# Exploiting All Laplacian Eigenvectors for Node Classification with Graph Transformers

**Vinam Arora**[*]
University of Pennsylvania
Philadelphia, PA, USA
vinam@upenn.edu

**Divyansha Lachi**[*]
University of Pennsylvania
Philadelphia, PA, USA
div11@upenn.edu

**Shivashriganesh P. Mahato**
University of Pennsylvania
Philadelphia, PA, USA
smahato@upenn.edu

**Mehdi Azabou**
Columbia University
New York, NY, USA
ma4766@columbia.edu

**Zihao Chen**
University of Pennsylvania
Philadelphia, PA, USA
zchen959@upenn.edu

**Eva L. Dyer**
University of Pennsylvania
Philadelphia, PA, USA
eva.dyer@upenn.edu

## Abstract

Graph transformers have emerged as powerful tools for modeling complex graph-structured data, offering the ability to capture long-range dependencies. They require explicit positional encodings to inject structural information, which are most commonly derived from the eigenvectors of the graph Laplacian. However, existing approaches utilize only a small set of low-frequency eigenvectors, assuming that smooth global structure is sufficiently informative. We show that, for the task of node classification, it is possible to exploit a much broader spectrum of eigenvectors and achieve significant gains, especially in heterophilic graphs. Additionally, we introduce a first-principles approach for ranking and selecting eigenvectors based on their importance for node classification. Our method is plug-and-play and delivers substantial improvements across diverse benchmarks, elevating even vanilla graph transformers to match or surpass state-of-the-art models.

## 1 Introduction

Graph transformers have become an increasingly powerful tool for learning from graph-structured data, offering flexible mechanisms to model interactions in a graph where otherwise traditional message-passing approaches might fail (e.g. modeling long-range dependencies). A key challenge in adapting transformers, where all nodes can interact with all other nodes, is explicitly encoding the connectivity structure. To address this, most graph transformers rely on positional encodings typically derived from the eigenvectors of the graph Laplacian [13].

In practice, however, these encodings are truncated: only the lowest-frequency eigenvectors are retained [13, 27, 8], with the assumption that they best capture the smooth, global structure needed for learning. This truncation introduces a strong inductive bias that prioritizes low-frequency information while discarding the rest of the spectrum. As a result, we suspect graph transformers may be limited in their ability to capture more localized or smooth patterns, hindering their performance on common real-world classification tasks. This could be problematic in graphs with weak homophily, or long-range interactions, where informative structure may lie in the very spectral components that are being removed.

[*]Equal contribution.

New Perspectives in Graph Machine Learning Workshop
39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Motivated by this hypothesis, we revisit the design of spectral positional encodings in graph transformers. First, we identify a bottleneck causing under-utilization of Laplacian eigenvectors in existing graph transformers. We then show that higher-frequency components often contain critical information for node classification tasks and propose a first-principles approach to selecting eigenvectors based on how class-relevant signals are distributed across the spectrum. Our method is plug-and-play, and compatible with a wide range of graph transformer architectures.

We demonstrate the effectiveness of our approach across a range of node classification and long-range graph benchmarks. Notably, incorporating higher-frequency eigenvectors leads to large gains on challenging heterophilic datasets: performance on Chameleon and Squirrel improves by over 22% and 30%, respectively, even for a simple transformer baseline. More advanced models like NAGphormer [8] and GraphGPS [27] also see improvements exceeding 20% on these datasets.

**Our contributions are as follows:**

- We bring to light simple yet consistently effective modifications to the input encoders of popular graph transformer backbones that enable utilization of broader spectrum of Laplacian eigenvectors.
- We introduce a label-aware strategy for selecting a task-relevant subset of Laplacian eigenvectors to be used as positional encodings in graph transformers. This selection mechanism can be applied as a plug-in to any transformer architecture.
- We conclusively demonstrate, through extensive empirical analysis, that including a well-chosen spectrum of Laplacian eigenvectors leads to *significant* gains in node classification performance of *existing* graph transformers across a wide range of benchmarks.

## 2 Motivation: Why should we broaden the spectrum?

A common assumption in graph transformers that use Laplacian eigenvectors for position embeddings is that low-frequency components alone carry sufficient structural information for node classification. This is reflected in most implementations, which use only a small number of the lowest eigenvectors—typically around 10 (Table 6)—thereby capturing only coarse, smooth patterns in the graph. However, this truncation is arbitrary and lacks both theoretical and empirical justification. If class-discriminative signals exist in higher-frequency bands, restricting to low frequencies may fundamentally limit a model's ability to learn fine-grained structural cues. Our concern is further supported by prior work in the spectral GNN literature, where high-frequency signals have been shown to be particularly effective for addressing heterophily (refer to Section 5). In these cases, low-frequency components often oversmooth node representations, while high-frequency information-often dismissed as noise can capture important local variations.

### 2.1 Initial experiments

We then ask a natural question:

> Can we improve performance by simply increasing the number of low-frequency eigenvectors used in a standard graph transformer?

To explore this, we start with the baseline GT model from [13], which concatenates node features $X \in \mathbb{R}^{N\times D}$ and position encodings $P \in \mathbb{R}^{N\times K}$, and passes them through a single linear layer to create node-level tokens. These are then processed by a vanilla transformer (detailed in Appendix A.1):

$$Y_{\text{GT}} = \text{Transformer}\left([X;P]W_{\text{in}}\right)W_{\text{out}} \tag{1}$$

Here, $P = V_{:,:K}$, where $V \in \mathbb{R}^{N\times N}$ is the Laplacian eigenvector matrix, and $K$ denotes the number of low-frequency eigenvectors included in $P$.

As shown in Figure 1 (dotted lines), naively increasing the number of eigenvectors does not improve performance, suggesting that simply injecting more spectral information is not sufficient. Interestingly, we find that GT starts utilizing a broader spectrum *with two small but critical changes*: (1) applying row-wise $\ell_2$ normalization to the position encodings, and (2) replacing the single input linear layer

with separate MLPs for node and position inputs. We use GT$^*$ to denote this model:

$$Y_{\text{GT}^*} = \text{Transformer}\left([\text{MLP}_{\text{node}}(X); \text{MLP}_{\text{pos}}(\text{norm}(P))]\right) W_{\text{out}} \tag{2}$$

As shown in Figure 1 (dashed lines), these small but targeted changes significantly improve the model's ability to utilize higher-frequency information. Normalization is critical here because the scale of Laplacian eigenvector elements is $\sim 1/N$, which quickly vanishes for reasonably sized graphs. Meanwhile, independent MLPs provide a more expressive mapping from raw inputs to transformer-compatible tokens.

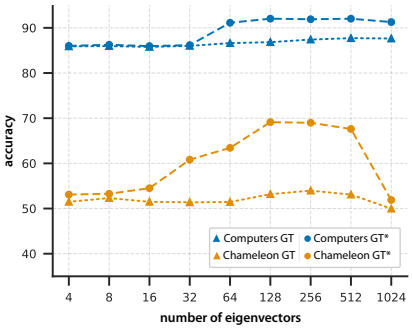


Figure 1: Classification accuracy of GT and GT* as a function of the number of eigenvectors used in position embeddings on two benchmarks.

To validate the effect of input encoder modification, we evaluated GT and GT$^*$ across four node classification benchmarks, while allowing $K$ to be chosen by hyperparameter search (along with the rest of the hyperparameters)[1]. As shown in Table 2, increasing $K$ and switching to the GT$^*$ architecture, boosts performance from 50.48% to 67.83% on Chameleon, and from 34.70% to 62.91% on Squirrel. Furthermore, applying these modifications to two additional graph transformer backbones, NAGphormer [8] and GraphGPS [27], shows consistent improvements (see Table 12 in Appendix F). These results confirm that using more of the graph spectrum can yield substantial gains over prior baselines provided it is paired with appropriate input encoders. With significant gains on both homophilic and heterophilic graphs, we make the following conclusion:

**Conclusion 1**

Standard graph transformers *can* significantly benefit from higher-frequency eigenvectors provided they include (1) proper normalization of position embeddings and (2) expressive input encoders (e.g., MLPs instead of linear layers).

Although these modifications may seem trivial in hindsight, we hypothesize that their absence in existing models explains why the benefits of broadening the graph spectrum within graph transformers have remained undiscovered until now.

These results invite a natural follow-up question: if higher-frequency eigenvectors are useful, what happens if we simply include all of them? In principle, a full-spectrum encoding should maximize the information available to the model. In practice, however, we find that this strategy leads to severe overfitting and degrades performance (Table 2). On `Squirrel`, for example, GT$^*$ drops from 62.9% (tuned) to 35.3% (full), with similar declines observed for NAGphormer and GraphGPS (Appendix F). This finding highlights an important challenge:

**Conclusion 2**

While using more of the graph spectrum is beneficial to some extent, indiscriminately including all eigenvectors harms generalization.

## 3 A label-aware method for spectral selection

Now that Section 2.1 has confirmed that graph transformers *can* indeed benefit from more eigenvectors, we revisit a limitation of that analysis: it still relies on a somewhat arbitrary truncation of the lowest $K$ eigenvectors of the Laplacian. There might still be useful information in the mid- and high-frequency bands. This motivates the need for a more principled method to select an informative, compact, and non-contiguous subset of eigenvectors. We frame this as the following problem:

[1]See Section 4.1 for a complete description of our experimental setup.

Table 2: **Classification results for baseline and modified approaches.** First row represents the typical utilization of Laplacian eigenvectors. Second row allows for using more eigenvectors, which does not result in significant performance improvements. Third row introduces the input encoder modifications and shows a big jump in performance. Finally, the fourth row corresponds to the situation where we provide the model *all* eigenvectors.

| Model | Eigenvector Selection | Chameleon | Squirrel | WikiCS | Computers |
|---|---|---|---|---|---|
| GT | $K \in [4, 16]$ (tuned) | 50.48 ± 2.08 | 34.70 ± 1.77 | 72.91 ± 0.59 | 85.65 ± 0.59 |
| GT | $K \in [4, N]$ (tuned) | 52.28 ± 2.88 | 37.71 ± 1.79 | 73.75 ± 0.63 | 87.21 ± 0.55 |
| GT* | $K \in [4, N]$ (tuned) | 67.83 ± 1.82 | 62.91 ± 1.04 | 78.46 ± 0.56 | 91.66 ± 0.41 |
| GT* | full spectrum (fixed) | 48.14 ± 3.05 | 35.30 ± 1.61 | 72.35 ± 0.72 | 85.20 ± 0.30 |

**Eigenvector selection problem:** Given the $N$ eigenvectors of the graph Laplacian, can we efficiently identify the $K$ most informative ones for node classification?

A naive approach would be to treat this as a subset selection problem over $\binom{N}{K}$ possible combinations—a search space far too large for practical hyperparameter tuning. Instead, we propose a spectral-energy-based method to rank and select eigenvectors. Intuitively, we treat the class labels of training nodes as a graph signal, compute its energy spectral density (ESD), and select the top-$K$ eigenvectors corresponding to frequencies with the highest class-label energy. This provides a simple and interpretable way to prioritize eigenvectors that align most strongly with class structure.

Let $\mathcal{Y}$ denote the label set, $N_{\text{train}}$ be the number of training nodes, and $C_{\text{train}} \in \{0,1\}^{N_{\text{train}} \times |\mathcal{Y}|}$ be the one-hot encoding of class labels for the training nodes. Let $V_{\text{train}} \in \mathbb{R}^{N_{\text{train}} \times N}$ be the matrix of Laplacian eigenvectors, restricted to the training nodes and ordered by increasing eigenvalue. Using the graph Fourier transform (Appendix A.4), we can compute the Fourier representation of the class signal as $\hat{C}_{\text{train}} = V_{\text{train}}^{\top} C_{\text{train}}$. Then the energy spectral density (ESD) of class labels for the $i^{\text{th}}$ graph frequency can be defined as:

$$E_i := \frac{1}{Z} \|\hat{C}_{\text{train},i}\|_2^2 \quad \text{with } Z \text{ chosen such that} \sum_{i=1}^{N} E_i = 1, \tag{3}$$

where $\hat{C}_{\text{train},i}$ is the $i^{\text{th}}$ row of $\hat{C}_{\text{train}}$. We visualize the ESD for several datasets in Figure 2.

To rank the utility of different eigenvectors, we first apply a smoothing operation to the energy density spectrum using a boxcar (moving average) filter.[2] This mitigates the effect of noise and emphasizes consistent frequency bands where class-relevant energy is concentrated. After smoothing, we select the $K$ eigenvectors corresponding to the highest values in the resulting spectrum. The complete procedure is summarized in Algorithm 1.

**Note.** While $K$ remains a hyperparameter, this approach replaces arbitrary low-frequency truncation with a more methodological and data-driven ranking.

## 4 Evaluation

In this section, we evaluate our approach on a diverse set of node classification benchmarks, analyzing its effectiveness across three established graph transformer architectures.

### 4.1 Experimental setup

**Datasets.** To evaluate the effectiveness of incorporating higher-frequency eigenvectors, we conduct experiments on both homophilic and heterophilic datasets. The homophilic datasets include Coauthor-Physics, Coauthor-CS [29], Amazon-Photo, Amazon-Computers [22], and WikiCS [23]. The heterophilic datasets consist of Chameleon, Squirrel [28], Tolokers and Amazon-Ratings [26].

[2] A smoothing window of 256 points is used for all experiments, which we found to work well across datasets.

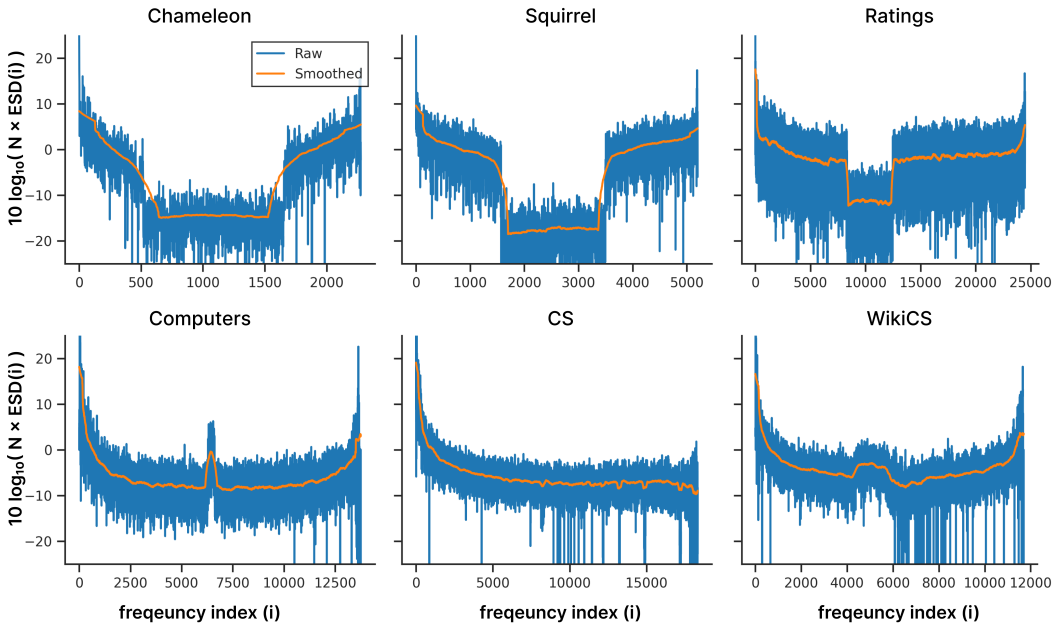


Figure 2: Energy spectral density of class labels for different graphs.

Additionally, we evaluate our approach on two recently introduced long-range datasets from the CityNetwork benchmark suite [18]. This diverse selection ensures that we assess model performance across varying levels of structural homogeneity, heterogeneity, and long-range dependency.

**Baseline and modified models.** We evaluate our approach on three existing graph transformer architectures: GT [13], NAGphormer [8], and GraphGPS [27]. We use the subscript BTS[3] to denote models with our ESD-based eigenvector selection approach (Section 2.1) and input encoder modifications (Section 3).

NAGphormer [8] restricts attention to $k$-hop neighborhoods using a normalized adjacency matrix. In $\text{NAGphormer}_{\text{BTS}}$ , we preserve this localized attention mechanism but replace the original position encoder with our spectral selection procedure and MLP-based encoding. GraphGPS [27] combines message passing with transformer-based global attention and uses LapPE for positional encoding. In $\text{GraphGPS}_{\text{BTS}}$ , we replace LapPE with our label-aware spectral selection and MLP-based encoder, creating a unified position encoding approach consistent with $\text{GT}_{\text{BTS}}$ and $\text{NAGphormer}_{\text{BTS}}$ .To ensure fair comparisons, we use the same training setup and hyperparameter tuning for both the baseline models and their modified versions.

The baseline models are trained with eigenvectors truncated to $K$ lowest frequencies, and $K$ is tuned in the range [4, 16], consistent with prior work (Table 6). In the case of BTS, $K$ is allowed a wider range of $[4, \min(N, 8192)]$, where $N$ denotes number of nodes. Please refer to Appendix E for more details about the hyperparameter tuning setup.

### 4.2 Results

**Results on heterophilic benchmarks.** The results for node classification on heterophilic benchmarks are presented in Table 3. We find substantial improvements when using BTS. For example, on *Chameleon*, performance improves by over 22%, and on *Squirrel*, by over 30% when BTS-filtered eigenvectors are used with a simple transformer architecture. Remarkably, this brings the vanilla transformer architecture (GT) into close competition with, and in some cases even surpassing, more complex graph transformer models proposed in recent literature.

These improvements are not surprising when viewed through the lens of the ESD of class labels. As shown in Figure 2, graphs like *Chameleon* and *Squirrel* exhibit significantly high class energy not only in the low-frequency region, but also in high-frequency components. To the best of our knowledge,

[3]BTS stands for "Broaden the Spectrum."

Table 3: **Node classification performance on heterophilic benchmarks.** Performance numbers for GT/$GT_{BTS}$, NAGphormer/$NAGphormer_{BTS}$, and GraphGPS/$GraphGPS_{BTS}$ were (re-)produced with our consistent experimental setup. Performance for other models are reported from existing literature. "-" indicates absence of a particular evaluation in existing literature. The **top-1st**, **top-2nd**, and **top-3rd** results are highlighted.

| Model | Chameleon Accuracy ↑ | Squirrel Accuracy ↑ | Tolokers AU-ROC ↑ | Ratings Accuracy ↑ |
|---|---|---|---|---|
| GCN | 38.44 ± 1.92 | 31.52 ± 0.71 | 83.64 ± 0.67 | 48.70 ± 0.63 |
| GraphSAGE | 58.73 ± 1.68 | 41.61 ± 0.74 | 82.43 ± 0.44 | **53.63 ± 0.39** |
| GAT | 48.36 ± 1.58 | 36.77 ± 1.68 | 83.70 ± 0.47 | 49.09 ± 0.63 |
| NodeFormer | 36.38 ± 3.85 | 38.89 ± 2.67 | 78.10 ± 1.03 | 43.79 ± 0.57 |
| SGFormer | 45.21 ± 3.72 | 42.65 ± 4.21 | - | **54.14 ± 0.62** |
| Exphormer | - | - | 83.53 ± 0.28 | 50.48 ± 0.34 |
| SpExphormer | - | - | 83.34 ± 0.31 | 50.48 ± 0.34 |
| GT | 50.48 ± 2.08 | 34.70 ± 1.77 | 80.30 ± 0.91 | 49.02 ± 0.61 |
| **$GT_{BTS}$** | **73.09 ± 1.00 +22.61↑** | **65.06 ± 1.93 +30.36↑** | 84.45 ± 0.66 **+4.15↑** | 50.37 ± 0.48 **+1.35↑** |
| NAGphormer | 52.41 ± 2.21 | 40.21 ± 1.77 | 83.69 ± 0.86 | 50.16 ± 0.69 |
| **$NAGphormer_{BTS}$** | **73.90 ± 1.68 +21.49↑** | **65.04 ± 1.69 +24.83↑** | **85.47 ± 0.72 +1.78↑** | 49.65 ± 0.65 **-0.51↓** |
| GraphGPS | 60.92 ± 2.54 | 43.43 ± 1.46 | **86.29 ± 0.68** | 50.19 ± 0.51 |
| **$GraphGPS_{BTS}$** | **73.16 ± 1.70 +12.24↑** | **65.87 ± 1.30 +22.44↑** | **86.31 ± 0.63 +0.02↑** | **51.33 ± 0.58 +1.14↑** |

Table 5: **Node classification accuracy (%) on homophilic benchmarks.** Results for GraphGPS/$GraphGPS_{BTS}$, NAGphormer/$NAGphormer_{BTS}$, and GT/$GT_{BTS}$ were (re-)produced with our consistent experimental setup. Performance for other models are reported from existing literature. The **top-1st**, **top-2nd**, and **top-3rd** results are highlighted.

| Model | Physics Accuracy ↑ | CS Accuracy ↑ | Photo Accuracy ↑ | Computers Accuracy ↑ | WikiCS Accuracy ↑ | ogbn-arXiv Accuracy ↑ |
|---|---|---|---|---|---|---|
| NodeFormer | 96.45 ± 0.28 | 95.64 ± 0.22 | 93.46 ± 0.35 | 86.98 ± 0.62 | 74.73 ± 0.94 | 59.90 ± 0.42 |
| SGFormer | 96.60 ± 0.18 | 94.78 ± 0.34 | 95.10 ± 0.47 | **91.99 ± 0.70** | 73.46 ± 0.56 | **72.63 ± 0.13** |
| Exphormer | 96.89 ± 0.09 | 94.93 ± 0.01 | 95.35 ± 0.22 | 91.47 ± 0.17 | 78.19 ± 0.29 | **71.27 ± 0.27** |
| SpExphormer | 96.70 ± 0.05 | 95.00 ± 0.15 | 95.33 ± 0.49 | 91.09 ± 0.08 | 78.20 ± 0.14 | 70.82 ± 0.24 |
| GT | 96.02 ± 0.20 | 94.66 ± 0.44 | 91.59 ± 0.68 | 85.65 ± 0.59 | 72.91 ± 0.59 | 55.68 ± 0.39 |
| **$GT_{BTS}$** | 96.90 ± 0.18 **+0.88↑** | 95.44 ± 0.33 **+0.78↑** | **95.95 ± 0.48 +4.36↑** | 91.46 ± 0.51 **+5.81↑** | **78.94 ± 0.26 +6.03↑** | 70.30 ± 0.12 **+14.62↑** |
| NAGphormer | 96.98 ± 0.13 | **95.71 ± 0.26** | 95.51 ± 0.41 | 91.39 ± 0.41 | 78.73 ± 0.66 | 69.43 ± 0.32 |
| **$NAGphormer_{BTS}$** | **97.05 ± 0.18 +0.07↑** | 95.42 ± 0.39 **-0.29↓** | **95.90 ± 0.37 +0.39↑** | **91.85 ± 0.44 +0.46↑** | **79.42 ± 0.55 +0.69↑** | **71.29 ± 0.13 +1.86↑** |
| GraphGPS | **97.13 ± 0.17** | **95.70 ± 0.38** | 95.35 ± 0.45 | 91.64 ± 0.46 | 77.67 ± 0.73 | 65.16 ± 1.45 |
| **$GraphGPS_{BTS}$** | **97.21 ± 0.14 +0.08↑** | **95.72 ± 0.37 +0.02↑** | **95.87 ± 0.42 +0.52↑** | **91.87 ± 0.45 +0.23↑** | **79.47 ± 0.48 +1.80↑** | 70.92 ± 0.33 **+5.76↑** |

the performance reported here for *Chameleon*, *Squirrel*, and *Tolokers* represents the strongest results achieved by any graph transformer model to date. These findings highlight that the historical reliance on low-frequency truncation was a critical bottleneck, masking the true representational and generalization potential of graph transformers.

**Results on homophilic benchmarks.** As shown in Table 5, BTS improves performance even on homophilic graphs. $GT_{BTS}$ achieves +5.8% on *Computers*, +6.0% on *WikiCS*, and +14.4% on *ogbn-arXiv*. Gains for NAGphormer and GraphGPS are smaller but consistent. These results are expectedly more modest than in heterophilic settings, as the spectral energy of class signals in homophilic graphs is more concentrated in low frequencies. Still, BTS captures the broader spectrum whenever useful, yielding robust improvements.

**Results on long-range benchmarks.** Graph transformers are naturally suited for capturing long-range dependencies due to their global attention mechanism. However, recent work has shown that graph transformers suffer from overglobalization [38] and perform poorly on long-range tasks. As shown in Table 4, our method achieves substantial improvements on the long-range benchmark [18] over baseline graph transformer architectures. These datasets exhibit particularly strong gains when using BTS-selected features. For instance, performance for GT on `Paris`

Table 4: **Node classification accuracy (%) on Long Range Benchmarks.** The **top-1st**, **top-2nd**, and **top-3rd** results are highlighted.

| Model | Paris | Shanghai |
|---|---|---|
| GCN | 47.30 ± 0.20 | 52.40 ± 0.30 |
| GraphSAGE | 49.10 ± 0.60 | **60.40 ± 0.30** |
| SGFormer | 45.00 ± 0.20 | 53.5 ± 0.30 |
| GT | 15.46 ± 3.93 | 21.05 ± 0.51 |
| **$GT_{BTS}$** | **53.79 ± 0.17** | 52.66 ± 0.83 |
| Δ | **+38.33 ↑** | **+31.61 ↑** |
| NAGphormer | 25.26 ± 0.34 | 24.94 ± 0.28 |
| **$NAGphormer_{BTS}$** | **53.68 ± 0.23** | **57.26 ± 0.34** |
| Δ | **+28.42 ↑** | **+32.32 ↑** |
| GraphGPS | 28.99 ± 0.31 | 28.46 ± 0.31 |
| **$GraphGPS_{BTS}$** | **54.12 ± 0.23** | **55.31 ± 0.33** |
| Δ | **+25.13 ↑** | **+26.85 ↑** |

improves by over 38%, and on `Shanghai` by 31%, bringing the vanilla transformer model on par with strong baseline methods.

## 5 Related Work

**Graph Transformers and Positional Encodings.** Graph Transformers (GTs) allow graph nodes to interact globally via self-attention [13]. Because self-attention is permutation-equivariant, GTs require positional encodings (PEs) to inject structural information. Laplacian eigenvectors have become a common choice [14], providing a spectral basis that reflects graph topology. However, nearly all existing GTs adopt a heuristic truncation: retaining only the first $k$ low-frequency eigenvectors. The low-frequency bias is evident in models such as GT [13], NAGphormer [8], GraphTrans [37], GraphGPS [27], UGT [17], SAN [16], and SAT [7]. Scalability-oriented variants such as ANS-GT [40], Gapformer [19], and Exphormer [31], which focus on efficient attention also retain the same positional encoding heuristic. TokenGT [16] extends beyond low frequencies by including both low- and high-frequency eigenvectors, but relies on fixed splits rather than task-driven selection.

**The Role of High-Frequency Signals in Node Classification.** In contrast, the MPNN literature has increasingly emphasized the importance of high-frequency information for node classification, especially in heterophilic graphs. Spectral methods explicitly operate in the frequency domain and modulate both low- and high-frequency signals [35, 11]. Early spectral models such as Spectral CNN [5] and ChebNet [10] introduced learnable filters over eigenvalues, and subsequent works demonstrated that high-frequency information is critical for node-level expressiveness, such as adaptive propagation [9], complete spectral filtering [21], and frequency-based feature selection [4]. These techniques explicitly exploit high-frequency modes, while adaptive approaches such as AdaGNN [12], and spectral attention [6] dynamically balance contributions from different parts of the spectrum.

## 6 Conclusion

In this work we revisit a widely accepted but rarely questioned assumption in graph transformers: that only a handful of low-frequency Laplacian eigenvectors suffice for positional encodings. We show that this design choice fundamentally limits node classification performance, particularly on heterophilic graphs where high-frequency components capture crucial fine-grained structure. Our analysis demonstrates that graph transformers can benefit substantially from a broader spectral representation, provided two simple adjustments are made—proper normalization of eigenvectors and more expressive input encoders. Building on this insight, we introduced a label-aware spectral selection strategy that ranks eigenvectors by their alignment with class-label energy. This principled approach avoids arbitrary truncation, identifies informative non-contiguous subsets of eigenvectors, and consistently outperforms both low-frequency and full-spectrum encodings. Empirically, it achieves state-of-the-art results across diverse benchmarks and transformer backbones without altering attention mechanisms or adding inference-time cost.

## 7 Funding Disclosure

This project was supported by NSF CAREER Award RI:2146072, NSF award CIF:RI:2212182 as well as generous gifts from the CIFAR Azrieli Global Scholars Program.

## References

[1] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631, 2019.

[2] M. Azabou, V. Ganesh, S. Thakoor, C.-H. Lin, L. Sathidevi, R. Liu, M. Valko, P. Veličković, and E. L. Dyer. Half-hop: A graph upsampling approach for slowing down message passing. In *International Conference on Machine Learning*, pages 1341–1360. PMLR, 2023.

[3] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[4] D. Bo, X. Wang, C. Shi, and H. Shen. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 3950–3957, 2021.

[5] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[6] H. Chang, Y. Rong, T. Xu, W. Huang, S. Sojoudi, J. Huang, and W. Zhu. Spectral graph attention network with fast eigen-approximation. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pages 2905–2909, 2021.

[7] D. Chen, L. O'Bray, and K. Borgwardt. Structure-aware transformer for graph representation learning. In *International Conference on Machine Learning*, pages 3469–3489. PMLR, 2022.

[8] J. Chen, K. Gao, G. Li, and K. He. Nagphormer: A tokenized graph transformer for node classification in large graphs. 2023.

[9] E. Chien, J. Peng, P. Li, and O. Milenkovic. Adaptive universal generalized pagerank graph neural network. *arXiv preprint arXiv:2006.07988*, 2020.

[10] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[11] X. Dong, D. Thanou, L. Toni, M. Bronstein, and P. Frossard. Graph signal processing for machine learning: A review and new perspectives. *IEEE Signal processing magazine*, 37(6):117–127, 2020.

[12] Y. Dong, K. Ding, B. Jalaian, S. Ji, and J. Li. Adagnn: Graph neural networks with adaptive frequency response filter. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pages 392–401, 2021.

[13] V. P. Dwivedi and X. Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

[14] V. T. Hoang, O. Lee, et al. A survey on structure-preserving graph transformers. *arXiv preprint arXiv:2401.16176*, 2024.

[15] V. T. Hoang and O.-J. Lee. Transitivity-preserving graph representation learning for bridging local connectivity and role-based similarity, 2023.

[16] D. Kreuzer, D. Beaini, W. L. Hamilton, V. Létourneau, and P. Tossou. Rethinking graph transformers with spectral attention. 2021.

[17] O.-J. Lee et al. Transitivity-preserving graph representation learning for bridging local connectivity and role-based similarity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 12456–12465, 2024.

[18] H. Liang, H. S. d. O. Borde, B. Sripathmanathan, M. Bronstein, and X. Dong. Towards quantifying long-range interactions in graph machine learning: a large graph dataset and a measurement. *arXiv preprint arXiv:2503.09008*, 2025.

[19] C. Liu, Y. Zhan, X. Ma, L. Ding, D. Tao, J. Wu, and W. Hu. Gapformer: Graph transformer with graph pooling for node classification. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI-23)*, pages 2196–2205, 2023.

[20] I. Loshchilov and F. Hutter. Decoupled weight decay regularization, 2019.

[21] S. Luan, M. Zhao, C. Hua, X.-W. Chang, and D. Precup. Complete the missing half: Augmenting aggregation filtering with diversification for graph convolutional networks. *arXiv preprint arXiv:2008.08844*, 2020.

[22] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 43–52, 2015.

[23] P. Mernyei and C. Cangea. Wiki-cs: A wikipedia-based benchmark for graph neural networks, 2022.

[24] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura, and P. Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.

[25] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.

[26] O. Platonov, D. Kuznedelev, M. Diskin, A. Babenko, and L. Prokhorenkova. A critical look at the evaluation of gnns under heterophily: Are we really making progress? *arXiv preprint arXiv:2302.11640*, 2023.

[27] L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

[28] B. Rozemberczki, C. Allen, and R. Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

[29] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[30] H. Shirzad, H. Lin, B. Venkatachalam, A. Velingker, D. Woodruff, and D. Sutherland. Even sparser graph transformers. *arXiv preprint arXiv:2411.16278*, 2024.

[31] H. Shirzad, A. Velingker, B. Venkatachalam, D. J. Sutherland, and A. K. Sinop. Exphormer: Sparse transformers for graphs. In *International Conference on Machine Learning*, pages 31613–31632. PMLR, 2023.

[32] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.

[33] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B.-J. Hsu, and K. Wang. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th international conference on world wide web*, pages 243–246, 2015.

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. 2017.

[35] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

[36] Q. Wu, W. Zhao, Z. Li, D. Wipf, and J. Yan. Nodeformer: A scalable graph structure learning transformer for node classification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[37] Z. Wu, P. Jain, M. Wright, A. Mirhoseini, J. E. Gonzalez, and I. Stoica. Representing long-range context for graph neural networks with global attention. *Advances in Neural Information Processing Systems*, 34:13266–13279, 2021.

[38] Y. Xing, X. Wang, Y. Li, H. Huang, and C. Shi. Less is more: on the over-globalizing problem in graph transformers. *arXiv preprint arXiv:2405.01102*, 2024.

[39] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T.-Y. Liu. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34:28877–28888, 2021.

[40] Z. Zhang, Q. Liu, Q. Hu, and C.-K. Lee. Hierarchical graph transformer with adaptive node sampling. *Advances in Neural Information Processing Systems*, 35:21171–21183, 2022.

# Appendix

## A Background

### A.1 Transformers

Transformers, introduced in [34], are the foundation of many modern deep learning architectures. Each layer of a transformer consists of a Multi-Head Self-Attention (MHSA) mechanism followed by a Feedforward Network (FFN). Given a sequence of tokens $X \in \mathbb{R}^{N \times D}$, where $N$ is the sequence length and $D$ is the token dimension, the attention layer computes:

$$\text{MHSA}(X) = \sum_{h=1}^{H} \text{softmax}\left(\frac{(XW_Q^h)(XW_K^h)^T}{\sqrt{d}}\right)(XW_V^h)W_O^h \tag{4}$$

The FFN applies a non-linear transformation $\text{FFN}(X) = \sigma(XW_1)W_2$, where $\sigma$ is an activation function such as ReLU. Each transformer block combines MHSA and FFN with residual connections:

$$\tilde{X} = X + \text{MHSA}(X), \quad Y = \tilde{X} + \text{FFN}(\tilde{X}) \tag{5}$$

Stacking such blocks enables the model to capture complex dependencies across the input sequence.

### A.2 Graph transformers

In simple graph transformers [13, 39], each node in the graph is represented by a token. Unlike Message Passing Neural Networks (MPNNs), which restrict pairwise interactions in each layer to a node's immediate neighbors, transformers inherently lack such biases and model all node-to-node interactions by design. Connectivity information about the graph is typically incorporated in one of two ways: either through positional encodings, with the input to the transformer being a combination of the feature vector of each node and its positional encoding [13, 27], or by adding structural bias directly into the attention matrix [36, 31]. In this work, we focus on the former approach.

### A.3 Laplacian positional encodings

Eigenvectors of the normalized graph-Laplacian are the most common positional encodings and have been identified as effective in prior work [13]. To formalize this concept, assume that we are given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}| = N$ nodes, its adjacency matrix $A$, and its degree matrix $D$, its normalized graph-Laplacian matrix is defined as:

$$L = I - D^{-1/2}AD^{-1/2} \tag{6}$$

Let $L = V\Lambda V^T$ denote the eigen decomposition of the normalized graph-Laplacian, with eigenvalues arranged in *increasing order*, i.e. $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_N$. In existing graph transformers [13], a node $n$ is assigned a $K$-dimensional positional encoding $\mathbf{p}_n$ constructed from the Laplacian eigenvectors with the $K$ smallest eigenvalues. $\mathbf{p}_n$ is defined as:

$$\mathbf{p}_n = [\mathbf{v}_{1n}, \mathbf{v}_{2n}, \ldots, \mathbf{v}_{Kn}]^T, \tag{7}$$

leading to an $N \times K$ positional encoding matrix $P = V_{:,:K}$. Here $\mathbf{v}_k \in \mathbb{R}^N$ corresponds to the $k^{\text{th}}$ eigenvector, i.e. the $k^{\text{th}}$ column of $V$.

**Remark.** Each eigenvector $\mathbf{v}_k \in \mathbb{R}^N$ defines a signal $\mathbf{v}_k : \mathcal{V} \to \mathbb{R}$ that varies over the graph nodes. Eigenvectors associated with small eigenvalues are thought to vary slowly across the graph, capturing *low-frequency* global variations, while eigenvectors corresponding to large eigenvalues vary rapidly, encoding *high-frequency* signals that change significantly across neighboring nodes [32, 24].

### A.4 Graph Fourier Transform

The Graph Fourier Transform (GFT) is a generalization of the classical Fourier transform to graph-structured data. Let $L$ denote the normalized graph Laplacian of $\mathcal{G}$, and let $V \in \mathbb{R}^{N \times N}$ be the

matrix of its eigenvectors, ordered by *increasing* eigenvalue. The GFT of a graph signal $\mathbf{x} : \mathcal{V} \to \mathbb{R}$, represented by a vector $\mathbf{x} \in \mathbb{R}^N$, is defined as:

$$\hat{\mathbf{x}} = V^\top \mathbf{x} \tag{8}$$

Here, $\hat{\mathbf{x}}_i = \mathbf{v}_i^\top \mathbf{x}$ denotes the $i^{\text{th}}$ graph-frequency component of $\mathbf{x}$, and $(\hat{\mathbf{x}}_i)^2$ denotes the spectral energy of $\mathbf{x}$ in this frequency. Intuitively, signals that vary smoothly over the graph have most of their energy concentrated in the lower frequencies, while signals that vary rapidly have energy concentrated in the higher frequencies. Note that signals can have significant energies in *both* low and high parts of the spectrum.

# B Highest Maximum Frequencies Used in the Literature

To better understand the typical frequency truncation choices in existing graph transformer models, we compile representative values of the maximum number of Laplacian eigenvectors ($K$) used across a range of published works. As shown in Table 6, most methods restrict $K$ to a small number—often below 16—reinforcing the low-pass inductive bias observed in current practice.

# C Baseline Sources

## C.1 Heterophilic Datasets

We use five real-world datasets with graphs that have a homophily level $\leq 0.30$: Actor [25], Chameleon and Squirrel [28], as well as Ratings and Tolokers [26]. Key statistics for these datasets are listed in Table 7. We follow the experimental setup in [25] for Actor, Chameleon, and Squirrel, and for Ratings and Tolokers, we adopt the setup described in [26], using the 10 train/validation/test splits provided.

The results for GCN-based methods and heterophily-based methods in Table 7 for Actor, Chameleon, and Squirrel have been sourced from [2]. Similarly, results for Ratings and Tolokers are sourced from [26], while results for transformer-based methods across all datasets are obtained from [30].

## C.2 Homophilic Datasets

We use five real-world datasets: Amazon Computers and Amazon Photos [22], Coauthor CS and Coauthor Physics [33], and WikiCS [23]. Key statistics for these datasets are listed in Table 8. The experimental setup follows that of [30], where the datasets are split into development and test sets. All hyperparameter tuning is performed on the development set, and the best models are subsequently evaluated on the test set.

We use a 60:20:20 train/validation/test split for the Amazon and Coauthor datasets. The results reported for all datasets in Table 5 are sourced from [30].

## C.3 Long Range Benchmark Datasets

To evaluate the ability of models to capture long-range dependencies, we use the City-Networks benchmark [18], which consists of large-scale road network graphs derived from OpenStreetMap data.

Table 6: Maximum number of eigenvectors ($K_{\text{max}}$) used by recent graph transformer models, based on publicly available code.

| Model | $K_{\text{max}}$ |
|---|---|
| NAGphormer [8] | 15 |
| GraphGPS [27] | 10 |
| SAN [16] | 10 |
| GT [13] | 10 |
| UGT [15] | 10 |
| Exphormer [31] | 10 |

Table 7: Statistics of heterophilic datasets used in our experiments.

| DATASET | NODES | EDGES | CLASSES | HOMOPHILY RATIO |
|---|---|---|---|---|
| CHAMELEON | 2,277 | 31,421 | 5 | 0.23 |
| SQUIRREL | 5,201 | 198,493 | 5 | 0.22 |
| TOLOKERS | 11,758 | 519,000 | 2 | 0.09 |
| RATINGS | 244,92 | 39,402 | 5 | 0.14 |

Table 8: Statistics of homophilic datasets used in our experiments.

| DATASET | NODES | EDGES | CLASSES | HOMOPHILY RATIO |
|---|---|---|---|---|
| PHYSICS | 34,493 | 495,924 | 5 | 0.92 |
| CS | 18,333 | 81,894 | 15 | 0.83 |
| PHOTO | 7,650 | 238,162 | 8 | 0.84 |
| COMPUTERS | 13,752 | 491,722 | 10 | 0.79 |
| WIKICS | 11,701 | 216,123 | 10 | 0.66 |
| OGBN-ARXIV | 169,343 | 1,166,243 | 40 | 0.65 |

We focus on two representative cities—Paris and Shanghai—which feature grid-like topology, low clustering coefficients, and large diameters. These characteristics make them particularly well-suited for studying long-range signal propagation. Key statistics for these datasets are provided in Table 9.

Following the experimental protocol in [18], we perform transductive node classification using a 10:10:80 train/validation/test split. The node labels are defined by eccentricity-based quantiles, ensuring that the task inherently depends on information from distant nodes.

Table 9: Statistics of City-Networks datasets used in our experiments.

| DATASET | NODES | EDGES | CLASSES | HOMOPHILY RATIO |
|---|---|---|---|---|
| PARIS | 114,127 | 182,511 | 10 | 0.70 |
| SHANGHAI | 183,917 | 262,092 | 10 | 0.75 |

### C.4 Baseline Model Performance Across Datasets from existing literature

The previously reported performance of baseline models (GT, GraphGPS, and NAGphormer) on multiple graph datasets is summarized in Table 10. The reported values, sourced from existing literature.

## D BTS Algorithm Details

For completeness, we provide pseudocode for the BTS eigenvector selection procedure. The algorithm computes the label energy spectrum in the graph Fourier domain, smooths it using a filter, and selects the top-$K$ eigenvectors based on the smoothed spectrum (Algorithm 1).

## E Optimizer Configuration and Hyperparameter Spaces

We use the AdamW optimizer [20] for all runs, and fixed the number of epochs to 200. We additionally employ the linear-warmup-cosine-decay learning rate schedule. Linear rate warmup happens over 10 epochs (fixed), and cosine decay happens over the remaining 190 epochs (also fixed). All other hyperparameters are chosen by the tuning algorithm, which is explained in Section 4.1. The complete hyperparameter space used in our experiments is detailed in Table 11.

We optimize hyperparameters using the Tree-structured Parzen Estimator (TPE) algorithm [3], as implemented in Optuna [1]. The number of tuning trials is adjusted based on the size of each dataset: for graphs with up to 7,500 nodes, we perform 300 tuning trials; for graphs with up to 15,000 nodes,

Table 10: Performance across datasets for GT, GraphGPS, and NAGphormer models previously reported in existing literature.

| Dataset | GT | GraphGPS | NAGphormer |
|---|---|---|---|
| Chameleon | - | 40.79 ± 4.03 | - |
| Squirrel | - | 39.67 ± 2.84 | - |
| Tolokers | - | 83.71 ± 0.48 | 78.32 ± 0.95 |
| Ratings | - | 53.10 ± 0.42 | 51.26 ± 0.72 |
| Physics | 97.05 ± 0.05 | 97.12 ± 0.19 | 97.34 ± 0.03 |
| CS | 94.64 ± 0.13 | 93.93 ± 0.12 | 95.75 ± 0.09 |
| Photo | 94.74 ± 0.13 | 95.06 ± 0.13 | 95.49 ± 0.11 |
| Computers | 91.18 ± 0.17 | 91.19 ± 0.54 | 91.22 ± 0.14 |
| WikiCS | - | 78.66 ± 0.49 | 77.16 ± 0.72 |
| Arxiv | - | 70.97 ± 0.41 | 70.13 ± 0.55 |

**Algorithm 1** Pseudocode for label-aware eigenvector selection

**Input:** Laplacian eigenvectors $\mathbf{V} \in \mathbb{R}^{N \times N}$,
Training node indices $\mathcal{I}_{\text{train}}$,
Training labels $\tilde{\mathbf{C}}_{\text{train}} \in \{0,1\}^{N_{\text{train}} \times C}$,
Number of eigenvectors to choose $K$, Smoothing window size $w$

**Output:** Indices $\mathcal{I}_K$ of selected eigenvectors

$\mathbf{V}_{\text{train}} \leftarrow \mathbf{V}[\mathcal{I}_{\text{train}}]$ ▷ Restrict to training nodes
$\hat{\mathbf{C}}_{\text{train}} \leftarrow \mathbf{V}_{\text{train}}^{\top} \tilde{\mathbf{C}}_{\text{train}}$ ▷ Graph Fourier transform
**for** $i = 1$ to $N$ **do** ▷ Compute energy spectrum
    $\mathbf{E}_i \leftarrow \|\hat{\mathbf{C}}_{\text{train},i}\|_2^2$
$\text{ESD} \leftarrow \mathbf{E} / \sum_{i=1}^{N} \mathbf{E}_i$ ▷ Normalize energy
$\overline{\text{ESD}} \leftarrow \text{BoxcarSmooth}(\text{ESD}, w)$ ▷ Smooth with window-size $w$
$\mathcal{I}_K \leftarrow \text{TopK}(\overline{\text{ESD}}, K)$ ▷ Select top-$K$ energies
**return** $\mathcal{I}_K$ ▷ Return selected indices

we allow 200 trials; and for larger graphs, we limit the number of trials to 100. Hyperparameters are selected based on validation-set performance, and all reported results correspond to test-set performance using the best configurations found. Performing complete hyperparameter tuning on a machine with 4 $\times$ NVidia L40S GPUs takes 2-4 hours depending on the size of the dataset.

# F Ablations

We conducted an ablation study to evaluate the impact of higher-order spectral components, encoder design, and label-aware selection on the Graph Transformer (GT). Here, we extend this analysis to all baseline transformer architectures. As shown in Table 12, the same trends hold across models, showing that access to more eigenvectors and a better encoder design can improve performance.

Table 11: Complete hyperparameter search space for all model variants presented in this paper.

| HYPERPARAMETER | SEARCH SPACE | SAMPLING TYPE |
|---|---|---|
| COMMON PARAMETERS | | |
| LEARNING RATE | $[10^{-4}, 10^{-1}]$ | LOGARITHMIC |
| WEIGHT DECAY | $[10^{-7}, 10^{-2}]$ | LOGARITHMIC |
| DROPOUT | $[0, 0.5]$ | LINEAR |
| ATTENTION DROPOUT | $[0, 0.5]$ | LINEAR |
| WINDOW LENGTH | $\{256, 512, 1024, 2048, 4096\}$ | |
| TRANSFORMER DEPTH | $\{1, 2, \ldots, 8\}$ | LINEAR |
| NUMBER OF ATTENTION HEADS | $\{0, 1, 2, 4, 8\}$ | |
| COMMON FOR $\text{GT}_{\text{BTS}}$ /$\text{NAGPHORMER}_{\text{BTS}}$ /$\text{GRAPHGPS}_{\text{BTS}}$ | | |
| NUMBER OF EIGENVECTORS ($K$) | $\{4, 8, 16, \ldots, 1024\}$ | |
| POS. FEATURE ENCODER - OUTPUT DIMENSION | $\{8, 16, 32, 64, 128\}$ | |
| POS. FEATURE ENCODER - HIDDEN DIMENSION | $\{16, 32, 64, \ldots, 2048\}$ | |
| POS. FEATURE ENCODER - # HIDDEN LAYERS | $\{1, 2, 3, 4\}$ | |
| NODE FEATURE ENCODER - OUTPUT DIMENSION | $\{8, 16, 32, 64, 128\}$ | |
| NODE FEATURE ENCODER - HIDDEN DIMENSION | $\{16, 32, 64, 128\}$ | |
| NODE FEATURE ENCODER - # HIDDEN LAYERS | $\{1\}$ | |
| SPECIFIC FOR GT | | |
| NUMBER OF EIGENVECTORS ($K$) | $\{4, 8, 16\}$ | |
| TOKEN DIMENSION | $\{64, 128, 256, 512\}$ | |
| SPECIFIC FOR NAGPHORMER | | |
| NUMBER OF EIGENVECTORS ($K$) | $\{4, 8, 16\}$ | |
| TOKEN DIMENSION | $\{64, 128, 256, 512\}$ | |
| NUMBER OF HOPS (SAME FOR $\text{NAGPHORMER}_{\text{BTS}}$ ) | $\{1, 2, 3, \ldots, 20\}$ | |
| SPECIFIC FOR GRAPHGPS | | |
| NUMBER OF EIGENVECTORS ($K$) | $\{4, 8, 16\}$ | |
| LAPPE - NUMBER OF LAYERS | $\{1, 2, 3, \ldots, 8\}$ | |
| LAPPE - NUMBER OF POST-LAYERS | $\{0, 1, 2, 3, 4\}$ | |

Table 12: Node classification performance with full eigenvector spectrum vs with left-truncated spectrum but tuned $K$.

| | | **Heterophilic** | | **Homophilic** | |
|---|---|---|---|---|---|
| **Model** | **Eigenvectors** | **Chameleon** | **Squirrel** | **WikiCS** | **Computers** |
| GT | $K \in [4, 16]$ (tuned) | 50.48 ± 2.08 | 34.70 ± 1.77 | 72.91 ± 0.59 | 85.65 ± 0.59 |
| GT | $K \in [4, N]$ (tuned) | 52.28 ± 2.88 | 37.71 ± 1.79 | 73.75 ± 0.63 | 87.21 ± 0.55 |
| GT* | $K \in [4, N]$ (tuned) | 67.83 ± 1.82 | 62.91 ± 1.04 | 78.46 ± 0.56 | 91.66 ± 0.41 |
| GT* | full spectrum (fixed) | 48.14 ± 3.05 | 35.30 ± 1.61 | 72.35 ± 0.72 | 85.20 ± 0.30 |
| GT-BTS | BTS, $K \in [4, N]$ (tuned) | 73.09 ± 1.68 | 65.06 ± 1.93 | 78.94 ± 0.26 | 91.46 ± 0.51 |
| NAGphormer | $K \in [4, 16]$ (tuned) | 52.41 ± 2.21 | 40.21 ± 1.77 | 78.73 ± 0.66 | 91.39 ± 0.41 |
| NAGphormer | $K \in [4, N]$ (tuned) | 57.06 ± 1.96 | 40.89 ± 2.06 | 79.70 ± 0.50 | 91.61 ± 0.42 |
| NAGphormer* | $K \in [4, N]$ (tuned) | 70.07 ± 2.33 | 63.87 ± 1.51 | 79.83 ± 0.63 | 91.96 ± 0.37 |
| NAGphormer* | full spectrum (fixed) | 59.63 ± 2.06 | 52.27 ± 1.28 | 79.63 ± 0.63 | 91.53 ± 0.47 |
| NAGphormer-BTS | BTS, $K \in [4, N]$ (tuned) | 73.90 ± 1.68 | 65.04 ± 1.69 | 79.42 ± 1.55 | 91.85 ± 0.44 |
| GraphGPS | $K \in [4, 16]$ (tuned) | 60.92 ± 2.54 | 43.43 ± 1.46 | 77.67 ± 0.73 | 91.64 ± 0.46 |
| GraphGPS | $K \in [4, N]$ (tuned) | 64.67 ± 2.98 | 47.12 ± 4.21 | 77.40 ± 0.45 | 91.60 ± 0.45 |
| GraphGPS* | $K \in [4, N]$ (tuned) | 70.24 ± 2.08 | 63.40 ± 1.16 | 78.68 ± 0.46 | 91.80 ± 0.40 |
| GraphGPS* | full spectrum (fixed) | 59.14 ± 1.95 | 42.88 ± 1.77 | 77.42 ± 0.98 | 91.24 ± 0.40 |
| GPS-BTS | BTS, $K \in [4, N]$ (tuned) | 73.16 ± 1.70 | 65.87 ± 1.30 | 79.47 ± 0.48 | 91.87 ± 0.45 |