UNROLLING MESSAGE-PASSING TO UNDERSTAND INFORMATION PROPAGATION IN GRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Message-passing neural networks (MPNNs) propagate information over graphs through repeated aggregation, yet the mechanisms governing signal preservation and degradation remain difficult to characterize exactly. We study this question through unrolled linear message-passing dynamics, where graph propagation is separated from nonlinear feature transformation. This separation makes it possible to derive exact node-to-node Jacobians and to measure two explicit propagation quantities: local sensitivity, which measures pairwise information transfer across the graph, and global sensitivity, which captures depth-induced collapse of information flow. Our analysis links propagation to graph topology and graph shift operators, identifying doubly stochastic learned operators as a principled way to prevent graph-side collapse in the deep regime. The theory also distinguishes topology-dependent degradation, related to over-squashing, from global vanishing caused by repeated nonlinear message passing. Guided by this analysis, we instantiate the theory in Bi-stream Unrolled Message Passing (BUMP), which combines a given graph operator with a learned doubly stochastic operator and fuses their representations after linear propagation. The framework applies to static and temporal graphs, and achieves strong empirical performance on long-range graph prediction, heterophilic node classification, and temporal graph forecasting.

1 INTRODUCTION

Graph Neural Networks (GNNs) (Wu et al., 2020a; Gravina & Bacciu, 2024), and in particular Message-Passing Neural Networks (MPNNs) (Gilmer et al., 2017), have become a standard paradigm for learning on graph-structured data. At their core, MPNNs iteratively propagate information along graph edges, enabling nodes to aggregate features from increasingly larger neighborhoods. Despite their broad adoption, the dynamics of information propagation in deep message-passing architectures remain poorly characterized. In the literature, challenges in effectively transferring information between nodes are often rationalized through notions such as over-squashing (Alon & Yahav, 2021; Di Giovanni et al., 2023) and vanishing gradients (Di Giovanni et al., 2023; Pascanu et al., 2013; Arroyo et al., 2025), which are commonly invoked as explanatory mechanisms for performance degradation arising from repeated aggregation. While useful for interpreting specific failure modes, these explanations do not, by themselves, constitute a unified or exact description of information propagation in graph-based models. A large body of work has sought to alleviate these limitations by directly targeting the symptoms of degraded message passing. Representative approaches include graph rewiring and curvature-based methods (Topping et al., 2022; Karhadkar et al., 2023; Gutteridge et al., 2023), transformer-style architectures (Kreuzer et al., 2021; Ying et al., 2021; Rampášek et al., 2022; Dwivedi et al., 2023; 2022a), and regularization strategies in weight space (Gravina et al., 2023; 2025). While often effective in practice, these methods are primarily designed as corrective interventions, and typically rely on substantial architectural modifications. As a consequence, they offer limited insight into the underlying dynamics that govern how information is propagated, preserved, or attenuated through repeated message passing. As a result, a principled and general understanding of information propagation in graph learning, independent of specific architectural remedies, remains elusive.

A motivating observation. To better understand the nature of the problem, we consider the task of learning node eccentricity in Table 1, a synthetic benchmark that requires information from the most distant nodes in the graph. Motivated by the role of propagation in

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

long-range graph tasks, we observe that a simple linear GCN, obtained by removing nonlinearities from a standard GCN (Kipf & Welling, 2016), substantially outperforms expressive architectures such as GAT (Veličković et al., 2018) or GraphGPS (Rampášek et al., 2022). This observation suggests that, in graph learning, the primary bottleneck may not be expressive power, but rather the *ability of message passing to propagate information reliably*. In particular, nonlinear transformations interleaved at every message-passing step may hinder stable information propagation, even when attention or global mechanisms are employed. Decoupling propagation from nonlinear transformation has already proved effective in static graph learning (Wu et al., 2019a; Gasteiger et al., 2019a). Crucially for our analysis, this decoupling makes the propagation dynamics analytically tractable: without intermediate nonlinearities, the linear recurrence can be unrolled exactly, enabling an explicit characterization of how information propagates across nodes and time. We therefore use the resulting recurrence as an analytical lens to study information propagation in both static and temporal settings.

Table 1: Learning the eccentricity of nodes in a graph.

	GCN	GAT	GPS	Linear GCN
Log(MSE) ↓	$0.8468_{\pm 0.0028}$	$0.7909_{\pm 0.0222}$	$0.6077_{\pm 0.0282}$	$-2.1424_{\pm 0.0014}$

From analysis to architecture. This perspective allows us to derive exact node-to-node recurrent-state Jacobians and to isolate the contribution of graph propagation from that of the channel dynamics. Our analysis characterizes how graph structure and operator choice affect information propagation under both repeated and time-varying composition, identifying conditions that prevent graph-side collapse. These results motivate *Bi-stream Unrolled Message Passing* (BUMP): one propagation stream retains the inductive bias of the observed graph, while a second learns an adaptive, doubly stochastic GSO designed to avoid graph-side collapse. The same formulation encompasses temporal graphs with fixed or evolving topology, while static graph learning is recovered as a special case.

We summarize our contributions as follows:

1. **Exact sensitivity theory for unrolled message passing.** We derive closed-form node-to-node Jacobians for fixed and time-varying unrolled graph dynamics, and use them to define local and global sensitivity as explicit measures of information propagation.
2. **Propagation guarantees and worst-cases.** We characterize when graph-side propagation remains stable or collapses, derive lower bounds on local and global sensitivity in the deep regime, and connect minimum local sensitivity to bottleneck topologies associated with over-squashing and minimum global sensitivity to vanishing gradients.
3. **Theory-driven bi-stream instantiation.** We introduce BUMP as a controlled architecture induced by the analysis: a given-graph stream preserves the original graph inductive bias, while a learned doubly stochastic stream provides an adaptive propagation route that is provably non-collapsing, even in time-varying settings.
4. **Extensive empirical validation.** Across 16 benchmarks spanning long-range graph prediction, heterophilic node classification, and temporal graph forecasting, BUMP consistently matches or outperforms state-of-the-art GNNs.

2 METHODOLOGY

In this section, we present Bi-stream Unrolled Message-Passing (BUMP), an architecture that instantiates the unrolled propagation dynamics analyzed in Section 3. In the following, we formalize the unrolled linear recurrence that enables effective information propagation, describe the dual-stream architecture that fuses fixed and learnable graph shift operators, and detail how this unified block structure scales to deep architectures for both static and temporal graph data.

Unrolled Linear Message-Passing. We denote the graph data as a sequence of input node features $[\mathbf{U}_t]_{t=1}^T$, with $\mathbf{U}_t \in \mathbb{R}^{n\times c'}$, with n the number of nodes, and c' the dimensionality of the input features. We note that, for static graphs, the sequence consists of a single element, i.e., $\mathbf{U}_1$ (as shown in the bottom-right of Figure 1). From the sequence of input states we obtain a sequence of hidden states $[\mathbf{X}_t]_{t=1}^T$, where $\mathbf{X}_t \in \mathbb{R}^{n\times c}$, with c the number of channels, via a linear message-passing scheme and channel mixing with learnable weight matrices $\mathbf{W} \in \mathbb{R}^{c\times c}, \mathbf{B} \in \mathbb{R}^{c'\times c}$ *shared across time steps*, as follows:

$$\mathbf{X}_{t+1} = \mathbf{A}_{t+1}\mathbf{X}_t\mathbf{W} + \mathbf{U}_{t+1}\mathbf{B}. \tag{1}$$

Equation (1) represents the linear recurrence on graphs, with $\mathbf{A}_t$ denoting the (possibly time-varying) GSO ruling the message-passing dynamics. The purely linear recurrent design choice enables both an exact sensitivity analysis (discussed in Section 3) and a fast parallel implementation (described in Appendix A). Specifically, setting the initial state $\mathbf{X}_0 = \mathbf{0} \in \mathbb{R}^{n\times c}$, Equation (1) can be unrolled

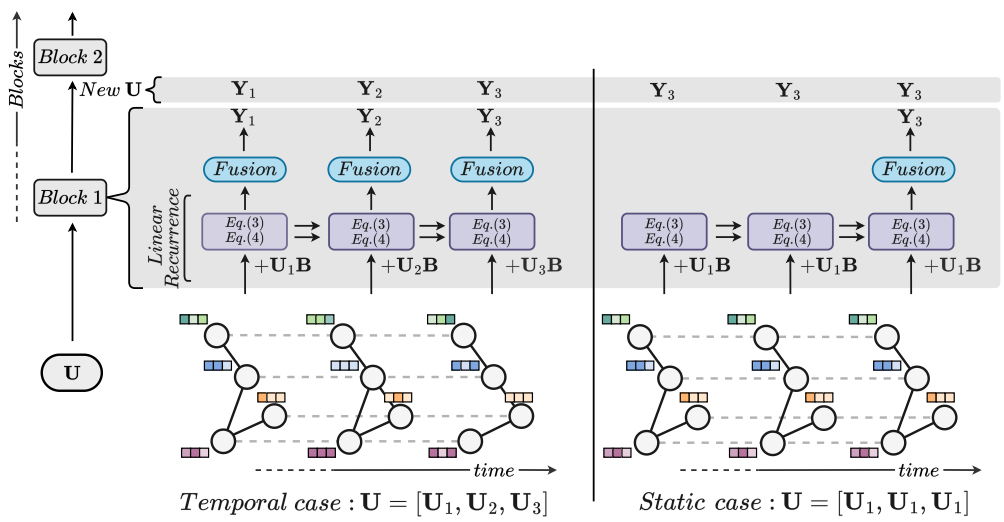

Figure 1: Illustration of BUMP for temporal and static cases, considering $k+1=3$ unrolled linear message-passing steps. In the temporal case, node embeddings evolve over time, $\mathbf{U}=[\mathbf{U}_1,\mathbf{U}_2,\mathbf{U}_3]$, whereas the static case uses fixed embeddings, $\mathbf{U}=[\mathbf{U}_1,\mathbf{U}_1,\mathbf{U}_1]$. Each block consists of two independent linear recurrences whose outputs are nonlinearly fused: one operates over the given graph structure (Equation (3)), while the other learns a latent structure directly from $\mathbf{U}$ (Equation (4)). Stacking multiple blocks yields a deep architecture.

exactly as follows:

$$\mathbf{X}_t = \mathbf{U}_t\mathbf{B} + \sum_{s=1}^{t-1}\big(\mathbf{A}_t\mathbf{A}_{t-1}\cdots\mathbf{A}_{t-s+1}\big)\mathbf{U}_{t-s}\mathbf{B}\mathbf{W}^s \tag{2}$$

and computed in a closed-form solution in a single step.

BUMP Block. A block of BUMP is designed to propagate information between nodes that are up to k hops apart, following the unrolled message-passing strategy defined in Equation (2). Specifically, we designed each block of BUMP as k iterations of two independent streams of linear message-passing dynamics, which define the core message-passing computations of a BUMP block and are given by the following two recurrent equations:

$$\mathbf{X}_{t+1} = \mathbf{A}_{t+1}\mathbf{X}_t\mathbf{W} + \mathbf{U}_{t+1}\mathbf{B}, \tag{3}$$

$$\mathbf{Z}_{t+1} = \hat{\mathbf{A}}_{t+1}\mathbf{Z}_t\mathbf{\Theta} + \mathbf{U}_{t+1}\mathbf{C}, \tag{4}$$

defined for $t=0,\ldots,k$. $\mathbf{A}_{t+1}$ denotes a given GSO (possibly time-varying) that preserves the intrinsic structure of the graph, $\hat{\mathbf{A}}_{t+1}=f(\mathbf{U}_{t+1})$ denotes a learnable GSO, and k a hyperparameter defining the depth of each BUMP block. We make the learnable GSO input-dependent but intentionally state-independent, which enables exact state-to-state Jacobian derivations and the formal sensitivity analysis developed in Section 3. We follow each iteration with a learnable graph-agnostic nonlinear mapping to fuse the two contributions. This simple node-wise fusion mechanism allows the model to flexibly balance between the original graph structure and a fully adaptive one:

$$\mathbf{Y}^{(1)}_{t+1} = \mathrm{MLP}^{(1)}(\mathbf{X}_{t+1}),\ \mathbf{Y}^{(2)}_{t+1} = \mathrm{MLP}^{(2)}(\mathbf{Z}_{t+1}) \tag{5}$$

$$[\boldsymbol{\alpha}^{(1)}_{t+1}\ \boldsymbol{\alpha}^{(2)}_{t+1}] = \mathrm{softmax}([\mathbf{Y}^{(1)}_{t+1},\mathbf{Y}^{(2)}_{t+1}]\mathbf{W}_\alpha) \tag{6}$$

$$\mathbf{Y}_{t+1} = \boldsymbol{\alpha}^{(1)}_{t+1}\odot\mathbf{Y}^{(1)}_{t+1} + \boldsymbol{\alpha}^{(2)}_{t+1}\odot\mathbf{Y}^{(2)}_{t+1}. \tag{7}$$

MLP denotes a multilayer perceptron of two dense layers with a nonlinearity in between them. $\mathbf{W}_\alpha\in\mathbb{R}^{2d\times 2}$ is a matrix weighting the concatenated nonlinear representation $[\mathbf{Y}^{(1)}_{t+1},\mathbf{Y}^{(2)}_{t+1}]\in\mathbb{R}^{n\times 2d}$. Both MLPs and the $\mathbf{W}_\alpha$ matrix are *shared across time steps*. The vectors $\boldsymbol{\alpha}^{(1)}_{t+1},\boldsymbol{\alpha}^{(2)}_{t+1}\in\mathbb{R}^{n\times 1}$ define a node-wise convex combination, allowing each node to adaptively emphasize the most informative representation between the two streams.

Unifying Static and Temporal Settings. In temporal graph settings, the input is naturally a sequence of graphs with time-varying features. Given a sequence $\mathbf{U} = [\mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_{k+1}]$, a single BUMP block produces an output sequence of the same length, $[\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_{k+1}]$, by iterating the linear message-passing steps in Equations (3) and (4) and applying the node-wise fusion mechanism (Equations (5) to (7)). For static graphs, we construct a sequence from a single input instance $\mathbf{U}_1$ in order to unify the treatment of the static and temporal cases. Specifically, we generate a constant input sequence $\mathbf{U} = [\mathbf{U}_1, \mathbf{U}_1, \ldots, \mathbf{U}_1]$ of length $k+1$. In this case, computation is more efficient: the fusion mechanism is applied only at the last time step of the block, and the final embedding serves directly as the input for the following block, which is $[\mathbf{Y}_{k+1}, \mathbf{Y}_{k+1}, \ldots, \mathbf{Y}_{k+1}]$. Figure 1 illustrates and summarizes the modes of operation described above.

The Deep BUMP Architecture. We can build a hierarchy of representations by constructing a deep BUMP architecture composed of stacked BUMP blocks. We use the output of a BUMP block as an input for the next one. We visually summarize this concept at the top of Figure 1. We note that, stacking multiple BUMP blocks allows the model to increase its effective aperture, aggregating information from further nodes. Specifically, the embedding $\mathbf{Y}_{k+1}$ encodes information aggregated up to the k-th hop. Therefore, stacking s BUMP blocks, each of depth k, allows to aggregate information from up to sk hops away. In Appendix E, we provide a multi-hop interpretation of a deep BUMP architecture, in the static case. For deep stacks of BUMP blocks, we can enhance the representation learning of our deep architecture through standard deep learning heuristics such as per-block residual connections (He et al., 2016) and normalization layers (Vaswani et al., 2017). Appendix I.1 presents an ablation study tracing the incremental impact of each BUMP component, progressing from an MPNN to a deep BUMP architecture, while Appendix H discusses its complexity and runtimes.

The Choice of the Graph Shift Operator. Our analysis motivates non-negative GSOs that support stable and non-vanishing propagation under repeated application. We focus on symmetric operators for analytical convenience, as symmetry enables a direct spectral characterization of the propagation dynamics, while not being necessary for stability or non-vanishing propagation. For the given graph stream, we use the symmetrically normalized adjacency matrix with self-loops, while for the learned stream we construct a symmetric doubly stochastic GSO. Different parameterizations satisfying these properties can be employed; the constructions considered in this work are detailed in Section B.

Table 2 summarizes the fixed/time-varying and given/learned GSO variants used in BUMP. In static graphs, each block combines a given fixed GSO with a learned fixed GSO; in temporal graphs, it combines a learned time-varying GSO with either a given fixed or time-varying GSO, depending on whether the observed topology changes over time.

Table 2: Possible GSO variants. Here, $\tilde{\mathbf{A}}$ denotes the adjacency matrix, while f_θ denotes a learnable parameterization of a symmetric stochastic GSO. Different parameterizations of f_θ are detailed in Section B.

GSO	Fixed	Time-varying
Given	(GF) $\mathbf{A} = \mathbf{D}^{-\frac{1}{2}}(\tilde{\mathbf{A}} + \mathbf{I})\mathbf{D}^{-\frac{1}{2}}$	(GTV) $\mathbf{A}_t = \mathbf{D}_t^{-\frac{1}{2}}(\tilde{\mathbf{A}}_t + \mathbf{I})\mathbf{D}_t^{-\frac{1}{2}}$
Learned	(LF) $\hat{\mathbf{A}} = f_\theta(\mathbf{U})$	(LTV) $\hat{\mathbf{A}}_t = f_\theta(\mathbf{U}_t)$
Bi-stream	Static: (GF) + (LF)	Temporal: (GF or GTV) + (LTV)

3 Information Propagation Analysis

In this section, we develop a theoretical analysis of the propagation capabilities of BUMP by characterizing the behavior of each propagation stream (i.e., Equations (3) and (4)) in static and temporal scenarios. To simplify notation, we denote a fixed GSO per block (i.e., GF and LF) as $\mathbf{A}$, while a time-varying GSO that changes at each step l within a block (i.e., GTV and LTV) as $\mathbf{A}_l$.

Following (Topping et al., 2022; Di Giovanni et al., 2023), we analyze information propagation via the Jacobians of the node representations, capturing the sensitivity of each node's representation to other nodes at previous time steps. We define two quantities computable from the Jacobians of node embeddings that link classical dynamical system notions, such as asymptotic stability, with phenomena commonly discussed in graph learning, such as vanishing gradients. This approach allows us to rigorously study how the unrolled linear message-passing dynamics propagate signals across nodes, both in static and temporal scenarios. Furthermore, we demonstrate that from first principles it is possible to construct example topologies that realize worst-case scenarios for information transfer. These topologies correspond closely to bottleneck structures, which are often associated with over-squashing in the literature. Additional analyses, including the role of nonlinearities at each recurrent step, are presented in Appendix D.

Notably, the fusion mechanism in a BUMP block (Equations (5) to (7)) defines a graph-agnostic convex combination of the representations coming from the two linear streams. This observation motivates our focus for sensitivity analysis on the linear recurrent equations within an BUMP block (Equations (3) and (4)), as they encapsulate the core dynamics relevant to information propagation on graphs.

Let $\mathbf{X}_s^{(j)}$ and $\mathbf{X}_t^{(i)}$ denote the embeddings of nodes j and i at time steps $s \leq t$. Similarly to Topping et al. (2022); Di Giovanni et al. (2023), we define:

Definition 3.1 (Local sensitivity)**.** The *local sensitivity* of the i-th node features to features of the j-th node, after $t - s$ applications of message-passing steps, is defined as the following spectral norm:

$$\mathcal{S}_{ij}[s:t] = \left|\left| \frac{\partial \mathbf{X}_t^{(i)}}{\partial \mathbf{X}_s^{(j)}} \right|\right|. \tag{8}$$

Equation (8) measures the influence of node j's features at time s on node i at time t.

Remark 3.2. If the local sensitivity between two nodes increases exponentially with $t - s$, then the learning dynamics of the MPNN is unstable; that is the typical case for linear MPNNs using the adjacency matrix or the Laplacian without any normalization Hariri et al. (2025). Therefore, **upper bounds on local sensitivity are linked with stable message propagation, in the deep regime**.

By unrolling the linear message-passing dynamics of a BUMP block, we obtain an explicit expression for the Jacobian between nodes j and i at times $s \leq t$, in terms of powers of the GSO. This relationship is formalized in Theorem 3.3; the proof is provided in Appendix C.1.

Theorem 3.3 (Exact Jacobian computation in BUMP)**.** *The Jacobian of the unrolled linear message-passing dynamics of a BUMP block, from node j at layer s to node i at layer $t \geq s$, can be computed exactly, and it has the following form:*

$$(GF, LF): \frac{\partial \mathbf{X}_t^{(i)}}{\partial \mathbf{X}_s^{(j)}} = \underbrace{(\mathbf{A}^{t-s})_{ij}}_{\textit{scalar}} \underbrace{(\mathbf{W}^\top)^{t-s}}_{\textit{matrix}}, \tag{9}$$

$$(GTV, LTV): \frac{\partial \mathbf{X}_t^{(i)}}{\partial \mathbf{X}_s^{(j)}} = \underbrace{\left(\prod_{l=s+1}^{t} \mathbf{A}_l\right)_{ij}}_{\textit{scalar}} \underbrace{(\mathbf{W}^\top)^{t-s}}_{\textit{matrix}} \tag{10}$$

This exact factorization is enabled by the state-independence of the GSO: learned GSOs may depend on the input $\mathbf{U}_t$, but not on the recurrent state $\mathbf{X}_t$.

These expressions separate graph-side and channel-side propagation. The scalar term $(\mathbf{A}^{t-s})_{ij}$, or its time-varying analogue, neatly determines how the GSO transfers influence from node j to node i. Consequently, GSOs that yield a bounded outcome under iterative multiplication promote stable BUMP dynamics, as highlighted in Remark 3.2. For the fixed-topology case (i.e., GF and LF), the operator $\mathbf{A}^t$ must be stable, while for the time-varying case (i.e., GTV and LTV), the operator $\mathbf{A}_{[0:t]} = \prod_{l=1}^{t} \mathbf{A}_l$ must be stable. In Theorem 3.4, we formally prove (see Appendix C.2) that all the possible GSOs that we use in BUMP exhibit this stability property.

Theorem 3.4 (Stability of the GSO)**.** *For GF and LF, the spectral radius $\rho(\mathbf{A}^t) = 1$ for every $t \geq 1$; for LTV, $\rho(\mathbf{A}_{[0:t]}) = 1$. Consequently, these powers and products neither diverge nor converge to the null matrix. In contrast, for GTV, $\rho(\mathbf{A}_{[0:t]}) \leq 1$ and the product remains bounded, but it may converge to the null matrix as $t \to \infty$.*

Theorem 3.4 implies that the GSO options ensure stable dynamics when performing a large number of message-passing operations in the BUMP's framework. For the case of time-varying GSO with very deep unrolled message-passing dynamics, the GTV stream may vanish information, but fusion with the LTV stream ensures information is preserved due to its provably non-vanishing behaviour. For the case of fixed GSO, in the deep regime, we can derive precise approximations and lower bounds of the local sensitivity, as stated in Theorem 3.5 and proved in Appendix C.3.

Theorem 3.5 (Deep unfolding approximation and lower bound local sensitivity)**.** *Assume an undirected and connected graph with n nodes and m edges. Then, for large values of $t - s$, the Jacobian of the unrolled linear message-passing dynamics of a BUMP block, from node j at layer s to node i*

at layer $t \geq s$, admits the following approximation:

$$(GF): \frac{\partial \mathbf{X}_t^{(i)}}{\partial \mathbf{X}_s^{(j)}} \approx \frac{\sqrt{(1+d_i)(1+d_j)}}{n+2m} \, (\mathbf{W}^\top)^{t-s}, \tag{11}$$

$$(LF): \frac{\partial \mathbf{X}_t^{(i)}}{\partial \mathbf{X}_s^{(j)}} \approx \frac{1}{n} \, (\mathbf{W}^\top)^{t-s}, \tag{12}$$

where d_l is the degree of the l-th node. Consequently, in the deep unfolding regime, the following lower bounds for the minimum local sensitivity hold:

$$(GF): \frac{2}{n+2m}||\mathbf{W}^{t-s}|| \leq \min_{i,j} \mathcal{S}_{ij}[s:t], \tag{13}$$

$$(LF): \frac{1}{n}||\mathbf{W}^{t-s}|| \leq \min_{i,j} \mathcal{S}_{ij}[s:t]. \tag{14}$$

Remark 3.6 (Notes on Over-Squashing). The minimum local sensitivity is realized for pairs of nodes among which the transfer of information is the most critical due to the structure of the graph. Theorem 3.5 allows us to quantify how different choices of GSO influence these lower bounds. In Appendix C.3.1, we use these insights to construct a topology that approaches the lower bound of Equation (13) for the GF case, realizing a worst-case scenario: a chain of cliques. This effect is intrinsically tied to the specific topology of the graph, and it aligns with prior studies that emphasize the challenges of learning on graphs with bottleneck structures (Topping et al., 2022; Mishayev et al., 2025). Hence, **lower bounds on the minimum local sensitivity provide a principled way to assess structural limitations of the graph and their relationship to over-squashing**. For example, rewiring techniques are known to mitigate over-squashing (Di Giovanni et al., 2023) by overcoming bottlenecks, a phenomenon that is conceptually consistent with the increased lower bound derived in the LF case in Theorem 3.5.

To assess the overall gradient information flow across the entire graph in the deep regime, we define:

Definition 3.7 (Global sensitivity)**.** The *global sensitivity* of node features of the overall graph after $t-s$ hops of message aggregation is defined as:

$$\mathcal{S}[s:t] = \max_{i,j} \mathcal{S}_{ij}[s:t]. \tag{15}$$

Remark 3.8 (Notes on Vanishing Gradients). The local sensitivity between two far-apart nodes can be physiologically small due to the particular topology of the graph (e.g. bottlenecks), or it can be even 0 if two nodes are not connected by any walk. However, if the local sensitivity converges to 0, in the deep regime, for all the pairs of nodes, i.e., if the global sensitivity converges to 0 regardless of the particular topology of the graph, then it means that the MPNN model is characterized by a vanishing information flow. Therefore, **lower bounds on global sensitivity provide a principled measure of vanishing information flow in the deep regime, linking it to vanishing gradients**.

We can leverage the exact Jacobians of Theorem 3.3 to prove the following lower bound on the global sensitivity, see Appendix C.4 for the proof.

Theorem 3.9 (Lower bound global sensitivity)**.** *For the case of LF and LTV, the global sensitivity of the unrolled linear message-passing dynamics of a BUMP block is lower bounded as follows:*

$$\frac{1}{n}||\mathbf{W}^{t-s}|| \leq \mathcal{S}[s:t], \tag{16}$$

where n is the number of nodes. The same lower bound holds true for the case of GF by assuming an undirected and connected graph.

Notably, even in the case of a time-varying topology, where the linear stream of GTV may lead to vanishing information, still the presence of the LTV stream ensures an overall non-vanishing behaviour of the information propagation in the deep regime of BUMP. This theoretical result demonstrates that BUMP ensures values of the global sensitivity strictly greater than zero, for any depth $t-s$. This result cannot be guaranteed in a standard MPNN, as the nonlinearity applied at each time step increasingly contributes to vanish information as the depth increases. Appendix D presents a detailed analysis of vanishing-gradient effects in nonlinear MPNNs and quantitatively evaluates how nonlinearities at each recurrent step influence signal propagation, with a focus on their role in vanishing-gradient phenomena in the deep regime.

Table 4: Mean test set score and std averaged over 4 random weight initializations on heterophilic tasks. **First**, **second**, and **third** best results.

Model	Roman-empire Acc ↑	Amazon-ratings Acc ↑	Minesweep. AUC ↑	Tolokers AUC ↑	Questions AUC ↑
MPNNs					
CO-GNN	**91.57**$_{\pm 0.32}$	**54.17**$_{\pm 0.37}$	**97.31**$_{\pm 0.41}$	**84.45**$_{\pm 1.17}$	**80.02**$_{\pm 0.86}$
GAT	80.87$_{\pm 0.30}$	49.09$_{\pm 0.63}$	92.01$_{\pm 0.68}$	83.70$_{\pm 0.47}$	77.43$_{\pm 1.20}$
Gated-GCN	74.46$_{\pm 0.54}$	43.00$_{\pm 0.32}$	87.54$_{\pm 1.22}$	77.31$_{\pm 1.14}$	76.61$_{\pm 1.13}$
GCN	73.69$_{\pm 0.74}$	48.70$_{\pm 0.63}$	89.75$_{\pm 0.52}$	83.64$_{\pm 0.67}$	76.09$_{\pm 1.27}$
SAGE	85.74$_{\pm 0.67}$	53.63$_{\pm 0.39}$	**93.51**$_{\pm 0.57}$	82.43$_{\pm 0.44}$	76.44$_{\pm 0.62}$
Graph Transformers					
Exphormer	**89.03**$_{\pm 0.37}$	53.51$_{\pm 0.46}$	90.74$_{\pm 0.53}$	83.77$_{\pm 0.78}$	73.94$_{\pm 1.06}$
GOAT	71.59$_{\pm 1.25}$	44.61$_{\pm 0.50}$	81.09$_{\pm 1.02}$	83.11$_{\pm 1.04}$	75.76$_{\pm 1.66}$
GPS	82.00$_{\pm 0.61}$	53.10$_{\pm 0.42}$	90.63$_{\pm 0.67}$	83.71$_{\pm 0.48}$	71.73$_{\pm 1.47}$
GT	86.51$_{\pm 0.73}$	51.17$_{\pm 0.66}$	91.85$_{\pm 0.76}$	83.23$_{\pm 0.64}$	77.95$_{\pm 0.68}$
Heterophily-Designated GNNs					
FAGCN	65.22$_{\pm 0.56}$	44.12$_{\pm 0.30}$	88.17$_{\pm 0.73}$	77.75$_{\pm 1.05}$	77.24$_{\pm 1.26}$
FSGNN	79.92$_{\pm 0.56}$	52.74$_{\pm 0.83}$	90.08$_{\pm 0.70}$	82.76$_{\pm 0.61}$	**78.86**$_{\pm 0.92}$
GBK-GNN	74.57$_{\pm 0.47}$	45.98$_{\pm 0.71}$	90.85$_{\pm 0.58}$	81.01$_{\pm 0.67}$	74.47$_{\pm 0.86}$
GPR-GNN	64.85$_{\pm 0.27}$	44.88$_{\pm 0.34}$	86.24$_{\pm 0.61}$	72.94$_{\pm 0.97}$	55.48$_{\pm 0.91}$
H2GCN	60.11$_{\pm 0.52}$	36.47$_{\pm 0.23}$	89.71$_{\pm 0.31}$	73.35$_{\pm 1.01}$	63.59$_{\pm 1.46}$
Graph SSMs					
GMN	87.69$_{\pm 0.50}$	**54.07**$_{\pm 0.31}$	91.01$_{\pm 0.23}$	**84.52**$_{\pm 0.21}$	–
GPS+Mamba	83.10$_{\pm 0.28}$	45.13$_{\pm 0.97}$	89.93$_{\pm 0.54}$	83.70$_{\pm 1.05}$	–
Ours					
BUMP	**91.72**$_{\pm 0.35}$	**53.81**$_{\pm 0.79}$	**97.45**$_{\pm 0.73}$	**85.44**$_{\pm 1.00}$	**78.68**$_{\pm 1.42}$

4 Experiments

We evaluate BUMP on static and temporal graph benchmarks spanning long-range graph property prediction, heterophilic node classification, and temporal forecasting. For static graphs, we use synthetic graph property tasks from Gravina et al. (2023) (i.e., diameter, SSSP, and eccentricity) and five heterophilic benchmarks from Platonov et al. (2023): Roman-empire, Amazon-ratings, Minesweeper, Tolokers, and Questions. For temporal graphs, we use Metr-LA, PeMS-Bay (Li et al., 2018), Chickenpox Hungary, PedalMe London, Wikipedia Math (Rozemberczki et al., 2021), and Twitter Tennis RG (Rozemberczki et al., 2021). We follow the original splits, protocols, and hyperparameter settings for each benchmark; additional datasets, baselines, statistics, hyperparameters, runtime, and complexity results are reported in Appendices G, F.1, F.2, F.3, and H.

4.1 Graph Property Prediction

We reports diameter, single-source shortest path (SSSP), and eccentricity prediction in Table 3. These tasks are the closest empirical counterpart to the sensitivity analysis, because successful prediction requires signals from distant nodes to remain usable after many propagation steps. We also report Peptides-func and Peptides-struct in Appendix G. We treat these as complementary evidence, while noting that LRGB is not universally regarded as requiring genuinely long-range interactions (Tönshoff et al., 2023).

Table 3: Mean test set $log_{10}(\mathrm{MSE})(\downarrow)$ and std averaged on 4 random weight initializations on Graph Property Prediction tasks. The lower, the better. **First**, **second**, and **third** best results for each task are color-coded.

Model	Diameter	SSSP	Eccentricity
MPNNs			
A-DGN	-0.5188$_{\pm 0.1812}$	**-3.2417**$_{\pm 0.0751}$	0.4296$_{\pm 0.1003}$
DGC	0.6028$_{\pm 0.0050}$	-0.1483$_{\pm 0.0231}$	0.8261$_{\pm 0.0032}$
GAT	0.8221$_{\pm 0.0752}$	0.6951$_{\pm 0.1499}$	0.7909$_{\pm 0.0222}$
GCN	0.7424$_{\pm 0.0466}$	0.9499$_{\pm 0.0001}$	0.8468$_{\pm 0.0028}$
GCNII	0.5287$_{\pm 0.0570}$	-1.1329$_{\pm 0.0135}$	0.7640$_{\pm 0.0355}$
GIN	0.6131$_{\pm 0.0990}$	-0.5408$_{\pm 0.4193}$	0.9504$_{\pm 0.0007}$
GRAND	0.6715$_{\pm 0.0490}$	-0.0942$_{\pm 0.3897}$	0.6602$_{\pm 0.1393}$
GraphCON	0.0964$_{\pm 0.0620}$	-1.3836$_{\pm 0.0092}$	0.6833$_{\pm 0.0074}$
GraphSAGE	0.8645$_{\pm 0.0401}$	0.2863$_{\pm 0.1843}$	0.7863$_{\pm 0.0207}$
SGC	**-2.6497**$_{\pm 0.0333}$	−1.5822$_{\pm 0.0001}$	**-2.3798**$_{\pm 0.0126}$
APPNP	**-2.0407**$_{\pm 0.0754}$	−1.7302$_{\pm 0.0468}$	**-2.0605**$_{\pm 0.0583}$
Transformers			
GPS	-0.5121$_{\pm 0.0426}$	**-3.5990**$_{\pm 0.1949}$	0.6077$_{\pm 0.0282}$
Ours			
BUMP	**-3.3421**$_{\pm 0.0447}$	**-4.8994**$_{\pm 0.0306}$	**-2.8257**$_{\pm 0.3003}$

4.2 Heterophilic Benchmark

Table 4 shows that BUMP is broadly competitive on heterophilic node classi-

Table 5: Average MSE and standard deviation (↓) of 10 experimental repetitions. Baseline results are reported from Rozemberczki et al. (2021); Errica et al. (2023); Eliasof et al. (2024b) . **First**, **second**, and **third** best methods for each task are color-coded.

Model	Chickenpox Hungary	PedalMe London	Wikipedia Math
Temporal GNNs			
A3T-GCN	$1.114_{\pm 0.008}$	$1.469_{\pm 0.027}$	$0.781_{\pm 0.011}$
AGCRN	$1.120_{\pm 0.010}$	$1.469_{\pm 0.030}$	$0.788_{\pm 0.011}$
CDE	$\mathbf{0.848}_{\pm 0.020}$	$\mathbf{0.810}_{\pm 0.063}$	$0.694_{\pm 0.028}$
DCRNN	$1.124_{\pm 0.015}$	$1.463_{\pm 0.019}$	$0.679_{\pm 0.020}$
DyGrAE	$1.120_{\pm 0.021}$	$1.455_{\pm 0.031}$	$0.773_{\pm 0.009}$
DynGESN	$0.907_{\pm 0.007}$	$1.528_{\pm 0.063}$	$0.610_{\pm 0.003}$
EGCN-O	$1.124_{\pm 0.009}$	$1.491_{\pm 0.024}$	$0.750_{\pm 0.014}$
GConvGRU	$1.128_{\pm 0.011}$	$1.622_{\pm 0.032}$	$0.657_{\pm 0.015}$
GC-LSTM	$1.115_{\pm 0.014}$	$1.455_{\pm 0.023}$	$0.779_{\pm 0.023}$
GRAND	$1.068_{\pm 0.021}$	$1.557_{\pm 0.049}$	$0.798_{\pm 0.034}$
GREAD	$0.983_{\pm 0.027}$	$1.291_{\pm 0.055}$	$0.704_{\pm 0.016}$
HMM4G	$0.939_{\pm 0.013}$	$1.769_{\pm 0.370}$	$\mathbf{0.542}_{\pm 0.008}$
MPNN LSTM	$1.116_{\pm 0.023}$	$1.485_{\pm 0.028}$	$0.795_{\pm 0.010}$
TDE-GNN	$\mathbf{0.787}_{\pm 0.018}$	$\mathbf{0.714}_{\pm 0.051}$	$\mathbf{0.565}_{\pm 0.017}$
T-GCN	$1.117_{\pm 0.011}$	$1.479_{\pm 0.012}$	$0.764_{\pm 0.011}$
Ours			
BUMP	$\mathbf{0.730}_{\pm 0.026}$	$\mathbf{0.643}_{\pm 0.059}$	$\mathbf{0.497}_{\pm 0.014}$

fication, although specialized baselines can be stronger on individual datasets. This is consistent with our goal: BUMP is not designed as a heterophily-specific architecture, but as a general propagation mechanism whose learned stream can complement the given topology.

4.3 Temporal Graph Forecasting

Tables 5 and 6 show that BUMP outperforms existing temporal GNNs, confirming its ability to model both spatio and temporal dependencies, including the case of time-varying topologies, Table 7.

5 Related Works

Effective Propagation on Graphs. While GNNs effectively model local structures via message-passing, they struggle with effective information propagation due to over-squashing and vanishing gradients (Alon & Yahav, 2021; Di Giovanni et al., 2023; Arroyo et al., 2025). Standard models like GCN (Kipf & Welling, 2016), GraphSAGE (Hamilton et al., 2017), and GIN (Xu et al., 2019) suffer from degraded performance on tasks requiring global context (Baek et al., 2021; Dwivedi et al., 2022b), especially in heterophilic graphs (Luan et al., 2024; Wang et al., 2024b). Solutions include graph rewiring (Topping et al., 2022; Karhadkar et al., 2023), weight-space regularization (Gravina et al., 2023; 2025), and physics-inspired dynamics (Eliasof et al., 2024a; Heilig et al., 2025), as well as multiscale approaches via graph pooling Finder et al. (2025). However, such approaches typically require pre-defined components, from rewiring to pooling techniques. Graph Transformers (GTs) like SAN (Kreuzer et al., 2021), Graphormer (Ying et al., 2021), and GPS (Rampášek et al., 2022) enhance expressivity using structural encodings (Dwivedi et al., 2023; 2022a), but suffer from quadratic complexity. Scalable alternatives include sparse and linearized attention mechanisms (Zaheer et al., 2020; Choromanski et al., 2021; Shirzad et al., 2023; 2024; Wu et al., 2023; Deng et al., 2024), though simple MPNNs often remain competitive (Tönshoff et al., 2023). Finally, several recent works have explored the use of state-space models in graph learning, particularly focusing on capturing long-range interactions (Wang et al., 2024a; Behrouz & Hashemi, 2024; Huang et al., 2024).

Learning Spatio-Temporal Interactions on Graphs. Temporal GNNs often combine GNNs with RNNs to model spatio-temporal dynamics (Gravina & Bacciu, 2024). Some adopt stacked architectures that separate spatial and temporal processing (Seo et al., 2018; Pareja et al., 2020; Panagopoulos et al., 2021; Bai et al., 2021; Cini et al., 2023a), while others integrate GNNs within RNNs for joint modeling (Li et al., 2018; 2019; Chen et al., 2022; Cini et al., 2023b; Ruiz et al., 2020). Our approach follows the latter, but goes further by embracing the principle of decomposing information

Table 6: Multivariate time series forecasting on the Metr-LA and PeMS-Bay datasets for Horizon 12. **First**, **second**, and **third** best results for each task are color-coded. Baseline results are reported from (Shao et al., 2022; Liu et al., 2023; Gao et al., 2024; Fan et al., 2024; Zhang et al., 2024).

Model	Metr-LA			PeMS-Bay		
	MAE ↓	RMSE ↓	MAPE ↓	MAE ↓	RMSE ↓	MAPE ↓
Graph Agnostic						
HA	6.99	13.89	17.54%	3.31	7.54	7.65%
FC-LSTM	4.37	8.69	14.00%	2.37	4.96	5.70%
SVR	6.72	13.76	16.70%	3.28	7.08	8.00%
VAR	6.52	10.11	15.80%	2.93	5.44	6.50%
Temporal GNNs						
AdpSTGCN	3.40	7.21	**9.45%**	1.92	4.49	4.62%
ASTGCN	6.51	12.52	11.64%	2.61	5.42	6.00%
DCRNN	3.60	7.60	10.50%	2.07	4.74	4.90%
GMAN	3.44	7.35	10.07%	1.86	**4.32**	4.37%
Graph WaveNet	3.53	7.37	10.01%	1.95	4.52	4.63%
GTS	3.46	7.31	9.98%	1.95	4.43	4.58%
MTGNN	3.49	7.23	9.87%	1.94	4.49	4.53%
RGDAN	**3.26**	**7.02**	9.73%	1.82	**4.20**	4.28%
STAEformer	**3.34**	**7.02**	9.70%	1.88	4.34	4.41%
STD-MAE	3.40	7.07	**9.59%**	**1.77**	**4.20**	**4.17%**
STEP	3.37	**6.99**	9.61%	**1.79**	**4.20**	**4.18%**
STGCN	4.59	9.40	12.70%	2.49	5.69	5.79%
STSGCN	5.06	11.66	12.91%	2.26	5.21	5.40%
Temporal Graph SSMs						
GGRNN	3.88	8.14	10.59%	2.34	5.14	5.21%
GraphSSM-S4	3.74	7.90	10.37%	1.98	4.45	4.77%
Ours						
BUMP	**3.16**	**6.88**	**9.17%**	**1.60**	**4.19**	**4.08%**

Table 7: Mean test MSE and std averaged over 10 experimental repetitions on Twitter Tennis RG Benchmark. Baseline results are reported from (Rozemberczki et al., 2021).

DCRNN	GConvLSTM	DyGrAE	EGCN-H	T-GCN	AGCRN	BUMP (Ours)
$2.049_{\pm 0.023}$	$2.049_{\pm 0.024}$	$\mathbf{2.031}_{\pm 0.006}$	$2.040_{\pm 0.018}$	$2.045_{\pm 0.027}$	$\mathbf{2.039}_{\pm 0.022}$	$\mathbf{2.028}_{\pm 0.015}$

propagation unrolled for multiple linear message-passing steps and nonlinear transformations of the resulting representations, unifying spatial and temporal reasoning through linear recurrence.

6 Conclusions

We present a principled framework for analyzing and controlling information propagation in graph neural networks. By unrolling linear message-passing dynamics and decoupling propagation from nonlinear feature transformations, we introduce Bi-stream Unrolled Message-Passing (BUMP), a unified model for static and temporal graphs that enables exact theoretical analysis. This perspective casts deep graph learning as the study of powers and products of graph shift operators, revealing how spectral properties, topology, and architectural design jointly govern signal preservation across depth. Our theoretical analysis provides exact node-to-node sensitivity expressions for unrolled message passing, offering precise characterizations of over-squashing and vanishing gradients. We identify a desirable class of graph shift operators, e.g., doubly stochastic matrices, as a principled and practically realizable choice to ensure stable propagation and non-vanishing sensitivity even in the deep regime. The bi-stream design allows BUMP to retain these guarantees in both static and temporal settings through learned time-varying operators when needed. Empirically, BUMP achieves strong performance on long-range graph property prediction, heterophilic node classification, and temporal graph forecasting. Its linear unrolled blocks enable parallel implementation and linear scaling, while outperforming state-of-the-art GNNs, graph transformers, and temporal models on propagation-demanding tasks.

AI USE STATEMENT

Generative AI tools were solely used to aid in polishing the writing of the manuscript. All AI-assisted content was reviewed and verified by the authors. Generative AI was not used to formulate the research ideas, methodology, or scientific conclusions. The authors take full responsibility for the final content of this work.

ETHICS STATEMENT

This work introduces computational methods for graph representation learning and does not involve human-subject research or the collection of personal or sensitive information by the authors. All experiments are conducted using established, publicly available benchmark datasets. Although graph neural networks may be applied in domains that involve privacy, fairness, or societal risks, the present work focuses on general methodological development and benchmark evaluation. Any use in sensitive or high-impact settings should therefore be accompanied by an application-specific assessment of potential risks. The authors have read and adhered to the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

The main paper fully describes the proposed method and experimental protocol. The appendix provides further information about the architecture, implementation, hyperparameters, datasets, and experimental configurations. We also document the procedures necessary to reproduce the baseline comparisons, ablation studies, and evaluations. The source code and configuration files used in our experiments will be made publicly available upon acceptance.

REFERENCES

Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *International Conference on Machine Learning*, pp. 21–29. PMLR, 2019.

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.

Alvaro Arroyo, Alessio Gravina, Benjamin Gutteridge, Federico Barbero, Claudio Gallicchio, Xiaowen Dong, Michael M. Bronstein, and Pierre Vandergheynst. On vanishing gradients, over-smoothing, and over-squashing in GNNs: Bridging recurrent and graph learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=N4cyRMuLyl.

Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N Kinch, R Dustin Schaeffer, et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, 2021.

Jiandong Bai, Jiawei Zhu, Yujiao Song, Ling Zhao, Zhixiang Hou, Ronghua Du, and Haifeng Li. A3T-GCN: Attention Temporal Graph Convolutional Network for Traffic Forecasting. *ISPRS International Journal of Geo-Information*, 10(7), 2021. ISSN 2220-9964. doi: 10.3390/ijgi10070485.

Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. Adaptive graph convolutional recurrent network for traffic forecasting. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20. Curran Associates Inc., 2020. ISBN 9781713829546.

Anirban Banerjee and Ranjit Mehatari. An eigenvalue localization theorem for stochastic matrices and its application to randić matrices. *Linear Algebra and its Applications*, 505:85–96, 2016.

Ali Behrouz and Farnoosh Hashemi. Graph mamba: Towards learning on graphs with state space models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 119–130, 2024.

Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Graph neural networks with convolutional arma filters. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3496–3507, 2021.

Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5): 3950–3957, May 2021. doi: 10.1609/aaai.v35i5.16514. URL https://ojs.aaai.org/index.php/AAAI/article/view/16514.

Xavier Bresson and Thomas Laurent. Residual Gated Graph ConvNets. *arXiv preprint arXiv:1711.07553*, 2018.

Benjamin Paul Chamberlain, James Rowbottom, Maria Gorinova, Stefan Webb, Emanuele Rossi, and Michael M Bronstein. GRAND: Graph neural diffusion. In *International Conference on Machine Learning (ICML)*, pp. 1407–1418. PMLR, 2021.

Jinsong Chen, Kaiyuan Gao, Gaichao Li, and Kun He. NAGphormer: A tokenized graph transformer for node classification in large graphs. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=8KYeilT3Ow.

Jinyin Chen, Xueke Wang, and Xuanheng Xu. GC-LSTM: graph convolution embedded LSTM for dynamic network link prediction. *Applied Intelligence*, 52(7):7513–7528, May 2022. ISSN 1573-7497. doi: 10.1007/s10489-021-02518-9.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and Deep Graph Convolutional Networks. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1725–1735. PMLR, 13–18 Jul 2020.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=n6jl7fLxrP.

Jeongwhan Choi, Seoyoung Hong, Noseong Park, and Sung-Bae Cho. Gread: graph neural reaction-diffusion networks. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ua6zuk0WRH.

Andrea Cini, Ivan Marisca, Filippo Maria Bianchi, and Cesare Alippi. Scalable Spatiotemporal Graph Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023a.

Andrea Cini, Ivan Marisca, Daniele Zambon, and Cesare Alippi. Taming local effects in graph-based spatiotemporal forecasting. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 55375–55393. Curran Associates, Inc., 2023b.

Chenhui Deng, Zichao Yue, and Zhiru Zhang. Polynormer: Polynomial-expressive graph transformer in linear time. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=hmv1LpNfXa.

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*, pp. 7865–7885. PMLR, 2023.

Yuhui Ding, Antonio Orvieto, Bobby He, and Thomas Hofmann. Recurrent distance filtering for graph representation learning. In *Forty-first International Conference on Machine Learning*, 2024.

Lun Du, Xiaozhou Shi, Qiang Fu, Xiaojun Ma, Hengyu Liu, Shi Han, and Dongmei Zhang. Gbk-gnn: Gated bi-kernel graph neural networks for modeling both homophily and heterophily. In *Proceedings of the ACM Web Conference 2022*, WWW '22, pp. 1550–1558, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450390965. doi: 10.1145/3485447.3512201. URL https://doi.org/10.1145/3485447.3512201.

Vijay Prakash Dwivedi and Xavier Bresson. A Generalization of Transformer Networks to Graphs. *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*, 2021.

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

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, 2022b.

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.

Moshe Eliasof, Eldad Haber, and Eran Treister. Feature transportation improves graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 11874–11882, 2024a.

Moshe Eliasof, Eldad Haber, Eran Treister, and Carola-Bibiane B Schönlieb. On the temporal domain of differential equation inspired graph neural networks. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li (eds.), *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pp. 1792–1800. PMLR, 02–04 May 2024b.

Federico Errica, Alessio Gravina, Davide Bacciu, and Alessio Micheli. Hidden markov models for temporal graph representation learning. In *Proceedings of the 31st European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, 2023.

Jin Fan, Wenchao Weng, Hao Tian, Huifeng Wu, Fu Zhu, and Jia Wu. Rgdan: A random graph diffusion attention network for traffic prediction. *Neural Networks*, 172:106093, 2024. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2023.106093.

Shahaf E. Finder, Ron Shapira Weber, Moshe Eliasof, Oren Freifeld, and Eran Treister. Improving the effective receptive field of message-passing neural networks. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=QJLGj57MfZ.

Ben Finkelshtein, Xingyue Huang, Michael M. Bronstein, and Ismail Ilkan Ceylan. Cooperative Graph Neural Networks. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=ZQcqXCuoxD.

Haotian Gao, Renhe Jiang, Zheng Dong, Jinliang Deng, Yuxin Ma, and Xuan Song. Spatial-temporal-decoupled masked pre-training for spatiotemporal forecasting. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, IJCAI '24, 2024. ISBN 978-1-956792-04-1. doi: 10.24963/ijcai.2024/442.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Combining neural networks with personalized pagerank for classification on graphs. In *International Conference on Learning Representations*, 2019a. URL https://openreview.net/forum?id=H1gL-2A9Ym.

Johannes Gasteiger, Stefan Weiß enberger, and Stephan Günnemann. Diffusion Improves Graph Learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019b.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for Quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *ICML'17*, pp. 1263–1272. JMLR.org, 2017.

Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.

Alessio Gravina and Davide Bacciu. Deep learning for dynamic graphs: Models and benchmarks. *IEEE Transactions on Neural Networks and Learning Systems*, 35(9):11788–11801, 2024. doi: 10.1109/TNNLS.2024.3379735.

Alessio Gravina, Davide Bacciu, and Claudio Gallicchio. Anti-Symmetric DGN: a stable architecture for Deep Graph Networks. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=J3Y7cgZOOS.

Alessio Gravina, Moshe Eliasof, Claudio Gallicchio, Davide Bacciu, and Carola-Bibiane Schönlieb. On oversquashing in graph neural networks through the lens of dynamical systems. In *The 39th Annual AAAI Conference on Artificial Intelligence*, 2025.

Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):922–929, Jul. 2019. doi: 10.1609/aaai.v33i01.3301922.

Benjamin Gutteridge, Xiaowen Dong, Michael M Bronstein, and Francesco Di Giovanni. Drew: Dynamically rewired message passing with delay. In *International Conference on Machine Learning*, pp. 12252–12267. PMLR, 2023.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 1025–1035. Curran Associates Inc., 2017. ISBN 9781510860964.

Ali Hariri, Alvaro Arroyo, Alessio Gravina, Moshe Eliasof, Carola-Bibiane Schönlieb, Davide Bacciu, Xiaowen Dong, Kamyar Azizzadenesheli, and Pierre Vandergheynst. Return of chebnet: Understanding and improving an overlooked GNN on long range tasks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=oLyfML1Qze.

Simon Haykin. *Neural networks and learning machines, 3/E*. Pearson Education India, 2009.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Xiaoxin He, Bryan Hooi, Thomas Laurent, Adam Perold, Yann LeCun, and Xavier Bresson. A generalization of vit/mlp-mixer to graphs. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23, 2023.

Simon Heilig, Alessio Gravina, Alessandro Trenta, Claudio Gallicchio, and Davide Bacciu. Port-hamiltonian architectural bias for long-range propagation in deep graph networks. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=03EkqSCKuO.

Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for Pre-training Graph Neural Networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJlWWJSFDH.

Yinan Huang, Siqi Miao, and Pan Li. What can we learn from state space models for machine learning on graphs? *arXiv preprint arXiv:2406.05815*, 2024.

Kedar Karhadkar, Pradeep Kr. Banerjee, and Guido Montufar. FoSR: First-order spectral rewiring for addressing oversquashing in GNNs. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=3YjQfCLdrzz.

T. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *Proceedings of the International Conference on Learning Representations*, 2016.

Kezhi Kong, Jiuhai Chen, John Kirchenbauer, Renkun Ni, C. Bayan Bruss, and Tom Goldstein. GOAT: A global transformer on large-scale graphs. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 17375–17390. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/kong23a.html.

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.

Jia Li, Zhichao Han, Hong Cheng, Jiao Su, Pengyun Wang, Jianfeng Zhang, and Lujia Pan. Predicting Path Failure In Time-Evolving Graphs. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1279–1289, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330847.

Jintang Li, Ruofan Wu, Xinzhou Jin, Boqun Ma, Liang Chen, and Zibin Zheng. State space models on temporal graphs: A first-principles study. *Advances in Neural Information Processing Systems*, 37:127030–127058, 2024.

Xiang Li, Renyu Zhu, Yao Cheng, Caihua Shan, Siqiang Luo, Dongsheng Li, and Weining Qian. Finding global homophily in graph neural networks when meeting heterophily. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 13242–13256. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/li22ad.html.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SJiHXGWAZ.

Hangchen Liu, Zheng Dong, Renhe Jiang, Jiewen Deng, Jinliang Deng, Quanjun Chen, and Xuan Song. Spatio-temporal adaptive embedding makes vanilla transformer sota for traffic forecasting. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pp. 4125–4129, 2023.

Zheng Lu, Chen Zhou, Jing Wu, Hao Jiang, and Songyue Cui and. Integrating granger causality and vector auto-regression for traffic prediction of large-scale wlans. *KSII Transactions on Internet and Information Systems*, 10(1):136–151, January 2016. doi: 10.3837/tiis.2016.01.008.

Sitao Luan, Chenqing Hua, Qincheng Lu, Liheng Ma, Lirong Wu, Xinyu Wang, Minkai Xu, Xiao-Wen Chang, Doina Precup, Rex Ying, Stan Z. Li, Jian Tang, Guy Wolf, and Stefanie Jegelka. The heterophilic graph learning handbook: Benchmarks, models, theoretical analysis, applications and challenges. *arXiv preprint arXiv:2407.09618*, 2024.

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 *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 23321–23337. PMLR, 23–29 Jul 2023.

Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. Simplifying approach to node classification in graph neural networks. *Journal of Computational Science*, 62:101695, 2022. ISSN 1877-7503. doi: https://doi.org/10.1016/j.jocs.2022.101695. URL https://www.sciencedirect.com/science/article/pii/S1877750322000990.

Carl D Meyer. *Matrix analysis and applied linear algebra*. SIAM, 2023.

Alessio Micheli and Domenico Tortorella. Discrete-time dynamic graph echo state networks. *Neurocomputing*, 496:85–95, 2022. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2022.05.001.

Yaaqov Mishayev, Yonatan Sverdlov, Tal Amir, and Nadav Dym. Short-range oversquashing. *arXiv preprint arXiv:2511.20406*, 2025.

Luis Müller, Mikhail Galkin, Christopher Morris, and Ladislav Rampášek. Attending to graph transformers. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=HhbqHBBrfZ.

George Panagopoulos, Giannis Nikolentzos, and Michalis Vazirgiannis. Transfer Graph Neural Networks for Pandemic Forecasting. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, 2021.

Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, 2020.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In Sanjoy Dasgupta and David McAllester (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1310–1318, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL https://proceedings.mlr.press/v28/pascanu13.html.

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

Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a General, Powerful, Scalable Graph Transformer. *Advances in Neural Information Processing Systems*, 35, 2022.

Benedek Rozemberczki, Paul Scherer, Yixuan He, George Panagopoulos, Alexander Riedel, Maria Astefanoaei, Oliver Kiss, Ferenc Beres, Guzman Lopez, Nicolas Collignon, and Rik Sarkar. PyTorch Geometric Temporal: Spatiotemporal Signal Processing with Neural Machine Learning Models. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, pp. 4564–4573, 2021.

Luana Ruiz, Fernando Gama, and Alejandro Ribeiro. Gated graph recurrent neural networks. *IEEE Transactions on Signal Processing*, 68:6303–6318, 2020.

T Konstantin Rusch, Ben Chamberlain, James Rowbottom, Siddhartha Mishra, and Michael Bronstein. Graph-coupled oscillator networks. In *International Conference on Machine Learning*, pp. 18888–18909. PMLR, 2022.

Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured Sequence Modeling with Graph Convolutional Recurrent Networks. In *Neural Information Processing*, pp. 362–373, Cham, 2018. Springer International Publishing. ISBN 978-3-030-04167-0.

Chao Shang, Jie Chen, and Jinbo Bi. Discrete graph structure learning for forecasting multiple time series. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=WEHSlH5mOk.

Zezhi Shao, Zhao Zhang, Fei Wang, and Yongjun Xu. Pre-training enhanced spatial-temporal graph neural network for multivariate time series forecasting. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pp. 1567–1577. ACM, 2022.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 1548–1554. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/214. URL https://doi.org/10.24963/ijcai.2021/214. Main Track.

Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J Sutherland, and Ali Kemal Sinop. Exphormer: Sparse transformers for graphs. In *International Conference on Machine Learning*, pp. 31613–31632. PMLR, 2023.

Hamed Shirzad, Honghao Lin, Balaji Venkatachalam, Ameya Velingker, David P Woodruff, and Danica J Sutherland. Even sparser graph transformers. *Advances in Neural Information Processing Systems*, 37:71277–71305, 2024.

Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, Aug 2004. ISSN 1573-1375. doi: 10.1023/B:STCO.0000035301.49549.88.

Chao Song, Youfang Lin, Shengnan Guo, and Huaiyu Wan. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):914–921, Apr. 2020. doi: 10.1609/aaai.v34i01.5438.

David Sussillo and Larry F Abbott. Random walk initialization for training very deep feedforward networks. *arXiv preprint arXiv:1412.6558*, 2014.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, volume 2 of *NIPS'14*, pp. 3104–3112, Cambridge, MA, USA, 2014. MIT Press.

Aynaz Taheri and Tanya Berger-Wolf. Predictive temporal embedding of dynamic graphs. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '19, pp. 57–64. Association for Computing Machinery, 2020. ISBN 9781450368681. doi: 10.1145/3341161.3342872.

Jan Tönshoff, Martin Ritzert, Eran Rosenbluth, and Martin Grohe. Where did the gap go? reassessing the long-range graph benchmark. In *The Second Learning on Graphs Conference*, 2023. URL https://openreview.net/forum?id=rIUjwxc5lj.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=7UmjRGzp-A.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Chloe Wang, Oleksii Tsepa, Jun Ma, and Bo Wang. Graph-mamba: Towards long-range graph sequence modeling with selective state spaces. *arXiv preprint arXiv:2402.00789*, 2024a.

Kun Wang, Guibin Zhang, Xinnan Zhang, Junfeng Fang, Xun Wu, Guohao Li, Shirui Pan, Wei Huang, and Yuxuan Liang. The heterophilic snowflake hypothesis: Training and empowering gnns for heterophilic graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, pp. 3164–3175, New York, NY, USA, 2024b. Association for Computing Machinery. ISBN 9798400704901. doi: 10.1145/3637528.3671791.

Xiyuan Wang and Muhan Zhang. How powerful are spectral graph neural networks. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 23341–23362. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/wang22am.html.

Yifei Wang, Yisen Wang, Jiansheng Yang, and Zhouchen Lin. Dissecting the Diffusion Process in Linear Graph Convolutional Networks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 5758–5769. Curran Associates, Inc., 2021.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6861–6871. PMLR, 09–15 Jun 2019a.

Yi Wu, Yanyang Xu, Wenhao Zhu, Guojie Song, Zhouchen Lin, Liang Wang, and Shaoguo Liu. Kdlgt: A linear graph transformer framework via kernel decomposition approach. In *IJCAI*, pp. 2370–2378, 2023.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, pp. 1907–1913. AAAI Press, 2019b. ISBN 9780999241141.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32:4–24, 2020a.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, pp. 753–763, New York, NY, USA, 2020b. Association for Computing Machinery. ISBN 9781450379984. doi: 10.1145/3394486.3403118.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

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.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 17283–17297. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf.

Xudong Zhang, Xuewen Chen, Haina Tang, Yulei Wu, Hanji Shen, and Jun Li. Adpstgcn: Adaptive spatial–temporal graph convolutional network for traffic forecasting. *Knowledge-Based Systems*, 301:112295, 2024. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2024.112295.

Kai Zhao, Qiyu Kang, Yang Song, Rui She, Sijie Wang, and Wee Peng Tay. Graph neural convection-diffusion with heterophily. In Edith Elkind (ed.), *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pp. 4656–4664. International Joint Conferences on Artificial Intelligence Organization, 8 2023. doi: 10.24963/ijcai.2023/518. URL https://doi.org/10.24963/ijcai.2023/518. Main Track.

Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(9):3848–3858, 2020. doi: 10.1109/TITS.2019.2935152.

Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. Gman: A graph multi-attention network for traffic prediction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34 (01):1234–1241, Apr. 2020. doi: 10.1609/aaai.v34i01.5477.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 7793–7804. Curran Associates, Inc., 2020.

Jiong Zhu, Ryan A. Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K. Ahmed, and Danai Koutra. Graph neural networks with heterophily. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):11168–11176, May 2021. doi: 10.1609/aaai.v35i12.17332. URL https://ojs.aaai.org/index.php/AAAI/article/view/17332.

A Fast Parallel Implementation

We detail the derivation and implementation of an efficient parallel algorithm for computing an BUMP block, focusing on the GF and LF stream variants, i.e., where the GSO of both streams is fixed within each block. However, the approach we present here can be extended to the time-varying setting. In fact, in the LTV setting, the unfolded operator involves products of the form $\mathbf{A}_{[0:i]} = \prod_{l=1}^{i} \mathbf{A}_l$ rather than simple powers $\mathbf{A}^i$ (see Equation (17)). Importantly, the operators $\mathbf{A}_1, \ldots, \mathbf{A}_k$ can be trained independently, due to the fact that the operators are learned by self-attention of the input features (rather than node embeddings), enabling the use of parallel scan operations to evaluate the summation in Equation (17) efficiently.

The unfolded recurrence of an BUMP block gives the following closed-form solution:

$$\mathbf{X}_{k+1} = \mathbf{A}^k \mathbf{U}_1 \mathbf{B} \mathbf{W}^k + \mathbf{A}^{k-1} \mathbf{U}_2 \mathbf{B} \mathbf{W}^{k-1} + \ldots + \mathbf{A} \mathbf{U}_k \mathbf{B} \mathbf{W} + \mathbf{U}_{k+1} \mathbf{B}. \tag{17}$$

Therefore the equation of an BUMP block reads:

$$\mathbf{X}_{k+1} = \sum_{i=0}^{k} \mathbf{A}^i \mathbf{U}_{k+1-i} \mathbf{B} \mathbf{W}^i, \tag{18}$$

$$\mathbf{Y}_{k+1} = \text{MLP}(\mathbf{X}_{k+1}), \tag{19}$$

The closed-form solution of an BUMP block tells us that we could implement the whole recurrence in one shot. However, the computation of the powers of both the GSO, $\mathbf{A}$, and the recurrent weights, $\mathbf{W}$, can be extremely expensive for generic matrices and large values of k. On the other hand, the powers of diagonal matrices are fairly easy to compute, since they are simply the powers of their diagonal entries. Below, we show how to reduce a generic dense real-valued BUMP block to an equivalent diagonalised complex-valued BUMP block.

Assume the following diagonalisation of the shift operator: $\mathbf{A} = \mathbf{P} \mathbf{\Lambda} \mathbf{P}^{-1}$. If undirected graph, i.e., $\mathbf{A}$ is symmetric, then by spectral theorem the $\mathbf{P}$ is a real orthogonal matrix (i.e. $\mathbf{P}^{-1} = \mathbf{P}^\top$) and $\mathbf{\Lambda}$ is real.

Assume the following diagonalisation of the weights: $\mathbf{W} = \mathbf{V} \mathbf{\Sigma} \mathbf{V}^{-1}$. If using dense real matrices as weights, then their diagonalisation is possible only assuming complex matrices of eigenvectors $\mathbf{V}$ and complex eigenvalues $\mathbf{\Sigma}$. Also, note that the set of defective matrices (i.e., non-diagonalizable in $\mathbb{C}$) has zero Lebesgue measure (Golub & Van Loan, 2013).

Assume the following MLP equations with 2 layers: $\text{MLP}(\mathbf{X}) = \phi(\mathbf{X}\mathbf{W}_1)\mathbf{W}_2$, where ϕ is a nonlinearity, and $\mathbf{W}_1, \mathbf{W}_2$ real dense matrices.

With the above assumptions, the BUMP block equations can be equivalently written as:

$$\mathbf{X}_{k+1} = \sum_{i=0}^{k} \mathbf{P} \mathbf{\Lambda}^i \mathbf{P}^{-1} \mathbf{U}_{k+1-i} \mathbf{B} \mathbf{V} \mathbf{\Sigma}^i \mathbf{V}^{-1}, \tag{20}$$

$$\mathbf{Y}_{k+1} = \phi(\mathbf{X}_{k+1} \mathbf{W}_1) \mathbf{W}_2, \tag{21}$$

which we can write as:

$$\mathbf{X}_{k+1} = \mathbf{P} \left(\sum_{i=0}^{k} \mathbf{\Lambda}^i \mathbf{P}^{-1} \mathbf{U}_{k+1-i} \mathbf{B} \mathbf{V} \mathbf{\Sigma}^i \right) \mathbf{V}^{-1}, \tag{22}$$

$$\mathbf{Y}_{k+1} = \phi(\mathbf{X}_{k+1} \mathbf{W}_1) \mathbf{W}_2, \tag{23}$$

Multiply on the left side both terms by $\mathbf{P}^{-1}$ and on the right side both terms by $\mathbf{V}$

$$\mathbf{P}^{-1} \mathbf{X}_{k+1} \mathbf{V} = \sum_{i=0}^{k} \mathbf{\Lambda}^i \mathbf{P}^{-1} \mathbf{U}_{k+1-i} \mathbf{B} \mathbf{V} \mathbf{\Sigma}^i \tag{24}$$

If we change coordinate reference to $\mathbf{Z}_{k+1} = \mathbf{P}^{-1}\mathbf{X}_{k+1}\mathbf{V}$, then we can write:

$$\mathbf{Z}_{k+1} = \sum_{i=0}^{k} \mathbf{\Lambda}^i \mathbf{P}^{-1} \mathbf{U}_{k+1-i} \mathbf{B} \mathbf{V} \mathbf{\Sigma}^i, \tag{25}$$

$$\mathbf{Y}_{k+1} = \phi(\mathbf{P}\mathbf{Z}_{k+1}\mathbf{V}^{-1}\mathbf{W}_1)\mathbf{W}_2, \tag{26}$$

Equations (25) and (26) give the same exact dynamics of the equations (18) and (19).

The matrix of complex eigenvectors $\mathbf{V}$ in (25) can be merged into the real matrix of weights $\mathbf{B}$ in equation (27). Therefore, we can call $\hat{\mathbf{B}}$ a complex matrix of weights that accounts for the term $\mathbf{BV}$. Similarly, the matrix eigenvectors $\mathbf{V}^{-1}$ in (26) can be merged into the matrix of weights $\mathbf{W}_1$ in equation (28), that we call $\hat{\mathbf{W}}_1$.

With these new notations, we can write the equivalent diagonalised complex-valued BUMP block:

$$\mathbf{Z}_{k+1} = \sum_{i=0}^{k} \mathbf{\Lambda}^i \hat{\mathbf{U}}_{k+1-i} \hat{\mathbf{B}} \mathbf{\Sigma}^i, \tag{27}$$

$$\mathbf{Y}_{k+1} = \phi(\mathbf{P}\mathbf{Z}_{k+1}\hat{\mathbf{W}}_1)\mathbf{W}_2, \tag{28}$$

where, in summary:

- input is pre-processed as $\hat{\mathbf{U}}_{k+1-i} = \mathbf{P}^{-1}\mathbf{U}_{k+1-i}$,
- $\mathbf{\Lambda}$ is the diagonal matrix of the eigenvalues of the GSO,
- learnable recurrent weights are $\hat{\mathbf{B}}$ (complex and dense), and $\mathbf{\Sigma}$ (complex and diagonal)
- learnable readout weights are $\hat{\mathbf{W}}_1$ (complex and dense), and $\mathbf{W}_2$ (real and dense)

Equations (27)-(28) tell us that we can implement the whole recurrence efficiently in a closed-form solution that only involves powers of diagonal matrices.

Remarkably, the entire BUMP block computation occurs in 10^{-3} seconds, see Figure 2. The figure

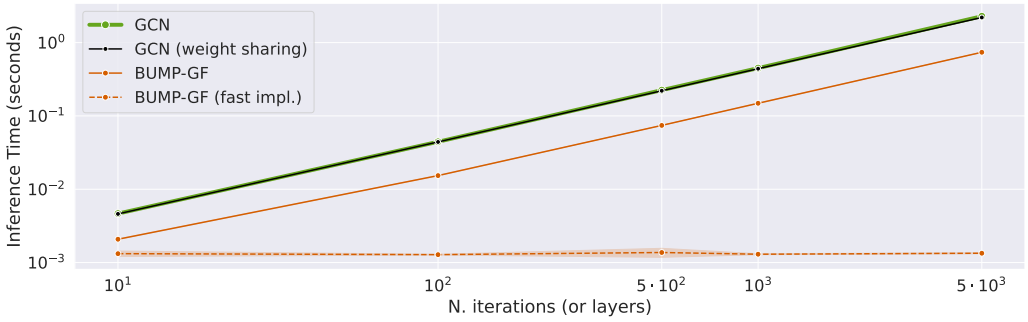

Figure 2: Inference time on a graph of $n = 100$ nodes (with number of edges 3058), input dimension $C = 1$, hidden_dim $= 32$, and increasing lengths $k = 10, 100, 500, 1000, 5000$. GCN is a standard GCN with $\tanh$ without residual with k layers. GCN (weight sharing) is the same, but just one layer is iterated k times. For simplicity, we restricted our BUMP to the single propagation stream of GF (BUMP-GF), i.e., when $\alpha_t^{(1)} = 1$ and $\alpha_t^{(2)} = 0$. BUMP-GF uses 1 block.

also shows that one block BUMP scales similarly to GCN and GCN (weight sharing), whose lines are overlapping, but it is slightly faster, owing to the lack of nonlinearity in the recurrence. This benefit grows with more iterations. On the other hand, the fast implementation of BUMP maintains constant runtime, provided enough GPU memory.

We observe that, a practical solution for memory-constrained GPUs is to chunk the computation for large number of recurrence steps. Assuming that S is the maximum number of time steps a GPU can accommodate, the parallel implementation can be divided into $NC = \text{num_steps}/S$ chunks.

These NC chunks can then be processed sequentially and their results combined. This approach roughly increases computational time by a factor of NC due to the single-GPU sequentialization. Alternatively, if NC GPUs are available, each chunk can be processed in parallel on a different GPU, since each intermediate computation depends on the current input (i.e., $\mathbf{U}_t$) and powers of the GSO and the learnable weights (i.e., $\mathbf{A}^{k-t-1}$ and $\mathbf{W}^{k-t-1}$) as shown in Equation (17). Then the results of each GPU can be merged to obtain the next prediction $\mathbf{X}_{t+1}$, greatly mitigating the slowdown.

B Construction of the Learnable Doubly Stochastic GSO

Our analysis motivates non-negative GSOs that support stable and non-vanishing propagation under repeated application. In BUMP, we instantiate the learned stream using symmetric doubly stochastic GSOs, which provide a convenient sufficient class satisfying these properties while enabling a direct spectral characterization. We construct such operators by symmetrically balancing a learnable symmetric non-negative kernel. We consider two alternative parameterizations of this kernel, differing primarily in their computational complexity.

Dense quadratic attention. Let

$$\mathbf{Q}_t = \mathbf{U}_t\mathbf{W}_Q, \qquad \mathbf{K}_t = \mathbf{U}_t\mathbf{W}_K,$$

with attention dimension d_a. We construct the symmetric score matrix

$$\mathbf{S}_t = \tfrac{1}{2}\left(\mathbf{Q}_t\mathbf{K}_t^\top + \mathbf{K}_t\mathbf{Q}_t^\top\right), \qquad \mathbf{M}_t = \exp(\mathbf{S}_t), \tag{29}$$

where the exponential is applied element-wise. By construction, $\mathbf{M}_t$ is symmetric and strictly positive. Explicitly forming the pairwise scores requires $\mathcal{O}(n^2 d_a)$ time and $\mathcal{O}(n^2)$ memory. Blocked evaluation can reduce peak working memory, but does not remove the quadratic computational dependence on the number of nodes.

Implicit linear attention. To avoid explicitly constructing an $n \times n$ kernel, we alternatively use a low-rank positive feature map. Let

$$\Phi_t = \phi(\mathbf{Q}_t), \qquad \bar{\Phi}_t = \phi(\mathbf{K}_t),$$

with $\Phi_t, \bar{\Phi}_t \in \mathbb{R}^{n\times r}$ and strictly positive feature map ϕ; in our implementation, we use softplus. We implicitly define

$$\widetilde{\mathbf{M}}_t = \tfrac{1}{2}\left(\Phi_t\bar{\Phi}_t^\top + \bar{\Phi}_t\Phi_t^\top\right) + \epsilon\mathbf{I}, \qquad \epsilon > 0. \tag{30}$$

Since ϕ is strictly positive, the resulting kernel is symmetric and strictly positive. Importantly, it does not need to be materialized. For any matrix $\mathbf{Z}$,

$$\widetilde{\mathbf{M}}_t\mathbf{Z} = \tfrac{1}{2}\left[\Phi_t(\bar{\Phi}_t^\top\mathbf{Z}) + \bar{\Phi}_t(\Phi_t^\top\mathbf{Z})\right] + \epsilon\mathbf{Z}. \tag{31}$$

Hence, kernel–matrix products can be evaluated using only the low-rank factors, yielding linear complexity in n for fixed feature and channel dimensions.

Symmetric Sinkhorn normalization. Both parameterizations produce a symmetric strictly positive kernel, which we denote generically by $\mathbf{M}_t$. To obtain the learned GSO used by BUMP, we symmetrically balance this kernel. We seek a positive vector $\mathbf{d}_t$ satisfying

$$\mathbf{d}_t \odot (\mathbf{M}_t\mathbf{d}_t) = \mathbf{1}, \tag{32}$$

and define

$$\mathbf{D}_t = \mathrm{diag}(\mathbf{d}_t), \qquad \hat{\mathbf{A}}_t = \mathbf{D}_t\mathbf{M}_t\mathbf{D}_t. \tag{33}$$

We compute the scaling using the symmetric Sinkhorn iteration

$$\mathbf{d}_t^{(\ell+1)} = \sqrt{\mathbf{d}_t^{(\ell)} \oslash (\mathbf{M}_t\mathbf{d}_t^{(\ell)})}, \qquad \mathbf{d}_t^{(0)} = \mathbf{1}, \tag{34}$$

where all operations are component-wise. We perform five symmetric Sinkhorn iterations, which we empirically find sufficient to obtain balanced operators across the considered datasets, while preserving the linear complexity of the implicit construction.

For a symmetric strictly positive kernel, symmetric matrix balancing admits a positive scaling (Sinkhorn & Knopp, 1967). The resulting $\hat{\mathbf{A}}_t$ is symmetric, strictly positive, and row stochastic; by symmetry, it is also column stochastic and therefore doubly stochastic. In particular, $\hat{\mathbf{A}}_t\mathbf{1} = \mathbf{1}$, its spectral radius is one, and repeated application remains bounded and cannot converge to the null operator. Thus, the construction satisfies the stability and non-vanishing properties motivating our choice of learned GSO.

Strict positivity is stronger than required for these basic propagation properties, but ensures the Perron–Frobenius conditions used in the deep-regime analysis of the fixed learned GSO in Theorem 3.5. Both kernel parameterizations therefore satisfy the assumptions underlying the theoretical results, while providing different computational trade-offs.

For the implicit construction, both balancing and propagation can be performed without materializing either the kernel or the learned GSO. In particular,

$$\hat{\mathbf{A}}_t\mathbf{Z} = \mathbf{D}_t\left[\widetilde{\mathbf{M}}_t(\mathbf{D}_t\mathbf{Z})\right], \tag{35}$$

where the action of $\widetilde{\mathbf{M}}_t$ is evaluated using Equation (31). Likewise, the products $\widetilde{\mathbf{M}}_t\mathbf{d}_t^{(\ell)}$ required by Equation (34) are evaluated through the same implicit kernel action. Consequently, the implicit construction avoids materializing any $n \times n$ matrix throughout both normalization and propagation.

C Proofs of the Theoretical Statements

Here, we provide all the proofs of lemmas, theorems, and corollaries stated in the main text.

C.1 Proof of Theorem 3.3

Theorem. The Jacobian of the unrolled linear message-passing dynamics of a BUMP block, from node j at layer s to node i at layer $t \geq s$, can be computed exactly, and it has the following form:

$$\frac{\partial \mathbf{X}_t^{(i)}}{\partial \mathbf{X}_s^{(j)}} = \underbrace{(\mathbf{A}^{t-s})_{ij}}_{\text{scalar}}\underbrace{(\mathbf{W}^\top)^{t-s}}_{\text{matrix}}, \qquad \text{or} \qquad \frac{\partial \mathbf{X}_t^{(i)}}{\partial \mathbf{X}_s^{(j)}} = \underbrace{\left(\prod_{l=s+1}^{t}\mathbf{A}_l\right)_{ij}}_{\text{scalar}}\underbrace{(\mathbf{W}^\top)^{t-s}}_{\text{matrix}}$$

Proof. There are 4 possible GSOs, as summarised in Table 2. Given Fixed (GF): $\mathbf{D}^{-\frac{1}{2}}(\tilde{\mathbf{A}} + \mathbf{I})\mathbf{D}^{-\frac{1}{2}}$, Given Time-Varying (GTV): $\mathbf{D}_t^{-\frac{1}{2}}(\tilde{\mathbf{A}}_t + \mathbf{I})\mathbf{D}_t^{-\frac{1}{2}}$; where $\tilde{\mathbf{A}}$ denotes the adjacency matrix, Learned Fixed (LF): $\mathbf{D}f_\theta(\mathbf{U})\mathbf{D}$, Learned Time-Varying (LTV): $\mathbf{D}_t f_\theta(\mathbf{U}_t)\mathbf{D}_t$; where $f_\theta(\cdot)$ is either learned by dense quadratic attention or by implicit linear attention, as detailed in Appendix B.
In this proof we use the notation $(\mathbf{M})_{ij}$ to denote the (i, j) entry of a matrix $\mathbf{M}$, and $\mathbf{M}^{(i)}$ to denote the i-th row of a matrix $\mathbf{M}$. Let us start with the recurrent equation $\mathbf{X}_{t+1} = \mathbf{A}\mathbf{X}_t\mathbf{W} + \mathbf{U}_{t+1}\mathbf{B}$. Therefore, the i-th node features are updated as follows: $\mathbf{X}_{t+1}^{(i)} = \sum_{l=1}^{n}(\mathbf{A})_{il}\mathbf{X}_t^{(l)}\mathbf{W} + \mathbf{U}_{t+1}^{(i)}\mathbf{B}$. Now, the only term involving $\mathbf{X}_t^{(j)}$ is $(\mathbf{A})_{ij}\mathbf{X}_t^{(j)}\mathbf{W}$. Therefore, the Jacobian reads $\dfrac{\partial \mathbf{X}_{t+1}^{(i)}}{\partial \mathbf{X}_t^{(j)}} = \dfrac{\partial}{\partial \mathbf{X}_t^{(j)}}\Big((\mathbf{A})_{ij}\mathbf{X}_t^{(j)}\mathbf{W}\Big)$. Note that even in the learned cases of LF and LTV, the GSO is learned via attention on $\mathbf{U}$ for the static or on $\mathbf{U}_t$ for the temporal case, hence $(\mathbf{A})_{ji}$ does not depend on $\mathbf{X}_t^{(j)}$. Thus $\dfrac{\partial \mathbf{X}_{t+1}^{(i)}}{\partial \mathbf{X}_t^{(j)}} = (\mathbf{A})_{ij}\dfrac{\partial}{\partial \mathbf{X}_t^{(j)}}\Big(\mathbf{X}_t^{(j)}\mathbf{W}\Big)$. Now, given a row vector $\mathbf{x} \in \mathbb{R}^c$ and a square matrix $\mathbf{M}$, then the function $\mathbf{f}(\mathbf{x}) = \mathbf{x}\mathbf{M}$, whose i-th component is $f_i = \sum_{l=1}^{c} x_l(\mathbf{M})_{li}$, has derivatives $\frac{\partial f_i}{\partial \mathbf{x}_j} = \frac{\partial}{\partial x_j}(x_j(\mathbf{M})_{ji}) = (\mathbf{M})_{ji}$. Hence, the Jacobian is $\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \mathbf{M}^\top$. Therefore, it holds $\dfrac{\partial \mathbf{X}_{t+1}^{(i)}}{\partial \mathbf{X}_t^{(j)}} = (\mathbf{A})_{ji}\mathbf{W}^\top$.
Proof GF and LF.

In both cases of GF and LF, we have a fixed GSO per block that we denote generically as $\mathbf{A}$ here. For the case of non-consecutive time steps, we can unfold the recurrent equation $\mathbf{X}_{t+1} = \mathbf{A}\mathbf{X}_t\mathbf{W} + \mathbf{U}_{t+1}\mathbf{B}$ between any two time steps $s \leq t$, as follows:

$$\mathbf{X}_t = \mathbf{A}^{t-s}\mathbf{X}_s\mathbf{W}^{t-s} + \sum_{l=0}^{t-s-1} \mathbf{A}^l\mathbf{U}_{t-l}\mathbf{B}\mathbf{W}^i. \tag{36}$$

From the unfolded recurrent equation 36 of a BUMP we can see that the only term involving $\mathbf{X}_s$ is $\mathbf{A}^{t-s}\mathbf{X}_s\mathbf{W}^{t-s}$. Thus, the Jacobian reads $\frac{\partial \mathbf{X}_t^{(i)}}{\partial \mathbf{X}_s^{(j)}} = \frac{\partial}{\partial \mathbf{X}_s^{(j)}}\Big((\mathbf{A}^{t-s}\mathbf{X}_s\mathbf{W}^{t-s})^{(i)}\Big) = \frac{\partial}{\partial \mathbf{X}_s^{(j)}}\Big((\mathbf{A}^{t-s})_{ij}\mathbf{X}_s^{(j)}\mathbf{W}^{t-s}\Big) = (\mathbf{A}^{t-s})_{ij}(\mathbf{W}^\top)^{t-s}$.

Proof GTV and LTV.
In both cases of GTV and LTV, we have a time-varying GSO that changes at each time step within a block that we denote generically as $\mathbf{A}_\tau$ here. For the case of non-consecutive time steps, we can unfold the recurrent equation $\mathbf{X}_{t+1} = \mathbf{A}_{t+1}\mathbf{X}_t\mathbf{W} + \mathbf{U}_{t+1}\mathbf{B}$ between any two time steps $s \leq t$, as follows:

$$\mathbf{X}_t = \mathbf{A}_{[s:t]}\mathbf{X}_s\mathbf{W}^{t-s} + \sum_{l=0}^{t-s-1} \mathbf{A}_{[s:l]}\mathbf{U}_{t-l}\mathbf{B}\mathbf{W}^i, \tag{37}$$

where, to ease the notation, we denoted $\mathbf{A}_{[s:t]} = \prod_{\tau=s+1}^{t} \mathbf{A}_\tau$. From the unfolded recurrent equation (37) of a BUMP we can see that the only term involving $\mathbf{X}_s$ is $\mathbf{A}_{[s:t]}\mathbf{X}_s\mathbf{W}^{t-s}$. Thus, the Jacobian reads $\frac{\partial \mathbf{X}_t^{(i)}}{\partial \mathbf{X}_s^{(j)}} = \frac{\partial}{\partial \mathbf{X}_s^{(j)}}\Big((\mathbf{A}_{[s:t]}\mathbf{X}_s\mathbf{W}^{t-s})^{(i)}\Big) = \frac{\partial}{\partial \mathbf{X}_s^{(j)}}\Big((\mathbf{A}_{[s:t]})_{ij}\mathbf{X}_s^{(j)}\mathbf{W}^{t-s}\Big) = (\mathbf{A}_{[s:t]})_{ij}(\mathbf{W}^\top)^{t-s}$.

□

C.2 Proof of Theorem 3.4

Theorem. For GF and LF, the spectral radius $\rho(\mathbf{A}^t) = 1$ for every $t \geq 1$; for LTV, $\rho(\mathbf{A}_{[0:t]}) = 1$. Consequently, these powers and products neither diverge nor converge to the null matrix. In contrast, for GTV, $\rho(\mathbf{A}_{[0:t]}) \leq 1$ and the product remains bounded, but it may converge to the null matrix as $t \to \infty$.

Proof. We divide the proof into 4 parts: GF, LF, GTV, LTV.
Proof for GF.
Observe that $\mathbf{A}^t = \big(\mathbf{D}^{-\frac{1}{2}}(\tilde{\mathbf{A}} + \mathbf{I})\mathbf{D}^{-\frac{1}{2}}\big)\big(\mathbf{D}^{-\frac{1}{2}}(\tilde{\mathbf{A}} + \mathbf{I})\mathbf{D}^{-\frac{1}{2}}\big)\ldots\big(\mathbf{D}^{-\frac{1}{2}}(\tilde{\mathbf{A}} + \mathbf{I})\mathbf{D}^{-\frac{1}{2}}\big) = \mathbf{D}^{-\frac{1}{2}}(\tilde{\mathbf{A}} + \mathbf{I})\Big(\mathbf{D}^{-1}(\tilde{\mathbf{A}} + \mathbf{I})\Big)^{t-1}\mathbf{D}^{-\frac{1}{2}}$. Now, $\mathbf{D}^{-1}(\tilde{\mathbf{A}} + \mathbf{I})$ is a stochastic matrix, and so also its powers are stochastic matrices. Therefore, $\mathbf{D}^{-\frac{1}{2}}\mathbf{A}^t\mathbf{D}^{\frac{1}{2}} = \Big(\mathbf{D}^{-1}(\tilde{\mathbf{A}} + \mathbf{I})\Big)^t$ is a stochastic matrix, for every $t \geq 1$. Thus 1 is an eigenvalue and all eigenvalues have modulus at most one (Meyer, 2023; Banerjee & Mehatari, 2016). Since $\mathbf{A}$ is symmetric, $\rho(\mathbf{A}^t) = \|\mathbf{A}^t\|_2 = 1$, so its powers are bounded and cannot converge to zero.

Proof for LF.
For LF, $\mathbf{A}$ is by construction a symmetric non-negative and doubly stochastic matrix. Hence $\mathbf{A}\mathbf{1} = \mathbf{1}$, which implies $\rho(\mathbf{A}) = \|\mathbf{A}\|_2 = 1$. The same statements hold for every power $\mathbf{A}^t$, which remains doubly stochastic, so its powers are bounded annd cannot converge to zero.

Proof for LTV.
For LTV, a product of doubly stochastic matrices is doubly stochastic, so the identical argument of the case LF applies to the case of LTV. In particular, $\mathbf{A}_{[0:t]}\mathbf{1} = \mathbf{1}$. Hence, the spectral radius of $\mathbf{A}_{[0:t]}$ is exactly equal to 1, and as a byproduct we have that $\mathbf{A}_{[0:t]}$ cannot explode nor converge to the null matrix for t approaching to infinity.

Proof for GTV.
Here we have $\mathbf{A}_t = \mathbf{D}_t^{-\frac{1}{2}}(\tilde{\mathbf{A}}_t + \mathbf{I})\mathbf{D}_t^{-\frac{1}{2}}$, where the time-varying topology is given by the time-varying adjacency matrix $\tilde{\mathbf{A}}_t$. Each $\mathbf{A}_t$ is symmetric, thus the spectral radius coincides with the

spectral norm, which is exactly 1. Hence $\rho(\mathbf{A}_{[0:t]}) \leq \prod_i ||\mathbf{A}_i|| \leq 1$. This proves that the product $\mathbf{A}_{[0:t]}$ remains bounded. Nevertheless, even though each matrix $\mathbf{A}_i$ has spectral radius exactly equal to 1 for each time step, the product $\prod_i \mathbf{A}_i$ can still converge to the null matrix. Therefore, the linear message-passing aggregation could potentially lead to vanishing information dynamics for a large number of iterations. As a counterexample, consider the following two adjacency matrices:

$$\tilde{\mathbf{A}}_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \quad \tilde{\mathbf{A}}_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}.$$

Symmetrically normalizing with self-loops gives

$$\mathbf{A}_1 = \mathbf{D}_1^{-\frac{1}{2}}(\tilde{\mathbf{A}}_1 + \mathbf{I})\mathbf{D}_1^{-\frac{1}{2}}, \quad \mathbf{A}_2 = \mathbf{D}_2^{-\frac{1}{2}}(\tilde{\mathbf{A}}_2 + \mathbf{I})\mathbf{D}_2^{-\frac{1}{2}},$$

and the product $\mathbf{A}_1\mathbf{A}_2$ has spectral radius strictly less than 1. Therefore the sequence $\mathbf{A}_1\mathbf{A}_2\mathbf{A}_1\mathbf{A}_2\ldots$ converges to the null matrix.

□

C.3 Proof of Theorem 3.5

Theorem. Assume an undirected and connected graph with n nodes and m edges. Then, for large values of $t - s$, the Jacobian of the linear recurrent equation of a BUMP block, from node j at layer s to node i at layer $t \geq s$, admits the following approximation:

$$\frac{\partial \mathbf{X}_t^{(i)}}{\partial \mathbf{X}_s^{(j)}} \approx \frac{\sqrt{(1+d_i)(1+d_j)}}{n+2m} \, (\mathbf{W}^\top)^{t-s}, \qquad \text{or} \qquad \frac{\partial \mathbf{X}_t^{(i)}}{\partial \mathbf{X}_s^{(j)}} \approx \frac{1}{n} \, (\mathbf{W}^\top)^{t-s},$$

where d_l is the degree of the l-th node.

Proof. We prove two cases: GF and LF.

GF case.

As discussed in the proof of Theorem 3.4, the matrix $\mathbf{D}^{-\frac{1}{2}}\mathbf{A}^t\mathbf{D}^{\frac{1}{2}}$ is a stochastic matrix, where $\mathbf{D}$ is the degree matrix of the adjacency plus the identity. Let $\lambda_1, \ldots, \lambda_n$ be all the eigenvalues (not necessarily distinct) of such a stochastic matrix, with corresponding eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$. Thus, $\mathbf{D}^{-\frac{1}{2}}\mathbf{A}^t\mathbf{D}^{\frac{1}{2}}\mathbf{v}_l = \lambda_l\mathbf{v}_l$, from which it follows, multiplying both sides by $\mathbf{D}^{\frac{1}{2}}$, that $\mathbf{A}^t\mathbf{D}^{\frac{1}{2}}\mathbf{v}_l = \lambda_l\mathbf{D}^{\frac{1}{2}}\mathbf{v}_l$. This means that the eigenvalues of $\mathbf{A}^t$ are exactly the same of those of the stochastic matrix $\mathbf{D}^{-\frac{1}{2}}\mathbf{A}^t\mathbf{D}^{\frac{1}{2}}$ with eigenvectors $\mathbf{D}^{\frac{1}{2}}\mathbf{v}_1, \ldots, \mathbf{D}^{\frac{1}{2}}\mathbf{v}_n$, for all t. Since the spectral radius of a stochastic matrix is 1, and the vector $\mathbf{1}$ with all components equal to 1 is necessarily an eigenvector due to the row-sum being 1 for a stochastic matrix, then it follows that the largest eigenvalue of $\mathbf{A}^t$ is 1 and $\mathbf{d} = \text{diag}(\mathbf{D}^{\frac{1}{2}})$ is an eigenvector corresponding to eigenvalue 1, for all t. The assumption of connectedness makes the operator irreducible, and the positive diagonal supplied by the self-loops makes it aperiodic; hence $\mathbf{A}$ is primitive. Perron-Frobenius implies that the largest eigenvalue has multiplicity 1 (Horn & Johnson, 2012).

We want to provide an estimation of the term $(\mathbf{A}^{t-s})_{ij}$, in the (GF) case of Theorem 3.3, for the case of large values of $t - s$, and assuming an undirected and connected graph. Since $\mathbf{A}$ is symmetric, the Spectral Theorem implies we can decompose as follows (Haykin, 2009), $\mathbf{A}^{t-s} = \sum_{l=1}^{n} \lambda_l^{t-s}\mathbf{q}_l\mathbf{q}_l^\top$, where $\mathbf{q}_l$ is the unitary eigenvector corresponding to the eigenvalue λ_l. Thus, for large values of $t - s$, all the spectral components corresponding to eigenvalues strictly less than 1 (in absolute value) tend to converge to 0. Thus it holds the approximation $\mathbf{A}^{t-s} \approx \mathbf{q}_1\mathbf{q}_1^\top$. Based on the reasoning above, we know that $\mathbf{q}_1$ must be the vector $\mathbf{d} = \text{diag}(\mathbf{D}^{\frac{1}{2}})$ normalised to be unitary. Thus, $\mathbf{q_1} = \dfrac{(\sqrt{1+d_1}, \ldots, \sqrt{1+d_n})}{\sqrt{\sum_{l=1}^{n}(1+d_l)}}$, where d_l is the degree of the l-th node. Therefore, $(\mathbf{q}_1\mathbf{q}_1^\top)_{ij} = \dfrac{\sqrt{(1+d_i)(1+d_j)}}{n+\sum_{l=1}^{n} d_l} = \dfrac{\sqrt{(1+d_i)(1+d_j)}}{n+2m}$, with n the number of nodes, and m the number of edges. Now, in the deep unfolding regime, we can use this approximation to get:

$$\min_{i,j}\left|\left|\frac{\partial \mathbf{X}_t^{(i)}}{\partial \mathbf{X}_s^{(j)}}\right|\right| \approx \frac{1}{n+2m}\left|\left|(\mathbf{W}^\top)^{t-s}\right|\right| \min_{i,j}\sqrt{(1+d_i)(1+d_j)} \geq \frac{2}{n+2m}\left|\left|(\mathbf{W}^\top)^{t-s}\right|\right|,$$

where the last inequality holds since in a connected graph the minimum degree is 1. Thus, we conclude that $\min_{i,j} \mathcal{S}_{ij}[s:t] \geq \frac{2}{n+2m}||(\mathbf{W}^\top)^{t-s}|| = \frac{2}{n+2m}||\mathbf{W}^{t-s}||$, noticing that $||\mathbf{W}^\top|| = ||\mathbf{W}||$.
.

LF case.
Similarly to the GF case, we want to provide an estimation of the term $(\mathbf{A}^{t-s})_{ij}$ for the case of large values of $t-s$. In the LF case, we know that $\mathbf{A}^{t-s}$ is a symmetric and doubly stochastic matrix. We use the decomposition $\mathbf{A}^{t-s} = \sum_{l=1}^{n} \lambda_l^{t-s} \mathbf{q}_l \mathbf{q}_l^\top$, where $\mathbf{q}_l$ is the unitary eigenvector corresponding to the eigenvalue λ_l. As discussed in the proof of the GF case, for large values of $t-s$, all the spectral components corresponding to eigenvalues strictly less than 1 (in absolute value) tend to converge to 0. Our GSO construction (see Appendix B) ensures a strictly positive operator. Note that, the Sinkhorn-Knopp algorithm assumes symmetric balancing by positive diagonal matrices, thus strict positivity is preserved. Thus the resulting operator is primitive. Because the balanced operator is doubly stochastic, $\mathbf{A1} = \mathbf{1}$; Perron-Frobenius makes this eigenvalue simple, and its unit eigenvector is $\mathbf{q_1} = \frac{(1,\dots,1)}{\sqrt{n}}$. Thus, for large values of $t-s$, it holds the approximation $\mathbf{A}^{t-s} \approx \mathbf{q_1}\mathbf{q_1}^\top$. Therefore, $(\mathbf{q_1}\mathbf{q_1}^\top)_{ij} = \frac{1}{n}$. Now, in the deep unfolding regime, we can use this approximation to get that $\min_{i,j} \left|\left|\frac{\partial \mathbf{X}_t^{(i)}}{\partial \mathbf{X}_s^{(j)}}\right|\right| \approx \frac{1}{n}\left|\left|\mathbf{W}^{t-s}\right|\right|$. Thus, we conclude that $\min_{i,j} \mathcal{S}_{ij}[s:t] \geq \frac{1}{n}||\mathbf{W}^{t-s}||$, noticing that $||\mathbf{W}^\top|| = ||\mathbf{W}||$. □

C.3.1 Example of a bad scenario for Equation (11)

Figure 3 illustrates an example of a bad scenario for Equation (11), i.e., a chain of m cliques of order d connected via bridge-nodes of degree 2 (the minimum to connect them). In the Figure, we consider $m = 6$ and $d = 10$. The pair of bridge nodes i and j depicted in red in Figure 3 are 12 hops apart, so it can be considered a relatively long-term interaction.

In the long-term approximation given by Equation (11), the local sensitivity between two bridge nodes of this topology scales as $\frac{1}{md^2}$, for long chains (m large) and big cliques (d large). In fact, in such a graph the vast majority of nodes have degree approximately $d-1$, thus $\sum_{l=1}^{n} d_l \approx n(d-1)$. Specifically, there are exactly $m-1$ nodes of degree 2 (bridge nodes), and md nodes with degree approximately $d-1$. Now, $n = m - 1 + md \approx md$, therefore $n + \sum_{l=1}^{n} d_l \approx n + n(d-1) = nd \approx md^2$. Scaling to long chains and large cliques, this approximation becomes more accurate, and so the local sensitivity between two bridge nodes is rescaled by the term $\frac{\sqrt{(1+d_i)(1+d_j)}}{n+\sum_{l=1}^{n} d_l} \approx \frac{3}{md^2}$.

C.4 Proof of Theorem 3.9

Theorem. For the case of LF and LTV, the global sensitivity of the unrolled linear message-passing dynamics of a BUMP block is lower bounded as follows:

$$\frac{1}{n}||\mathbf{W}^{t-s}|| \leq \mathcal{S}[s:t],$$

where n is the number of nodes. The same lower bound holds true for the case of GF by assuming an undirected and connected graph.

Proof. We divide the proof in 2 cases: first GF, and then both LF and LTV cases.
GF case.
By Equations (8), (9) and (15), we get $\mathcal{S}[s:t] = \max_{i,j} |(\mathbf{A}^{t-s})_{ij}|||(\mathbf{W}^\top)^{t-s}|| = \max_{i,j} |(\mathbf{A}^{t-s})_{ij}|||\mathbf{W}^{t-s}||$. Let n be the number of nodes. The square of the maximum entry of an (n,n) matrix $\mathbf{M}$ is always greater than the arithmetic mean of all the square coefficients, in other words, $\frac{||\mathbf{M}||_F^2}{n^2} \leq \max_{i,j} \mathbf{M}_{i,j}^2$, where $||\mathbf{M}||_F$ denotes the Frobenius norm. Therefore, $\frac{||\mathbf{M}||_F}{n} \leq \max_{i,j} |\mathbf{M}_{i,j}|$. Now, the symmetry of $\mathbf{A}$ implies there are $\lambda_1, \dots, \lambda_n$ real eigenvalues with corresponding orthonormal eigenvectors $\mathbf{q}_1, \dots, \mathbf{q}_n$ so that we can decompose $\mathbf{A}^{t-s} = \sum_{l=1}^{n} \lambda_l^{t-s} \mathbf{q}_l \mathbf{q}_l^\top$. Thus, the Frobenius norm is $||\mathbf{A}^{t-s}||_F = \sqrt{\sum_{l=1}^{n} \lambda_l^{2(t-s)} \, ||\mathbf{q}_l||^2} = \sqrt{\sum_{l=1}^{n} \lambda_l^{2(t-s)}} \geq |\lambda_1|^{t-s}$,

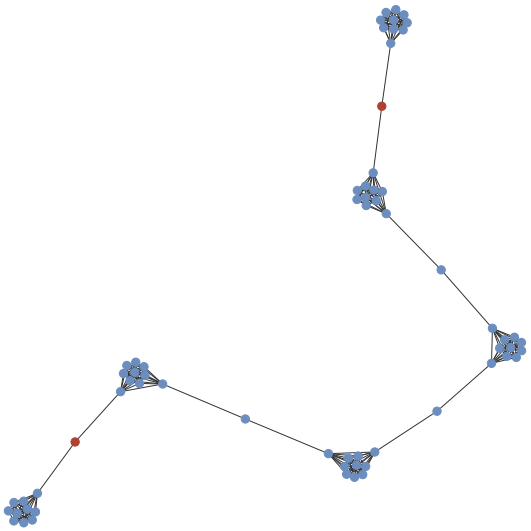

Figure 3: A chain of six cliques (containing ten nodes each) connected via bridge-nodes of degree 2. The pair of red nodes is a pair of nodes that minimizes the quantity in Equation (11). Note that the red nodes are 12 hops apart, so it can be considered long-term.

where $|\lambda_1|$ is the largest in absolute value between all the eigenvalues, i.e., the spectral radius $\rho(\mathbf{A})$.

$$\max_{i,j} |(\mathbf{A}^{t-s})_{ij}| \geq \frac{||\mathbf{A}^{t-s}||_F}{n} \geq \frac{\rho(\mathbf{A})^{t-s}}{n}, \tag{38}$$

from which we get the thesis

$$\mathcal{S}[s:t] = \max_{i,j} |(\mathbf{A}^{t-s})_{ij}| \, ||\mathbf{W}^{t-s}|| \geq \frac{\rho(\mathbf{A})^{t-s}}{n} ||\mathbf{W}^{t-s}||.$$

Finally, the spectral radius $\rho(\mathbf{A})$ is exactly 1 due to Theorem 3.4.

LF and LTV cases.

In this proof, we denote by $\mathbf{A}_{[s:t]}$ either the power matrix $\mathbf{A}^{t-s}$ (the LF case) or the product of matrices $\prod_{l=s+1}^{t} \mathbf{A}_l$ (the LTV case); recalling that in both cases the resulting matrix is a doubly stochastic matrix. By Equations (8), (9) and (15), we get $\mathcal{S}[s:t] = \max_{i,j} |(\mathbf{A}_{[s:t]})_{ij}| ||(\mathbf{W}^\top)^{t-s}|| = \max_{i,j} |(\mathbf{A}_{[s:t]})_{ij}| ||\mathbf{W}^{t-s}||$. Let n be the number of nodes. Then the maximum entry of a doubly stochastic matrix of dimension n is lower-bounded by $\frac{1}{n}$, otherwise the sum on the rows cannot reach 1. Therefore

$$\max_{i,j} |(\mathbf{A}_{[s:t]})_{ij}| \geq \frac{1}{n}, \tag{39}$$

from which we get the thesis

$$\mathcal{S}[s:t] = \max_{i,j} |(\mathbf{A}^{t-s})_{ij}| \, ||\mathbf{W}^{t-s}|| \geq \frac{1}{n} ||\mathbf{W}^{t-s}||.$$

□

D THE VANISHING GRADIENT TENDENCY IN NONLINEAR MPNNS

For an BUMP block, the local sensitivity $\mathcal{S}_{ij}[s:t]$ of the features of the i-th node to features of the j-th node after $t-s$ applications of message-passing aggregations, is exactly the norm of the Jacobian of Equation (9), i.e. the norm of the product of the (i,j)-entry of $\mathbf{A}^{t-s}$ and the matrix $(\mathbf{W}^\top)^{t-s}$. For standard MPNN approaches, the local sensitivity has a more complicated expression due to nonlinearities at each aggregation step, but usually there are 3 key contributors: one from several multiplications of the shift operator (akin to $\mathbf{A}^{t-s}$ in our BUMP), one from several multiplications of

the weights (akin to $(\mathbf{W}^\top)^{t-s}$ in our BUMP), and one from several multiplications of the derivative of the nonlinearity evaluated on the sequence of embeddings $\mathbf{D}(s), \mathbf{D}(s+1), .., \mathbf{D}(t)$. Usually, the nonlinearity is pointwise, so $\mathbf{D}(t)$ is a diagonal matrix with entries usually in $[0, 1]$, thus contributing to vanishing the gradient more and more at each time step. Hence, if the subsequent multiplications of weights and nonlinearity-based terms tend to vanish, while the powers of the shift operator $\mathbf{A}$ are bounded (as it is for the cases proved in Lemma 3.4 then the local sensitivity tends to vanish *for all pairs of nodes*, for $t - s$ large enough. This will be reflected in the global sensitivity, which also will tend to vanish, for $t - s$ large enough, regardless of the topology of the graph. This demonstrates that global sensitivity effectively quantifies the severity of vanishing gradient issues in MPNN models plagued by this problem. Note further that the local sensitivity of the unrolled linear message-passing dynamics in each block of our BUMP has the exact form of $||(\mathbf{A}^{t-s})_{ij}(\mathbf{W}^\top)^{t-s}||$, while for standard MPNN approaches with nonlinearities at each time step, the vanishing effect will be stronger, as we can formally prove below for the case of ReLU-GCN in Theorem D.1.

In Theorem D.1, we quantify the additional vanishing induced by ReLU activations in deep GCNs with weight-sharing by comparing them to the linear message-passing dynamics of the GF stream of BUMP; the proof is below.

Theorem D.1 (GCN vanishes more than BUMP)**.** *Let us consider a GCN that aggregates information from k hops away, i.e., with k layers, equipped with the ReLU activation function. Then, the GCN vanishes information at a $2^{-\frac{k}{2}}$ faster rate than our BUMP block with k linear message-passing steps.*

Proof. The state-update equation of a GCN with a residual connection is $\mathbf{X}_{t+1} = \sigma(\mathbf{A}\mathbf{X}_t\mathbf{W} + \mathbf{X}_t)$. Therefore, the features of i-th node at time $t+1$ are updated as $\mathbf{X}^{(i)}_{t+1} = \sigma\Big(\sum_{l=1}^{n}(\mathbf{A})_{il}\mathbf{X}^{(l)}_t\mathbf{W} + \mathbf{X}^{(i)}_t\Big)$. Similarly to the proof of Theorem 3.3, we can write

$$\frac{\partial \mathbf{X}^{(i)}_{t+1}}{\partial \mathbf{X}^{(j)}_t} = \frac{\partial}{\partial \mathbf{X}^{(j)}_t}\Big(\sigma\Big((\mathbf{A})_{ij}\mathbf{X}^{(j)}_t\mathbf{W}\Big)\Big) = \\ = \mathrm{diag}\Big(\sigma'\Big((\mathbf{A})_{ij}\mathbf{X}^{(j)}_t\mathbf{W}\Big)\Big)(\mathbf{A})_{ij}\mathbf{W}^\top,$$

where we assumed that $i \neq j$, so that the residual connection term does not appear in the derivative w.r.t. $\mathbf{X}^{(j)}_t$. Since we are considering $\sigma = \mathrm{ReLU}$, the diagonal entries $\sigma'\Big((\mathbf{A})_{ij}\mathbf{X}^{(j)}_t\mathbf{W}\Big)$ are either 0 or 1. Let's assume that the components of the vector $\sigma'\Big((\mathbf{A})_{ij}\mathbf{X}^{(j)}_t\mathbf{W}\Big)$ are independent and identically distributed (i.i.d.) Bernoulli random variables, each with probability $\frac{1}{2}$ of taking the value 0. Now, let's consider a walk $\{(i_t, j_t)\}_{t=0}^{k-1}$ of length k connecting the j-th node at a reference time $t = 0$ to the i-th node at time $t = k$. Then, the Jacobian of GCN along such a walk reads:

$$\frac{\partial \mathbf{X}^{(i)}_k}{\partial \mathbf{X}^{(j)}_0} = \prod_{t=0}^{k-1}\mathbf{P}_t\mathbf{M}_t,$$

where $\mathbf{P}_t = \mathrm{diag}\Big(\sigma'\Big((\mathbf{A})_{i_tj_t}\mathbf{X}^{(j_t)}_t\mathbf{W}\Big)\Big)$, and $\mathbf{M}_t = (\mathbf{A})_{i_tj_t}\mathbf{W}^\top$. On the other hand, the Jacobian of the linear recurrent equation (3) of a BUMP block, assuming a given fixed GSO, with a number k of linear recurrent steps computed along the same walk reads:

$$\frac{\partial \mathbf{X}^{(i)}_k}{\partial \mathbf{X}^{(j)}_0} = \prod_{t=0}^{k-1}\mathbf{M}_t.$$

We aim to prove that, for a generic vector $\mathbf{x}$ with entries i.i.d. random variables distributed symmetrically about zero (e.g. according to a Normal distribution with zero mean), it holds the approximation $||\prod_{t=0}^{k-1}\mathbf{P}_t\mathbf{M}_t\mathbf{x}|| \approx 2^{-\frac{k}{2}}||\prod_{t=0}^{k-1}\mathbf{M}_t\mathbf{x}||$. We prove the thesis using a recursive argument. First, we observe that, denoting $\mathbf{y} = \mathbf{M}_0\mathbf{x}$, then we can write

$$||\mathbf{P}_0\mathbf{M}_0\mathbf{x}||^2 = ||\mathbf{P}_0\mathbf{y}||^2 = (p_1y_1)^2 + \ldots + (p_ny_n)^2. \tag{40}$$

Now, since the p_i are assumed i.i.d. Bernoulli random variables, each with probability $\frac{1}{2}$ of taking the value 0, in the sum of (40), roughly a portion of half of the contributions from $\mathbf{y}$ are zeroed out

due to the action of $\mathbf{P}_0$. Therefore,

$$||\mathbf{P}_0\mathbf{M}_0\mathbf{x}||^2 = ||\mathbf{P}_0\mathbf{y}||^2 \approx \frac{1}{2}||\mathbf{y}||^2 = \frac{1}{2}||\mathbf{M}_0\mathbf{x}||^2. \tag{41}$$

Note that the larger the dimension of the graph n, the more accurate the approximation of (41). Therefore, we conclude that $||\mathbf{P}_0\mathbf{M}_0\mathbf{x}|| \approx 2^{-\frac{1}{2}}||\mathbf{M}_0\mathbf{x}||$. Now, we proceed recursively by denoting $\tilde{\mathbf{x}}_t = \mathbf{P}_{t-1}\mathbf{M}_{t-1}\dots\mathbf{P}_0\mathbf{M}_0\mathbf{x}$, and defining the scalars $c_t = \dfrac{||\mathbf{M}_t\tilde{\mathbf{x}}_t||}{||\tilde{\mathbf{x}}_t||} > 0$, for all $t = 1, \dots, k-1$. Then, we can write

$$\begin{aligned}
&||\mathbf{P}_{k-1}\mathbf{M}_{k-1}\mathbf{P}_{k-2}\mathbf{M}_{k-2}\dots\mathbf{P}_0\mathbf{M}_0\mathbf{x}|| = \\
&= ||\mathbf{P}_{k-1}\mathbf{M}_{k-1}\tilde{\mathbf{x}}_{k-1}|| \approx \\
&\approx 2^{-\frac{1}{2}}||\mathbf{M}_{k-1}\tilde{\mathbf{x}}_{k-1}|| = \\
&= 2^{-\frac{1}{2}}c_{k-1}||\tilde{\mathbf{x}}_{k-1}|| = \\
&= 2^{-\frac{1}{2}}c_{k-1}||\mathbf{P}_{k-2}\mathbf{M}_{k-2}\tilde{\mathbf{x}}_{k-2}|| \approx \\
&\approx 2^{-\frac{1}{2}}c_{k-1}2^{-\frac{1}{2}}c_{k-2}||\tilde{\mathbf{x}}_{k-2}|| \approx \dots \\
&\approx 2^{-\frac{k}{2}}c_{k-1}c_{k-2}\dots c_0||\mathbf{x}||.
\end{aligned}$$

On the other hand, for the case of BUMP, it reads:

$$\begin{aligned}
&||\mathbf{M}_{k-1}\mathbf{M}_{k-2}\dots\mathbf{M}_0\mathbf{x}|| = c_{k-1}||\mathbf{M}_{k-2}\dots\mathbf{M}_0\mathbf{x}|| = \\
&= c_{k-1}c_{k-2}||\mathbf{M}_{k-3}\dots\mathbf{M}_0\mathbf{x}|| = \dots \\
&= c_{k-1}c_{k-2}\dots c_0||\mathbf{x}||.
\end{aligned}$$

This proves that a standard GCN vanishes information $2^{-\frac{k}{2}}$ faster than BUMP.
We assumed weight sharing in the GCN, but the same proof holds assuming different weights $\mathbf{W}_1, \dots, \mathbf{W}_k$ at each GCN layer, by simply using the same exact weight matrices for the linear equation of BUMP. □

In the following section, we further show that ReLU-based GCNs still exhibit strong vanishing effects, even under variance-preserving scaling.

D.1 Additional Vanishing Effects Beyond Variance-Preserving Scaling.

The analysis in Theorem D.1 quantifies the contraction introduced by pointwise nonlinearities through a Bernoulli(0.5) model of the ReLU derivative masks. One may attempt to counteract this contraction by rescaling the activation function (e.g., using $\sqrt{2}\,\mathrm{ReLU}$) or by similarly adopting variance-preserving initialization schemes such as He initialization He et al. (2015), which are designed to maintain stable signal norms in feedforward architectures. These techniques effectively compensate for the *expected* shrinkage caused by the diagonal derivative matrices. Nonetheless, when nonlinear transformations are repeatedly applied through the *same* weight matrix, as in weight-sharing settings or recurrent message-passing, the i.i.d. assumptions underlying variance-preserving theory no longer hold. As observed in prior work on deep and recurrent networks Sussillo & Abbott (2014), repeated application of a fixed operator induces directional compounding of contractions or expansions along its singular directions, leading to a substantially stronger vanishing effect than predicted by independent-layer analyses. In the remainder of this subsection, we discuss this phenomenon in more detail and provide an empirical illustration showing that, even after compensating for Bernoulli contraction via $\sqrt{2}$-scaling, significant gradient decay persists at large depths.

To illustrate this phenomenon, we report a simple controlled experiment in an RNN-like setting (no graph structure) over $k = 1000$ recurrent steps. We compare (i) a ReLU RNN with weight sharing of equation $\mathbf{x}_{t+1} = \mathrm{ReLU}(\mathbf{W}\,\mathbf{x}_t + \mathbf{B}\mathbf{u}_{t+1})$, (ii) the same model with $\sqrt{2}$-scaled ReLU (as in variance-preserving scheme), i.e. $\mathbf{x}_{t+1} = \sqrt{2}\,\mathrm{ReLU}(\mathbf{W}\,\mathbf{x}_t + \mathbf{B}\mathbf{u}_{t+1})$, and (iii) a linear RNN, i.e. $\mathbf{x}_{t+1} = \mathbf{W}\,\mathbf{x}_t + \mathbf{B}\mathbf{u}_{t+1}$. For each setting, we measure the spectral norm of the Jacobian $\left\|\frac{\partial \mathbf{x}_k}{\partial \mathbf{x}_0}\right\|$ over

five independent trials. The recurrent matrix $\mathbf{W} \in \mathbb{R}^{128\times128}$ is sampled from a standard normal distribution and rescaled to have spectral radius 1; $\mathbf{B} \in \mathbb{R}^{128\times10}$ and the inputs $\mathbf{u}_t \in \mathbb{R}^{10}$ are sampled i.i.d. from a standard normal distribution for each time step.

Scenario	$\sqrt{2}$ **scaling**	**Mean Jacobian norm over 5 trials**
Shared W, ReLU	No	5.52×10^{-174}
Shared W, ReLU	Yes	9.74×10^{-26}
Shared W, Linear	No	2.025

Table 8: Jacobian norm $\left\|\frac{\partial \mathbf{x}_k}{\partial \mathbf{x}_0}\right\|$ for $k = 1000$ recurrent steps under different nonlinearities and initialization schemes.

The results in Table 8 show that, even if $\sqrt{2}$-scaling is devised to preserve layerwise variance, the repeated application of the same $\mathbf{W}$ causes contractions or expansions to compound along the same singular directions of $\mathbf{W}$. By contrast, in the feedforward / independent-weights case, each random matrix $\mathbf{W}_t$ rotates and redistributes contraction across directions, which mitigates consistent compounding. For this reason, the Bernoulli-based estimate of $2^{-k/2}$ of Theorem D.1 should be interpreted as a best-case contraction rate. Indeed, for $k = 1000$, the Bernoulli assumption in our theorem predicts $2^{-500} \approx 10^{-150}$, yet our empirical results show an even stronger vanishing effect (around 10^{-175}). The empirical gap of $\approx 10^{25}$ observed in Table 8 is fully consistent with theory: it reflects the additional contraction created by repeatedly applying the same operator, deviating from the i.i.d. assumption. This geometric compounding effect is not captured by the Bernoulli analysis alone, yet it accumulates on top of it and becomes significant at large depths. Thus, the vanishing gap between the unrolled linear message-passing dynamics of BUMP and opportunely rescaled nonlinear GNN models still emerges empirically at large depths.

E Multi-hop Interpretation of a Deep BUMP Architecture

In the scenario of static graphs, a multi-hop interpretation of BUMP is possible when it is restricted to the single propagation stream of GF, i.e., when $\alpha_t^{(1)} = 1$ and $\alpha_t^{(2)} = 0$. However, BUMP is fundamentally different from multi-hop GNNs approaches, since it operates through strictly 1-hop message passing at each iteration and does not perform aggregation from far-away hops by design. Nonetheless, to better understand its behavior in deeper architectures, we explore how a multi-hop perspective can be used for interpretation, drawing contrasts with a representative multi-hop model, Drew (Gutteridge et al., 2023).

For this purpose, let us consider the static case with GF, with the input being the sequence $[\mathbf{U}_1, \ldots, \mathbf{U}_1]$. The linearity of the recurrent equation of an BUMP block allows us to unfold the recurrent equation as follows:

$$\mathbf{X}_{k+1} = \mathbf{A}^{k+1}\mathbf{X}_0\mathbf{W}^{k+1} + \sum_{i=0}^{k} \mathbf{A}^i\mathbf{U}_1\mathbf{B}\mathbf{W}^i. \tag{42}$$

Therefore, assuming a zero initial state and including the MLP into the equation, we have the following expression in the output of the first BUMP block:

$$\mathbf{Y}_{k+1} = \mathrm{MLP}\Big(\sum_{i=0}^{k} \mathbf{A}^i\mathbf{U}_1\mathbf{B}\mathbf{W}^i\Big). \tag{43}$$

Due to the various powers of the shift operator $\mathbf{I}, \mathbf{A}, \mathbf{A}^2, \ldots, \mathbf{A}^k$, we can interpret Equation (43) as a k-hop aggregation of the input graph $\mathbf{U}_1$. Now, the sequence $[\mathbf{Y}_{k+1}, \ldots, \mathbf{Y}_{k+1}]$ is the input to the second BUMP block. Therefore, stacking the second BUMP block, and considering a residual connection from the first BUMP block, we have the following expression in the output of the second BUMP block:

$$\mathbf{Y}_{2(k+1)} = \mathbf{Y}_{k+1} + \mathrm{MLP}\Big(\sum_{i=0}^{k} \mathbf{A}^i\mathbf{Y}_{k+1}\mathbf{B}_2\mathbf{W}_2^i\Big), \tag{44}$$

where $\mathbf{B}_2, \mathbf{W}_2$, are the shared weights of the second BUMP block. In general, in a deep BUMP architecture of s blocks, we have the following expression in the output of the s-th BUMP block:

$$\mathbf{Y}_{s(k+1)} = \mathbf{Y}_{(s-1)(k+1)} + \text{MLP}\Big(\sum_{i=0}^{k} \mathbf{A}^i \mathbf{Y}_{(s-1)(k+1)} \mathbf{B}_s \mathbf{W}_s^i\Big). \tag{45}$$

To reveal the multi-hop view, we denote $\hat{\mathbf{Y}}^{(s)} = \mathbf{Y}_{s(k+1)}$, $\hat{\mathbf{W}}_i^{(s)} = \mathbf{B}_s \mathbf{W}_s^i$, and describe the deep BUMP architecture at the granularity of its blocks, as follows:

$$\hat{\mathbf{Y}}^{(s)} = \hat{\mathbf{Y}}^{(s-1)} + \text{MLP}\Big(\sum_{i=0}^{k} \mathbf{A}^i \hat{\mathbf{Y}}^{(s-1)} \hat{\mathbf{W}}_i^{(s)}\Big). \tag{46}$$

This multi-hop interpretation of a deep BUMP architecture resembles the DRew-GCN architecture (Gutteridge et al., 2023), a multi-hop MPNN employing a dynamically rewired message passing strategy with delay. In fact, the recurrent equation of DRew-GCN, rephrased in our BUMP notation for ease of comparison, is defined as:

$$\mathbf{Y}^{(s+1)} = \mathbf{Y}^{(s)} + \sigma\left(\sum_{i=1}^{s+1} \mathbf{A}(i) \mathbf{Y}^{(s-\tau_\nu(i))} \mathbf{W}_i^{(s)}\right), \tag{47}$$

where $\mathbf{A}(i)$ is the degree-normalised shift operator that considers all the neighbors at an *exact* i hops from each respective root node, $\mathbf{W}_i^{(s)}$ are weight matrices, and $\tau_\nu(i)$ is a positive integer (the *delay*) defining the temporal window for the aggregation of past embeddings. Comparing Equation (46) and Equation (47) we can summarize the following differences:

- DRew aggregates information using $\mathbf{A}(i)$, a function of the GSO that counts neighbors at an *exact* i hops distance, while BUMP considers the powers of the GSO, $\mathbf{A}^i$, thus accounting for all the possible walks of length i. Similarly, the learnable weights in BUMP reflect the architectural bias induced by the recurrence, as they are structured through powers of a base matrix, specifically following the form $\hat{\mathbf{W}}_i^{(s)} = \mathbf{B}_s \mathbf{W}_s^i$.
- DRew nonlinearly aggregates information via a pointwise nonlinearity σ, while BUMP employs a more expressive 2-layers MLP.
- BUMP uses the same features for multi-hop aggregation (corresponding to $\tau_\nu(i) \equiv 0$), whereas DRew aggregates features from previous layers with a delay $\tau_\nu(i) = \max(0, i - \nu)$, effectively introducing a temporal rewiring of the graph.

Although the unfolding of BUMP yields expressions involving powers of the GSO, this resemblance to multi-hop architectures such as DRew (Gutteridge et al., 2023) is purely superficial. Unlike models that aggregate information from distant nodes within a single layer, BUMP performs strictly 1-hop message passing at each iteration. The higher-order GSO terms emerge naturally from the recurrence, not from an architectural bias toward multi-hop aggregation. This formulation, grounded in first principles, preserves the original graph topology and constitutes a structurally distinct approach. We provide in Table 9 a comparison of DRew-GCN (results taken from Gutteridge et al. (2023)) with our BUMP on the Peptides-func and Peptides-struct from the LRGB task (Dwivedi et al., 2022b). Notably, BUMP outperforms DRew-GCN on the Peptides-struct task, suggesting that the structural architectural bias introduced by the recurrence, combined with MLP adaptivity, offers a stronger advantage than aggregating information via rewired connections from delayed past features. In contrast, on the Peptides-func task, the performance of the two models falls within each other's standard deviation, indicating no statistically significant difference between DRew-GCN, despite its dynamic rewiring strategy with delay, and BUMP. In Appendix G we report an extended evaluation on the LRGB benchmark.

F Experimental Details

F.1 Employed baselines

In our experiments, the performance of our method is compared with various state-of-the-art GNN baselines from the literature. Specifically, we consider:

Table 9: Results for Peptides-func and Peptides-struct averaged over 3 training seeds. DRew-GCN results are taken from Gutteridge et al. (2023). We report BUMP's results in the case of a single propagation stream corresponding to GF to adhere to the multi-hop interpretation of our method. The **best** scores are in bold.

Model	Peptides-func AP ↑	Peptides-struct MAE ↓
DRew-GCN	**69.96**$_{\pm 0.76}$	0.2781$_{\pm 0.0028}$
BUMP (ours)	69.93$_{\pm 0.52}$	**0.2458**$_{\pm 0.0017}$

- classical MPNN-based methods, i.e., GCN (Kipf & Welling, 2016), GraphSAGE (Hamilton et al., 2017), GAT (Veličković et al., 2018), GatedGCN (Bresson & Laurent, 2018), GIN (Xu et al., 2019), ARMA (Bianchi et al., 2021), GINE (Hu et al., 2020), GCNII (Chen et al., 2020), CoGNN (Finkelshtein et al., 2024), SGC (Wu et al., 2019a), and APPNP (Gasteiger et al., 2019a);
- heterophily-specific models, i.e., H2GCN (Zhu et al., 2020), CPGNN (Zhu et al., 2021), FAGCN (Bo et al., 2021), GPR-GNN (Chien et al., 2021), FSGNN (Maurya et al., 2022), GloGNN (Li et al., 2022), GBK-GNN (Du et al., 2022), and JacobiConv (Wang & Zhang, 2022);
- physics-inspired MPNNs, i.e., DGC (Wang et al., 2021), GRAND (Chamberlain et al., 2021), GraphCON (Rusch et al., 2022), A-DGN (Gravina et al., 2023), GREAD (Choi et al., 2023), CDE (Zhao et al., 2023), and TDE-GNN (Eliasof et al., 2024b);
- Graph Transformers, i.e., Transformer (Vaswani et al., 2017; Dwivedi & Bresson, 2021), GT (Shi et al., 2021), SAN (Kreuzer et al., 2021), GPS (Rampášek et al., 2022), GOAT (Kong et al., 2023), Exphormer (Shirzad et al., 2023), NAGphormer (Chen et al., 2023), GRIT (Ma et al., 2023), and GraphViT (He et al., 2023);
- Higher-Order DGNs, i.e., DIGL (Gasteiger et al., 2019b), MixHop (Abu-El-Haija et al., 2019), DRew (Gutteridge et al., 2023), and GRED (Ding et al., 2024).
- SSM-based GNN, i.e., Graph-Mamba (Wang et al., 2024a), GMN (Behrouz & Hashemi, 2024), GPS+Mamba (Behrouz & Hashemi, 2024), GGRNN (Ruiz et al., 2020), and GraphSSM (Li et al., 2024).
- Graph-agnostic temporal predictors, i.e., Historical Average (AV), SVR (Smola & Schölkopf, 2004), and FC-LSTM (Sutskever et al., 2014), and VAR (Lu et al., 2016);
- Spatio-temporal GNNs, i.e., DCRNN (Li et al., 2018), GConvGRU (Seo et al., 2018), Graph WaveNet (Wu et al., 2019b), ASTGCN (Guo et al., 2019), STSGCN (Song et al., 2020), GMAN (Zheng et al., 2020), MTGNN (Wu et al., 2020b), AGCRN (Bai et al., 2020), T-GCN (Zhao et al., 2020), DyGrAE (Taheri & Berger-Wolf, 2020), EGCN-O (Pareja et al., 2020), A3T-GCN (Bai et al., 2021), MPNN LSTM (Panagopoulos et al., 2021), GTS (Shang et al., 2021), STEP (Shao et al., 2022), GC-LSTM (Chen et al., 2022), Dyn-GESN (Micheli & Tortorella, 2022), HMM4G (Errica et al., 2023), STAEformer (Liu et al., 2023), RGDAN (Fan et al., 2024), AdpSTGCN (Zhang et al., 2024), STD-MAE (Gao et al., 2024), GConvLSTM (Seo et al., 2018), and EGCN-H (Pareja et al., 2020).

F.2 Datasets statistics

In our experiments, we compute the performance of our BUMP on widely used benchmarks for both static and temporal graphs. Specifically, we consider:

- long-range propagation tasks, i.e., the three graph property prediction tasks proposed by Gravina et al. (2023) ("Diameter", "SSSP", and "Eccentricity") and the "Peptide-func" and "Peptide-struct" tasks from the long-range graph benchmark (Dwivedi et al., 2022b);
- heterophilic tasks, i.e., "Roman-empire", "Amazon-ratings", "Minesweeper", "Tolokers", and "Questions" (Platonov et al., 2023);
- temporal tasks, i.e., "Metr-LA" and "PeMS-Bay" for traffic forecasting (Li et al., 2018), and the "Chickenpox Hungary", "PedalMe London", "Wikipedia math", and "Twitter Tennis" forecasting tasks introduced by Rozemberczki et al. (2021).

In Table 10, we report the statistics of the employed datasets.

Table 10: Dataset statistics

	Task	Nodes	Edges	Graphs (or Timesteps)	Frequency
Static	Diameter	25 - 35	22 - 553	7,040	–
	SSSP	25 - 35	22 - 553	7,040	–
	Eccentricity	25 - 35	22 - 553	7,040	–
	Peptide-func	150.94 (avg)	307.30 (avg)	15,535	–
	Peptide-struct	150.94 (avg)	307.30 (avg)	15,535	–
	Roman-empire	22,662	32,927	1	–
	Amazon-ratings	24,492	93,050	1	–
	Minesweeper	10,000	39,402	1	–
	Tolokers	11,758	519,000	1	–
	Questions	48,921	153,540	1	–
Temporal	Metr-LA	207	1,515	34,272	5 mins
	PeMS-Bay	325	2,369	52,116	5 mins
	Chickenpox Hungary	20	102	522	Weekly
	PedalMe London	15	225	36	Weekly
	Wikipedia math	1,068	27,079	731	Daily
	Twitter Tennis RG	1,000	varies	120	Hourly

F.3 Hyperparameter space

In Table 11, we report the grid of hyperparameters employed in our experiments by our method on all the considered benchmarks.

Table 11: The grid of hyperparameters employed during model selection for the graph property prediction tasks (*GPP*), Long Range Graph Benchmark (*LRGB*), heterophilic benchmarks (*Hetero*), and spatio-temporal benchmarks (*Temporal*).

Hyperparameters	Values			
	GPP	***LRGB***	***Hetero***	***Temporal***
Optimizer	Adam	AdamW	AdamW	AdamW
Learning rate	0.003	0.001, 0.0005, 0.0001	0.001, 0.0005 ,0.0001	0.005, 0.001, 0.0005 ,0.0001
Weight decay	10^{-6}	0, 0.0001, 0.001	0, 0.0001, 0.001	0, 0.0001, 0.001
Dropout	0	0, 0.5	0, 0.4, 0.5, 0.6,	0, 0.5
N. recurrences	1, 5, 10, 20	1, 2, 4, 8, 16	1, 2, 4, 8, 16	1, 2, 4, 8, 16
Embedding dim	10, 20, 30	32,64,128,256	32,64,128,256	32,64,128,256
Attention dim	16	32,64,128,256	32,64,128,256	32,64,128,256
top-k_{attn}	–	32,64,128	32,64,128	32,64,128
N. Blocks	1, 2	1, 2, 4, 8, 16	1, 2, 4, 8, 16	1, 2, 4, 8, 16
Structure of $\mathbf{U}$	$\mathbf{U} = [\mathbf{U}_1, \dots, \mathbf{U}_1]$			$\mathbf{U} = [\mathbf{U}_1, \mathbf{U}_2, \dots]$

G Results on the Long-Range Graph Benchmark

To further evaluate the performance of our BUMP, we consider two tasks of the Long-Range Graph Benchmark (LRGB) (Dwivedi et al., 2022b).

Setup. We evaluate BUMP on the Peptides-func and Peptides-struct tasks from the LRGB benchmark, which involve predicting functional and structural properties of peptides that require modeling long-range dependencies. We follow the original experimental setup and 500k parameter budget.

Results. As shown in Table 12, BUMP outperforms standard MPNNs, transformer-based GNNs, and most multi-hop and SSM-based models. It achieves the highest average ranking across tasks without relying on global attention or graph rewiring. Compared to other graph SSMs, BUMP delivers strong performance while preserving permutation-equivariance.

Ablations. As discussed in Section 2, the MLP in Equation (5) is implemented as a standard MLP with 2 linear layers and a nonlinearity in between. To better understand the role of the nonlinearity

Table 12: Results for Peptides-func and Peptides-struct averaged over 3 training seeds. Re-evaluated methods employ the 3-layer MLP readout proposed in Tönshoff et al. (2023). Note that all MPNN-based methods include structural and positional encoding. The **first**, **second**, and **third** best scores are colored. Baseline results are reported from Dwivedi et al. (2022b); Gutteridge et al. (2023); Tönshoff et al. (2023); He et al. (2023); Ding et al. (2024); Gravina et al. (2025). ‡ means 3-layer MLP readout and residual connections are employed.

Model	Peptides-func AP ↑	Peptides-struct MAE ↓	avg. Rank ↓
MPNNs			
A-DGN	$59.75_{\pm 0.44}$	$0.2874_{\pm 0.0021}$	28.0
GatedGCN	$58.64_{\pm 0.77}$	$0.3420_{\pm 0.0013}$	31.0
GCN	$59.30_{\pm 0.23}$	$0.3496_{\pm 0.0013}$	31.5
GCNII	$55.43_{\pm 0.78}$	$0.3471_{\pm 0.0010}$	32.5
GINE	$54.98_{\pm 0.79}$	$0.3547_{\pm 0.0045}$	34.0
GRAND	$57.89_{\pm 0.62}$	$0.3418_{\pm 0.0015}$	31.0
GraphCON	$60.22_{\pm 0.68}$	$0.2778_{\pm 0.0018}$	26.0
SWAN	$67.51_{\pm 0.39}$	$0.2485_{\pm 0.0009}$	13.5
Multi-hop GNNs			
DIGL+MPNN	$64.69_{\pm 0.19}$	$0.3173_{\pm 0.0007}$	27.0
DIGL+MPNN+LapPE	$68.30_{\pm 0.26}$	$0.2616_{\pm 0.0018}$	17.5
DRew-GatedGCN	$67.33_{\pm 0.94}$	$0.2699_{\pm 0.0018}$	20.5
DRew-GatedGCN+LapPE	$69.77_{\pm 0.26}$	$0.2539_{\pm 0.0007}$	13.0
DRew-GCN	$69.96_{\pm 0.76}$	$0.2781_{\pm 0.0028}$	15.5
DRew-GCN+LapPE	**71.50**$_{\pm 0.44}$	$0.2536_{\pm 0.0015}$	8.5
DRew-GIN	$69.40_{\pm 0.74}$	$0.2799_{\pm 0.0016}$	19.0
DRew-GIN+LapPE	**71.26**$_{\pm 0.45}$	$0.2606_{\pm 0.0014}$	10.0
GRED	$70.85_{\pm 0.27}$	$0.2503_{\pm 0.0019}$	8.0
MixHop-GCN	$65.92_{\pm 0.36}$	$0.2921_{\pm 0.0023}$	24.5
MixHop-GCN+LapPE	$68.43_{\pm 0.49}$	$0.2614_{\pm 0.0023}$	16.5
Transformers			
GraphGPS+LapPE	$65.35_{\pm 0.41}$	$0.2500_{\pm 0.0005}$	16.5
Graph ViT	$69.42_{\pm 0.75}$	**0.2449**$_{\pm 0.0016}$	6.0
GRIT	$69.88_{\pm 0.82}$	$0.2460_{\pm 0.0012}$	6.0
Transformer+LapPE	$63.26_{\pm 1.26}$	$0.2529_{\pm 0.0016}$	21.0
SAN+LapPE	$63.84_{\pm 1.21}$	$0.2683_{\pm 0.0043}$	23.5
Modified and Re-evaluated‡			
DRew-GCN+LapPE	$69.45_{\pm 0.21}$	$0.2517_{\pm 0.0011}$	12.0
GatedGCN	$67.65_{\pm 0.47}$	$0.2477_{\pm 0.0009}$	12.0
GCN	$68.60_{\pm 0.50}$	$0.2460_{\pm 0.0007}$	8.5
GINE	$66.21_{\pm 0.67}$	$0.2473_{\pm 0.0017}$	13.0
GraphGPS+LapPE	$65.34_{\pm 0.91}$	$0.2509_{\pm 0.0014}$	18.0
Graph SSMs			
GMN	$70.71_{\pm 0.83}$	$0.2473_{\pm 0.0025}$	**5.5**
Graph-Mamba	$67.39_{\pm 0.87}$	$0.2478_{\pm 0.0016}$	13.5
Ours			
BUMP-GF	$69.93_{\pm 0.52}$	**0.2458**$_{\pm 0.0017}$	**5.0**
BUMP-LF	$65.09_{\pm 0.74}$	$0.2728_{\pm 0.0022}$	23.5
BUMP	**70.89**$_{\pm 0.75}$	**0.2452**$_{\pm 0.0019}$	**2.5**

and MLP's depth, we ablate in Table 13 on the performance of ReLU, GELU, and ELU functions, and in Table 14 we ablate over MLP depths of 1, 2, and 3. We restrict the ablation to the case of a single propagation stream corresponding to the GF configuration, i.e., when $\alpha_t^{(1)} = 1$ and $\alpha_t^{(2)} = 0$. Our results show that the performance of the tested nonlinearities is statistically similar, while a two-layer MLP is a good balance between computational demand and performance.

H Complexity and Runtimes.

We discuss the theoretical complexity of our method, followed by a comparison of runtimes with other methods.

Complexity Analysis. Our BUMP consists of a stack of blocks. Each of them performs a linear recurrence of k iterations followed by the application of a nonlinear map, as defined in Equations (3) and (5). Note that k is either the length of the temporal graph sequence or a hyperparameter. Given the similarities between the linear recurrence in BUMP and standard MPNNs, described in Section 2,

Table 13: Results for Peptides-func and Peptides-struct averaged over 3 training seeds for three different nonlinearities in the MLP of Equation (5), i.e., ReLU, GELU, and ELU, in the case of a single propagation stream corresponding to the GF configuration.

Model	**Peptides-func** AP ↑	**Peptides-struct** MAE ↓
BUMP w/ ReLU	$69.93_{\pm 0.52}$	$0.2458_{\pm 0.0017}$
BUMP w/ GELU	$69.88_{\pm 0.49}$	$0.2456_{\pm 0.0018}$
BUMP w/ ELU	$69.95_{\pm 0.60}$	$0.2459_{\pm 0.0011}$

Table 14: Results for Peptides-func and Peptides-struct averaged over 3 training seeds for three different depths of the MLP in Equation (5) in the case of a single propagation stream corresponding to the GF configuration.

Model	**Peptides-func** AP ↑	**Peptides-struct** MAE ↓
BUMP w/ 1 layer MLP	$69.12_{\pm 0.43}$	$0.2461_{\pm 0.0009}$
BUMP w/ 2 layers MLP	$69.93_{\pm 0.52}$	$0.2458_{\pm 0.0017}$
BUMP w/ 3 layers MLP	$69.91_{\pm 0.57}$	$0.2451_{\pm 0.0014}$

the recurrence retains the complexity of standard MPNNs. Therefore, the Equation (3) is linear in the number of node $|V|$ and edges $|E|$, achieving a time complexity of $\mathcal{O}(k \cdot (|V| + |E|))$, with k the number of iterations. Considering $\mathcal{O}(m)$ the time complexity of the MLP/fusion in Equations (5) to (7), then the final time complexity of one BUMP block is $\mathcal{O}(k \cdot (|V| + |E|) + m)$ in the static case and $\mathcal{O}(k \cdot (|V| + |E| + m))$ in the temporal case. For the stream where the GSO is learned, the complexity depends on whether the operator is computed via full attention or implicit linear attention. If the learned GSO is fully dense, computing the attention scores requires evaluating all pairs of nodes, yielding a quadratic dependence on the number of nodes. The time complexity per block becomes $\mathcal{O}\big(k \cdot |V|^2 + m\big)$ in the static case, and $\mathcal{O}\big(k \cdot (|V|^2 + m)\big)$ in the temporal case, as both the attention computation and the MLP/fusion are applied at each time step. In contrast, the implicit linear attention construction in Section B avoids materializing the dense $|V| \times |V|$ learned GSO. Kernel-matrix products are evaluated directly through the low-rank feature maps, with a cost linear in the number of nodes for fixed feature and channel dimensions. The same implicit computation is used during symmetric Sinkhorn normalization; since we perform a fixed number of five balancing iterations, normalization also remains linear in $|V|$. Consequently, the learned stream preserves a time complexity of $\mathcal{O}\big(k \cdot |V| + m\big)$ in the static case and $\mathcal{O}\big(k \cdot (|V| + m)\big)$ in the temporal case, up to fixed feature-dimension factors. Moreover, neither the kernel nor the learned GSO needs to be explicitly stored, avoiding the quadratic memory requirement of dense attention. This allows the model to scale efficiently to large graphs while retaining the flexibility of attention-based propagation.

Runtimes. We provide runtimes for BUMP and compare it with other methods, such as Graph GPS and GCN, in Table 15. In all cases, we use a model with 256 hidden dimensions and a varying depth effective by changing the number of recurrences from 2 to 16 in our BUMP with 2 BUMP blocks, and the number of layers is the depth for other methods. We report the training and inference times in milliseconds, as well as the downstream performance obtained on the Roman-Empire dataset. As shown in the table, BUMP delivers stronger performance than graph transformers at only a fraction of their computational cost, i.e., maintaining a runtime comparable to GCN, which scales linearly with the graph size. Notably, our BUMP achieves better performance than GCN and GPS, and maintains its performance as depth increases, different than GCN. All runtimes are measured on an NVIDIA A6000 GPU with 48GB of memory.

Table 15: Training and Inference Runtime (milliseconds) and obtained node classification accuracy (%) on the Roman-Empire dataset. BUMP is in the case of a single propagation stream corresponding to the GF configuration.

Metrics	Method	Depth			
		4	8	16	32
Training (ms)	GCN	18.38	33.09	61.86	120.93
Inference (ms)		9.30	14.64	27.95	53.55
Accuracy (%)		73.60	61.52	56.86	52.42
Training (ms)	GPS	1139.05	2286.96	4545.46	OOM
Inference (ms)		119.10	208.26	427.89	OOM
Accuracy (%)		81.97	81.53	81.88	OOM
Training (ms)	$GPS_{GAT+Performer}$ (RWSE)	1179.08	2304.77	4590.26	OOM
Inference (ms)		120.11	209.98	429.03	OOM
Accuracy (%)		84.89	87.01	86.94	OOM
Training (ms)	BUMP	23.19	41.44	72.09	141.82
Inference (ms)		10.93	18.87	38.87	67.59
Accuracy (%)		85.73	88.02	90.82	90.91

I Ablations

I.1 From plain MPNN to BUMP

We perform an ablation study to isolate the incremental contribution of each BUMP's components to the performance gains in reconstructing graph-structural information that depends on learning long-range dependencies; specifically for computing quantities like the diameter of a graph, the single-source-shortest-paths (SSSP), and the eccentricity of a node, see Section 4.1 for more details on these tasks. Results of this ablation are reported in Table 16.

Table 16: Architecture ablation study. Mean test $log_{10}(MSE)$ and std averaged on 4 random weight initialization on Graph Property Prediction tasks (Section 4.1). The lower, the better. The evaluation include: a nonlinear multilayer GCN (`GCN`); a linear multilayer GCN (`Linear GCN`); a linear multilayer GCN with weight sharing (`Linear GCN (ws)`); Linear GCN (ws) followed by an MLP (`1 Block Linear GCN`); a stack of multiple 1 Block Linear GCN (`Multi-Blocks Linear GCN`); two multi-blocks linear GNN with standard deep learning heuristics, such as residual connections and normalisation layers between blocks, using either a given or a learned GSO fixed within blocks (`BUMP-GF` and `BUMP-LF`, respectively); and our BUMP, which fuses BUMP-GF and BUMP-LF to perform a bi-stream linear message passing.

Model	**Diameter ↓**	**SSSP ↓**	**Eccentricity ↓**
GCN	$0.7424_{\pm 0.0466}$	$0.9499_{\pm 0.0001}$	$0.8468_{\pm 0.0028}$
Linear GCN	$-2.1255_{\pm 0.0984}$	$-1.5822_{\pm 0.0002}$	$-2.1424_{\pm 0.0014}$
Linear GCN (ws)	$-2.2678_{\pm 0.1277}$	$-1.5823_{\pm 0.0001}$	$-2.1447_{\pm 0.001}$
1 Block Linear GCN	$-2.2734_{\pm 0.1513}$	$-1.5836_{\pm 0.0025}$	$-2.1869_{\pm 0.0058}$
Multi-Blocks Linear GCN	$-2.3531_{\pm 0.3183}$	$-1.5821_{\pm 0.0001}$	$-2.1861_{\pm 0.0066}$
BUMP-GF	$-3.2353_{\pm 0.1735}$	$-4.6321_{\pm 0.0779}$	$-2.9724_{\pm 0.0271}$
BUMP-LF	$-1.3277_{\pm 0.0008}$	$-1.8279_{\pm 0.0003}$	$-1.8834_{\pm 0.0002}$
BUMP	$\mathbf{-3.3421}_{\pm 0.0447}$	$\mathbf{-4.8994}_{\pm 0.0306}$	$\mathbf{-2.8257}_{\pm 0.3003}$

We devised an ablation aimed to incrementally add/remove components starting from a plain GCN and ending with the full BUMP architecture. We first remove the nonlinearity from a GCN (second row in Table 16), then add weight sharing to obtain a linear recurrence (third row), then introduce a shared MLP over the recurrent steps to obtain an BUMP block (fourth row), next stack multiple BUMP blocks (fifth row), and finally add residual connections and normalization between blocks and a bi-stream message passing (last row).

The ablation conducted reveals that removing the nonlinearity from GCN yields a significant performance improvement. Introducing weight sharing, effectively incorporating recurrence into the

linear graph diffusion process, yields a slight performance boost while considerably reducing the number of parameters. Appending an MLP at the last time step of this linear recurrent architecture does not result in statistically significant gains, except marginally for the Eccentricity task. Likewise, constructing a hierarchical block structure does not noticeably enhance performance. These limited improvements suggest that, for the three tasks considered, the linear recurrence mechanism alone, provided a long enough recurrence, is sufficient to capture meaningful representations to reconstruct graph's structural information. Finally, incorporating standard deep learning heuristics, along with full bi-stream message passing, further strengthens the full BUMP architecture, consistently improving performance across all tasks.

J Extended Comparison on the Heterophilic Benchmark

To further evaluate the performance of BUMP, we report a more complete comparison for the heterophilic task in Table 17. Specifically, we include more MPNN-based models, graph transformers, and heterophily-designated GNNs.

In Table 17, we color the top three methods. Different from the main body of the paper, here we also include sub-variants of methods in the highlighted results, providing an additional perspective on the findings. Notably, our BUMP achieves the best average ranking across all datasets in the heterophilic benchmarks.

K Broader Impact

This work advances machine learning methods for graph-structured data, which may benefit applications such as scientific discovery, recommendation systems, infrastructure optimization, and biological network analysis. As a foundational methodological contribution, the work is not tied to a specific deployment setting. Nevertheless, graph learning methods could potentially be applied in domains involving sensitive relational data, including social networks or user interaction graphs, where concerns related to privacy, surveillance, bias propagation, or unfair decision-making may arise. Since this paper does not introduce a domain-specific deployment or release sensitive datasets, we believe the societal risks are indirect and similar to those associated with graph machine learning research broadly.

Table 17: Mean test set score and std averaged over 4 random weight initializations on heterophilic datasets. The higher, the better. **First**, **second**, and **third** best results for each task are color-coded. Baseline results are reported from Finkelshtein et al. (2024); Behrouz & Hashemi (2024); Platonov et al. (2023); Müller et al. (2024); Luan et al. (2024). "$*$" in the rank column means that the average has been computed over less trials.

Model	Roman-empire Acc ↑	Amazon-ratings Acc ↑	Minesweeper AUC ↑	Tolokers AUC ↑	Questions AUC ↑	avg. Rank ↓
Luan et al. (2024)						
MLP-1	$64.12_{\pm 0.61}$	$38.60_{\pm 0.41}$	$50.59_{\pm 0.83}$	$71.89_{\pm 0.82}$	$70.33_{\pm 0.96}$	43.0
MLP-2	$66.04_{\pm 0.71}$	$49.55_{\pm 0.81}$	$50.92_{\pm 1.25}$	$74.58_{\pm 0.75}$	$69.97_{\pm 1.16}$	36.4
SGC-1	$44.60_{\pm 0.52}$	$40.69_{\pm 0.42}$	$82.04_{\pm 0.77}$	$73.80_{\pm 1.35}$	$71.06_{\pm 0.92}$	40.4
Graph-agnostic						
ResNet	$65.88_{\pm 0.38}$	$45.90_{\pm 0.52}$	$50.89_{\pm 1.39}$	$72.95_{\pm 1.06}$	$70.34_{\pm 0.76}$	39.4
ResNet+adj	$52.25_{\pm 0.40}$	$51.83_{\pm 0.57}$	$50.42_{\pm 0.83}$	$78.78_{\pm 1.11}$	$75.77_{\pm 1.24}$	33.8
ResNet+SGC	$73.90_{\pm 0.51}$	$50.66_{\pm 0.48}$	$70.88_{\pm 0.90}$	$80.70_{\pm 0.97}$	$75.81_{\pm 0.96}$	30.8
MPNNs						
CO-GNN(Σ, Σ)	**91.57**$_{\pm 0.32}$	$51.28_{\pm 0.56}$	$95.09_{\pm 1.18}$	$83.36_{\pm 0.89}$	**80.02**$_{\pm 0.86}$	8.8
CO-GNN(μ, μ)	**91.37**$_{\pm 0.35}$	**54.17**$_{\pm 0.37}$	**97.31**$_{\pm 0.41}$	$84.45_{\pm 1.17}$	$76.54_{\pm 0.95}$	**7.8**
GAT	$80.87_{\pm 0.30}$	$49.09_{\pm 0.63}$	$92.01_{\pm 0.68}$	$83.70_{\pm 0.47}$	$77.43_{\pm 1.20}$	19.4
GAT-sep	$88.75_{\pm 0.41}$	$52.70_{\pm 0.62}$	$93.91_{\pm 0.35}$	$83.78_{\pm 0.43}$	$76.79_{\pm 0.71}$	11.0
GAT (LapPE)	$84.80_{\pm 0.46}$	$44.90_{\pm 0.73}$	$93.50_{\pm 0.54}$	$84.99_{\pm 0.54}$	$76.55_{\pm 0.84}$	17.6
GAT (RWSE)	$86.62_{\pm 0.53}$	$48.58_{\pm 0.41}$	$92.53_{\pm 0.65}$	$85.02_{\pm 0.67}$	$77.83_{\pm 1.22}$	13.0
GAT (DEG)	$85.51_{\pm 0.56}$	$51.65_{\pm 0.60}$	$93.04_{\pm 0.62}$	$84.22_{\pm 0.81}$	$77.10_{\pm 1.23}$	13.8
Gated-GCN	$74.46_{\pm 0.54}$	$43.00_{\pm 0.32}$	$87.54_{\pm 1.22}$	$77.31_{\pm 1.14}$	$76.61_{\pm 1.13}$	33.2
GCN	$73.69_{\pm 0.74}$	$48.70_{\pm 0.63}$	$89.75_{\pm 0.52}$	$83.64_{\pm 0.67}$	$76.09_{\pm 1.27}$	27.4
GCN (LapPE)	$83.37_{\pm 0.55}$	$44.35_{\pm 0.36}$	$94.26_{\pm 0.49}$	$84.95_{\pm 0.78}$	$77.79_{\pm 1.34}$	16.0
GCN (RWSE)	$84.84_{\pm 0.55}$	$46.40_{\pm 0.55}$	$93.84_{\pm 0.48}$	**85.11**$_{\pm 0.77}$	$77.81_{\pm 1.40}$	13.4
GCN (DEG)	$84.21_{\pm 0.47}$	$50.01_{\pm 0.69}$	$94.14_{\pm 0.50}$	$82.51_{\pm 0.83}$	$76.96_{\pm 1.21}$	18.0
SAGE	$85.74_{\pm 0.67}$	$53.63_{\pm 0.39}$	$93.51_{\pm 0.57}$	$82.43_{\pm 0.44}$	$76.44_{\pm 0.62}$	17.2
Graph Transformers						
Exphormer	$89.03_{\pm 0.37}$	$53.51_{\pm 0.46}$	$90.74_{\pm 0.53}$	$83.77_{\pm 0.78}$	$73.94_{\pm 1.06}$	17.8
NAGphormer	$74.34_{\pm 0.77}$	$51.26_{\pm 0.72}$	$84.19_{\pm 0.66}$	$78.32_{\pm 0.95}$	$68.17_{\pm 1.53}$	32.2
GOAT	$71.59_{\pm 1.25}$	$44.61_{\pm 0.50}$	$81.09_{\pm 1.02}$	$83.11_{\pm 1.04}$	$75.76_{\pm 1.66}$	33.0
GPS	$82.00_{\pm 0.61}$	$53.10_{\pm 0.42}$	$90.63_{\pm 0.67}$	$83.71_{\pm 0.48}$	$71.73_{\pm 1.47}$	22.8
$\text{GPS}_{\text{GCN+Performer}}$ (LapPE)	$83.96_{\pm 0.53}$	$48.20_{\pm 0.67}$	$93.85_{\pm 0.41}$	$84.72_{\pm 0.77}$	$77.85_{\pm 1.25}$	14.2
$\text{GPS}_{\text{GCN+Performer}}$ (RWSE)	$84.72_{\pm 0.65}$	$48.08_{\pm 0.85}$	$92.88_{\pm 0.50}$	$84.81_{\pm 0.86}$	$76.45_{\pm 1.51}$	18.2
$\text{GPS}_{\text{GCN+Performer}}$ (DEG)	$83.38_{\pm 0.68}$	$48.93_{\pm 0.47}$	$93.60_{\pm 0.47}$	$80.49_{\pm 0.97}$	$74.24_{\pm 1.18}$	24.4
$\text{GPS}_{\text{GAT+Performer}}$ (LapPE)	$85.93_{\pm 0.52}$	$48.86_{\pm 0.38}$	$92.62_{\pm 0.79}$	$84.62_{\pm 0.54}$	$76.71_{\pm 0.98}$	16.0
$\text{GPS}_{\text{GAT+Performer}}$ (RWSE)	$87.04_{\pm 0.58}$	$49.92_{\pm 0.68}$	$91.08_{\pm 0.58}$	$84.38_{\pm 0.91}$	$77.14_{\pm 1.49}$	16.4
$\text{GPS}_{\text{GAT+Performer}}$ (DEG)	$85.54_{\pm 0.58}$	$51.03_{\pm 0.60}$	$91.52_{\pm 0.46}$	$82.45_{\pm 0.89}$	$76.51_{\pm 1.19}$	21.6
$\text{GPS}_{\text{GCN+Transformer}}$ (LapPE)	OOM	OOM	$91.82_{\pm 0.41}$	$83.51_{\pm 0.93}$	OOM	35.4
$\text{GPS}_{\text{GCN+Transformer}}$ (RWSE)	OOM	OOM	$91.17_{\pm 0.51}$	$83.53_{\pm 1.06}$	OOM	36.0
$\text{GPS}_{\text{GCN+Transformer}}$ (DEG)	OOM	OOM	$91.76_{\pm 0.61}$	$80.82_{\pm 0.95}$	OOM	38.0
$\text{GPS}_{\text{GAT+Transformer}}$ (LapPE)	OOM	OOM	$92.29_{\pm 0.61}$	$84.70_{\pm 0.56}$	OOM	31.8
$\text{GPS}_{\text{GAT+Transformer}}$ (RWSE)	OOM	OOM	$90.82_{\pm 0.56}$	$84.01_{\pm 0.96}$	OOM	35.4
$\text{GPS}_{\text{GAT+Transformer}}$ (DEG)	OOM	OOM	$91.58_{\pm 0.56}$	$81.89_{\pm 0.85}$	OOM	37.8
GT	$86.51_{\pm 0.73}$	$51.17_{\pm 0.66}$	$91.85_{\pm 0.76}$	$83.23_{\pm 0.64}$	$77.95_{\pm 0.68}$	15.6
GT-sep	$87.32_{\pm 0.39}$	$52.18_{\pm 0.80}$	$92.29_{\pm 0.47}$	$82.52_{\pm 0.92}$	$78.05_{\pm 0.93}$	14.0
Heterophily-Designated GNNs						
CPGNN	$63.96_{\pm 0.62}$	$39.79_{\pm 0.77}$	$52.03_{\pm 5.46}$	$73.36_{\pm 1.01}$	$65.96_{\pm 1.95}$	42.0
FAGCN	$65.22_{\pm 0.56}$	$44.12_{\pm 0.30}$	$88.17_{\pm 0.73}$	$77.75_{\pm 1.05}$	$77.24_{\pm 1.26}$	32.6
FSGNN	$79.92_{\pm 0.56}$	$52.74_{\pm 0.83}$	$90.08_{\pm 0.70}$	$82.76_{\pm 0.61}$	**78.86**$_{\pm 0.92}$	19.2
GBK-GNN	$74.57_{\pm 0.47}$	$45.98_{\pm 0.71}$	$90.85_{\pm 0.58}$	$81.01_{\pm 0.67}$	$74.47_{\pm 0.86}$	29.8
GloGNN	$59.63_{\pm 0.69}$	$36.89_{\pm 0.14}$	$51.08_{\pm 1.23}$	$73.39_{\pm 1.17}$	$65.74_{\pm 1.19}$	43.0
GPR-GNN	$64.85_{\pm 0.27}$	$44.88_{\pm 0.34}$	$86.24_{\pm 0.61}$	$72.94_{\pm 0.97}$	$55.48_{\pm 0.91}$	40.2
H2GCN	$60.11_{\pm 0.52}$	$36.47_{\pm 0.23}$	$89.71_{\pm 0.31}$	$73.35_{\pm 1.01}$	$63.59_{\pm 1.46}$	41.4
JacobiConv	$71.14_{\pm 0.42}$	$43.55_{\pm 0.48}$	$89.66_{\pm 0.40}$	$68.66_{\pm 0.65}$	$73.88_{\pm 1.16}$	38.0
Graph SSMs						
GMN	$87.69_{\pm 0.50}$	**54.07**$_{\pm 0.31}$	$91.01_{\pm 0.23}$	$84.52_{\pm 0.21}$	–	12.0^{*}
GPS + Mamba	$83.10_{\pm 0.28}$	$45.13_{\pm 0.97}$	$89.93_{\pm 0.54}$	$83.70_{\pm 1.05}$	–	27.0^{*}
Ours						
BUMP-GF	$90.91_{\pm 0.48}$	$53.65_{\pm 0.71}$	**95.33**$_{\pm 0.72}$	**85.26**$_{\pm 0.93}$	$78.18_{\pm 1.34}$	**3.4**
BUMP-LF	$87.98_{\pm 0.51}$	$50.16_{\pm 0.49}$	$81.90_{\pm 0.74}$	$82.74_{\pm 0.89}$	$76.81_{\pm 1.42}$	22.2
BUMP	**91.72**$_{\pm 0.35}$	**53.81**$_{\pm 0.79}$	**97.45**$_{\pm 0.73}$	**85.44**$_{\pm 1.00}$	**78.68**$_{\pm 1.42}$	**1.8**