# PATH LAPLACIAN ENCODINGS FOR DIRECTED GRAPHS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Directed graphs naturally model many real-world systems in which interactions are *asymmetric*, such as citation networks, web graphs, and information-flow networks. However, graph learning methods commonly rely on message passing with symmetrized graph representations or positional encodings that only partially exploit edge directionality. We introduce *PathLapPE*, a novel spectral positional encoding (PE) derived from the path Laplacian on directed graphs. PathLapPE provides node- and edge-level features that encode directional higher-order structure and can be incorporated into standard graph learning architectures. Empirical results on node- and graph-level benchmark tasks show that PathLapPE yields consistent improvements across several architectures, especially when combined with direction-aware message passing. Compared with magnetic Laplacian positional encodings, a widely studied spectral positional encoding for directed graphs, PathLapPE does not require additional fine-tuning of directionality hyperparameters while offering competitive runtime and performance.

## 1 INTRODUCTION

Many real-world relational systems are naturally represented by directed graphs: citations point from one document to another, hyperlinks encode navigation, circuits and programs contain logical dependencies, and biological or transportation networks often involve intrinsically asymmetric interactions. In such settings, edge orientation is not a secondary attribute but an integral part of the semantic content of the data. Nevertheless, a large fraction of graph neural network architectures either assume undirected graphs or preprocess directed graphs by replacing each directed edge with an undirected one. This naive symmetrization can destroy key information for downstream tasks.

Several approaches have been proposed to incorporate directionality into graph learning architectures. These include modifying message passing so that information flowing along incoming and outgoing edges is processed differently (Dir-GNN, Rossi et al., 2024), utilizing directed Laplacians (MagNet, Zhang et al., 2021), and injecting directed information through positional or structural encodings (PSE), attention biases, or edge features (Geisler et al., 2023; Huang et al., 2025).

Positional and structural encodings have become central tools in modern graph learning by enriching node and edge representations (Dwivedi et al., 2022; Cantürk et al., 2024; Huang et al., 2024). Positional encodings typically aim to distinguish where nodes lie in the graph, for instance through Laplacian eigenvectors, random-walk features, distances, or resistance-based quantities. Structural encodings, in contrast, describe the role or local environment of a node or edge, such as degrees, orbit counts, local subgraph statistics, or other invariants.

Despite substantial progress on positional encodings for undirected graphs, direction-aware positional encodings remain comparatively underdeveloped. Existing directed encodings include directed walk encodings (Geisler et al., 2023), walk profiles (Huang et al., 2025), and magnetic Laplacian encodings (Geisler et al., 2023; Huang et al., 2025). These methods have demonstrated that direction-aware encodings can significantly improve the expressivity of graph Transformers and GNNs on tasks where orientation matters. However, magnetic-Laplacian-based encodings also come with practical limitations. Their behavior is sensitive to the choice of the magnetic phase parameter $q$, and strong performance often requires either careful tuning of this parameter or the concatenation of encodings associated with several values of $q$. While multi-$q$ constructions improve the ability of magnetic encodings to represent directed walk patterns, they also require several complex-valued



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

spectral decompositions and increase the size of the resulting positional encoding. This creates a trade-off between directional expressivity and computational simplicity.

In this work, we propose *PathLapPE*, a positional encoding for directed graphs constructed from the *path Laplacians* of the directed path complex. Instead of encoding direction through complex phases, PathLapPE uses the algebraic structure of directed paths to produce real-valued node- and edge-level spectral features. Our contributions are as follows:

1. We introduce *PathLapPE*, a spectral PE derived from the path complex of a directed graph. It provides node- and edge-level features, with the latter encoding both edge orientation and directed 2-path structure. Unlike magnetic Laplacian PEs, it requires no phase parameter tuning or multi-$q$ stacking. It can be easily incorporated into standard GNN and graph Transformer architectures. To the best of our knowledge, this is the first direction-aware positional encoding for graph learning based on the path Laplacian.
2. We introduce *EdgeSPE*, an extension of SPE (Huang et al., 2024) to edge-level encodings, which allows for performance gains with edge-level PathLapPE features.
3. We show that PathLapPE captures directionality and consistently matches or outperforms other spectral PEs on an array of node- and graph-level directed benchmarks across several GNN architectures, with the largest gains for direction-aware models such as bidirectional and edge-gated GNNs.

## 2 Related Work

**Bidirectional graph neural networks.** Classical message passing neural networks (MPNNs) update node representations by aggregating features from local neighborhoods. Architectures such as GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), and GIN (Xu et al., 2019) differ in their choices of aggregation and update functions and usually assume an undirected graph or operate on a symmetrized version of the input. This can be problematic for directed graphs, where incoming and outgoing edges may encode different types of relationships.

Recent work has therefore proposed message passing schemes that explicitly preserve edge orientation. Bidirectional (or directed) GNN adapts standard message passing architectures to directed graphs by aggregating messages from in-neighbors and out-neighbors separately (Jaume et al., 2019; Huang et al., 2025; Rossi et al., 2024). Given a graph $G = (V, \vec{E})$ and node features $\{\mathbf{h}_v\}_{v\in V}$, the features for the next layer $\{\mathbf{h}_v\}_{v\in V}$ after message passing can be written as

$$\mathbf{h}'_v = \alpha\,\psi_{\text{in}}\Big(\mathbf{h}_v,\ \underset{u\in\mathcal{N}_{\text{in}}(v)}{\text{AGG}}\ \phi_{\text{in}}(\mathbf{h}_u,\mathbf{h}_v)\Big) + (1-\alpha)\,\psi_{\text{out}}\Big(\mathbf{h}_v,\ \underset{u\in\mathcal{N}_{\text{out}}(v)}{\text{AGG}}\ \phi_{\text{out}}(\mathbf{h}_u,\mathbf{h}_v)\Big),$$

where the $\alpha$ parameter controls the weight assigned to each direction. Other directed or edge-aware variants, including GAT- and GIN-type architectures with edge features, can also exploit orientation-dependent information when it is made available through the input representation. Our work is complementary to these architectural approaches: rather than modifying the message passing scheme, we construct direction-aware spectral features that encode higher-order relations that these message passing schemes can better leverage.

**Spectral and directed positional encodings.** Positional encodings enrich node or edge representations with information about their location in the graph. Spectral positional encodings are commonly obtained from eigenvectors of a graph operator, most often the Laplacian. Given a Laplacian $\mathbf{L}$, a $k$-dimensional Laplacian positional encoding (LapPE) assigns to each node $v$ the vector $\text{PE}(v) = (\mathbf{u}_1(v), \ldots, \mathbf{u}_k(v))$, where $\mathbf{u}_1, \ldots, \mathbf{u}_k$ are the first $k$ eigenvectors of $\mathbf{L}$ (i.e., those corresponding to the smallest eigenvalues). These encodings capture global geometric information that may not be available to purely local message passing layers. However, Laplacian eigenvectors come with sign and basis ambiguities: each eigenvector is defined only up to sign, and eigenspaces associated with repeated eigenvalues are defined only up to a change of orthonormal basis. This has motivated invariant and equivariant approaches to spectral encodings, such as SignNet, BasisNet (Lim et al., 2023), and SPE (Huang et al., 2024).

For directed graphs, positional encodings must also account for edge orientation. Existing approaches include directed walk profiles and magnetic-Laplacian-based encodings. The magnetic Laplacian is a complex Hermitian operator whose phases encode edge direction, and it has been used both in directed spectral neural architectures such as MagNet and as a source of direction-aware positional encodings (Zhang et al., 2021; Geisler et al., 2023). Recent multi-$q$ magnetic Laplacian (Multi-$q$-MagLapPE) encodings concatenate spectral features obtained from several choices of the magnetic phase parameter $q$, improving the ability to represent directed walk profiles (Huang et al., 2025). This additional expressivity, however, comes with the need to tune, compute, and store several complex-valued spectral decompositions and increases the size of the resulting encoding.

Our method, PathLapPE, is similar in spirit to these magnetic approaches but differs in its underlying Laplacian. Instead of encoding orientation through complex phases, it uses the path Laplacians associated with the directed path complex (Grigor'yan, 2022). This yields real-valued encodings that can be used at both the node and edge levels.

**Path complexes and directed motifs in graph learning.** Higher-order directed structure has entered graph learning in two main ways. MotifNet (Monti et al., 2018) defines convolutions from symmetric node-level Laplacians of a predefined set of directed motifs, without lifting the graph. A second line of work lifts the graph to a path complex (Truong & Chin, 2024)[1], and runs message passing on that domain, in analogy with simplicial and cellular architectures. However, such higher-order message-passing schemes are computationally expensive due to the processing of higher-order tensors (Maron et al., 2019), and the theoretical expressivity guarantees provided by higher-order MPNNs typically fail to account for the practical limitations of GNNs, leading only to modest performance gains at a significant computational cost (Kemper et al., 2025).

Our approach differs from both. PathLapPE does not require choosing motifs, as directed higher-order structures arise naturally from the boundary maps of the path complex. In contrast to lifted architectures, we use the path Laplacian only to compute node- and edge-level positional encodings; the downstream model then runs standard message passing on the original graph, so it receives directional and higher-order information from the path complex without the cost of higher-order message passing.

## 3 From directed graphs to positional encodings

### 3.1 Path Laplacian

Path Laplacians are a family of Laplace-type operators for directed graphs, derived from the boundary maps of the path complex. We briefly recall here the construction of path complexes and path Laplacians associated with directed graphs, and refer the reader to Grigor'yan (2022) and Grigor'yan et al. (2014) for more details.

We consider a directed graph $G = (V, \vec{E})$ and assume throughout that $G$ has no self-loops. An *elementary $p$-path* on $V$ is an ordered sequence of vertices $v_0 v_1 \dots v_p$, where repetitions are allowed (and we do not require that $(v_i, v_{i+1}) \in \vec{E}$). We call a *$p$-path* any finite formal sum $\sigma = \sum_{v_0,\dots,v_p \in V} \lambda_{v_0 \dots v_p} \, v_0 \dots v_p$, with coefficients $\lambda_{v_0 \dots v_p} \in \mathbb{R}$. We define the vector space $\Lambda_p$ as the space of all $p$-paths where elements of the form $v_i v_i$ are identified with the zero vector. An elementary $p$-path $v_0 v_1 \dots v_p$ is called *allowed* if $(v_i, v_{i+1})$ is a directed edge in $\vec{E}$ for all $0 \le i \le p-1$. We denote by $\mathcal{A}_p$ the vector space spanned by the allowed $p$-paths. For any $p > 0$, we define the boundary map $\partial_p : \Lambda_p \to \Lambda_{p-1}$ by first defining

$$\partial_p(v_0 \dots v_p) = \sum_{k=0}^{p} (-1)^k v_0 \dots v_{k-1} v_{k+1} \dots v_p, \tag{1}$$

for elementary $p$-paths and then extending $\partial_p$ to all of $\Lambda_p$ by linearity. We set $\Lambda_{-1} = \mathcal{A}_{-1} = \{0\}$ to be the zero vector space and let $\partial_0 : \Lambda_0 \to \Lambda_{-1}$ be the zero map. These maps satisfy $\partial_{p-1} \circ \partial_p = 0$,

[1] The path complex of Truong & Chin (2024) is based on simple paths in simple graphs, with a path identified with its reverse, and differs from the directed path complex used here.

and therefore define a chain complex (see Grigor'yan et al. (2020, Lemma 2.1))

$$\cdots \xrightarrow{\partial_{p+1}} \Lambda_p \xrightarrow{\partial_p} \Lambda_{p-1} \xrightarrow{\partial_{p-1}} \cdots .$$

**Example 1.** *Consider the directed triangle graph shown in Figure 1(a). The graph consists of three vertices or 0-paths $a, b$, and $c$. Any ordered sequence of the vertices defines an elementary path, for instance, the ordered pair $ca$ is an elementary 1-path. However, $ca$ is not an allowed path because $(c, a) \notin \vec{E}$. This graph has exactly three allowed 1-paths corresponding to the three edges in $\vec{E}$, $u := ab, v := bc$, and $w := ac$. The only allowed 2-path is given by $t := abc$, which can be obtained by concatenating the directed edges $(a, b)$ and $(b, c)$. To understand the definition of $\mathcal{A}_1$, we observe that $u$ and $v$ are allowed 1-paths and so the (non-elementary) 1-path defined by $u + 2v$ is an element of $\mathcal{A}_1$, the vector space spanned by allowed 1-paths, even though it is not itself an allowed 1-path.*

The following example illustrates that, in general, the boundary of an allowed path is not necessarily an allowed path and that the spaces $\mathcal{A}_p$ do not necessarily form a chain complex.

**Example 2.** *Consider the directed path graph with three vertices, $V = \{a, b, c\}$ and two edges, $\vec{E} = \{(a, b), (b, c)\}$ (shown as a subgraph in Figure 1(a)). Then, the 2-path $abc$ is an allowed path and, by definition, $\partial_2(abc) = bc - ac + ab$. However, $ac$ is not an allowed 1-path and there is no way to write $\partial_2(abc)$ as a linear combination of the allowed paths. Thus, $\partial_2(abc) \notin \mathcal{A}_1$.*

In order to obtain a chain complex, one considers the subspace $\Omega_p$ of *$\partial$-invariant allowed paths*,

$$\Omega_p = \{\sigma \in \mathcal{A}_p : \partial_p(\sigma) \in \mathcal{A}_{p-1}\}. \tag{2}$$

In this paper, we will mostly be interested in the vector space $\Omega_p$ and so, unless otherwise stated, the boundary maps will always refer to the linear maps $\partial_p : \Omega_p \to \Omega_{p-1}$ obtained by restricting the boundary maps in Equation 1 (denoted by the same symbol in a minor abuse of notation). We equip each $\Lambda_p$ with the standard inner product for which the elementary $p$-paths form an orthonormal basis, and we equip $\Omega_p \subseteq \Lambda_p$ with the induced inner product. The adjoint of the boundary map $\partial_p : \Omega_p \to \Omega_{p-1}$ is denoted by $\partial_p^* : \Omega_{p-1} \to \Omega_p$. The $p$-path Laplacian is defined by $\mathbf{L}_p = \partial_{p+1}\partial_{p+1}^* + \partial_p^*\partial_p$. Equivalently, if $\mathbf{B}_p$ denotes the matrix representation of $\partial_p$, then, in an orthonormal basis, the $p$-path Laplacian has the matrix representation (also denoted $\mathbf{L}_p$ in a minor abuse of notation)

$$\mathbf{L}_p = \mathbf{B}_{p+1}\mathbf{B}_{p+1}^\top + \mathbf{B}_p^\top\mathbf{B}_p. \tag{3}$$

The first term $\mathbf{B}_{p+1}\mathbf{B}_{p+1}^\top$ is called the *up Laplacian*, while the second term $\mathbf{B}_p^\top\mathbf{B}_p$ is called the *down Laplacian*. Furthermore, unlike magnetic-Laplacian constructions, the path Laplacian is obtained directly from the directed path structure of the graph and does not require the choice of an additional phase parameter. Moreover, because the operator acts on spaces of directed paths, it naturally produces spectral information at different orders. A fully worked example is provided in Example 3 (Appendix C).

### 3.2 Path Laplacian PEs

In this work, for computational efficiency, we construct path complexes up to path length $p = 2$. The corresponding path Laplacians provide spectral information at both the vertex and edge levels: the 0-path Laplacian $\mathbf{L}_0$ acts on vertices and is used to define node-level positional encodings, whereas the 1-path Laplacian $\mathbf{L}_1$ acts on directed edges and is used to define edge-level positional encodings. An explicit formula for $\mathbf{L}_0$ is given by

$$\mathbf{L}_0(u, v) = \mathbf{D}(u, v) - \big(\mathbf{A}(u, v) + \mathbf{A}(v, u)\big),$$

where $\mathbf{D}$ is the diagonal degree matrix associated with $\mathbf{A} + \mathbf{A}^\top$, and $\mathbf{A}$ is the adjacency matrix defined by $\mathbf{A}(u, v) = 1$ if $(u, v) \in \vec{E}$, and zero otherwise. We note that if $G$ has no bidirectional edges, then $\mathbf{L}_0$ is equivalent to the traditional (combinatorial) graph Laplacian on the symmetrized graph associated with $G$ (constructed by placing an (undirected) edge between $u$ and $v$ if $(u, v) \in \vec{E}$ or $(v, u) \in \vec{E}$). Moreover, the edge Laplacian is given by

$$\mathbf{L}_1 = \mathbf{B}_2\mathbf{B}_2^\top + \mathbf{B}_1^\top\mathbf{B}_1. \tag{4}$$

The edge-level operator incorporates both incidence information through the down Laplacian and directed two-step path information through the up Laplacian. These encodings can then be incorporated into graph neural networks or graph Transformers as additional structural features, allowing the model to access directional higher-order information beyond the symmetrized adjacency structure.

**Beyond edge directionality.** We highlight here a feature of the path Laplacian that makes it a natural backbone for extracting positional encodings. As discussed above, two-step relations are encoded in the edge Laplacian $\mathbf{L}_1$ through the up-Laplacian term $\mathbf{B}_2\mathbf{B}_2^\top$; see Equation 4. Moreover, Grigor'yan et al. (2014, Proposition 2.9) shows that the space $\Omega_2$ of $\partial$-invariant allowed 2-paths admits a basis spanned by the directed motifs in Figure 1. Thus, when the boundary map $\partial_2 : \Omega_2 \to \Omega_1$ is represented in such a basis, the matrix $\mathbf{B}_2$ encodes the incidence relations between edges and directed 2-path motifs. Although we do not explicitly fix this basis in practice, this perspective suggests that the up Laplacian $\mathbf{B}_2\mathbf{B}_2^\top$ captures structural information beyond edge directionality alone.

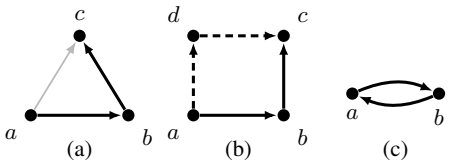



Figure 1: Basis elements of $\Omega_2$: invariant elementary 2-paths in (a) and (c), and the difference of the solid and dashed 2-paths in (b).

**A basis for $\Omega_2$.** The space of $p$-paths admits a decomposition $\Lambda_p = \mathcal{A}_p \oplus \mathcal{D}_p$ into the direct sum of $\mathcal{A}_p$ and the vector space $\mathcal{D}_p$ spanned by the non-allowed elementary $p$-paths. Then, for any $p \geq 1$, the restriction of the boundary map defined in Eq. (1) to the space of allowed paths $\partial_p : \mathcal{A}_p \to \Lambda_{p-1}$ can be expressed as a sum of two maps $\partial_p = \partial_p^{\mathcal{A}} + \partial_p^{\mathcal{D}}$ defined by $\partial_p^{\mathcal{A}} = \pi_{p-1}^{\mathcal{A}} \circ \partial_p$, and $\partial_p^{\mathcal{D}} = \pi_{p-1}^{\mathcal{D}} \circ \partial_p$, where $\pi_{p-1}^{\mathcal{A}}$ and $\pi_{p-1}^{\mathcal{D}}$ denote the projection maps onto $\mathcal{A}_{p-1}$ and $\mathcal{D}_{p-1}$, respectively. The vector space of $\partial$-invariant paths is precisely $\Omega_p = \ker \partial_p^{\mathcal{D}}$. The matrix representation $\mathbf{B}_p$ of the restricted boundary map $\partial_p : \Omega_p \to \Omega_{p-1}$ depends on choices of bases for $\Omega_p$, and $\Omega_{p-1}$. Constructing such bases is a key step in the construction of the path Laplacian in Equation 3.

In general, computing a kernel basis for large matrices can be time-consuming. However, since we are mainly interested in the case $p \leq 2$, this can be significantly simplified. The following theorem provides an explicit basis for the space of $\partial$-invariant paths $\Omega_2$. Algorithm 1, which implements this construction, is provided in Appendix C. For the following result, we assume that the graph contains no motifs of the form illustrated in Fig. 1(c). We note that unlike the basis obtained by Grigor'yan et al. (2014, Proposition 2.9), our basis is orthonormal which allows for efficient numerical implementation.

**Theorem 1.** *Consider a directed graph $G$ with vector space of allowed 2-paths $\mathcal{A}_2$. Let $\mathcal{B}_{\mathcal{A}}^{(2)}$ denote the basis of $\mathcal{A}_2$ formed by all elementary allowed 2-paths. Then $\Omega_2$ admits an orthonormal basis consisting of the $\partial$-invariant elementary 2-paths $\sigma \in \mathcal{B}_{\mathcal{A}}^{(2)}$, together with vectors of the form*

$$\mathbf{h}_\ell = \frac{1}{\sqrt{\ell(\ell+1)}} \left( \sum_{q=1}^{\ell} \sigma_q - \ell\sigma_{\ell+1} \right), \tag{5}$$

*for $\ell = 1, \ldots, m-1$, where, for each non-allowed elementary 1-path $\tau$, $\sigma_1, \ldots, \sigma_m$ is a fixed ordering of all elementary allowed 2-paths satisfying $\partial_2^{\mathcal{D}}(\sigma_q) = -\tau$.*

Theorem 1 allows an efficient construction of the boundary matrix $\mathbf{B}_2$, hence of the path Laplacian.

### 3.3 EdgeSPE: Sign- and basis-invariant edge-level Path Laplacian PEs

In this section, we briefly review Stable and Expressive Positional Encodings (SPE), an effective encoding strategy introduced in Huang et al. (2024) to make spectral PEs sign- and basis-invariant. Standard SPE was developed for node-level encodings and therefore cannot be directly applied to the edge-level PathLapPE encodings corresponding to $\mathbf{L}_1$ without suitable modifications. We therefore develop a modified method, EdgeSPE, which extends SPE to edge-level encodings.

Letting $\mathbf{V} \in \mathbb{R}^{n \times d}$ be a matrix containing the first $d$ eigenvectors of a Laplacian matrix $\mathbf{L}$ and letting $\lambda \in \mathbb{R}^d$ denote a vector of the corresponding eigenvalues, the original SPE is defined by

$$\mathrm{SPE}(\mathbf{V}, \lambda) = \rho\big(\mathbf{V}\mathrm{diag}(\phi_1(\lambda))\mathbf{V}^\top, \ \ldots, \ \mathbf{V}\mathrm{diag}(\phi_m(\lambda))\mathbf{V}^\top\big), \tag{6}$$

where $\rho : \mathbb{R}^{n \times n \times m} \to \mathbb{R}^{n \times p}$ and each of the $\phi_\ell : \mathbb{R}^d \to \mathbb{R}^d$ are permutation equivariant functions. Intuitively, the $\phi_\ell$ act as a soft, learnable partition of the spectrum, in contrast to the hard partition into eigenspaces used by BasisNet (Lim et al., 2023). See Appendix D for further discussion.

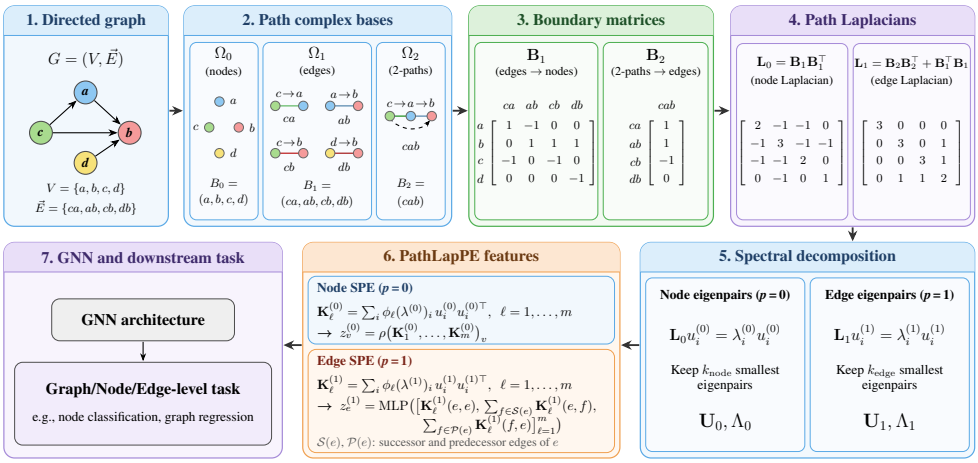



Figure 2: Overview of PathLapPE construction and feature extraction pipeline.

**EdgeSPE for PathLapPE.** To extend SPE to the edge-level setting, we let $\mathbf{U} \in \mathbb{R}^{|E| \times d}$ be a matrix containing the first $d$ eigenvectors of $\mathbf{L}_1$ and let $\lambda^{(1)} \in \mathbb{R}^d$ denote a vector of the corresponding eigenvalues. Analogously to the kernels $\mathbf{V}\mathrm{diag}(\phi_i(\lambda))\mathbf{V}^\top$, used in the original SPE (see Eq. (6)), we form edge-level SPE kernels $\mathbf{K}_i = \mathbf{U}\mathrm{diag}\big(\phi_i(\lambda^{(1)})\big)\mathbf{U}^\top \in \mathbb{R}^{|E| \times |E|}$, where $1 \leq i \leq m$. For each edge $e$, we let $\mathcal{S}(e) = \{f : \mathrm{tail}(f) = \mathrm{head}(e)\}$ and $\mathcal{P}(e) = \{f : \mathrm{head}(f) = \mathrm{tail}(e)\}$ denote its successors and its predecessors. Then for all $i$, we compute three encodings given by $\mathbf{h}_i^{\mathrm{diag}}(e) = \mathbf{K}_i(e,e)$, $\mathbf{h}_i^{\mathrm{succ}}(e) = \sum_{f \in \mathcal{S}(e)} \mathbf{K}_i(e,f)$, and $\mathbf{h}_i^{\mathrm{pred}}(e) = \sum_{f \in \mathcal{P}(e)} \mathbf{K}_i(f,e)$. Finally, we define the output of EdgeSPE by $\mathbf{EdgeSPE}(e) = \mathbf{MLP}\big(\mathbf{h}^{\mathrm{diag}}(e), \mathbf{h}^{\mathrm{succ}}(e), \mathbf{h}^{\mathrm{pred}}(e)\big)$.

## 4 Experiments

We evaluate PathLapPE as a positional encoding on a wide array of directed graph settings, including synthetic and real-world benchmarks, regression and classification targets, small and large-scale datasets, node- and graph-level tasks. We additionally analyze the computational cost of PathLapPE. All datasets are directed, and all relevant statistics are provided in Tables 4 and 5 (Appendix F). Additional details on training and model hyperparameters for all tasks are provided in Appendix G.

We compare PathLapPE with (i) no PEs, (ii) LapPE and (iii) RWPE (both Dwivedi et al. (2022)), and (iv) Multi-$q$-MagLapPE PEs (Huang et al., 2025), using a variety of both conventional and direction-aware GNNs.

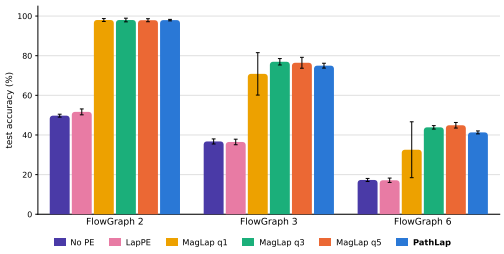



Figure 3: FlowGraph experiments, mean + std for five runs. Only directional PEs (PathLapPE and Multi-$q$-MagLapPE) perform above random.

In PathLapPE computation, we remove all self-loops, and pad zeros when $k > |V|$ or $|E|$. In addition to uni- and bidirectional GAT and GINE models, we evaluate with BiGCNE, a model we introduce in Appendix E as an extension of bidirectional GCN that supports edge features that act through a gating mechanism. Moreover, we process all node-level spectral positional encodings using the generalized SPE model introduced in Huang et al. (2025), which yields invariant and stable representations. Similarly, we process edge-level PathLapPE encodings with our EdgeSPE, and report results both with and without EdgeSPE.

### 4.1 Synthetic flow graphs

We begin with a simple task based on synthetic flow graphs designed to probe the directional information in PEs, primarily serving as a sanity check. FlowGraph datasets (Wang et al., 2024)

Table 1: Node classification accuracy (mean test accuracy %) on the WEBKB datasets. The best result within each model family is in bold and the second best is underlined.

| Model | PE | CORNELL | WISCONSIN | TEXAS | Avg. Rank ↓ |
|---|---|---|---|---|---|
| GAT | No PE | $48.37 \pm 7.59$ | $52.55 \pm 4.11$ | $59.73 \pm 4.12$ | 5.17 |
| | RWPE | $54.05 \pm 7.96$ | $53.72 \pm 6.55$ | $60.00 \pm 4.38$ | 3.17 |
| | LapPE | $54.05 \pm 7.75$ | $52.35 \pm 8.82$ | $59.73 \pm 4.84$ | 4.67 |
| | Multi-$q$-MagLapPE, $q \in \{0.01, 0.05, 0.10\}$ | $\underline{55.68 \pm 7.67}$ | $53.14 \pm 7.01$ | $\underline{60.27 \pm 4.24}$ | 2.67 |
| | PathLapPE (ours) | $53.78 \pm 6.17$ | $\mathbf{56.28 \pm 5.31}$ | $59.46 \pm 4.59$ | 4.00 |
| | PathLapPE (ours, w/ EdgeSPE) | $\mathbf{55.95 \pm 8.55}$ | $\underline{55.88 \pm 7.69}$ | $\mathbf{61.35 \pm 4.04}$ | **1.33** |
| BiGINE | No PE | $54.32 \pm 4.84$ | $58.43 \pm 3.43$ | $56.22 \pm 9.52$ | 5.83 |
| | RWPE | $55.41 \pm 8.66$ | $58.43 \pm 5.37$ | $61.35 \pm 6.25$ | 5.17 |
| | LapPE | $\mathbf{64.60 \pm 5.41}$ | $76.86 \pm 9.33$ | $64.60 \pm 9.57$ | 3.17 |
| | Multi-$q$-MagLapPE, $q \in \{0.01, 0.05, 0.10\}$ | $62.97 \pm 10.74$ | $\underline{79.02 \pm 6.34}$ | $69.19 \pm 7.56$ | 3.00 |
| | PathLapPE (ours) | $\underline{64.05 \pm 7.21}$ | $78.43 \pm 8.57$ | $\mathbf{71.62 \pm 8.76}$ | 2.33 |
| | PathLapPE (ours, w/ EdgeSPE) | $\mathbf{64.60 \pm 7.90}$ | $\mathbf{80.59 \pm 5.88}$ | $\underline{69.73 \pm 10.57}$ | **1.50** |
| BiGCNE | No PE | $60.27 \pm 5.98$ | $69.41 \pm 6.08$ | $65.14 \pm 8.40$ | 5.33 |
| | RWPE | $58.92 \pm 6.60$ | $65.88 \pm 6.80$ | $68.11 \pm 7.07$ | 5.67 |
| | LapPE | $76.75 \pm 4.45$ | $85.49 \pm 5.08$ | $81.89 \pm 4.04$ | 3.83 |
| | Multi-$q$-MagLapPE, $q \in \{0.01, 0.05, 0.10\}$ | $\underline{77.30 \pm 3.17}$ | $85.49 \pm 4.45$ | $\mathbf{83.24 \pm 7.07}$ | 2.33 |
| | PathLapPE (ours) | $77.03 \pm 5.87$ | $\underline{86.67 \pm 5.13}$ | $\mathbf{83.24 \pm 7.07}$ | 2.17 |
| | PathLapPE (ours, w/ EdgeSPE) | $\mathbf{78.65 \pm 4.50}$ | $\mathbf{87.26 \pm 3.84}$ | $\underline{82.16 \pm 4.80}$ | **1.67** |

are generated by creating clusters of nodes that are then sequentially ordered with a probabilistic directional flow defined between consecutive clusters: $n\%$ of the edges flow in one direction (e.g., from cluster $c$ to $c+1$), whereas $(100-n)\%$ flow in the opposite direction. Classes are defined by $n\%$ so that the task is directly tied to learning edge directionality. After computing positional encodings, we convert the graph to undirected (i.e., $A(u,v) \leftarrow \max\{A(u,v), A(v,u)\}$) and use a standard (i.e., unidirectional) GINE to force the network to rely only on the PE to solve the task.

**Results.** We evaluate on FLOWGRAPH $K \in \{2, 3, 6\}$, where $K$ denotes the number of classes defined in each task, and present the results in Figure 3. As expected, No PE and LapPE performance are essentially random, as the GINE layers have no directional information to work with. Multi-$q$-Laplacian PEs and PathLapPE can nearly fully solve the task for $K = 2$, and perform well above random for $K \in \{3, 6\}$, indicating that they capture directionality. We observe that a single $q$-parameter is more unstable than PathLapPE, which the Multi-$q$ variants alleviate.

### 4.2 WEBKB DATASETS

We next consider the WebKB datasets CORNELL, WISCONSIN, and TEXAS (Pei et al., 2020), in which nodes represent web pages and directed edges represent hyperlinks. Node features consist of bag-of-words representations. The task is to classify the nodes as student, project, course, staff, or faculty. We compare PathLapPE with and without EdgeSPE against LapPE, RWPE and Multi-$q$-MagLapPE. For the latter, we report the best results over $q \in \{0.01,\ 0.05,\ 0.10\}$. Additional information on tuning and selected hyperparameters is provided in Appendix G.2, Table 7.

**Results.** In Table 1, we report mean test accuracy over the ten splits from Pei et al. (2020). PathLapPE improves over the no PE baseline across all architectures and datasets and achieves the best average rank for each model. The gains are largest for BiGINE and BiGCNE, reaching 22.2 and 18.4 percentage points, respectively, compared with at most 7.6 points for GAT. The larger gains for BiGINE and BiGCNE may reflect their ability to incorporate edge features directly into message passing. EdgeSPE further improves performance in most settings, indicating that the direction-aware edge encodings provide complementary information beyond the node-level PathLapPE.

### 4.3 LARGE-SCALE EVALUATION ON ROMAN-EMPIRE

The WebKB datasets are relatively small and known to be noisy; thus, we next evaluate on a *directed* variant of ROMAN-EMPIRE (Platonov et al., 2023), a much larger dataset with 22,662 nodes, derived from the Roman Empire Wikipedia page. Each node corresponds to a word; an edge exists between

Table 2: Node classification results (mean test acc. %) on ROMAN-EMPIRE. Best result is **bolded**, second best is underlined. Best $k$ (num. eigenpairs) reported, Table 6 lists results for all $k$ values.

| PE | BiGINE | BiGINE-shared | GAT | BiGAT | BiGCNE |
|---|---|---|---|---|---|
| Base (No PE) | $89.48 \pm 0.40$ | $83.03 \pm 0.63$ | $83.01 \pm 0.35$ | $90.36 \pm 0.30$ | $87.79 \pm 0.47$ |
| LaplacianPE | $90.97 \pm 0.42$ | $87.68 \pm 0.42$ | $84.19 \pm 0.48$ | $\mathbf{91.15 \pm 0.40}$ | $89.58 \pm 0.57$ |
| Multi-$q$-MagLapPE, $q = 0.05$ | $90.95 \pm 0.36$ | $87.66 \pm 0.24$ | $83.93 \pm 0.56$ | $91.13 \pm 0.44$ | $89.63 \pm 0.36$ |
| Multi-$q$-MagLapPE, $\#q = 3, q \in \{0.05, 0.1, 0.15\}$ | $90.95 \pm 0.33$ | $87.28 \pm 0.39$ | $\underline{84.32 \pm 0.39}$ | $91.10 \pm 0.45$ | $89.82 \pm 0.35$ |
| PathLapPE (ours) | $\underline{91.33 \pm 0.23}$ | $\mathbf{89.50 \pm 0.36}$ | $\mathbf{84.39 \pm 0.39}$ | $91.10 \pm 0.46$ | $\mathbf{90.29 \pm 0.32}$ |
| PathLapPE (ours, w/ EdgeSPE) | $\mathbf{91.37 \pm 0.32}$ | $\underline{89.17 \pm 0.56}$ | $83.90 \pm 0.42$ | $\underline{91.14 \pm 0.27}$ | $\underline{90.06 \pm 0.45}$ |

two nodes if one follows the other in the text, or the words are syntactically connected in a sentence. Node features are based on word embeddings. The task is to predict the syntactic role of each word.

In addition to previous models, we evaluate bidirectional GAT and *BiGINE-shared*, a bidirectional GINE that shares the MLP weights of the GINE convolutions across the two directions. Weight sharing forces the model to rely more on the edge features, i.e., the spectral PEs, rather than on model expressivity to extract directional information, allowing us to evaluate the impact of individual PEs more distinctively. We evaluate two Multi-$q$-MagLapPE configurations: the best single $q$-value of $0.05$, and three stacked $q$ values. For model architectures and hyperparameter selection, we largely follow Platonov et al. (2023); full experimental details are provided in Appendix G.3, Table 7.

**Results.** Table 2 reports the performance of PathLapPE compared to No PE, LaplacianPE, and Multi-$q$-MagLapPE baselines on ROMAN-EMPIRE across five GNN architectures. PathLapPE achieves the best (or approximately equal-best, for BiGAT) result for every architecture. The gap between PathLapPE and the baselines is more pronounced for bidirectional GINE and GCNE, and is particularly large for our BiGINE-shared model, which is structurally restricted to rely on edge features for directionality. We also note that BiGINE-shared benefits more from PEs in general than the other architectures. Interestingly, both conventional GAT and its bidirectional variant benefit the least from (i) PEs in general, and (ii) PathLapPE specifically, with most PEs falling within one standard deviation of the PathLapPE results.

### 4.4 OPEN CIRCUIT BENCHMARK & HIGH-LEVEL SYNTHESIS

Finally, we consider inductive, graph-level regression tasks on the OPEN CIRCUIT BENCHMARK (OCB) and HIGH-LEVEL SYNTHESIS (HLS) datasets. For each task, we follow the implementations and optimized hyperparameters for BiGINE and GINE from Huang et al. (2025).

**OCB** (Dong et al., 2023) consists of 8,026 operational amplifier-based (op-amp) analog circuits as directed graphs, and the task is to predict DC gain (Gain), bandwidth (BW) and phase margin (PM) for each circuit. We evaluate two Multi-$q$-MagLapPE configurations, each with ten $q$ values but with phase parameters of 0.01 and 0.05.

**HLS** (Wu et al., 2022) represents 18,570 C/C++ programs as directed intermediate representation (IR) graphs for performance analysis on FPGA integrated circuits. The task is to predict digital signal processor (DSP) and look-up table (LUT) usage, and critical path timing slack (CP, permitted delay that does not affect overall program runtime) for each program. We evaluate two Multi-$q$-MagLapPE configurations, each with five $q$ values but with phase parameters of 0.01 and 0.1.

**Results.** Table 3 presents test results for the six tasks across both benchmarks, and the average rank of each PE. PathLapPE with EdgeSPE is the best method on average by some margin with average ranks of 1.50 and 1.83 for BiGINE and GINE respectively, as well as the most consistent: of the twelve model $\times$ task combinations evaluated, it is the best method on six, and top-2 on eleven of them – the sole remaining task being GINE $\times$ CP, where PEs are not helpful in general. However, PathLapPE *without* EdgeSPE (where we simply use a linear mapping of the raw edge-PEs) is relatively inconsistent, and falls behind not only the EdgeSPE variant but also other PEs on several tasks. The benefits of EdgeSPE are thus more apparent in such inductive settings where learning approximate sign- and basis-invariance is valuable, whereas in transductive node-level tasks one can precompute PEs on a single basis, as all training and inference happens over a single graph. Finally, we noticed that Multi-$q$-MagLapPE suffered from instabilities with GINE, where some runs

Table 3: Test set results for OPEN CIRCUIT BENCHMARK and HIGH-LEVEL SYNTHESIS. Mean ± std reported across five seeds. Best result is **bolded**, second best is underlined. Multi-$q$-MagLapPE uses $\#q = 10$ for OCB, and $\#q = 5$ for HLS.

| | | OCB (RMSE ↓) | | | HLS (MSE ↓) | | | Avg. |
|---|---|---|---|---|---|---|---|---|
| | PE | GAIN | BW | PM | DSP | LUT | CP | Rank ↓ |
| BIGINE | No PE | 0.394 ±0.007 | 4.497 ±0.111 | 1.123 ±0.011 | **2.254 ±0.068** | 1.793 ±0.174 | 1.152 ±0.077 | 4.08 |
| | LapPE | 0.394 ±0.014 | 4.223 ±0.077 | 1.114 ±0.022 | 2.445 ±0.216 | **1.767 ±0.030** | 1.158 ±0.132 | 3.75 |
| | Multi-$q$-MagLapPE, $\#q = \{5, 10\}, q = 0.01$ | 0.393 ±0.010 | 4.156 ±0.143 | 1.112 ±0.015 | 2.325 ±0.103 | 1.783 ±0.052 | 1.186 ±0.107 | 3.17 |
| | Multi-$q$-MagLapPE, $\#q = \{5, 10\}, q \in \{0.05, 0.1\}$ | 0.388 ±0.015 | 4.215 ±0.128 | 1.121 ±0.018 | 2.335 ±0.137 | 1.856 ±0.134 | 1.383 ±0.348 | 4.17 |
| | PathLapPE (ours) | 0.384 ±0.014 | 4.236 ±0.102 | 1.147 ±0.005 | 2.625 ±0.121 | 2.096 ±0.083 | **1.113 ±0.093** | 4.33 |
| | PathLapPE (ours, w/ EdgeSPE) | **0.367 ±0.005** | **4.036 ±0.092** | **1.102 ±0.026** | 2.322 ±0.083 | 1.771 ±0.131 | 1.118 ±0.069 | **1.50** |
| GINE | No PE | 0.415 ±0.007 | 4.817 ±0.110 | 1.126 ±0.019 | **2.662 ±0.215** | 2.009 ±0.092 | **0.654 ±0.034** | 2.92 |
| | LapPE | 0.411 ±0.020 | 4.349 ±0.076 | 1.127 ±0.030 | 2.738 ±0.279 | 2.031 ±0.139 | 0.680 ±0.025 | 3.33 |
| | Multi-$q$-MagLapPE, $\#q = \{5, 10\}, q = 0.01$ | 0.410 ±0.017 | 4.328 ±0.159 | **1.117 ±0.015** | 2.885 ±0.251 | 21.35 ±37.64 | 1.639 ±2.152 | 3.50 |
| | Multi-$q$-MagLapPE, $\#q = \{5, 10\}, q \in \{0.05, 0.1\}$ | 1.109 ±0.226 | 44.98 ±39.94 | 1.120 ±0.016 | 2.910 ±0.034 | 13.67 ±18.62 | 2.795 ±2.281 | 5.17 |
| | PathLapPE (ours) | 0.415 ±0.021 | 4.252 ±0.146 | 1.149 ±0.024 | 3.043 ±0.209 | 2.353 ±0.095 | 0.691 ±0.047 | 4.25 |
| | PathLapPE (ours, w/ EdgeSPE) | **0.379 ±0.009** | **4.226 ±0.128** | 1.119 ±0.019 | 2.717 ±0.137 | **1.946 ±0.061** | 0.697 ±0.046 | **1.83** |

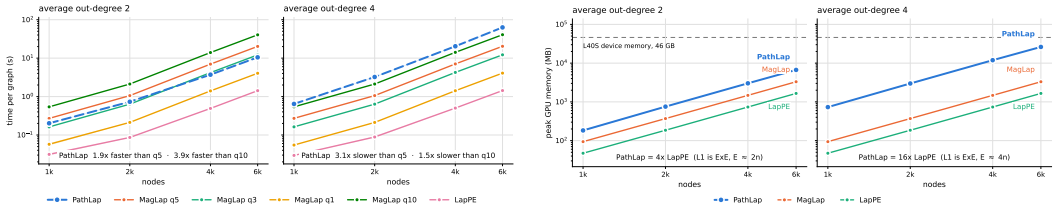



Figure 4: Time (left pair) and memory (right pair) cost of PE computation on directed ER graphs on an NVIDIA L40S GPU. Dense solver on GPU, $k = 8$, three repeats for each experiment.

converged to reasonable losses, while others fully diverged over the course of training. We provide a more extensive discussion in Appendix G.4.

### 4.5 COMPUTATIONAL COST OF PATHLAPPE

PathLapPE has different scaling properties than other spectral encodings: the $\mathbf{L}_0$ and $\mathbf{L}_1$ path Laplacians scale with $|V|$ (like LapPE) and $|E|$, respectively, so PathLapPE computation is more efficient on sparser graphs. In Figure 4, we grow Erdős–Rényi graphs at two fixed densities (average out-degree two and four) and measure PE computation time and GPU memory. On the sparser graphs, PathLapPE is faster than Multi-$q$-MagLapPE for $\#q \geq 3$ at scale. On the denser graphs, its cost is just above that of $\#q = 10$. PathLapPE uses more memory than Multi-$q$-MagLapPE, which processes $q$ values sequentially at fixed memory cost. For large graphs, sparse eigensolvers (Virtanen et al., 2020) can reduce this memory at a modest cost in time. Appendix H, Table 10 reports time and memory on our benchmarks; on the largest, ROMAN-EMPIRE (22,662 nodes), sparse PathLapPE takes three seconds and 70 MB, making PE computation costs negligible compared to training. Overall, the additional computational cost of PathLapPE relative to Multi-$q$-LapPE is modest, especially considering the additional benefit that PathLapPE requires no tuning.

## 5 CONCLUSION

We introduced PathLapPE, a spectral positional encoding for directed graphs derived from the Laplacian of the path complex. PathLapPE provides real-valued node- and edge-level encodings that capture edge orientation and directed 2-path structure, without the computational burden of higher-order message-passing. We additionally provide EdgeSPE, which further improves the performance of edge-level encodings. Across a wide range of node- and graph-level directed benchmarks, PathLapPE performs as well as or better than other spectral PEs, with the largest gains for direction-aware architectures. Unlike magnetic Laplacian encodings, it has no phase parameter to tune and requires only one real eigendecomposition per Laplacian, making it computationally efficient. Extending our PE to path Laplacians of higher order and reducing memory cost on denser graphs, as well as the inclusion into graph Transformers, are natural directions for future work.

### AI USE STATEMENT

Generative AI usage in this work has been strictly limited to light text editing, adjustments of provided figures (e.g., TikZ adjustments for Fig. 2), and writing minor boilerplate code. Generative AI tools have *not* been used for the development of the mathematical framework nor the accompanying formulation or proving of mathematical claims, experiment design, dataset generation, data analysis nor core development of the codebase. We have, of course, reviewed all AI-assisted work. LLM adjustments to any text and figures have been checked by multiple authors; any LLM-generated code adjustments have also been verified and tested for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REFERENCES

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, April 2024. doi: 10.1145/3620665.3640366. URL https://docs.pytorch.org/assets/pytorch2-2.pdf.

Semih Cantürk, Renming Liu, Olivier Lapointe-Gagné, Vincent Létourneau, Guy Wolf, Dominique Beaini, and Ladislav Rampášek. Graph positional and structural encoder. In *International Conference on Machine Learning*, 2024.

Zehao Dong, Weidong Cao, Muhan Zhang, Dacheng Tao, Yixin Chen, and Xuan Zhang. CktGNN: Circuit graph neural network for electronic design automation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=NE2911Kq1sp.

Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=wTTjnvGphYj.

William Falcon and The PyTorch Lightning team. PyTorch Lightning, March 2019. URL https://github.com/Lightning-AI/lightning.

Matthias Fey, Jinu Sunil, Akihiro Nitta, Rishi Puri, Manan Shah, Blaz Stojanovic, Ramona Bendias, Alexandria Barghi, Vid Kocijan, Zecheng Zhang, Xinwei He, Jan Eric Lenssen, and Jure Leskovec. PyG 2.0: Scalable Learning on Real World Graphs. *Temporal Graph Learning Workshop @ KDD*, 2025.

Simon Geisler, Yujia Li, Daniel J Mankowitz, Ali Taylan Cemgil, Stephan Günnemann, and Cosmin Paduraru. Transformers meet directed graphs. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 11144–11172. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/geisler23a.html.

Alexander Grigor'yan. Advances in path homology theory of digraphs. *Notices of the International Consortium of Chinese Mathematicians*, 10(2):61–124, 2022.

Alexander Grigor'yan, Yong Lin, Yuri Muranov, and Shing-Tung Yau. Homotopy theory for digraphs. *Pure and Applied Mathematics Quarterly*, 10(4):619–674, 2014.

Alexander A Grigor'yan, Yong Lin, Yu V Muranov, and Shing-Tung Yau. Path complexes and their homologies. *Journal of Mathematical Sciences*, 248(5):564–599, 2020.

Yinan Huang, William Lu, Joshua Robinson, Yu Yang, Muhan Zhang, Stefanie Jegelka, and Pan Li. On the stability of expressive positional encodings for graphs. In *International Conference on Learning Representations*, 2024.

Yinan Huang, Haoyu Wang, and Pan Li. What are good positional encodings for directed graphs? In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=s4Wm71LFK4.

Guillaume Jaume, An phi Nguyen, María Rodríguez Martínez, Jean-Philippe Thiran, and Maria Gabrani. edGNN: a simple and powerful GNN for directed labeled graphs, 2019. URL https://arxiv.org/abs/1904.08745.

Niklas Kemper, Tom Wollschläger, and Stephan Günnemann. What expressivity theory misses: Message passing complexity for GNNs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=ATYKgiDqt5.

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

Derek Lim, Joshua David Robinson, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. Sign and basis invariant networks for spectral graph representation learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Q-UHqMorzil.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/bb04af0f7ecaee4aae62035497da1387-Paper.pdf.

Federico Monti, Karl Otness, and Michael M Bronstein. MotifNet: a motif-based graph convolutional network for directed graphs. In *2018 IEEE data science workshop (DSW)*, pp. 225–228. IEEE, 2018.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-GCN: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1e2agrFvS.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tJbbQfw-5wv.

Emanuele Rossi, Bertrand Charpentier, Francesco Di Giovanni, Fabrizio Frasca, Stephan Günnemann, and Michael M. Bronstein. Edge directionality improves learning on heterophilic graphs. In Soledad Villar and Benjamin Chamberlain (eds.), *Proceedings of the Second Learning on Graphs Conference*, volume 231 of *Proceedings of Machine Learning Research*, pp. 25:1–25:27. PMLR, 27–30 Nov 2024. URL https://proceedings.mlr.press/v231/rossi24a.html.

Quang Truong and Peter Chin. Weisfeiler and lehman go paths: Learning topological features via path complexes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 15382–15391, 2024.

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

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Qitong Wang, Georgios Kollias, Vasileios Kalantzis, Naoki Abe, and Mohammed J Zaki. Directed graph transformers. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=otTFPjziiK.

Nan Wu, Hang Yang, Yuan Xie, Pan Li, and Cong Hao. High-level synthesis performance prediction using GNNs: benchmarking, modeling, and advancing. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, DAC '22, pp. 49–54, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450391429. doi: 10.1145/3489517.3530408. URL https://doi.org/10.1145/3489517.3530408.

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

Omry Yadan. Hydra - a framework for elegantly configuring complex applications. Github, 2019. URL https://github.com/facebookresearch/hydra.

Xitong Zhang, Yixuan He, Nathan Brugnone, Michael Perlmutter, and Matthew Hirn. MagNet: A neural network for directed graphs. In *Advances in Neural Information Processing Systems*, volume 34, pp. 27003–27015, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/e32084632d369461572832e6582aac36-Abstract.html.

## A LIMITATIONS

In this work, we have considered path Laplacians only up to order $p = 2$; higher-order constructions are possible, albeit at increasing computational costs. In conjunction with this point, PathLapPE construction and eigendecomposition is inherently more memory-intensive on dense graphs due to its scaling with respect to the number of edges. While efficient workarounds exist particularly for eigendecomposition, extending to even higher-order paths would exacerbate these memory limitations. Our evaluation also largely focuses on uni- and bidirectional GNNs; while we expect the observed performance gains to naturally extend to graph Transformers, further evaluations on Transformer-based architectures would solidify this claim. Finally, in our research we have observed that there is limited work on whether directionality is truly beneficial to a given task on directed benchmarks; future work on investigating types of directed graphs that would benefit from PathLapPE would increase the potential impact of our method.

## B THE MAGNETIC LAPLACIAN

For undirected graphs, the standard graph Laplacian provides a canonical self-adjoint operator whose eigenvectors encode global graph structure. In the directed setting, however, there is no unique analogue of the graph Laplacian. Different constructions preserve different aspects of directionality. In this section, we briefly describe magnetic Laplacians, which encode directionality through complex phases.

Let $G = (V, \vec{E})$ be a directed graph with adjacency matrix $\mathbf{A}$, where $\mathbf{A}_{uv} = 1$ if $(u, v) \in \vec{E}$ and $\mathbf{A}_{uv} = 0$ otherwise. Magnetic Laplacian constructions first define a symmetrized adjacency matrix $\mathbf{A}_s = \frac{1}{2}(\mathbf{A} + \mathbf{A}^\top)$, which records the undirected support of the graph, possibly with weights reflecting whether one or both orientations are present. Directionality is then incorporated through a skew-symmetric phase matrix $\Theta^{(q)} = 2\pi q(\mathbf{A} - \mathbf{A}^\top)$, where $q$ is a tunable phase parameter controlling the strength of the directional signal. The corresponding complex Hermitian adjacency matrix is given by

$$\mathbf{H}^{(q)} = \mathbf{A}_s \odot \exp\left(i\Theta^{(q)}\right),$$

where $\odot$ denotes entrywise multiplication. The degree matrix associated with $\mathbf{A}_s$ is given by $\mathbf{D}_s(u, u) = \sum_{v \in V} \mathbf{A}_s(u, v)$. The unnormalized magnetic Laplacian is then defined by

$$\mathbf{L}^{(q)}_{\text{mag}} = \mathbf{D}_s - \mathbf{H}^{(q)}.$$

The normalized version

$$\mathcal{L}^{(q)}_{\text{mag}} = \mathbf{I} - \mathbf{D}_s^{-1/2}\mathbf{H}^{(q)}\mathbf{D}_s^{-1/2}$$

is often used in graph learning.

## C PROOF OF THEOREM 1

In this section, we provide a proof of Theorem 1. We recall that any directed graph $G = (V, \vec{E})$ gives rise to a vector space $\Lambda_p$ spanned by elementary $p$-paths, a vector space $\mathcal{A}_p \subseteq \Lambda_p$ spanned by allowed elementary $p$-paths and finally the main object of interest in this section, the vector space $\Omega_p \subseteq \mathcal{A}_p$ of $\partial_p$-invariant allowed paths. In this section, $p = 1, 2$, and we are interested in constructing a basis for $\Omega_2$ in closed form. Before proving Theorem 1, we first consider the following example, in which we compute a basis for $\Omega_2$ in a simple case.

**Example 3.** *As in Example 1, consider the directed triangle represented in Figure 1(a) and the elementary paths $u = ab, v = bc$, $w = ac$, and $t = abc$. Then, an ordered basis for $\Omega_0$ is given by the vertices $a, b$, and $c$ and an ordered basis for $\Omega_1$ is given by the 1-paths corresponding to the directed edges $u$, $v$, and $w$. Since $\partial t = bc - ac + ab = v - w + u$, we conclude that $t$ is a $\partial$-invariant allowed path, hence an element of $\Omega_2$. One may also verify that there are no other independent elements of $\Omega_2$ and so $\Omega_2$ is a one-dimensional vector space with basis $\{t\}$. Given these choices of bases (in the order listed above), the boundary matrices are given by*

$$\mathbf{B}_1 = \begin{bmatrix} -1 & 0 & -1 \\ 1 & -1 & 0 \\ 0 & 1 & 1 \end{bmatrix}; \quad \mathbf{B}_2 = \begin{bmatrix} 1 & 1 & -1 \end{bmatrix}^\top.$$

*The unweighted edge Laplacian is therefore* $\mathbf{L}_1 = \mathbf{B}_2\mathbf{B}_2^\top + \mathbf{B}_1^\top\mathbf{B}_1 = 3\mathbf{I}_3$.

We recall from Section 3.2 that $\Lambda_1$ can be expressed as the direct sum $\mathcal{A}_1 \oplus \mathcal{D}_1$ and the corresponding projection maps $\pi_{\mathcal{A}}^{(1)} : \Lambda_1 \to \mathcal{A}_1$, and $\pi_{\mathcal{D}}^{(1)} : \Lambda_1 \to \mathcal{D}_1$ induce the maps

$$\partial_2^{\mathcal{A}} : \mathcal{A}_2 \to \mathcal{A}_1; \quad \partial_2^{\mathcal{A}}(\sigma) \mapsto \pi_{\mathcal{A}}^{(1)} \circ \partial_2(\sigma),$$

and

$$\partial_2^{\mathcal{D}} : \mathcal{A}_2 \to \mathcal{D}_1; \quad \partial_2^{\mathcal{D}}(\sigma) \mapsto \pi_{\mathcal{D}}^{(1)} \circ \partial_2(\sigma). \tag{7}$$

Here $\partial_2 : \Lambda_2 \to \Lambda_1$ is the usual boundary map on $p$-paths. Note that, by definition, $\Omega_2$ is equal to $\ker \partial_2^{\mathcal{D}}$ since $\sigma$ is in $\Omega_2$ if and only if $\partial_2(\sigma) \in \mathcal{A}_1$ and $\mathcal{D}_1$ is the orthogonal complement of $\mathcal{A}_1$.

We consider the basis of $\Lambda_p$ consisting of elementary $p$-paths, and fix an ordering for the basis, denoted by:

$$\mathcal{B}_\Lambda^{(p)} := \{\sigma_i^{(p)} |\ i = 1, \ldots, \dim(\Lambda_p)\}$$

The basis of $\mathcal{A}_p$ is the subset $\mathcal{B}_{\mathcal{A}}^{(p)} \subseteq \mathcal{B}_\Lambda^{(p)}$ corresponding to the allowed paths. We similarly denote by $\mathcal{B}_{\mathcal{D}}^{(1)} \subseteq \mathcal{B}_\Lambda^{(1)}$ the basis of $\mathcal{D}_1$ formed by the non-allowed elementary 1-paths. Finally, we denote by $\mathbf{B}_2^{\mathcal{D}}$ the matrix representation of the map $\partial_2^{\mathcal{D}}$ defined in Equation (7), expressed with respect to the bases $\mathcal{B}_{\mathcal{A}}^{(2)}$ and $\mathcal{B}_{\mathcal{D}}^{(1)}$.

*Proof.* Consider an arbitrary elementary allowed 2-path $\sigma_j^{(2)} = v_0v_1v_2$. By definition,

$$\partial_2(\sigma_j^{(2)}) = v_1v_2 - v_0v_2 + v_0v_1. \tag{8}$$

Since $\sigma_j^{(2)}$ is an allowed path, the edges $v_0v_1$ and $v_1v_2$ are allowed 1-paths (i.e., directed edges in $G$). Therefore,

$$\partial_2^{\mathcal{D}}(\sigma_j^{(2)}) = -\pi_{\mathcal{D}}^{(1)}(v_0v_2).$$

Hence, $\sigma_j^{(2)} \in \Omega_2$ if and only if $v_0v_2 \in \mathcal{A}_1$. Thus, if $v_0v_2$ is an allowed 1-path, then $\mathbf{B}_2^{\mathcal{D}}[:, j] = 0$ and so $\sigma_j^{(2)}$ is $\partial$-invariant and an element of $\Omega_2$. Otherwise, let $\sigma_i^{(1)} = v_0v_2 \in \mathcal{B}_{\mathcal{D}}^{(1)}$. Then, $\mathbf{B}_2^{\mathcal{D}}[i, j] = -1$. Assume there exist exactly $m$ elementary allowed 2-paths $\sigma_{j_1}^{(2)}, \ldots, \sigma_{j_m}^{(2)}$ satisfying

$$\partial_2^{\mathcal{D}}\left(\sigma_{j_q}^{(2)}\right) = -\sigma_i^{(1)}, \qquad q = 1, \ldots, m.$$

Then, the restriction of $\mathbf{B}_2^{\mathcal{D}}[i, :]$ to these basis vectors corresponds to the constant row vector of length $m$ given by

$$-\mathbf{1}^\top = (-1, \ldots, -1).$$

Since $\Omega_2 \simeq \ker \mathbf{B}_2^{\mathcal{D}}$, it remains to construct a basis for the kernel of $-\mathbf{1}^\top$. If $m = 1$, this kernel is trivial. For $m > 1$, consider the column vectors in $\mathbb{R}^m$ given by

$$\frac{1}{\sqrt{\ell(\ell+1)}}\left(\underbrace{1, \ldots, 1}_{\ell}, -\ell, 0, \ldots, 0\right)^\top, \qquad 1 \le \ell < m. \tag{9}$$

It is straightforward to verify that these vectors constitute an orthonormal basis for $\ker(-\mathbf{1}^\top)$. Writing them with respect to the basis elements $\sigma_{j_1}^{(2)}, \ldots, \sigma_{j_m}^{(2)}$ gives the 2-paths

$$\mathbf{h}_\ell = \frac{1}{\sqrt{\ell(\ell+1)}}\left(\sum_{q=1}^{\ell} \sigma_{j_q}^{(2)} - \ell\sigma_{j_{\ell+1}}^{(2)}\right), \qquad \ell = 1, \ldots, m-1.$$

Finally, every column of $\mathbf{B}_2^{\mathcal{D}}$ is either zero or contains exactly one nonzero entry. The nonzero columns can therefore be partitioned according to the non-allowed elementary 1-path appearing in their boundary. Combining the $\partial$-invariant elementary 2-paths corresponding to the zero columns with the vectors $\mathbf{h}_\ell$ constructed for each such group gives a basis for $\ker \mathbf{B}_2^{\mathcal{D}}$, and therefore for $\Omega_2$. □

**Algorithm 1** Construction of a basis for $\Omega_2$

**Input:** Directed graph $G = (V, \vec{E})$
Construct the set $\mathcal{A}_2$ of allowed 2-paths and $\mathbf{B}_2^{\mathcal{D}}$, the matrix representation of $\partial_2^{\mathcal{D}}$.
Initialize $\mathcal{B}_{\Omega_2} \leftarrow \varnothing$ and $\mathcal{R} \leftarrow \varnothing$.
**for** $i = 1, \ldots, |\mathcal{A}_2|$ **do**
  **if** $\mathbf{B}_2^{\mathcal{D}}[:, i] = 0$ **then**
    Add $e_i$ to $\mathcal{B}_{\Omega_2}$.
  **else**
    Let $r$ be the unique row such that $(\mathbf{B}_2^{\mathcal{D}})_{ri} \neq 0$.
    **if** $r \notin \mathcal{R}$ **then**
      Let

$$J_r = \{j_1, \ldots, j_s\} := \{j : (\mathbf{B}_2^{\mathcal{D}})_{rj} \neq 0\}.$$

      **for** $\ell = 1, \ldots, s-1$ **do**
        Add to $\mathcal{B}_{\Omega_2}$

$$h_\ell = \frac{1}{\sqrt{\ell(\ell+1)}} \left( \sum_{q=1}^{\ell} e_{j_q} - \ell e_{j_{\ell+1}} \right)$$

      **end for**
      $\mathcal{R} \leftarrow \mathcal{R} \cup \{r\}$.
    **end if**
  **end if**
**end for**
**return** $\mathcal{B}_{\Omega_2}$.

## D Further Discussion on SPE

As shown in Huang et al. (2024), SPE is invariant to the choice of eigenbasis and in particular is invariant to sign flips. We briefly review why that is the case.

Let $\mu_1 < \ldots < \mu_r$ denote the distinct eigenvalues among $\lambda_1, \ldots, \lambda_d$. For each $1 \leq j \leq r$, we let $I_j = \{\, i : \lambda_i = \mu_j \,\}$ and let $\mathbf{V}_j \in \mathbb{R}^{n \times |I_j|}$ denote the submatrix of $\mathbf{V}$ whose columns are the corresponding eigenvectors (i.e., the $\mathbf{V}_i$, $i \in I_j$). Writing the kernel $\mathbf{Q}_\ell$ as a sum of rank-one terms yields

$$\mathbf{Q}_\ell = \mathbf{V} \operatorname{diag}(\phi_\ell(\lambda)) \mathbf{V}^\top = \sum_{i=1}^{d} \phi_\ell(\lambda)_i \mathbf{V}_i \mathbf{V}_i^\top.$$

Let $i$ and $i'$ be two indices with $\lambda_i = \lambda_{i'}$, and let $\pi$ be the transposition $\pi$ of $i$ and $i'$. By construction, $\pi\lambda = \lambda$, so permutation equivariance gives

$$\phi_\ell(\lambda) = \phi_\ell(\pi\lambda) = \pi\, \phi_\ell(\lambda),$$

i.e., $\phi_\ell(\lambda)_i = \phi_\ell(\lambda)_{i'}$. Therefore, letting $w_{\ell j}$ denote the unique value of $\phi_\ell(\lambda)_i$ for $i \in I_j$, we may write

$$\mathbf{Q}_\ell = \sum_{j=1}^{r} w_{\ell j} \mathbf{V}_j \mathbf{V}_j^\top. \tag{10}$$

Each of the terms $\mathbf{V}_j \mathbf{V}_j^\top$ is simply a projection matrix onto the eigenspace of $\mu_j$. Projection does not depend on the choice of eigenbasis and so $\mathbf{Q}_\ell$ is invariant to the choice of basis (in particular it is invariant to sign flips). Therefore SPE is invariant as well.

Huang et al. (2024) also show, in Theorem 3.1, that if $\phi_\ell$ and $\rho$ are Lipschitz, then SPE is stable with respect to the input Laplacian $\mathbf{L}$ (with a Lipschitz constant that depends on the eigengap $\gamma = \lambda_{d+1} - \lambda_d$). Furthermore, in Proposition 3.2, it is shown that SPE can universally approximate any continuous basis invariant function. Finally, they also show that SPE makes effective use of the eigenvalues (in addition to the eigenvectors) and can count cycles in Propositions 3.3 and 3.4.

# E BiGCNE: Directed Gated GCN with Node and Edge Positional Encodings

PathLapPE provides positional information at both the node and edge levels. To leverage our direction-aware edge PE while maintaining a fair comparison with other node-level PEs, we use an edge-gated extension of bidirectional message passing, which we denote by **BiGCNE**. Following the Dir-GNN framework (Rossi et al., 2024), incoming information and outgoing information are aggregated separately. In addition, the contribution of each neighboring node is modulated by a learned gate that depends on the positional encoding of the corresponding directed edge.

Let $\mathbf{x}_i$ denote the input features of node $i$, let $\mathbf{p}_i^{\text{node}}$ denote its node-level positional encoding, and let $\mathbf{p}_{ij}^{\text{edge}}$ denote the edge-level positional encoding associated with the directed edge $(i,j) \in \vec{E}$. The node-level positional encoding is incorporated into the initial node representation by concatenation, that is, $\tilde{\mathbf{x}}_i = \left[\mathbf{x}_i \,\|\, \mathbf{p}_i^{\text{node}}\right]$. Moreover, we let $\mathbf{h}_i^{(0)}$ denote the resulting initial hidden representation after the input embedding. At each BiGCNE layer $\ell$, messages from incoming and outgoing neighbors are computed separately. For an incoming edge $(j,i) \in \vec{E}$, the corresponding gate is given by

$$\gamma_{ji}^{\text{in},(\ell)} = \sigma\Big(\big(\mathbf{w}_{g,\text{in}}^{(\ell)}\big)^\top \mathbf{p}_{ji}^{\text{edge}} + \mathbf{b}_{g,\text{in}}^{(\ell)}\Big),$$

where $\mathbf{w}_{g,\text{in}}^{(\ell)}$ and $\mathbf{b}_{g,\text{in}}^{(\ell)}$ are learnable weights and biases, respectively. The incoming messages are then obtained by

$$\mathbf{m}_i^{\text{in},(\ell)} = \text{AGG}_{(j,i)\in\vec{E}}\left(\gamma_{ji}^{\text{in},(\ell)} \mathbf{W}_{\text{in}}^{(\ell)} \mathbf{h}_j^{(\ell)}\right).$$

Similarly, for an outgoing edge $(i,j) \in \vec{E}$, we define the gate

$$\gamma_{ij}^{\text{out},(\ell)} = \sigma\Big(\big(\mathbf{w}_{g,\text{out}}^{(\ell)}\big)^\top \mathbf{p}_{ij}^{\text{edge}} + \mathbf{b}_{g,\text{out}}^{(\ell)}\Big),$$

and the outgoing messages by

$$\mathbf{m}_i^{\text{out},(\ell)} = \text{AGG}_{(i,j)\in\vec{E}}\left(\gamma_{ij}^{\text{out},(\ell)} \mathbf{W}_{\text{out}}^{(\ell)} \mathbf{h}_j^{(\ell)}\right).$$

Here, $\sigma$ denotes the sigmoid function. We use scalar edge gates, so that each $\gamma_{ij}^{\text{in},(\ell)}$ or $\gamma_{ij}^{\text{out},(\ell)}$ controls the overall contribution of the corresponding message. The incoming and outgoing branches have independent learnable parameters. In our implementation, AGG is mean aggregation. The current node representation and the two directional aggregates are then concatenated and combined through a learnable transformation:

$$\mathbf{h}_i^{(\ell+1)} = \text{Dropout}\left(\text{ReLU}\left(\mathbf{W}_{\text{com}}^{(\ell)}\left[\mathbf{h}_i^{(\ell)} \,\|\, \mathbf{m}_i^{\text{in},(\ell)} \,\|\, \mathbf{m}_i^{\text{out},(\ell)}\right] + \mathbf{b}_{\text{com}}^{(\ell)}\right)\right).$$

When the input and output dimensions agree, a residual connection is added (i.e. $\mathbf{h}_i^{(\ell+1)} = \mathbf{h}_i^{(\ell+1)} + \mathbf{h}_i^{(\ell)}$). Unlike the classical Dir-GNN, which combines incoming and outgoing messages using a predefined parameter $\alpha$, we use the edge-level positional encoding to modulate message passing through the learned gates at each layer. This direction-aware architecture preserves structural and directionality information captured through the PE that would otherwise be diluted by traditional architectures such as GCN. A directed gated encoder modulates information flow rather than simply appending the aggregated edge PEs to the node features.

# F Dataset statistics

In Tables 4 and 5, we report basic statistics for the benchmark datasets considered in this paper for node-level and graph-level tasks, respectively.

# G Experimental details & Hyperparameters

## G.1 Synthetic flow graphs

FlowGraph datasets (Wang et al., 2024) are generated by creating clusters of nodes with random interconnections that are then sequentially ordered with a probabilistic directional flow defined between consecutive clusters: $n\%$ of the edges flow in one direction (e.g., from cluster $c$ to $c+1$),

Table 4: Dataset statistics for transductive node-level tasks

| **Dataset** | CORNELL | WISCONSIN | TEXAS | ROMAN-EMPIRE |
|---|---|---|---|---|
| # Nodes | 183 | 251 | 183 | 22,662 |
| # Edges | 295 | 499 | 325 | 44,363 |
| # Features | 1,703 | 1,703 | 1,703 | 300 |
| # Classes | 5 | 5 | 5 | 18 |
| License | CC-BY 4.0 | CC-BY 4.0 | CC-BY 4.0 | MIT |

Table 5: Dataset statistics for inductive graph-level tasks

| **Dataset** | FLOWGRAPH2 | FLOWGRAPH3 | FLOWGRAPH6 | OCB | HLS |
|---|---|---|---|---|---|
| # Graphs | 2,000 | 3,000 | 6,000 | 8,026[a] | 18,570 |
| Avg. # Nodes | 114.46 | 111.66 | 111.40 | 9.89 | 94.70 |
| Avg. # Edges | 150.30 | 150.70 | 149.93 | 15.02 | 122.12 |
| # Node Features | 0 | 0 | 0 | 3 | 7 |
| # Edge Features | 0 | 0 | 0 | 0 | 2 |
| Task | Binary class. | 3-way class. | 6-way class. | Regression | Regression |
| License | CC BY 4.0 | CC BY 4.0 | CC BY 4.0 | CC BY 4.0 | CC BY 4.0 |

[a] The full dataset consists of 10,000 graphs, but we focus on three-stage amplifier graphs only – two-stage amplifiers are presented as a separate OOD generalization task. While Huang et al. (2025) denote that they report two-stage results, their codebase and results are consistent with three-stage amplifiers instead.

whereas $(100 - n)\%$ flow in the opposite direction. Classes are defined by $n\%$ so that the task is directly tied to learning edge directionality, with each class assigned 1,000 samples. After computing positional encodings, we convert the graph to undirected and use a vanilla (i.e., unidirectional) GINE to force the network to rely only on the PE to solve the task. Hyperparameters for FLOWGRAPH are provided in Table 7.

### G.2 WEBKB DATASETS

On all WEBKB datasets, the number of message passing layers is set to two, and a grid search is performed over the following parameter values: dropout $\in \{0.0,\ 0.5\}$, number of eigenvalues $k_{\text{node}}, k_{\text{edge}} \in \{16,\ 32\}$, and $q \in \{0.01,\ 0.05,\ 0.10\}$ for Multi-$q$-MagLapPE. Note that we do not stack the magnetic Laplacian encodings for several values of $q$ here; the tested values of $q$ and $\alpha$ were selected based on the best results reported in Zhang et al. (2021) and Rossi et al. (2024). Hyperparameters for all WEBKB datasets are provided in Table 7.

### G.3 ROMAN-EMPIRE

ROMAN-EMPIRE is derived from the Wikipedia entry of the same name, where each node in the graph corresponds to a word in the text, not necessarily unique. An edge exists between two nodes if (i) one follows the other in the text, or (ii) the words are syntactically connected in a sentence. Node features are based on word embeddings, and the task is to predict the syntactic role of each word. Despite the semantic relationships in the graph implying directed edges, Platonov et al. (2023) provide an undirected graph. Rossi et al. (2024) then provide a directed version using the same general construction but processing the graph to account for directionality; we evaluate our method on this directed version.

We evaluate two Multi-$q$-MagLapPE configurations: the best single $q$-value of $0.05$, after searching over $q \in \{0.05, 0.1, 0.15\}$, as well as a version that uses all three $q$ values, following Huang et al. (2025) – stacking PEs derived from multiple $q$ values typically outperforms any single $q$-value. For all PEs, we search over $k \in \{8, 16, 32, 64\}$, i.e., the number of nontrivial eigenvectors/eigenvalues, and report the best results. Table 6 provides the full results for all $k$ values.

Finally, since PathLapPE provides *two* sets of encodings, node- and edge-level, we evaluate a more parameter-restricted version that maps each set of encodings to *half* of the node/edge hidden di-

mension of 512 before concatenation for a fairer comparison with the baselines; we denote this variant PathLapPE ($256 \times 2$). For model architectures and hyperparameter selection, we largely follow Platonov et al. (2023); full experimental details and selected hyperparameters are provided in Table 7.

**Complete results.** Here, we present the full set of results summarized in Table 2 of the main text. Specifically, we report results for all tested $k$ values; the final column (**Best**) corresponds to Table 2. The additional results demonstrate that (i) the superiority of PathLapPE is largely independent of the number of eigenvectors/eigenvalues considered, and that (ii) in almost all cases, the PEs are robust to the choice of $k$, meaning that 8 or 16 is likely sufficient for ROMAN-EMPIRE. Finally, restricting the dimensionality of PathLapPE encodings has no tangible drawback, as the two variants are always close and well within one standard deviation.

Table 6: Results across encoder architectures and positional encodings on ROMAN-EMPIRE. Bold indicates the best $k$ for each variant. In the last column, the top three PEs for each encoder are denoted with **gold**, **silver** and **bronze**.

| Encoder | PE Variant | $k$=8 | $k$=16 | $k$=32 | $k$=64 | Best |
|---|---|---|---|---|---|---|
| BiGINE-shared | Base (No PE) | $83.03 \pm 0.63$ | $83.03 \pm 0.63$ | $83.03 \pm 0.63$ | $83.03 \pm 0.63$ | $83.03 \pm 0.63$ |
| | LaplacianPE | $\mathbf{87.68 \pm 0.42}$ | $87.44 \pm 0.23$ | $87.60 \pm 0.31$ | $87.53 \pm 0.42$ | $87.68 \pm 0.42$ |
| | Multi-$q$-MagLapPE, $q = 0.05$ | $\mathbf{87.66 \pm 0.24}$ | $87.47 \pm 0.36$ | $87.52 \pm 0.42$ | $87.64 \pm 0.40$ | $87.66 \pm 0.24$ |
| | Multi-$q$-MagLapPE, $q \in \{0.05, 0.1, 0.15\}$ | $87.10 \pm 0.68$ | $\mathbf{87.28 \pm 0.39}$ | $87.17 \pm 0.52$ | $87.16 \pm 0.53$ | $87.28 \pm 0.39$ |
| | PathLapPE ($256 \times 2$) | $89.31 \pm 0.44$ | $\mathbf{89.43 \pm 0.42}$ | $89.36 \pm 0.41$ | $89.25 \pm 0.37$ | **$89.43 \pm 0.42$** |
| | PathLapPE | $\mathbf{89.50 \pm 0.36}$ | $89.15 \pm 0.29$ | $89.12 \pm 0.41$ | $89.25 \pm 0.41$ | **$89.50 \pm 0.36$** |
| | PathLapPE (EdgeSPE) | $89.11 \pm 0.48$ | $\mathbf{89.17 \pm 0.56}$ | $89.01 \pm 0.57$ | $89.05 \pm 0.45$ | **$89.17 \pm 0.56$** |
| BiGINE | Base (No PE) | $89.48 \pm 0.40$ | $89.48 \pm 0.40$ | $89.48 \pm 0.40$ | $89.48 \pm 0.40$ | $89.48 \pm 0.40$ |
| | LaplacianPE | $90.93 \pm 0.44$ | $\mathbf{90.97 \pm 0.42}$ | $90.84 \pm 0.38$ | $90.89 \pm 0.31$ | $90.97 \pm 0.42$ |
| | Multi-$q$-MagLapPE, $q = 0.05$ | $90.93 \pm 0.34$ | $90.90 \pm 0.30$ | $\mathbf{90.95 \pm 0.36}$ | $90.86 \pm 0.32$ | $90.95 \pm 0.36$ |
| | Multi-$q$-MagLapPE, $q \in \{0.05, 0.1, 0.15\}$ | $90.86 \pm 0.23$ | $\mathbf{90.95 \pm 0.33}$ | $90.94 \pm 0.37$ | $90.91 \pm 0.30$ | $90.95 \pm 0.33$ |
| | PathLapPE ($256 \times 2$) | $91.28 \pm 0.30$ | $91.30 \pm 0.31$ | $\mathbf{91.38 \pm 0.29}$ | $91.31 \pm 0.34$ | **$91.38 \pm 0.29$** |
| | PathLapPE | $91.28 \pm 0.30$ | $91.03 \pm 0.28$ | $91.19 \pm 0.32$ | $\mathbf{91.33 \pm 0.23}$ | **$91.33 \pm 0.23$** |
| | PathLapPE (EdgeSPE) | $91.28 \pm 0.38$ | $\mathbf{91.37 \pm 0.32}$ | $91.30 \pm 0.29$ | $91.37 \pm 0.26$ | **$91.37 \pm 0.32$** |
| GAT | Base (No PE) | $83.01 \pm 0.35$ | $83.01 \pm 0.35$ | $83.01 \pm 0.35$ | $83.01 \pm 0.35$ | $83.01 \pm 0.35$ |
| | LaplacianPE | $83.94 \pm 0.82$ | $83.69 \pm 0.68$ | $\mathbf{84.19 \pm 0.48}$ | $84.06 \pm 0.47$ | **$84.19 \pm 0.48$** |
| | Multi-$q$-MagLapPE, $q = 0.05$ | $83.76 \pm 0.41$ | $83.79 \pm 0.54$ | $83.75 \pm 0.45$ | $\mathbf{83.93 \pm 0.56}$ | $83.93 \pm 0.56$ |
| | Multi-$q$-MagLapPE, $q \in \{0.05, 0.1, 0.15\}$ | $\mathbf{84.32 \pm 0.39}$ | $84.26 \pm 0.65$ | $84.03 \pm 0.42$ | $83.79 \pm 0.60$ | **$84.32 \pm 0.39$** |
| | PathLapPE ($256 \times 2$) | $83.72 \pm 0.64$ | $\mathbf{84.17 \pm 0.31}$ | $83.87 \pm 0.61$ | $83.96 \pm 0.52$ | $84.17 \pm 0.31$ |
| | PathLapPE | $\mathbf{84.39 \pm 0.39}$ | $83.93 \pm 0.48$ | $84.20 \pm 0.45$ | $83.97 \pm 0.48$ | **$84.39 \pm 0.39$** |
| | PathLapPE (EdgeSPE) | $83.80 \pm 0.61$ | $\mathbf{83.90 \pm 0.42}$ | $83.69 \pm 0.76$ | $83.54 \pm 0.54$ | $83.90 \pm 0.42$ |
| BiGAT | Base (No PE) | $90.36 \pm 0.30$ | $90.36 \pm 0.30$ | $90.36 \pm 0.30$ | $90.36 \pm 0.30$ | $90.36 \pm 0.30$ |
| | LaplacianPE | $\mathbf{91.15 \pm 0.40}$ | $91.03 \pm 0.39$ | $90.97 \pm 0.59$ | $91.06 \pm 0.44$ | **$91.15 \pm 0.40$** |
| | Multi-$q$-MagLapPE $q = 0.05$ | $91.12 \pm 0.37$ | $90.93 \pm 0.25$ | $90.74 \pm 0.60$ | $\mathbf{91.13 \pm 0.44}$ | $91.13 \pm 0.44$ |
| | Multi-$q$-MagLapPE, $q \in \{0.05, 0.1, 0.15\}$ | $91.06 \pm 0.45$ | $90.95 \pm 0.44$ | $\mathbf{91.10 \pm 0.45}$ | $91.04 \pm 0.35$ | $91.10 \pm 0.45$ |
| | PathLapPE ($256 \times 2$) | $91.01 \pm 0.36$ | $90.93 \pm 0.42$ | $\mathbf{91.19 \pm 0.33}$ | $90.94 \pm 0.42$ | **$91.19 \pm 0.33$** |
| | PathLapPE | $90.99 \pm 0.34$ | $90.91 \pm 0.37$ | $91.07 \pm 0.44$ | $\mathbf{91.10 \pm 0.46}$ | $91.10 \pm 0.46$ |
| | PathLapPE (EdgeSPE) | $90.95 \pm 0.35$ | $91.05 \pm 0.36$ | $90.91 \pm 0.49$ | $91.14 \pm 0.27$ | **$91.14 \pm 0.27$** |
| BiGCNE | Base (No PE) | $87.79 \pm 0.47$ | $87.79 \pm 0.47$ | $87.79 \pm 0.47$ | $87.79 \pm 0.47$ | $87.79 \pm 0.47$ |
| | LaplacianPE | $89.31 \pm 0.51$ | $\mathbf{89.58 \pm 0.57}$ | $89.40 \pm 0.41$ | $89.46 \pm 0.53$ | $89.58 \pm 0.57$ |
| | Multi-$q$-MagLapPE, $q = 0.05$ | $89.50 \pm 0.53$ | $\mathbf{89.63 \pm 0.36}$ | $89.51 \pm 0.39$ | $89.53 \pm 0.52$ | $89.63 \pm 0.36$ |
| | Multi-$q$-MagLapPE, $q \in \{0.05, 0.1, 0.15\}$ | $\mathbf{89.82 \pm 0.35}$ | $89.66 \pm 0.35$ | $89.77 \pm 0.59$ | $89.76 \pm 0.45$ | $89.82 \pm 0.35$ |
| | PathLapPE ($256 \times 2$) | $90.14 \pm 0.31$ | $\mathbf{90.24 \pm 0.28}$ | $90.21 \pm 0.40$ | $90.20 \pm 0.48$ | **$90.24 \pm 0.28$** |
| | PathLapPE | $90.22 \pm 0.34$ | $90.24 \pm 0.56$ | $\mathbf{90.29 \pm 0.32}$ | $90.23 \pm 0.32$ | **$90.29 \pm 0.32$** |
| | PathLapPE (EdgeSPE) | $89.99 \pm 0.44$ | $\mathbf{90.06 \pm 0.45}$ | $90.04 \pm 0.45$ | $89.99 \pm 0.48$ | **$90.06 \pm 0.45$** |

Table 7: Key model and training hyperparameters for the FLOWGRAPH, WEBKB and ROMAN-EMPIRE datasets. All FLOWGRAPH and WEBKB datasets share the same configs.

| | FLOWGRAPH | WEBKB | ROMAN-EMPIRE |
|---|---|---|---|
| ***Data*** | | | |
| Task | graph cls. | node cls. | node cls. |
| Graphs | 2,000 / 3,000 / 6,000 | 1 | 1 |
| Classes | 2 / 3 / 6 | 5 | 18 |
| Input feature dim. | 1[a] | 1,703 | 300 |
| Batching | batch size 32 | full graph | full graph |
| Splits | 500 / 300 / 200 per class | 10 fixed splits | 10 fixed splits |
| ***Optimization*** | | | |
| Optimizer | AdamW | Adam | AdamW |
| Learning rate | $8 \cdot 10^{-3}$ | $5 \cdot 10^{-3}$ | $3 \cdot 10^{-5}$ |
| Weight decay | 0 | $5 \cdot 10^{-4}$ | 0 |
| LR schedule | none | ReduceLROnPlateau[b] | none |
| Max epochs | 200 | 3,000 | 1,000 |
| Early stopping (val/acc) | – | 500 | 400 |
| ***Backbone*** | | | |
| Layers (pre / MP / post) | 1 / 4 / 1 | 1 / 2 / 1 | 1 / 3 / 1 |
| Stage type | skipsum | – | skipsum |
| Jumping knowledge | – | max | – |
| FFN after convolution | – | – | ✓ |
| Hidden dim. $d$ | 32 | 128 | 512 |
| Normalization | BatchNorm | L2 Norm | LayerNorm |
| Dropout | 0.0 | {0.0, 0.5} | 0.2 |
| Activation | ReLU | ReLU | GELU |
| Graph pooling | mean | – | – |
| ***Positional encodings*** | | | |
| LapPE $k$ | 64 | {16, 32} | {8, 16, 32, 64} |
| MagLapPE $k$ / $q$ / $\#q$ | 64 / 0.05 / {1, 3, 5} | {16, 32} / {0.01, 0.05, 0.1} / 1 | {8, 16, 32, 64} / {0.05, 0.1, 0.15} / {1, 3} |
| PathLapPE $k$ (node / edge) | 64 | {16, 32} / {16, 32} | {8, 16, 32, 64} |
| Encoder | SPE (+ EdgeSPE) | SPE (+ EdgeSPE) | SPE (+ EdgeSPE) |

[a] FlowGraph has no node features; we attach a constant $x_i = 1$.
[b] $\gamma = 0.1$, patience 200, monitoring validation loss.

### G.4 OPEN CIRCUIT BENCHMARK & HIGH-LEVEL SYNTHESIS

OPEN CIRCUIT BENCHMARK (OCB) and HIGH-LEVEL SYNTHESIS (HLS) are both inductive, graph-level benchmarks, with three regression tasks defined over each benchmark. For each task, we follow the implementations and optimized hyperparameters for BiGINE and GINE from Huang et al. (2025), the latter operating on the undirected version of the graphs.

**OCB** (Dong et al., 2023) consists of 8,026 operational amplifier-based (op-amp) analog circuits as directed graphs, and the task is to predict DC gain (Gain), bandwidth (BW) and phase margin (PM) for each circuit. We evaluate two Multi-$q$-MagLapPE configurations, each with ten $q$ values but with phase parameters of 0.01 and 0.05. Following Huang et al. (2025), we use random 90/5/5% splits for training, validation and testing over three-stage amplifiers only.[2]

**HLS** (Wu et al., 2022) represents 18,570 C/C++ programs as directed intermediate representation (IR) graphs for performance analysis on FPGA integrated circuits. The task is to predict digital signal processor (DSP) and look-up table (LUT) usage, and critical path timing slack (CP, permitted delay that does not affect overall program runtime) for each program. We evaluate two Multi-$q$-MagLapPE configurations, each with five $q$ values but with phase parameters of 0.01 and 0.1. We again follow Huang et al. (2025) and randomly select 1,000 graphs each for validation and testing, and train on the remaining graphs.

**Experimental setup.** Hyperparameters for all OCB and HLS experiments are provided in Tables 8 and 9, respectively. We observed several discrepancies between our results and those of Huang et al. (2025). Primarily, despite using the suggested optimized model hyperparameters and $q$ values, our Multi-$q$-MagLapPE results were marginally worse than those reported in Huang et al. (2025) on BiGINE, and significantly worse for GINE. For GINE in particular, Multi-$q$-MagLapPE

[2] While Huang et al. (2025) claim that they report two-stage results, their codebase and results are consistent with three-stage amplifiers, while two-stage amplifiers are presented as a separate OOD generalization task.

suffered from instabilities despite using five or ten $q$ values: some seeds converged to reasonable losses, while others fully diverged over the course of training, as indicated by high error values accompanied by extreme standard deviations. Finally, despite no additional tuning, we noted that our baseline No PE & LapPE configs for BiGINE $\times$ LUT significantly outperformed the numbers reported in Huang et al. (2025).

Table 8: Key model and training hyperparameters for the OPEN CIRCUIT BENCHMARK experiments. All backbones (both uni- and bidirectional GINE) share a single configuration per target.

| | GAIN | BW | PM |
|---|---|---|---|
| ***Data*** | | | |
| Input node features | | 3 | |
| Input edge features | | – | |
| Graphs | 8,026 | 8,026 | 8,026 |
| Splits | 90/5/5% | 90/5/5% | 90/5/5% |
| Batch size | 128 | 64 | 64 |
| ***Optimization*** | | | |
| Optimizer | | Adam ($\beta_1, \beta_2 = 0.9, 0.999$) | |
| Learning rate | $6.7 \cdot 10^{-3}$ | $6.5 \cdot 10^{-3}$ | $2.1 \cdot 10^{-3}$ |
| Weight decay | 0 | 0 | 0 |
| LR schedule | | StepLR (step 100, $\gamma = 0.95$) | |
| Loss | MSE | MSE | MSE |
| Max epochs | | 300 | |
| Model selection | | best val. MSE checkpoint (no early stopping) | |
| ***Backbone*** | | | |
| Layers (pre / MP / post) | 1 / 4[a] / 3 | 1 / 3[a] / 2 | 1 / 4[a] / 4 |
| Hidden dim. $d$ | 96 | 192 | 288 |
| Normalization | | BatchNorm (inside each conv MLP) | |
| Dropout | 0.1 | 0.0 | 0.2 |
| Activation | ReLU | ReLU | ReLU |
| Pooling | sum | sum | sum |

[a] First over the device-grouping subgraph, then over the full circuit graph.

### G.5 FURTHER IMPLEMENTATION DETAILS

Our codebase is based on PyTorch (Ansel et al., 2024), PyTorch Geometric (Fey et al., 2025), Hydra (Yadan, 2019) (released under an MIT license) and PyTorch Lightning (Falcon & The PyTorch Lightning team, 2019) (Apache 2.0 license).

The path complex construction was adapted from `https://github.com/SteveHuntsmanBAESystems/PerformantPathHomology` (CC BY-NC-SA 4.0 License). We also use the magnetic Laplacian and SPE implementations in `https://github.com/Graph-COM/Multi-q-Maglap`; we also extend their repository for our OCB and HLS evaluations. All experiments were conducted on NVIDIA L40S GPUs, with 8 CPUs and 48 GB RAM.

Table 9: Key model and training hyperparameters for the HIGH-LEVEL SYNTHESIS experiments. Unlike OCB, the tuned configuration depends on whether the backbone is unidirectional (GINE) or bidirectional (BiGINE), so both are reported per target. A single value in a row indicates the parameter applies to all configurations.

| | DSP | | LUT | | CP | |
|---|---|---|---|---|---|---|
| | Uni-GNN | Bi-GNN | Uni-GNN | Bi-GNN | Uni-GNN | Bi-GNN |
| ***Data*** | | | | | | |
| Categorical node features | 7 | | | | | |
| Categorical edge features | 2 | | | | | |
| Graphs (CDFG) | 18,570 | | | | | |
| Splits | 16,570 / 1,000 / 1,000 | | | | | |
| Batch size | 32 | 64 | 32 | 64 | 32 | 64 |
| ***Optimization*** | | | | | | |
| Optimizer | Adam ($\beta_1, \beta_2 = 0.9, 0.999$) | | | | | |
| Learning rate | $5.92 \cdot 10^{-3}$ | $3.10 \cdot 10^{-3}$ | $1.50 \cdot 10^{-3}$ | $2.30 \cdot 10^{-3}$ | $5.92 \cdot 10^{-3}$ | $3.10 \cdot 10^{-3}$ |
| Weight decay | 0 | | | | | |
| LR schedule | StepLR (step 100, $\gamma = 0.95$) | | | | | |
| Loss | MSE | | | | | |
| Max epochs | 400 | | | | | |
| ***Backbone*** | | | | | | |
| Layers (pre / MP / post) | $1/2/2$ | $1/4/3$ | $1/4/2$ | $1/4/2$ | $1/2/2$ | $1/4/3$ |
| Hidden dim. $d$ | 56 | 84 | 84 | 56 | 56 | 84 |
| Normalization | BatchNorm (inside each conv MLP) | | | | | |
| Dropout | 0.1 | 0.2 | 0.3 | 0.1 | 0.1 | 0.2 |
| Activation | ReLU | | | | | |
| Pooling | sum | sum | sum | sum | max | max |

## H COMPUTATIONAL COST EXPERIMENTS

PathLapPE uses more memory than Multi-$q$-MagLapPE, which processes $q$ values sequentially at fixed memory cost. For large graphs, sparse eigensolvers (Virtanen et al., 2020) reduce this memory at a modest cost in time. Table 10 reports time and memory on our benchmarks; on the largest, ROMAN-EMPIRE (22,662 nodes), sparse PathLapPE takes three seconds and 70 MB, making computational costs negligible compared to training.

Table 10: Positional encoding precompute cost per graph. Each dataset is denoted with the optimal eigensolver strategy used for decomposition and the type of memory measured.

| | ROMAN-EMPIRE shift-invert, heap mem. | | FLOWGRAPH CPU dense, heap mem. | | OCB GPU dense, GPU mem. | |
|---|---|---|---|---|---|---|
| PE | Time (s) | Mem (MB) | Time (ms) | Mem (MB) | Time (ms) | Mem (MB) |
| LapPE | $0.154 \pm 0.007$ | 4.49 | $3.00 \pm 0.23$ | 0.24 | $2.20 \pm 0.13$ | 0.56 |
| Multi-$q$-MagLapPE, $\#q = 1$ | $0.257 \pm 0.001$ | 8.50 | $2.96 \pm 0.17$ | 0.25 | $2.21 \pm 0.10$ | 1.12 |
| Multi-$q$-MagLapPE, $\#q = 3$ | $0.704 \pm 0.001$ | 11.3 | $6.59 \pm 0.44$ | 0.27 | $4.23 \pm 0.24$ | 1.12 |
| Multi-$q$-MagLapPE, $\#q = 5$ | $1.074 \pm 0.016$ | 14.1 | $10.15 \pm 0.67$ | 0.30 | $6.23 \pm 0.38$ | 1.12 |
| Multi-$q$-MagLapPE, $\#q = 10$ | $2.102 \pm 0.022$ | 21.0 | $19.43 \pm 1.81$ | 0.33 | $11.06 \pm 1.09$ | 1.12 |
| PathLapPE (ours) | $3.053 \pm 0.015$ | 70.5 | $22.46 \pm 1.15$ | 0.62 | $14.31 \pm 0.83$ | 0.60 |