng, D., and Lindauer, M. (2023). Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *WIREs Data Mining and Knowledge Discovery*, 13(2).

[Bojchevski and Günnemann, 2018] Bojchevski, A. and Günnemann, S. (2018). Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. In *6th International Conference on Learning Representations*.

[den Boef et al., 2021] den Boef, J., Cornelisse, J., and Groth, P. (2021). GraphPOPE: Retaining Structural Graph Information Using Position-aware Node Embeddings. In *DL4KG@ ISWC*.

[Ebadi et al., 2025] Ebadi, A., Kaur, M., and Liu, Q. (2025). Hyperparameter optimization and neural architecture search algorithms for graph Neural Networks in cheminformatics. *Computational Materials Science*, 254.

[Falkner et al., 2018] Falkner, S., Klein, A., and Hutter, F. (2018). BOHB: Robust and Efficient Hyperparameter Optimization at Scale. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1437–1446.

[Gao et al., 2020] Gao, Y., Yang, H., Zhang, P., Zhou, C., and Hu, Y. (2020). Graph Neural Architecture Search. volume 2, pages 1403–1409.

[Hamilton et al., 2017] Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034.

[Hu et al., 2021] Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. (2021). Open Graph Benchmark: Datasets for Machine Learning on Graphs. arXiv:2005.00687 [cs, stat].

[Li et al., 2022] Li, X., Zhu, R., Cheng, Y., Shan, C., Luo, S., Li, D., and Qian, W. (2022). Finding Global Homophily in Graph Neural Networks When Meeting Heterophily. In *Proceedings of the 39th International Conference on Machine Learning*, pages 13242–13256.

[Matoušek, 1998] Matoušek, J. (1998). On the L2-discrepancy for anchored boxes. *Journal of Complexity*, 14(4):527–556.

[Procházka et al., 2023] Procházka, P., Mareš, M., and Dědič, M. (2023). Which Graph Properties Affect GNN Performance for a Given Downstream Task? In *Proceedings of the 23nd Conference Information Technologies – Applications and Theory (ITAT 2023)*, volume 3498 of *CEUR Workshop Proceedings*, pages 58–66, Tatranské Matliare, Slovakia. CEUR-WS.org.

[Rozemberczki et al., 2021] Rozemberczki, B., Allen, C., and Sarkar, R. (2021). Multi-Scale attributed node embedding. *Journal of Complex Networks*, 9(2).

[Shchur et al., 2019] Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. (2019). Pitfalls of Graph Neural Network Evaluation. arXiv:1811.05868 [cs, stat].

[Vodilovska et al., 2023] Vodilovska, V., Gievska, S., and Ivanoska, I. (2023). Hyperparameter Optimization of Graph Neural Networks for mRNA Degradation Prediction. In *2023 46th MIPRO ICT and Electronics Convention (MIPRO)*, pages 423–428.

[Yang et al., 2023] Yang, L., Tian, Y., Xu, M., Liu, Z., Hong, S., Qu, W., Zhang, W., Cui, B., Zhang, M., and Leskovec, J. (2023). VQGraph: Rethinking Graph Representation Space for Bridging GNNs and MLPs.

[Yang et al., 2016] Yang, Z., Cohen, W., and Salakhudinov, R. (2016). Revisiting Semi-Supervised Learning with Graph Embeddings. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 40–48.

[Yuan et al., 2021a] Yuan, Y., Wang, W., and Pang, W. (2021a). A Genetic Algorithm with Tree-structured Mutation for Hyperparameter Optimisation of Graph Neural Networks. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 482–489.

[Yuan et al., 2021b] Yuan, Y., Wang, W., and Pang, W. (2021b). A systematic comparison study on hyperparameter optimisation of graph neural networks for molecular property prediction. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 386–394, New York, NY, USA. Association for Computing Machinery.

[Yuan et al., 2021c] Yuan, Y., Wang, W., and Pang, W. (2021c). Which hyperparameters to optimise? an investigation of evolutionary hyperparameter optimisation in graph neural network for molecular property prediction. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1403–1404.

[Zeng et al., 2019] Zeng, H., Zhou, H., Srivastava, A., Kannan, R., and Prasanna, V. (2019). Graph-SAINT: Graph Sampling Based Inductive Learning Method. In *International Conference on Learning Representations*.

[Zhou et al., 2022] Zhou, K., Huang, X., Song, Q., Chen, R., and Hu, X. (2022). Auto-GNN: Neural architecture search of graph neural networks. *Frontiers in Big Data*, 5. Publisher: Frontiers.

[Zhu et al., 2020] Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., and Koutra, D. (2020). Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. In *Advances in Neural Information Processing Systems*, volume 33, pages 7793–7804, online. Curran Associates, Inc.