Sketch-Augmented Features Improve Learning Long-Range Dependencies in Graph Neural Networks

Ryien Hosseini[1], **Filippo Simini**[2], **Venkatram Vishwanath**[2],
Rebecca Willett[1,3], **Henry Hoffmann**[1]
[1]University of Chicago [2]Argonne National Laboratory
[3] NSF-Simons National Institute for Theory and Mathematics in Biology
{ryien,willett,hankhoffmann}@uchicago.edu
{fsimini,venkat}@anl.gov

Abstract

Graph Neural Networks learn on graph-structured data by iteratively aggregating local neighborhood information. While this local message passing paradigm imparts a powerful inductive bias and exploits graph sparsity, it also yields three key challenges: (i) oversquashing of long-range information, (ii) oversmoothing of node representations, and (iii) limited expressive power. In this work we inject randomized global embeddings of node features, which we term *Sketched Random Features*, into standard GNNs, enabling them to efficiently capture long-range dependencies. The embeddings are unique, distance-sensitive, and topology-agnostic—properties which we analytically and empirically show alleviate the aforementioned limitations when injected into GNNs. Experimental results on real-world graph learning tasks confirm that this strategy consistently improves performance over baseline GNNs, offering both a standalone solution and a complementary enhancement to existing techniques such as graph positional encodings.

1 Introduction

Graph Neural Networks (GNNs) are a widely adopted approach for representation learning on graph-structured data. Standard GNN architectures employ a message-passing framework [26], wherein each node's features are iteratively propagated and aggregated along edges. Thus, GNNs leverage a strong topology-induced bias: node features are iteratively refined by their neighbors, enabling local interactions to shape final node or graph embeddings. Moreover, these localized computations naturally exploit graph sparsity, facilitating efficient training. Despite this success, three persistent challenges remain:

- **Oversquashing:** Signals from distant nodes are compressed into a fixed size vector when traversing multiple hops, hindering a GNN's ability to capture long-range interactions [2].
- **Oversmoothing:** As network depth grows, node representations converge exponentially to nearly identical values, eliminating meaningful distinctions [44, 56].
- **Limited Expressive Power:** Standard GNNs have expressive power no greater than the 1-dimensional Weisfeiler–Lehman heuristic [67], leading them to fail at distinguishing many non-isomorphic graphs or structurally distinct subgraphs [68].

To address these limitations, two primary lines of research have emerged. The first adapts transformers [65] for graph-structured data, either by combining message passing with dense attention layers [59, 70, 38, 46] or by replacing message passing altogether [39, 49]. This approach allows the model to attend to arbitrary node pairs regardless of geodesic distance, thus circumventing many limitations of message-passing GNNs. However, these methods typically incur $O(N^2)$ memory and computation cost in the number of nodes N and rely on carefully designed node-level signals (called *positional*

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

or *structural encodings*) to inject graph structure back into the model [39]. These encodings can be challenging to design, and their theoretical properties remain only partially understood [63].

A second line of work augments traditional GNNs with encodings—often inspired by transformers—to improve performance by enhancing expressiveness [53]. These include purely random node features that break symmetry and guarantee universal separation of non-isomorphic graphs [1, 62]. However, such unstructured noise can slow convergence or degrade learning in practice [7]. In contrast, *structural encodings* (e.g., spectral embeddings [23]) are topologically distance-sensitive but rely on deterministic, graph-based representations that are difficult to make both *unique* and *equivariant* under node permutation. This is because, unlike Euclidean spaces, graphs lack a canonical coordinate system that defines cardinal direction, creating ambiguities that the GNN must handle. Notably, both these approaches primarily focus on topology-derived encodings, and do not consider informative node features typically present in real-world graphs.

This observation motivates our approach: *instead of abandoning message passing or augmenting it solely with structural information, we propose leveraging node features themselves to address GNN limitations.* To this end, we introduce **Sketched Random Features (SRF)**, presented in Algorithm 1, a simple yet powerful method for injecting global, feature-distance-sensitive signals into GNNs. SRF augments each node's features with *sketches*, or randomized low-dimensional projections, of all node features in a kernel space, preserving feature similarity in expectation and breaking symmetry through randomization. This paper describes how sketched random features can augment feature representations in every layer of a message passing GNN to yield accurate predictions in a variety of settings. By the Johnson–Lindenstrauss lemma [35, 8], such projections preserve important distance relationships in feature space even in significantly reduced dimensions. We leverage these properties to construct node-feature representations that we show overcome the challenges above.

Consider a graph's node feature matrix $X \in \mathbb{R}^{N\times F}$, where the i-th column of X^T, $\mathbf{x}_i \in \mathbb{R}^F$ denotes the features of node i, and a positive-definite similarity function (kernel) $\kappa : \mathbb{R}^F \times \mathbb{R}^F \to \mathbb{R}$ defined between these features. Let $\mathcal{E} : \mathbb{R}^{N\times F} \to \mathbb{R}^{N\times D}$ denote a mapping that embeds X into a random feature space via feature map $\varphi(\cdot)$.

While *any* random kernel feature $\varphi(\mathbf{x}_i) \in \mathbb{R}^D$ (e.g., via random Fourier features [57]) already provides an unbiased approximation of similarity between node features, i.e., $\mathbb{E}\big[\varphi(\mathbf{x}_i)^\top \varphi(\mathbf{x}_j)\big] = \kappa(\mathbf{x}_i, \mathbf{x}_j)$, we go further by applying a *cross-node random projection* $\mathcal{S}^{(k)} \in \mathbb{R}^{N\times N}$. Let $\Phi \in \mathbb{R}^{N\times D}$ (with $(D \ll N)$ be the matrix of random kernel features, where the i-th row of Φ is $\varphi(\mathbf{x}_i)$. Multiplying $\mathcal{S}^{(k)}$ and Φ yields the **kernel sketch** $Z = \mathcal{S}^{(k)}\Phi$, where the i-th column of Z^T, denoted $\mathbf{z}_i$, is a linear combination of *all* node features in the kernel space. Crucially, these sketched embeddings $\{\mathbf{z}_i\}$ still preserve kernel relationships in expectation, i.e. for any i, j, $\mathbb{E}\big[\mathbf{z}_i^\top \mathbf{z}_j\big] = \kappa\big(\mathbf{x}_i, \mathbf{x}_j\big)$.

Algorithm 1 Sketched Feature GNN

Input $\mathcal{G} = (V, E)$; $X \in \mathbb{R}^{N\times F}$; k;L
$\Phi = \mathcal{E}(X)$
$Z = \mathcal{S}^{(k)}(\Phi)$
Initialize: $\mathbf{h}_i^{(0)} = \mathbf{x}_i$ for all $i \in V$
for layer $\ell = 0$ to $L-1$ **do**
  for each node $i \in V$ **do**
    $\tilde{\mathbf{h}}_i^{(\ell)} = [\mathbf{h}_i^{(\ell)} | \mathbf{z}_i]$
    $\mathbf{h}_i^{(\ell+1)} = f\left(\tilde{\mathbf{h}}_i^{(\ell)}, \{\tilde{\mathbf{h}}_j^{(\ell)} : j \in \mathcal{N}(i)\}\right)$
return $\mathbf{h}_i^{(L)}$

Interestingly, this sketched random feature matrix provides properties that, when injected into GNNs, *are surprisingly effective* in mitigating the aforementioned limitations of message-passing graph neural networks. In particular, these sketches:

- **are unique yet distance-sensitive.** Each sketch $\mathbf{z}_i$ is unique, breaking symmetries, while preserving distances in the node-feature space with high probability (Propositions 3.2 and 3.4). This property confers *universality* (enabling GNNs to distinguish non-isomorphic graphs) and counters *oversmoothing* by preventing embeddings from converging.
- **contain topology-agnostic cross-node information.** Since each $\mathbf{z}_i$ is formed by a linear combination of all node embeddings in the random feature space, it injects a *global* signal into the GNN (Proposition 3.3). This mitigates *oversquashing* by enabling immediate cross-node interactions, regardless of geodesic distance.
- **maintain equivariance in expectation.** With suitable random projections, the cross-node sketch preserves pairwise feature relationships under node permutation (Proposition 3.5).

In this work, we demonstrate analytically (Section 3) and empirically (Section 4) that these sketched random features provide the necessary global, distance-sensitive signals to overcome fundamental limitations of message-passing GNNs while maintaining computational efficiency. Unlike traditional uses of sketching for dimensionality reduction, our method leverages sketching as a mechanism for mixing global feature information across nodes, enabling efficient propagation of non-local signals without altering the graph topology. Experiments on real-world graph learning tasks demonstrate that SRF-augmented models often outperform baseline GNNs, offering a computationally efficient alternative to existing structural positional encodings. Moreover, by leveraging orthogonal information derived from node *features* rather than topology, our approach can be integrated with existing structural encoding methods to yield further performance improvements.

2 Background and Related Work

Preliminaries. We consider finite node-attributed graphs $\mathcal{G} = (V, E, X)$ with $|V| = N$ nodes, $|E|$ edges, and node feature matrix $X \in \mathbb{R}^{N \times F}$. Each column of X^T, $\mathbf{x}_i \in \mathbb{R}^F$, represents the features of node i. A *message-passing GNN (MPGNN)* [26] initializes each node representation with its features, i.e. for node i, $\mathbf{h}_i^0 = \mathbf{x}_i$, and iteratively updates each node's hidden representation $\mathbf{h}_i^{(\ell)}$ as

$$\mathbf{h}_i^{(\ell+1)} \;=\; f\Big(\mathbf{h}_i^{(\ell)}, \{\mathbf{h}_j^{(\ell)} : j \in \mathcal{N}(i)\}\Big), \tag{1}$$

where $\mathcal{N}(i)$ denotes the multiset of node i's neighbors and $f(\cdot)$ encapsulates learnable transformations (e.g., linear layers, MLPs) and aggregation mechanisms (e.g., sum, mean, attention). After L layers, each node's hidden state $\mathbf{h}_i^{(L)}$ can be used for node-level tasks or pooled for graph-level tasks. Several popular MPGNN architectures [68, 40, 27, 66] can be considered special cases of this paradigm and differ only in specific choices of learnable functions and aggregations.

2.1 Known Limitations of GNNs

Despite the widespread adoption of MPGNNs, three fundamental limitations have attracted significant attention. We briefly describe each challenge below and provide a more comprehensive discussion and review of prior mitigation approaches for the limitations in Appendix A.1.

Oversquashing. As shown in Equation 1, MPGNNs propagate information through local neighborhoods. Consequently, when two nodes i and j have geodesic distance d, any signal from node i requires at least d message-passing layers to reach node j, and vice versa. During this process, the number of nodes in the effective receptive field can grow exponentially with depth, whereas the hidden dimension $|\mathbf{h}|$ typically remains fixed. This mismatch leads to *oversquashing* [2], where long-range information is compressed and loses influence on downstream tasks.

Alon and Yahav [2] demonstrate oversquashing empirically with their *Tree-NeighborsMatch* dataset, where a root node must predict a label based on leaf node signals traversing multiple tree levels. Despite the label being trivial to determine from direct leaf access, compression across the exponentially growing receptive field ($\mathcal{O}(2^L)$ at depth L) severely degrades performance as tree depth increases.

Oversmoothing. A commonly observed phenomenon in MPGNNs is the convergence of node embeddings $\mathbf{h}$ to an indistinguishable constant vector as network depth grows [61, 44, 56, 12]. In practice, this leads to GNNs being deployed with substantially fewer layers than their counterparts in other domains (e.g., deep convolutional neural networks [37]), despite depth often being critical for strong performance [19]. This effect is frequently attributed to the smoothing effect of repeated message-passing steps, which iteratively mix each node's features with those of its neighbors [44]. Although moderate smoothing can aid learning, it frequently converges rapidly to a subspace of trivial signals [56, 12], yielding nearly identical node representations and thus impeding metric performance. Following recent work [61, 12], we measure oversmoothing via a *Dirichlet energy* metric defined on node embeddings $H^{(\ell)} \;=\; \big[\mathbf{h}_1^{(\ell)}, \ldots, \mathbf{h}_N^{(\ell)}\big]^\top \in \mathbb{R}^{N \times F}$ after ℓ MPGNN layers (cf. Equation 1):

$$\mathcal{D}\big(H^{(\ell)}\big) \;=\; \frac{1}{N} \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{N}(i)} \big\|\mathbf{h}_i^{(\ell)} - \mathbf{h}_j^{(\ell)}\big\|_2^2, \tag{2}$$

A rapid decrease in $\mathcal{D}(H^{(\ell)})$ as ℓ grows indicates oversmoothing.

Limited Expressiveness. Ideally, GNNs should distinguish non-isomorphic graphs, as this is a prerequisite for universal approximation of graph-invariant functions [53]. However, standard MPGNNs' expressiveness is upper-bounded by the 1-dimensional Weisfeiler–Lehman (1-WL) heuristic [67], which measures graph structure through iterative neighborhood aggregation. This expressivity constraint directly limits MPGNNs' ability to distinguish many non-isomorphic graphs that have distinct structural properties but identical neighborhood aggregation patterns [68]. In response, *k-order Weisfeiler–Lehman GNNs (k-WLGNNs)* were introduced [52, 50]. They operate on k-tuples of nodes and match the expressive power of the generalized k-WL heuristic, but typically require $\mathcal{O}(N^k)$ computation and memory. Although recent work has shown that this can be reduced in certain cases to $\mathcal{O}(N^2)$ and $\mathcal{O}(N^3)$ [4], these methods remain expensive for many applications.

To address this limitation efficiently, recent work proposes *positional* or *structural encodings* that augment node features with additional signals. For instance, purely *random* node features [18] render GNNs universal [62, 1], but often impair empirical performance [7]. Hence, recent efforts have shifted toward *graph-derived* features, seeking to make semantically meaningful encodings while retaining invariance to node permutations. Proposed approaches include spectral embeddings [23, 45, 48], subgraph-based representations [43, 9], and homomorphism counts [55, 34, 6]. However, designing an encoding that is simultaneously *unique*, *distance-sensitive*, and *equivariant* is difficult as it is closely tied to the *graph canonization* problem—known to be at least as hard as distinguishing graph isomorphism in general [5]. Recently, Pearl [36] proposes learning positional encodings with GNNs augmented with either standard basis vectors (B-Pearl) or random features (R-Pearl). This leads to better asymptotic complexity but can be expensive for small and medium graphs in practice.

In the following subsection, we introduce kernel methods and random feature approximations that serve as the foundation for our approach to address these limitations.

2.2 Kernels, Random Kernel Features, and Sketching

Consider a positive-definite function $\kappa : \mathbb{R}^F \times \mathbb{R}^F \to \mathbb{R}$. By the Moore–Aronszajn theorem [3], there exists a Hilbert space $\mathcal{H}$ and a *feature map* $\phi : \mathbb{R}^F \to \mathcal{H}$ such that

$$\kappa(\mathbf{x}, \mathbf{y}) \;=\; \langle \phi(\mathbf{x}),\, \phi(\mathbf{y}) \rangle_{\mathcal{H}} \quad \text{for all } \mathbf{x}, \mathbf{y} \in \mathbb{R}^d \tag{3}$$

In other words, any such function κ, known as a *kernel*, defines a lifting ϕ and inner product $\langle \cdot, \cdot \rangle$. This defines a geometry in $\mathcal{H}$ through the norm $\|\phi(\mathbf{x}) - \phi(\mathbf{y})\|_{\mathcal{H}}$, which in turn defines a notion of *distance* between $\mathbf{x}$ and $\mathbf{y}$ and thus $\kappa(\mathbf{x}, \mathbf{y})$ can be interpreted as similarity measure that we refer to as the *kernel similarity* between $\mathbf{x}$ and $\mathbf{y}$. For instance, for *linear kernel* $\kappa(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y}$, the map ϕ is simply the identity, $\phi(\mathbf{x}) = \mathbf{x}$, and distance in $\mathcal{H}$ corresponds to standard Euclidean distance in $\mathbb{R}^d$.

Kernel methods [13, 30, 25] facilitate non-linear modeling by leveraging κ to implicitly measure feature similarity in the space $\mathcal{H}$, thereby allowing otherwise linear models (e.g., SVMs [11]) to capture complex relationships without explicitly mapping data into that space. However, these methods typically require instantiation of *kernel matrix* $K_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ for a dataset $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^F$, which leads to $\mathcal{O}(N^2)$ memory complexity. Consequently, kernel methods often become impractical at large scales, motivating efficient approximations.

A powerful class of such approximations are *random kernel features*. These methods build on the famed *Johnson–Lindenstrauss lemma*, which proves the existence of a random linear map that approximately preserves pairwise distances in a dataset. Concretely, consider the same dataset $\mathcal{X} \subset \mathbb{R}^F$ from the last example. The JL lemma states that for any $\varepsilon \in (0, 1)$, there exists linear map $R \in \mathbb{R}^{D \times F}$ to dimension $D < F$ (with $D = \mathcal{O}(\frac{1}{\varepsilon^2} \log F)$) such that for all i, j

$$(1 - \varepsilon) \, \|\mathbf{x}_i - \mathbf{x}_j\|^2 \;\leq\; \|R\,\mathbf{x}_i - R\,\mathbf{x}_j\|^2 \;\leq\; (1 + \varepsilon) \, \|\mathbf{x}_i - \mathbf{x}_j\|^2 \tag{4}$$

A *Johnson–Lindenstrauss Transform (JLT)* implements such a linear map R via random projection (e.g., a Gaussian matrix, which achieves the bound above whp). In the context of kernel approximations, one seeks a mapping $\varphi : \mathbb{R}^F \to \mathbb{R}^D$ that provides an unbiased estimate of the kernel:

$$\mathbb{E}\big[\varphi(\mathbf{x})^\top \, \varphi(\mathbf{y})\big] = \kappa(\mathbf{x}, \mathbf{y}). \tag{5}$$

In the simplest case of the *linear kernel*, $\kappa(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y}$, we can directly take $\varphi_{\text{linear}}(\mathbf{x}) = R\,\mathbf{x}$, yielding a lower-dimensional estimation $\varphi_{\text{linear}}(\mathbf{x}) \in \mathbb{R}^D$. This provides an unbiased estimate of $\mathbf{x}^\top \mathbf{y}$ while reducing computational costs relative to an explicit $\mathcal{O}(N^2)$ kernel matrix.

More recent work has shown that *nonlinear* kernels can also be estimated via random transformations akin to the JLT [47]. In essence, one applies suitable nonlinearities to the randomly projected inputs, producing an unbiased approximation of a desired kernel (i.e. Equation 5). A prominent example is the use of *random Fourier features* [57], which enable efficient approximations of popular shift-invariant kernels such as the radial basis function (RBF) kernel ϕ_{RBF} and Laplacian kernel $\phi_{\mathcal{L}}$. Concretely, for the RBF kernel, one draws D frequencies $\boldsymbol{\omega}_d$ for $d \in [D]$ from a Gaussian distribution, and applies a trigonometric mapping:

$$\varphi_{\text{RBF}}(\mathbf{x}) \;=\; \sqrt{\frac{2}{D}}\left[\cos(\boldsymbol{\omega}_1^\top \mathbf{x} + b_1),\, \ldots,\, \cos(\boldsymbol{\omega}_D^\top \mathbf{x} + b_D)\right]^\top, \tag{6}$$

where $b_k \in [0, 2\pi)$ are sampled uniformly at random. Similarly, $\phi_{\mathcal{L}}$ is approximated by sampling $\boldsymbol{\omega}$ from a Cauchy distribution and applying the same transformation.

In the next section, we apply these ideas in two distinct ways. First, we apply kernel embedding approximations (e.g., Equation 6) to estimate the feature map ϕ. Second, we apply JLT projections to the resulting kernel embeddings and show that doing so leads to desirable qualities when augmenting MPGNNs. To avoid confusion, we distinguish these two transformations clearly: the first set are **kernel** transformations of node features and the second are **Sketched Random Features (SRF)**.

3 Sketched Random Features

3.1 Defining SRF

Building on the ideas discussed in Section 1-2, our goal is to construct *Sketched Random Features* that (1) are distance-sensitive (2) unique, (3) encode signals across all nodes, and (4) remain permutation-equivariant under node relabelings.

Embedding Operator $\mathcal{E}$. Let $X \in \mathbb{R}^{N\times F}$ be the raw feature matrix, with each column of X^T, denoted $\mathbf{x}_i$, corresponding to node i. We define an embedding operator $\mathcal{E} : \mathbb{R}^{N\times F} \to \mathbb{R}^{N\times D}$ that applies function $\varphi : \mathbb{R}^F \to \mathbb{R}^D$ to each column of X^T independently. That is, $\mathcal{E}(X^T)_{:,i} \;=\; \varphi(\mathbf{x}_i)$.

In this work, we focus on φ functions that map features to random embeddings whose inner products yield unbiased estimates (Equation 5) of a kernel κ (Equation 3). Examples are φ_{linear}, $\varphi_{\mathcal{L}}$, and φ_{RBF}, as discussed in Section 2.2, yielding embedding operators $\mathcal{E}_{\text{linear}}$, $\mathcal{E}_{\mathcal{L}}$, and $\mathcal{E}_{\text{RBF}}$, respectively. Hence, we define the *kernel feature matrix* as

$$\Phi \;=\; \mathcal{E}(X) \;\in\; \mathbb{R}^{N\times D}.$$

Sketch Operator $\mathcal{S}$. We next introduce a sketch operator $\mathcal{S} : \mathbb{R}^{N\times D} \to \mathbb{R}^{N\times D}$. This random projection matrix implements a Johnson Lindenstrauss Transform (JLT) (Section 2.2) which provides approximate preservation of pairwise distances with high probability (see Equation 4). In this work, we focus on the *additive Gaussian* (AG) sketch, which we define as:

$$\mathcal{S}_{\text{AG}}(\Phi) \;=\; \left(I + \tfrac{1}{\sqrt{N}}\, G\right)\Phi, \tag{7}$$

where $I \in \mathbb{R}^{N\times N}$ is the identity matrix, and $G \in \mathbb{R}^{N\times N}$ has i.i.d. entries $G_{ij} \sim \mathcal{N}(0,1)$. Multiplying Φ by the sketch forms random linear combinations of all rows in Φ.

Unlike typical JLT applications, we *do not* reduce dimension N. Instead, we employ this random projection because it possesses properties (Propositions 3.1–3.5) that we show in Section 3.2 can mitigate the standard limitations of message-passing GNNs. Importantly, the dimension D is determined by the embedding operator $\mathcal{E}$ rather than $\mathcal{S}$.

Multi-Projection Sketching. $\mathcal{S}_{\text{AG}}$ introduces a one dimensional random projection of each column of Φ. We can generalize this to multiple dimensions via concatenation. Specifically, we introduce the k-order sketch operator $\mathcal{S}^{(k)}_{\text{AG}} : \mathbb{R}^{N\times D} \to \mathbb{R}^{N\times kD}$:

$$\mathcal{S}^{(k)}_{\text{AG}}(\Phi) \;=\; \left[\left(I + \tfrac{1}{\sqrt{N}}\, G^{(1)}\right)\Phi \;\middle\|\; \left(I + \tfrac{1}{\sqrt{N}}\, G^{(2)}\right)\Phi \;\middle\|\; \ldots \middle\|\; \left(I + \tfrac{1}{\sqrt{N}}\, G^{(k)}\right)\Phi\right]$$

where each $G^{(k)} \in \mathbb{R}^{N\times N}$ is drawn independently with i.i.d. $\mathcal{N}(0,1)$ entries. This operation concatenates k independent AG sketches, enabling k-dimensional estimates of each feature.

Sketched Random Features (SRF). Bringing these components together, we define the *kernel sketch* matrix Z as

$$Z \;=\; \mathcal{S}^{(k)}\big(\mathcal{E}(X)\big) \;\in\; \mathbb{R}^{N\times(kD)}, \tag{8}$$

where the new feature embedding of node i is given by $\mathbf{z}_i \triangleq Z_{i,:}$[1]. In our experiments (Section 4), we primarily analyze the (multi-) projection AG sketch $\mathcal{S}^{(k)}_{\mathrm{AG}}$ as introduced above, but also consider the trivial identity sketch $\mathcal{S}_{\mathrm{id}} = I$ as an ablation study. For embedding operator $\mathcal{E}$, we consider the operators based on the random kernel features discussed in Section 2.2: $\mathcal{E}_{\mathrm{linear}}, \mathcal{E}_{\mathcal{L}}$, and $\mathcal{E}_{\mathrm{RBF}}$.

3.2 Enhancing MPGNNs with Node Feature Sketches

We next turn to incorporate SRF into MPGNNs. To do so, we augment the node hidden states $\mathbf{h}_i^{(\ell)}$ at each layer with the corresponding sketched embedding $\mathbf{z}_i$ via concatination. Let $\widetilde{\mathbf{h}}_i^{(\ell)} = [\mathbf{h}_i^{(\ell)} | \mathbf{z}_i]$ denote this augmented representation, where $[\cdot|\cdot]$ represents vector concatenation along the feature dimension. The updated MPGNN formulation becomes (cf. Equation 1):

$$\mathbf{h}_i^{(\ell+1)} \;=\; f\big(\widetilde{\mathbf{h}}_i^{(\ell)}, \{\widetilde{\mathbf{h}}_j^{(\ell)} : j \in \mathcal{N}(i)\}\big). \tag{9}$$

By injecting $\mathbf{z}_i$ at each layer, the model has access to both local node features and the information encoded by SRF. This strategy addresses common MPGNN weaknesses discussed in Section 2, as detailed in the proceeding subsection. Algorithm 1 summarizes the SRF-enhanced GNN procedure.

3.3 Properties of Sketched Random Features

We now establish key theoretical properties[2] of SRF and demonstrate how they address core limitations of message-passing GNNs. The following propositions characterize the fundamental mathematical properties of our approach.

Proposition 3.1 (Unbiased Cross-Terms in the Kernel Matrix)**.** *SRF provides an unbiased estimation of cross-terms in the Kernel matrix. Specifically, for any distinct nodes i and j, the cross-term of the Gram matrix is unbiased:* $\mathbb{E}_{X,G}[\mathbf{z}_i^\top \mathbf{z}_j] = \kappa(\mathbf{x}_i, \mathbf{x}_j)$.

Proof. From Equation 7, the (i,j)-th inner product is

$$\mathbf{z}_i^\top \mathbf{z}_j = \sum_{p,q=1}^{N} (I_{ip} + \tfrac{1}{\sqrt{N}} G_{ip})(I_{jq} + \tfrac{1}{\sqrt{N}} G_{jq})\, \varphi(\mathbf{x}_p)^\top \varphi(\mathbf{x}_q).$$

We note that for $i \neq j$, $\mathbb{E}[G_{ip}] = 0$ and $\mathbb{E}[G_{ip}G_{jq}] = \delta_{ij}\delta_{pq} = 0$. Thus, by Equation 5, we have

$$\mathbb{E}_{X,G}[\mathbf{z}_i^\top \mathbf{z}_j] = \sum_{p,q=1}^{N} I_{ip} I_{jq}\, \mathbb{E}\big[\varphi(\mathbf{x}_p)^\top \varphi(\mathbf{x}_q)\big] = \mathbb{E}\big[\varphi(\mathbf{x}_i)^\top \varphi(\mathbf{x}_j)\big] = \kappa(\mathbf{x}_i, \mathbf{x}_j).$$

□

Proposition 3.2 (Kernel Distance Sensitivity)**.** *With high probability, there exists a positive $c \sim \mathcal{O}(N^{-1/2})$ such that for any nodes i, j:*

$$(1-c)\|\varphi(\mathbf{x}_i) - \varphi(\mathbf{x}_j)\|_2 \leq \|\mathbf{z}_i - \mathbf{z}_j\|_2 \leq (1+c)\|\varphi(\mathbf{x}_i) - \varphi(\mathbf{x}_j)\|_2$$

Proof. By construction, $\mathbf{z}_i - \mathbf{z}_j \;=\; \Big(I + \frac{1}{\sqrt{N}} G\Big)\big(\varphi(\mathbf{x}_i) \;-\; \varphi(\mathbf{x}_j)\big)$ since each row $\mathbf{z}_i$ corresponds to the i-th row of $\big(I + \frac{1}{\sqrt{N}} G\big)\,\mathcal{E}(X)$. If $\varphi(\mathbf{x}_i) = \varphi(\mathbf{x}_j)$, then $\|\mathbf{z}_i - \mathbf{z}_j\|_2 = 0$ and the claim holds trivially. Thus, we assume $\varphi(\mathbf{x}_i) \neq \varphi(\mathbf{x}_j)$ and define $\mathbf{v} = \varphi(\mathbf{x}_i) - \varphi(\mathbf{x}_j)$. Applying the triangle inequality:

$$\big|\|\mathbf{v}\|_2 - \tfrac{1}{\sqrt{N}}\|G\,\mathbf{v}\|_2\big| \;\leq\; \big\|\mathbf{v} + \tfrac{1}{\sqrt{N}}\, G\,\mathbf{v}\big\|_2 \;\leq\; \|\mathbf{v}\|_2 + \tfrac{1}{\sqrt{N}}\|G\,\mathbf{v}\|_2$$

[1] In the featureless limit where all $\mathbf{x}_i$ are identical, SRF degenerates to random node individualization [62, 1], preserving expressive power through randomization.

[2] For notational clarity, we present proofs under base case $\mathcal{S}^{(1)}_{\mathrm{AG}}$ and as the extension to $\mathcal{S}^{(k)}_{\mathrm{AG}}$ is straightforward.

Next, the JL lemma (Equation 4) guarantees that, given $\varepsilon \in [0, 1]$, with high probability over the choice of the random Gaussian matrix G[3]:

$$\sqrt{1-\varepsilon}\,\|\mathbf{v}\|_2 \;\leq\; \tfrac{1}{\sqrt{N}}\|G\,\mathbf{v}\|_2 \;\leq\; \sqrt{1+\varepsilon}\,\|\mathbf{v}\|_2.$$

Substituting these bounds back into the triangle inequality shows

$$\left(1-\sqrt{\frac{1+\varepsilon}{N}}\right) \leq \frac{\left\|\mathbf{z}_i-\mathbf{z}_j\right\|_2}{\|\mathbf{v}\|_2} \leq \left(1+\sqrt{\frac{1+\varepsilon}{N}}\right).$$

Thus, letting $c = \sqrt{(1+\varepsilon)/N}$ completes the proof. □

Proposition 3.3 (Cross-Node Information)**.** *For any node i, the sketched embedding $\mathbf{z}_i$ contains a linear combination of the embeddings of* all *nodes in the graph.*

Remark. Proposition 3.3 ensures that each SRF embedding encodes cross-node information due to the row sketch construction.

Proposition 3.4 (Almost Sure Uniqueness)**.** *$\{\mathbf{z}_i\}$ are unique with probability 1.*

Proof. Without loss of generality, consider the case where the node features are identical, e.g. $\mathbf{x}_i = \mathbf{1}$. Then $\varphi(\mathbf{x}_i) = \varphi(\mathbf{1})$ for all i, and from the definition of SRF with $\mathcal{S}_{AG}^{(1)}$ we have for row i of Z:

$$\mathbf{z}_i = \varphi(\mathbf{1}) + \frac{1}{\sqrt{N}}\sum_{j=1}^{N} G_{ij}\varphi(\mathbf{1})$$

Since $\{\sum_{j=1}^{N} G_{ij}\}$ are independent Gaussian random variables, the coefficients multiplying $\varphi(\mathbf{1})$ are almost surely unique. Thus, $\mathbf{z}_i \neq \mathbf{z}_j$ for $i \neq j$ with probability 1. In the general case where node features differ, $\varphi(\mathbf{x}_i)$ and $\varphi(\mathbf{x}_j)$ may also differ, providing additional distinctness. □

Proposition 3.5 (Permutation Equivariance in Expectation)**.** *Let π be a fixed permutation of node indices, with corresponding permutation matrix P_π. Let the function $f \triangleq \mathcal{S}_{AG} \circ \mathcal{E}$ be the SRF operation (Equation 8). Then f is permutation equivariant in expectation: $\mathbb{E}_{X,G}[f(P_\pi X)] = \mathbb{E}_{X,G}[P_\pi f(X)]$.*

Proof. By definition, $f(X) = \left(I + \frac{1}{\sqrt{N}}G\right)\Phi(X)$ and $\mathbb{E}[G] = 0$. Then

$$\mathbb{E}[f(P_\pi X)] = \mathbb{E}\Big[\left(I + \tfrac{1}{\sqrt{N}}G\right)P_\pi\Phi(X)\Big] = P_\pi\Phi(X),$$

and similarly

$$\mathbb{E}[P_\pi f(X)] = \mathbb{E}\Big[P_\pi\left(I + \tfrac{1}{\sqrt{N}}G\right)\Phi(X)\Big] = P_\pi\Phi(X).$$

□

SRF Mitigates Oversquashing. SRF mitigates oversquashing by encoding topology-agnostic cross-node information in each sketched embedding $\mathbf{z}_i$, as shown in Proposition 3.3. While this encoding into $\mathbb{R}^D$ (where typically $D \ll N$) is not lossless, it enables direct information flow between all node pairs independent of their geodesic distance. This circumvents the exponential information bottleneck inherent in multi-hop message passing [2]. Our empirical analysis in Section 4.1 demonstrates that SRF achieves linear rather than exponential information loss as N increases.

SRF Alleviates Oversmoothing. SRF addresses oversmoothing by introducing node-specific, distance-sensitive embeddings that preserve variance across nodes (Propositions 3.1 and 3.2, respectively). The inclusion of unique $\mathbf{z}_i$ vectors (Proposition 3.4) repeatedly injected into message passing (Equation 9) ensures $\widetilde{\mathbf{h}}_i^{(\ell)} = [\mathbf{h}_i^{(\ell)} | \mathbf{z}_i]$ remain distinct as ℓ grows. Our empirical results (Section 4.1) show this approach maintains higher representation diversity even in base features $\mathbf{h}_i^{(\ell)}$, preventing representational collapse with depth.

[3]Note that while the version of the JL lemma discussed in Section 2.2 (Equation 4) provides bounds for squared norms, taking square roots is valid since all terms are positive.

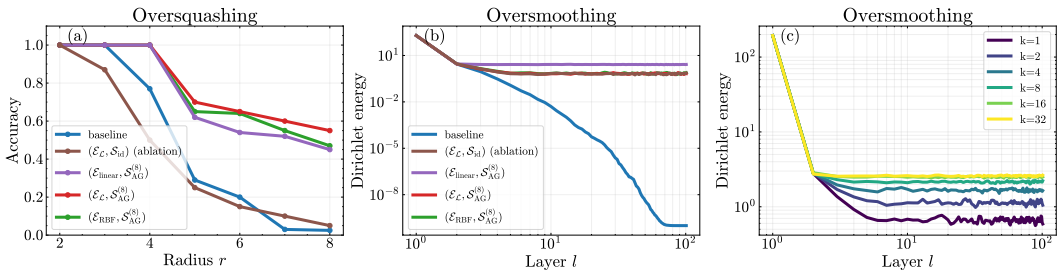

Figure 1: Analysis of oversquashing and oversmoothing across model variants. (a) Accuracy vs. graph radius showing the effect of oversquashing across methods. (b) Dirichlet energy across layers for different methods. (c) Dirichlet energy across for $\mathcal{E}_\mathcal{L}$ with varying k. Lower Dirichlet energy indicates more oversmoothing.

Universality of SRF in MPGNNs. SRF provides a principled way to achieve universality in MPGNNs. As discussed in Section 2, purely random features guarantee universality, but often fail to improve metric performance in practice. In contrast, structural encodings provide semantically meaningful and equivariant signals which lead to improved empirical performance, but often fail to provide an encoding that is unique, distance sensitive and equivariant, due to the inherent difficulty of the graph canonization problem. SRF strikes a balance between these approaches: it is unique (Proposition 3.4), ensuring universality, while preserving distance in the node feature kernel space whp (Proposition 3.2). Additionally, it is permutation equivariant in expectation (Proposition 3.5).

Memory and Runtime Complexity. While the JLT is traditionally implemented with dense random projection matrices, memory and runtime efficiency can be improved by using structured random matrices (SRMs) [15] as detailed in Appendix A.2. SRMs lower storage requirements to $\mathcal{O}(N)$ and enable matrix-vector multiplication in $\mathcal{O}(N \log N)$. Notably, these matrices preserve the theoretical guarantees of Section 3. This yields favorable asymptotic complexity compared to many positional encoding methods. Detailed complexity analysis and comparisons are in Appendix A.2.

4 Experiments

We empirically evaluate SRF-enhanced GNNs [4], first validating that SRF addresses key MPGNN limitations Section 4.1, then demonstrating performance on real-world benchmarks (Section 4.2). We use GIN [68] for unattributed-edge graphs and GINE [31] for edge-featured graphs, with additional architectures tested in Appendix B to verify architecture-agnostic benefits.

We evaluate embedding operators $\mathcal{E}_{\text{linear}}$, $\mathcal{E}_\mathcal{L}$, and $\mathcal{E}_{\text{RBF}}$ with additive Gaussian sketch $\mathcal{S}_{\text{AG}}^{(k)}$, denoting configurations as $\left(\mathcal{E}, \mathcal{S}_{\text{AG}}^{(k)}\right)$. To isolate sketching effects from kernel features alone, we ablate with identity operator $\mathcal{S}_{\text{id}} = I$, comparing $\left(\mathcal{E}_\mathcal{L}, \mathcal{S}_{\text{id}}\right)$ against $\left(\mathcal{E}_\mathcal{L}, \mathcal{S}_{\text{AG}}^{(k)}\right)$. Full dataset descriptions, baseline details, hyperparameter search procedures, and other experimental details are provided in Appendix C. We include additional experiments in the Appendix: validation of SRF benefits across diverse GNN architectures (Appendix B) and analysis of SRF hyperparameters (embedding dimension D and projection count k) on performance (Appendix D).

4.1 Synthetic Learning Tasks

Oversquashing. We evaluate SRF's ability to mitigate oversquashing using the *Tree-NeighborsMatch* synthetic benchmark [2] discussed in Section 2. Recall that as radius r increases, exponentially more information must be aggregated by the message passing algorithm, making the task progressively harder for standard MPGNNs. Results in Figure 1 show that while baseline GIN and ablation $\left(\mathcal{E}_\mathcal{L}, \mathcal{S}_{\text{id}}\right)$ suffer severe performance degradation beyond $r = 4$, SRF-enhanced models maintain higher accuracy at large radii, regardless of choice of $\mathcal{E}$. Notably, all SRF variants achieve

[4] Our source code is available at https://github.com/ryienh/sketched-random-features.

Table 1: Performance on synthetic expressiveness benchmarks. All results averaged over 5 runs. Standard deviations (≤ 0.002 across all experiments) omitted for clarity.

Dataset	*Baseline*	*Ablation*	*Ours* ($\mathcal{S}_{\mathrm{AG}}^{(1)}$)			*Ours* ($\mathcal{S}_{\mathrm{AG}}^{(8)}$)		
	None	($\mathcal{E}_{\mathcal{L}}, \mathcal{S}_{\mathrm{id}}$)	$\mathcal{E}_{\mathrm{linear}}$	$\mathcal{E}_{\mathcal{L}}$	$\mathcal{E}_{\mathrm{RBF}}$	$\mathcal{E}_{\mathrm{linear}}$	$\mathcal{E}_{\mathcal{L}}$	$\mathcal{E}_{\mathrm{RBF}}$
CSL (Acc ↑)	0.100	0.100	1.000	1.000	1.000	1.000	1.000	1.000
EXP (Acc ↑)	0.518	0.520	1.000	1.000	1.000	1.000	1.000	1.000

perfect accuracy up to $r = 4$ and demonstrate more graceful performance decay thereafter, with accuracy remaining above 40% even at $r = 8$. This improved performance at large radii empirically validates our theoretical analysis that SRF enables comparatively more effective long-range communication in MPGNNs.

Oversmoothing. To assess SRF against oversmoothing, we follow Rusch et al. [61]'s protocol using Cora [51] with randomized features ($X_{jk} \sim \mathcal{N}(0, 1)$) and measuring Dirichlet energy (Equation 2). Figure 1(b) shows baseline GNNs exhibit near-exponential energy decay while SRF variants maintain substantially higher energy. While feature injection alone helps prevent decay (ablation study), Figure 1(c) shows that increasing projections (k) further preserves node distinctness in deeper layers.

Expressiveness. We evaluate the universal approximation capabilities of SRF-enhanced MPGNNs using two graph isomorphism discrimination benchmarks: CSL [54] and EXP [1], containing graphs indistinguishable by 1-WL and 2-WL tests respectively. Table 1 shows SRF-enhanced models achieving perfect discrimination on both datasets, while baselines and ablations (without sketching) perform no better than random chance.

4.2 Real-world Graph Learning Tasks

We evaluate SRF-enhanced GNNs on real-world datasets, benchmarking against existing positional encoding approaches (Section 2.1). Our results show SRF-enhanced GNNs *outperform many positional encoding approaches* (Table 2, left) while offering *substantial efficiency gains*: about 3 times faster runtime and about two orders of magnitude less memory than PEARL on evaluated datasets (Figure 2). They show *robustness in out-of-distribution tasks* where other approaches falter, remaining competitive with state-of-the-art approaches (Table 2, right). Finally, they can be *combined with positional encodings for cumulative improvements* due to their complementary nature (Table 3).

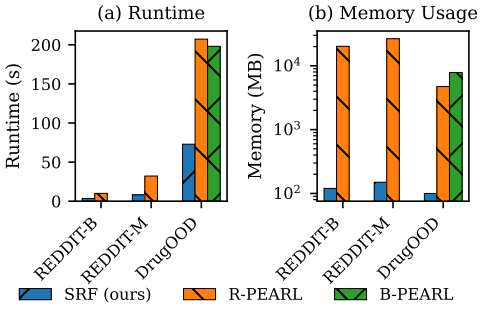

Figure 2: Training efficiency comparison of SRF (ours), R-PEARL, and B-PEARL. (a) Runtime in seconds. (b) Memory usage in MB (log scale).

Baselines. We consider several positional encoding baselines discussed in Section 2.1 and Appendix A: random node features (rand id) [1, 62], SignNet and BasisNet [45], efficient approximations of SignNet (SignNet-8S, SignNet-8L) [36], SPE [32] with its efficient variant SPE-8S, and Pearl (R-PEARL, B-PEARL) [36]. For some experiments, we include graph transformers and attention-enhanced GNNs: GPS [58], SAN [41], SUN [24], GNN-AK [71], Graph ViT [29].

Social Network Classification. We evaluate on REDDIT-B and REDDIT-M datasets [69], which represent online discussion threads where nodes are users and edges indicate comment interactions. These datasets require models to classify discussion graphs into their respective subreddits (online communities). As shown in Table 2, SRF-augmented GNNs consistently outperform baseline methods. Notably, even our simplest variant ($\mathcal{E}_{\mathrm{linear}}$) demonstrates substantial improvements over prior state-of-the-art methods, with $\mathcal{E}_{\mathrm{RBF}}$ providing the strongest performance. The ablation confirms these gains derive from sketching rather than merely adding kernel features.

Table 2: Performance comparison of SRF enhanced GNNs and several positional encoding baselines on Reddit (% accuracy) and DrugOOD (% AUC) datasets. Results for baselines from Kanatsoulis et al. [36]. Values missing from the literature are denoted by "—". OOM indicates out of memory. First, second, and third best results are highlighted in blue, green, and orange, respectively.

	Reddit (Acc ↑)		DrugOOD (AUC ↑)		
	REDDIT-B	REDDIT-M	Assay	Scaffold	Size
Baselines:					
GINE (No PE)	91.8 ± 1.0	56.9 ± 2.0	71.68 ± 1.10	68.00 ± 0.60	66.04 ± 0.70
GINE + rand id	91.8 ± 1.6	57.0 ± 2.1	—	—	—
SignNet	OOM	OOM	72.27 ± 0.97	66.43 ± 1.06	64.03 ± 0.70
SignNet-8S	92.4 ± 1.1	57.8 ± 0.8	—	—	—
SignNet-8L	79.5 ± 12.3	41.4 ± 2.7	—	—	—
BasisNet	—	—	71.66 ± 0.05	66.32 ± 5.68	60.79 ± 3.19
SPE	—	—	72.53 ± 0.66	69.64 ± 0.49	66.02 ± 0.49
SPE-8S	—	—	71.72 ± 0.71	68.72 ± 0.63	65.74 ± 2.20
R-PEARL	93.0 ± 1.3	59.4 ± 1.0	72.24 ± 0.30	69.20 ± 1.00	65.89 ± 1.30
B-PEARL	—	—	71.22 ± 0.42	69.51 ± 0.62	66.58 ± 0.67
Ablation:					
$(\mathcal{E}_{\mathcal{L}}, \mathcal{S}_{\text{id}})$	92.56 ± 0.58	58.32 ± 0.47	71.89 ± 0.37	65.34 ± 0.26	63.29 ± 0.63
Ours: $(\mathcal{S}_{\text{AG}}^{(8)})$					
$\mathcal{E}_{\text{linear}}$	94.06 ± 0.39	60.18 ± 0.39	72.29 ± 0.30	68.79 ± 0.64	66.45 ± 0.24
$\mathcal{E}_{\mathcal{L}}$	94.00 ± 0.35	60.33 ± 0.37	72.51 ± 0.69	69.43 ± 0.90	67.23 ± 0.54
$\mathcal{E}_{\text{RBF}}$	94.13 ± 0.69	60.53 ± 0.29	72.63 ± 0.41	69.60 ± 0.48	66.67 ± 0.35

Molecular Graph Out-of-Distribution (OOD) Generalization. We evaluate OOD generalization on DrugOOD [33], which tests molecular property prediction across three domain shifts (Assay, Scaffold, Size). As Kanatsoulis et al. [36] demonstrate, these OOD tasks are particularly challenging for positional encoding approaches, with many advanced methods actually degrading performance compared to GNN backbones alone. Conversely, Table 2 demonstrates that our *feature-based* augmentation approach effectively overcomes this limitation, with all SRF variants consistently outperforming the baseline GINE model across all splits. Notably, our approach achieves state-of-the-art performance on the Assay and Size shifts, and remains competitive on the Scaffold split.

Long-Range Interactions in Peptide Structure Prediction. To assess whether SRF complements structural approaches, we evaluate on Peptides-struct [22], a benchmark with long-range dependencies where graph transformers typically excel. Table 3 shows that combining PEARL positional encodings and SRF augmentation $(\mathcal{E}_{\text{RBF}}, \mathcal{S}_{\text{AG}}^{(8)})$ closes the performance gap between graph transformers and GNNs, verifying SRF provides complementary benefits to positional encodings.

Table 3: Performance comparison for peptide-struct (MAE, lower is better). Baseline results reported by Kanatsoulis et al. [36]. Error bars presented in Appendix C but do not exceed $\pm$ 0.004.

Baselines							Ours	
R-PEARL	B-PEARL	GPS	SAN+ RWSE	GNN-AK+	SUN	Graph ViT	R-PEARL +SRF	B-PEARL +SRF
0.247	0.248	0.252	0.255	0.274	0.250	0.245	0.245	0.243

5 Conclusion

We have presented an unconventional application of the Johnson-Lindenstrauss transform for enhancing message-passing graph neural networks. Our theoretical analysis and empirical results demonstrate that such a strategy overcomes well-known shortcomings of MPGNNs with minimal computational overhead, making it a practical enhancement for existing architectures. A promising direction for future work is to investigate whether analogous sketching techniques generalize to topology-aware embeddings (e.g., graph random kernel features [16]) or structural encodings.

Acknowledgments and Disclosure of Funding

This research used resources of the Argonne Leadership Computing Facility, a U.S. Department of Energy (DOE) Office of Science user facility at Argonne National Laboratory and is based on research supported by the U.S. DOE Office of Science-Advanced Scientific Computing Research Program, under Contract No. DE-AC02-06CH11357. Additional funding support comes from the National Science Foundation (CCF-2119184 CNS-2313190 CCF-1822949 CNS-1956180). RW gratefully acknowledges the support of NSF DMS-2023109, DOE DE-SC0022232, the NSF-Simons National Institute for Theory and Mathematics in Biology (NITMB) through NSF (DMS-2235451) and Simons Foundation (MP-TMPS-00005320), and the Margot and Tom Pritzker Foundation.

References

[1] Ralph Abboud, Ismail Ilkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The surprising power of graph neural networks with random node initialization. *arXiv preprint arXiv:2010.01179*, 2020.

[2] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=i80OPhOCVH2.

[3] Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

[4] Waiss Azizian and Marc Lelarge. Expressive power of invariant and equivariant graph neural networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=lxHgXYN4bwl.

[5] László Babai and Eugene M Luks. Canonical labeling of graphs. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 171–183, 1983.

[6] Linus Bao, Emily Jin, Michael Bronstein, İsmail İlkan Ceylan, and Matthias Lanzinger. Homomorphism counts as structural encodings for graph learning. *arXiv preprint arXiv:2410.18676*, 2024.

[7] Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M. Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. In *International Conference on Learning Representations*, 2022.

[8] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250, 2001.

[9] Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):657–668, 2022.

[10] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.

[11] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[12] Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020.

[13] Colin Campbell. Kernel methods: a survey of current techniques. *Neurocomputing*, 48(1-4): 63–84, 2002.

[14] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International conference on machine learning*, pages 1725–1735. PMLR, 2020.

[15] Krzysztof M Choromanski, Mark Rowland, and Adrian Weller. The unreasonable effectiveness of structured random orthogonal embeddings. *Advances in neural information processing systems*, 30, 2017.

[16] Krzysztof Marcin Choromanski. Taming graph kernels with random features. In *International Conference on Machine Learning*, pages 5964–5977. PMLR, 2023.

[17] Fan RK Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

[18] George Dasoulas, Ludovic Dos Santos, Kevin Scaman, and Aladin Virmaux. Coloring graph neural networks for node disambiguation. *arXiv preprint arXiv:1912.06058*, 2019.

[19] Nima Dehmamy, Albert-László Barabási, and Rose Yu. Understanding the representation power of graph neural networks in learning graph topology. *Advances in Neural Information Processing Systems*, 32, 2019.

[20] Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Lio, and Michael M Bronstein. On over-squashing in message passing neural networks: The impact of width, depth, and topology. In *International Conference on Machine Learning*, pages 7865–7885. PMLR, 2023.

[21] Zehao Dong, Muhan Zhang, Philip RO Payne, Michael A Province, Carlos Cruchaga, Tianyu Zhao, Fuhai Li, and Yixin Chen. Rethinking the power of graph canonization in graph representation learning with stability. *arXiv preprint arXiv:2309.00738*, 2023.

[22] Vijay Prakash Dwivedi, Ladislav Rampášek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. *Advances in Neural Information Processing Systems*, 35:22326–22340, 2022.

[23] Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48, 2023.

[24] Fabrizio Frasca, Beatrice Bevilacqua, Michael Bronstein, and Haggai Maron. Understanding and extending subgraph gnns by rethinking their symmetries. *Advances in Neural Information Processing Systems*, 35:31376–31390, 2022.

[25] Benyamin Ghojogh, Ali Ghodsi, Fakhri Karray, and Mark Crowley. Reproducing kernel hilbert space, mercer's theorem, eigenfunctions, nystr\" om method, and use of kernels in machine learning: Tutorial and survey. *arXiv preprint arXiv:2106.08443*, 2021.

[26] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[27] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[28] Arman Hasanzadeh, Ehsan Hajiramezanali, Shahin Boluki, Mingyuan Zhou, Nick Duffield, Krishna Narayanan, and Xiaoning Qian. Bayesian graph neural networks with adaptive connection sampling. In *International conference on machine learning*, pages 4094–4104. PMLR, 2020.

[29] Xiaoxin He, Bryan Hooi, Thomas Laurent, Adam Perold, Yann LeCun, and Xavier Bresson. A generalization of vit/mlp-mixer to graphs. In *International conference on machine learning*, pages 12724–12745. PMLR, 2023.

[30] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. 2008.

[31] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

[32] Yinan Huang, William Lu, Joshua Robinson, Yu Yang, Muhan Zhang, Stefanie Jegelka, and Pan Li. On the stability of expressive positional encodings for graphs. *arXiv preprint arXiv:2310.02579*, 2023.

[33] Yuanfeng Ji, Lu Zhang, Jiaxiang Wu, Bingzhe Wu, Lanqing Li, Long-Kai Huang, Tingyang Xu, Yu Rong, Jie Ren, Ding Xue, et al. Drugood: Out-of-distribution dataset curator and benchmark for ai-aided drug discovery–a focus on affinity prediction problems with noise annotations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 8023–8031, 2023.

[34] Emily Jin, Michael Bronstein, Ismail Ilkan Ceylan, and Matthias Lanzinger. Homomorphism counts for graph neural networks: All about that basis. *arXiv preprint arXiv:2402.08595*, 2024.

[35] William B Johnson, Joram Lindenstrauss, and Gideon Schechtman. Extensions of lipschitz maps into banach spaces. *Israel Journal of Mathematics*, 54(2):129–138, 1986.

[36] Charilaos I Kanatsoulis, Evelyn Choi, Stephanie Jegelka, Jure Leskovec, and Alejandro Ribeiro. Learning efficient positional encodings with graph neural networks. *arXiv preprint arXiv:2502.01122*, 2025.

[37] Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. A survey of the recent architectures of deep convolutional neural networks. *Artificial intelligence review*, 53: 5455–5516, 2020.

[38] Jinwoo Kim, Saeyoon Oh, and Seunghoon Hong. Transformers generalize deepsets and can be extended to graphs & hypergraphs. *Advances in Neural Information Processing Systems*, 34: 28016–28028, 2021.

[39] Jinwoo Kim, 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*, 35:14582–14595, 2022.

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

[41] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34:21618–21629, 2021.

[42] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9267–9276, 2019.

[43] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems*, 33:4465–4478, 2020.

[44] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[45] Derek Lim, Joshua Robinson, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. Sign and basis invariant networks for spectral graph representation learning. *arXiv preprint arXiv:2202.13013*, 2022.

[46] Kevin Lin, Lijuan Wang, and Zicheng Liu. Mesh graphormer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12939–12948, 2021.

[47] Fanghui Liu, Xiaolin Huang, Yudong Chen, and Johan AK Suykens. Random features for kernel approximation: A survey on algorithms, theory, and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):7128–7148, 2021.

[48] George Ma, Yifei Wang, and Yisen Wang. Laplacian canonization: A minimalist approach to sign and basis invariant spectral embedding. *Advances in Neural Information Processing Systems*, 36:11296–11337, 2023.

[49] Liheng Ma, Chen Lin, Derek Lim, Adriana Romero-Soriano, Puneet K Dokania, Mark Coates, Philip Torr, and Ser-Nam Lim. Graph inductive biases in transformers without message passing. In *International Conference on Machine Learning*, pages 23321–23337. PMLR, 2023.

[50] Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. In *International Conference on Learning Representations*, 2018.

[51] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163, 2000.

[52] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609, 2019.

[53] Christopher Morris, Yaron Lipman, Haggai Maron, Bastian Rieck, Nils M Kriege, Martin Grohe, Matthias Fey, and Karsten Borgwardt. Weisfeiler and leman go machine learning: The story so far. *The Journal of Machine Learning Research*, 24(1):15865–15923, 2023.

[54] Ryan Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Relational pooling for graph representations. In *International Conference on Machine Learning*, pages 4663–4673. PMLR, 2019.

[55] Daniel Neuen. Homomorphism-distinguishing closedness for graphs of bounded tree-width. *arXiv preprint arXiv:2304.07011*, 2023.

[56] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1ldO2EFPr.

[57] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

[58] Ladislav Rampášek, Michael 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*, 35:14501–14515, 2022.

[59] Ladislav Rampášek, Michael 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*, 35:14501–14515, 2022.

[60] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019.

[61] T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993*, 2023.

[62] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks. In *Proceedings of the 2021 SIAM international conference on data mining (SDM)*, pages 333–341. SIAM, 2021.

[63] Ahsan Shehzad, Feng Xia, Shagufta Abid, Ciyuan Peng, Shuo Yu, Dongyu Zhang, and Karin Verspoor. Graph transformers: A survey. *arXiv preprint arXiv:2407.09777*, 2024.

[64] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*, 2021.

[65] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

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

[67] Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *nti, Series*, 2(9):12–16, 1968.

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

[69] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1365–1374, 2015.

[70] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34:28877–28888, 2021.

[71] Lingxiao Zhao, Wei Jin, Leman Akoglu, and Neil Shah. From stars to subgraphs: Uplifting any gnn with local structure awareness. *arXiv preprint arXiv:2110.03753*, 2021.

[72] Kaixiong Zhou, Xiao Huang, Daochen Zha, Rui Chen, Li Li, Soo-Hyun Choi, and Xia Hu. Dirichlet energy constrained learning for deep graph neural networks. *Advances in Neural Information Processing Systems*, 34:21834–21846, 2021.

A SRF Background and Complexity Analysis

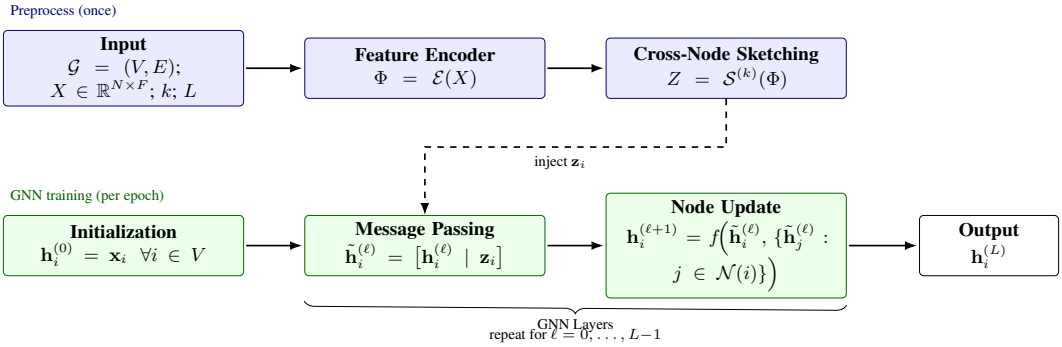

Figure 3: Block diagram visualizing the SRF method defined in Algorithm 1. SRF is computed once (top, blue) and then concatenated to node states at every GNN layer during training (bottom, green).

This section expands on the limitations of MPGNNs discussed in Section 2.1, reviews prior mitigation strategies, and compares the computational complexity of SRF with recent baselines. Figure 3 overviews the SRF preprocessing strategy defined in Algorithm 1 and described in detail in Section 3.

A.1 Additional Discussion on Prior Work

Oversquashing. Theoretical work formalizes oversquashing by analyzing the partial derivative of the hidden states of nodes i and j at depth l, i.e. $\left|\partial \mathbf{h}_i^{(l)} / \partial \mathbf{h}_j^{0}\right|$, with Topping et al. [64] showing it is bounded by $c \cdot A^d$ for constant c, graph adjacency matrix A, and geodesic distance d, thereby implying that node $\mathbf{h}_i$ and $\mathbf{h}_j$ become exponentially less sensitive to each other as d grows. Furthermore, Di Giovanni et al. [20] validate the intuition posed in Alon and Yahav [2], proving that increasing hidden dimension $|\mathbf{h}|$ can alleviate oversquashing, whereas greater depth $|L|$ alone fails to resolve it.

Recent work seeks to mitigate oversquashing in MPGNNs by defining a *rewiring* $\mathcal{Q} : \mathbb{R}^{n \times n} \to \mathbb{R}^{n \times n}$ that operates on the adjacency matrix A to yield $\mathcal{Q}(A)$, hence producing a rewired graph $\mathcal{G}'$. This process reduces either the graph diameter or the Cheeger constant (which quantifies the connectivity of a graph relative to its cut size) [17][5]. Thus, these strategies alleviate oversquashing by altering topology to remove bottlenecks. However, this leads to an altered and usually denser graph $\mathcal{G}'$, undermining the original topological bias of message passing and increasing computational overhead.

Oversmoothing. Common ways to mitigate oversmoothing include strategies based on regularization [28, 72, 60] or skip connections [42, 14]. Nevertheless, striking a balance between preventing oversmoothing and preserving the expressive power of deeper GNNs remains challenging. Indeed, recent evidence suggests that merely suppressing oversmoothing without enhancing a model's capacity may still limit its ultimate performance [61]. We refer interested readers to the recent survey by Rusch et al. [61] for a more detailed discussion.

Limited Expressiveness. A pedagogical example of the shortcomings of existing positional encodings described in Section 2 is the use of Laplacian eigenvectors (e.g. [23]), which can produce multiple valid solutions depending on the chosen sign or basis, and may fail to distinguish co-spectral but non-isomorphic graphs. This limitation motivates a body of existing encoding approaches such as learned [50] and heuristic-based [21] eigenvector canonization. However, as described in the main manuscript, designing encodings based on topology that are unique, distance sensitive, and equivariant in polynomial time may be impossible due to its relatedness to graph canonization [5]. We refer interested readers to the recent survey by Morris et al. [53] for a more comprehensive overview of positional encodings for GNNs.

As described in the main manuscript, recent work presented by Kanatsoulis et al. [36] instead aim to *learn* positional encodings using a message passing module that is trained end-to-end with the main

[5]We credit Di Giovanni et al. [20] for unifying these prior approaches and offering a comprehensive overview.

("backbone") GNN. The PEARL methodology begins by anonymizing the input graph by stripping away all node and edge attributes. For each node in the graph, the framework generates M random (R-PEARL) or basis (B-PEARL) attributes. These generated samples are then individually processed through a GNN and its outputs are combined using a pooling function ρ to create (equivariant) positional encodings. In the final stage, the framework processes the graph structure alongside the generated PEs and any existing node or graph attributes, using either a GNN or a Graph Transformer for the final analysis. Thus, PEARL's positional encodings enjoy better asympotic complexity than many classical (e.g. spectral-based) positional encoding methods, including a linear, rather than cubic cost, in the number of nodes in the input graph. However, unlike these classical methods, PEARL pays this $\mathcal{O}(n)$ cost for each graph at every epoch of training.

A.2 Complexity Analysis

SRF Complexity and SRMs. We analyze the runtime complexity of computing Sketched Random Features (SRF) when using structured random matrices (SRMs) as the sketching operator, as introduced in Section 3 of the main manuscript. SRMs, introduced by Choromanski et al. [15], are designed to accelerate Johnson–Lindenstrauss-style projections while preserving statistical properties of dense Gaussian matrices. Specifically, SRMs such as SD-product matrices (e.g., Hadamard–Rademacher constructions) admit fast matrix-vector multiplication in time $\mathcal{O}(N \log N)$, rather than the $\mathcal{O}(N^2)$ cost incurred by unstructured dense matrices. When applied to a full matrix $\Phi \in \mathbb{R}^{N \times D}$ consisting of D feature vectors, the corresponding matrix-matrix multiplication with a SRM costs $\mathcal{O}(N^2 \log N)$.

Our SRF construction proceeds in two stages. Given an input feature matrix $X \in \mathbb{R}^{N \times F}$, we first compute kernel embeddings by applying a random feature map $\phi : \mathbb{R}^F \to \mathbb{R}^D$ (e.g., random Fourier features [57]) to each row of X, yielding the matrix $\Phi = E(X) \in \mathbb{R}^{N \times D}$. This step has complexity $\mathcal{O}(NFD)$ under the assumption that ϕ involves dense linear projections.

In the second stage, we apply a sketching operator $S^{(k)}$ consisting of k independent SRMs to Φ, producing the final sketched matrix $Z = S^{(k)}(\Phi) \in \mathbb{R}^{N \times kD}$. Since each SRM projection costs $\mathcal{O}(N^2 \log N)$, the total cost across k independent sketches is $\mathcal{O}(kN^2 \log N)$. Combining both stages, the overall runtime complexity of SRF is:

$$\mathcal{O}(NFD + kN^2 \log N).$$

This is asymptotically more efficient than using dense Gaussian projections, which incur a cost of $\mathcal{O}(kN^3)$, while preserving similar guarantees on distance preservation and kernel approximation [15].

Finally, in the context of our broader complexity analysis, which considers scaling with respect to the number of nodes N and the number of training epochs T, we observe that SRF features are computed once at initialization and reused throughout training. Thus, like spectral encodings, SRF contributes a one-time cost and does not scale with T. The dominant term is the ROM projection, yielding an effective runtime complexity of $\mathcal{O}(N^2 \log N)$ and memory complexity of $\mathcal{O}(N)$ with respect to the graph size.

Additional Discussion on Computational Complexity. We analyze the computational and memory complexity of baseline methods described in Sections 3 and 4. Table 4 presents the complexity of these methods with respect to the number of nodes N. The analysis distinguishes between methods that require preprocessing versus those that perform computations during each forward pass. Preprocessing methods like random id [1] and SRF (ours) incur their computational cost once before training, then impose minimal overhead per epoch, while other methods perform their stated complexity computations during each forward pass. In scenarios where graphs are processed over many training epochs, preprocessing approaches achieve computational advantages as the number of epochs approaches or exceeds the graph size. Many graph learning benchmarks commonly found in the literature, including those used to evaluate PEARL and SPE, involve small graphs processed over many training epochs, favoring preprocessing approaches. Runtime benchmarks in Figure 2 complement this complexity analysis, demonstrating that SRF achieves superior computational efficiency in such scenarios. This efficiency advantage is also partially attributed to PEARL's requirement for a large number of graph samples that we empirically observe often scale with the number of graphs in the dataset. In scenarios where the required sample count is substantially smaller than the number of nodes, PEARL's empirical runtime would likely be comparatively more favorable. Future work should empirically validate this hypothesis.

Table 4: Comparison of computational and memory complexity for various GNN positional encoding methods with respect to number of nodes N. Methods marked with * incur costs once before training, while others incur costs per forward pass. When the number of training epochs is comparable to graph size, preprocessing methods achieve computational advantages.

Method	Computational Complexity	Memory Complexity
Random id*	$\mathcal{O}(N)$	$\mathcal{O}(N)$
SignNet	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$
SignNet-8S	$\mathcal{O}(N^3)$	$\mathcal{O}(N)$
SignNet-8L	$\mathcal{O}(N)$	$\mathcal{O}(N)$
BasisNet	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$
SPE	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$
SPE-8S	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$
R-PEARL	$\mathcal{O}(N)$	$\mathcal{O}(N)$
B-PEARL	$\mathcal{O}(N^2)$	$\mathcal{O}(N)$
SRF (ours)*	$\mathcal{O}(N^2 \log N)$	$\mathcal{O}(N)$

A.3 Additional Discussion on Baselines

For completeness, we describe the key methodological components of baseline encoding approaches referenced throughout the main manuscript. A qualitative summary of the differences between different baseline methods and SRF is provided in Table 5.

SignNet and BasisNet. Lim et al. [45] present SignNet and BasisNet, neural architectures that solve the sign and basis ambiguity problems in spectral graph positional encodings. SignNet addresses sign invariance by parameterizing functions of the form $f(v_1, \ldots, v_k) = \rho\left([\phi(v_i) + \phi(-v_i)]_{i=1}^k\right)$, where the structure $\phi(v_i) + \phi(-v_i)$ ensures invariance to sign flips of each eigenvector v_i, and ϕ and ρ are neural networks (e.g., GIN and MLP respectively). BasisNet handles basis invariance in higher-dimensional eigenspaces by computing $f(V_1, \ldots, V_l) = \rho\left([\text{IGN}_{d_i}(V_i V_i^T)]_{i=1}^l\right)$, where $V_i \in \mathbb{R}^{n \times d_i}$ are orthonormal bases of eigenspaces, the mapping $V \mapsto VV^T$ produces the orthogonal projector which is invariant to basis changes VQ for orthogonal Q, and IGNs (Invariant Graph Networks) process the resulting matrices while maintaining permutation equivariance. Both methods can incorporate eigenvalues and node features as additional inputs, and the processed eigenvectors are concatenated with original node features before being fed to downstream prediction models. Kanatsoulis et al. [36] introduce efficient variants: SignNet-8S and BasisNet-8S utilize the 8 smallest eigenvalues while maintaining $O(N^3)$ computational complexity, with SignNet-8S achieving reduced $O(N)$ memory complexity compared to the original $O(N^2)$. SignNet-8L uses the 8 largest eigenvalues, achieving both $O(N)$ computational and memory complexity.

Hybrid Models and Miscellanies. GPS [58] proposes a modular Graph Transformer framework that decouples local message-passing from global attention, combining three components: positional/structural encodings, local aggregation mechanisms, and global attention in a unified architecture. SAN [41] uses a learned positional encoding based on the Laplacian spectrum, which is added to node features before processing with a fully-connected Transformer. GNN-AK+ [71] extends standard MPNNs by replacing star-pattern aggregation with general subgraph pattern aggregation, where each node representation is computed from an induced subgraph encoding rather than just immediate neighbors. Graph ViT [29] adapts the Vision Transformer architecture, achieving linear complexity while capturing long-range dependencies and mitigating over-squashing through global receptive fields. SUN [24] combines subgraph-based methods with neural networks to enhance expressivity beyond traditional message-passing limitations.

B GNN Architecture Analysis

To verify that our approach generalizes across different GNN architectures, we evaluate SRF with three additional backbone architectures: Graph Convolutional Networks (GCN) [40], Graph Attention Networks (GAT)[66], and Graph Attention Networks v2 (GATv2)[10]. We compare against random

Table 5: Qualitative comparison of SRF to common baselines. Properties summarize typical behavior discussed in the main text and Appendix. Entries consolidate theoretical and empirical properties established or referenced throughout the manuscript. †Denotes families of methods; specific variants may differ in details, but the stated properties reflect their typical guarantees and practical behavior.

Method	Unique Representation?	Distance Sensitive?	Invariant or Equivariant?	Mitigates Oversquashing?	Alleviates Oversmoothing?
Random Node Features	Almost surely	No	In expectation	No	No
Spectral Encodings	No	Yes	Yes	No	No
Subgraph encodings†	No	No	Yes	No	Unclear
Homomorphism counts†	No	No	Yes	No	Unclear
PEARL	Unclear	With high probability	Yes	No	No
SRF (Ours)	**Almost surely**	**With high probability**	**In expectation**	**Yes**	**Yes**

feature injection baselines that maintain the same total number of learnable parameters. Following Abboud et al. [1], we inject random features, but unlike their approach, we perform reinjection at each layer to provide a fair comparison with our method's parameter count. All experiments use the hyperparameter configurations of the GIN/E models for which results are presented in Section 4.

Table 6 presents the results on REDDIT-B and REDDIT-M datasets. Across all architectures, most SRF variants consistently outperform random feature baselines, demonstrating that the performance gains are not dependent on the specific choice of GNN backbone. However, we find that random features do slightly outperform a single variant, $(\mathcal{E}_{\text{RBF}}, \mathcal{S}^{(8)}_{\text{AG}})$ on one experiment (REDDIT-B dataset and GAT backbone). Interestingly, relative performance differences between SRF variants across architectures, are slightly different than in the trend found in Section 4.2, with $(\mathcal{E}_{\text{linear}}, \mathcal{S}^{(8)}_{\text{AG}})$ achieving slightly stronger comparative performance for some dataset/backbone pairs. Future work should investigate whether these performance variations arise from architecture-specific hyperparameter sensitivities or inherent compatibility differences between SRF variants and GNN backbones.

C Experimental Details

C.1 Dataset Descriptions

Here, we describe properties of the real world graph datasetrs used in our experiments. All datasets are used in accordance with their respective licenses and terms of use.

Reddit-B and Reddit-M. The Reddit datasets [69] consist of graph classification tasks where each graph represents an online discussion thread. Nodes correspond to users and edges indicate response relationships between users' comments. Reddit-B contains 2,000 graphs across 2 classes with an average of 429.6 nodes per graph, where the task is to classify discussion threads as belonging to question/answer-based communities versus discussion-based communities. Reddit-M contains 5,000 graphs across 5 classes with an average of 508.5 nodes per graph, where the task is to predict which specific subreddit a discussion thread belongs to.

DrugOOD. The DrugOOD dataset [33] evaluates models on out-of-distribution generalization for drug-target binding affinity prediction. The dataset focuses on domain shifts arising from different bioassays (Assay), molecular scaffolds (Scaffold), and molecular sizes (Size), testing the ability to generalize to unseen experimental conditions, molecular structures, and compound sizes respectively. Each sample consists of paired protein-compound data with binary classification labels indicating binding activity.

Peptides-struct. The Peptides-struct dataset [22] from the Long Range Graph Benchmark comprises over 15,000 molecular graphs with more than 2 million nodes total, where individual graphs range from 8 to 444 nodes. The dataset involves multi-label regression on 3D structural properties of peptides, including inertia, length, sphericity, and geometric fit measures. Graphs are constructed with heavy atoms as nodes and chemical bonds as edges, requiring models to capture long-range interactions without explicit 3D coordinate information.

Table 6: Performance (Accuracy %) comparison of SRF variants across different GNN architectures on Reddit datasets. Random feature injection is a parameter-matched baseline with reinjection at each layer.

Architecture	Method	REDDIT-M	REDDIT-B
GCN	Random Features	53.46 ± 1.00	91.60 ± 1.04
	$(\mathcal{E}_{\text{linear}}, \mathcal{S}_{\text{AG}}^{(8)})$	58.04 ± 1.00	93.30 ± 0.72
	$(\mathcal{E}_{\mathcal{L}}, \mathcal{S}_{\text{AG}}^{(8)})$	57.20 ± 0.90	93.30 ± 0.72
	$(\mathcal{E}_{\text{RBF}}, \mathcal{S}_{\text{AG}}^{(8)})$	58.08 ± 0.89	93.05 ± 0.16
GAT	Random Features	47.75 ± 0.84	86.55 ± 1.89
	$(\mathcal{E}_{\text{linear}}, \mathcal{S}_{\text{AG}}^{(8)})$	51.40 ± 1.04	89.05 ± 1.17
	$(\mathcal{E}_{\mathcal{L}}, \mathcal{S}_{\text{AG}}^{(8)})$	49.64 ± 0.96	88.15 ± 1.43
	$(\mathcal{E}_{\text{RBF}}, \mathcal{S}_{\text{AG}}^{(8)})$	49.72 ± 1.02	86.15 ± 1.43
GATv2	Random Features	46.88 ± 0.89	84.2 ± 0.76
	$(\mathcal{E}_{\text{linear}}, \mathcal{S}_{\text{AG}}^{(8)})$	52.04 ± 0.56	87.4 ± 0.74
	$(\mathcal{E}_{\mathcal{L}}, \mathcal{S}_{\text{AG}}^{(8)})$	48.76 ± 0.89	87.6 ± 0.42
	$(\mathcal{E}_{\text{RBF}}, \mathcal{S}_{\text{AG}}^{(8)})$	48.40 ± 1.27	86.7 ± 1.30

C.2 Training Details and Hyperparameter Search Procedure

All experiments use SRF with parameters $k = 8$ and search over the SRF hyperparameter $D \cdot k \in \{16, 32, 64, 128, 256\}$ (See Section 3). Hyperparameter optimization is conducted using Weights and Biases across all datasets. The Adam optimizer is used throughout all experiments.

Reddit Datasets. Following standard practice [36], we employ 10-fold cross-validation and report results for the best epoch across 330 training epochs. We report mean and standard deviation across folds. We use CrossEntropy loss with sum pooling. The hyperparameter grid search includes: number of layers $\in \{3, 4, 5, 6, 7, 8\}$, hidden dimensions $\in \{32, 64, 128, 256, 512\}$, batch size $\in \{16, 32, 64, 128\}$, and learning rate $\in [10^{-5}, 10^{-2}]$.

DrugOOD. We follow the experimental setup from [36], training for 150 epochs and reporting results on the out-of-distribution test set using L1 loss. The hyperparameter search includes: number of layers $\in \{3, 4, 5, 6, 7\}$, batch size $\in \{16, 32, 64, 128\}$, learning rate $\in [10^{-5}, 10^{-2}]$, layer normalization $\in \{\text{true}, \text{false}\}$, batch normalization $\in \{\text{true}, \text{false}\}$, and hidden dimensions $\in \{32, 64, 80, 90, 100, 110\}$.

Peptides-struct. Models are trained for 500 epochs using L1 loss with residual connections. The hyperparameter search includes: number of layers $\in \{4, 5, 6, 7, 8, 9\}$, hidden dimensions $\in \{70, 95, 105, 135, 150\}$, batch size $\in \{10, 25, 50, 75\}$, learning rate $\in [10^{-5}, 10^{-2}]$, layer normalization $\in \{\text{true}, \text{false}\}$, and batch normalization $\in \{\text{true}, \text{false}\}$. We additionally include the hyperparameters used by PEARL [36] in their evaluation, as we combine the methods (Section 4). Results are reported for $\mathcal{E}_{\textbf{linear}}$ sketch variant. Error bars as reported by Kanatsoulis et al. [36] are ± 0.001 for PEARL-B, PEARL-R, GPS, SUN, and SAN+RWSE; ± 0.002 for Graph ViT; ± 0.004 for SAN+LapPE; and 0.00 for GNN-AK+. The error bars for our variants are ± 0.004.

C.3 Hardware and Software Tools

Our graph processing and learning experiments utilize the open-source PyTorch Geometric library as the primary framework, with NetworkX serving as a supplementary tool for graph operations. To ensure reproducibility and accurate runtime evaluation, we include a catalog of all software dependencies and their specific versions in the Supplementary material. An anonymized implementation of our codebase is also made available in the Supplementary material. All performance evaluations were conducted using an AMD EPYC 7713 64-Core Processor running Red Hat Enterprise Linux 9.3

and a NVIDIA DGX A100 GPUs (80GB memory). At times during experimentation, a cluster of 8 such GPUs were used to run parallel experiments.

C.4 Limitations

While SRF provides a principled and efficient mechanism for enhancing GNNs with global, feature-based information, it also has several limitations. First, its effectiveness relies on the presence of informative node features; in domains where features are sparse, noisy, or absent, performance gains may be limited. Second, as SRF relies on randomized projections, its properties hold in expectation; in practice, this introduces variance across runs, although our empirical results indicate this effect is negligible. Third, SRF is inherently topology-agnostic and may fail to capture structural signals that positional encodings or spectral methods explicitly model, but as noted, SRF can be used to complement these approaches. Finally, SRF introduces modest computational overhead compared to a vanilla GNN, though it is much more efficient than spectral methods.

D Analysis of SRF Embedding Dimension and Projection Count

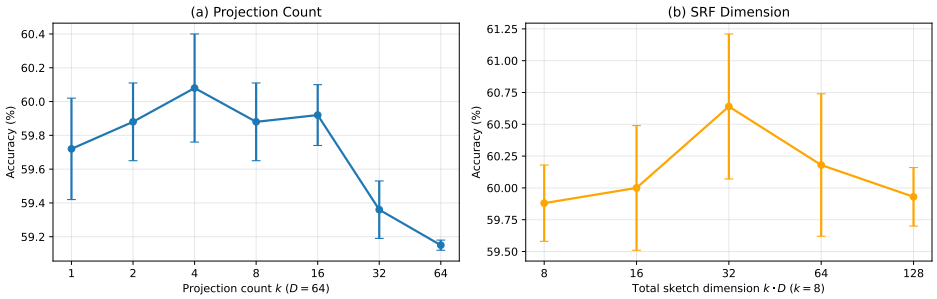

Figure 4: Hyperparameter sensitivity analysis on REDDIT-M using $(\mathcal{E}_{\text{linear}}, S_{AG}^{(k)})$. (a) Performance (accuracy %) vs. projection count k with $D = 64$ fixed. (b) Performance (accuracy %) vs. total sketch dimension $k \cdot D$ with $k = 8$ fixed.

The SRF framework introduces two hyperparameters that control the quality and computational cost of the sketched embeddings: embedding dimension D and projection count k 3. Understanding the sensitivity of SRF performance to these parameters is helpful for practical deployment.

The embedding dimension D governs the fidelity of the underlying kernel approximation as established in the main manuscript. Larger D provides more accurate kernel estimates but increases computational overhead. The projection count k determines the rank approximation of the random projection, with higher k potentially improving the quality of cross-node information encoding at the cost of increased overhead.

In this Appendix, we evaluate the impact of these hyperparameters with a simple case-study experiment. To isolate the effect of projection count k, we fix $D = 64$ and vary $k \in \{1, 2, 4, 8, 16, 32, 64\}$. To assess the impact of embedding dimension D while controlling for total sketch dimensionality, we fix the total sketched feature size $k \cdot D \in \{8, 16, 32, 64, 128\}$ and set $k = 8$, allowing D to vary accordingly.

Our hyperparameter sensitivity results are presented in Figure 4 for the REDDIT-M dataset using operator $(\mathcal{E}_{\text{linear}}, S_{AG}^{(k)})$. Due to computational limitations, we focus the analysis on this single dataset-operator combination.

Projection Count Analysis (Figure 4a). The results demonstrate a performance trade-off when varying the projection count k and maintaining fixed total sketch dimensionality. Performance peaks at $k = 4$ and subsequently declines as k increases. This pattern is likely due to an underlying trade-off between the number of independent projections and the quality of said individual projections: with fixed $D = 64$, increasing k provides more diverse cross-node information but at the cost of lower-dimensional kernel feature representations. The decline in performance at high k values suggests

that the degradation in kernel approximation quality eventually outweighs the benefits of additional projections.

SRF Dimension Analysis (Figure 4b). When varying the total sketch dimension $k \cdot D$ with fixed $k = 8$, we observe a modest performance increase from $k \cdot D = 8$ to $k \cdot D = 64$, followed by diminishing returns at higher dimensions. However, the substantial variance across runs suggest these differences may not be statistically significant. The performance decline at $k \cdot D = 128$ indicates that excessive sketch dimensionality may lead to overfitting on this moderately-sized dataset. Future work should investigate scaling properties more thoroughly on larger datasets where overfitting concerns are mitigated.

Practical Guidance on Hyperparameter Tuning. Based on our empirical study of SRF hyperparameters and the complexity analysis, we provide the following practical guidance for selecting SRF configurations. For kernel selection, we recommend RBF embeddings $\mathcal{E}_{\mathrm{RBF}}$ when computational constraints prevent extensive kernel sweeps. The projection count k typically exhibits an inverted-U effect on performance: accuracy improves up to a point before declining; across evaluated datasets we find $k = 4$ to be a reasonable default. Finally, increasing the total sketch dimension $k \cdot D$ improves performance with linear memory/runtime cost but shows diminishing returns; we suggest $k \cdot D = 64$ as a practical starting point, with adjustments based on dataset size and available compute.

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: Our analytical claims are justified in Section 3. Our empirical claims are demonstrated with experiments in Section 4.

   Guidelines:

   - The answer NA 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 NA 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: Limitations are explicitly discussed in Appendix C.

   Guidelines:

   - The answer NA 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: All propositions and accompanying proofs (Section 3) clearly state assumptions, are numbered, and cross referenced in the text.

Guidelines:

- The answer NA 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: We describe our method clearly and with detail in Section 3, including with an algorithm block (Algorithm 1). Our experiments (Section 4) also are described in detail, with additional details to facilitate reproducibility in Appendix C. Finally, we release open source code which is linked in the main manuscript.

   Guidelines:

   - The answer NA 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: All experiments were conducted on existing and publicly available datasets, with detailed descriptions in Appendix C. A link to our open source code is also available in the main manuscript.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.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://nips.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, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Training and test details necessary to understand the results are provided in Appendix C.

   Guidelines:

   - The answer NA 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: Relevant experiments report error bars, either in Section 4 or in Appendix C.

   Guidelines:

   - The answer NA 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: Information about computer resources used for experiments are in Appendix C.

   Guidelines:

   - The answer NA 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: Our work conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA 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: [No]

    Justification: Our work contributes to foundational research in machine learning and is not tied to particular applications. Thus, we follow the guidelines below and do not explicitly discuss societal impacts.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our work does not release new datasets or sensitive models.

    Guidelines:

    - The answer NA 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: Creators of assets, including code, data, and models, are credited via citation in the manuscript and licenses and terms are discussed for appropriate assets (e.g. datasets) in Appendix C.

    Guidelines:

    - The answer NA 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: All pertinent details of the models and code are described in Section 3, Appendices A, and C, and the instructions in our open source code repository.

Guidelines:

- The answer NA 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: [NA]

Justification: Our paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer NA 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: [NA]

Justification: Our paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer NA 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 rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: Our work does not involve LLMs.

Guidelines:

- The answer NA 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 (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.