GIST: Gauge-Invariant Spectral Transformers for Scalable Graph Neural Operators

Mattia Rigotti[*,†] **Nicholas Thumiger**[*] **Thomas Frick**[*]
IBM Research

Abstract

Neural operators on irregular meshes face a fundamental tension. Spectral positional encodings, a natural choice in graph learning for capturing geometry, require cubic-complexity eigendecomposition and inadvertently break gauge invariance through numerical solver artifacts; existing efficient approximations typically sacrifice gauge symmetry by design. Both failure modes break discretization invariance: models fail to transfer across mesh resolutions of the same domain, and similarly across different graphs of related structure in inductive settings. We propose GIST (Gauge-Invariant Spectral Transformer), a scalable neural operator that resolves this tension by restricting attention to pairwise inner products of efficient approximate spectral embeddings. We prove these inner products estimate an exactly gauge-invariant graph kernel at end-to-end $\mathcal{O}(N)$ complexity, and establish a formal connection between gauge invariance and discretization-invariant learning with bounded mismatch error. To our knowledge, GIST is the first scalable graph neural operator with a provable discretization-mismatch bound. Empirically, GIST sets state-of-the-art on the AirfRANS, ShapeNet-Car, DrivAerNet, and DrivAerNet++ mesh benchmarks (up to 750K nodes), and additionally matches strong baselines on standard graph benchmarks (e.g., 99.53% micro-F1 on PPI). Code available at `https://github.com/ibm/gist-neurips-2026`.

1 Introduction

Originally developed for natural language processing, Transformers [1] have since been adapted to data of increasing structural complexity, including sets [2], images [3], and video [4]. Recent work has extended them to graphs, capturing global relationships through self-attention and overcoming the long-range dependency limitations of message-passing GNNs [5, 6].

Two barriers to scalable graph neural operators. Adapting transformers to graphs introduces two distinct barriers that have not been simultaneously addressed. First, a *computational barrier*: exact spectral graph embeddings, while theoretically natural, require eigendecomposition of the graph Laplacian, scaling as $\mathcal{O}(N^3)$ in N, the number of nodes, which is prohibitively expensive at large scale. Second, a *gauge invariance barrier*: the inherent freedom to rotate eigenvectors, flip signs, or choose among degenerate eigenvectors is broken by exact methods through numerical solver artifacts (sign choices, eigenspace ordering, degeneracy handling, see Bronstein et al. 7) and by approximate methods through their commitment to a specific basis. A model trained on embeddings from one set of choices learns features that do not transfer, causing severe generalization loss in inductive learning where train and test involve different graph structures. This becomes critical for neural operators, where different mesh resolutions of the same physical domain yield different Laplacians with distinct gauge choices, so spectral embeddings at different resolutions are incomparable, as are their attention

[*]Equal contribution.
[†]Corresponding author: `mrg@zurich.ibm.com`.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

kernels. For models that use spectral information, gauge invariance (independence from arbitrary spectral choices) is therefore key both to robust inductive learning and to neural operators with provably bounded discretization error.

Existing approaches address one barrier at a time: (1) gauge-invariant spectral methods like SignNet [8] and SPE [9] maintain gauge invariance but require full eigendecomposition; (2) approximate spectral methods achieve linear complexity but break gauge invariance; (3) generic linear transformers reduce attention complexity but ignore graph structure. Recent sign- and basis-invariant methods recover gauge invariance through careful attention design but still require $\mathcal{O}(N^3)$ eigendecomposition or $\mathcal{O}(N^2)$ attention, limiting scalability.

Main contributions. We propose *Gauge-Invariant Spectral Transformers (GIST)*, overcoming both barriers through a key insight: gauge invariance is preserved by restricting attention to inner products between spectral embeddings, which remain invariant under approximate spectral computations. We show that for a broad class of efficient $\mathcal{O}(N)$ spectral embeddings $\widetilde{\Phi} = f(P) \cdot R$, where f is any spectral filter and $R \in \mathbb{R}^{N \times r}$ is random with $\mathbb{E}[RR^\top] = \mathbb{1}$, inner products are unbiased estimators of the graph kernel $K = f(P)f(P)^\top$, which depends only on P and is therefore exactly gauge-invariant. Restricting attention to these inner products and combining with linear attention yields end-to-end $\mathcal{O}(N)$ complexity, recovering gauge invariance exactly at the population level with statistical error $\mathcal{O}(r^{-1/2})$. Our main contributions are:

1. establishing, to our knowledge, the first provable discretization-mismatch bound for a graph neural operator, by proving that gauge invariance guarantees discretization-invariant learning with bounded error;
2. proposing GIST, to our knowledge the first graph neural operator architecture that simultaneously achieves gauge invariance, end-to-end $\mathcal{O}(N)$ complexity, and global spectral expressivity, validated by SOTA results on mesh problems up to 750K nodes;
3. identifying gauge invariance breaking as the underlying failure mode shared by exact and approximate spectral methods, and demonstrating its impact on generalization in inductive settings.

2 Related works

Graph transformers. Graph transformers integrate structural information through positional encodings: shortest-path and centrality features (Graphormer [10]), learnable structural encodings (LSPE [5]), relative encodings (GRPE [11]), and full-Laplacian-spectrum encodings (SAN [12]). Spectral variants improve scalability through approximate bases (SpecFormer [13]) or polynomial Laplacian filters (PolyFormer [14]) but forsake gauge invariance. Hybrid local-global designs (GraphGPS [15], Exphormer [16]) and tokenized approaches (TokenGT [17], NAGphormer [18]) address attention complexity, yet inherit positional-encoding overheads that grow super-linearly and none algorithmically preserve gauge invariance: GraphGPS, for instance, integrates spectral positional encodings but inherits their sign and basis ambiguities directly. Hierarchical Graph Transformer [6] reaches million-node scaling through coarsening at the cost of fine spectral information, while NodeFormer [19] achieves linear-complexity attention via a kernelized Gumbel-Softmax that learns latent graph structure but operates without spectral encodings. Consequently, these frameworks are rarely evaluated at the scale of `ogbn-arxiv`, where maintaining structural encodings is computationally prohibitive (GraphGPS, for instance, runs out of memory; see Table 2).

Sign- and basis-invariant positional encodings. A complementary line of work addresses gauge invariance at the architecture level: SignNet and BasisNet [8] apply sign- and basis-equivariant networks to exact Laplacian eigenvectors, and SPE [9] extends this with stable, eigenvalue-dependent partitions of degenerate eigenspaces. These methods inherit the $\mathcal{O}(N^3)$ cost of full eigendecomposition, since they require explicit eigenvectors. GIST instead achieves invariance at the kernel level: attention depends only on gauge-invariant entries $K(i,j) = [f(P)f(P)^\top]_{ij}$, estimated by inner products of $\mathcal{O}(N)$ random spectral embeddings, eliminating eigendecomposition entirely. A non-spectral parallel paradigm is taken by PEARL [20], which approximates eigenvector functions through message passing on random inputs combined with statistical pooling.

Scalable attention architectures. Recent advances tackled self-attention's quadratic scaling via cross-attention bottlenecks that map inputs to fixed-size latent representations or concepts [21, 22], kernel-based attention mechanisms using random feature approximations [23], feature map decomposition methods that linearize the attention computation [24], and memory-efficient variants with sub-linear complexity [25]. As noted by Dao and Gu [26], many such linear transformer models are directly related to linear recurrent models such as state-space-models [27–30].

Neural operators. Neural operators learn maps between continuous function spaces and ideally satisfy two properties: *discretization invariance* (a single parameter set applies across mesh resolutions of the same continuum problem) and *global integration* (representing nonlocal interactions via learned integral kernels) [31]. Foundational families realize these differently: FNO parameterizes kernels in the spectral domain via FFT [32], GNO via message passing on irregular meshes [33], and CNOs via continuous convolutions specified in the continuum [34]. Applying these methods to high-resolution meshes typically required downsampling, re-discretization onto regular lattices (e.g., SDF-based volumetric grids), or task-aware coarsening, since per-resolution graph sizes quickly become intractable. Hybrid encoder-decoder designs handle complex geometries: GINO couples a graph encoder with a latent FNO on a proxy grid [35], DeepONet uses branch/trunk networks for arbitrary coordinate queries [36], and U-NO adds a multi-resolution backbone [37].

Transformers as neural operators. Self-attention is itself a learned, data-dependent kernel integral, naturally supporting discretization-invariant operator learning when combined with suitable positional features [38, 39, 2, 40]. The Galerkin Transformer [41] formalized this interpretation, showing that softmax-free linear attention recovers a Petrov-Galerkin projection of an integral operator and giving attention a numerics-grounded reading as a learned discretized operator. Transolver casts PDE operator learning as attention from query coordinates to context tokens [42], while GNOT stabilizes operator training on irregular meshes via geometric normalization and gating [43]. Closer to our multi-scale design, AeroGTO [44] and APT [45] fuse local message passing with global attention for operator learning. GIST shares this local-global structure, inspired by EfficientViT [46], but its global branch injects spectral graph information through gauge-invariant attention, the property behind the discretization-mismatch bound of Section 4.1.

Positioning GIST. Existing graph transformers and scalable attention methods address complementary but not simultaneous challenges. Gauge-invariant spectral methods based on sign- and basis-invariant architectures applied to exact eigenvectors (SignNet [8], SPE [9]) maintain theoretical expressiveness but incur significant computational costs from full eigendecomposition. Approximate spectral methods achieve better scalability but typically do not preserve gauge invariance, which can cause generalization failures when the gauge choice differs between training and testing. Generic linear transformers reduce attention complexity but typically ignore graph structure entirely. GIST uniquely combines graph awareness through spectral embeddings, computational efficiency through random projections, gauge invariance through a modified attention mechanism, and linear attention for end-to-end linear scaling. It also preserves the discretization-invariance and global integration properties needed for neural operator applications on mesh regression, unifying graph learning and continuous function approximation in a single framework. Transformer-based operator architectures such as Transolver [42], GNOT [43], GINO [35], and the Galerkin Transformer [41] demonstrate resolution-agnosticism empirically; GIST complements this line of work by additionally providing a provable discretization-mismatch bound (Proposition A.1).

3 Approach

3.1 Preliminaries

Self-attention and positional encoding. Given query, key and value representations q_i, k_i, v_i of N tokens with $i = 1, \ldots, N$, self-attention [1] famously computes outputs as a weighted sum of values with attention weights determined by query-key similarities:

$$o_i = \sum_{j=1}^{N} \alpha_{ij} v_j, \quad \text{where } \alpha_{ij} = \mathrm{softmax}_j \left(\frac{q_i^\top k_j}{\sqrt{d}} \right). \tag{1}$$

A key insight [47] is that positional information can be injected through relative positional biases: $e_{ij} = \frac{q_i^\top k_j}{\sqrt{d}} + b_{ij}$, where b_{ij} reflects distances between positions. For graphs, this can be generalized by replacing b_{ij} with distance measures reflecting the graph structure.

Graph Laplacian and spectral embeddings. The (normalized) *graph Laplacian* $\mathcal{L} = \mathbb{1} - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ induces a natural metric via the *resistance distance*: $\Omega(i,j) = (e_i - e_j)^\top \mathcal{L}^\dagger (e_i - e_j)$, where e_i is the ith standard basis vector and $\mathcal{L}^\dagger$ is the Moore-Penrose pseudoinverse [48]. The resistance distance can be expressed via *Laplacian eigenmaps*, which satisfy:

$$\Omega(i,j) = ||\phi_i - \phi_j||^2 \quad \text{where} \quad (\phi_i)_k = \frac{1}{\sqrt{\lambda_k}}(u_k)_i, \tag{2}$$

with λ_k, u_k being the eigenvalues and eigenvectors of $\mathcal{L}$. These eigenmaps are natural graph positional encodings [49], but exact computation requires $\mathcal{O}(N^3)$ eigendecomposition, prohibitive for large graphs.

Approximate spectral embeddings and the gauge invariance problem. Note from (2) that the graph metric is determined by pairwise inner products of eigenmaps, since $\Omega(i,j) = \langle \phi_i, \phi_i \rangle - 2\langle \phi_i, \phi_j \rangle + \langle \phi_j, \phi_j \rangle$. This means we do not need the eigenvectors themselves, only an efficient way to estimate this kernel. A broad class of randomized methods achieves this, computing embeddings as

$$\widetilde{\Phi} = f(P) \cdot R \ \in \mathbb{R}^{N \times r}, \tag{3}$$

where P is a graph operator (e.g., the random-walk matrix $D^{-1}A$ or the normalized Laplacian) with eigendecomposition $P = U \Lambda U^\top$, f is a spectral filter ($f(P) = U \operatorname{diag}(f(\lambda_1), \ldots, f(\lambda_N))\, U^\top$), $R \in \mathbb{R}^{N \times r}$ is a random matrix with $\mathbb{E}[RR^\top] = \mathbb{1}$, and $\tilde{\phi}_i \in \mathbb{R}^r$ is the i-th row of $\widetilde{\Phi}$. Laplacian eigenmaps (2) are the special case $f(\lambda) = \lambda^{-1/2}$, $P = \mathcal{L}$; polynomial filters, rational filters, and random-walk diffusions also fit this template, with $f(P) \cdot R$ computable via repeated sparse matrix-vector products without forming $f(P)$ explicitly.

A key property shared by *all* methods in this class is that their pairwise inner products are *unbiased estimators* of the graph kernel $K(i,j) := [f(P)f(P)^\top]_{ij}$:

$$\mathbb{E}\Big[\langle \tilde{\phi}_i, \tilde{\phi}_j \rangle\Big] = K(i,j) = [f(P)f(P)^\top]_{ij}, \tag{4}$$

where the expectation is over the draw of R and follows from $\mathbb{E}[RR^\top] = \mathbb{1}$ (proof in Lemma A.5); when P is symmetric this reduces to $K(i,j) = \sum_l f(\lambda_l)^2 (u_l)_i (u_l)_j$. Crucially, $K(i,j)$ is a function of P alone and therefore *exactly* gauge-invariant: it is unchanged by sign flips, eigenspace rotations, and numerical solver choices (Proposition A.6). The estimation error is statistical and vanishes as $\mathcal{O}(1/\sqrt{r})$ (Lemma A.7; empirical check in Appendix C.3). In our implementation we use FastRP [50], a polynomial filter $f(P) = \sum_{i=1}^{k} P^i$ computed via sparse random projections and power iterations at $\mathcal{O}(Nrk)$ cost (see Algorithm 1), but the guarantees above hold for any method in this class, e.g. graph diffusion kernels [51] or Chebyshev polynomial filters [52].

Importantly, exact spectral methods can also introduce gauge dependence through eigensolver artifacts (sign choices, eigenspace ordering), making gauge invariance a concern across both exact and approximate approaches [7].

Motivation for gauge-invariant operations. While the inner products of approximate spectral embeddings estimate a gauge-invariant kernel, the embeddings themselves depend on arbitrary choices: the random matrix R, eigensolver conventions, and eigenvalue ordering. Any model operating on these embeddings directly, rather than on their inner products, will learn features that do not transfer across graphs, resulting in a generalization failure. Our approach addresses this by using approximate spectral embeddings as positional encodings, but restricting the Transformer to operate only on their gauge-invariant inner products.

3.2 Our approach: GIST

The key insight is that spectral embeddings of the form $\widetilde{\Phi} = f(P) \cdot R$ produce inner products $\langle \tilde{\phi}_i, \tilde{\phi}_j \rangle$ that are unbiased estimators of the exactly gauge-invariant kernel $K(i,j) = [f(P)f(P)^\top]_{ij}$ ((4)).

By restricting all operations to depend on the embeddings only through these inner products, gauge invariance is preserved exactly at the population level (with only statistical noise of order $\mathcal{O}(1/\sqrt{r})$) while maintaining computational efficiency.

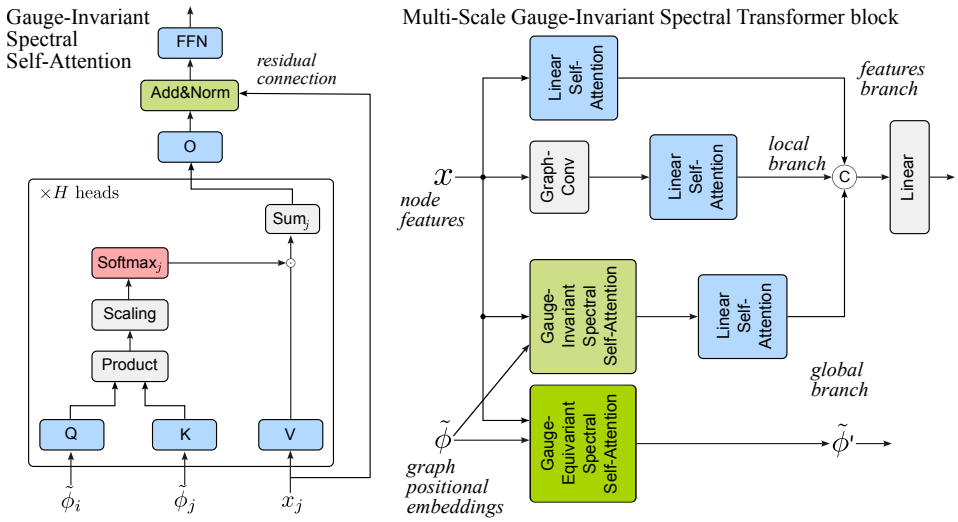

Figure 1: Gauge-Invariant Spectral Transformer. *Left*: Gauge-Invariant Spectral Self-Attention uses spectral embeddings $\tilde{\phi}$ as queries and keys and node features x as values; restricting $\tilde{\phi}$ to the query-key role preserves gauge invariance. For clarity, the panel shows the standard quadratic softmax attention implementation; GIST implements its linear-attention counterpart, which keeps the same queries, keys and values and has linear cost. *Right*: the multi-scale GIST block (inspired by EfficientViT) concatenates (C) and linearly projects the node-feature outputs of three parallel branches: feature, local and global. Embeddings $\tilde{\phi}$ reach these outputs only through the attention weights of Gauge-Invariant Spectral Self-Attention; Gauge-Equivariant Spectral Self-Attention refines them into $\tilde{\phi}'$ for the next block.

Gauge-Invariant Spectral Self-Attention. We now introduce our main contribution: *Gauge-Invariant Spectral Transformer (GIST)*. The first ingredient of GIST is *Gauge-Invariant Spectral Self-Attention*, which operates on approximate spectral embeddings $\tilde{\phi}_i \in \mathbb{R}^r$ (the i-th row of $\widetilde{\Phi} = f(P)\cdot R$) but restricts attention computations to *inner products* between embeddings. The key observation is that while the individual embeddings $\tilde{\phi}_i$ depend on the random matrix R, their inner products $\langle \tilde{\phi}_i, \tilde{\phi}_j \rangle$ are unbiased estimators of the gauge-invariant kernel $K(i,j)$ ((4)), which depends only on P. Attention weights computed from these inner products are therefore gauge-invariant and transfer across graphs.

Formally, for each node $i = 1, \ldots, N$, Gauge-Invariant Spectral Self-Attention modifies the standard self-attention mechanism as follows:

$$q_i = \tilde{\phi}_i, \quad k_i = \tilde{\phi}_i, \quad v_i = W_v x_i,$$

where W_v is a learned linear projection of the node features x_i, as usual in transformers. This ensures attention logits are based on inner products: $e_{ij} = q_i^\top k_j = \tilde{\phi}_i^\top \tilde{\phi}_j = K(i,j) + \mathcal{O}(1/\sqrt{r})$, where $K(i,j) = [f(P)f(P)^\top]_{ij}$ is the exactly gauge-invariant kernel from (4) (see Fig. 1, left). By limiting the embeddings to the query-key computation and not using them as values, we ensure that downstream layers (which operate on node features) cannot access gauge-dependent information. Algorithm 1 in Section B gives our specific implementation of the spectral embedding computation (which uses FastRP), and Algorithm 2 details the attention implementation. Notice that the softmax in (1) is replaced by linear attention in our implementation (see the *Linear Self-Attention* paragraph below).

Gauge-Equivariant Spectral Self-Attention. The Gauge-Invariant Spectral Self-Attention operation thus preserves gauge invariance, but at the cost of giving up a lot of the flexibility of regular

self-attention. In particular, there is no mechanism for modifying the vectors $\tilde{\phi}_i$ through learning while preserving the property that attention logits track the gauge-invariant kernel K. Applying a learnable linear map W to $\tilde{\phi}_i$, for instance, would replace K with the parameter-dependent rescaling $\big(\mathrm{Tr}(W^\top W)/r\big) \cdot K$ in expectation. A feature-dependent scalar rescaling $s(x_i)\tilde{\phi}_i$, in contrast, preserves the kernel structure as a feature-driven modulation: $(s(x_i)\tilde{\phi}_i)^\top (s(x_j)\tilde{\phi}_j) = s(x_i)s(x_j)(\tilde{\phi}_i^\top \tilde{\phi}_j) = s(x_i)s(x_j) \cdot K(i,j) + \mathcal{O}(1/\sqrt{r})$, an *equivariant* operation in node features.

Remarkably, such a gauge-equivariant operation can also be straight-forwardly implemented via a modification of self-attention by modifying (1) as follows:

$$q_i = W_q x_i, \quad k_i = W_k x_i, \quad v_i = \tilde{\phi}_i,$$

and the output in (1) such that it is constrained to operate on $\tilde{\phi}_i$, i.e. $\tilde{\phi}_i^{l+1} = \sum_{j=1}^{N} \alpha_{ij} v_j$, where $\tilde{\phi}_i^{l+1}$ indicates the graph positional encoding that will be used in the next layer $l+1$. Algorithm 3 in Section B details how the implementation of Gauge-Equivariant Spectral Self-Attention relates to regular Self-Attention.

Here W_q, W_k act on node features only, and no learnable map acts on the embedding coordinates: these depend on the random draw of R, which differs from graph to graph, so weights fitted to them would be specific to one draw (in expectation over R, a linear map only rescales K, as noted above). The block thus generalizes the per-node rescaling $s(x_i)\tilde{\phi}_i$ to feature-dependent mixing across nodes, with coefficients α_{ij} computed from node features alone: like the rescaling, it combines embeddings of different nodes and never mixes the coordinates of an embedding. Combining the two blocks recovers learnable query-key projections within a gauge-respecting representation: the gauge-invariant block exposes $K(i,j)$ as attention logits, while the gauge-equivariant block reinstates W_q, W_k on node features with $\tilde{\phi}$ as values.

Linear self-attention, and multi-scale architecture. Gauge-Invariant Spectral Self-Attention ensures that we can compute reliable graph positional encoding with linear time complexity in the number of nodes in the graph N. In order to maintain that linear scaling end-to-end, all GIST attention blocks replace softmax attention (introduced above and depicted in Fig. 1, left, for clarity) with the *linear transformer* by Katharopoulos et al. [24], using a feature map φ so that $\alpha_{ij} \propto \langle \varphi(q_i), \varphi(k_j) \rangle$.

Crucially, as feature map we use $\varphi(x) = \mathrm{ReLU}(x)$, which is known to correspond to a well-defined kernel (the first-order arc-cosine kernel [53]). In practice, we use $\varphi(x) = \mathrm{ReLU}(x - \theta)$, with a trainable scalar θ. Gauge invariance of the attention weights is established at the sample level: for any fixed realization of R, $\tilde{\phi}_i = [f(P)R]_i$ is a deterministic function of the gauge-independent operator P (no eigendecomposition is performed), so $\langle \varphi(\tilde{\phi}_i), \varphi(\tilde{\phi}_j) \rangle$ is exactly invariant under sign flips, eigenspace rotations, and solver choices. In expectation, the inner products estimate the gauge-invariant kernel $K(i,j) = [f(P)f(P)^\top]_{ij}$ ((4)), connecting the population-level attention to K. For further considerations on the choice of the feature map $\varphi(\cdot)$ see Appendix B.

To mitigate linear attention's drawbacks (notably lack of sharp attention scores compared to softmax attention), we design a parallel architecture inspired by EfficientViT [46], a multi-scale linear attention architecture. Our *Multi-Scale Gauge-Invariant Spectral Transformer Block* comprises three parallel branches: a feature branch (linear transformer on node features x), a local branch (graph convolution on x followed by linear transformer), and a global branch (Gauge-Invariant Spectral Self-Attention on x and $\tilde{\phi}$, followed by linear transformer). In parallel, Gauge-Equivariant Spectral Self-Attention on x and $\tilde{\phi}$ produces the refined embeddings used by the next block. The local branch emphasizes local information that linear attention would otherwise diffuse, while the global branch integrates global graph information. See Figure 1 (right panel) for the architecture. Ablation studies in Appendix C.5 confirm that each branch meaningfully contributes to performance. This parallel design also covers the alternative construction of concatenating node features and embeddings to form queries and keys. Without learnable projections, the feature map acts blockwise on the concatenation, $\varphi([x_i; \tilde{\phi}_i]) = [\varphi(x_i); \varphi(\tilde{\phi}_i)]$, so the attention score splits exactly into $\langle \varphi(x_i), \varphi(x_j) \rangle + \langle \varphi(\tilde{\phi}_i), \varphi(\tilde{\phi}_j) \rangle$, and the linear-attention output is a per-node convex combination, weighted by the two normalizers, of a feature-only attention and Gauge-Invariant Spectral Self-Attention sharing one value projection. Our branches compute the two terms separately, adding learnable W_q, W_k to the feature term and separate output projections; a learnable projection of the concatenated vector would instead act on the embedding coordinates, which we avoid as discussed above.

Complexity scaling analysis. GIST computational complexity is dominated by two components. First, spectral embedding computation scales as $\mathcal{O}(N \cdot r \cdot k)$, where N is the number of nodes, r is the embedding dimension, and k is the number of power iterations (e.g., k polynomial filter steps for FastRP). Second, linear transformer blocks combining Gauge-Invariant and Gauge-Equivariant Spectral Self-Attention scale as $\mathcal{O}(N \cdot d^2 + N \cdot r \cdot d)$ on d-dimensional node features and r-dimensional spectral embeddings, where the $N \cdot d^2$ term comes from the feature projections and feature-only linear attention, and the $N \cdot r \cdot d$ term from the attention steps that couple r-dim spectral embeddings with d-dim node features. Overall, this gives end-to-end $\mathcal{O}(N \cdot d^2 + N \cdot r \cdot d + N \cdot r \cdot k)$ scaling, i.e. linear in N. This contrasts with $\mathcal{O}(N^3)$ for exact eigendecomposition and $\mathcal{O}(N^2 d)$ for standard quadratic attention. This linear scaling is empirically validated in Appendix C.4: Figure 4 demonstrates that both VRAM consumption and forward pass time grow linearly with graph size up to 500K nodes on DrivAerNet samples, confirming our theoretical analysis and enabling the large-scale neural operator experiments in Section 4.

4 Results

This section validates GIST both theoretically and empirically: a provable discretization-mismatch bound (§4.1), a controlled cross-resolution mechanism check (§4.2), node classification on five standard graph benchmarks, and surface-pressure regression on four neural operator benchmarks (§4.4).

4.1 Theoretical guarantees: GIST as discretization-invariant neural operator

We now formalize how gauge invariance enables discretization-invariant learning with bounded error. Physical problems (computational fluid dynamics, structural mechanics, shape analysis) are discretized into computational meshes (i.e. graphs), with different resolutions producing different graph Laplacians and spectral decompositions involving arbitrary gauge choices (sign flips, eigenspace rotations, solver artifacts). Without gauge invariance, parameters fail to transfer across discretizations; gauge invariance ensures attention weights converge to a well-defined continuum kernel with bounded discretization mismatch error.

Proposition 4.1 (Informal formulation of Corollary A.8)**.** *The output of a gauge-invariant spectral self-attention block of GIST, applied to two nested random samplings of the same m-dimensional manifold, converges across discretizations with error $\mathcal{O}(n^{-1/(m+4)}) + \mathcal{O}(r^{-1/2})$, where n is the coarser resolution and r is the spectral embedding dimension. This ensures learned parameters transfer across mesh resolutions with bounded error that vanishes as resolution and embedding dimension increase.*

Proof sketch. Cross-discretization output mismatch decomposes additively into a discretization error and a statistical error. Graph Laplacians of n-point samples of an m-dimensional manifold converge spectrally to the Laplace-Beltrami operator at rate $\mathcal{O}(n^{-1/(m+4)})$ [54, 55], so any kernel $K = f(P)f(P)^\top$ from a smooth spectral filter converges across discretizations at the same rate (Corollary A.3). Inner products of $\widetilde{\Phi} = f(P) \cdot R$ unbiasedly estimate K with statistical error $\mathcal{O}(r^{-1/2})$ (Lemma A.7), and K depends only on P, hence is exactly gauge-invariant (Proposition A.6). Since gauge-invariant attention depends on the embeddings only through these inner products, the triangle inequality yields an output-mismatch bound $\mathcal{O}(n^{-1/(m+4)}) + \mathcal{O}(r^{-1/2})$. Full statements and proofs in Appendix A.

This result distinguishes GIST from prior spectral methods: while approximate methods achieve computational efficiency, they lack gauge invariance and thus cannot provide discretization error bounds; while exact methods could maintain invariance, their $\mathcal{O}(N^3)$ cost prevents scaling to high-resolution meshes where such bounds are most valuable. GIST uniquely combines gauge invariance with end-to-end linear complexity, achieving both theoretical guarantees and practical scalability for neural operator applications.

4.2 Discretization invariance: controlled mechanism check

To isolate the mechanism predicted by Proposition 4.1, we construct a minimal setup in which gauge invariance is the only architectural difference between two matched models, leaving the broader empirical assessment of the full architecture to the operator-learning benchmarks of Section 4.4.

Table 1: Controlled discretization-invariance check on DrivAerNet surface pressure. One model per architecture is trained at 50% decimation and evaluated without retraining on the same cars at four resolutions (R^2, mean $\pm$ std across 600 cars). Both architectures match at the training resolution; away from it, the non-gauge-invariant baseline loses up to 47% of its R^2, the gauge-invariant one at most 19%.

Evaluation resolution	**GIST (gauge-invariant)** $R^2 \uparrow$	**Non-gauge-invariant** $R^2 \uparrow$
25% ($\sim$125K)	**0.796**$\pm$0.028	0.541$\pm$0.041
50% ($\sim$250K), training res.	0.987$\pm$0.012	0.989$\pm$0.018
75% ($\sim$375K)	**0.861**$\pm$0.018	0.702$\pm$0.035
100% ($\sim$500K)	**0.836**$\pm$0.021	0.524$\pm$0.033

Experimental setup. We train two architectures with matched parameters on 600 cars of DrivAerNet [56] at 50% decimation ($\sim$250K vertices) and evaluate them without retraining on the same cars at 25, 50, 75 and 100% of the vertices, predicting surface pressure. The *gauge-invariant* architecture is a GIST block followed by a linear transformer block. The *non-gauge-invariant* baseline uses a linear transformer block with spectral embeddings summed to node features (treating them as standard positional encodings), which breaks gauge invariance by using the projected embeddings directly as features rather than restricting attention to their inner products.

Results. Table 1 shows both architectures achieve the same R^2 at the training resolution (0.987 vs 0.989), confirming matched model capacity. Away from it, the gauge-invariant architecture degrades gradually (at most a 19% relative drop, 0.796 R^2 at 25%), while the non-gauge-invariant baseline fails catastrophically (up to a 47% relative drop, 0.524 R^2 at full resolution). Within the scope of this controlled comparison, gauge invariance is the sole architectural factor that distinguishes successful cross-resolution transfer from catastrophic failure, consistent with the mechanism predicted by Proposition 4.1. Exact gauge-invariant methods (SignNet [8], SPE [9]) are gauge-invariant by construction and inherit the same cross-resolution transfer property. Their $\mathcal{O}(N^3)$ eigendecomposition, however, restricts them to the small-graph regime and does not apply at the mesh resolutions of Section 4.4, where the discretization-mismatch bound becomes consequential. The same mechanism explains why gauge invariance helps little on the node-classification benchmarks (Section 4.3): on the single-graph benchmarks, train and test share one graph and hence one gauge, and on PPI the strongest baselines are message-passing GNNs, gauge-invariant by construction.

4.3 Node classification tasks

We evaluate GIST on six standard node-classification benchmarks: the Planetoid citation graphs Cora, CiteSeer and PubMed [57–59], the Amazon Photo co-purchase network and the ogbn-arxiv citation graph [60, 61], and the PPI protein interaction benchmark. The first five are single-graph benchmarks where train, validation, and test are disjoint node subsets of one graph; PPI consists of 24 disjoint tissue-specific protein–protein interaction graphs and is the multi-graph case in this suite, where train and test involve different graph structures and gauge invariance becomes most consequential. We follow the official splits in each case, reporting micro-averaged F1 on the multi-label PPI benchmark and accuracy elsewhere. Spectral embedding hyperparameters (k and r) are selected by HPO; Appendix C.2 confirms robust performance across a wide range, with accuracy saturating at relatively low r, consistent with the $\mathcal{O}(r^{-1/2})$ concentration bound (Lemma A.7).

GIST is competitive across all six benchmarks (Table 2), with four notable highlights: best mean accuracy on CiteSeer ($73.85\% \pm 0.63$), PubMed ($82.60\% \pm 0.37$), Photo ($95.65\% \pm 0.46$, on par with SpecFormer's $95.48\% \pm 0.16$), and matching SOTA on PPI ($99.53\% \pm 0.01$ micro-F1, matching GCNII). On Cora and ogbn-arxiv, GIST stays within 1.3 and 0.4 points of the best reported method, at end-to-end $\mathcal{O}(N)$ complexity (GraphGPS goes OOM on ogbn-arxiv).

Table 2: Node classification on standard graph benchmarks. GIST is competitive across all six benchmarks at end-to-end $\mathcal{O}(N)$ complexity, with the best mean accuracy on CiteSeer, PubMed and Photo, and matching SOTA on PPI. Cora, CiteSeer, PubMed, Photo, and ArXiv report test accuracy; PPI reports micro-F1; all metrics are percentages and higher-is-better, reported as mean $\pm$ std across 6 random seeds for GIST. Sources are cited per model for baselines. *source reports 95% CI rescaled to std. †source does not specify the uncertainty type. '–' indicates results not reported by the source. Bold denotes the best mean in each column.

Model	**Cora** (accuracy ↑)	**CiteSeer** (accuracy ↑)	**PubMed** (accuracy ↑)	**Photo** (accuracy ↑)	**ArXiv** (accuracy ↑)	**PPI** (micro-F1 ↑)
GCN [62–64, 13]	85.10±0.67	73.14±0.67	81.12±0.52	88.26±0.37*	71.74±0.29	51.50±0.60†
GraphSAGE [62, 65, 64]	83.88±0.65	72.26±0.55	79.72±0.50	–	71.49±0.27	76.80
GAT [62, 65, 64, 13]	84.46±0.55	72.22±0.84	80.28±0.64	90.94±0.34*	71.59±0.38	97.30±0.02
GATv2 [66, 64]	82.90	71.60	78.70	–	71.87±0.25	96.30
GaAN [67]	–	–	–	–	–	98.71
Cluster-GCN [68]	–	–	–	–	–	99.36
GraphSAINT [63]	–	–	–	–	–	99.50±0.00†
GCNII [69, 13]	**85.50**±0.50	73.40±0.60	80.20±0.40	89.94±0.16*	72.04±0.10*	**99.53**±0.01
GCNIII [70]	–	–	–	–	–	99.50±0.03
GraphGPS [15]	–	–	–	–	OOM	–
SGFormer [62, 71]	84.82±0.85	72.72±1.15	80.60±0.49	–	**72.63**±0.13	–
SpecFormer [13]	–	–	–	95.48±0.16*	72.37±0.09*	–
PolyFormer [14]	–	–	–	–	72.42±0.19	–
Exphormer (LapPE) [16]	–	–	–	95.27±0.42	72.44±0.28†	–
GIST (ours)	84.25±0.65	**73.85**±0.63	**82.60**±0.37	**95.65**±0.46	72.31±0.29	**99.53**±0.01

4.4 Neural operators

We validate GIST's discretization-invariant properties (Section 4.1) on four surface pressure prediction benchmarks of increasing scale, all involving node-level regression of aerodynamic fields on unseen geometries.

AirfRANS [72] is a high-fidelity computational fluid dynamics dataset of 2D airfoils under Reynolds-Averaged Navier-Stokes (RANS) simulation, comprising 1000 airfoil designs with approximately 200K mesh nodes per sample across Reynolds numbers 2–6M and angles of attack from -5° to 15°. ShapeNet-Car [73] is a collection of 3D car geometries used for aerodynamic prediction, containing approximately 3700 mesh points per sample. DrivAerNet and DrivAerNet++ [56] are distinct high-fidelity CFD datasets of parametric car geometries: DrivAerNet comprises 4,000 designs with approximately 500K surface vertices per car, while DrivAerNet++ extends this to over 10,000 designs with up to $\sim$750K vertices per car and greater geometric diversity across multiple vehicle classes (SUVs, sedans, hatchbacks). All datasets model cars or airfoils as graphs with nodes representing surface mesh vertices and edges following mesh connectivity.

Table 3: Surface pressure prediction error on AirfRANS and ShapeNet-Car. GIST outperforms Transolver on both, with an $8\times$ reduction in MSE on AirfRANS. Baseline is Transolver [42]. Reported metrics are MSE on AirfRANS and relative ℓ_2 on ShapeNet-Car; GIST mean $\pm$ std over 3 seeds.

Model	**AirfRANS** ↓	**ShapeNet-Car** ↓
Transolver	0.0142	7.45%
GIST (ours)	**0.0017**±0.0002	**7.02%**±0.05

On the intermediate-scale benchmarks (AirfRANS and ShapeNet-Car, Table 3), GIST outperforms Transolver [42], achieving 0.0017 MSE on AirfRANS (vs. 0.0142) and 7.02% relative ℓ_2 on ShapeNet-Car (vs. 7.45%). Both results follow the same train/test splits and surface-pressure metrics (MSE over surface nodes on AirfRANS, relative ℓ_2 per car on ShapeNet-Car) as reported by Transolver, using the standard AirfRANS protocol of Bonnet et al. [72]. On the large-scale benchmarks (DrivAerNet and DrivAerNet++, Table 4), GIST achieves state-of-the-art on both: 20.03% relative ℓ_2 error on DrivAerNet (vs. 20.35% for the previous best) and 18.60% on DrivAerNet++ (vs. 20.05%), outperforming RegDGCNN [56], Transolver [42], FigConvNet [74], and TripNet [75]. GIST's linear scaling enables direct processing of these meshes (up to $\sim$750K vertices on DrivAerNet++) without downsampling, while prior methods require lossy projection to regular

grids or lower-dimensional latent spaces. GIST's spectral attention provides global receptive fields in a single layer, which aids performance on all these tasks.

Table 4: Surface pressure prediction error on DrivAerNet and DrivAerNet++. GIST sets state-of-the-art on both benchmarks, processing meshes directly at full resolution.

	DrivAerNet		**DrivAerNet++**	
Model	**MSE** $\downarrow$	**Rel** $\ell_2 \downarrow$	**MSE** $\downarrow$	**Rel** $\ell_2 \downarrow$
RegDGCNN	9.01×10^{-2}	28.49%	8.29×10^{-2}	27.72%
Transolver	5.37×10^{-2}	22.52%	7.15×10^{-2}	23.87%
FigConvNet	4.38×10^{-2}	20.98%	4.99×10^{-2}	20.86%
TripNet	4.23×10^{-2}	20.35%	4.55×10^{-2}	20.05%
GIST (ours)	$\mathbf{4.00 \times 10^{-2}}$	**20.03%**	$\mathbf{3.63 \times 10^{-2}}$	**18.60%**

5 Conclusions

We presented GIST, a gauge-invariant spectral transformer that sets state-of-the-art on four mesh-regression benchmarks at up to 750K nodes, with contributions spanning three levels. First, we proved formally that gauge invariance guarantees discretization-invariant neural operators, establishing, to our knowledge, the first provable discretization-mismatch bound $\mathcal{O}(n^{-1/(m+4)}) + \mathcal{O}(r^{-1/2})$ for a graph neural operator. Second, we proposed an architecture that achieves gauge invariance, end-to-end $\mathcal{O}(N)$ complexity, and global spectral expressivity simultaneously, three properties that no prior method combines. Third, we identified gauge invariance breaking as the underlying failure mode shared by both exact and approximate spectral methods, causing severe generalization loss in inductive settings. By restricting attention to inner products of spectral embeddings, which are unbiased estimators of an exactly gauge-invariant kernel, GIST recovers the symmetry algorithmically while maintaining linear scaling. Empirically, GIST sets a new state-of-the-art on mesh regression across AirfRANS, ShapeNet-Car, DrivAerNet, and DrivAerNet++ (up to 500K and 750K nodes for the latter two), processing these meshes directly at full resolution where methods requiring quadratic attention or cubic eigendecomposition cannot scale; on standard graph benchmarks (Cora, CiteSeer, PubMed, Photo, ogbn-arxiv, PPI), GIST achieves competitive results.

Broader impact, limitations, and future work. GIST is a methodological contribution to graph neural operators, with applications in mesh-based scientific computing such as CFD-based aerodynamic and structural simulation; we do not foresee direct negative societal impacts beyond those generic to scientific simulation tooling. On the theoretical side, the discretization-mismatch bound rests on the assumptions of Calder and García Trillos [55], namely i.i.d. uniform sampling on a bounded-curvature manifold, conditions that CFD meshes, which are refined non-uniformly (more finely where geometry or flow fields vary most), satisfy only approximately. All our mesh experiments use the benchmarks' non-uniform meshes, so the reported results already reflect the effect of non-uniform sampling; extending the bound to non-uniform sampling is a natural next step. Empirically, our scope spans homophilic graphs and physical meshes, leaving heterophilic settings for future investigation; the benefit of gauge invariance is also limited on single-graph benchmarks, where train and test share one gauge (Section 4.2). We also precompute the spectral embedding under the assumption of a static input graph, and adapting GIST to dynamic graphs whose topology evolves over time, by recomputing or updating the embedding online, is left to future work.

Acknowledgments and Disclosure of Funding

This work was supported by the European Union's Horizon Europe research and innovation programme through the NATURE-DEMO project under grant agreement No 101157448, and by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract No 24.00279. The authors declare no competing interests.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[2] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, 2019.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

[4] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

[5] 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 (ICLR)*, 2022.

[6] Wenhao Zhu, Tianyu Wen, Guojie Song, Xiaojun Ma, and Liang Wang. Hierarchical transformer for scalable graph learning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2023.

[7] 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.

[8] 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 *International Conference on Learning Representations (ICLR)*, 2023.

[9] 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 (ICLR)*, 2024.

[10] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[11] Wonpyo Park, Woonggi Chang, Donggeon Lee, Juntae Kim, and Seung-won Hwang. GRPE: Relative positional encoding for graph transformer. *arXiv preprint arXiv:2201.12787*, 2022.

[12] Devin Kreuzer, Dominique Beaini, William L. Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[13] Deyu Bo, Chuan Shi, Lele Wang, and Renjie Liao. Specformer: Spectral graph neural networks meet transformers. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

[14] Jiahong Ma, Mingguo He, and Zhewei Wei. PolyFormer: Scalable node-wise filters via polynomial graph transformer. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2118–2129, 2024.

[15] Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35, 2022.

[16] Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J. Sutherland, and Ali Kemal Sinop. Exphormer: Sparse transformers for graphs. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

[17] Jinwoo Kim, Tien Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. *Advances in Neural Information Processing Systems*, 35, 2022.

[18] Jinsong Chen, Kaiyuan Gao, Gaichao Li, and Kun He. NAGphormer: A tokenized graph transformer for node classification in large graphs. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

[19] Qitian Wu, Wentao Zhao, Zenan Li, David Wipf, and Junchi Yan. NodeFormer: A scalable graph structure learning transformer for node classification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[20] Charilaos Kanatsoulis, Evelyn Choi, Stefanie Jegelka, Jure Leskovec, and Alejandro Ribeiro. Learning efficient positional encodings with graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2025.

[21] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International Conference on Machine Learning*, pages 4651–4664. PMLR, 2021.

[22] M. Rigotti, C. Miksovic, I. Giurgiu, T. Gschwind, and P. Scotton. Attention-based interpretability with concept transformers. In *International Conference on Learning Representations (ICLR)*, 2022.

[23] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations (ICLR)*, 2020.

[24] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

[25] Valerii Likhosherstov, Krzysztof M. Choromanski, Jared Quincy Davis, Xingyou Song, and Adrian Weller. Sub-linear memory: How to make performers slim. *Advances in Neural Information Processing Systems*, 34:6707–6719, 2021.

[26] Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

[27] Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state space layers. *Advances in neural information processing systems*, 34:572–585, 2021.

[28] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations (ICLR)*, 2022.

[29] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[30] Leena Chennuru Vankadara, Jin Xu, Moritz Haas, and Volkan Cevher. On feature learning in structured state space models. *Advances in Neural Information Processing Systems*, 37: 86145–86179, 2024.

[31] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, 24:1–97, 2023.

[32] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

[33] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.

[34] Bogdan Raonić, Roberto Molinaro, Tim De Ryck, Tobias Rohner, Francesca Bartolucci, Rima Alaifari, Siddhartha Mishra, and Emmanuel de Bézenac. Convolutional neural operators for robust and accurate learning of PDEs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

[35] Zongyi Li, Nikola Borislavov Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Prakash Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, and Anima Anandkumar. Geometry-informed neural operator for large-scale 3d PDEs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

[36] Lu Lu, Pengzhan Jin, Guofei Pang, Zongyi Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

[37] Md Ashiqur Rahman, Zachary E. Ross, and Kamyar Azizzadenesheli. U-NO: U-shaped neural operators. *Transactions on Machine Learning Research (TMLR)*, 2022.

[38] Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. In *Proceedings of EMNLP-IJCNLP*, pages 4344–4353. Association for Computational Linguistics, 2019.

[39] Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J. Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions? In *International Conference on Learning Representations (ICLR)*, 2020.

[40] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Cătălin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J. Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver IO: A general architecture for structured inputs & outputs. In *International Conference on Learning Representations (ICLR)*, 2021.

[41] Shuhao Cao. Choose a transformer: Fourier or Galerkin. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[42] Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for PDEs on general geometries. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pages 53681–53705. JMLR.org, 2024.

[43] Zhongkai Hao, Zhengyi Wang, Hang Su, Chengyang Ying, Yinpeng Dong, Songming Liu, Ze Cheng, Jian Song, and Jun Zhu. GNOT: A general neural operator transformer for operator learning. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202. PMLR, 2023.

[44] Pengwei Liu, Pengkai Wang, Xingyu Ren, Hangjie Yuan, Zhongkai Hao, Chao Xu, Shengze Cai, and Dong Ni. AeroGTO: An efficient graph-transformer operator for learning large-scale aerodynamics of 3D vehicle geometries. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 18924–18932, 2025.

[45] Xin Ju, Hadrian Fung, Yuyan Zhang, Carl Jacquemyn, Matthew D. Jackson, Randolph R. Settgast, Sally M. Benson, and Gege Wen. Adaptive physics transformer with fused global-local attention for subsurface energy systems. In *Forty-Third International Conference on Machine Learning*, June 2026.

[46] Han Cai, Junyan Li, Muyan Hu, Chuang Gan, and Song Han. EfficientViT: Multi-scale linear attention for high-resolution dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2022.

[47] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proceedings of the NAACL-HLT*, 2018.

[48] D. J. Klein and M. Randić. Resistance distance. *Journal of Mathematical Chemistry*, 12(1): 81–95, 1993.

[49] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv:2012.09699 [cs]*, 2021.

[50] Haochen Chen, Syed Fahad Sultan, Yingtao Tian, Muhao Chen, and Steven Skiena. Fast and accurate network embeddings via very sparse random projection. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 399–408. Association for Computing Machinery, 2019.

[51] Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th International Conference on Machine Learning*, pages 315–322, 2002.

[52] David K. Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[53] Youngmin Cho and Lawrence Saul. Kernel methods for deep learning. In *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009.

[54] Mikhail Belkin and Partha Niyogi. Towards a theoretical foundation for laplacian-based manifold methods. *Journal of Computer and System Sciences*, 74(8):1289–1308, 2008.

[55] Jeff Calder and Nicolás García Trillos. Improved spectral convergence rates for graph laplacians on epsilon-graphs and k-nn graphs. *Applied and Computational Harmonic Analysis*, 60:123–175, 2022.

[56] Mohamed Elrefaie, Angela Dai, and Faez Ahmed. DrivAerNet: A parametric car dataset for data-driven aerodynamic design and graph-based drag prediction. In *Proceedings of the ASME International Design Engineering Technical Conferences (IDETC-CIE)*, 2024.

[57] Prithviraj Sen, Gal Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

[58] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 40–48, 2016.

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

[60] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[61] Matthias Fey, Jinu Sunil, Akihiro Nitta, Rishi Puri, Manan Shah, Blaž Stojanovič, Ramona Bendias, Alexandria Barghi, Vid Kocijan, Zecheng Zhang, Xinwei He, Jan E. Lenssen, and Jure Leskovec. PyG 2.0: Scalable learning on real world graphs. In *Temporal Graph Learning Workshop @ KDD*, 2025.

[62] Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Classic GNNs are strong baselines: Reassessing GNNs for node classification. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track (NeurIPS D&B)*, 2024.

[63] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. GraphSAINT: Graph sampling based inductive learning method. In *International Conference on Learning Representations (ICLR)*, 2020.

[64] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations (ICLR)*, 2022.

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

[66] Julie Choi. Personalized PageRank graph attention networks. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[67] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. GaAN: Gated attention networks for learning on large and spatiotemporal graphs. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018.

[68] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 257–266, 2019.

[69] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

[70] Yancheng Chen, Wenguo Yang, and Zhipeng Jiang. Wide & deep learning for node classification. *arXiv preprint arXiv:2505.02020*, 2025.

[71] Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. Sgformer: Simplifying and empowering transformers for large-graph representations. *Advances in neural information processing systems*, 36:64753–64773, 2023.

[72] Florent Bonnet, Jocelyn Mazari, Paola Cinnella, and Patrick Gallinari. Airfrans: High fidelity computational fluid dynamics dataset for approximating reynolds-averaged navier–stokes solutions. *Advances in Neural Information Processing Systems*, 35:23463–23478, 2022.

[73] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[74] Chris Choy, Alexey Kamenev, Jean Kossaifi, Max Rietmann, Jan Kautz, and Kamyar Azizzadenesheli. Factorized implicit global convolution for automotive computational fluid dynamics prediction. *arXiv preprint arXiv:2502.04317*, 2025.

[75] Qian Chen, Mohamed Elrefaie, Angela Dai, and Faez Ahmed. TripNet: Learning large-scale high-fidelity 3d car aerodynamics with triplane networks. *arXiv preprint arXiv:2503.17400*, 2025.

[76] Mark Rudelson and Roman Vershynin. Hanson-wright inequality and sub-gaussian concentration. *Electronic Communications in Probability*, 2013.

[77] Ping Li, Trevor J. Hastie, and Kenneth W. Church. Very sparse random projections. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 287–296. ACM, 2006.

A Proofs

A.1 Proof of Proposition A.1: Gauge-Invariant Spectral Self-Attention

We prove that the Gauge-Invariant Spectral Self-Attention mechanism is discretization-invariant with quantifiable discretization mismatch error through three stages of analysis. The proof establishes that the attention kernel $K(i,j) = [f(P)f(P)^\top]_{ij}$ estimated by the spectral embeddings converges to a well-defined continuum kernel as mesh resolution increases, allowing us to bound the discretization mismatch error between any two discretizations of the same manifold. We carry out the analysis on the rescaled Laplacian $\mathcal{L}_n$, for which sharp spectral-convergence rates are available [55]; the same conclusions extend to rescaled variants of $\mathcal{L}_n$, including the random-walk operator $P_n = D^{-1}A$ used in our FastRP implementation (Remark A.4 below).

For completeness, we restate the full proposition:

Proposition A.1 (Full Statement)**.** *Gauge-Invariant Spectral Self-Attention is a discretization-invariant Neural Operator with bounded discretization mismatch error. Let M be a compact m-dimensional Riemannian manifold and $\mathcal{G}_n$ a sequence of graphs obtained by sampling n points $\{x_i\}_{i=1}^n \subset M$ as nodes, with $n \to \infty$. Let f be a spectral filter applied to a graph operator P_n of $\mathcal{G}_n$, and let $\phi_i^n = f(P_n)_{i\cdot}^\top$ be the spectral embedding of node i (with $\ker(P_n)$ projected out when f has a pole at 0). Then:*

(i) ***(Convergence to a continuum kernel.)*** *When $f(\lambda) = \lambda^{-1/2}$ and $P_n = \mathcal{L}_n$, the inner products $\langle \phi_i^n, \phi_j^n \rangle$ converge to the Green's function $G_M(x_i, x_j)$ of the Laplace-Beltrami operator at rate $\mathcal{O}(n^{-1/(m+4)})$ (Proposition A.2). For spectral filters f for which $g(\lambda) := f(\lambda)^2$ is Lipschitz away from 0 and $\sum_l f(\mu_l)^2 < \infty$, the kernel $K_n(i,j) = \langle \phi_i^n, \phi_j^n \rangle = [f(P_n)f(P_n)^\top]_{ij}$ converges to a continuum kernel $K_\infty(x_i, x_j)$ at the same rate (Corollary A.3).*

(ii) ***(Discretization mismatch bound.)*** *For any two nested discretizations $\mathcal{G}_n \subseteq \mathcal{G}_{n'}$ of the same manifold M with $n \leq n'$ (e.g., obtained by mesh decimation), the attention kernel values at common nodes $x_i, x_j \in \mathcal{G}_n$ satisfy*

$$\left| \langle \tilde{\phi}_i^n, \tilde{\phi}_j^n \rangle - \langle \tilde{\phi}_i^{n'}, \tilde{\phi}_j^{n'} \rangle \right| = \mathcal{O}\left(n^{-1/(m+4)}\right) + \mathcal{O}\left(r^{-1/2}\right),$$

where $\tilde{\phi}_i^n$ is the i-th row of $\widetilde{\Phi}^n = f(P_n) \cdot R^n$. The two sources combine additively via the triangle inequality: the first is the discretization mismatch error, the second the statistical estimation error from the random embedding.

(iii) ***(Gauge invariance.)*** *GIST's attention kernel $K(i,j) = [f(P)f(P)^\top]_{ij}$ is exactly invariant under all gauge transformations of the eigenvector basis (sign flips, rotations within degenerate eigenspaces, solver choices). Combined with (ii), this ensures that learned parameters transfer across discretizations (discretization-invariance): the attention weights converge to a gauge-invariant continuum limit that does not depend on the arbitrary eigenvector conventions of any particular discretization.*

The proof proceeds in three stages corresponding to the key technical components.

A.1.1 Stage 1: spectral convergence and Green's function

Proposition A.2. *Let $\mathcal{L}_n$ be the (appropriately rescaled) graph Laplacian of a graph $\mathcal{G}_n$ obtained by sampling n nodes uniformly from a compact m-dimensional Riemannian manifold M, with the bandwidth normalization required by Calder and García Trillos [55] so that $\mathcal{L}_n$ converges spectrally to the (unbounded) Laplace-Beltrami operator Δ_M. Let λ_k^n, u_k^n be eigenvalues and eigenvectors of $\mathcal{L}_n$ in this normalization, and μ_k, ψ_k the eigenvalues and eigenfunctions of Δ_M on M. Then:*

(a) Spectral convergence: $|\lambda_k^n - \mu_k| = \mathcal{O}(n^{-1/(m+4)})$ and $\|u_k^n - \psi_k\|_{L^2} = \mathcal{O}(n^{-1/(m+4)})$ (up to log factors).

(b) Green's function convergence: the discrete Green's function on $\mathcal{G}_n$ (the inner product of Laplacian eigenmaps, see (2)*) converges to the continuum Green's function:*

$$\sum_{\lambda_k^n>0} \frac{1}{\lambda_k^n}(u_k^n)_i(u_k^n)_j = G_M(x_i,x_j) + \mathcal{O}(n^{-1/(m+4)}),$$

where $G_M(x_i,x_j) = \sum_{k=1}^{\infty} \frac{1}{\mu_k}\psi_k(x_i)\psi_k(x_j)$ is the Green's function of Δ_M on M.

Proof. Part (a) follows from the spectral convergence theory for graph Laplacians on manifolds [55]: with appropriate graph construction, both eigenvalues and eigenvectors of $\mathcal{L}_n$ converge to those of Δ_M at rate $n^{-1/(m+4)}$ (up to log factors). Part (b) follows from part (a) by termwise convergence of the eigenfunction expansion of the pseudoinverse of $\mathcal{L}_n$. □

Corollary A.3 (Kernel convergence for general spectral filters)**.** *Let $\mu_1 > 0$ be the spectral gap of Δ_M, and let f be a spectral filter satisfying (a) $g(\lambda) := f(\lambda)^2$ is Lipschitz on $[\mu_1,\infty)$ with Lipschitz constant L_g, and (b) $\sum_l f(\mu_l)^2 < \infty$ (trace-class). Then the discrete kernel*

$$K_n(i,j) := \sum_{\lambda_l^n>0} f(\lambda_l^n)^2\,(u_l^n)_i(u_l^n)_j$$

(projecting out the kernel of $\mathcal{L}_n$; equivalent to $[f(P_n)f(P_n)^\top]_{ij}$ when f has no pole at 0) converges to the continuum kernel $K_\infty(x_i,x_j) := \sum_l f(\mu_l)^2\,\psi_l(x_i)\psi_l(x_j)$ at rate $\mathcal{O}(n^{-1/(m+4)})$.

Proof. By Proposition A.2(a), $|\lambda_k^n - \mu_k| = \mathcal{O}(n^{-1/(m+4)})$ and $\|u_k^n - \psi_k\|_{L^2} = \mathcal{O}(n^{-1/(m+4)})$ for each fixed k. The Lipschitz condition on g over $[\mu_1,\infty)$ gives $|g(\lambda_k^n) - g(\mu_k)| \le L_g|\lambda_k^n - \mu_k| = \mathcal{O}(n^{-1/(m+4)})$ for $\lambda_k^n, \mu_k \ge \mu_1$. Combined with the eigenvector convergence (each term in the sum carries eigenvalue and eigenvector errors at the same rate), this yields per-term convergence at rate $\mathcal{O}(n^{-1/(m+4)})$. The trace-class assumption $\sum_l f(\mu_l)^2 < \infty$ provides the uniform tail control needed to upgrade per-term convergence to convergence of the full sum at the same rate.

The Laplacian-eigenmap case ($f(\lambda) = \lambda^{-1/2}$) falls outside the Lipschitz hypothesis because $g(\lambda) = 1/\lambda$ has a pole at $\lambda = 0$; that case is handled directly by Proposition A.2(b), where the kernel of $\mathcal{L}_n$ is projected out and K_∞ specializes to the Green's function G_M. □

Remark A.4 (Rescaled variants of $\mathcal{L}_n$)*.* Corollary A.3 extends to any operator P_n obtained from $\mathcal{L}_n$ by a smooth eigenvalue transformation: the spectrum of P_n is then a smooth reparameterization of $\sigma(\mathcal{L}_n)$, and the eigenvalue and eigenvector convergence of Proposition A.2 transfers at the same rate $\mathcal{O}(n^{-1/(m+4)})$ up to bounded similarity factors. A smooth filter f on P_n is equivalently a smooth filter on $\mathcal{L}_n$, so the Lipschitz hypothesis of Corollary A.3 is preserved. Our FastRP implementation (Algorithm 1) is the case $P_n = D^{-1}A_n = \mathbb{1} - \mathcal{L}_{\mathrm{rw},n}$ with $f(P) = \sum_{i=1}^{k} P^i$, both smooth.

A.1.2 Stage 2: spectral embeddings as kernel estimators

We consider the general class of approximate spectral embeddings $\widetilde{\Phi} = f(P)\cdot R \in \mathbb{R}^{N\times r}$, where P is a graph operator, f is a spectral filter (see §3 for definitions), and $R \in \mathbb{R}^{N\times r}$ is a random matrix with $\mathbb{E}[RR^\top] = \mathbb{1}$. We write $\phi_i = f(P)_{i\cdot}^\top \in \mathbb{R}^N$ for the i-th row of $f(P)$ (the "exact" spectral embedding of node i) and $\tilde{\phi}_i \in \mathbb{R}^r$ for the i-th row of $\widetilde{\Phi}$ (its random projection); Laplacian eigenmaps (2) are the special case $f(\lambda) = \lambda^{-1/2}$, $P = \mathcal{L}$. The node-indexed ϕ_i used here differs vectorwise from the eigenvalue-indexed coordinates $(\phi_i)_k = \lambda_k^{-1/2}(u_k)_i$ of (2) (the two are related by an orthogonal change of basis), but they agree at the inner-product level, so the kernel statements below are independent of which representation is used.

Lemma A.5 (Kernel estimator)**.** *For any spectral filter f and random matrix $R \in \mathbb{R}^{N\times r}$ with $\mathbb{E}[RR^\top] = \mathbb{1}$, the inner products of $\widetilde{\Phi} = f(P)\cdot R$ satisfy*

$$\mathbb{E}\Big[\langle\tilde{\phi}_i,\tilde{\phi}_j\rangle\Big] = K(i,j) := [f(P)f(P)^\top]_{ij}.$$

When P is symmetric with eigenvectors u_l, this reduces to $K(i,j) = \sum_l f(\lambda_l)^2\,(u_l)_i(u_l)_j$.

Proof. $\mathbb{E}[\widetilde{\Phi}\,\widetilde{\Phi}^\top] = f(P)\,\mathbb{E}[RR^\top]\,f(P)^\top = f(P)\cdot\mathbb{1}\cdot f(P)^\top = f(P)f(P)^\top$. Taking the (i,j) entry gives the result. □

Proposition A.6 (Exact gauge invariance of K)**.** *The kernel $K(i,j) = [f(P)f(P)^\top]_{ij}$ is exactly invariant under all gauge transformations of the eigenvector basis: sign flips $u_l \to s_l u_l$, orthogonal rotations within degenerate eigenspaces, and different numerical solver choices.*

Proof. $K(i,j) = [f(P)f(P)^\top]_{ij}$ is an entry of the matrix $f(P)f(P)^\top$, which is a function of P alone and does not depend on the choice of eigenvector basis. All gauge transformations leave P unchanged, so K is unchanged. □

Lemma A.7 (Concentration bound)**.** *Let $\mathbf{a} = f(P)_{i\cdot}^\top$ and $\mathbf{b} = f(P)_{j\cdot}^\top$ be the i-th and j-th rows of $f(P)$, and let $R \in \mathbb{R}^{N\times r}$ have i.i.d. sub-Gaussian entries with mean zero and variance $1/r$. Then:*

(i) Tail bound. *By the Hanson-Wright inequality [76], for any $t > 0$:*

$$\Pr\Big[\Big|\langle\tilde{\phi}_i,\tilde{\phi}_j\rangle - K(i,j)\Big| > t\Big] \le 2\exp\!\left(-c\cdot r\cdot\min\!\left(\frac{t^2}{\|\mathbf{a}\|^2\|\mathbf{b}\|^2}, \frac{t}{\|\mathbf{a}\|\|\mathbf{b}\|}\right)\right),$$

where $c > 0$ is an absolute constant depending only on the sub-Gaussian norm of the entries of R.

(ii) Scaling of r. *Fix a target accuracy $\varepsilon > 0$ and a failure probability $\delta \in (0,1)$. Setting $t = \varepsilon\|\mathbf{a}\|\|\mathbf{b}\|$ in the bound above, and noting that for $\varepsilon \le 1$ the minimum is achieved by the quadratic term $t^2/(\|\mathbf{a}\|^2\|\mathbf{b}\|^2) = \varepsilon^2$, we require*

$$2\exp(-c\cdot r\cdot\varepsilon^2) \le \delta \qquad\Longleftrightarrow\qquad r \ge \frac{\log(2/\delta)}{c\,\varepsilon^2}.$$

Therefore $r = \mathcal{O}(\varepsilon^{-2}\log(1/\delta))$ random-projection dimensions suffice to guarantee

$$\Big|\langle\tilde{\phi}_i,\tilde{\phi}_j\rangle - K(i,j)\Big| \le \varepsilon\,\|\mathbf{a}\|\|\mathbf{b}\|$$

with probability at least $1-\delta$. Equivalently, for fixed δ, the estimation error is $\mathcal{O}(\|\mathbf{a}\|\|\mathbf{b}\|/\sqrt{r})$.

Proof. Write $R = [\mathbf{r}_1\ \cdots\ \mathbf{r}_r]$ where each column $\mathbf{r}_s \in \mathbb{R}^N$ has i.i.d. sub-Gaussian entries with mean zero and variance $1/r$. Then

$$\langle\tilde{\phi}_i,\tilde{\phi}_j\rangle = \sum_{s=1}^{r}(\mathbf{a}^\top\mathbf{r}_s)(\mathbf{b}^\top\mathbf{r}_s) = \mathbf{a}^\top\left(\sum_{s=1}^{r}\mathbf{r}_s\mathbf{r}_s^\top\right)\mathbf{b} = \mathbf{a}^\top(RR^\top)\mathbf{b}.$$

Each summand $(\mathbf{a}^\top\mathbf{r}_s)(\mathbf{b}^\top\mathbf{r}_s)$ is independent with mean $\mathbb{E}[\mathbf{a}^\top\mathbf{r}_s\mathbf{r}_s^\top\mathbf{b}] = (1/r)\,\mathbf{a}^\top\mathbf{b} = (1/r)\,K(i,j)$, so $\mathbb{E}[\langle\tilde{\phi}_i,\tilde{\phi}_j\rangle] = K(i,j)$. The product $(\mathbf{a}^\top\mathbf{r}_s)(\mathbf{b}^\top\mathbf{r}_s)$ can be written as $\mathbf{r}_s^\top(\mathbf{a}\mathbf{b}^\top)\mathbf{r}_s$, a quadratic form in a sub-Gaussian random vector; symmetrizing the (asymmetric) form $\mathbf{a}\mathbf{b}^\top$ to $M := \frac{1}{2}(\mathbf{a}\mathbf{b}^\top + \mathbf{b}\mathbf{a}^\top)$ leaves the value of the quadratic form unchanged and gives matrix norms $\|M\|_F \le \|\mathbf{a}\|\|\mathbf{b}\|$ and $\|M\|_{\mathrm{op}} \le \|\mathbf{a}\|\|\mathbf{b}\|$. Summing r such terms and applying the Hanson-Wright inequality [76] to $\langle\tilde{\phi}_i,\tilde{\phi}_j\rangle - K(i,j) = \mathbf{a}^\top(RR^\top - \mathbb{1})\mathbf{b}$ yields the tail bound in part (i). Part (ii) follows by inverting this bound. □

A.1.3 Stage 3: gauge invariance of GIST

GIST computes attention weights via linear attention with feature map $\varphi = \mathrm{ReLU}$ (as in Algorithm 2): the effective logits are $e_{ij} = \langle\varphi(\tilde{\phi}_i), \varphi(\tilde{\phi}_j)\rangle$. The gauge invariance of e_{ij} rests on a sample-wise observation: for any fixed realization of the random matrix R, the embedding $\tilde{\phi}_i = [f(P)R]_i$ is a deterministic function of P (computed without any eigendecomposition), so gauge transformations of the eigenvector basis, which leave P unchanged, leave $\tilde{\phi}_i$, $\varphi(\tilde{\phi}_i)$, and e_{ij} exactly unchanged. The argument is distribution-free and applies uniformly to any random-projection scheme of the form (3). In expectation, $\mathbb{E}[\langle\tilde{\phi}_i,\tilde{\phi}_j\rangle] = K(i,j) = [f(P)f(P)^\top]_{ij}$ (Lemma A.5), tying the population-level attention to the gauge-invariant kernel K (Proposition A.6). This establishes part (iii) of Proposition A.1.

A.1.4 Discretization mismatch error analysis

Combining Stages 1 and 2, for two nested discretizations $\mathcal{G}_n \subseteq \mathcal{G}_{n'}$ of the same manifold M and any pair of common nodes $i, j \in \mathcal{G}_n$, the attention kernel mismatch is bounded by the triangle inequality:

$$\left|\langle\tilde{\phi}_i^n, \tilde{\phi}_j^n\rangle - \langle\tilde{\phi}_i^{n'}, \tilde{\phi}_j^{n'}\rangle\right| \leq \underbrace{\left|\langle\tilde{\phi}_i^n, \tilde{\phi}_j^n\rangle - K_n(i,j)\right|}_{\mathcal{O}(1/\sqrt{r}) \text{ by Lemma A.7}} + \underbrace{|K_n(i,j) - K_{n'}(i,j)|}_{\mathcal{O}(n^{-1/(m+4)}) \text{ by Corollary A.3}} + \underbrace{\left|K_{n'}(i,j) - \langle\tilde{\phi}_i^{n'}, \tilde{\phi}_j^{n'}\rangle\right|}_{\mathcal{O}(1/\sqrt{r}) \text{ by Lemma A.7}},$$

where $K_n(i,j) = [f(P_n)f(P_n)^\top]_{ij}$ denotes the kernel on discretization $\mathcal{G}_n$. The middle term follows from Corollary A.3: both K_n and $K_{n'}$ converge to the same continuum kernel $K_\infty(x_i, x_j) = \sum_l f(\mu_l)^2\psi_l(x_i)\psi_l(x_j)$ at rate $\mathcal{O}(n^{-1/(m+4)})$, so by the triangle inequality $|K_n(i,j) - K_{n'}(i,j)| = \mathcal{O}(n^{-1/(m+4)})$. The total bound is therefore

$$\left|\langle\tilde{\phi}_i^n, \tilde{\phi}_j^n\rangle - \langle\tilde{\phi}_i^{n'}, \tilde{\phi}_j^{n'}\rangle\right| = \mathcal{O}\left(\frac{1}{\sqrt{r}}\right) + \mathcal{O}\left(n^{-1/(m+4)}\right), \tag{5}$$

with the two error sources combining additively via the triangle inequality. Choosing $r = \mathcal{O}(\varepsilon^{-2})$ and $n = \mathcal{O}(\varepsilon^{-(m+4)})$ achieves total error ε.

A.1.5 From kernel mismatch to output mismatch

The analysis above bounds the gauge-invariant attention kernel. The following corollary propagates that bound through the linear-attention normalization of a gauge-invariant spectral self-attention block, completing the chain from part (ii) of Proposition A.1 to a discretization-mismatch bound on the corresponding output.

Corollary A.8 (Output mismatch for a gauge-invariant attention block)**.** *Consider a gauge-invariant spectral self-attention block of GIST applied to node features x_i, with shared spectral embeddings $\tilde{\phi}_i$ at the input. Assume:*

- *(a)* Bounded embeddings: $\sup_{n,i} \|f(P_n)_{i\cdot}\| \leq \rho$ *and* $\|\tilde{\phi}_i^n\| \leq \rho$ *(the former controls the* $\|\mathbf{a}\|\|\mathbf{b}\|$ *scaling in Lemma A.7).*
- *(b)* Bounded features: $\|x_i\|, \|v_j\| \leq B$*, and node features agree at common nodes across nested discretizations.*
- *(c)* Bounded weights: *all learnable weight matrices have spectral norm at most* w*.*
- *(d)* Non-degenerate normalizer: $\inf_{n,i} \frac{1}{N}\sum_j \varphi(\tilde{\phi}_i^n)^\top\varphi(\tilde{\phi}_j^n) \geq Z_* > 0$ *(the average attention weight is bounded away from zero). With $q_i = k_i = \tilde{\phi}_i$ this implies the linear-attention normalizer $Z_i^n = \sum_j \varphi(q_i)^\top\varphi(k_j) \geq N\, Z_*$, scaling linearly with the discretization.*
- *(e)* Lipschitz feature map: φ *is* Lip_φ*-Lipschitz on the ball of radius* ρ*.*
- *(f)* Lipschitz Gram regularity of the population kernel: $\bar{k}(i,j) := \mathbb{E}_R[\langle\varphi(\tilde{\phi}_i), \varphi(\tilde{\phi}_j)\rangle]$ *depends Lipschitz-continuously on the Gram quantities* $(\|f(P)_{i\cdot}\|^2, \|f(P)_{j\cdot}\|^2, \langle f(P)_{i\cdot}, f(P)_{j\cdot}\rangle)$ *on the bounded set defined by (a).*

Let o_i^n denote the output at node i on discretization $\mathcal{G}_n$. Then, for any two nested discretizations $\mathcal{G}_n \subseteq \mathcal{G}_{n'}$ of the same manifold M as in Proposition A.1 and any common node $i \in \mathcal{G}_n$,

$$\left\|o_i^n - o_i^{n'}\right\| \;=\; C \cdot \left[\mathcal{O}\left(n^{-1/(m+4)}\right) + \mathcal{O}\left(r^{-1/2}\right)\right],$$

where the constant C depends polynomially on ρ, B, w, Z_*^{-1} and Lip_φ but not on n or r.

Proof sketch. (1) From kernel mismatch to raw-weight mismatch. The raw linear-attention weights of the gauge-invariant block are $e_{ij} = \langle\varphi(\tilde{\phi}_i), \varphi(\tilde{\phi}_j)\rangle$ (the "effective logits" of Stage 3), with normalizer $Z_i = \sum_j e_{ij}$. The kernel-mismatch bound (5) (instantiated with $\|\mathbf{a}\|, \|\mathbf{b}\| \leq \rho$) controls $|\langle\tilde{\phi}_i^n, \tilde{\phi}_j^n\rangle - \langle\tilde{\phi}_i^{n'}, \tilde{\phi}_j^{n'}\rangle|$ uniformly in i, j, and the diagonal case $i = j$ controls $|\|\tilde{\phi}_i^n\|^2 - \|\tilde{\phi}_i^{n'}\|^2|$. By assumption (f), the population kernel $\bar{k}(i,j) := \mathbb{E}_R[e_{ij}]$ is Lipschitz in these Gram quantities, so

$|\bar{k}_n(i,j) - \bar{k}_{n'}(i,j)| = \mathcal{O}(n^{-1/(m+4)})$. Combined with $\mathcal{O}(r^{-1/2})$ concentration of e_{ij} around $\bar{k}(i,j)$ (Hanson-Wright on the bounded set), the triangle inequality yields $|e_{ij}^n - e_{ij}^{n'}| = \mathcal{O}(n^{-1/(m+4)}) + \mathcal{O}(r^{-1/2})$.

(2) From raw-weight mismatch to output mismatch. The values are $v_j = W_v x_j$, gauge- and discretization-independent at common nodes by (b) and (c). The map $\{e_{ij}\}_j \mapsto o_i = \sum_j e_{ij} v_j / Z_i$ is Lipschitz under (b) and (d), with the constant absorbing the linear scaling $Z_i \geq N Z_*$ of the normalizer guaranteed by (d) so that the per-pair perturbation in step (1) does not accumulate with N; the output mismatch then inherits the rate of step (1). Linear projections, Lipschitz nonlinearities, and residual paths downstream of attention preserve the rate under (c). □

Assumptions (a)–(c) are standard regularity conditions for trained transformer blocks; (d) is generic for ReLU on non-trivial embeddings; (f) is satisfied by Gaussian random projections, where $\bar{k}$ is the first-order arc-cosine kernel [53].

Remark A.9 (Refinements of the bound). Three technical points sharpen the statement above without changing the leading-order rate:

(i) Excluded filters. Assumption (a) requires a uniformly bounded discrete kernel diagonal $K_n(i,i) \leq \rho^2$ and therefore excludes filters whose continuum kernel diverges on the diagonal, notably $f(\lambda) = \lambda^{-1/2}$ on manifolds with $m \geq 2$ (where $K_\infty(x_i, x_i) = G_M(x_i, x_i)$ has the standard logarithmic ($m = 2$) or $|x_i - x_j|^{2-m}$ ($m \geq 3$) Green's-function singularity). Trace-class filters with bounded continuum diagonal, such as heat kernels $f(\lambda) = e^{-t\lambda}$ and bounded polynomial filters (including the FastRP filter $f(P) = \sum_{i=1}^k P^i$), satisfy (a).

(ii) Uniform statistical error. The per-pair concentration of Lemma A.7 extends to all $\binom{n}{2}$ node pairs via a union bound, replacing $\mathcal{O}(r^{-1/2})$ by $\mathcal{O}(\sqrt{\log n / r})$.

(iii) Spatial quadrature. The discrete output is a Monte-Carlo approximation of the underlying continuum operator integral; for i.i.d. uniform sampling the resulting quadrature error is $\mathcal{O}(n^{-1/2})$, dominated by the spectral rate $\mathcal{O}(n^{-1/(m+4)})$ and absorbed into it.

B Algorithms

Note that in the pseudocode we use bold notation for matrices and vectors ($\mathbf{A}, \mathbf{\Phi}, \mathbf{Q}$) and follow the row-vector convention standard in machine learning: $\mathbf{\Phi} \in \mathbb{R}^{N \times r}$ has nodes as rows and embedding dimensions as columns. In the main text, we use non-bold notation for compactness, with $\phi_i \in \mathbb{R}^r$ denoting the i-th node embedding (the i-th row of $\mathbf{\Phi}$, treated as a column vector when forming inner products) and upper-case characters denoting matrices. The symbol $\mathbf{1}_N$ denotes the all-ones column vector in $\mathbb{R}^N$.

Algorithm 1 Spectral embeddings via FastRP [50]

Require: Graph adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, embedding dimensionality r, iteration power k
Ensure: Matrix of N node graph positional embeddings $\mathbf{\Phi} \in \mathbb{R}^{N \times r}$

Produce very sparse random projection $\mathbf{R} \in \mathbb{R}^{N \times r}$ according to Li et al. [77]
$\mathbf{P} \leftarrow \mathbf{D}^{-1/2} \cdot \mathbf{A} \cdot \mathbf{D}^{-1/2}$ the normalized adjacency matrix, where $\mathbf{D}$ is the degree matrix
$\mathbf{\Phi_1} \leftarrow \mathbf{P} \cdot \mathbf{R}$
for $i = 2$ to k **do**
    $\mathbf{\Phi}_i \leftarrow \mathbf{P} \cdot \mathbf{\Phi}_{i-1}$
end for
$\mathbf{\Phi} = \mathbf{\Phi}_1 + \mathbf{\Phi}_2 + \cdots + \mathbf{\Phi}_k$
return $\mathbf{\Phi}$

Below we provide pseudo-code for the core computations of GIST, the *Gauge-Invariant Spectral Self-Attention* block and the *Gauge-Equivariant Spectral Self-Attention* block. For illustration purposes, we compare the algorithms to a stripped down implementation of self-attention. Modifications relative to vanilla self-attention are highlighted in red (additions) and ~~strikethrough~~ (removals).

Algorithm 2 Gauge-Invariant Spectral Self-Attention

Require: Node feature tokens $\mathbf{X} \in \mathbb{R}^{N \times d}$, graph positional embeddings $\mathbf{\Phi} \in \mathbb{R}^{N \times r}$
Ensure: Output sequence $\mathbf{O} \in \mathbb{R}^{N \times d}$ to be applied to features $\mathbf{X}$

// Compute attention matrices
~~$\mathbf{Q} \leftarrow \mathbf{X} \cdot \mathbf{W}_Q$ where $\mathbf{W}_Q \in \mathbb{R}^{d \times d}$~~ $\mathbf{Q} \leftarrow \mathbf{\Phi}$
~~$\mathbf{K} \leftarrow \mathbf{X} \cdot \mathbf{W}_K$ where $\mathbf{W}_K \in \mathbb{R}^{d \times d}$~~ $\mathbf{K} \leftarrow \mathbf{\Phi}$
$\mathbf{V} \leftarrow \mathbf{X} \cdot \mathbf{W}_V$ where $\mathbf{W}_V \in \mathbb{R}^{d \times d}$
// Compute linear attention with feature map $\varphi(x) = \mathrm{ReLU}(x - \theta)$, with trainable scalar θ
$\tilde{\mathbf{Q}}, \tilde{\mathbf{K}} \leftarrow \varphi(\mathbf{Q}), \varphi(\mathbf{K})$
$\mathbf{S} \leftarrow \tilde{\mathbf{K}}^T \mathbf{V}$ ▷ Compute key-value matrix: $\mathbb{R}^{r \times d}$
$\mathbf{Z} \leftarrow 1/(\tilde{\mathbf{Q}}(\tilde{\mathbf{K}}^T \mathbf{1}_N) + \epsilon)$ ▷ Normalization factors: $\mathbb{R}^N$
$\mathbf{O} \leftarrow (\tilde{\mathbf{Q}}\mathbf{S}) \odot \mathbf{Z}$ ▷ Normalized output: element-wise product
return $\mathbf{O}$

Algorithm 3 Gauge-Equivariant Spectral Self-Attention

Require: Node feature tokens $\mathbf{X} \in \mathbb{R}^{N \times d}$, graph positional embeddings $\mathbf{\Phi} \in \mathbb{R}^{N \times r}$
Ensure: Output sequence $\mathbf{O} \in \mathbb{R}^{N \times r}$ to be applied to graph positional embeddings $\mathbf{\Phi}$

// Compute attention matrices
$\mathbf{Q} \leftarrow \mathbf{X} \cdot \mathbf{W}_Q$ where $\mathbf{W}_Q \in \mathbb{R}^{d \times d}$
$\mathbf{K} \leftarrow \mathbf{X} \cdot \mathbf{W}_K$ where $\mathbf{W}_K \in \mathbb{R}^{d \times d}$
~~$\mathbf{V} \leftarrow \mathbf{X} \cdot \mathbf{W}_V$ where $\mathbf{W}_V \in \mathbb{R}^{d \times d}$~~ $\mathbf{V} \leftarrow \mathbf{\Phi}$
// Compute linear attention [24]
$\tilde{\mathbf{Q}}, \tilde{\mathbf{K}} \leftarrow \varphi(\mathbf{Q}), \varphi(\mathbf{K})$
$\mathbf{S} \leftarrow \tilde{\mathbf{K}}^T \mathbf{V}$ ▷ Compute key-value matrix: $\mathbb{R}^{d \times r}$
$\mathbf{Z} \leftarrow 1/(\tilde{\mathbf{Q}}(\tilde{\mathbf{K}}^T \mathbf{1}_N) + \epsilon)$ ▷ Normalization factors: $\mathbb{R}^N$
$\mathbf{O} \leftarrow (\tilde{\mathbf{Q}}\mathbf{S}) \odot \mathbf{Z}$ ▷ Normalized output: element-wise product
return $\mathbf{O}$

Note on the choice of feature map φ. While in Algorithm 3 we do not need to impose that restriction, in Algorithm 2 we use the feature map $\varphi(x) = \mathrm{ReLU}(x - \theta)$ with a trainable scalar θ, chosen for its simplicity, non-negativity (yielding valid attention weights in the linear-attention formulation), and Lipschitz continuity. ReLU also has a well-known correspondence with the first-order arc-cosine kernel [53]: for Gaussian random features $\mathbf{a}, \mathbf{b} \in \mathbb{R}^r$, the inner product $\langle \varphi(\mathbf{a}), \varphi(\mathbf{b}) \rangle$ converges (as $r \to \infty$) to $k_1(\mathbf{a}, \mathbf{b}) = \frac{1}{\pi}\|\mathbf{a}\|\|\mathbf{b}\|[\sin\vartheta + (\pi - \vartheta)\cos\vartheta]$, where ϑ is the angle between $\mathbf{a}$ and $\mathbf{b}$, a function of the Gram quantities $(\|\mathbf{a}\|, \|\mathbf{b}\|, \mathbf{a}^\top \mathbf{b})$.

The gauge invariance of GIST's attention weights is established at the sample level (Stage 3 of the proof of Proposition A.1): for any fixed realization of R, $\tilde{\phi}_i = [f(P)R]_i$ is a deterministic function of P, so any element-wise feature map yields attention weights that are exactly invariant under all gauge transformations of the eigenvector basis. The argument therefore applies to ReLU and to any other element-wise feature map.

The original linear attention work by Katharopoulos et al. [24] used $\varphi(x) = \mathrm{elu}(x) + 1$, which is also non-negative and works well in practice; our framework accommodates either choice. Investigating alternative feature maps (e.g., polynomial kernels, random Fourier features for RBF-like kernels) is left for future work.

C Additional empirical studies

C.1 Training details

GIST models are trained with the AdamW optimizer, weight decay between zero and 5×10^{-4}, and a base learning rate between 2×10^{-5} and 2×10^{-3}, with linear warmup followed by linear decay. These choices are selected on the validation split as part of the per-benchmark hyperparameter optimization (Section C.2). Loss functions follow the task: cross-entropy for single-label node

classification (Cora, CiteSeer, PubMed, Photo, ogbn-arxiv), binary cross-entropy with logits for the multi-label PPI benchmark, and mean squared error for surface-pressure regression on AirfRANS, ShapeNet-Car, DrivAerNet, and DrivAerNet++.

The GIST architectures used in our experiments stack two to three multi-scale blocks with hidden dimension $d \in \{128, 256, 512\}$; spectral-embedding hyperparameters (r, k) are tuned within the robust ranges identified in Section C.2. Spectral embeddings are precomputed once per graph and cached, so their cost is amortized across training and is not included in the per-step training time. Single-graph datasets are trained full-graph where memory permits; PPI and mesh datasets are trained in mini-batches sized to the per-graph memory budget.

Table 5 lists the per-benchmark configurations and parameter counts; the corresponding configuration files are included with the source code at `https://github.com/ibm/gist-neurips-2026`.

Table 5: GIST configurations of the released code. L: number of multi-scale blocks; d: hidden dimension; d_b: width of the feature and local branches; (r, k): FastRP embedding dimension and number of power iterations; θ_0: initial value of the trainable threshold θ; Drop.: dropout; LR: learning rate; WD: weight decay; Warm.: warmup epochs; Batch: graphs per batch. All models use 4 attention heads and MLP ratio 2. Baseline results are taken from the cited publications, with the configurations reported there.

Dataset	Params	L	d	d_b	r	k	θ_0	Drop.	LR	WD	Epochs	Warm.	Batch
Cora	6.55M	2	256	256	512	128	2.0	0.8	1e-3	5e-4	100	1	1
CiteSeer	17.13M	2	512	256	512	128	2.0	0.8	3e-4	5e-4	100	1	1
PubMed	3.94M	2	256	128	512	256	1.0	0.8	1e-4	5e-4	1000	10	1
Photo	4.01M	2	256	128	512	256	2.0	0.7	1e-3	0	500	5	1
ogbn-arxiv	6.22M	2	256	256	512	256	1.0	0.7	2e-3	0	1000	10	1
PPI	5.83M	3	256	128	256	64	1.0	0.3	2.5e-4	0	1000	10	2
AirfRANS	6.18M	2	256	256	256	64	1.0	0.0	3e-5	0	300	0	1
ShapeNet-Car	5.79M	3	256	128	256	64	1.0	0.0	1e-4	0	200	0	1
DrivAerNet	4.71M	3	256	64	256	64	1.5	0.0	2.5e-5	1e-5	100	2	1

C.2 GIST hyperparameter robustness

In order to study the sensitivity of GIST's performance to variations in its spectral embedding hyperparameters, we train multiple simplified GIST architectures (two-block Gauge-Invariant Spectral Self-Attention linear transformers) on the Cora benchmark while varying the power iteration parameter k and the embedding dimension r of the spectral embedding. These parameters directly control the quality of spectral embeddings while balancing computational efficiency.

As shown in Figure 2, GIST exhibits robust performance across a wide range of both parameters. The left panel sweeps k with r fixed, a clear but relatively shallow peak in accuracy around the optimal value of $k \approx 32$. This on one hand suggests that even modest iteration counts are sufficient to capture the essential spectral structure, but also indicates that the choice of the power iteration is quite robust. The right panel varies embedding dimension r with k fixed, demonstrating a smooth monotonic improvement as r increases. Crucially, saturation occurs relatively quickly: performance gains beyond $r = 256$ are marginal, validating our choice of reasonable embedding dimensions that maintain computational efficiency.

These results empirically validate two important properties: (1) GIST does not require extensive hyperparameter tuning around these spectral parameters, suggesting stable generalization; and (2) the linear end-to-end complexity achieved with modest k and r values is both computationally practical and empirically effective. Combined with the gauge-invariance guarantees that prevent dependence on arbitrary spectral choices, these hyperparameters provide a principled and empirically robust way to control the approximation quality of spectral embeddings without sacrificing scalability.

C.3 Convergence to the exact gauge-invariant kernel

Figure 3 checks the statistical term of Proposition 4.1 directly: on Cora, the arc-cosine attention weights estimated from finite-r spectral embeddings converge to their exact, gauge-invariant limit at the predicted $\mathcal{O}(r^{-1/2})$ rate, for both Gaussian projections and the sparse projections used by FastRP.

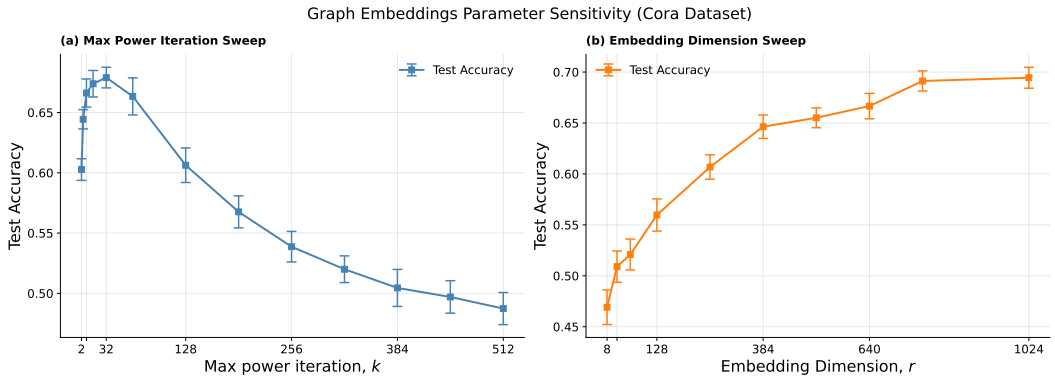

Figure 2: Sensitivity study of GIST spectral embeddings parameters. The plots show the final test accuracy of a two-block Gauge-Invariant Spectral Self-Attention linear transformer trained on Cora while sweeping over the power iteration parameter k with $r = 256$ (left panel), and sweeping over the embedding dimension r with $k = 32$ (right panel). Test accuracy is fairly robust around the best value of either parameter. As expected, r is monotonically related to higher performance, as higher r correspond to better approximations of the eigenmaps. Accuracy conveniently saturates relatively fast, justifying the use of reasonably low r. The plots show mean test accuracy averaged across 10 seeds and corresponding standard deviation as error bars.

The comparison isolates the kernel estimate. Both curves run parallel to the $\mathcal{O}(r^{-1/2})$ reference line, with the sparse projection slightly above the Gaussian one and the gap narrowing as r grows. At $r = 256$, the typical embedding dimension in our experiments, the relative error is about 2–3% for both projections.

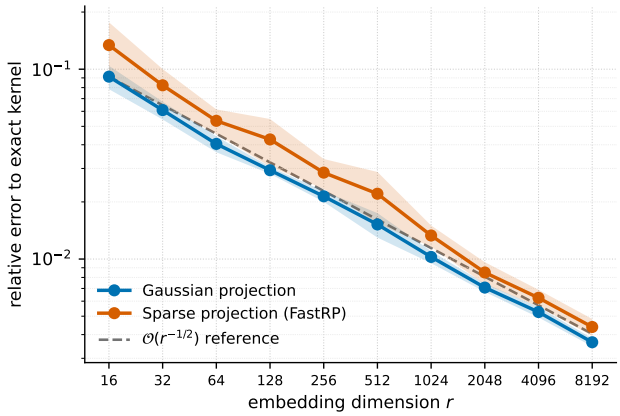

Figure 3: Convergence of the finite-r kernel estimate to the exact gauge-invariant kernel on Cora. For each r, row-normalized first-order arc-cosine attention weights are computed from the embeddings $\widetilde{\Phi} = f(P)R$ (FastRP filter, $k = 128$) and compared with the same weights computed from the exact rows of $f(P)$, the $r \to \infty$ limit; the y-axis is the relative Frobenius error. Both projections follow the $\mathcal{O}(r^{-1/2})$ rate of Lemma A.7. Shaded bands: ± 1 std over 10 projection seeds.

C.4 GIST scalability study

From a computational standpoint, the end-to-end cost of GIST is dominated by two components: spectral embedding generation and the subsequent transformer blocks. For the former, we employ a FastRP-style approximation in which the Laplacian spectral information is captured via repeated multiplication of a sparse random walk matrix with a low-dimensional random projection. Each power iteration requires $\mathcal{O}(|E|r)$ operations, where $|E|$ is the number of edges and r is the embedding dimension, and the total cost over k iterations is $\mathcal{O}(k|E|r)$. On meshes and graphs with bounded

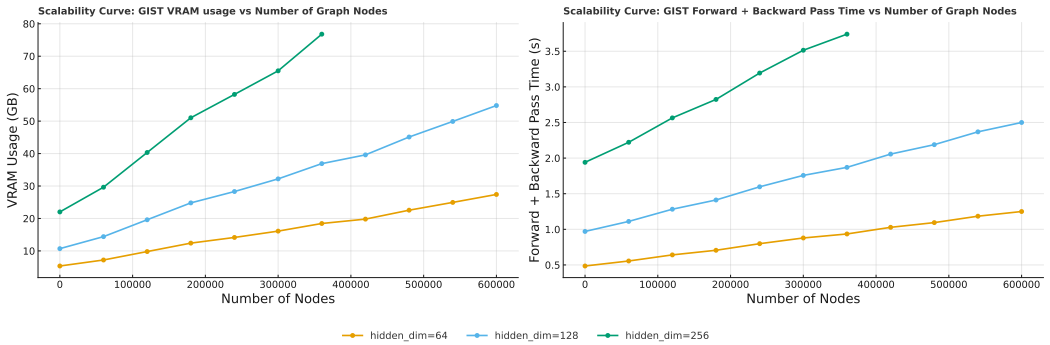

Figure 4: Scalability study of GIST. All experiments use a fixed 3-layer model while varying the hidden dimensionality. VRAM consumption was measured as a function of the number of nodes in the input graph. Graph sizes were controlled using random node dropout applied to samples from the DrivAerNet dataset, enabling a systematic evaluation of memory scaling behavior.

average degree, $|E| = \mathcal{O}(N)$, so the overall spectral embedding stage scales linearly in the number of nodes N. This embedding is computed once per graph and then reused across all GIST layers, so its cost is amortized over the full network depth.

The GIST layers themselves preserve this linear scaling. Each block combines: (i) a feature branch based on linear attention, (ii) a local branch using graph convolution followed by linear attention, and (iii) a global branch using Gauge-Invariant Spectral Self-Attention followed by linear attention, plus Gauge-Equivariant Spectral Self-Attention to refine the embeddings for the next block. In all cases, attention is implemented in the form $\varphi(Q)(\varphi(K)^\top V)$ with an element-wise feature map $\varphi(\cdot)$, which yields $\mathcal{O}(Nd^2)$ complexity for d-dimensional features instead of the $\mathcal{O}(N^2 d)$ cost of quadratic attention. Together with the $\mathcal{O}(N)$ spectral embedding stage, this results in an overall complexity of $\mathcal{O}(N(d^2 + rd + rk))$ per forward pass, i.e., linear in the number of nodes. The empirical VRAM and wall-clock measurements in Figure 4 corroborate this analysis: both memory usage and forward time grow approximately linearly with the number of graph nodes for all hidden dimensions, up to graphs with hundreds of thousands of nodes sampled from DrivAerNet.

Table 6 reports wall-clock times on DrivAerNet with the released code.

Table 6: GIST wall-clock times on DrivAerNet (NVIDIA H100 80GB; per-mesh times are mean $\pm$ std over 50 test meshes, mean 409K nodes). FastRP embeddings are computed once per mesh and cached; inference runs in bf16 with `torch.compile`. Peak GPU memory is 32.9 GiB per GPU for training and 3.3 GiB for FastRP and inference together.

Stage	GPUs	Time
Training (100 epochs)	4	3.4 h
FastRP embedding	1	1.30 ± 0.11 s / mesh
Inference (forward pass)	1	0.049 ± 0.003 s / mesh

C.5 Multi-scale architecture ablation study

To validate the design choices in the Multi-Scale GIST architecture, we systematically ablate each of the three parallel branches shown in Figure 1 (right panel) on the PPI dataset, where the full architecture achieves state-of-the-art performance (see Table 2). For each ablation, we train with identical hyperparameters but remove one branch: (1) feature processing, (2) local graph convolution, or (3) global spectral attention. These ablation experiments are repeated across 10 random seeds.

Table 7 shows that all three branches contribute meaningfully, with performance drops ranging from 4.29% to 7.90%. The local branch (Branch 2) has the strongest impact (-7.90%), validating the EfficientViT-inspired design principle that local operations provide focused information complementing the diffuse patterns from linear attention. The global spectral branch (Branch 3, -4.29%)

confirms that long-range dependencies are essential, while the feature branch (Branch 1, -5.04%) provides complementary signal beyond structural information. Overall, these results demonstrate that the Multi-Scale GIST effectively integrates complementary information sources.

Table 7: Ablation study on PPI dataset showing the contribution of each branch of the Multi-Scale GIST (see Figure 1, right panel). Test accuracy is reported as percentage of baseline performance. Results averaged over 10 seeds with standard deviations indicated as uncertainty intervals.

Ablation	Test Accuracy (% baseline)	Δ (%)
No ablation	100.0	0.0
Branch 1 (feature)	95.0 ± 2.8	-5.0
Branch 2 (local)	92.1 ± 1.8	-7.9
Branch 3 (global)	95.7 ± 3.7	-4.3

C.6 Pronunciation

GIST is pronounced with a soft G, like *gist* (the English word for the central point or essence of something), not with a hard G as in *gift*. Although insisting on a hard G won't necessarily be considered wrong. After all, that is just a different gauge choice.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction summarize the theoretical contribution (Section 4.1) and the empirical contributions (Section 4).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See the "Broader impact, limitations, and future work" paragraph at the end of Section 5.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Full statement, assumptions, and proof are given in Appendix A.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The architecture is specified in Section 3 and Appendix B, training details are in Appendix C.1, and all datasets are public benchmarks with cited splits; additional details are available in the released source code (`https://github.com/ibm/gist-neurips-2026`).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Source code, configurations and instructions to reproduce the main results are available at `https://github.com/ibm/gist-neurips-2026`. All datasets used in this work are publicly available through their respective benchmarks.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: See Appendix C.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Node-classification results (Table 2) and the ablation (Table 7) report mean $\pm$ standard deviation across 6 and 10 random seeds, and AirfRANS and ShapeNet-Car results (Table 3) across 3 seeds; the cross-resolution study (Table 1) reports standard deviation across 600 cars, and the per-mesh times (Table 6) across 50 test meshes. DrivAerNet and DrivAerNet++ results (Table 4) are single training runs, owing to the cost of training on meshes of up to 750K nodes.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Compute is reported for DrivAerNet (∼400K-node meshes) in Table 6: training takes 3.4 h on four NVIDIA H100 80GB GPUs (100 epochs), and per-mesh FastRP and inference take 1.30 s and 0.049 s on one H100, with peak memory 32.9 GiB per GPU for training and 3.3 GiB for inference.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses public benchmark datasets and involves no human subjects; we have reviewed the NeurIPS Code of Ethics and the work conforms with it.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Discussed in the closing paragraph of Section 5.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release pretrained generative models, scraped datasets, or other high-risk artefacts.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets and software libraries used are cited and used under their respective public licenses.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.

- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The released code (`https://github.com/ibm/gist-neurips-2026`) is documented in its repository and distributed under the Apache-2.0 license.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The work does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The work does not involve human-subjects research.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not used as a core component of the methodology.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.