Long-Range Spatio-Temporal Graph Propagation Through Oscillations

Alessio Gravina[1,*] **Alessandro Trenta**[1,*] **Tai Hoang**[2,*] **Andrea Ceni**[1]
Gerhard Neumann[2] **Davide Bacciu**[1]

[1] University of Pisa [2] Karlsruhe Institute of Technology

Abstract

Graph Neural Networks (GNNs) are powerful tools for learning from spatio-temporal data, as interactions in space can be naturally described by a graph structure. However, capturing long-range dependencies becomes substantially harder when information must flow through space and time simultaneously. Most existing approaches extend GNNs from static graphs to temporal settings by alternating propagation between time and space. In this work, we introduce STORM, a differential-equation-inspired GNN that exploits oscillatory dynamics to propagate information effectively in the joint spatio-temporal domain. By combining a wave-equation update with dissipative and external forcing terms, STORM balances conservative and non-conservative dynamics. We provide a bottom-up analysis of the model, highlighting its propagation behavior and stability, and show that STORM is universal. We empirically validate our method on diverse benchmarks, including tasks designed for analyzing long-range spatio-temporal dependencies, real-world forecasting, and a new long-range task inspired by physical simulations. Across these settings, STORM consistently matches or outperforms strong baselines, establishing a new state of the art for long-range spatio-temporal graph learning.

1 Introduction

Graph neural networks (GNNs) [82, 100, 12, 24] are nowadays the standard paradigm for learning from graph-structured data, enabling significant advances in fields such as social networks [64, 56] or molecular biology [43, 121, 50, 3]. Despite their success, a persistent challenge in modern deep learning for graphs [11] is the inability of GNNs to maintain effective propagation dynamics over long distances due to over-smoothing [13, 95], over-squashing [1, 25, 85], and more generally vanishing gradients [2].

Modeling long-range dependencies over graphs evolving in time further exacerbates this challenge, since the temporal dimension introduces an additional axis for information flow, amplifying the dissipative effects observed in static graphs [81]. Temporal graph learning [62, 76, 45], in particular, requires architectures that can simultaneously capture spatial interactions, temporal evolution, and long-horizon dependencies without information dissipation. While significant progress has been made by extending GNNs for static graphs to the temporal domain, most existing approaches rely on diffusive dynamics [70, 101, 125], which fundamentally limit their ability to preserve and propagate information over extended space and time scales.

Recent approaches in the Recurrent Neural Network (RNN) domain have shown that oscillatory dynamics can play a pivotal role in stabilizing information flow through time [97, 63, 14, 99].

*Equal contribution. Correspondence to alessio.gravina@di.unipi.it, alessandro.trenta@phd.unipi.it, tai.hoang@kit.edu

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

Specifically, by preventing the common pitfalls of gradient issues [91], these methods effectively enable long-term memory. Similarly, the field of GNNs for static graphs has seen the emergence of architectures that also exploit oscillatory dynamics [30, 96, 114], allowing for effective long-range spatial propagation. By modeling information propagation as a wave-like process, these methods preserve signal energy and enable effective communication across distant nodes. The effectiveness of oscillatory dynamics in both RNNs and GNNs is rooted in the fundamental properties of waves. Indeed, waves can travel indefinitely at a constant speed without losing amplitude or energy [33], and they can also be linearly composed and decomposed, allowing for superposition of waves propagating from different directions [66]. Collectively, these results indicate an underlying principle whereby oscillatory dynamics serve as a structural prior that mitigates information loss, facilitating stable long-range propagation across spatial and temporal dimensions.

Despite these parallel developments, the temporal graph setting remains largely disconnected from this unifying dynamical perspective. Most temporal GNNs combine static message passing [43] with recurrent updates [60, 19], implicitly mixing diffusive spatial dynamics with temporal recurrence, often without explicit control over stability or energy flow. As a result, long-range spatio-temporal dependencies are still difficult to model in a principled and scalable way.

In this work, we introduce STORM (*Spatio-Temporal Oscillatory dynamics via graph Resistance Model*), a temporal graph neural architecture belonging to the class of differential-equation–inspired graph neural networks (DE-GNNs) [75, 44], which brings oscillatory dynamics to the temporal graph domain. STORM extends the wave-based propagation principles of static oscillatory GNNs to evolving graph signals, coupling second-order spatial dynamics with structured temporal recurrence. By explicitly modeling the interaction between oscillations in space and time, STORM provides a unified framework in which memory, propagation, and stability naturally emerge from the underlying dynamics. To incorporate input signals, STORM includes an external forcing term that injects node features and drives the system's evolution. Together with a dissipative component, this term regulates the balance between conservative and non-conservative dynamics, allowing the model to capture a broader range of spatio-temporal patterns. Finally, STORM learns adaptive edge resistances that modulate the wave characteristics along each connection, allowing the model to favor informative pathways and enhance long-range information propagation. Our results demonstrate that embedding oscillatory dynamics into temporal graph models leads to improved stability, stronger long-range dependencies, and better performance across a range of temporal graph benchmarks.

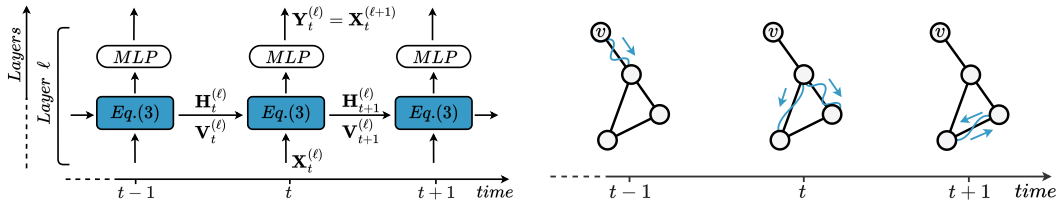
The key contributions of this work are the following: **(i)** We propose STORM, the first temporal GNN architecture based on the wave equation for graphs, which enables balancing and integrating non-dissipative long-range propagation with non-conservative behaviors. Additionally, it incorporates adaptive resistances that allow each edge to modulate the information flow. **(ii)** We theoretically demonstrate how the different components affect the long-range sensitivity of our model and how information is propagated through it, and we prove the stability and universality of STORM. **(iii)** We conduct extensive experiments to demonstrate the benefits of our method. STORM consistently achieves on par or better performance than existing methods across diverse tasks and benchmarks, including a novel physics simulation-based benchmark that we introduce to specifically evaluate long-range spatio-temporal propagation capabilities.

2 STORM

In this section, we introduce STORM (Spatio-Temporal Oscillatory dynamics via graph Resistance Model), a novel temporal GNN architecture in which information propagation is governed by an oscillatory dynamics, enabling long-range spatio-temporal propagation between nodes. Figure 1 illustrates the overall architecture together with the resulting oscillatory dynamics. We report in Appendix A a discussion positioning STORM with respect to existing approaches for long-range propagation and spatio-temporal learning on graphs.

2.1 Preliminaries

We consider a temporal (or dynamic) graph as an evolving system of interacting entities called nodes that is fully observable over time. More formally, given $t_1 < t_2 < \ldots < T$ an ordered set of timestamps, a temporal graph is the sequence of graph snapshots $\mathcal{G}(t) = \{\mathcal{G}_t \,|\, t \in [t_1, T]\}$, where each snapshot $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathbf{X}_t)$ provides an observation of the whole graph's state at a particular



(a) Deep architecture of STORM. (b) Oscillatory propagation originating from node v.

Figure 1: On the left, (a) The STORM sequence-to-sequence architecture. At each layer ℓ and time step t, STORM receives an input signal $\mathbf{X}_t^{(\ell)}$, which is processed through the oscillatory dynamics defined in Equation (3), conditioned on the layer-wise hidden states $\mathbf{H}_t^{(\ell)}$ from the previous time step. Each iteration of Equation (3) produces an intermediate state that is both propagated to the next iteration and transformed by an MLP, yielding a representation that can either be used to generate the layer output $\mathbf{Y}_t^{(\ell)}$ or forwarded as input to the subsequent layer in the stack, i.e., $\mathbf{X}_t^{(\ell+1)}$. On the right, (b) the oscillatory information propagation within a single layer, illustrated with node v acting as the source of the flow. For ease of visualization, we consider a fixed topology in (b).

time t. $\mathcal{V}_t$ is the set of n nodes and $\mathcal{E}_t \subseteq \mathcal{V}_t \times \mathcal{V}_t$ is the set of m edges present at time t. Analogously, $\mathbf{X}_t \in \mathbb{R}^{n \times c}$ defines the node attributes at time t. We note that this definition of temporal graph is often identified as Discrete-Time Dynamic Graph [62, 45] or Snapshot-based Temporal Graph [76]. Following [39] and given a node function $f(v) : \mathcal{V} \to \mathbb{R}$, we can define the graph Laplacian for a specific snapshot $\mathcal{G}_t$ as

$$(\Delta f)(v) = (\widehat{\mathbf{L}} f)(v) = (\mathbf{D} f)(v) - (\mathbf{A} f)(v) = \sum_{u \in \mathcal{N}_t(v)} a_{uv}(f(v) - f(u)) \tag{1}$$

where $\mathbf{D}$ and $\mathbf{A}$ are the weighted degree and adjacency matrices[2] of $\mathcal{G}_t$, $a_{uv} \in \mathbb{R}$ is a weight on the edge $(u, v) \in \mathcal{E}_t$, and $\mathcal{N}_t(v) = \{u \in \mathcal{V}_t | (u, v) \in \mathcal{E}_t\}$ is the temporal neighborhood of node v. Finally, we associate with each node v in the temporal graph a hidden state vector $\mathbf{h}^v(t)$, which represents the time-dependent latent state of node v within the system. Collectively, the hidden states of all nodes at time t are arranged in the matrix $\mathbf{H}(t) = [\mathbf{h}^0(t), \ldots, \mathbf{h}^{n-1}(t)]^\top \in \mathbb{R}^{n \times d}$. We note that in the domain of differential equations, the continuous dynamics expressed by the neural network is discretized through iterative applications of a numerical method over the set of timestamps $t_1, t_2, \ldots, T$ in a process that can be assimilated to that of an RNN [18, 55, 51]. Therefore, given the initial condition $\mathbf{h}^v(0) = \mathbf{0}$, the t-th step of the discretization computes the node states $\mathbf{h}_t^v$ which approximate $\mathbf{h}^v(t)$.

To ease notation, in Section 2.2 we assume a time-invariant graph topology, such that the node and edge sets satisfy $\mathcal{V}_t = \mathcal{V}$ and $\mathcal{E}_t = \mathcal{E}$ for all t, similarly to [81], and thus drop the time index t from $\widehat{\mathbf{L}}$. We retain this assumption in Section 3, although some results also extend to time-varying topologies (see Appendix C.6), for which we also provide empirical validation (see Appendix E.2).

2.2 Oscillatory Spatio-Temporal Propagation on Graph

To induce oscillatory dynamics in the spatio-temporal information propagation, we designed the evolution of node states to be governed by the following second-order differential equation:

$$\ddot{\mathbf{H}}(t) = -\widehat{\mathbf{L}}\mathbf{H}(t)\mathbf{W} - D(\mathbf{H}(t)) \odot \dot{\mathbf{H}}(t) + F(\mathbf{X}(t), \mathbf{H}(t)), \tag{2}$$

subject to the initial condition $\mathbf{H}(0) = \bar{\mathbf{H}}$, i.e., a transformation of the first input nodes' states. $\widehat{\mathbf{L}}$ is weighted graph Laplacian in Equation (1) which encodes topological information [3]; $\mathbf{W} \in \mathbb{R}^{d \times d}$ is a learnable weight matrix; $F(\cdot)$ is an external forcing term driven by the input signal $\mathbf{X}(t)$ and latent state $\mathbf{H}(t)$; $D(\cdot)$ models dissipative effects; and $\odot$ indicates the Hadamard product.

[2] $\mathbf{D} \in \mathbb{R}^{n \times n}$ is a diagonal matrix such that $\mathbf{D}_{vv} = \sum_{u=0}^{n} \mathbf{A}_{uv}$, and $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the adjacency matrix such that $\mathbf{A}_{uv} = a_{uv} \in \mathbb{R}_{\setminus\{0\}}$ if $(u, v) \in \mathcal{E}$, and zero otherwise. Generally, $a_{uv} = 1$.

[3] In the case of a time-evolving graph topology, the Laplacian becomes time dependent and is accordingly denoted by $\widehat{\mathbf{L}}(t)$ to reflect the changes in $\mathcal{E}_t$.

Analogously to the setting of static graphs [30, 96, 114], the term $\ddot{\mathbf{H}}(t) = -\widehat{\mathbf{L}}\mathbf{H}(t)\mathbf{W}$ induces oscillatory propagation over the graph structure. In the spatio-temporal setting, this mechanism enables signals injected at a given time step to persist in the system, effectively embedding temporal memory directly into the propagation process (as discussed in Section 3). Unlike diffusive dynamics, which progressively attenuate node representations, the oscillatory behavior induced by the Laplacian term allows the model to maintain informative representations over extended spatial and temporal horizons. To further strengthen the model, enabling oscillatory propagation across different physical media, we adaptively learn the weights of the graph Laplacian defined in Equation (1)[4], allowing edges to differentially regulate the transmission of information. Under this formulation, the weights a_{uv} act as resistance terms between nodes u and v; consequently, interactions between pairs of nodes can be interpreted as occurring through heterogeneous media (e.g., air and water), where oscillatory characteristics are edge-specific. In other terms, a_{uv} controls how easily information propagates between nodes u and v.

The forcing term $F(\mathbf{X}(t), \mathbf{H}(t))$ injects external input information into the system, acting as a time-dependent driving force that continuously excites the latent oscillatory system with the observed node features. This mechanism allows temporal inputs to smoothly influence the latent state without abrupt changes in the information propagation dynamics. Conversely, the dissipative term $D(\mathbf{H}(t))$ acts as a learnable friction that regulates the system's energy by selectively damping oscillations, ensuring stability and enabling controlled forgetting when required by the temporal evolution of the system. We detail the implementations of F and D for our experiments in Appendix B.

The formulation of Equation (2) naturally decomposes spatio-temporal propagation into conservative transport, dissipation, and excitation, similarly to damped wave equations [37]. This provides a more general setting than purely conservative oscillatory dynamics, trading between the conservative behavior typical of undamped wave equations and non-conservative dynamics.

Discretization. As for classical differential equations, solving Equation (2) requires the adoption of a numerical discretization scheme. In this work, we employ a finite-difference approach [35] to approximate the continuous-time dynamics of Equation (2), as for other oscillatory-based models in other domains [97, 63, 96, 114]. To this end, we introduce an auxiliary velocity variable defined as the first-order temporal derivative of the hidden states, namely $\mathbf{V}(t) = \dot{\mathbf{H}}(t)$. The resulting system can then be discretized by iteratively updating the hidden states and velocities as

$$\begin{cases} \mathbf{V}_{t+1} = \mathbf{V}_t - \epsilon\Big(\widehat{\mathbf{L}}\mathbf{H}_t\mathbf{W} + D(\mathbf{H}_t) \odot \mathbf{V}_t - F(\mathbf{H}_t, \mathbf{X}_t)\Big), \\ \mathbf{H}_{t+1} = \mathbf{H}_t + \epsilon\mathbf{V}_{t+1} \end{cases} \tag{3}$$

where ϵ denotes the discretization step size. We initialize the system with $\mathbf{H}_0 = \bar{\mathbf{H}}_0$ and $\mathbf{V}_0 = \mathbf{H}_0\mathbf{W}_V$, where $\mathbf{W}_V \in \mathbb{R}^{d\times d}$ is a learnable matrix that maps the input features to the initial velocity.

The Deep Architecture. Following principles from recurrent neural networks [90, 87, 52, 15], we enable hierarchical propagation of spatio-temporal information across the graph by constructing a deep architecture composed of a stack of multiple layers, each interleaved with an MLP to further enhance the effectiveness of the model:

$$\begin{aligned} \mathbf{H}^{(\ell)}_{t+1}, \mathbf{V}^{(\ell)}_{t+1} &= \text{STORM}(\mathbf{X}_t, \mathbf{H}^{(\ell)}_t, \mathbf{V}^{(\ell)}_t) \\ \mathbf{Y}^{(\ell)}_t &= \text{MLP}(\mathbf{H}^{(\ell)}_{t+1}), \end{aligned} \tag{4}$$

where $\mathbf{Y}^{(\ell)}_t \in \mathbb{R}^{n\times c'}$ is the output of the layer ℓ for timestep t. Therefore, in the multi-layer case, the input sequence of a layer corresponds to the output of the previous: $\mathbf{X}^{(\ell+1)}_t = \mathbf{Y}^{(\ell)}_t$.

In summary, the STORM can be interpreted as a sequence-to-sequence recurrent system where spatial and temporal information are processed simultaneously. At each layer ℓ and time step t, the model receives the current input features $\mathbf{X}^{(\ell)}_t$ (i.e., the original input $\mathbf{X}_t$ or the output of the previous layer $\mathbf{Y}^{(\ell-1)}_t$) together with the previous hidden state $\mathbf{H}^{(\ell)}_t$ (and velocity $\mathbf{V}^{(\ell)}_t$), which are processed through the discretized update equations in Equation (3) to compute the next hidden state and velocity. The hidden state at each step is then passed through an MLP to produce an intermediate representation, which can either be used for the final prediction or fed as input to subsequent layers in the stack. This process is visually summarized in Figure 1.

[4]We note that disconnected node pairs always have zero adaptive resistance; hence, the weight a_{uv} is defined only for existing edges.

3 Theoretical Analysis

In this section, we theoretically analyze the propagation capabilities of STORM, as well as its stability and expressiveness. By describing how features evolve through time and space, we see that the sensitivity never vanishes, allowing features to propagate endlessly. This, coupled with the dissipation component, enables the selection of relevant features and ensures the stability of the model as a dynamical system. Finally, we prove that our model is universal. All theoretical results focus on a single layer of STORM, with the layer index $\cdot^{(\ell)}$ omitted for notational ease. We report complexity analysis of STORM in Appendix E.1, and extensions to changing topologies in Appendix C.6.

3.1 Information Propagation and Dissipation

We first provide a bottom-up construction of STORM, starting from basic wave dynamics in graphs and then adding dissipation. By doing so, we explore and interpret how the input signal is incorporated and propagated by our model. For simplification purposes, we consider one-dimensional embeddings and call the node feature vector $\mathbf{H}(t) \in \mathbb{R}^n$. Additionally, we consider a fixed symmetric Laplacian $\hat{\mathbf{L}}$ (we discuss the non-symmetric case in Appendix C.1), and initial conditions given by $\mathbf{H}(0) = \mathbf{H}_0, \dot{\mathbf{H}}(0) = 0$. In both steps of our analysis, initial conditions do not play a fundamental role, as they do not change the fundamental features of propagation dynamics, since different values for $\mathbf{V}_0$ would only add a constant phase term in the trigonometric functions.

Step 1: Forced Harmonic Graph Oscillator. At its core, STORM exploits the nice properties of wave propagation in a graph via the equation $\ddot{\mathbf{H}}(t) = -\hat{\mathbf{L}}\mathbf{H}(t)$. This equation has an explicit solution: $\mathbf{H}(t) = \cos\left(\sqrt{\hat{\mathbf{L}}}t\right)\mathbf{H}_0$ (see Appendix C.1 for further details). This formula clearly indicates that, although it oscillates, the initial information is propagated endlessly in the system and never vanishes. In STORM, the input $\mathbf{X}(t)$ is introduced by a forcing term. The simplest implementation of the external force in Equation (2) is $F(\cdot) = \mathbf{X}(t)$, leading to $\ddot{\mathbf{H}}(t) = -\hat{\mathbf{L}}\mathbf{H}(t) + \mathbf{X}(t)$. By calling $\sqrt{\hat{\mathbf{L}}}^{\dagger}$ the pseudoinverse of $\sqrt{\hat{\mathbf{L}}}$, and given $\mathbf{X}(t)$ orthogonal to $\ker(\hat{\mathbf{L}})$ for each t (see Appendix C.1 for the general case, which only introduces a global bias in the graph), one can write the solution $\mathbf{H}(t)$ as

$$\cos\left(\sqrt{\hat{\mathbf{L}}}t\right)\mathbf{H}_0 + \sqrt{\hat{L}}^{\dagger}\int_0^t \sin\left(\sqrt{\hat{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)d\tau, \tag{5}$$

The derivation and the details about Equation (5) are provided in Appendix C.1. In general, we can write a similar relation between states $\mathbf{H}(t)$ and $\mathbf{H}(s)$ at different times $s < t$ by shifting the definition of the initial condition. Immediately, this leads to the sensitivity matrix being $\frac{\partial \mathbf{H}(t)}{\partial \mathbf{H}(s)} = \cos\left(\sqrt{\hat{\mathbf{L}}}(t-s)\right)$, which means that, once information is inserted into the graph, it stays within it. Given that the input drives the dynamics of STORM, for long-range tasks, it is important to analyze how changes of $\mathbf{X}(t)$ affect the long-term behavior of our model. Given the form of Equation (5), we can explicitly describe this influence. If we consider a new input function $\bar{\mathbf{X}}(t)$ that is equal to $\mathbf{X}(t)$ except for a small constant difference $\delta\mathbf{X}$ in the time interval $[t-h, t]$, the difference between new and old states $\delta\mathbf{H}(t)$ is

$$\delta\mathbf{H}(t) = -\hat{L}^{\dagger}\delta\mathbf{X}\left(1 - \cos\left(\sqrt{\hat{\mathbf{L}}}h\right)\right). \tag{6}$$

The derivation of Equation (6) is provided in Appendix C.1. Hence, changes in the input are propagated through time, and their effect is proportional to their magnitude.

Step 2: Adding Dissipation. Since a purely conservative behavior might not be beneficial in effectively solving real-world problems, we add a damping term to the dynamics defined in "Step 1". We consider the easiest implementation of dissipation, i.e., $D(\cdot) = 2k\dot{\mathbf{H}}(t)$ with $k \in \mathbb{R}^+$ being the damping coefficient, obtaining $\ddot{\mathbf{H}}(t) = -\hat{\mathbf{L}}\mathbf{H}(t) - 2k\dot{\mathbf{H}}(t) + \mathbf{X}(t)$. Under the initial conditions $\mathbf{H}(0) = \mathbf{H}_0, \dot{\mathbf{H}}(0) = -k\mathbf{H}_0$, we can derive an explicit solution and analyze it:

$$\mathbf{H}(t) = e^{-kt}\cos\left(\sqrt{\tilde{\mathbf{L}}}t\right)\mathbf{H}_0 + \sqrt{\tilde{\mathbf{L}}}^{\dagger}\int_0^t e^{-k(t-\tau)}\sin\left(\sqrt{\tilde{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)d\tau. \tag{7}$$

where $\tilde{\mathbf{L}} = \hat{\mathbf{L}} - k^2\mathbf{I}$. The long-term behavior of the system is thus determined by the eigenvalues of $\hat{\mathbf{L}}$ and the learned coefficient k for the dissipation. Specifically, the eigenvalues and eigenspaces

of the Laplacian $\hat{\mathbf{L}}$ are related to different topological features of the graph, as commonly exploited in positional embeddings [8, 27]. By choosing the magnitude of k, STORM adaptively filters out these features and is able to, for example, reduce global biases or control the input influence on future embeddings. We refer to Appendix C.1 for a deeper discussion.

3.2 Marginal Stability of STORM

The addition of dissipation plays a central role in shaping the long-term behavior of dynamical systems. Conservative dynamics preserve energy and support multiple persistent oscillatory trajectories, whereas dissipation can induce asymptotic stability toward a stable input-driven regime. Globally asymptotically stable systems guarantee robust long-term behavior but often suppress transient dynamics that are crucial for predictive performance. In contrast, unstable systems amplify perturbations uncontrollably, leading to loss of reliability and generalization issues. Marginally stable regimes strike a balance between these extremes, combining dynamical richness with controlled behavior. In this section, we show that STORM is designed to operate in such a regime.

As a first step, we analyze the asymptotic structure of the STORM layer dynamics by characterizing the steady states of the system. This result (proved in Appendix C.2) establishes the structure of the equilibria towards which the model may converge under sufficient dissipation.

Theorem 3.1 (Steady states)**.** *Consider a STORM layer with constant forcing term $\mathbf{F}$, Laplacian $\widehat{\mathbf{L}}$ of a connected graph, and invertible weight matrix $\mathbf{W}$. Then any steady state must have the form*

$$\mathbf{H}^* = \widehat{\mathbf{L}}^{\dagger}\mathbf{F}\mathbf{W}^{-1} + \mathbf{1}\mathbf{c}^{\top}, \tag{8}$$

where $\mathbf{1}\mathbf{c}^{\top}$ is a matrix whose columns lie in the kernel of the Laplacian $\widehat{\mathbf{L}}$ parametrised by $\mathbf{c} \in \mathbb{R}^d$, and $\mathbf{1} \in \mathbb{R}^n$ the vector of all ones. Moreover, if $\mathbf{1}^{\top}\mathbf{F} \neq \mathbf{0}$ then no steady state exists.

Asymptotic convergence alone does not determine how the system processes information away from equilibrium, especially in complex time-varying scenarios. In particular, the rate of convergence towards the steady state indicates whether transient dynamics are rapidly extinguished or instead persist long enough to effectively be exploited in downstream prediction. Moreover, excessively fast contraction toward equilibrium has been associated with the over-smoothing phenomenon [96], leading to poor performances. In contrast, a system operating in a regime of marginal stability has enough time to develop relevant features and separate trajectories effectively before dissipating them, without compromising long-term stability. The following theorem, which we prove in Appendix C.3, formalizes this property by showing that STORM operates within this regime.

Theorem 3.2 (Marginal stability)**.** *In a STORM layer with constant coefficients, small coupling, and small step size, each node can operate around a regime of marginal stability. Moreover, a necessary condition for stability of a STORM layer is that $0 \leq D(\mathbf{H}(t)) \leq \frac{2}{\epsilon}$.*

The non-negativity requirement on the dissipation term thus emerges as a necessary structural condition for the stability of STORM and motivates the parameterization adopted in the model, which guarantees it by construction. Finally, the hyperparameter ϵ must be chosen sufficiently small depending on the depth of a layer. In the full STORM formulation, dissipation, forcing, and graph coupling are time-varying and learned jointly, yielding adaptive dynamics that allow the model to move flexibly within, or near the boundary of, the stability regime in response to task demands.

3.3 Sensitivity Analysis through an Upper Bound

Information propagation in GNNs can be analyzed by measuring the sensitivity of learned node representations to initial features at previous steps. This perspective has been explored both in static GNN architectures [113, 25] and in the spatio-temporal setting [81], providing a principled framework for understanding effective propagation dynamics. Building on this, we now derive an explicit bound on the sensitivity of STORM to perturbations in distant node embeddings, both in space and time. This result quantifies how structural properties of the graph, together with learned dynamics of STORM and discretization parameters, jointly shape the spatio-temporal propagation of information.

Theorem 3.3 (Upper bound sensitivity)**.** *Let us consider two nodes u and v being at distance L from each other, and let us denote as $\mathbf{h}_t^u$ the embedding of node u at time $t \in [0, L]$. Assume $D(\mathbf{H}_t) = ReLU(\mathbf{H}_t\mathbf{W}_D)$, where $\mathbf{W}_D \in \mathbb{R}^{d\times d}$ is a matrix of weights, and $F(\mathbf{H}_t, \mathbf{X}_t) =$*

$\tanh\left(\boldsymbol{S}[\mathbf{H}_t, \mathbf{X}_t][\mathbf{W}_{Fh}, \mathbf{W}_{Fx}]^\top\right)$*, where* $\mathbf{W}_{Fh} \in \mathbb{R}^{d\times d}$*,* $\mathbf{W}_{Fx} \in \mathbb{R}^{c\times d}$ *are matrices of weights,* $[\cdot,\cdot]$ *denotes the horizontal concatenation of matrices,* $[\cdot,\cdot]^\top$ *denotes the vertical concatenation of matrices, and* $\mathbf{S}$ *denotes the graph shift operator that rules the forcing dynamics. Assuming* $\mathbf{A}$ *is the adjacency matrix and node embeddings to stay within a bounded norm* R*, then*

$$\left\| \frac{\partial \mathbf{h}_L^u}{\partial \mathbf{h}_0^v} \right\| \leq (\mathbf{A}^L)_{vu} \left(2 + \epsilon 3R \|\mathbf{W}_D\|_2 + \epsilon^2 \left(\|\mathbf{S}\|_{\max} \|\mathbf{W}_{Fh}\|_2 + \|\widehat{\mathbf{L}}\|_{\max} \|\mathbf{W}\|_2 \right) \right)^L. \tag{9}$$

We provide the proof in Appendix C.4 [5]. The bound in Theorem 3.3 makes explicit the mechanisms through which sensitivity decays, or potentially accumulates, over graph distance. In particular, the factor $(\mathbf{A}^L)_{vu}$ captures the combinatorial effect of graph connectivity, while the multiplicative term depending on the one-step Jacobian norm reflects the contribution of the local dynamical update. This decomposition mirrors bounds derived in prior analyses of oversquashing, where sensitivity is shown to decay exponentially with distance under contraction-dominated dynamics [25, 47]. In contrast, the explicit dependence on dissipation, forcing, and temporal variation in Equation (9) reveals how STORM can regulate sensitivity through learned dynamics together with architectural depth. As a result, STORM admits regimes in which long-range interactions are preserved without sacrificing stability.

3.4 Universality of STORM

The dynamics of STORM not only allow for effective long-range propagation of signals and inputs, but are also theoretically sound in terms of expressivity and universality of the model. Taking inspiration from the framework of Neural Oscillators [67], we consider the task of approximating a *causal* and *continuous* functional Φ that maps a node-discriminating [121] input signal $\mathbf{X}(t) \in C([0,T];\mathbb{R}^{n\times c})$to the corresponding output $\mathbf{Y}(t) \in C([0,T];\mathbb{R}^{n\times c'})$.

Theorem 3.4 (Universality)**.** *Let* $\Phi : C([0,T];\mathbb{R}^{n\times c}) \to C([0,T];\mathbb{R}^{n\times c'})$ *be a causal and continuous operator, and consider inputs* $\mathbf{X}(t)$ *that discriminate nodes (i.e., no non-trivial graph automorphisms fix the inputs). Let* Ψ_θ *be the operator that maps the input* $\mathbf{X}(t)$ *to the solution of Equation* (2) *with initial conditions given by* $\mathbf{H}_0 = \mathbf{X}(0)\mathbf{W}_H$ *and* $\mathbf{V}_0 = \mathbf{H}_0\mathbf{W}_V$*, i.e.,* Ψ_θ *represents STORM. Let* $K \subset C([0,T];\mathbb{R}^{n\times c})$ *be compact. Then, for any* $\epsilon > 0$*, there exist parameters* θ *s.t.*

$$\sup_{t\in[0,T]} |\Phi(\mathbf{X})(t) - \Psi_\theta(\mathbf{X})(t)| \leq \epsilon \quad \forall \mathbf{X} \in K. \tag{10}$$

We refer to Appendix C.5 for the proof. This theorem shows that STORM can tune in to any specific input sequence, process its information, and reliably predict the output with arbitrary prediction.

4 Experiments

In this section, we empirically validate the practical benefits of our method. In Section 4.1 we assess STORM on a newly introduced task called SURF-WAVE, which is specifically designed to evaluate long-range spatio-temporal propagation. To further analyze this capability, we also evaluate our method on the RocketMan benchmark introduced by [81]. Finally, in Section 4.3, we benchmark STORM on real-world traffic and weather forecasting tasks. We report runtimes and complexity analyses of STORM in Appendix E.1. We present experiments on time-varying topology in Appendix E.2. Additional ablation studies are reported in Appendix F, where we provide a more comprehensive understanding of STORM, covering the model's depth and width, dissipation, forcing, edge resistance, and step size. Across all experiments, we compare our method against state-of-the-art temporal GNNs, with details of the considered baselines provided in Appendix D.2. We report the hyperparameter spaces used in our experiments in Appendix D.3. We openly release the code at `https://github.com/thobotics/STORM`.

4.1 Surface Wave Forecasting Task

To assess the ability of STORM to capture long-range spatio-temporal interactions between distant nodes, we introduce a novel physics-based benchmark, called Surface Wave Forecasting (SURF-WAVE). We extend the WaveBall task proposed by [59] to a sequence-to-sequence spatio-temporal

[5] Theorem 3.3 can be easily adapted to other activations or different functions as discussed in Appendix C.4.

forecasting setting. Specifically, SURF-WAVE simulates the motion of three balls, initialized at different locations, on an L-shaped water surface, together with the surface waves they generate over time. More details about the task are reported in Appendix D.1.

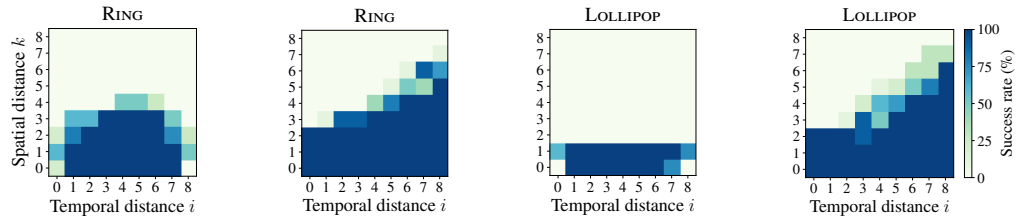
As reported in Table 1, STORM consistently outperforms all considered baselines (additional visualizations are in Appendix E.3). We note that, since each ball node is connected only to the corner nodes of the L-shaped grid (see Figure 3b), information about the ball trajectories must traverse large spatial distances before influencing the wave field. At the same time, the wave dynamics induce long-term temporal dependencies, requiring the model to retain and propagate information over extended horizons. Consequently, accurate forecasting hinges on effective long-range spatio-temporal propagation to maintain consistency between the ball motion and the resulting wave patterns. The strong performance of STORM therefore highlights the effectiveness of the proposed propagation scheme in capturing long-range spatio-temporal dependencies. The strong performance of STORM can therefore be attributed to its wave-based propagation mechanism, which naturally supports long-range information flow, together with its expressive power and theoretical universality, enabling it to faithfully model the coupled spatio-temporal dynamics of the system.

Table 1: SURF-WAVE task results. We report test MSE ($\times 10^3$) $\pm$ STD over 5 runs. **First**, **second**, and **third** results are color-coded. TTS variants follow the time-then-space scheme proposed in [41] also employed in [81].

Model	MSE $\downarrow$
AGCRN	6.93 $\pm$ 0.13
AR	10.86 $\pm$ 0.01
DCRNN	**4.95 $\pm$ 0.05**
EvolveGCN	12.42 $\pm$ 0.68
FC-RNN	12.35 $\pm$ 6.62
Graph WaveNet	5.14 $\pm$ 0.10
Graph WaveNet-TTS	5.07 $\pm$ 0.02
GRU-GCN	5.74 $\pm$ 0.16
MPTCN-TTS	6.10 $\pm$ 0.60
RNN	4.93 $\pm$ 0.06
Transformer	**4.85 $\pm$ 0.04**
STORM (ours)	**4.67 $\pm$ 0.03**

4.2 RocketMan Benchmark

We further investigate the effectiveness of the model in long-range spatio-temporal propagation using the RocketMan benchmark introduced by [81]. RocketMan is a memory task in which the model must retrieve the average value at a time step $t - i$ for nodes exactly k hops away from a target node, given a graph and a time window of random values. We use the same experimental setup as in [81]. Results are reported in Figure 2, with detailed numerical results provided in Appendix E.4. Across both graph topologies, MPTCN exhibits a sharp degradation in performance as either the spatial or temporal distance increases, failing to propagate information beyond short ranges. In contrast, STORM consistently maintains high accuracy across large spatio-temporal distances, as evidenced by the uniformly high success rates in the upper-right regions of the heatmaps. This behavior empirically confirms that the oscillatory dynamics of our architecture enable stable information propagation over long spatial and temporal horizons. Notably, the performance gap widens in the lollipop graph, where the presence of a topological bottleneck amplifies over-squashing effects, limiting effective signal propagation across distant nodes.



(a) MPTCN ($P = 2$) (b) STORM (ours) (c) MPTCN ($P = 2$) (d) STORM (ours)

Figure 2: Long-range spatio-temporal propagation effectiveness on the RocketMan benchmark, comparing STORM against MPTCN (with convolutional filter size $P = 2$). Larger dark blue areas indicate a perfect success rate across a wider variety of temporal and spatial distances in the task.

4.3 Real-World Forecasting Tasks

To further evaluate the performance of our STORM, we assess its effectiveness in capturing complex spatio-temporal information in real-world applications, i.e., traffic and weather forecasting. In both

applications, the goal is to predict future node values from time-series data. Specifically, we employ the popular benchmarks of Metr-LA and PeMS-Bay [70], which focus on traffic, and EngRAD [80] for weather forecasting. We adopt the experimental protocols introduced by [70] for traffic forecasting and by [22] for weather forecasting.

Traffic Forecasting. We report traffic forecasting results, considering a prediction horizon of 12 steps, in Table 2. STORM achieves the best performance across all evaluation metrics (MAE, RMSE, and MAPE) on both datasets, consistently outperforming strong temporal GNN baselines. Notably, the improvements are particularly pronounced in RMSE and MAPE, indicating more accurate modeling of both average trends and extreme variations.

Weather Forecasting. Results on the EngRAD forecasting task are reported in Table 3 (an extended comparison is available in Appendix E.5). Among all baselines, STORM achieves the second-best performance, outperforming both graph-agnostic and graph-aware spatio-temporal architectures.

Table 2: 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 [103, 73, 40, 34, 124].

Model	Metr-LA			PeMS-Bay		
	MAE ↓	RMSE ↓	MAPE ↓	MAE ↓	RMSE ↓	MAPE ↓
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%
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%
STORM (ours)	**3.05**	**6.30**	**8.63%**	**1.59**	**3.69**	**3.59%**

Compared to MPTCN, which is specifically designed to mitigate over-squashing in temporal graphs, STORM achieves competitive performance, underscoring the effectiveness of our approach also in the weather forecasting setting.

Table 3: EngRAD forecasting results. Additional comparisons are in Appendix E.5. We report test MAE ± STD; our result is averaged over 5 runs. "+MP" indicates the use of message-passing and "+Emb." the used of additional embeddings to improve model expressiveness. **First**, **second**, and **third** results are color-coded.

Model	**MAE ↓**
FC-RNN	55.37 ± 1.57
LocalRNN	58.80 ± 0.31
RNN	46.83 ± 0.19
GCRNN	43.93 ± 0.55
GCRNN (+Emb.)	**43.14 ± 0.19**
MPTCN	**40.38 ± 0.08**
RNN+MP	44.48 ± 0.24
RNN+MP (+Emb.)	43.70 ± 0.33
STORM (ours)	**41.74 ± 0.17**

5 Conclusions

We introduced STORM, a temporal GNN that enables long-range spatio-temporal propagation by evolving latent states with an oscillatory (wave-like) dynamics on graphs. Augmented with dissipative and external forcing terms, STORM balances conservative and non-conservative behaviors, allowing it to filter irrelevant signals while preserving and propagating important information over long horizons. Together with adaptive edge resistances, the model can further regulate message flow by favoring pathways that are more informative. To justify these design choices, we provided a bottom-up analysis of the individual components, showing how they affect long-range sensitivity and proving key theoretical properties such as stability and universality. Empirically, STORM consistently matches or outperforms strong baselines across diverse benchmarks, ranging from long-range spatio-temporal tasks to real-world forecasting, with an additionally newly introduced physics simulation-based benchmark to directly stress-test long-range spatio-temporal propagation.

Future work includes extending the STORM framework to irregularly sampled dynamic graphs, where observation times are non-uniformly spaced [94, 51], and continuous-time dynamic graph settings [62].

Acknowledgements

AG, AT, AC, and DB acknowledge funding from EU-EIC EMERGE (Grant No. 101070918). TH and GN acknowledge funding from the German Research Foundation (DFG) through Research Unit FOR 5339. The authors thank NHR@KIT for computing resources on HoreKa, partially funded by the DFG. NHR@KIT is jointly supported by the Federal Ministry of Research, Technology and Space and the Ministry of Science, Research and the Arts of Baden-Württemberg through the NHR program. The authors thank Ivan Marisca for kindly providing support on the ROCKETMAN task.

References

[1] Uri Alon and Eran Yahav. On the Bottleneck of Graph Neural Networks and its Practical Implications. In *International Conference on Learning Representations*, 2021.

[2] Alvaro Arroyo, Alessio Gravina, Benjamin Gutteridge, Federico Barbero, Claudio Gallicchio, Xiaowen Dong, Michael Bronstein, and Pierre Vandergheynst. On Vanishing Gradients, Over-Smoothing, and Over-Squashing in GNNs: Bridging Recurrent and Graph Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pages 74356–74393. Curran Associates, Inc., 12 2025.

[3] Davide Bacciu, Federico Errica, Alessio Gravina, Lorenzo Madeddu, Marco Podda, and Giovanni Stilo. Deep Graph Networks for Drug Repurposing With Multi-Protein Targets. *IEEE Transactions on Emerging Topics in Computing*, 12(1):177–189, 2024.

[4] 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.

[5] 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.

[6] Federico Barbero, Andrea Banino, Steven Kapturowski, Dharshan Kumaran, João Guilherme Madeira Araújo, Alex Vitvitskyi, Razvan Pascanu, and Petar Veličković. Transformers need glasses! information over-squashing in language tasks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[7] Federico Barbero, Ameya Velingker, Amin Saberi, Michael M. Bronstein, and Francesco Di Giovanni. Locality-aware graph rewiring in GNNs. In *The Twelfth International Conference on Learning Representations*, 2024.

[8] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[9] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[10] Mitchell Black, Zhengchao Wan, Amir Nayyeri, and Yusu Wang. Understanding oversquashing in gnns through the lens of effective resistance. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[11] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[12] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2014.

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

[14] Andrea Ceni, Andrea Cossu, Maximilian W Stölzle, Jingyue Liu, Cosimo Della Santina, Davide Bacciu, and Claudio Gallicchio. Random oscillators network for time series processing. In *International Conference on Artificial Intelligence and Statistics*, pages 4807–4815. PMLR, 2024.

[15] Andrea Ceni, Alessio Gravina, Claudio Gallicchio, Davide Bacciu, Carola-Bibiane Schonlieb, and Moshe Eliasof. Message-Passing State-Space Models: Improving Graph Learning with Modern Sequence Modeling. *arXiv preprint arXiv:2505.18728*, 2025.

[16] Bo Chang, Minmin Chen, Eldad Haber, and Ed H. Chi. AntisymmetricRNN: A dynamical system view on recurrent neural networks. In *International Conference on Learning Representations*, 2019.

[17] Jinyin Chen, Xueke Wang, and Xuanheng Xu. GC-LSTM: Graph convolution embedded LSTM for dynamic network link prediction. *Applied Intelligence*, pages 1–16, 2022.

[18] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural Ordinary Differential Equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[19] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics, October 2014.

[20] Andrea Cini, Ivan Marisca, Filippo Maria Bianchi, and Cesare Alippi. Scalable Spatiotemporal Graph Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(6):7218–7226, Jun. 2023.

[21] Andrea Cini, Ivan Marisca, Daniele Zambon, and Cesare Alippi. Taming local effects in graph-based spatiotemporal forecasting. In *Advances in Neural Information Processing Systems*, volume 36, pages 55375–55393. Curran Associates, Inc., 2023.

[22] Andrea Cini, Ivan Marisca, Daniele Zambon, and Cesare Alippi. Graph deep learning for time series forecasting. *ACM Computing Surveys*, 57(12):1–34, 2025.

[23] Fabrizio De Castelli, Alessio Gravina, Moshe Eliasof, Carola-Bibiane Schönlieb, and Davide Bacciu. Adaptive Memory Retention in Dynamic Graphs. In *Forty-third International Conference on Machine Learning*, 7 2026.

[24] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[25] Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Liò, and Michael Bronstein. On over-squashing in message passing neural networks: the impact of width, depth, and topology. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[26] J.R. Dormand. *Numerical Methods for Differential Equations: A Computational Approach*. Engineering Mathematics. Taylor & Francis, 1996.

[27] 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.

[28] Vijay Prakash Dwivedi, Ladislav Rampášek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long Range Graph Benchmark. In *Advances in Neural Information Processing Systems*, volume 35, pages 22326–22340. Curran Associates, Inc., 2022.

[29] Moshe Eliasof, Alessio Gravina, Andrea Ceni, Claudio Gallicchio, Davide Bacciu, and Carola-Bibiane Schönlieb. Graph Adaptive Autoregressive Moving Average Models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 15232–15265. PMLR, 7 2025.

[30] Moshe Eliasof, Eldad Haber, and Eran Treister. PDE-GCN: Novel Architectures for Graph Neural Networks Motivated by Partial Differential Equations. In *Advances in Neural Information Processing Systems*, volume 34, pages 3836–3849. Curran Associates, Inc., 2021.

[31] Federico Errica, Henrik Christiansen, Viktor Zaverkin, Takashi Maruyama, Mathias Niepert, and Francesco Alesiani. Adaptive message passing: A general framework to mitigate oversmoothing, oversquashing, and underreaching. *arXiv preprint arXiv:2312.16560*, 2024.

[32] 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.

[33] Lawrence C Evans. *Partial Differential Equations*. Graduate studies in mathematics. American Mathematical Society, Providence, RI, 2 edition, March 2010.

[34] 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.

[35] Joel H Ferziger and Milovan Peric. *Computational methods for fluid dynamics*. Springer, Berlin, Germany, 3 edition, November 2001.

[36] Ben Finkelshtein, Xingyue Huang, Michael M. Bronstein, and Ismail Ilkan Ceylan. Cooperative Graph Neural Networks. In *Forty-first International Conference on Machine Learning*, 2024.

[37] Joel Franklin. *Mathematical Methods for Oscillations and Waves*. Cambridge University Press, 2020.

[38] Joel Friedman and Jean-Pierre Tillich. Calculus on graphs. *ArXiv*, cs.DM/0408028, 2004.

[39] Joel Friedman and Jean-Pierre Tillich. Wave equations for graphs and the edge-based laplacian. *Pacific J. Math.*, 216(2):229–266, October 2004.

[40] 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.

[41] Jianfei Gao and Bruno Ribeiro. On the equivalence between temporal and static equivariant graph representations. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 7052–7076. PMLR, 17–23 Jul 2022.

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

[43] 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*, page 1263–1272. JMLR.org, 2017.

[44] Alessio Gravina. Shaping Graph Neural Networks with Dynamical Systems. *Intelligenza Artificiale*, 0(0):17248035261488802, 2026.

[45] 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.

[46] 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.

[47] Alessio Gravina, Moshe Eliasof, Claudio Gallicchio, Davide Bacciu, and Carola-Bibiane Schönlieb. On Oversquashing in Graph Neural Networks Through the Lens of Dynamical Systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(16):16906–16914, Apr. 2025.

[48] Alessio Gravina, Claudio Gallicchio, and Davide Bacciu. Non-dissipative Propagation by Randomized Anti-symmetric Deep Graph Networks. In *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 25–36, Cham, 2025. Springer Nature Switzerland.

[49] Alessio Gravina, Giulio Lovisotto, Claudio Gallicchio, Davide Bacciu, and Claas Grohnfeldt. Long Range Propagation on Continuous-Time Dynamic Graphs. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 16206–16225. PMLR, 21–27 Jul 2024.

[50] Alessio Gravina, Jennifer L. Wilson, Davide Bacciu, Kevin J. Grimes, and Corrado Priami. Controlling astrocyte-mediated synaptic pruning signals for schizophrenia drug repurposing with deep graph networks. *PLOS Computational Biology*, 18(5):1–19, 05 2022.

[51] Alessio Gravina, Daniele Zambon, Davide Bacciu, and Cesare Alippi. Temporal graph odes for irregularly-sampled time series. In Kate Larson, editor, *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 4025–4034. International Joint Conferences on Artificial Intelligence Organization, 8 2024.

[52] Albert Gu, Karan Goel, and Christopher Re. Efficiently Modeling Long Sequences with Structured State Spaces. In *International Conference on Learning Representations*, 2022.

[53] 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.

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

[55] E. Haber and L. Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34(1), 2017.

[56] 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, page 1025–1035. Curran Associates Inc., 2017.

[57] 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 *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pages 136166–136196. Curran Associates, Inc., 12 2025.

[58] 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.

[59] Tai Hoang, Alessandro Trenta, Alessio Gravina, Niklas Freymuth, Philipp Becker, Davide Bacciu, and Gerhard Neumann. Improving Long-Range Interactions in Graph Neural Simulators via Hamiltonian Dynamics. In *The Fourteenth International Conference on Learning Representations*, 4 2026.

[60] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 11 1997.

[61] 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.

[62] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation Learning for Dynamic Graphs: a Survey. *J. Mach. Learn. Res.*, 21(1), jan 2020.

[63] T. Anderson Keller and Max Welling. Neural wave machines: Learning spatiotemporally structured representations with locally coupled oscillatory recurrent neural networks. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 16168–16189. PMLR, 23–29 Jul 2023.

[64] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*, 2017.

[65] 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.

[66] L D Landau and E M Lifshitz. *Mechanics and electrodynamics*. Elsevier, October 2013.

[67] Samuel Lanthaler, T. Konstantin Rusch, and Siddhartha Mishra. Neural oscillators are universal. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 46786–46806. Curran Associates, Inc., 2023.

[68] William Leeney, Alessio Gravina, and Davide Bacciu. Non-Dissipative Graph Propagation for Non-Local Community Detection. In *2025 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 7 2025.

[69] 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*, KDD '19, page 1279–1289, New York, NY, USA, 2019. Association for Computing Machinery.

[70] 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.

[71] Huidong Liang, Haitz Sáez de Ocáriz Borde, Baskaran Sripathmanathan, Michael Bronstein, and Xiaowen Dong. Towards quantifying long-range interactions in graph machine learning: a large graph dataset and a measurement. *arXiv preprint arXiv:2503.09008*, 2025.

[72] Daniel Liberzon. *Switching in Systems and Control*. Birkhäuser Boston, 2003.

[73] 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*, pages 4125–4129, 2023.

[74] Xu Liu, Yutong Xia, Yuxuan Liang, Junfeng Hu, Yiwei Wang, Lei Bai, Chao Huang, Zhenguang Liu, Bryan Hooi, and Roger Zimmermann. LargeST: A benchmark dataset for large-scale traffic forecasting. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[75] Zewen Liu, Xiaoda Wang, Bohan Wang, Zijie Huang, Carl Yang, and Wei Jin. Graph odes and beyond: A comprehensive survey on integrating differential equations with graph neural networks. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, page 6118–6128, New York, NY, USA, 2025. Association for Computing Machinery.

[76] Antonio Longa, Veronica Lachi, Gabriele Santin, Monica Bianchini, Bruno Lepri, Pietro Lio, Franco Scarselli, and Andrea Passerini. Graph neural networks for temporal graphs: State of the art, open challenges, and opportunities, 2023.

[77] 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.

[78] 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*, pages 23321–23337. PMLR, 23–29 Jul 2023.

[79] Miles Macklin. Warp: A high-performance python framework for gpu simulation and graphics. `https://github.com/nvidia/warp`, March 2022. NVIDIA GPU Technology Conference (GTC).

[80] Ivan Marisca, Cesare Alippi, and Filippo Maria Bianchi. Graph-based forecasting with missing data through spatiotemporal downsampling. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[81] Ivan Marisca, Jacob Bamberger, Cesare Alippi, and Michael M. Bronstein. Over-squashing in spatiotemporal graph neural networks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[82] Alessio Micheli. Neural Network for Graphs: A Contextual Constructive Approach. *IEEE Transactions on Neural Networks*, 20(3):498–511, 2009.

[83] Alessio Micheli and Domenico Tortorella. Discrete-time dynamic graph echo state networks. *Neurocomputing*, 496(C):85–95, jul 2022.

[84] Luca Miglior, Matteo Tolloso, Alessio Gravina, and Davide Bacciu. Can You Hear Me Now? A Benchmark for Long-Range Graph Propagation. In *The Fourteenth International Conference on Learning Representations*, 4 2026.

[85] Yaaqov Mishayev, Yonatan Sverdlov, Tal Amir, and Nadav Dym. Short-range oversquashing. In *The Fourth Learning on Graphs Conference*, 2025.

[86] Kenta Oono and Taiji Suzuki. Graph Neural Networks Exponentially Lose Expressive Power for Node Classification. In *International Conference on Learning Representations*, 2020.

[87] Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *International Conference on Machine Learning*, pages 26670–26698. PMLR, 2023.

[88] George Panagopoulos, Giannis Nikolentzos, and Michalis Vazirgiannis. Transfer Graph Neural Networks for Pandemic Forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(6):4838–4845, May 2021.

[89] 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 Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[90] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks. In *The second International Conference on Learning Representations*, 2014.

[91] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1310–1318, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[92] 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.

[93] Benedek Rozemberczki, Paul Scherer, Yixuan He, George Panagopoulos, Alexander Riedel, Maria Astefanoaei, Oliver Kiss, Ferenc Beres, Guzmán López, 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 & Knowledge Management*, CIKM '21, page 4564–4573. Association for Computing Machinery, 2021.

[94] Yulia Rubanova, Ricky T. Q. Chen, and David K Duvenaud. Latent Ordinary Differential Equations for Irregularly-Sampled Time Series. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[95] T. Konstantin Rusch, Michael M. Bronstein, and Siddhartha Mishra. A Survey on Oversmoothing in Graph Neural Networks. *arXiv preprint arXiv:2303.10993*, 2023.

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

[97] T. Konstantin Rusch and Siddhartha Mishra. Coupled oscillatory recurrent neural network (cornn): An accurate and (gradient) stable architecture for learning long time dependencies. In *International Conference on Learning Representations*, 2021.

[98] T. Konstantin Rusch and Siddhartha Mishra. Unicornn: A recurrent model for learning very long time dependencies. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9168–9178. PMLR, 18–24 Jul 2021.

[99] T. Konstantin Rusch and Daniela Rus. Oscillatory state-space models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[100] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[101] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *25th International Conference on Neural Information Processing*, ICONIP, pages 362–373. Springer, 2018.

[102] Chao Shang, Jie Chen, and Jinbo Bi. Discrete graph structure learning for forecasting multiple time series. In *International Conference on Learning Representations*, 2021.

[103] 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*, pages 1567–1577. ACM, 2022.

[104] Dai Shi, Andi Han, Lequan Lin, Yi Guo, and Junbin Gao. Exposition on over-squashing problem on GNNs: Current Methods, Benchmarks and Challenges, 2023.

[105] 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 *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 1548–1554. International Joint Conferences on Artificial Intelligence Organization, 8 2021.

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

[107] Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, Aug 2004.

[108] Shriyank Somvanshi, Md Monzurul Islam, Mahmuda Sultana Mimi, Sazzad Bin Bashar Polock, Gaurab Chhetri, and Subasish Das. From s4 to mamba: A comprehensive survey on structured state space models. *arXiv preprint arXiv:2503.18970*, 2025.

[109] 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.

[110] Joshua Southern, Francesco Di Giovanni, Michael M. Bronstein, and Johannes F. Lutzeyer. Understanding virtual nodes: Oversquashing and node heterogeneity. In *The Thirteenth International Conference on Learning Representations*, 2025.

[111] 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*, page 3104–3112, Cambridge, MA, USA, 2014. MIT Press.

[112] 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, page 57–64. Association for Computing Machinery, 2020.

[113] 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.

[114] Alessandro Trenta, Alessio Gravina, and Davide Bacciu. SONAR: Long-Range Graph Propagation Through Information Waves. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pages 145054–145087. Curran Associates, Inc., 12 2025.

[115] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[116] Eugene Vorontsov, Chiheb Trabelsi, Samuel Kadoury, and Chris Pal. On orthogonality and learning recurrent networks with long term dependencies. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3570–3578. PMLR, 06–11 Aug 2017.

[117] Peihao Wang, Ruisi Cai, Yuehao Wang, Jiajun Zhu, Pragya Srivastava, Zhangyang Wang, and Pan Li. Understanding and mitigating bottlenecks of state space models through the lens of recency and over-smoothing. In *The Thirteenth International Conference on Learning Representations*, 2025.

[118] Danni Wu, Yuanyuan Xu, Xuemin Lin, Wenjie Zhang, and Ying Zhang. Understanding evolving graph structures for large discrete-time dynamic graph representation. *Proc. VLDB Endow.*, 19(5):862–875, May 2026.

[119] 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, page 753–763, New York, NY, USA, 2020. Association for Computing Machinery.

[120] 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, page 1907–1913. AAAI Press, 2019.

[121] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*, 2019.

[122] Yuanyuan Xu, Wenjie Zhang, Xuemin Lin, and Ying Zhang. Unidyg: A unified and effective representation learning approach for large dynamic graphs. *IEEE Transactions on Knowledge and Data Engineering*, 37(7):4373–4388, 2025.

[123] 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.

[124] 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.

[125] 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.

[126] 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.

A Related Work

Long-Range Propagation. Over the past years, effectively modeling long-range dependencies has been a central challenge in deep learning for graphs and time series [98, 87, 108, 104, 2].

In static graphs, this difficulty primarily arises from phenomena such as over-smoothing [13, 86, 95] and over-squashing [1, 113, 25, 85], both of which are closely linked to vanishing gradients [2]. As a consequence, static GNNs often manifested a performance degradation in tasks requiring more global context [28, 84, 71]. To mitigate these problems, a wide range of approaches has been proposed. Graph rewiring methods [113, 42, 61, 54, 10, 7] explicitly modify the graph topology to facilitate long-range communication, while Graph Transformers [65, 92, 105, 106, 123, 78] enable direct interactions between arbitrary node pairs via attention mechanisms. Alternative strategies include weight space regularization [46, 48, 68, 47], leveraging port-Hamiltonian dynamics [58], filtering messages during information flow [31, 36], adopting adaptive graph filters based on learnable ARMA models [29], revisiting spectral GNNs with non-dissipative dynamics [57], introducing virtual nodes to ease long-range communication [110], or exploiting oscillatory dynamics [96, 114]. Notably, these efforts largely overlook the role of time-evolving graphs. Furthermore, the temporal graph setting intensifies the long-range propagation problem because the temporal dimension introduces an additional axis of information flow [81].

Similarly, in the RNN domain, modeling long-range dependencies is hindered by vanishing and exploding gradients [9, 91]. Prior work has addressed these issues through weight space regularization [116, 16], stable [55] or oscillatory [97, 98, 63, 99] dynamics, and state space models (SSMs) [52, 108]. More recently, the phenomenon of over-squashing has been linked to Transformers architectures for language models [6], while over-smoothing has been observed in deep SSM architectures [117].

Our STORM is inspired by oscillatory GNNs and RNNs [96, 114, 97, 98] and adopts a mathematically grounded formulation of graph calculus [38, 39] to model information propagation in temporal graphs as an oscillatory dynamics that offers stronger theoretical guarantees for long-range spatio-temporal propagation.

We stress that, while STORM is inspired by oscillatory GNNs for static graphs [96, 114], it differs fundamentally from these approaches in both modeling scope and capabilities, despite sharing a similar oscillatory propagation mechanism (i.e., $\ddot{\mathbf{H}} = -\mathbf{L}\mathbf{H}(t)\mathbf{W}$). In oscillatory GNNs designed for static graphs, time arises only implicitly through the discretization of the dynamics (i.e., as the number of propagation steps) and the evolution of the node features, while input signals are injected once as the initial condition. In contrast, STORM is explicitly designed for temporal graphs, naturally supporting time-varying topologies, where time is an explicit dimension of data and input signals are continuously incorporated as node states evolve. As a result, STORM captures the joint evolution of node representations across both space and time through a simple iterative process that adapts to incoming signals (a challenge that is inherently absent in the static graph setting). This highlights that STORM constitutes a non-trivial extension of oscillatory dynamics to temporal graphs. Importantly, this distinction also implies that empirical comparisons with oscillatory GNNs for static graphs are not directly meaningful: those models are not formulated for temporal graph settings, and any adaptation would fundamentally alter their original dynamics and assumptions. As such, rather than

serving as competitive baselines, they should be understood as a related but orthogonal line of work, from which STORM departs to address temporal dynamics in a principled manner.

Learning Spatio-Temporal Interactions on Graphs. Temporal GNNs often combine GNNs with RNNs to model spatio-temporal dynamics [62, 76, 45]. Some adopt stacked architectures that separate spatial and temporal processing [101, 89, 88, 4, 20, 81], while others integrate GNNs within RNNs for joint modeling [70, 69, 17, 21]. [41] classify these two types of architectures into Time-then-Space (TTS) and Time-and-Space (T&S). Recent approaches have also explored Fourier graph attention to model temporal structures at multiple time granularities in both continuous- and discrete-time dynamic graphs [122], and efficient neighbor management strategies for discrete-time dynamic graphs [118]. Despite advances in temporal graph learning, most existing approaches are not explicitly designed to support long-range spatio-temporal propagation. In this direction, [81] propose a TTS architecture to mitigate over-squashing in temporal graphs, while [49] introduce a framework tailored to long-range tasks in the different setting of continuous-time dynamic graphs, and [23] proposes an impulsive neural ode for learning dynamic graphs that is capable of trading-off between memory retention and noise filtering via weight space regularization. Differently, our STORM exploits a T&S architecture with oscillatory dynamics, enabling effective long-range propagation in temporal graphs.

B External Force, Dissipation, and Adaptive Resistance

The temporal evolution of node states defined in Equation (2) is governed by the oscillatory dynamics of the graph wave equation, together with a dissipative term and an external forcing term. The latter two components are introduced to more accurately model the dynamics of real-world systems. Specifically, in our experiments, we implement the dissipative mechanism as a non-linear transformation of the hidden state, i.e.,

$$D(\mathbf{H}_t) = \mathrm{ReLU}(\mathbf{H}_t\mathbf{W}), \tag{11}$$

which introduces state-dependent damping and allows the model to adaptively control the degree of dissipation during propagation.

We investigate different parameterizations of the external forcing term $F(\mathbf{X}_t, \mathbf{H}_t)$ to inject input-dependent information into the dynamics. In all configurations, the forcing function takes as input the concatenation of the current hidden state $\mathbf{H}_t$ and the node features $\mathbf{X}_t$. We consider the following variants:

1. **GCN + MLP forcing:** $F(\cdot) = \mathrm{GCN}(\cdot) + \mathrm{MLP}(\cdot)$, combining a graph-aware transformation with a node-wise nonlinear correction;
2. **GCN forcing:** $F(\cdot) = \mathrm{GCN}(\cdot)$, modeling the forcing purely through neighborhood aggregation;
3. **MLP forcing:** $F(\cdot) = \mathrm{MLP}(\cdot)$, capturing forcing effects independently across nodes.

We defined the MLP as $\mathrm{MLP}(\mathbf{x}) = \mathbf{W}\,\mathrm{ReLU}(\mathbf{x}) + \mathbf{b}$.

To model oscillatory propagation along edges as occurring through heterogeneous physical media, we adaptively learn the weights of the graph Laplacian defined in Equation (1) for existing edges. From a physical perspective, these learned weights can be interpreted as controlling the local propagation properties of the medium. In analogy with the classical wave equation, where the wave velocity determines how signals travel through space, the coefficients a_{uv} regulate the propagation dynamics along each edge, allowing each edge to differentially regulate the transmission of information, potentially inducing phase shifts and delays in propagation. Specifically, for a given edge (u, v), we compute an edge-specific resistance coefficient via a two-layer MLP:

$$a_{uv} = \mathrm{MLP}\Big([\mathbf{h}^u, \mathbf{h}^v]\Big) \tag{12}$$

where $[\mathbf{h}^u, \mathbf{h}^v]$ denotes the concatenation of the hidden states of nodes u and v. We discuss possible implications of the non-symmetric implementation of the Laplacian at the end of Section C.1. To investigate the effect of non-negative edge resistances, we also consider a form in which the output of the MLP is passed through an absolute value operation. In our experiments, we consider two variants of edge resistance modeling: (i) the *fixed* variant, where edge resistances are computed once at the beginning of each layer and then kept constant throughout the propagation steps; and (ii) the *adaptive* variant, where edge resistances are recomputed at every time step.

C Proofs of the Theoretical Statements

C.1 Information Propagation and Dissipation

In this Section, we prove and discuss the explicit solutions to the forced and damped graph wave equations in the forms used in Section 3.1. First, we provide some useful definitions and properties of the Laplacian matrix with fixed and symmetric resistances:

- The matrix $\hat{\mathbf{L}}$ is symmetric and positive semidefinite, i.e., , its eigenvalues are $\lambda_i \geq 0$. This is a direct consequence of it being symmetric and being diagonally dominant ($\hat{\mathbf{L}}_{ii} \geq \sum_{j \neq i} \hat{\mathbf{L}}_{ij}$).
- The smallest eigenvalue is $\lambda_0 = 0$. This is because the sums of the rows in the Laplacian matrix are zero, hence $\hat{\mathbf{L}}\mathbf{1} = 0$. The eigenspace of 0 represents a bias term in the whole graph, that is, a constant value added to all nodes.
- Each eigenvalue and eigenvector represents topological features in the graph, as exploited by Laplacian embeddings [8, 27].
- $\hat{\mathbf{L}}$ has a positive semidefinite square root $\sqrt{\hat{\mathbf{L}}}$ with real entries, which admits a pseudoinverse $\sqrt{\hat{\mathbf{L}}}^{\dagger}$, also with real entries.
- It holds that $\mathbf{h}^{\top}\hat{\mathbf{L}}\mathbf{h} = \sum_{(u,v)\in\mathcal{E}} a_{uv}(\mathbf{h}_u - \mathbf{h}_v)^2$ (see [114] for the proof).

In our theory, we often express the oscillations in the graph in terms of the function $\cos\left(\sqrt{\hat{\mathbf{L}}}t\right)$. This function is defined with the Taylor series of the cosine function, as

$$\cos\left(\mathbf{A}t\right) = \mathbf{I} - \frac{1}{2}\mathbf{A}^2t^2 + \frac{1}{4!}\mathbf{A}^4t^4 + \ldots = \sum_{i=0}^{\infty} \frac{(-1)^i}{2i!}\mathbf{A}^{2i}t^{2i} \tag{13}$$

A similar definition applies to the sine function.

We now provide the derivation of the explicit solutions from Section 3.1. We assume that $\mathbf{X}(t)$ is orthogonal to the kernel of the Laplacian $\hat{\mathbf{L}}$ for every t, and we discuss this case at the end of the proof. In particular, the component of the input $\mathbf{X}(t)$ relative to the kernel adds or removes global biases in the graph, adapting to trends in the input.

Lemma C.1. *Assume $\mathbf{X}(t)$ is orthogonal to the kernel of the Laplacian $\hat{\mathbf{L}}$ for every t. Let $\mathbf{H}(t) \in \mathbb{R}^n$ be the solution to the forced wave equation on a Graph with one-dimensional features, that is*

$$\ddot{\mathbf{H}}(t) = -\hat{\mathbf{L}}\mathbf{H}(t) + \mathbf{X}(t), \tag{14}$$

with initial condition $\mathbf{H}(0) = \mathbf{H}_0$ and $\dot{\mathbf{H}}(0) = 0$. Then, $\mathbf{H}(t)$ can be expressed as

$$\mathbf{H}(t) = \cos\left(\sqrt{\hat{\mathbf{L}}}t\right)\mathbf{H}_0 + \sqrt{\hat{L}}^{\dagger}\int_0^t \sin\left(\sqrt{\hat{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)d\tau, \tag{15}$$

where $\sqrt{\hat{L}}^{\dagger}$ is the pseudoinverse of $\sqrt{\hat{L}}$

Remark C.2. The initial condition $\dot{\mathbf{H}}(0) = 0$ is not necessary, as it is sufficient to add a phase term inside the cosine function to account for the different initial velocity.

Proof. It is sufficient to perform the explicit calculations. In particular:

$$\begin{aligned}
\dot{\mathbf{H}}(t) = &- \sqrt{\hat{\mathbf{L}}}\sin\left(\sqrt{\hat{\mathbf{L}}}t\right)\mathbf{H}_0 + \\
&+ \sqrt{\hat{\mathbf{L}}}^{\dagger}\sqrt{\hat{\mathbf{L}}}\int_0^t \cos\left(\sqrt{\hat{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)d\tau + \\
&+ \left[\sin\left(\sqrt{\hat{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)\right]_{\tau=t} \\
= &- \sqrt{\hat{\mathbf{L}}}\sin\left(\sqrt{\hat{\mathbf{L}}}t\right)\mathbf{H}_0 + \int_0^t \cos\left(\sqrt{\hat{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)d\tau
\end{aligned} \tag{16}$$

and

$$\begin{aligned}\ddot{\mathbf{H}}(t) =& -\hat{\mathbf{L}}\cos\left(\sqrt{\hat{\mathbf{L}}}t\right)\mathbf{H}_0 + \\ & -\sqrt{\hat{\mathbf{L}}}\int_0^t \sin\left(\sqrt{\hat{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)d\tau + \\ & +\left[\cos\left(\sqrt{\hat{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)\right]_{\tau=t} \\ =& -\hat{\mathbf{L}}\cos\left(\sqrt{\hat{\mathbf{L}}}t\right)\mathbf{H}_0 - \sqrt{\hat{\mathbf{L}}}\int_0^t \sin\left(\sqrt{\hat{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)d\tau + \mathbf{X}(t) \\ =& -\hat{\mathbf{L}}\mathbf{H}(t) + \mathbf{X}(t)\end{aligned} \tag{17}$$

□

To see how a difference in the inputs affects future states, we consider a new input function $\bar{\mathbf{X}}(t)$, which is the same as $\mathbf{X}(t)$ apart from a constant term $\delta\mathbf{X}$ in the time interval $[t-h, t]$. By developing the calculations from the explicit solution, the difference in the new and old state $\delta\mathbf{H}(t)$ is exactly

$$\begin{aligned}&\sqrt{\hat{L}}^{\dagger}\int_{t-h}^{t}\sin\left(\sqrt{\hat{\mathbf{L}}}(t-\tau)\right)\delta\mathbf{X}d\tau \\ =& -\hat{L}^{\dagger}\delta\mathbf{X}\left(1-\cos\left(\sqrt{\hat{\mathbf{L}}}h\right)\right).\end{aligned} \tag{18}$$

The kernel component. In the above discussion, we have assumed that the input $\mathbf{X}(t)$ is orthogonal to the Laplacian kernel $\ker(\hat{\mathbf{L}})$. The introduction of this term adds a term to the final solution in Lemma C.1 given by:

$$\int_0^t (t-\tau)P_{\ker(\hat{\mathbf{L}})}\mathbf{X}(\tau)\mathrm{d}\tau, \tag{19}$$

where $P_{\ker(\hat{\mathbf{L}})}$ is the orthogonal projection onto the kernel. To understand how this term influences the node features $\mathbf{H}(t)$, we recall how the Laplacian kernel is constituted. If the graph is connected, its eigenspace is generated by the constant vector on the graph $(1, \ldots, 1)$. This mode corresponds to a uniform offset shared by all nodes, on which the Laplacian exerts no restoring force. Consequently, the external input drives the evolution of this global component directly, allowing the model to adapt the overall graph-wide bias independently of the oscillatory dynamics associated with the positive eigenvalues.

Theorem C.3. *Let $\mathbf{H}(t) \in \mathbb{R}^n$ be the solution to the forced and dampened wave equation on a Graph with one-dimensional features, that is $\ddot{\mathbf{H}}(t) = -\hat{\mathbf{L}}\mathbf{H}(t) - 2k\dot{\mathbf{H}}(t) + \mathbf{X}(t)$, with initial condition $\mathbf{H}(0) = \mathbf{H}_0$ and $\dot{\mathbf{H}}(0) = -k\mathbf{H}_0$. Then, calling $\tilde{\mathbf{L}} = \hat{\mathbf{L}} - k^2\mathbf{I}$, the solution can be expressed as*

$$\mathbf{H}(t) = e^{-kt}\cos\left(\sqrt{\tilde{\mathbf{L}}}t\right)\mathbf{H}_0 + \sqrt{\tilde{\mathbf{L}}}^{\dagger}\int_0^t e^{-k(t-\tau)}\sin\left(\sqrt{\tilde{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)d\tau. \tag{20}$$

Proof. Again, it is a matter of calculating the solution's derivatives explicitly:

$$\begin{aligned}
\dot{\mathbf{H}}(t) =& - ke^{-kt}\cos\left(\sqrt{\tilde{\mathbf{L}}}t\right)\mathbf{H}_0 + \\
& - \sqrt{\tilde{\mathbf{L}}}e^{-kt}\sin\left(\sqrt{\tilde{\mathbf{L}}}t\right)\mathbf{H}_0 + \\
& + \left[\sqrt{\tilde{\mathbf{L}}}^{\dagger}\sqrt{\tilde{\mathbf{L}}}e^{-k(t-\tau)}\sin\left(\sqrt{\tilde{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)\right]_{\tau=t} + \\
& - k\sqrt{\tilde{\mathbf{L}}}^{\dagger}\int_0^t e^{-k(t-\tau)}\sin\left(\sqrt{\tilde{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)d\tau + \\
& + \sqrt{\tilde{\mathbf{L}}}^{\dagger}\sqrt{\tilde{\mathbf{L}}}\int_0^t e^{-k(t-\tau)}\cos\left(\sqrt{\tilde{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)d\tau \\
=& - ke^{-kt}\cos\left(\sqrt{\tilde{\mathbf{L}}}t\right)\mathbf{H}_0 + \\
& - \sqrt{\tilde{\mathbf{L}}}e^{-kt}\sin\left(\sqrt{\tilde{\mathbf{L}}}t\right)\mathbf{H}_0 + \\
& - k\sqrt{\tilde{\mathbf{L}}}^{\dagger}\int_0^t e^{-k(t-\tau)}\sin\left(\sqrt{\tilde{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)d\tau + \\
& + \int_0^t e^{-k(t-\tau)}\cos\left(\sqrt{\tilde{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)d\tau
\end{aligned} \tag{21}$$

and

$$\begin{aligned}
\ddot{\mathbf{H}}(t) =& \, k^2e^{-kt}\cos\left(\sqrt{\tilde{\mathbf{L}}}\right)\mathbf{H}_0 + \\
& + k\sqrt{\tilde{\mathbf{L}}}e^{-kt}\sin\left(\sqrt{\tilde{\mathbf{L}}}t\right)\mathbf{H}_0 + \\
& + k\sqrt{\tilde{\mathbf{L}}}e^{-kt}\sin\left(\sqrt{\tilde{\mathbf{L}}}t\right)\mathbf{H}_0 + \\
& - \tilde{\mathbf{L}}e^{-kt}\cos\left(\sqrt{\tilde{\mathbf{L}}}t\right)\mathbf{H}_0 + \\
& \left[-k\sqrt{\tilde{\mathbf{L}}}^{\dagger}e^{-k(t-\tau)}\sin\left(\sqrt{\tilde{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)d\tau\right]_{\tau=t} + \\
& + k^2\sqrt{\tilde{\mathbf{L}}}^{\dagger}\int_0^t e^{-k(t-\tau)}\sin\left(\sqrt{\tilde{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)d\tau + \\
& - k\sqrt{\tilde{\mathbf{L}}}^{\dagger}\sqrt{\tilde{\mathbf{L}}}\int_0^t e^{-k(t-\tau)}\cos\left(\sqrt{\tilde{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)d\tau + \\
& + \left[\sqrt{\tilde{\mathbf{L}}}^{\dagger}\sqrt{\tilde{\mathbf{L}}}e^{-k(t-\tau)}\cos\left(\sqrt{\tilde{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)d\tau\right]_{\tau=t} + \\
& - k\sqrt{\tilde{\mathbf{L}}}^{\dagger}\sqrt{\tilde{\mathbf{L}}}\int_0^t e^{-k(t-\tau)}\cos\left(\sqrt{\tilde{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)d\tau + \\
& - \sqrt{\tilde{\mathbf{L}}}^{\dagger}\sqrt{\tilde{\mathbf{L}}}\sqrt{\tilde{\mathbf{L}}}\int_0^t e^{-k(t-\tau)}\sin\left(\sqrt{\tilde{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)d\tau
\end{aligned} \tag{22}$$

By rearranging the terms, this becomes

$$\begin{aligned}
\ddot{\mathbf{H}}(t) =& - (\tilde{\mathbf{L}} - k^2\mathbf{I})e^{-kt}\cos\left(\sqrt{\tilde{\mathbf{L}}}\right)\mathbf{H}_0 + \\
& + 2k\sqrt{\tilde{\mathbf{L}}}e^{-kt}\sin\left(\sqrt{\tilde{\mathbf{L}}}t\right)\mathbf{H}_0 + \\
& - (\tilde{\mathbf{L}} - k^2\mathbf{I})\sqrt{\tilde{\mathbf{L}}}^{\dagger}\int_0^t e^{-k(t-\tau)}\sin\left(\sqrt{\tilde{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)d\tau + \\
& - 2k\int_0^t e^{-k(t-\tau)}\cos\left(\sqrt{\tilde{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)d\tau + \\
& + \mathbf{X}(t)
\end{aligned} \tag{23}$$

Now, by computing the quantity $-\hat{\mathbf{L}}\mathbf{H}(t) - 2k\dot{\mathbf{H}}(t) + \mathbf{X}(t)$, one can see that

$$
\begin{aligned}
-\hat{\mathbf{L}}\mathbf{H}(t) - 2k\dot{\mathbf{H}}(t) + \mathbf{X}(t) = \\
&- (\tilde{\mathbf{L}} + k^2\mathbf{I})e^{-kt}\cos\left(\sqrt{\tilde{\mathbf{L}}}t\right)\mathbf{H}_0 + \\
&- (\tilde{\mathbf{L}} + k^2\mathbf{I})\sqrt{\tilde{\mathbf{L}}}^{\dagger}\int_0^t e^{-k(t-\tau)}\sin\left(\sqrt{\hat{\tilde{\mathbf{L}}}}(t-\tau)\right)\mathbf{X}(\tau)d\tau + \\
&+ 2k^2 e^{-kt}\cos\left(\sqrt{\tilde{\mathbf{L}}}t\right)\mathbf{H}_0 + \\
&+ 2k\sqrt{\tilde{\mathbf{L}}}e^{-kt}\sin\left(\sqrt{\tilde{\mathbf{L}}}t\right)\mathbf{H}_0 + \\
&+ 2k^2\sqrt{\tilde{\mathbf{L}}}^{\dagger}\int_0^t e^{-k(t-\tau)}\sin\left(\sqrt{\tilde{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)d\tau + \\
&- 2k\int_0^t e^{-k(t-\tau)}\cos\left(\sqrt{\tilde{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)d\tau + \mathbf{X}(t) \\
= &- (\tilde{\mathbf{L}} - k^2\mathbf{I})e^{-kt}\cos\left(\sqrt{\tilde{\mathbf{L}}}t\right)\mathbf{H}_0 + \\
&- (\tilde{\mathbf{L}} - k^2\mathbf{I})\sqrt{\tilde{\mathbf{L}}}^{\dagger}\int_0^t e^{-k(t-\tau)}\sin\left(\sqrt{\hat{\tilde{\mathbf{L}}}}(t-\tau)\right)\mathbf{X}(\tau)d\tau \\
&+ 2k\sqrt{\tilde{\mathbf{L}}}e^{-kt}\sin\left(\sqrt{\tilde{\mathbf{L}}}t\right)\mathbf{H}_0 + \\
&- 2k\int_0^t e^{-k(t-\tau)}\cos\left(\sqrt{\tilde{\mathbf{L}}}(t-\tau)\right)\mathbf{X}(\tau)d\tau + \mathbf{X}(t) = \ddot{\mathbf{H}}(t)
\end{aligned}
\tag{24}
$$

□

Interpretation. Depending on the eigenvalues of $\hat{\mathbf{L}} - k^2\mathbf{I}$, the corresponding eigenvectors undergo a different behavior. To have a complete understanding of what happens to the model, we start again from the equation

$$\ddot{\mathbf{H}}(t) = -\hat{\mathbf{L}}\mathbf{H}(t) - 2k\dot{\mathbf{H}}(t) + \mathbf{X}(t) \tag{25}$$

Since the solution has the form $e^{\Omega t}\mathbf{h}_0$, by plugging this in we obtain

$$(\Omega^2 + 2k\Omega + \hat{\mathbf{L}})e^{\Omega t}\mathbf{h}_0 = 0 \tag{26}$$

We already know what the form of Ω is from the last theorem, but it is important to notice already that we want Ω and $\hat{\mathbf{L}}$ to have the same eigenspaces and eigenvectors. By choosing as base the eigenvectors $\mathbf{e}_i$ of $\hat{\mathbf{L}}$ with eigenvalues ω_i^2 (we use the square since they are non-negative), we convert the above equation into one for the eigenvalues of Ω, which we call $\bar{\omega}_i$:

$$\bar{\omega}_i^2 + 2k\bar{\omega}_i + \omega_i^2 = 0 \tag{27}$$

which has solutions

$$\bar{\omega}_i = -k \pm \sqrt{k^2 - \omega_i^2} \tag{28}$$

This eigendecomposition is fundamental to understanding the evolution of the system. In particular, a solution can be described as a linear combination of n terms, called *normal modes*:

$$\mathbf{H}(t) = \sum_{i=1}^{n} r_i(t)\mathbf{e}_i, \tag{29}$$

and each of them is related to the two eigenvalues above. We now make some observations:

- If $k^2 \geq \omega_i^2$ we have that $\bar{\omega}_i$ is real and non-positive. In case of the first eigenvalue $\omega_i^2 = 0$, the corresponding two eigenvalues of Ω are $-2k$ and 0. Hence, there is a component of the bias that exponentially decays, while the other remains constant. For the others, the two eigenvalues are always negative and real, indicating that these components exponentially decay.

- If $k^2 \leq \omega_i^2$ we have the usual oscillatory behavior, which comes from the imaginary term from the square root. Still, the term $-k$ gives an exponential decay.

By choosing the appropriate values of the damping coefficient, the model learns to filter certain features in the graph.

Non-symmetric Laplacian $\hat{\mathbf{L}}$. We now discuss the case of a non-symmetric Laplacian matrix $\hat{\mathbf{L}}$, which will become useful in the practical implementation of STORM.

If we decompose the Laplacian $\hat{\mathbf{L}} = \hat{\mathbf{S}} + \hat{\mathbf{K}}$ as a sum of a symmetric component $\frac{\hat{\mathbf{L}}+\hat{\mathbf{L}}^\top}{2} = \hat{\mathbf{S}} = \hat{\mathbf{S}}^\top$ and a skew-symmetric one $\frac{\hat{\mathbf{L}}-\hat{\mathbf{L}}^\top}{2} = \hat{\mathbf{K}} = -\hat{\mathbf{K}}^\top$, the symmetric component behaves exactly as we discussed throughout this Section, recovering conservative wave-propagation dynamics. The skew-symmetric component introduces a purely non-reciprocal, direction-dependent transport term; notably, for any skew-symmetric matrix $\hat{\mathbf{K}}$, $\mathbf{h}^\top \hat{\mathbf{K}} \mathbf{h} = 0$, meaning that it does not directly inject or dissipate energy in the system. Instead, it modifies the direction of propagation while preserving the energy balance considered in our stability analysis.

Hence, by implementing a general Laplacian that is not generally symmetric, we retain a conservative component of the wave-propagation dynamics while adding direct transfer of information when needed, all in a single learnable component. Intuitively, having a different weight $\mathbf{a}_{vu} \neq \mathbf{a}_{uv}$ models a medium where information travels at different speeds in opposite directions, providing a more flexible generalization of the fixed symmetric-resistance case.

C.2 Proof of steady state under sufficient dissipation

Theorem 3.1 (Steady states)**.** *Consider a STORM layer with constant forcing term $\mathbf{F}$, Laplacian $\widehat{\mathbf{L}}$ of a connected graph, and invertible weight matrix $\mathbf{W}$. Then any steady state must have the form*

$$\mathbf{H}^* = \widehat{\mathbf{L}}^\dagger \mathbf{F} \mathbf{W}^{-1} + \mathbf{1}\mathbf{c}^\top, \tag{8}$$

where $\mathbf{1}\mathbf{c}^\top$ is a matrix whose columns lie in the kernel of the Laplacian $\widehat{\mathbf{L}}$ parametrised by $\mathbf{c} \in \mathbb{R}^d$, and $\mathbf{1} \in \mathbb{R}^n$ the vector of all ones. Moreover, if $\mathbf{1}^\top \mathbf{F} \neq \mathbf{0}$ then no steady state exists.

Proof. A steady state $\mathbf{H}^*$ is characterised by $\ddot{\mathbf{H}}(t) = \dot{\mathbf{H}}(t) = \mathbf{0}$. Plugging these conditions into the STORM equation, under the assumption of constant forcing $\mathbf{F}$, reads:

$$\widehat{\mathbf{L}} \mathbf{H}^* \mathbf{W} = \mathbf{F}.$$

This means that $\mathbf{H}^*$ must solve the linear equation: $\widehat{\mathbf{L}} \mathbf{H}^* = \mathbf{F} \mathbf{W}^{-1}$. If the graph is connected, then $\ker(\widehat{\mathbf{L}})$ is spanned by the vector of all ones $\mathbf{1}$. Therefore, if $\mathbf{H}^*$ is a solution of $\widehat{\mathbf{L}} \mathbf{H}^* = \mathbf{F} \mathbf{W}^{-1}$, then also $\tilde{\mathbf{H}} = \mathbf{H}^* + \mathbf{1}\mathbf{c}^\top$ for some $\mathbf{c} \in \mathbb{R}^n$, is a solution of $\widehat{\mathbf{L}} \mathbf{H}^* = \mathbf{F} \mathbf{W}^{-1}$. Therefore, any steady state of STORM must have the following structure:

$$\mathbf{H}^* = \widehat{\mathbf{L}}^\dagger \mathbf{F} \mathbf{W}^{-1} + \mathbf{1}\mathbf{c}^\top.$$

Notably, a necessary condition for the existence of a steady state is that $\mathbf{1}^\top \mathbf{F} = \mathbf{0}$, because $\mathbf{1}^\top \widehat{\mathbf{L}} = \mathbf{0}$; otherwise, $\mathbf{1}^\top \widehat{\mathbf{L}} \mathbf{H}^* = \mathbf{1}^\top \mathbf{F} \mathbf{W}^{-1}$ has no solution. □

C.3 Proof of marginal Stability

Theorem 3.2 (Marginal stability)**.** *In a STORM layer with constant coefficients, small coupling, and small step size, each node can operate around a regime of marginal stability. Moreover, a necessary condition for stability of a STORM layer is that $0 \leq D(\mathbf{H}(t)) \leq \frac{2}{\epsilon}$.*

Proof. From Equation (3), substituting the $\mathbf{V}$ equation into the $\mathbf{H}$ equation:

$$\begin{aligned}
\mathbf{H}_{t+1} &= \mathbf{H}_t + \epsilon \left(\mathbf{V}_t - \epsilon \left(\widehat{\mathbf{L}} \mathbf{H}_t \mathbf{W} + D(\mathbf{H}_t) \odot \mathbf{V}_t - F(\mathbf{H}_t, \mathbf{X}_t) \right) \right) \\
&= \mathbf{H}_t + \epsilon \mathbf{V}_t - \epsilon^2 \left(\widehat{\mathbf{L}} \mathbf{H}_t \mathbf{W} + D(\mathbf{H}_t) \odot \mathbf{V}_t - F(\mathbf{H}_t, \mathbf{X}_t) \right) \\
&= \mathbf{H}_t - \epsilon^2 \left(\widehat{\mathbf{L}} \mathbf{H}_t \mathbf{W} - F(\mathbf{H}_t, \mathbf{X}_t) \right) + \left(\epsilon \mathbf{1} - \epsilon^2 D(\mathbf{H}_t) \right) \odot \mathbf{V}_t,
\end{aligned}$$

where $\mathbf{1}$ is the matrix with all ones, same dimension of $\mathbf{H}$. Now we use the relation $\mathbf{V}_t = \dfrac{\mathbf{H}_t - \mathbf{H}_{t-1}}{\epsilon}$ to get that $\left(\epsilon\mathbf{1} - \epsilon^2 D(\mathbf{H}_t)\right) \odot \mathbf{V}_t = (\mathbf{1} - \epsilon D(\mathbf{H}_t)) \odot (\mathbf{H}_t - \mathbf{H}_{t-1})$. Hence, we obtain the following second-order difference equation:

$$\begin{aligned}\mathbf{H}_{t+1} = &(2\mathbf{1} - \epsilon D(\mathbf{H}_t)) \odot \mathbf{H}_t \\ &- \epsilon^2 \left(\widehat{\mathbf{L}}\mathbf{H}_t\mathbf{W} - F(\mathbf{H}_t, \mathbf{X}_t)\right) \\ &- (\mathbf{1} - \epsilon D(\mathbf{H}_t)) \odot \mathbf{H}_{t-1}.\end{aligned}$$

The second-order difference equation of STORM can be reduced into a vectorised form, which for the case of constant damping $\mathbf{D}$ and forcing $\mathbf{F}$, reads as follows:

$$\text{vec}(\mathbf{H}_{t+1}) = \mathbf{A}\text{vec}(\mathbf{H}_t) + \mathbf{B}\text{vec}(\mathbf{H}_{t-1}) + \text{vec}(\epsilon^2\mathbf{F}), \tag{30}$$

where

$$\mathbf{A} = \text{diag}\left(\text{vec}\left(2\mathbf{1} - \epsilon\mathbf{D}\right)\right) - \epsilon^2\mathbf{W}^\top \otimes \widehat{\mathbf{L}} \tag{31}$$

$$\mathbf{B} = -\text{diag}\left(\text{vec}\left(\mathbf{1} - \epsilon\mathbf{D}\right)\right) \tag{32}$$

Let us analyse the behaviour of the corresponding homogeneous part of this equation by writing the second-order vectorised equation into a first-order vectorised equation as follows:

$$\begin{bmatrix} \text{vec}(\mathbf{H}_{t+1}) \\ \text{vec}(\mathbf{H}_t) \end{bmatrix} = \mathbf{M} \begin{bmatrix} \text{vec}(\mathbf{H}_t) \\ \text{vec}(\mathbf{H}_{t-1}) \end{bmatrix}, \tag{33}$$

where the matrix to iterate is

$$\mathbf{M} = \left[\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \mathbf{I} & \mathbf{0} \end{array}\right]. \tag{34}$$

In this form, we have that

$$\begin{bmatrix} \text{vec}(\mathbf{H}_{t+1}) \\ \text{vec}(\mathbf{H}_t) \end{bmatrix} = \mathbf{M}^t \begin{bmatrix} \text{vec}(\mathbf{H}_1) \\ \text{vec}(\mathbf{H}_0) \end{bmatrix}. \tag{35}$$

When the spectral radius of the matrix $\mathbf{M}$ is strictly less than 1, then the dynamics of the homogeneous equation converges to the steady state $\mathbf{H}^*$, that is the solution of the equation $\widehat{\mathbf{L}}\mathbf{H}^*\mathbf{W} = \mathbf{F}$, which depends on the forcing term. Contrariwise, the homogeneous equation will exhibit unstable behaviour in the long term if the spectral radius of the matrix $\mathbf{M}$ is greater than 1, producing unbounded responses in the long term. On the other hand, the STORM layer operates in a regime of marginal stability if the spectral radius of $\mathbf{M}$ is approximately 1. Let's assume a small enough coupling $\mathbf{W}$ and step size ϵ, so that the term $\epsilon^2\mathbf{W}^\top \otimes \widehat{\mathbf{L}}$ becomes negligible. Under this assumption, the matrix $\mathbf{M}$ reduces to block diagonal matrices $\mathbf{A} = \text{diag}(\mathbf{a}), \mathbf{B} = \text{diag}(\mathbf{b})$, and we can derive a complete characterisation of the spectrum of $\mathbf{M}$. Each eigenvalue λ of $\mathbf{M}$ has the form $\lambda = \dfrac{a_i \pm \sqrt{a_i^2 + 4b_i^2}}{2}$ Now, for STORM we have $a_i = 2 - \epsilon d_i$, and $b_i = \epsilon d_i - 1$, where d_i is the i-th component of $\text{vec}(\mathbf{D})$. Thus, $\sqrt{a_i^2 + 4b_i^2} = |\epsilon d_i|$. Therefore, the eigenvalues have the form:

$$\lambda = \frac{2 - \epsilon d_i \pm |\epsilon d_i|}{2} \begin{cases} 1 - \epsilon d_i, & (-) \\ 1, & (+) \end{cases}. \tag{36}$$

In particular, we get the following characterization of the spectral radius of $\mathbf{M}$: $\rho(\mathbf{M}) = \max(1, \max_i\{1 - \epsilon d_i\})$. This gives the necessary condition of stability $d_i \geq 0$, otherwise there is an eigenvalue $\lambda > 1$, and also the condition $d_i \leq \frac{2}{\epsilon}$, otherwise there is an eigenvalue $\lambda < -1$. Choosing a learnable dissipation term with non-negative components, like in STORM allows the model to adaptively tune the dynamics toward more stable or at the edge stability dynamics by leaning to not dissipate in certain dimensions, i.e. moving towards $d_i = 0$. Finally, the small $\epsilon^2\mathbf{W}^\top \otimes \widehat{\mathbf{L}}$ term adds a further learnable matrix that allows the STORM layer to operate more flexibly around the marginally stable regime. □

C.4 Proof of upper bound sensitivity

Theorem 3.3 (Upper bound sensitivity)**.** *Let us consider two nodes u and v being at distance L from each other, and let us denote as $\mathbf{h}_t^u$ the embedding of node u at time $t \in [0, L]$. Assume $D(\mathbf{H}_t) = ReLU\,(\mathbf{H}_t\mathbf{W}_D)$, where $\mathbf{W}_D \in \mathbb{R}^{d\times d}$ is a matrix of weights, and $F(\mathbf{H}_t, \mathbf{X}_t) = \tanh\left(\boldsymbol{S}[\mathbf{H}_t, \mathbf{X}_t][\mathbf{W}_{Fh}, \mathbf{W}_{Fx}]^\top\right)$, where $\mathbf{W}_{Fh} \in \mathbb{R}^{d\times d}$, $\mathbf{W}_{Fx} \in \mathbb{R}^{c\times d}$ are matrices of weights, $[\cdot,\cdot]$ denotes the horizontal concatenation of matrices, $[\cdot,\cdot]^\top$ denotes the vertical concatenation of matrices, and $\mathbf{S}$ denotes the graph shift operator that rules the forcing dynamics. Assuming $\mathbf{A}$ is the adjacency matrix and node embeddings to stay within a bounded norm R, then*

$$\left\|\frac{\partial \mathbf{h}_L^u}{\partial \mathbf{h}_0^v}\right\| \leq (\mathbf{A}^L)_{vu}\left(2 + \epsilon 3R\|\mathbf{W}_D\|_2 + \epsilon^2\left(\|\mathbf{S}\|_{\max}\|\mathbf{W}_{Fh}\|_2 + \|\widehat{\mathbf{L}}\|_{\max}\|\mathbf{W}\|_2\right)\right)^L. \tag{9}$$

Proof. We write the second-order difference equation in a node-wise fashion to highlight the sensitivity of a node u at time step t with respect to another node v at time step s, with $s < t$. For this computation, we use the assumption $D(\mathbf{H}_t) = \text{ReLU}\,(\mathbf{H}_t\mathbf{W}_D)$, where $\mathbf{W}_D \in \mathbb{R}^{d\times d}$ is a matrix of weights, and $F(\mathbf{H}_t, \mathbf{X}_t) = \tanh\left(\mathbf{S}[\mathbf{H}_t, \mathbf{X}_t][\mathbf{W}_{Fh}, \mathbf{W}_{Fx}]^\top\right)$, where $\mathbf{W}_{Fh} \in \mathbb{R}^{d\times d}$ and $\mathbf{W}_{Fx} \in \mathbb{R}^{c\times d}$ are matrices of weights, $[\cdot,\cdot]$ denotes the horizontal concatenation of matrices, $[\cdot,\cdot]^\top$ denotes the vertical concatenation of matrices, and $\mathbf{S} \in \mathbb{R}^{n\times n}$ denotes the graph shift operator that rules the forcing dynamics. As already demonstrated in the proof of Theorem 3.2 we can write the discrete-time STORM layer equation in matricial form as follows:

$$\begin{aligned}\mathbf{H}_{t+1} = &(2\mathbf{1} - \epsilon D(\mathbf{H}_t)) \odot \mathbf{H}_t \\ &- \epsilon^2\left(\widehat{\mathbf{L}}\mathbf{H}_t\mathbf{W} - F(\mathbf{H}_t, \mathbf{X}_t)\right) \\ &- (\mathbf{1} - \epsilon D(\mathbf{H}_t)) \odot \mathbf{H}_{t-1}.\end{aligned}$$

We rewrite this equation from the perspective of node u, as follows:

$$\mathbf{h}_{t+1}^u = 2\mathbf{I} - \epsilon\text{ReLU}\left(\mathbf{h}_t^u\mathbf{W}_D\right) \odot \mathbf{h}_t^u \tag{37}$$

$$+ \epsilon\text{ReLU}\left(\mathbf{h}_t^u\mathbf{W}_D\right) \odot \mathbf{h}_{t-1}^u \tag{38}$$

$$+ \epsilon^2\left[\tanh\left(\left(\mathbf{S}[\mathbf{H}_t, \mathbf{X}_t]\right)^u[\mathbf{W}_{Fh}, \mathbf{W}_{Fx}]^\top\right) - \left(\widehat{\mathbf{L}}\mathbf{H}_t\right)^u\mathbf{W}\right], \tag{39}$$

where $\left(\mathbf{S}[\mathbf{H}_t, \mathbf{X}_t]\right)^u = \left[\sum_{v\in\mathcal{N}(u)}\mathbf{S}_{uv}\mathbf{h}_t^v, \sum_{v\in\mathcal{N}(u)}\mathbf{S}_{uv}\mathbf{x}_t^v\right]$ and $\left(\widehat{\mathbf{L}}\mathbf{H}_t\right)^u = \sum_{v\in\mathcal{N}(u)}\widehat{\mathbf{L}}_{uv}\mathbf{h}_t^v$ denote the rows related to the aggregation to the u node. Deriving this relation with respect to $\mathbf{h}_t^v$, for a given node v, yields the one-step sensitivity matrix:

$$\frac{\partial \mathbf{h}_{t+1}^u}{\partial \mathbf{h}_t^v} = 2\mathbf{I} - \epsilon\text{diag}\left(\text{ReLU}\left(\mathbf{h}_t^t\mathbf{W}_D\right)\right) \tag{40}$$

$$+ \epsilon\left[\text{diag}\left(\left(\mathbf{h}_{t-1}^u - \mathbf{h}_t^u\right) \odot \mathbf{1}_{\{\mathbf{h}_t^u\mathbf{W}_D>0\}}\right)\right]\mathbf{W}_D^\top \tag{41}$$

$$+ \epsilon^2\left[\mathbf{S}_{vu}\text{diag}(\tanh'\left(\left(\mathbf{S}[\mathbf{H}_t, \mathbf{X}_t]\right)^u[\mathbf{W}_{Fh}, \mathbf{W}_{Fx}]^\top\right))\mathbf{W}_{Fh}^\top - \widehat{\mathbf{L}}_{vu}\mathbf{W}^\top\right], \tag{42}$$

where $\mathbf{1}_{\{\cdot\}}$ is the indicator function, equal to 1 if the condition holds and 0 otherwise, thus

$$(\mathbf{1}_{\{\mathbf{h}_t^u\mathbf{W}_D>0\}})_i = \begin{cases}1, & \text{if } (\mathbf{h}_t^u\mathbf{W}_D)_i > 0,\\ 0, & \text{if } (\mathbf{h}_t^u\mathbf{W}_D)_i \leq 0.\end{cases}$$

From the one-step sensitivity matrix we can derive the following upper bound:

$$\max_{u,v}\left\|\frac{\partial \mathbf{h}_{t+1}^u}{\partial \mathbf{h}_t^v}\right\| \leq 2 + \epsilon\|D(\mathbf{H}_t)\|_{\max} \tag{43}$$

$$+ \epsilon\max_u\|\mathbf{h}_{t-1}^u - \mathbf{h}_t^u\|_\infty\|\mathbf{W}_D\|_2 \tag{44}$$

$$+ \epsilon^2\left(\|\mathbf{S}\|_{\max}\|\mathbf{W}_{Fh}\|_2 + \|\widehat{\mathbf{L}}\|_{\max}\|\mathbf{W}\|_2\right), \tag{45}$$

where for a given matrix $\mathbf{M}$, we denoted $\|\mathbf{M}\|_{\max} = \max_{u,v}|\mathbf{M}_{uv}|$, $\|\mathbf{M}\|_2$ the spectral norm, and for a given vector $\mathbf{m}$, we denoted $\|\mathbf{m}\|_\infty = \max_i|\mathbf{m}_i|$. We conclude by noticing that, if all

embeddings stay within the ball of radius R, then the maximum of $\|\mathbf{h}_{t-1}^u - \mathbf{h}_t^u\|_\infty$ is $2R$, and that $\|D(\mathbf{H}_t)\|_{\max} = \max_u \|\text{ReLU}(\mathbf{h}_t^u, \mathbf{W}_D)\|_\infty \leq \max_u ||\mathbf{h}_t^u||_2 ||\mathbf{W}_D||_2 \leq R||\mathbf{W}_D||_2$.

Different from other spatio-temporal architectures [20, 22, 81], our STORM integrates spatial and temporal information at each step. Thus, if two nodes u and v are at a distance L, then in a number L of steps the information from node v can reach node u through a given walk $(v = v_0, v_1, \ldots, v_L = u)$. This means we can decompose the sensitivity of how node u is affected by node v in the following way:

$$\frac{\partial \mathbf{h}_L^u}{\partial \mathbf{h}_0^v} = \sum_{v0 \to \ldots \to v_L} \frac{\partial \mathbf{h}_L^{v_L}}{\partial \mathbf{h}_{L-1}^{v_{L-1}}} \cdots \frac{\partial \mathbf{h}_2^{v_2}}{\partial \mathbf{h}_1^{v_1}} \frac{\partial \mathbf{h}_1^{v_1}}{\partial \mathbf{h}_0^{v_0}}, \tag{46}$$

where the sum is computed along all possible walks from v to u. Now, the number of walks starting from v and reaching u is $(\mathbf{A}^L)_{vu}$, from which we can derive the following upper bound:

$$\left\| \frac{\partial \mathbf{h}_L^u}{\partial \mathbf{h}_0^v} \right\| \leq (\mathbf{A}^L)_{vu} \left(\max_{u,v} \left\| \frac{\partial \mathbf{h}_{t+1}^u}{\partial \mathbf{h}_t^v} \right\| \right)^L. \tag{47}$$

□

We note that Theorem 3.3 and its proof can be easily adapted to other activation functions or different dissipation and forcing terms. In particular, the dissipative component influences two terms:

- $|\mathbf{D}(\mathbf{H}_t)|$. If we substitute ReLU with any differentiable and positive (as required by Theorem 3.2) function $\mathbf{D}$, for simplicity with $\mathbf{D}(0) = 0$, this term can be bounded as $|\mathbf{D}'||\mathbf{h}_t| \leq |\mathbf{D}'|R$, where the first term is the Lipschitz constant of $\mathbf{D}$ and R indicates the radius of the ball R where node states are contained.
- $\max_u \|\mathbf{h}_{t-1}^u - \mathbf{h}_t^u\|_\infty \|\mathbf{W}_D\|_2$. For this term, it is sufficient to notice that it was calculated by taking the derivative of ReLU $(\mathbf{h}_t^u \mathbf{W}_D)$ with respect to $\mathbf{h}_t^u$. As in the previous point, considering a general activation, an additional term $|\mathbf{D}'|$ appears in the upper bound, introducing a multiplicative factor due to its Lipschitz constant (which was 1 for ReLU).

About the external forcing term, we see that its main influence is given by its derivative in Equation (42). Similarly to the dissipative component, one can substitute $\tanh$ with any differentiable activation function with a bounded derivative. This leads to a σ' term instead of $\tanh'$, with the final addition of the term $c_\sigma = \max |\sigma'|$. The main result remains very similar.

C.5 Universality

Before showing the proof of the Universality theorem in Section 3.4, we provide a more detailed description of the Neural Oscillator framework, along with some important definitions. The general objective is to learn an operator Φ that maps a time-dependent input signal $\mathbf{X}(t)$ to an output signal $\Phi(\mathbf{X}(t)) = \mathbf{Y}(t)$. We assume that the signals are continuous functions in the space

$$C([0,T]; \mathbb{R}^{n \times c}) = \left\{ \mathbf{X} : [0,T] \to \mathbb{R}^{n \times c} | t \mapsto \mathbf{X}(t) \text{ is continuous} \right\} \tag{48}$$

Compared to [67], we do not require input signals to be null at 0, as we allow our model to have non-zero initial conditions. Additionally, Φ as an operator is required to be continuous (with respect to the L^∞ norm between function spaces) and *causal*. Formally, Φ is causal if, given two input signals $\mathbf{X}, \mathbf{Y}$ such that $\mathbf{X}|_{[0,t]} = \mathbf{Y}|_{[0,t]}$, then $\Phi(\mathbf{X})(t) = \Phi(\mathbf{Y})(t)$. In practice, this means that present outputs depend only on past states. The main difference from the original result from [67] comes from the fact that STORM retains the architectures, and thus the limitations, of an MPNN architecture (i.e., permutation-invariant operations and shared weights). Thus, operators that distinguish nodes with identical inputs cannot be modeled [121], and we therefore restrict to the case of inputs that can indeed distinguish nodes, such as positional encodings.

The general form of Neural Oscillator described in [67] is given by

$$\begin{cases} \ddot{\mathbf{h}}(t) = \sigma(\mathbf{B}\mathbf{h}(t) + \mathbf{C}\mathbf{X}(t) + \mathbf{b}) \\ \mathbf{h}(0) = \dot{\mathbf{h}}(0) = 0 \\ \mathbf{z}(t) = \mathbf{D}\mathbf{h} + \mathbf{d} \end{cases} \tag{49}$$

where σ is a C^∞ activation function with $\sigma(0) = 0$ and $\sigma'(0) = 1$, which includes the identity function. The solution to Equation (49) actually describes an operator between the input $\mathbf{X}(t)$ and the output $\mathbf{z}(t)$. Then, the main result can be summarized by the following statement:

Theorem C.4 (Theorem 3.1 of [67])**.** *Let $\Phi : C_0([0,T];\mathbb{R}^c) \to C_0([0,T];\mathbb{R}^c)$ be a causal and continuous operator and let $K \subset C_0([0,T];\mathbb{R}^c)$ be compact. Then, for any $\epsilon > 0$, there exists a latent dimension d, weights $\mathbf{B}, \mathbf{C}, \mathbf{D}$, and biases $\mathbf{b}, \mathbf{d}$ such that the output of the multi-layer neural oscillator with input satisfies:*

$$\sup_{t\in[0,T]} |\Phi(u)(t) - z(t)| \leq \epsilon, \quad \forall u \in K. \tag{50}$$

To prove the universality of STORM, it is sufficient to cast our model into the framework of [67]. We report here our statement:

Theorem 3.4 (Universality)**.** *Let $\Phi : C([0,T];\mathbb{R}^{n\times c}) \to C([0,T];\mathbb{R}^{n\times c'})$ be a causal and continuous operator, and consider inputs $\mathbf{X}(t)$ that discriminate nodes (i.e., no non-trivial graph automorphisms fix the inputs). Let Ψ_θ be the operator that maps the input $\mathbf{X}(t)$ to the solution of Equation* (2) *with initial conditions given by $\mathbf{H}_0 = \mathbf{X}(0)\mathbf{W}_H$ and $\mathbf{V}_0 = \mathbf{H}_0\mathbf{W}_V$, i.e., Ψ_θ represents STORM. Let $K \subset C([0,T];\mathbb{R}^{n\times c})$ be compact. Then, for any $\epsilon > 0$, there exist parameters θ s.t.*

$$\sup_{t\in[0,T]} |\Phi(\mathbf{X})(t) - \Psi_\theta(\mathbf{X})(t)| \leq \epsilon \quad \forall \mathbf{X} \in K. \tag{10}$$

Proof. To prove this Theorem, it is sufficient to cast the definition of our model on graphs to the one in Equation (49). We report here the definition of STORM:

$$\begin{cases} \ddot{\mathbf{H}}(t) = -\hat{\mathbf{L}}\mathbf{H}(t)\mathbf{W} - D(\mathbf{H}(t)) \odot \dot{\mathbf{H}}(t) + F(\mathbf{X}(t), \mathbf{H}(t) \\ \mathbf{H}(0) = \mathbf{X}(0)\mathbf{W}_I \\ \dot{\mathbf{H}}(0) = \mathbf{H}(0)\mathbf{W}_V \end{cases} \tag{51}$$

First, we reconcile with the zero initial conditions case of [67]. To do so, we consider new variables $\tilde{\mathbf{H}}(t) = \mathbf{H}(t) - \cos\left(\sqrt{\hat{\mathbf{L}}}t\right)\mathbf{C}_1 - \sin\left(\sqrt{\hat{\mathbf{L}}}t\right)\mathbf{C}_2$. We immediately notice that $\tilde{\mathbf{H}}(0) = \mathbf{H}(0) - \mathbf{C}_1 = \tilde{\mathbf{W}}_I\tilde{\mathbf{X}}(0) - \mathbf{C}_1$ and $\dot{\tilde{\mathbf{H}}}(0) = \dot{\mathbf{H}}(0) - \sqrt{\hat{\mathbf{L}}}\mathbf{C}_2 = \tilde{\mathbf{W}}_V\tilde{\mathbf{X}}(0) - \sqrt{\hat{\mathbf{L}}}\mathbf{C}_1$.

Now, our objective is to reconstruct a form of the Neural Oscillator as a particular case of STORM. To do so, we will make some simplifications and use specific forms of the weights of our model, although, in reality, STORM is more general and allows for more flexibility.
Specifically, we consider the following:

- We choose the matrix $\mathbf{C}_1 = \tilde{\mathbf{W}}_I\tilde{\mathbf{X}}(0)$ so that the initial condition of $\tilde{\mathbf{H}}$ is zero.
- Analogously, we choose the matrices $\tilde{\mathbf{W}}_V$ and $\mathbf{C}_2$ so that the initial velocity of $\tilde{\mathbf{H}}$ is zero. In reality, we could have chosen to always have a null initial condition by setting both matrices to zero. However, this further generalization allows us to better tune the model with the input signal.
- We set the dissipative term to be zero. As per the discussion in Sections 2 and 3 of [67], adding a dissipative term does not change the universality result. Indeed, it makes the model more general, and STORM can still learn to have null dissipation.

Now, we consider the eigendecomposition of the Laplacian matrix $\hat{\mathbf{L}} = \mathbf{Q}\Lambda\mathbf{Q}^\top$, where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$ is the diagonal matrix of eigenvalues, and $\mathbf{Q}$ contains the corresponding eigenvectors. For each channel $j \in \{1, \ldots, d\}$, we project the corresponding features onto the eigenspaces of the Laplacian, obtaining a new feature matrix $\mathbf{Y}(t)$, such that, on each feature channel, $\mathbf{Y}(t)_{\cdot,j} = \mathbf{Q}^\top\tilde{\mathbf{H}}_{\cdot,j}$. This new matrix, instead of representing the node features, represents the features of the oscillatory modes given by the Laplacian eigenspace.

We now make an additional restriction and assume that the weight matrix $\mathbf{W} \in \mathbb{R}^{d\times d}$ is diagonal, $\mathbf{W} = \text{diag}(\mu_1, \ldots, \mu_d)$, with $\mu_j > 0$, so that channels are independent. In reality, STORM will be more general, providing additional flexibility during learning. In this final form, each mode i

and channel j behaves as an oscillator $\ddot{y}_{ij} = -\lambda_i \mu_j y_{ij} + f_{ij}$, where f_{ij} is the projected forcing component. In particular, f_{ij} is obtained from linear and non-linear transformations of the input $\mathbf{X}(t)$.

If the graph is connected, $\lambda_1 > 0$ (otherwise we show universality on individual connected components). Thus, by considering the first oscillatory mode, we have access to d independent oscillators $\ddot{y}_{ij} = -\lambda_i \mu_j y_{ij} + f_{ij}$ that can be tuned directly by learning the parameters μ_j. This set of oscillators we constructed is structurally analogous to Eq. (2.9)/(3.3) of [67], and lets us invoke Lemmas 3.6 and 3.8 of the same work directly, which underlie the universality theorem.

We additionally observe that, in our work, we consider a general forcing term of the input sequence, constituted by a GCN and an MLP that take as input $\mathbf{X}(t)$ and $\mathbf{H}(t)$. For simplicity, as in the original work by [67], one can consider a linear version of it. Knowing that MLPs are universal models, our model is more general than a standard Neural Oscillator.

Recalling that we use a similar readout layer, we conclude that the Neural Oscillator in [67] and Equation (49) can be viewed as a particular case of STORM. Hence, STORM is universal. Most importantly, this proof shows that our model is universal even for input signals that do not start as null. □

Remark C.5. One could immediately see that STORM is in the exact form of a Neural Oscillator. The core of this proof is to show that our model is flexible enough to behave as a Neural Oscillator and that it can receive input signals that do not start at rest. To do so, we tune the initial condition of STORM via the input matrices.

C.6 Extending the results to changing topologies

Throughout our theoretical discussion, we assumed that the topology of the graph remains fixed in time. An interesting question is whether the theoretical statements in Section 3 and Appendix C remain valid in the varying topology case. We consider them one by one:

- Theorem 3.1 characterizes the stable states of STORM at a given point in time. Naturally, this steady state changes as the topology does so. There is no interesting extension of this result to this case.
- Theorem 3.2 extends naturally to a piecewise constant topology, with some remarks. The proof of the stability conditions is based on a small-coupling approximation, which makes the graph-dependent term negligible. This covers the case of gradual changes in the graph, where these approximations remain valid, but not severe ones, that is, for example, complete changes in the topology within a single time-step. This is an open question also in related fields, such as control theory [72]: switching between two different systems (even if individually stable) does not necessarily produce stable trajectories, especially with abrupt changes and working around marginal stability.
- Theorem 3.3 can be easily generalized to the case of changing topologies. In particular, it is sufficient to consider time-varying matrices (adjacency), (Laplacian), and (shift operator in the forcing term). Then, the Jacobian $\frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_t}$ can be computed in terms of these, and the maximum over time steps can be taken. Finally, the final steps of the proof need to take into account the number of walks between nodes with changing topologies.
- Theorem 3.4 does not easily convert to the changing topology case, as the proof is based on a fixed Laplacian matrix. Furthermore, since changes in the topology change the set of permutation-invariant functions on the graph, the set of operators that STORM can approximate changes with it.

D Experimental Details

D.1 Description of the Surface Wave Forecasting Task

We introduce a novel physics-based benchmark to study the ability of temporal GNNs to capture long-range spatio-temporal dependencies, termed the *Surface Wave Forecasting* task (SURF-WAVE). The task builds upon the WaveBall benchmark proposed by [59], extending it from static graph inputs to a sequence-to-sequence forecasting scenario over graphs.

Specifically, we use NVIDIA Warp [79] to simulate a two-dimensional wave field $u(\mathbf{x}, t)$ over an L-shaped domain $\Omega \subset \mathbb{R}^2$, governed by the forced wave equation

$$\partial_{tt} u(\mathbf{x}, t) \; - \; c^2 \, \Delta u(\mathbf{x}, t) \; = \; f(\mathbf{x}, t), \tag{52}$$

where the forcing term $f(\mathbf{x}, t)$ is generated by three moving balls interacting with the surface. The balls follow sinusoidal trajectories with distinct amplitudes, frequencies, and phase offsets, inducing surface waves that propagate across the domain.

The water surface is discretized as a grid graph, while the balls are modeled as mobile nodes moving over the grid. Given an input window of $T = 25$ time steps, the objective is to forecast the subsequent $H = 25$ steps of both the ball states and the wave field. To explicitly stress long-range communication, we connect each ball node only to the six corner nodes of the L-shaped grid. As a result, accurate prediction requires effective long-range spatio-temporal propagation to maintain consistency between the ball trajectories and the resulting wave dynamics.

Finally, we note that while the underlying physical process used to generate the task is fully known, it is independent of the evaluated models, providing a controlled setting in which to isolate the effects of spatio-temporal over-squashing. In principle, any model capable of sufficiently long-range information propagation and temporal memory should be able to solve the task, irrespective of its specific internal mechanism.

A visual illustration of the simulation setup and the corresponding graph topology is provided in Figure 3. SURF-WAVE's statistics are reported in Table 4.

Table 4: Statistics of the SURF-WAVE.

Nodes	Edges	Type	Time Steps	Input Window Size	Forecasting Horizon	Train/Valid/Test Split
678	2616	Undirected	50	25	25	700/100/200

Training details. In this task, we optimize a weighted mean-squared error that combines the wave-field regression and the ball-state prediction:

$$\mathcal{L}_t = \mathrm{MSE}(\hat{\mathbf{u}}_t, \mathbf{u}_t) + \lambda \, \mathrm{MSE}(\hat{\mathbf{p}}_t, \mathbf{p}_t) \, , \tag{53}$$

where we set $\lambda = 10$. Here, $\mathbf{u}_t = [u(t), \dot{u}(t)]$ denotes the ground truth wave state at the grid nodes, comprising both wave amplitudes and velocities, and $\hat{u}$ denotes the corresponding model prediction. Similarly, $\mathbf{p}_t = [\mathbf{p}(t), \dot{\mathbf{p}}(t)]$ represents the ground truth ball state, including positions and velocities, with $\hat{\mathbf{p}}_t$ denoting its predicted counterpart. The node features are defined as $\mathbf{X}_t = [\mathbf{u}(t), \dot{\mathbf{u}}(t), \mathbf{p}(t), \dot{\mathbf{p}}(t), \boldsymbol{\tau}]$ with $\boldsymbol{\tau}$ being a one-hot node-type indicator (i.e., boundary node, grid node, or ball node). We use $\boldsymbol{\tau}$ to mask the losses: the wave loss is computed only on grid nodes, while the ball loss is computed only on ball nodes (boundary nodes are excluded from both terms).

We train with batch size 8, use Adam as an optimizer with a weight decay 10^{-4}, gradient clipping at 5.0, and a cosine-annealing learning-rate schedule. We train for up to 5000 epochs with early stopping (patience $= 1000$) for both STORM and all baselines.

D.2 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

- Graph-agnostic temporal predictors, i.e., Historical Average (HA), univariate linear AR model (AR), SVR [107], and FC-LSTM [111], and VAR [77], Transformer [115], as well as FC-RNN and LocalRNN from [22];
- Temporal GNNs, i.e., DCRNN [70], GCRNN [101], Graph WaveNet [120], ASTGCN [53], GMAN [126], T-GCN [125], DyGrAE [112], EvolveGCN [89], AGCRN [5], MTGNN [119], STSGCN [109], GTS [102], MPNN LSTM [88], A3T-GCN [4], STEP [103], DynGESN [83], GRU-GCN [41], STAEformer [73], HMM4G [32], STD-MAE [40], RGDAN [34], AdpSTGCN [124], MPTCN [81], as well as RNN with isotropic message passing (IMP) and RNN anisotropic message passing (AMP) from [22];.

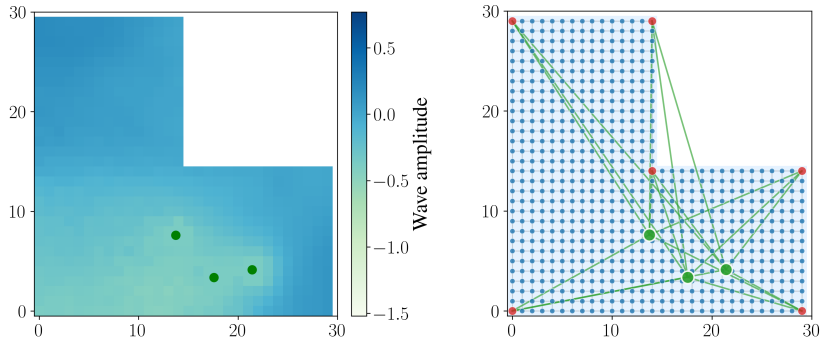


Figure 3: Surface Wave Forecasting task, i.e., SURF-WAVE. (a) The physical simulation, highlighting the wave amplitude after the movement of the three balls. (b) The graph topology employed in the SURF-WAVE task. The water surface is represented as an L-shaped grid graph (with nodes and edges in blue), while balls are modeled as nodes moving across the grid (colored in green). Ball nodes are connected only to the six corner nodes of the L-shaped grid (colored in red).

Table 5: Common hyperparameter settings and grid search ranges used across all tasks.

Component	**Setting / Search space**
Optimizer	Adam
LR scheduler	CosineAnnealingLR
Hidden size	128
Learning rate (lr)	$\{5\times10^{-5}, 1\times10^{-4}, 2\times10^{-4}, 5\times10^{-4}, 1\times10^{-3}, 2\times10^{-3}, 5\times10^{-3}\}$
Step size (ϵ)	$\{0.02, 0.05, 0.1, 0.2, 0.25, 0.3, 0.4\}$
Num. Layers (L)	$\{1, 2\}$
External forcing	{GCN + MLP forcing, GCN forcing, MLP forcing}
Residual forcing	{`False`, `True`}
Edge-esistance	{`Fixed`, `Adaptive`}

D.3 Hyperparameter Space

The hyperparameter search space used by STORM is summarized in Table 5, while Table 6 reports the best-performing configuration for each task. For Rocketman, we tune hyperparameters separately for each temporal–spatial setting. Additional results for different hyperparameter choices are provided in Appendix E.

E Additional Results

E.1 Runtime and Complexity Analysis

Table 7 reports the wall-clock runtime per forward pass on SURF-WAVE task, together with the corresponding predictive performance. All measurements are taken on a single NVIDIA A100 GPU, and we report the mean $\pm$ standard deviation (in milliseconds) over 20 repeated runs. Overall, STORM achieves competitive runtime among neural baselines while delivering the best predictive accuracy.

Finally, we note that a single propagation step of STORM exhibits runtime and memory consumption comparable to a standard MPNN [43], which has linear complexity in the graph size. Consequently, Equation (3) scales linearly with the number of nodes $|V|$ and edges $|E|$, yielding a total time complexity of $\mathcal{O}\left(T \cdot (|V| + |E|)\right)$, where T denotes the number of time steps. Accounting for the $\mathcal{O}(m)$ cost of the MLP in Equation (4), the overall time complexity of a single STORM layer becomes $\mathcal{O}\left(T \cdot (|V| + |E| + m)\right)$. When L layers are considered, the overall STORM complexity is $\mathcal{O}\left(TL \cdot (|V| + |E| + m)\right)$.

Table 6: Selected hyperparameters per task (best configuration from the grid in Table 5).

Task	LR	ϵ	External force	L	Edge-resistance
Rocketman	5×10^{-4}	0.30	GCN + MLP forcing	1 or 2	`Fixed` or `Adaptive`
Metr-LA	5×10^{-4}	0.10	GCN + MLP forcing	1	`Fixed`
PeMS-Bay	5×10^{-4}	0.10	GCN + MLP forcing	1	`Fixed`
EngRAD	5×10^{-4}	0.20	GCN + MLP forcing	1	`Adaptive`
WaveBalls	5×10^{-5}	0.25	GCN + MLP forcing	1	`Fixed`
Twitter Tennis	1×10^{-3}	0.005	GCN + MLP forcing	1	`Fixed`

Table 7: Runtime and MSE comparison on SURF-WAVE. We report mean $\pm$ std. wall-clock time per forward pass (ms) over 20 runs. **First**, **second**, and **third** results are color-coded.

Model	Runtime (ms)	MSE
DCRNN	32.02 ± 0.10	**4.95 ± 0.05**
AGCRN	35.39 ± 0.61	6.93 ± 0.13
Graph WaveNet	10.35 ± 0.07	5.14 ± 0.10
Graph WaveNet-TTS	10.15 ± 0.40	**5.07 ± 0.02**
MPTCN-TTS	1.91 ± 0.02	6.10 ± 0.60
Our		
STORM	26.56 ± 0.07	**4.67 ± 0.03**

Regarding memory, we note that a single STORM propagation step retains the same complexity of a standard MPNN, i.e., $\mathcal{O}\left((|V| + |E|)d\right)$ (where d is the hidden dimension). The final MLP in Equation (4) contributes only an additional $\mathcal{O}\left(dc\right)$ cost (where c is the output dimension), yielding a total complexity of $\mathcal{O}\left((|V| + |E|)d + dc\right)$ for one layer. During backpropagation through time, the overall complexity becomes $\mathcal{O}\left(T(|V| + |E|)d + dc\right)$, which scales linearly in both the graph size and the temporal horizon.

Since STORM does not introduce costly global operations (e.g., attention over all node pairs, like in Graph Transformers) and each update step only involves local message passing and element-wise operations (similarly to classical GNNs), we believe it remains well suited for large-scale real-world graphs. For comparison, a spatio-temporal Transformer such as STAEformer, which applies a temporal Transformer followed by a Graph Transformer, has $\mathcal{O}\left(|V|T^2 + |V|^2T + m\right)$ time and memory complexity (if both transformers have one layer and one attention head), owing to the quadratic cost of temporal and spatial self-attention.

To further assess the scalability of STORM, Table 8 compares its training time, inference time, and peak memory with those of the baselines on PeMS-Bay [70] and on the Greater Los Angeles split of LargeST [74]. For a fair comparison, all models are scaled to the same budget of about 150k parameters by adjusting only their width, while their depth and all other hyperparameters follow the published configurations. On PeMS-Bay, STORM trains about as fast as DCRNN and Graph WaveNet, and almost twice as fast as STAEformer, which also requires nearly three times more memory. STORM overtakes DCRNN, AGCRN, and Graph WaveNet on LargeST, training up to twice as fast, whereas STAEformer, whose spatial attention is quadratic in the number of nodes, runs out of memory. Overall, the results show that STORM achieves competitive computational efficiency while scaling favorably to larger graphs.

E.2 STORM and Time-Varying Topologies

To further evaluate the effectiveness of our STORM on the more general setting of evolving topologies, we consider the Twitter Tennis benchmark [93], where both node features and edges change over time. Specifically, Twitter Tennis is a mention graph in which nodes are Twitter accounts and their labels encode the number of mentions between them. We follow the experimental setup of [32].

Table 8: Training/inference runtime and peak GPU memory on PeMS-Bay and LargeST. For a fair comparison, we consider a single layer for each architecture and a parameter budget of $\sim$150k parameters. Times are averaged over 90 training steps (3 runs of 30 steps) on a single NVIDIA A100 GPU (40 GB), with a batch size of 64 on PeMS-Bay and of 8 on LargeST. STAEformer runs out of memory on LargeST.

Model	**Metric**	**PeMS-Bay** Nodes: 325 Edges: 2,369 Timestep: 52,116	**LargeST** Nodes: 3,834 Edges: 98,703 Timestep: 525,888
AGCRN	Train (ms/batch)	57.5 ± 0.2	216.4 ± 0.6
	Inf. (ms/batch)	22.0 ± 0.0	170.7 ± 0.1
	Peak mem. train (MiB)	1428	3733
	Peak mem. inf. (MiB)	118	251
DCRNN	Train (ms/batch)	99.6 ± 0.1	193.7 ± 0.1
	Inf. (ms/batch)	44.2 ± 0.1	87.7 ± 0.1
	Peak mem. train (MiB)	3417	5081
	Peak mem. inf. (MiB)	272	426
Graph WaveNet	Train (ms/batch)	93.1 ± 0.1	271.1 ± 0.6
	Inf. (ms/batch)	38.2 ± 0.1	116.8 ± 0.3
	Peak mem. train (MiB)	1461	2066
	Peak mem. inf. (MiB)	629	1612
GRU-GCN	Train (ms/batch)	26.3 ± 2.2	51.1 ± 1.4
	Inf. (ms/batch)	13.9 ± 0.0	32.8 ± 0.0
	Peak mem. train (MiB)	3794	5814
	Peak mem. inf. (MiB)	2603	3977
STAEformer	Train (ms/batch)	190.9 ± 0.3	OOM
	Inf. (ms/batch)	64.9 ± 0.1	OOM
	Peak mem. train (MiB)	9535	OOM
	Peak mem. inf. (MiB)	2647	OOM
STORM (ours)	Train (ms/batch)	101.0 ± 0.1	138.0 ± 0.1
	Inf. (ms/batch)	44.1 ± 0.1	62.2 ± 0.1
	Peak mem. train (MiB)	3440	4017
	Peak mem. inf. (MiB)	786	913

As shown in Table 9, STORM achieves performance comparable to the best baseline (with overlapping confidence intervals), indicating state-of-the-art results and demonstrating its effectiveness also with evolving graph topologies.

E.3 SURF-WAVE Visualization

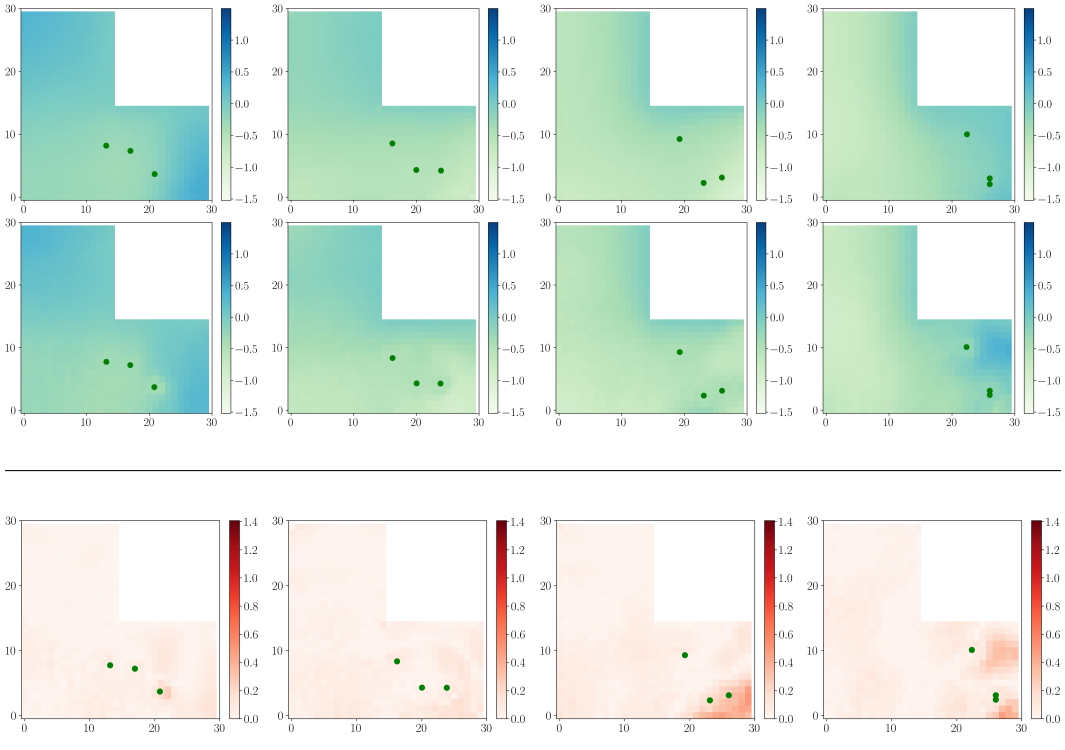
We present a qualitative comparison between predictions and ground truth for SURF-WAVE in Figure 4. In general, STORM accurately predicts the ball trajectories and produces a consistent corresponding wave field, in line with the quantitative results in Table 1, where STORM achieves the lowest MSE among all baselines. While the model captures the overall dynamics effectively, some fine-scale details exhibit slightly larger localized errors, particularly in the lower-right region at time steps t=15 and t=20. These observations highlight opportunities for further refinement in high-frequency wave modeling, which we leave for future work.

E.4 Full RocketMan Benchmark Results

In Figure 5, we report the full success-rate heatmaps for the RocketMan benchmark. We take the MPTCN results directly from [81]. For STORM, we use the best hyperparameter setting identified by the ablation study in Appendix F. As also discussed in Section 4.2, STORM achieves high success rates across a larger number of spatio-temporal distance combinations than MPTCN, corresponding to more configurations in the heatmap. This behavior highlights the benefit of our propagation scheme in mitigating oversquashing in both the spatial and temporal dimensions, allowing enhanced long-range

Table 9: Twitter Tennis results. We report test MSE ± STD over 10 runs. **First**, **second**, and **third** results are color-coded. Baseline results are reported from [32].

Model	MSE ↓
A3T-GCN	0.477 ± 0.005
DCRNN	0.478 ± 0.004
DyGrAE	0.480 ± 0.005
DynGESN	**0.300 ± 0.003**
EvolveGCN	0.481 ± 0.003
GCRNN	0.477 ± 0.007
GC-LSTM	0.475 ± 0.010
HMM4G	**0.333 ± 0.004**
MPNN LSTM	0.482 ± 0.001
T-GCN	0.478 ± 0.004
STORM (ours)	**0.302 ± 0.016**



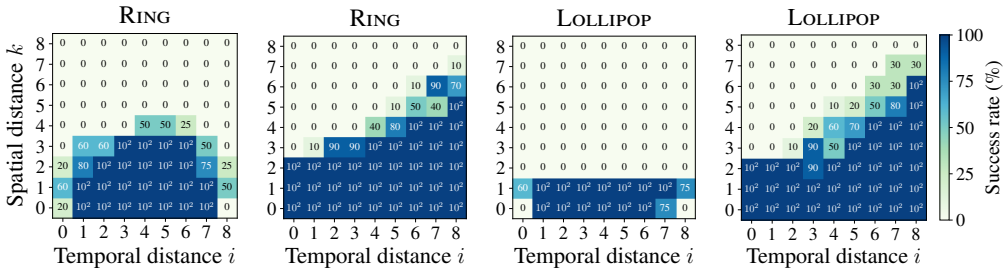
(a) $t = 5$ (b) $t = 10$ (c) $t = 15$ (d) $t = 20$

Figure 4: Comparison between predicted trajectory by STORM (top), its corresponding ground truth (middle) and the absolute errors for the wave amplitude (bottom) from timestep $t = \{5, 10, 15, 20\}$ on SURF-WAVE task.

spatio-temporal propagation. Appendix F.2 reports results for STORM at reduced width, including a comparison with MPTCN at a similar number of parameters.

E.5 Extended Comparison on the EngRad Benchmark

To further evaluate the performance of STORM, we report a more complete comparison for the EngRAD benchmark in Table 10. Specifically, we include more variants of the methods in Section 4.3, including isotropic and anisotropic message passing mechanisms, additional learnable node embeddings that enhance the model expressiveness, and more layers. Notably, our STORM achieves



(a) MPTCN ($P = 2$) (b) STORM (ours) (c) MPTCN ($P = 2$) (d) STORM (ours)

Figure 5: Rocketman task with full values.

second-best performance, closely trailing MPTCN, highlighting the effectiveness of our approach also in the weather forecasting setting.

Table 10: EngRAD forecasting results. Baseline results are taken from [22, 81]. We report test MAE $\pm$ STD; our result is averaged over 5 runs. "IMP" denotes isotropic message passing, "AMP" denotes anisotropic message passing, "+Emb." indicates the use of additional learnable node embeddings to enhance model expressiveness. R_N denotes row-normalized convolution, and L indicates the number of layers. **First**, **second**, and **third** results are color-coded.

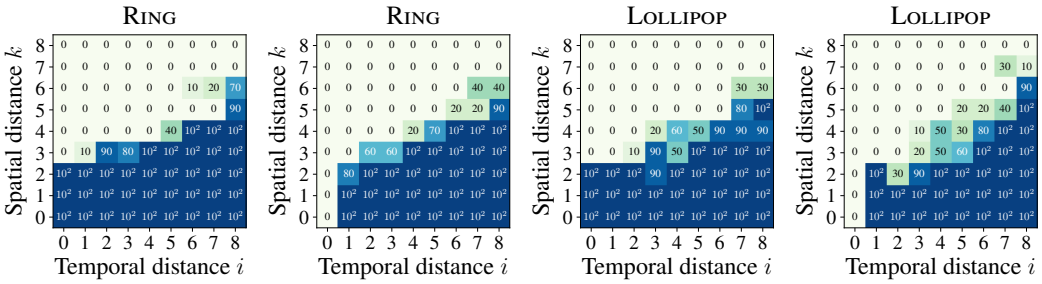
Model	**MAE** $\downarrow$
Graph Agnostic	
FC-RNN	55.37 ± 1.57
LocalRNN	58.80 ± 0.31
RNN	47.41 ± 0.57
RNN (+Emb.)	46.83 ± 0.19
Temporal GNNs	
GCRNN w/ IMP	45.55 ± 0.33
GCRNN w/ AMP	43.93 ± 0.55
GCRNN w/ IMP (+Emb.)	44.90 ± 0.33
GCRNN w/ AMP (+Emb.)	43.14 ± 0.19
MPTCN (R_N, L=6)	41.82 ± 0.38
MPTCN (R_N, L=3)	$\mathbf{41.78 \pm 0.09}$
MPTCN (R_N, L=1)	$\mathbf{40.38 \pm 0.08}$
RNN+IMP	44.48 ± 0.24
RNN+AMP	44.62 ± 0.35
RNN+IMP (+Emb.)	43.96 ± 0.42
RNN+AMP (+Emb.)	43.70 ± 0.33
Ours	
STORM	$\mathbf{41.74 \pm 0.17}$

F Ablation Studies

In this section, we analyze the contribution of the individual components of STORM to performance on the RocketMan task. Therefore, we examine how each component helps mitigate spatio-temporal over-squashing, enabling long-range propagation in both time and space.

Except from the ablation on the number of layers L, we considered L=2 as the default choice and report results up to a spatial distance $k \leq 6$ to reduce the number of experiments.

Number of layers. We first ablate the number of STORM layers $L \in \{1, 2\}$ in Figure 6. Overall, increasing the depth to L=2 leads to improved long-range *spatial* performance, as reflected by higher success rates at larger spatial distances. This improvement is particularly pronounced on the lollipop graph, a topology to stress spatial over-squashing due to its bottleneck structure [25]. This result is consistent with the intuition that increasing L effectively enlarges the propagation radius within a

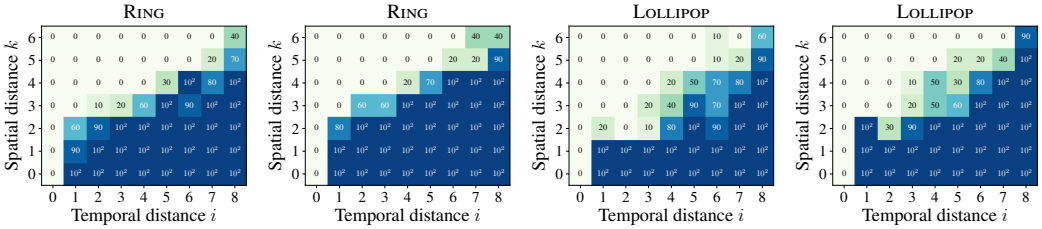


(a) $L = 1$ (b) $L = 2$ (c) $L = 1$ (d) $L = 2$

Figure 6: Performance with different number of layers L on the RocketMan task.

layer, allowing information to travel further across the graph. In contrast, for small temporal offsets (e.g., i=0), the shallower configuration (L=1) can outperform the deeper one (L=2). This suggests that when only short-range temporal dependencies are required, increasing the propagation depth may introduce excessive information mixing, degrading the predictive accuracy.

We recall that in the remaining ablations, we fix L=2 and report results up to spatial distance $k \leq 6$ to reduce the number of runs.



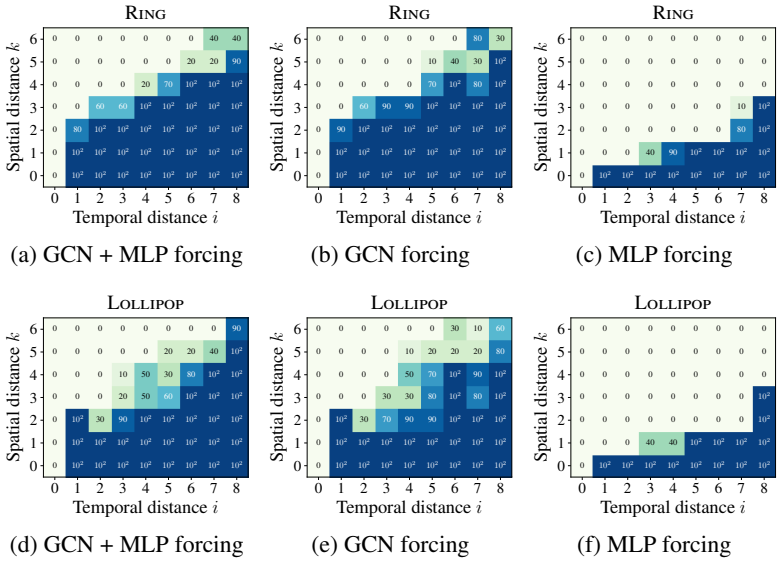
(a) w/o Dissipation (b) w/ Dissipation (c) w/o Dissipation (d) w/ Dissipation

Figure 7: Performance with and without dissipation on the RocketMan task.

Role of dissipation. Next, we study the effect of the dissipation term. In Figure 7 we compare a model variant without dissipation against our default setting with a learnable dissipation module (implemented as $D(\mathbf{H}_t) = \mathrm{ReLU}(\mathbf{H}_t\mathbf{W})$). Adding dissipation consistently increases the number of successful cells, especially in the harder long-range regimes. This suggests that dissipation improves expressiveness and stability by enabling controlled damping/forgetting, rather than forcing the model to rely on purely conservative propagation.

Role of forcing. We then evaluate the impact of different parametrizations for the forcing term $F(\cdot)$, as defined in Appendix B. As illustrated in Figure 8, configurations incorporating graph-aware transformations, i.e., *GCN + MLP forcing* and *GCN forcing*, consistently outperform the *MLP forcing* variant across all graph topologies. The performance drop observed when removing the GCN component suggests that neighborhood information plays an important role already at the signal injection stage, thereby enhancing the model's ability to capture complex spatio-temporal patterns. Furthermore, we observe that the GCN + MLP forcing variant provides a consistent improvement over the pure GCN implementation, suggesting that the additional non-linear residual provided by the MLP helps refine the injected features before their propagation through the second-order dynamics.

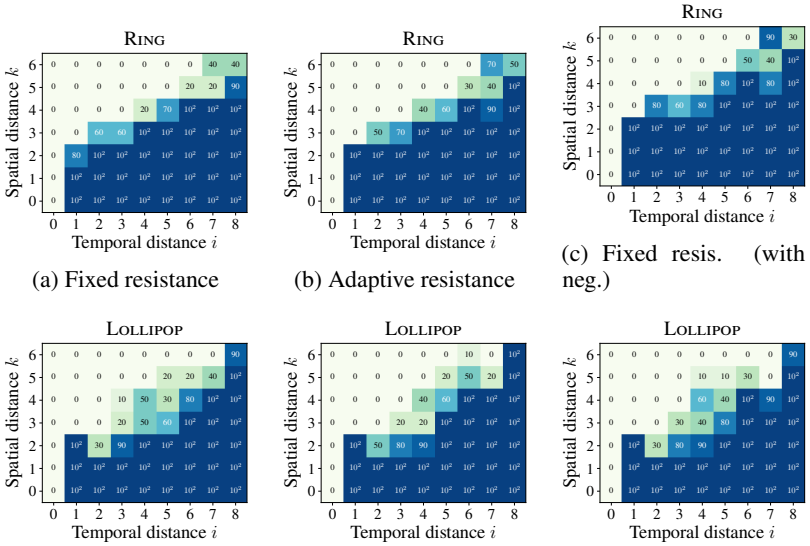
Fixed and adaptive edge resistance. Finally, we ablate different strategies (introduced in Appendix B) for computing edge resistance in Figure 9. Specifically, In the *fixed* variant, edge resistances are computed once at the beginning of each layer and then reused throughout the propagation steps. In the *adaptive* variant, resistances are recomputed at each time step, allowing the model to modulate information flow more flexibly over time. We observe that adaptive resistance can improve success rates in the more challenging long-range configurations, at the cost of additional computation during the forward pass. We also consider a variant that allows negative resistances; its effect is moderate and less consistent, suggesting limited benefit compared to the adaptive-positive setting.



(a) GCN + MLP forcing (b) GCN forcing (c) MLP forcing

(d) GCN + MLP forcing (e) GCN forcing (f) MLP forcing

Figure 8: Performance of three different forcing term formulations (i.e., *GCN + MLP forcing*, *GCN forcing*, and *MLP forcing*) on the RocketMan task.



(a) Fixed resistance (b) Adaptive resistance (c) Fixed resis. (with neg.)

(d) Fixed resistance (e) Adaptive resistance (f) Fixed resis. (with neg.)

Figure 9: Performance of different edge resistance formulations on the RocketMan task.

F.1 On the Importance of the Step Size

The step size ϵ is a hyperparameter of our STORM, and it plays an important role as it affects the quality of the solution to the wave equation[6]. In practice, in Table 11, we observed that the performance does not change significantly with small variations in the step size, while it is sufficient to try different orders of magnitude to achieve optimal results. For the ablation in Table 11, we considered the traffic forecasting benchmarks (with a prediction horizon of 12 steps) and trained an identical configuration of STORM with 6 different step sizes.

Moreover, we note that ϵ also plays a role in the stability of the system. We observe that a high number of iterations requires a lower step size, similar to classical numerical solvers for physical equations

[6]We stress that hyperparameters are important to find the optimal solution, but can be easily found via simple grid search or more advanced techniques.

Table 11: Performance on traffic forecasting tasks for the same configuration of STORM with different step size (i.e., ϵ) values.

ϵ	**0.01**	**0.02**	**0.03**	**0.05**	**0.1**	**0.2**
Pems-Bay	1.625 ± 0.003	1.599 ± 0.005	1.588 ± 0.004	$\mathbf{1.584 \pm 0.004}$	1.587 ± 0.002	1.791 ± 0.284
Metr-LA	3.099 ± 0.011	3.075 ± 0.020	3.054 ± 0.008	3.052 ± 0.007	$\mathbf{3.044 \pm 0.010}$	3.147 ± 0.109

[26]. Table 12 provides an example of the Metr-LA task (with increased prediction horizon up to 131 steps), where we show the evolution of the Jacobian over time (i.e., $(\|\mathbf{J}(t)\| - \|\mathbf{J}(t-1)\|)/\|\mathbf{J}(t-1)\|$).

We remark that STORM is implemented as an explicit discretization of the continuous-time solution of Equation (2) using the Euler method. However, one could adopt a different discretization scheme, yielding different stability regions and computational costs.

Table 12: The evolution of the Jacobian over time (i.e., $(\|\mathbf{J}(t)\| - \|\mathbf{J}(t-1)\|)/\|\mathbf{J}(t-1)\|$) on the Metr-LA task for the same configuration of STORM with different step size (i.e., ϵ) values. The symbol † denotes divergence, while values below 10^{-5} are shown as ≈ 0.

ϵ	$t=1$	$t=31$	$t=61$	$t=91$	$t=101$	$t=131$
0.01	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0
0.02	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0
0.05	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0
0.1	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	†
0.2	≈ 0	≈ 0	3.86×10^{-1}	†	†	†

F.2 Performance at Different Model Widths

A natural question is whether the advantages of STORM come from its propagation mechanism or simply from its capacity. To separate these two factors, we reduce the hidden size h of STORM and measure how its performance changes on the traffic forecasting benchmarks and on RocketMan.

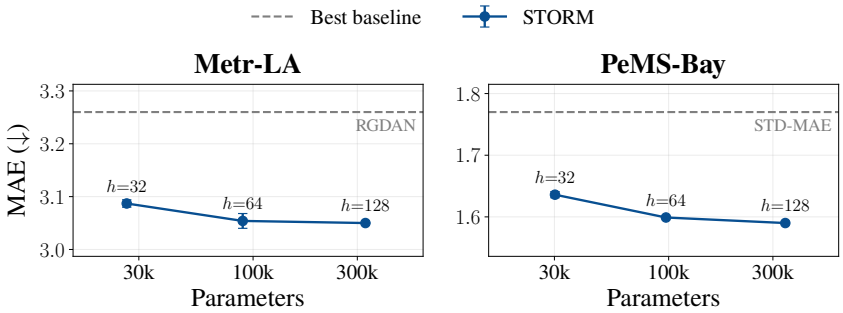
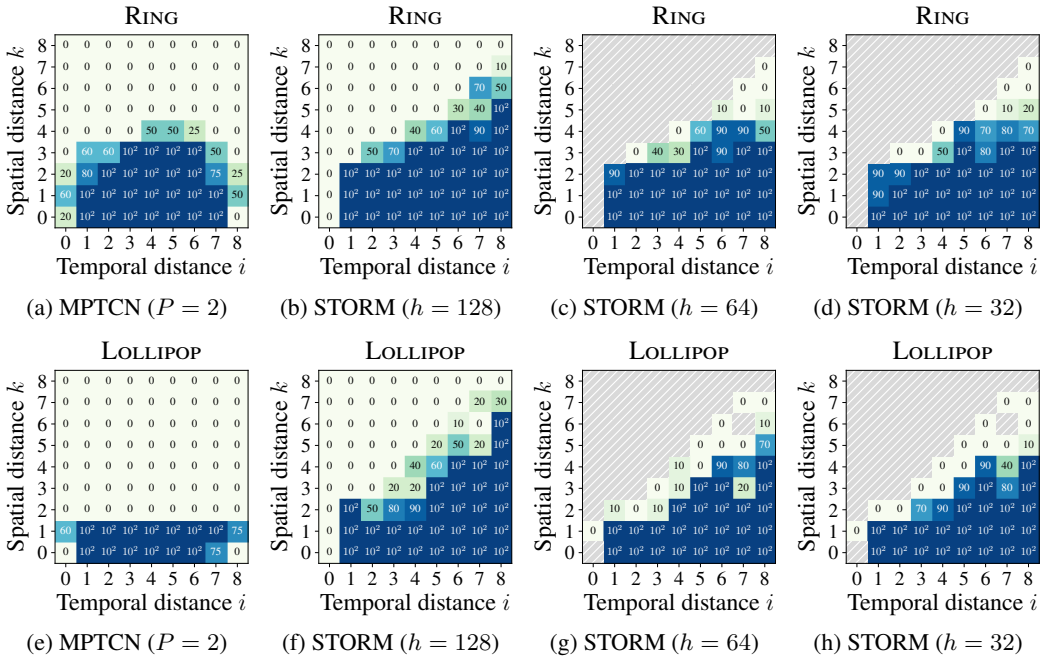


Figure 10: Parameter efficiency of STORM on traffic forecasting. Test MAE of STORM versus its number of parameters, for hidden sizes $h \in \{32, 64, 128\}$. Results for $h \in \{32, 64\}$ are averaged over 5 seeds, while $h = 128$ corresponds to the model in Table 2. The dashed line marks the best baseline in Table 2.

Traffic forecasting. We retrain STORM on Metr-LA and PeMS-Bay with smaller hidden sizes, keeping all other hyperparameters identical to those selected in Table 2. As shown in Figure 10, the accuracy of STORM is robust to its width. Halving the hidden size (i.e., from $h = 128$ to $h = 64$) leaves the MAE nearly unchanged while using less than a third of the parameters. Even at a quarter of the original width ($h = 32$), the MAE increases by less than 3%, and STORM still outperforms all baselines in Table 2 on both datasets.

RocketMan. Reducing the width also allows us to compare STORM and MPTCN at a similar number of parameters. As shown in Figure 11, the smallest variant of STORM ($h = 32$), whose parameter



(a) MPTCN ($P = 2$) (b) STORM ($h = 128$) (c) STORM ($h = 64$) (d) STORM ($h = 32$)

(e) MPTCN ($P = 2$) (f) STORM ($h = 128$) (g) STORM ($h = 64$) (h) STORM ($h = 32$)

Figure 11: Success rate (%) on RocketMan for STORM with decreasing hidden size h, compared with MPTCN, on the ring (top) and lollipop (bottom) graphs. MPTCN uses between 13.5k and 39.3k parameters depending on the configuration, while STORM uses 515k, 130k, and 33k parameters for $h = 128$, 64, and 32, respectively. Success rates of STORM are computed over 10 seeds in a separate width sweep, and thus differ from Figure 5, where each cell reports the best of several configurations. Hatched cells were not evaluated at reduced width, since STORM with $h = 128$ already has a 0% success rate there. MPTCN results are taken from [81].

count falls within the range of MPTCN, still propagates information over larger spatio-temporal distances. On the ring graph, it reaches a high success rate (at least 90%) in more configurations than MPTCN, and it succeeds at spatial distances where MPTCN is correct at most half of the time. The gap widens on the lollipop graph, whose bottleneck amplifies over-squashing and where MPTCN fails beyond one hop, whereas STORM reaches a high success rate in twice as many configurations and still propagates information up to four hops. As expected, a smaller width shrinks the region of high success, mostly at the largest spatial distances, yet STORM still outperforms MPTCN even at its smallest size.

Overall, these results indicate that the strong performance of STORM comes not from a larger number of parameters, but from its oscillatory propagation mechanism, which remains effective at a small fraction of the original model size.

G Broader Impact and Limitations

This paper aims to contribute to the field of Machine Learning, specifically by advancing the long-range propagation capabilities of Graph Neural Networks (GNNs) in temporal graph settings. The proposed approach is motivated by the need to effectively model long-range spatio-temporal dependencies, which are central to a wide range of applications. In this sense, the proposed framework may also inform future developments in related areas of GNN research.

While we believe this direction is impactful, it is important to emphasize that STORM is designed specifically to address the propagation problem in GNNs through the lens of oscillatory dynamics, with a focus on settings requiring long-range interaction modeling across both space and time, providing a principled understanding of the underlying dynamics and their expected behavior, while maintaining low computational overhead (as shown and analyzed in Appendix E.1), and achieving significant improvements over the state of the art in its target setting.

However, the method is tailored to regularly sampled discrete-time dynamic graphs. As such, it does not directly extend to irregularly sampled dynamic graphs, where observation times are non-uniform, nor to continuous-time dynamic graph settings. Addressing these more general settings remains an important direction for future work.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: All the claims found in the abstract and introduction are supported by the results and theorems discussed in the paper.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We mention (in Section 5 and Appendix G) that the the scope of this work is limited to discrete-time dynamic graphs (D-TDGs) and leave the expansion to other settings, including continuous-time dynamic graphs and irregularly sampled D-TDGs for future work.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: The paper provides a set of theorems that justify the choice of the developed model, as well as its theoretical properties. The corresponding proofs are presented in the appendices.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: Section 2 and Appendix B describe all the details on the methodology and its practical implementation details. All details for the SURF-WAVE task are reported in Appendix D.1. All the other experiments are based on open-source datasets available online. The setup of each experiment is described in Section 4 and Appendix D. Hyperparameters along with best-performing configurations are reported in Appendix D.3.

Guidelines:

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

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: We provide all the details to implement our method in Section 2 and Appendix B, as well as describing the setup of each experiment in Section 4 and Appendix D, providing sufficient information to reproduce our experiments. Moreover, all of our experiments fully rely on open-source datasets available online (except for the SURF-WAVE task, whose details are reported in Appendix D.1). We openly release the code at `https://github.com/thobotics/STORM`.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We specify that we follow the same training procedure for the datasets as their papers of reference. SURF-WAVE task is detailed in Appendix D.1. We also report hyperparameters along with best-performing configurations in Appendix D.3.

   Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [Yes]

Justification: Results are provided in terms of the average over multiple runs along with the standard deviation to verify the significance of results.

Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We provide detailed information about the used resources and runtimes in the Appendix.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research conducted in this paper conforms in every aspect with the NeurIPS Code of Ethics and our research does not represent a potential harm for the society.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We discuss the broader impacts of our work in Appendix G.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The release of our data and models does not pose a direct risk of misuse.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

Justification: For each dataset and task used in this work, we correctly cite and mention the work introducing it and relevant related studies. Licenses and terms of use are properly respected.

Guidelines:

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

13. **New assets**

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

Answer: [Yes]

Justification: Our new method and dataset are well described throughout our paper. We openly release our code and data upon acceptance.

Guidelines:

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

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

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

Answer: [N/A]

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

Guidelines:

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

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

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

Answer: [N/A]

Justification: No crowdsourcing or human subjects were involved in the experiments conducted for this paper.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: The core method development in our research does not involve LLMs as any important, original or non-standard components.

Guidelines:

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