# Rethinking Graph Transformers with Spectral Attention

**Devin Kreuzer** *
McGill University, Mila
Montreal, Canada
devin.kreuzer@mail.mcgill.ca

**Dominique Beaini** *
Valence Discovery
Montreal, Canada
dominique@valencediscovery.com

**William L. Hamilton**
McGill University, Mila
Montreal, Canada
wlh@cs.mcgill.ca

**Vincent Létourneau**
University of Ottawa
Ottawa, Canada
vletour2@uottawa.ca

**Prudencio Tossou**
Valence Discovery
Montreal, Canada
prudencio@valencediscovery.com

## Abstract

In recent years, the Transformer architecture has proven to be very successful in sequence processing, but its application to other data structures, such as graphs, has remained limited due to the difficulty of properly defining positions. Here, we present the *Spectral Attention Network* (SAN), which uses a learned positional encoding (LPE) that can take advantage of the full Laplacian spectrum to learn the position of each node in a given graph. This LPE is then added to the node features of the graph and passed to a fully-connected Transformer. By leveraging the full spectrum of the Laplacian, our model is theoretically powerful in distinguishing graphs, and can better detect similar sub-structures from their resonance. Further, by fully connecting the graph, the Transformer does not suffer from over-squashing, an information bottleneck of most GNNs, and enables better modeling of physical phenomenons such as heat transfer and electric interaction. When tested empirically on a set of 4 standard datasets, our model performs on par or better than state-of-the-art GNNs, and outperforms any attention-based model by a wide margin, becoming the first fully-connected architecture to perform well on graph benchmarks.

## 1 Introduction

The prevailing strategy for graph neural networks (GNNs) has been to directly encode graph structure structure through a sparse message-passing process [14, 16]. In this approach, vector messages are iteratively passed between nodes that are connected in the graph. Multiple instantiations of this message-passing paradigm have been proposed, differing in the architectural details of the message-passing apparatus (see [16] for a review).

There is a growing trend across deep learning towards more flexible architectures, which avoid strict and structural inductive biases. Most notably, the exceptionally successful Transformer architecture removes any structural inductive bias by encoding the structure via soft inductive biases, such as

*Equal contribution.

Fourth Workshop on Machine Learning and the Physical Sciences (NeurIPS 2021).

positional encodings [27]. In the context of GNNs, the self-attention mechanism of a Transformer can be viewed as passing messages between all nodes, regardless of the input graph connectivity.

In this work, we offer a principled investigation of how Transformer architectures can be applied in graph representation learning. **Our primary contribution** is the development of novel and powerful learnable positional encoding methods, which are rooted in spectral graph theory. Our positional encoding (PE) technique — and the resulting *spectral attention network (SAN)* architecture — addresses key theoretical limitations in prior graph Transformer work [11] and provably exceeds the expressive power of standard message-passing GNNs. We show that full Transformer-style attention provides consistent empirical gains compared to an equivalent sparse message-passing model, and we demonstrate that our SAN architecture is competitive with or exceeding the state-of-the-art on several well-known graph benchmarks. An overview of the entire method is presented in Figure 4 in the appendix, with a link to the anonymous code here: `https://anonymous.4open.science/r/SAN-5C8C`.

## 2 Theoretical Motivations and Physical Insights

### 2.1 Absolute and relative positional encoding with eigenfunctions

In this section, we investigate how eigenfunctions of the Laplacian can be used to define absolute and relative PEs in graphs and measure physical interactions between nodes in graphs.

#### 2.1.1 Eigenvectors equate to sine functions over graphs

In the Transformer architecture, a fundamental aspect is the use of sine and cosine functions as PEs for sequences [27]. However, sinusoids cannot be clearly defined for arbitrary graphs, since there is no clear notion of position along an axis. Instead, their equivalent is given by the eigenvectors $\boldsymbol{\phi}$ of the graph Laplacian $\boldsymbol{L}$. Indeed, in a Euclidean space, the Laplacian (or Laplace) operator corresponds to the divergence of the gradient and its eigenfunctions are sine/cosine functions, with the squared frequencies corresponding to the eigenvalues (we sometimes interchange the two notions from here on). Hence, in the graph domain, the eigenvectors of the graph Laplacian are the natural equivalent of sine functions, and this intuition was employed in multiple recent works which use the eigenvectors as PEs for GNNs [12], for directional flows [3] and for Transformers [11].

Being equivalent to sine functions, we naturally find that the Fourier Transform of a function $\mathscr{F}[f]$ applied to a graph gives $\mathscr{F}[f](\lambda_i) = \langle f, \boldsymbol{\phi}_i \rangle$, where the eigenvalue is considered as a position in the Fourier domain of that graph [5]. Thus, the eigenvectors are best viewed as vectors positioned on the axis of eigenvalues rather than components of a matrix as illustrated in Figure 1.

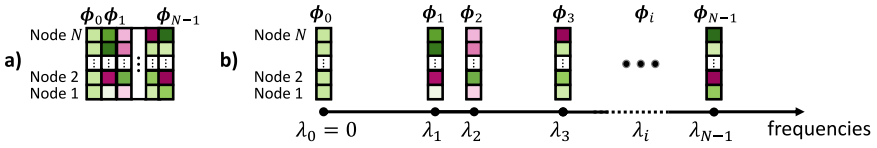


Figure 1: a) Standard view of the eigenvectors as a matrix. b) Eigenvectors $\boldsymbol{\phi}_i$ viewed as vectors positionned on the axis of frequencies (eigenvalues).

#### 2.1.2 What do eigenfunctions tell us about relative positions?

Importantly, the eigenvectors of the Laplacian also hold important information about the physics of a system and can reveal distance metrics. This is not surprising as the Laplacian is a fundamental operator in physics and is notably used in Maxwell's equations [13] and the heat diffusion [5].

In electromagnetic theory, the (pseudo)inverse of the Laplacian, known in mathematics as the Green's function of the Laplacian [7], represents the electrostatic potential of a given charge. In a graph, the same concept uses the pseudo-inverse of the Laplacian $\boldsymbol{G}$ and can be computed by its eigenfunctions. See equation 1 , where $\boldsymbol{G}(j_1, j_2)$ is the electric potential between nodes $j_1$ and $j_2$, $\hat{\boldsymbol{\phi}}_i$ and $\hat{\lambda}_i$ are the $i$-th eigenvectors and eigenvalues of the symmetric Laplacian $\boldsymbol{D}^{\frac{-1}{2}}\boldsymbol{L}\boldsymbol{D}^{\frac{-1}{2}}$, and $\boldsymbol{D}$ is the degree

matrix, and $\hat{\phi}_{i,j}$ the $j$-th row of the vector.

$$\boldsymbol{G}(j_1, j_2) = d_{j_1}^{\frac{1}{2}} d_{j_2}^{\frac{-1}{2}} \sum_{i>0} \frac{(\hat{\phi}_{i,j_1} \hat{\phi}_{i,j_2})^2}{\hat{\lambda}_i} \tag{1}$$

Further, the original solution of the heat equation given by Fourier relied on a sum of sines/cosines known as a Fourier series [6]. As eigenvectors of the Laplacian are the analogue of these functions in graphs, we find similar solutions. Knowing that heat kernels are correlated to random walks [5, 3], we use the interaction between two heat kernels to define in equation 2 the diffusion distance $d_D$ between nodes $j_1, j_2$ [5, 8]. Similarly, the biharmonic distance $d_B$ was proposed as a better measure of distances [22]. Here we use the eigenfunctions of the regular Laplacian $\boldsymbol{L}$.

$$d_D^2(j_1, j_2) = \sum_{k>0} e^{-2t\lambda_i} (\boldsymbol{\phi}_{i,j_1} - \boldsymbol{\phi}_{i,j_2})^2 \quad , \quad d_B^2(j_1, j_2) = \sum_{i>0} \frac{(\boldsymbol{\phi}_{i,j_1} - \boldsymbol{\phi}_{i,j_2})^2}{\lambda_i^2} \tag{2}$$

### 2.1.3 Hearing the shape of a graph and its sub-structures

Another well-known property of eigenvalues is how they can be used to discriminate between different graph structures and sub-structures, as they can be interpreted as the frequencies of resonance of the graph. This led to the famous question about whether we can hear the shape of a drum from its eigenvalues [19], with the same questions also applying to geometric objects [10] and 3D molecules [25]. Various success was found with the eigenfunctions being used for partial functional correspondence [24], algorithmic understanding geometries [21], and style correspondence [10]. Examples of eigenvectors for molecular graphs are presented in Figure 2.

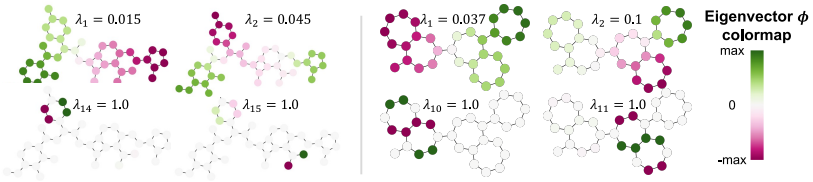


Figure 2: Examples of eigenvalues $\lambda_i$ and eigenvectors $\boldsymbol{\phi}_i$ for molecular graphs. The low-frequency eigenvectors $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2$ are spread accross the graph, while higher frequencies, such as $\boldsymbol{\phi}_{14}, \boldsymbol{\phi}_{15}$ for the left molecule or $\boldsymbol{\phi}_{10}, \boldsymbol{\phi}_{11}$ for the right molecule, often resonate in local structures.

## 2.2 Laplace Eigenfunctions *etiquette*

In the following section, we present key principles from spectral graph theory to consider when constructing PEs for graphs, most of which have been overlooked by prior methods.

**Normalization**. Given an eigenvalue of the Laplacian, there is an associated eigenspace of dimension greater than 1. To make use of this information in our model, a single eigenvector has to be chosen.

**Eigenvalues**. Another fundamental aspect is that the eigenvalue associated with each eigenvector supplies valuable information. An ordering of the eigenvectors based on their eigenvalue works in sequences since the frequencies are pre-determined. However, this assumption does not work in graphs since the eigenvalues in their spectrum can vary. For example, in Figure 2, we observe how an ordering would miss the fact that both molecules resonate at $\lambda = 1$ in different ways.

**Multiplicities**. Another important problem with choosing eigenfunctions is the possibility of a high multiplicity of the eigenvalues, i.e. when an eigenvalue appears as a root of the characteristic polynomial more than once. In this case, the associated eigenspace may have dimension 2 or more as we can generate a valid eigenvector from any linear combination of eigenvectors with the same eigenvalue.

**Variable number of eigenvectors**. A graph $G_i$ can have at most $N_i$ linearly independent eigenvectors with $N_i$ being its number of nodes. Prior work [11] elected to select a fixed number $k$ eigenvectors for each graph, where $k \leq N_i, \forall i$. This produces a major bottleneck when the smallest graphs have significantly fewer nodes than the largest graphs in the dataset since a very small proportion of eigenvectors will be used for large graphs. This inevitably causes loss of information and motivates

the need for a model which constructs fixed PEs of dimension $k$, where $k$ does not depend on the number of eigenvectors in the graph.

**Sign invariance**. As noted earlier, there is a sign ambiguity with the eigenvectors. With the sign of $\phi$ being independent of its normalization, we are left with a total of $2^k$ possible combination of signs when choosing $k$ eigenvectors of a graph.

## 3 Model Architecture

Here, we propose a method for learning node PEs motivated by the principles from section 2.2. The idea of our LPE is inspired by Figure 1, where the eigenvectors $\phi$ are represented as a non-uniform sequence with the eigenvalue $\lambda$ being the position on the frequency axis.

The proposed LPE architecture is presented in Figure 5 in the appendix. First, we create an embedding matrix of size $2 \times m$ for each node $j$ by concatenating the $m$-lowest eigenvalues with their associated eigenvectors. Here, $m$ is a hyper-parameter for the maximum number of eigenvectors to compute and is analog to the variable-length sequence for a standard Transformer. For graphs where $m > N$, a masked-padding is simply added. A linear layer is then applied on the dimension of size 2 to generate new embeddings of size $k$. A Transformer Encoder then computes self-attention on the sequence of length $m$ and hidden dimension $k$. Finally, a sum pooling reduces the sequence into a fixed $k$-dimensional node embedding. This embedding is then concatenated to each nodes' initial feature vector, before being passed to the Main Graph Transformer, whose details are described in the appendix.

The LPE model addresses key limitations of previous graph Transformers and is aligned with the first four *etiquettes* presented in section 2.2. Furthermore, the model has access to the entire spectrum (if selecting $m = N_{max}$, where $N_{max}$ is the largest number of nodes a graph has in the dataset) and is theoretically capable of learning the physical interactions described in 2.1.2. This is especially important when the graph models physical, chemical, or biological structures, but can also help understanding pixel interaction in images [1, 2].

## 4 Experimental Results

When comparing to the state-of-the-art (SOTA) models in the literature in Figure 3, we observe that our SAN model consistently performs better on all synthetic datasets from [12], highlighting the strong expressive power of the model. Other top-performing models, namely PNA [9] and DGN [3], use a message-passing approach [14] with multiple aggregators. When compared to attention-based models, SAN consistently outperforms the SOTA by a wide margin. To the best of our knowledge, SAN is the first fully-connected model to perform well on graph tasks, as is evident by the poor performance of the *GT (full)* model.

| | ZINC | PATTERN | CLUSTER | MOLHIV |
|---|---|---|---|---|
| **Model** | **MAE** | **% ACC** | **% ACC** | **% ROC-AUC** |
| GCN | 0.367 ± 0.011 | 71.892 ± 0.334 | 68.498 ± 0.976 | 76.06 ± 0.97 |
| GraphSage | 0.398 ± 0.002 | 50.492 ± 0.001 | 63.844 ± 0.110 | - |
| GatedGCN | 0.282 ± 0.015 | 85.568 ± 0.088 | 73.840 ± 0.326 | - |
| GatedGCN-PE | 0.214 ± 0.013 | 86.508 ± 0.085 | 76.082 ± 0.196 | |
| GIN | 0.526 ± 0.051 | 85.387 ± 0.136 | 64.716 ± 1.553 | 75.58 ± 1.40 |
| PNA | 0.142 ± 0.010 | - | - | 79.05 ± 1.32 |
| DGN | - | - | - | **79.70 ± 0.97** |
| **Attention-based** | | | | |
| GAT | 0.384 ± 0.007 | 78.271 ± 0.186 | 70.587 ± 0.447 | - |
| GT (sparse) | 0.226 ± 0.014 | 84.808 ± 0.068 | 73.169 ± 0.662 | - |
| GT (full) | 0.598 ± 0.049 | 56.482 ± 3.549 | 27.121 ± 8.471 | - |
| SAN (ours) | **0.139 ± 0.006** | **86.581 ± 0.037** | **76.691 ± 0.247** | 77.85 ± 0.65 |


Figure 3: Comparing our tuned model on datasets from [12, 17], against GCN [20], GraphSage [15], GIN [30], GAT [28], GatedGCN [4], PNA [9], and DGN [3]. Means and uncertainties are derived from four runs with different seeds, except MolHIV which uses 10 runs with identical seed. The number of parameters is fixed to $\sim 500k$ for ZINC, PATTERN and CLUSTER.

## 5 Societal Impact

The presented work is focused on theoretical and methodological improvements to graph neural networks, so there are limited direct societal impacts. However, indirect negative impacts could be caused by malicious applications developed using the algorithm. One such example is the tracking of people on social media by representing their interaction as graphs, thus predicting and influencing their behavior towards an external goal. It also has an environmental impact due to the greater energy use that arises from the computational cost $O(m^2N + N^2)$ being larger than standard message passing or convolutional approaches of $O(E)$.

## References


[1] Dominique Beaini, Sofiane Achiche, Alexandre Duperré, and Maxime Raison. Deep green function convolution for improving saliency in convolutional neural networks. *The Visual Computer*, 37(2):227–244, 2020.

[2] Dominique Beaini, Sofiane Achiche, and Maxime Raison. Improving convolutional neural networks via conservative field regularisation and integration.

[3] Dominique Beaini, Saro Passaro, Vincent Létourneau, William L. Hamilton, Gabriele Corso, and Pietro Liò. Directional graph networks. *ICML2021*, 2021.

[4] Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017.

[5] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[6] F. Cajori. *A History of Mathematics*. AMS Chelsea Publishing Series. AMS Chelsea, 1999.

[7] Fan Chung and S. T. Yau. Discrete green's functions. *Journal of Combinatorial Theory, Series A*, 91(1):191–214, 2000.

[8] Ronald R. Coifman and Stéphane Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, 2006. Special Issue: Diffusion Maps and Wavelets.

[9] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *arXiv preprint arXiv:2004.05718*, 2020.

[10] Luca Cosmo, Mikhail Panine, Arianna Rampini, Maks Ovsjanikov, Michael M. Bronstein, and Emanuele Rodola. Isospectralization, or how to hear shape, style, and correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[11] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs, 2020.

[12] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

[13] Richard Phillips Feynman, Robert Benjamin Leighton, and Matthew Sands. *The Feynman lectures on physics; New millennium ed.* Basic Books, New York, NY, 2010. Originally published 1963-1965.

[14] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272. JMLR. org, 2017.

[15] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

[16] William L. Hamilton. *Graph Representation Learning*. Morgan and Claypool, 2020.

[17] 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. *arXiv preprint arXiv:2005.00687*, 2020.

[18] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. *arXiv:1802.04364 [cs, stat]*, 2018.

[19] Mark Kac. Can one hear the shape of a drum? *The American Mathematical Monthly*, 73(4):1, 1966.

[20] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[21] B. Levy. Laplace-beltrami eigenfunctions towards an algorithm that "understands" geometry. In *IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*, pages 13–13, 2006.

[22] Yaron Lipman, Raif M. Rustamov, and Thomas A. Funkhouser. Biharmonic distance. *ACM Trans. Graph.*, 29(3), July 2010.

[23] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[24] Emanuele Rodolà, Luca Cosmo, Michael M. Bronstein, Andrea Torsello, and Daniel Cremers. Partial functional correspondence, 2015.

[25] Joshua Schrier. Can one hear the shape of a molecule (from its coulomb matrix eigenvalues)? *Journal of Chemical Information and Modeling*, 60(8):3804–3811, 2020. Publisher: American Chemical Society.

[26] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey, 2020.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[28] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[29] 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. *arXiv preprint arXiv:1909.01315*, 2019.

[30] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

## Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? Yes
   (b) Did you describe the limitations of your work? Yes, mainly in section D.
   (c) Did you discuss any potential negative societal impacts of your work? Yes, in section 5.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? Yes.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? Yes, assumptions are provided.
   (b) Did you include complete proofs of all theoretical results? No proofs were required.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? Yes. The URL is included in the final sentence of the Introduction, with the link repeated here `https://anonymous.4open.science/r/SAN-5C8C`.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? Yes. All hyper-parameters are specified in E.2. The splits are provided by the public benchmarks [12, 17].

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? Yes. Each experiment was run 4 or 10 times, with reported mean and standard deviation.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? Yes, provided in appendix E.3.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? Yes. See section 4. The reference for the datasets are [12, 17], for the main frameworks are [23, 29], and for the MIT-licensed code we used [11].

(b) Did you mention the license of the assets? Yes. All under MIT license.

(c) Did you include any new assets either in the supplemental material or as a URL? Not applicable.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? Yes. The datasets are provided publicly, and cited appropriately [12, 17].

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? Not applicable. As presented in the appendix E.1, the data is either artificially generated, or on a molecular dataset for HIV inhibition.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? Not applicable.

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? Not applicable.

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? Not applicable.

# A Overview of SAN

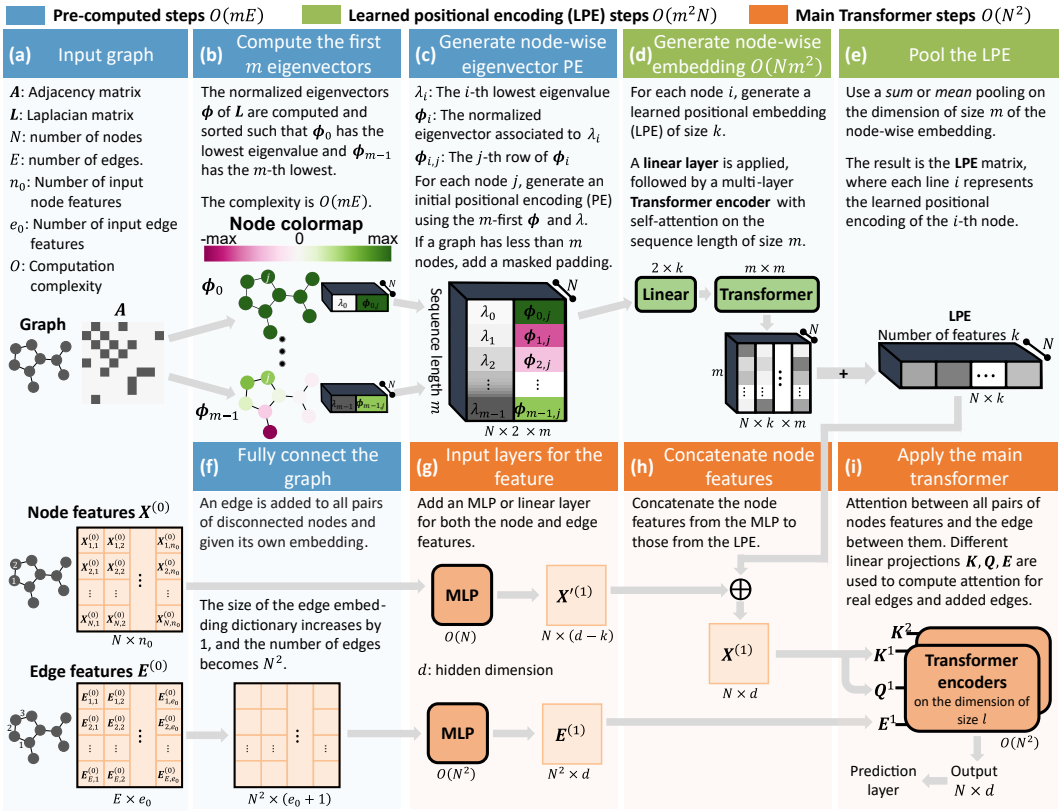


Figure 4: The proposed SAN model with the node LPE, a generalization of Transformers to graphs.

# B LPE Transformer

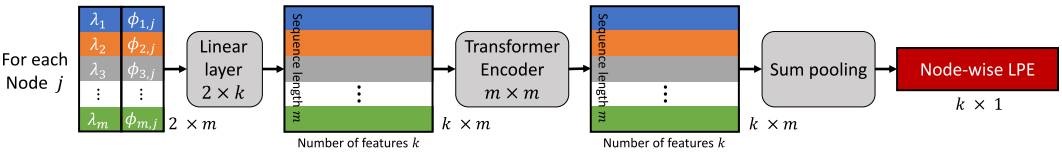


Figure 5: Learned positional encoding (LPE) architectures, with the model being aware of the graph's Laplace spectrum by considering $m$ eigenvalues and eigenvectors, where we permit $m \leq N$, with $N$ denoting the number of nodes. Since the Transformer loops over the nodes, each node can be viewed as an element of a batch to parallelize the computation. Here $\phi_{i,j}$ is the $j$-th element of the eigenvector paired to the $i$-th lowest eigenvalue $\lambda_i$.

# C Details of the Main Graph Transformer

In the following, note that $\boldsymbol{h}_i^l$ is the $i$-th node's features at the $l$-th layer, and $\boldsymbol{e}_{ij}$ is the edge feature embedding between nodes $i$ and $j$. Our model employs multi-head attention over all nodes:

$$\hat{\boldsymbol{h}}_i^{l+1} = \boldsymbol{O}_h^l \Big\|_{k=1}^{H} \Big(\sum_{j \in V} w_{ij}^{k,l} \boldsymbol{V}^{k,l} \boldsymbol{h}_j^l\Big) \tag{3}$$

where $\boldsymbol{O}_h^l \in \mathbb{R}^{d \times d}$, $\boldsymbol{V}^{k,l} \in \mathbb{R}^{d_k \times d}$, $H$ denotes the number of heads, $L$ the number of layers, and $||$ concatenation. Note that $d$ is the hidden dimension, while $d_k$ is the dimension of a head ($\frac{d}{H} = d_k$).

A key addition from our work is the design of an architecture that performs full-graph attention while preserving local connectivity with edge features via two sets of attention mechanisms: one for nodes connected by real edges in the sparse graph and one for nodes connected by added edges in the fully-connected graph. The attention weights $w_{ij}^{k,l}$ in equation 3 at layer $l$ and head $k$ are given by:

$$\hat{\boldsymbol{w}}_{ij}^{k,l} = \left\{ \begin{array}{ll} \frac{\boldsymbol{Q}^{1,k,l}\boldsymbol{h}_i^l \circ \boldsymbol{K}^{1,k,l}\boldsymbol{h}_j^l \circ \boldsymbol{E}^{1,k,l}\boldsymbol{e}_{ij}}{\sqrt{d_k}} & \text{if } i \text{ and } j \text{ are connected in sparse graph} \\ \frac{\boldsymbol{Q}^{2,k,l}\boldsymbol{h}_i^l \circ \boldsymbol{K}^{2,k,l}\boldsymbol{h}_j^l \circ \boldsymbol{E}^{2,k,l}\boldsymbol{e}_{ij}}{\sqrt{d_k}} & \text{otherwise} \end{array} \right\} \tag{4}$$

$$w_{ij}^{k,l} = \left\{ \begin{array}{ll} \frac{1}{1+\gamma} \cdot \text{softmax}(\sum_{d_k} \hat{\boldsymbol{w}}_{ij}^{k,l}) & \text{if } i \text{ and } j \text{ are connected in sparse graph} \\ \frac{\gamma}{1+\gamma} \cdot \text{softmax}(\sum_{d_k} \hat{\boldsymbol{w}}_{ij}^{k,l}) & \text{otherwise} \end{array} \right\} \tag{5}$$

where $\circ$ denotes element-wise multiplication and $\boldsymbol{Q}^{1,k,l}$, $\boldsymbol{Q}^{2,k,l}$, $\boldsymbol{K}^{1,k,l}$, $\boldsymbol{K}^{2,k,l}$, $\boldsymbol{E}^{1,k,l}$, $\boldsymbol{E}^{2,k,l} \in \mathbb{R}^{d_k \times d}$. $\gamma \in \mathbb{R}^+$ is a hyperparameter which tunes the amount of bias towards full-graph attention, allowing flexibility of the model to different datasets and tasks where the necessity to capture long-range dependencies may vary. Note that softmax outputs are clamped between $-5$ and $5$ for numerical stability and that the keys, queries and edge projections are different for pairs of connected nodes ($\boldsymbol{Q}^1, \boldsymbol{K}^1, \boldsymbol{E}^1$) and disconnected nodes ($\boldsymbol{Q}^2, \boldsymbol{K}^2, \boldsymbol{E}^2$).

A multi-layer perceptron (MLP) with residual connections and normalization layers are then applied to update representations, in the same fashion as the GT method [11].

$$\hat{\hat{\boldsymbol{h}}}^{l+1} = \text{Norm}(\boldsymbol{h}_i^l + \hat{\boldsymbol{h}}_i^{l+1}), \quad \hat{\hat{\boldsymbol{h}}}_i^{l+1} = \boldsymbol{W}_2^l \text{ReLU}(\boldsymbol{W}_1^l \hat{\hat{\boldsymbol{h}}}_i^{l+1}), \quad \boldsymbol{h}_i^{l+1} = \text{Norm}(\hat{\hat{\boldsymbol{h}}}^{l+1} + \hat{\hat{\boldsymbol{h}}}_i^{l+1}) \tag{6}$$

with the weight matrices $\boldsymbol{W}_1^l \in \mathbb{R}^{2d \times d}$, $\boldsymbol{W}_2^l \in \mathbb{R}^{d \times 2d}$. Edge representations are not updated as it adds complexity with little to no performance gain. Bias terms are omitted for presentation.

# D Limitations

The first limitation of the node-wise LPE, and noted in Table **??** is the lack of sign invariance of the model. A random sign-flip of an eigenvector can produce different outputs for the LPE, meaning that the model needs to learn a representation invariant to these flips. We resolve this issue with the edge-wise LPE proposed in **??**, but it comes at a computational cost.

Another limitation of the approach is the computational complexity of the LPE being $O(m^2N)$, or $O(N^3)$ if considering all eigenfunctions. Further, as nodes are batched in the LPE, the total memory on the GPU will be *num_params* * *num_nodes_in_batch* instead of *num_params* * *batch_size*. Although this is limiting, the LPE is not parameter hungry, with $k$ usually kept around 16. Most of the model's parameters are in the *Main Graph Transformer* of complexity $O(N^2)$.

Despite Transformers having increased complexity, they managed to revolutionalize the NLP community. We argue that to shift away from the message-passing paradigm and generalize Transformers to graphs, it is natural to expect higher computational complexities. This is exacerbated by sequences being much simpler to understand than graphs due to their linear structure. Future work could overcome this by using variations of Transformers that scale linearly or logarithmically [26].

# E Appendix - Implementation details

## E.1 Benchmarks and datasets

To test our models' performance, we rely on standard benchmarks proposed by [12] and [17] and provided under the MIT license. In particular, we chose ZINC, PATTERN, CLUSTER, and MolHIV.

**ZINC [12]**. A synthetic molecular graph regression dataset, where the predicted score is given by the subtraction of computationally estimated properties $logP - SA$. Here, $logP$ is the computed octanol-water partition coefficient, and $SA$ is the synthetic accessibility score [18].

**CLUSTER [12]**. A synthetic benchmark for node classification. The graphs are generated with Stochastic Block Models, a type of graph used to model communities in social networks. In total, 6

communities are generated and each community has a single node with its true label assigned. The task is to classify which nodes belong to the same community.

**PATTERN [12]**. A synthetic benchmark for node classification. The graphs are generated with Stochastic Block Models, a type of graph used to model communities in social networks. The task is to classify the nodes into 2 communities, testing the GNNs ability to recognize predetermined subgraphs.

**MolHIV [17]**. A real-world molecular graph classification benchmark. The task is to predict whether a molecule inhibits HIV replication or not. The molecules in the training, validation, and test sets are divided using a scaffold splitting procedure that splits the molecules based on their two-dimensional structural frameworks. The dataset is heavily imbalanced towards negative samples. It is also known that this dataset suffers from a strong de-correlation between validation and test set performance, meaning that more hyperparameter fine-tuning on the validation set often leads to lower test set results.

### E.2 SOTA Comparison study

For the results in Figure 3, we tuned some of the hyperparameters, using the following strategies. The optimal parameters are in **bold**.

**ZINC**. Due to the $500k$ parameter budget, we tuned the pairing $\{$*GT layers*, *GT hidden dimension*$\} \in \{\{6, 72\}, \{8, 64\}, \textbf{\{10,56\}}\}$ and *readout* $\in \{$"mean", **"sum"**$\}$ **PATTERN**. Due to the $500k$ parameter budget and long training times, we only tuned the pairing $\{$*GT layers*, *GT hidden dimension*$\} \in \{\textbf{\{4,80\}}, \{6, 64\}\}$ **CLUSTER**. Due to the $500k$ parameter budget and long training times, we only tuned the pairing $\{$*GT layers*, *GT hidden dimension*$\} \in \{\{12, 64\}, \textbf{\{16,48\}}\}$ **MolHIV**. With no parameter budget, we elected to do a more extensive parameter tuning in a two-step process while measuring validation metrics on 3 runs with identical seeds.

1. We tuned *LPE dimension* $\in \{8, \mathbf{16}\}$, *GT layers* $\in \{4, 6, 8, \mathbf{10}\}$, *GT hidden dimension* $\in \{48, \mathbf{64}, 72, 80, 96\}$
2. With the highest performing validation model from step 1, we then tuned *dropout* $\in \{0, \mathbf{0.01}, 0.025\}$ and *weight decay* $\in \{\mathbf{0}, 10^{-6}, 10^{-5}\}$

With the final optimized parameters, we reran 10 experiments with identical seeds.

### E.3 Computation details

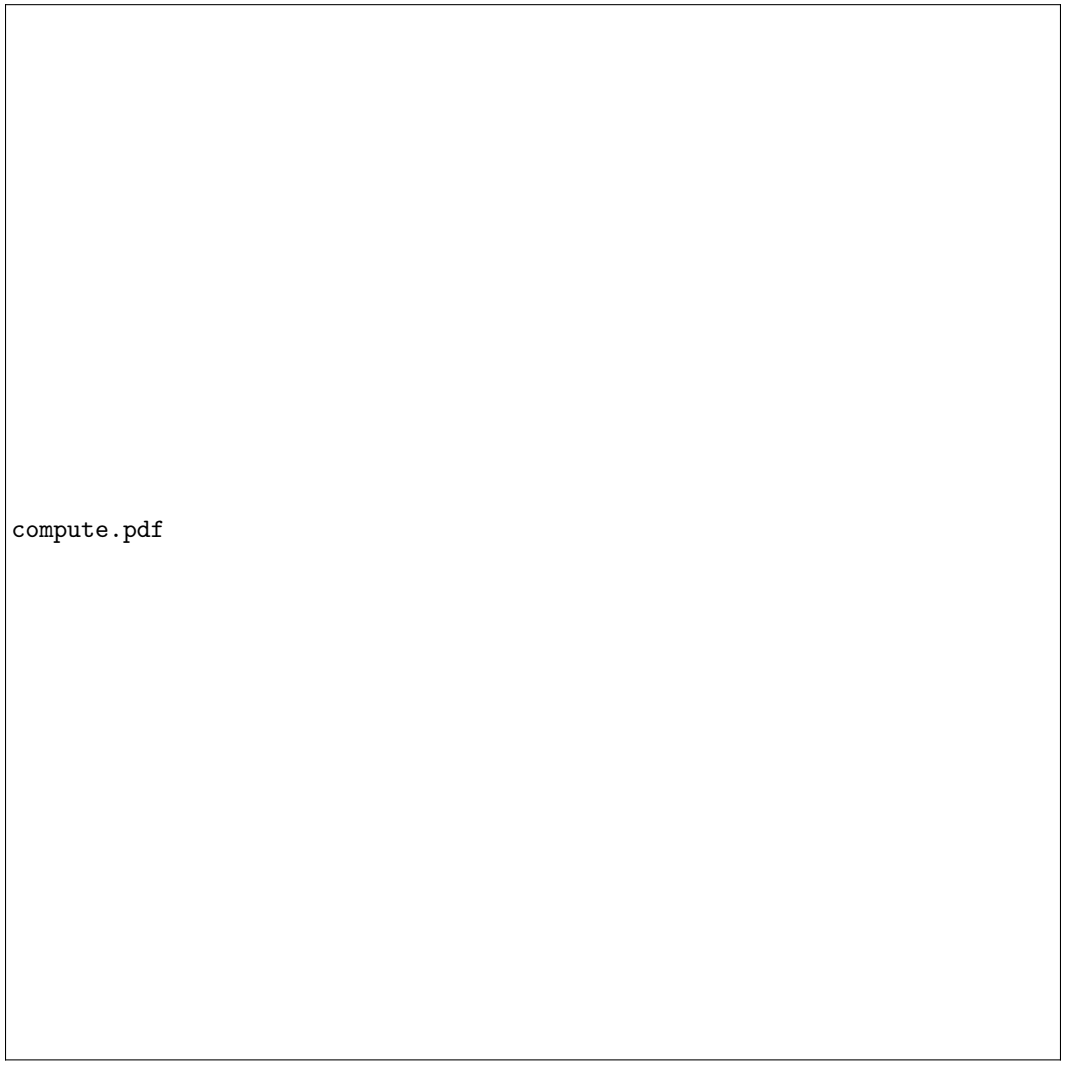


Figure 6: Computational details for SOTA Comparison study.