Beyond Oversquashing: Understanding Signal Propagation in GNNs Via Observables

Eden Nagar
Technion - Israel Institute of Technology
Faculty of Mathematics

Ya-Wei Eileen Lin
Technical University of Munich
Munich Center for Machine Learning

Ron Levie
Technion - Israel Institute of Technology
Faculty of Mathematics

Abstract

Graph Neural Networks (GNNs) perform computations on graphs by routing the signal between graph regions using a graph shift operator or a message passing scheme. Often, the propagation of the signal leads to a loss of information, where the signal tends to diffuse across the graph instead of being deliberately routed between regions of interest. Two notions that depict this phenomenon are oversmoothing and oversquashing. In this paper, we propose an alternative approach for modeling signal propagation, inspired by quantum mechanics, using the notion of observables. Specifically, we model the place in the graph where the signal lies, how much the signal is concentrated there, and how much of the signal is propagated towards a location of interest when applying a GNN. Using these new concepts, we prove that standard spectral GNNs have poor signal propagation capabilities. We then propose a new type of spectral GNN, termed *Schrödinger GNN*, which we show has a superior capacity to route the signal across the graph.

1 Introduction

Graph Neural Networks (GNNs) [1, 2] have emerged as powerful tools, enabling breakthrough applications across diverse domains [3, 4], including molecular science [5–7], physics simulations [8, 9], social network analysis [10, 11], and recommendation systems [12, 13]. A GNN is a layered architecture that takes a graph with node features, often referred to as the signal, and returns some output, e.g., another signal on the same graph. The hidden states of the signal across the layers can be interpreted as a gradual flow or propagation of the node features, since the GNN computes the signal at the next layer using local operations on the previous layer.

Often, to solve a problem on graphs, the GNN should be able to direct the propagation of the signal from certain regions of the graph to others. For example, the function of an enzyme is often understood through the notion of allosteric regulation [14, 15]: activation in one site of the enzyme (the receptor) changes the dynamics of the molecule, leading to some change in another site, called the active site. To be able to predict such a behavior using a GNN, the GNN should be able to propagate the signal about the binding site, which captures structural properties of the receptor, to the distant active site. However, one limitation of typical GNNs is that the signal gets diffused in all directions the more layers are used in the network, rather than being propagated, or routed, in a coherent way between regions in the graph. This limits the applicability of typical GNNs when a deliberate routing of the signal is required to solve the task. Two standard notions that are commonly regarded as quantifying this phenomenon are *oversmoothing* [16–21] and *oversquashing* [22–29]

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

However, the first notion, oversmoothing, which is often quantified via the Dirichlet energy [30, 31], describes how quickly the signal varies, or oscillates, across the whole graph, not how much the signal can be kept concentrated, or coherent, when propagating it from one region to another. The second phenomenon, oversquashing, describes the phenomenon where long range information is compressed through topological bottlenecks. Hence, analyses of oversquashing are typically based on various definitions quantifying bottlenecks, e.g., Cheeger constant, graph curvature, and effective resistance [23, 24, 32]. In the same papers, oversquashing is quantified via the magnitudes of the partial derivatives of the GNN, at node m with respect to node n, for all m, n. Namely, large derivatives indicate that all edges affect all other edges significantly enough. The magnitudes of these derivatives are typically linked to quantities of bottlenecks via special inequalities. Rewiring algorithms are designed to maximize the overall flow along the graph (i.e., to minimize topological bottlenecks), and hence, by the special inequalities, to also increase the GNNs derivatives.

Our contribution. Here, we identify the following limitation in oversquashing analyses. As noted above, often to solve the task, specific nodes should send their information to other specific nodes. However, by focusing on the graph structure and trying to increase the communication between all pairs of nodes via rewiring, regardless of the task, the signal still diffuses across the graph by the GNN. Increasing all pairwise communications does not promote the ability of the GNN to single out specific pairs of locations of interest on the graph, allowing a coherent transmission of information between them, while having small GNN gradients for the rest of the nodes.

Our goal is to analyze information flow in GNNs beyond oversquashing. We aim to directly study how coherent the signal stays when it is routed between regions of the graph. For this, we propose an alternative way to model and probe different aspects of the content of the signal and its flow. Specifically, we model (i) the location in the graph where the signal lies, (ii) how much the content of the signal is concentrated about this location, and, (iii) how much of the signal is propagated from one location of the graph to another when applying a GNN. These three concepts are defined via the notion of observables and their mean and variance, similarly to the approach in quantum mechanics. Our focus is not trying to amplify *all* pairwise node communications, but rather to study the ability of GNNs to route the signal from each location to a respective *specific destination*. Here, a good flow of information is characterized by the ability of the GNN to choose these destinations in view of the task. We note that measuring signal content using observables was also investigated in the signal processing literature [33–36].

We then prove that standard spectral GNNs have poor signal propagation capabilities: they keep the location of the content of the signal unchanged, and only increase the spread of the signal about this location. We hence propose a novel spectral GNN, called *Schrödinger GNN*, which has provably good signal flow properties. Namely, with *Schrödinger filters*, we can direct the propagation of the signal in any desired direction in the graph.

Schrödinger GNNs are based on two main components: a unitary graph shift operator (GSO), and complex modulated signals. The unitary GSO is analogous to the Schrödinger operator in classical quantum mechanics, and specifically, in the free particle dynamics. It assures that the content of the signal is transformed in a geometry preserving way, rather than being diffused. Moreover, Schrödinger GNNs consider some of the input feature channels as encoding an abstract notion of ambient location in the graph. We call these features *formal location*. The rest of the feature channels are called *the signal*. The idea is to be able to shift the signal across the formal location, in any desired direction. For illustration, in a social network, we might want to shift the *income* signal along the *age* direction, to allow comparing salaries of different age groups. To quantify the propagation properties of signals, we consider an observable corresponding to each formal location feature, namely, an operator that measures the formal location of signals. Moreover, to guarantee that the formal location of signals shifts when applying GNNs, we form in the signal complex oscillations along the direction of each formal location. We show that this leads roughly to a constant speed of the formal location of signals when applying linear Schrödinger filters.

We empirically validate our theory on graph classification and regression benchmarks.

2 Measuring signal localization and propagation

We now introduce the quantities used throughout the paper to measure where signals are localized.

2.1 General notations

For $N \in \mathbb{N}$ we denote $[N] = \{1, \ldots, N\}$. We treat a vector $a = (a_1, \ldots, a_n) \in \mathbb{C}^N$ as functions $a : [N] \to \mathbb{C}$, where $a(n) = a_n$ for $n \in [N]$. For $z \in \mathbb{C}$, we denote complex conjugation by $\overline{z}$, and real and imaginary parts by $\mathrm{Re}(z)$ and $\mathrm{Im}(z)$.

Consider a graph $\mathcal{G} = (V, E)$ with the vertex set $V = [N]$ and the edge set $E \subset [N]^2$. We denote by $\mathcal{N}(v)$ the neighborhood of vertex $v \in V$. We consider only undirected weighted graphs, and denote the adjacency matrix by $A = (a_{n,m})_{n,m} \in \mathbb{R}^{N \times N}$, where $A = A^\top$ and $a_{n,m} = 0$ whenever $\{n, m\} \notin E$. A *graph-signal* is a pair $(\mathcal{G}, f)$ where $\mathcal{G}$ is a graph and $f = (f_k(n))_{k \in [K], n \in [N]} \in \mathbb{R}^{N \times K}$ is a 2D array called a *signal*, with each $f_k \in \mathbb{R}^N$. Here, we see f also as a mapping $f : V \to \mathbb{C}^K$, and see each f_k as a mapping $f_k : V \to \mathbb{C}$ called a *channel*. By convention, nodes or locations are substituted using brackets $f(n) \in \mathbb{R}^K$ and channels are written with subscript $f_k \in \mathbb{R}^N$. Accordingly, the value of channel number k of f at node n is denoted by $f_k(n)$. A graph shift operator (GSO) is any operator that encodes the graph's structure, e.g., the adjacency matrix or any graph Laplacian. The inner product of two signals $f, g \in \mathbb{C}^{N \times 1}$ is $\langle f, g \rangle = \sum_{v \in V} f(v)\overline{g(v)}$, and norm of f is $\|f\|_2^2 = \langle f, f \rangle$. For a matrix $\boldsymbol{M}$, its operator norm is $\|\boldsymbol{M}\|_{\mathrm{op}} = \sup_{\|x\|_2 = 1} \|\boldsymbol{M}x\|_2$. For $f \in \mathbb{C}^N$, we denote by $\mathrm{diag}(f)$ the diagonal matrix with diagonal elements $\mathrm{diag}(f)_{n,n} = f_n$. The commutator of two square matrices is $[\boldsymbol{M}, \boldsymbol{Y}] = \boldsymbol{M}\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{M}$.

2.2 Observables and the signal routing measure

In a general Hilbert space $\mathcal{H}$ of signals, an *observable* is a self-adjoint operator M in $\mathcal{H}$, i.e., $M^* = M$. By the spectral theorem, any self-adjoint operator in a finite dimensional space can be written as $M = \sum_j \lambda_j P_j$ where $\{\lambda_j\}_j$ are real eigenvalues and $\{P_j\}_j$ are the orthogonal eigenprojections. This decomposition motivates treating a self-adjoint operator as an *observable* of a *physical quantity* [37]. Namely, we interpret the eigenvalues as values that the physical quantity can attain, and P_j as projections upon spaces of signals that have λ_j as the value of their physical quantity. For example, the diagonal operator $D : \mathbb{C}^N \to \mathbb{C}^N$ defined by $(Dg)(j) = jg(j)$ can be thought of as a *location observable* on the discrete line $[N]$. Here, the eigenvectors $e_j = (0, \ldots, 0, 1, 0, \ldots, 0)$ (with 1 only at the j-th entry) are thought of as pure states (or pure signal) with location exactly $\lambda_j = j$. Any signal $g \in \mathbb{C}^N$ is a linear combination of the *pure location states* $\{e_j\}_j$, i.e., $g = \sum_j g(j)e_j$ with $g(j) \in \mathbb{C}$. When the state g is normalized to $\|g\|_2 = 1$, we can interpret $|g(j)|^2$ as the weight, or probability, of g being at location j. While g does not have one exact location, we can define its *mean location* as $\mathcal{E}_D(g) = \sum_j |g(j)|^2 j$, and its location variance as $\mathcal{V}_D(g) = \sum_j |g(j)|^2 (j - \mathcal{E}_D(g))^2$. Using operator notations, these two quantities can be written as $\mathcal{E}_D(g) = \langle Dg, g \rangle$ and $\mathcal{V}_D(g) = \|(D - \mathcal{E}_D(g)I)g\|_2^2$, where I is the identity operator in $\mathbb{C}^N$.

This discussion motivates the general construction of observables in quantum mechanics. For a self-adjoint operator M and normalized state $g \in \mathcal{H}$, the *expected value* (or *mean*) of M with respect to g is defined to be $\mathcal{E}_M(g) := \langle Mg, g \rangle$. Note that when $\mathcal{H} = \mathbb{C}^N$, we have $\mathcal{E}_M(g) = \sum_i \lambda_i \langle P_i g, g \rangle$, which is interpreted similarly to the above example of location observable. The *variance* of M with respect to g is defined to be

$$\mathcal{V}_M(g) := \|(M - \mathcal{E}_M(g)I)g\|_2^2 = \left\langle (M - \mathcal{E}_M(g)I)^2 g, g \right\rangle = \mathcal{E}_{M^2}(g) - \mathcal{E}_M(g)^2. \tag{1}$$

In addition to the classical notions of mean and variance, we propose quantifying how well a signal is transmitted towards a target value of the physical quantity. Consider a scenario where we have an initial signal $g^{(0)}$, and we would like to transmit this signal to be concentrated about some value $r \in \mathbb{R}$ with respect to some observable M. For that, suppose that we operate on $g^{(0)}$, e.g., with a GNN, and transform it to $g^{(t)}$. The following definition quantifies how well $g^{(t)}$ achieves this goal.

Definition 1 (Signal routing measure)**.** *For an observable M, normalized* initial *signal $g^{(0)}$ and* final *signal $g^{(t)}$, and a target value $r \in \mathbb{R}$, the* signal routing measure *is defined to be*

$$\mathcal{P}_M(g^{(0)}, g^{(t)}, r) = \frac{\langle (M - Ir)^2 g^{(t)}, g^{(t)} \rangle}{\mathcal{V}_M(g^{(0)})}. \tag{2}$$

In the setting of Definition 1, the observable M represents a physical quantity of interest. The term $\langle (M - Ir)^2 g^{(t)}, g^{(t)} \rangle$ quantifies how much the values of the physical quantity of $g^{(t)}$ are concentrated

about r, and the denominator normalizes this with respect to how well the physical quantity of the initial state $g^{(0)}$ is concentrated. It is easy to verify the identity

$$\mathcal{P}_M(g^{(0)}, g^{(t)}, r) = \frac{\mathcal{V}_M(g^{(t)}) + (r - \mathcal{E}_M(g^{(t)}))^2}{\mathcal{V}_M(g^{(0)})}. \tag{3}$$

Hence, to minimize the routing measure, one should construct an operation that transforms $g^{(0)}$ to some $g^{(t)}$, keeping the variance of $g^{(t)}$ small (relatively to the variance of $g^{(0)}$), while making the expected value of $g^{(0)}$ as close as possible to the target value r.

3 Signal propagation in Schrödinger graph signal processing

Next, we introduce Schrödinger graph signal processing, and analyze signal propagation under it.

3.1 Feature location observables

Consider a graph-signal $(\mathcal{G}, q)$ with $q = (q_1, \ldots, q_M) : V \to \mathbb{C}^M$. We treat some of the channels of q as the signal and some as some abstract notion of locations. Namely, for some $1 < J < M$ we call $g = (q_1, \ldots, q_J)$ the *signal*, and call $f = (q_{J+1}, \ldots, q_M)$ the *feature locations*. Denote $K = M - J$ and $f = (f_1, \ldots, f_K)$. As we show later, working with complex-valued signals is important for routing signals between graph regions. Hence, we consider $g : V \to \mathbb{C}^J$ with $\|g_j\|_2 = 1$, and consider real-valued feature locations $f : V \to \mathbb{R}^K$, which need not be normalized. Define the *feature location observables* $X_{f_k} = \text{diag}(f_k)$, for $k \in [K]$. By the fact that f_k is real-valued, X_{f_k} is self-adjoint. Now, $\mathcal{E}_{X_{f_k}}(g_j) = \sum_{n \in [N]} f_k(n) \left|g_j(n)\right|^2$ is interpreted as the f_k-value about which the energy of g_j is centered, and $\mathcal{V}_{X_{f_k}}(g_j)$ is the spread of the energy of g_j about this center.

3.2 Partial derivatives and the second order feature derivative GSO

Our construction uses a special definition of a GSO based on derivatives.

Definition 2 (f_k-partial derivative)**.** *Given a feature location $f_k : V \to \mathbb{R}$, we define the* f_k-partial derivative $\nabla_{f_k} \in \mathbb{C}^{N \times N}$ *as follows. For $n, m \in V$*

$$(\nabla_{f_k})_{n,m} = a_{n,m}(f_k(n) - f_k(m)). \tag{4}$$

Note that ∇_{f_k} is skew-symmetric (i.e., $\nabla_{f_k}^* = -\nabla_{f_k}$), and hence $\nabla_{f_k}^2$ is self-adjoint.

Definition 3 (Schrödinger Laplacian)**.** *Given K feature locations $f = (f_1, \ldots, f_K)$, the corresponding* Schrödinger Laplacian *is defined to be*

$$\Delta_f = -\sum_{k \in [K]} \nabla_{f_k}^2. \tag{5}$$

The Schrödinger Laplacian is self-adjoint as a sum of bounded self-adjoint operators. This makes the following operator unitary.

Definition 4 (Schrödinger operator)**.** *Given feature locations $f : V \to \mathbb{R}^K$ and* time $t \in \mathbb{R}$*, the corresponding* Schrödinger operator *is defined to be $\mathcal{S}[t, f] = e^{-it\Delta_f}$.*

As we define later in the section, *Schrödinger graph signal processing* is based on filtering signals using Schrödinger operators as GSOs. In this paper, we develop the theory for Schrödinger operators based on Schrödinger Laplacians, as these special GSOs lead to theoretical guarantees. However, the Schrödinger signal processing methodology can be extended to be based on general GSOs, like standard graph Laplacians, though with less theoretical guarantees.

Let us draw an analogy to the classical theory. In the free particle Schrödinger equation, we consider the space $\mathbb{R}^3$ as the "graph," consider the coordinates x, y, z as the locations, and $\partial_x, \partial_y, \partial_z$ as the partial derivatives. Here, $\Delta_{x,y,z} = -\partial_x^2 - \partial_y^2 - \partial_z^2$ is the classical Laplace operator. Given a wave function $g^{(0)} : \mathbb{R}^3 \to \mathbb{C}$ representing a particle at time 0, $g^{(t)} = \mathcal{S}[t, (x, y, z)]g^{(0)}$ is the particle at time t. In our case, given a signal $g^{(0)}$ on the graph, thought of as the state at time 0, we denote $g^{(t)} = \mathcal{S}[t, f]g^{(0)}$, and interpret $g^{(t)}$ as the signal at time t.

3.3 Analyzing signal propagation via splitting

Note that typical signals are not localized about one feature location. For example, the grayscale signal of an image is typically supported across all x, y locations. Hence, the expected location and location variance are not meaningful localization notions for such signals (see Figure 1 for illustration). Still, we can conceptually apply a localization analysis with observables as follows. We decompose the signal g into a sum of chunks $g = g^1, \ldots, g^L$, e.g., by multiplying the signal by windows in the formal locations $g^l = w^l(f_k)g$, where $w^1, \ldots, w^L : \mathbb{R} \to \mathbb{R}$ form a partition of unity. Here we assume that each w^l is well localized about one location value. Then, each chunk g^l has a meaningful mean location, and we can track how Schrödinger operators propagate this location. Moreover, by tracking how much the Schrödinger operator increases the variance of the chunk, we interpret how much the content of the signal in this chunk is diffused, scatters, or dispersed, when propagating it. Note that this analysis makes sense by the linearity of the Schrödinger operator. In our methodology, we do not decompose g to chunks when applying a GNN. Rather, this decomposition is for conceptualizing the signal propagation, and also for diagnosis of trained GNNs (see Section 4).

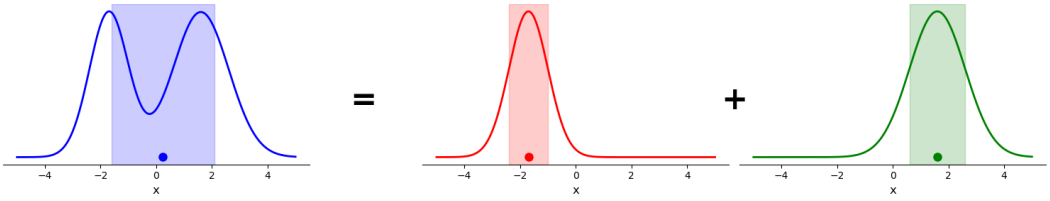

Figure 1: Windowed localization of a graph signal. Decomposition of a signal g to $g^0 + g^1$. The horizontal axis is the real feature-location coordinate. Expected feature locations are marked by a dot, and the variance is signified by a color band.

3.4 Dynamics of 1D signals via feature momentum

In the classical theory, the partial derivatives are called the *momentum observables*, and the mean $\mathcal{E}_{i\partial_x}(g)$ is interpreted as the expected momentum, or speed, of particle g. Analogously, we interpret the f_k-partial derivative $i\nabla_{f_k}$ as observables of momentum or velocity along f_k. This interpretation is made precise by developing dynamical equations of signals under the Schrödinger operator. In the following discussion, we focus on a single-channel signal $g = g_1$ and a single feature location $f = f_1$. We first show that the expected momentum of a signal is constant under Schrödinger dynamics.

Theorem 5 (Constant expected momentum)**.** *Let $g^{(0)} : V \to \mathbb{C}$ be a normalized signal and $f : V \to \mathbb{R}$ a feature location. Then, for every $t \in \mathbb{R}$,*

$$\mathcal{E}_{i\nabla_f}(g^{(t)}) = \mathcal{E}_{i\nabla_f}(g^{(0)}). \tag{6}$$

We then show that the rate of change of the expected location is equal to some smoothed version of the expected momentum. For that, we first define smoothing with respect to feature directions.

Definition 6 (f-smoothing operator)**.** *Let f be a feature location. The f-smoothing operator $W_f : \mathbb{C}^N \to \mathbb{C}^N$ is defined for every signal $g \in \mathbb{C}^N$ and vertex $v \in V$ by*

$$(W_f g)(v) = \sum_{w \in \mathcal{N}(v)} a_{v,w}(f(w) - f(v))^2 g(w). \tag{7}$$

By definition, the f-smoothing operator mixes the values of the signal g only along edges where the feature f changes. It is hence interpreted as smoothing along the f direction.

Theorem 7 (Expected feature location derivative under Schrödinger dynamics)**.** *Let $g^{(0)} : V \to \mathbb{C}$ be a normalized signal and $f : V \to \mathbb{R}$ a feature location. Let $g^{(t)} = \mathcal{S}[t, f]g^{(0)}$ for every $t \in \mathbb{R}$. Then,*

$$\frac{\partial}{\partial t}\mathcal{E}_{X_f}(g^{(t)}) = 2\mathrm{Re}\left(\langle i\nabla_f g^{(t)}, W_f g^{(t)}\rangle\right). \tag{8}$$

The right-hand-side of Eq. (8) is interpreted as a smoothed version of the expected momentum $\mathcal{E}_{i\nabla_f}(g^{(t)}) = \langle i\nabla_f g^{(t)}, g^{(t)}\rangle$. Hence, Theorem 7 states that the rate of change of the expected location is equal to a smoothed expected momentum. In Appendix C.3, we show that for smooth

enough signals, the rate of change of the expected location is close to the exact expected momentum. Since the expected momentum is constant, the theorem suggests that the rate of change of the expected location is roughly constant, as long as the signal stays smooth enough. This analysis hence justifies calling $i\nabla_f$ the momentum, or velocity, observable.

Theorem 7 is analogous to the classical case, where the rate of change of the expected location of a free particle is equal to its expected momentum, which is constant. See Appendix D for more details. An equation for the dynamics of the variance is given in Appendix C.4.

3.5 Achieving translations via feature modulation

We wish to be able to translate the expected feature location of signals using Schrödinger operators. In typical graph data, all features are real. However, as we show next, for real value signals, the expected momentum is always zero.

Example 8. *Let g be a real-valued signal and f a (real) feature location. The corresponding expected momentum is $\mathcal{E}_{i\nabla_f}(g) = \langle i\nabla_f g, g\rangle = i\langle \nabla_f g, g\rangle$. Since f and g are real valued, so is $\langle \nabla_f g, g\rangle$. Hence, $\mathcal{E}_{i\nabla_f}(g)$ is imaginary. However, it must also be real as an expected value of a self-adjoint observable. Indeed, $\langle i\nabla_f g, g\rangle = \langle g, i\nabla_f g\rangle = \overline{\langle i\nabla_f g, g\rangle}$. Hence, we must have $\mathcal{E}_{i\nabla_f}(g) = 0$.*

Hence, given a real-valued signal, to be able to route it between feature regions, we must first modify it to be complex-valued. We do this via the feature modulation operator.

Definition 9 (Feature modulation)**.** *Given a feature location $h : V \to \mathbb{R}$ and a* phase *$\theta \in \mathbb{R}$, the corresponding* feature modulation operator *is defined to be $D[\theta h] := \mathrm{diag}(e^{i\theta h})$, where $e^{i\theta h}$ is the vector with entry $(e^{i\theta h})(v) = e^{i\theta h(v)}$ for each node $v \in V$.*

Next, we show that modulating a real-valued signal gives nonzero expected momentum in general.

Theorem 10 (Expected momentum of modulated signal)**.** *Given a normalized signal $g : V \to \mathbb{R}$, feature locations $f, h : V \to \mathbb{R}$, and a phase $\theta \in \mathbb{R}$, the expected momentum of $D[\theta h]g$ satisfies*

$$\mathcal{E}_{i\nabla_f}(D[\theta h]g) = \sum_{(m,n)\in E} a_{m,n} g(m)g(n)(f(n) - f(m))\sin(\theta(h(n) - h(m))). \tag{9}$$

Theorem 10 can be interpreted as follows. Consider the edge signals $e_{g,h}, e_f : E \to \mathbb{R}$ defined by

$$e_{g,h}(v,w) = g(v)g(w)\sin\big(\theta(h(w) - h(v))\big), \quad e_f(v,w) = f(v) - f(w). \tag{10}$$

The right-hand-side of Eq. (9) is the edge-space inner product $\langle e_{g,h}, e_f\rangle$. Hence, as long as we choose a modulating feature h such that $e_{g,h}$ and e_f are not orthogonal, the expected momentum of $D[\theta h]g$ will be nonzero.

3.6 Dynamics of multi-channel signals and observables

We next derive the dynamics under the Schrödinger operator for multidimensional locations.

Theorem 11 (Expected multi-feature derivative)**.** *Given the Schrödinger Laplacian $\Delta_f = -\sum_{j\in[K]} \nabla^2_{f_j}$, a normalized signal $g^{(0)}$, and $g^{(t)} = \mathcal{S}[t, f]g^{(0)}$, for every $k \in [K]$ we have*

$$\frac{\partial}{\partial t}\mathcal{E}_{X_{f_k}}(g^{(t)}) = 2\mathrm{Re}\left(\langle i\nabla_{f_k} g^{(t)}, W_{f_k} g^{(t)}\rangle\right) - \sum_{j\neq k}\left\langle [i\nabla^2_{f_j}, X_{f_k}]g^{(t)}, g^{(t)}\right\rangle. \tag{11}$$

Ideally, we would like the rate of change of the expected X_{f_k} location to be a smoothed version of the expected $i\nabla_{f_k}$ momentum. However, we see that in Eq. (11) there are additional cross terms. This leads to the following definition.

Definition 12 (ϵ-commuting features)**.** *The feature locations $(f_1, f_2, \ldots, f_K)$ are called* ϵ-commuting *if for every pair $i \neq j \in [K]$, the matrix $E = ([\nabla^2_{f_j}, X_{f_i}])_{i,j\in[K]}$ satisfies $\|E\|_{\mathrm{op}} \leq \epsilon$.*

For a sequence of ϵ commuting features, the Schrödinger dynamics satisfies

$$\left|\frac{\partial}{\partial t}\mathcal{E}_{X_{f_k}}(g^{(t)}) - 2\mathrm{Re}\big(\langle i\nabla_f g^{(t)}, W_f g^{(t)}\rangle\big)\right| \leq (K-1)\epsilon. \tag{12}$$

Hence, here as well we have the interpretation that for smooth enough signals, the rate of change of all expected locations is close to their corresponding expected momenta. see Theorem 20 for details.

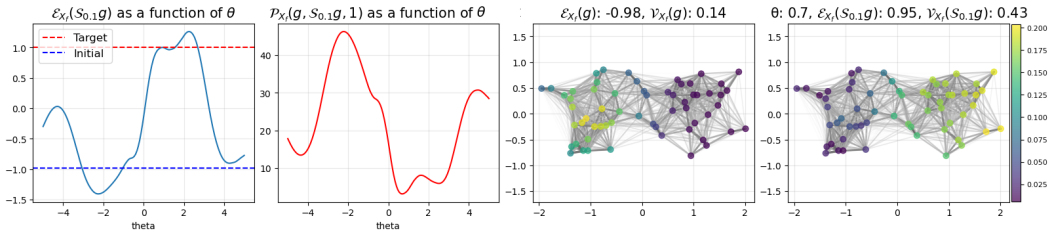

Figure 2: Signal transport under modulation for the phase-induced signal transport. The routing measure is represented as a function of modulation phase, and the expected feature location as a function of modulation phase. We next plot the initial node energy and the propagated energy after applying the selected phase.

3.7 Orthogonalizing the feature directions

The signal $q : V \to \mathbb{R}^M$ in the raw data is not ϵ-commuting in general. Hence, in Schrödinger GNNs, as a first step, we transform the feature q to a sequence of features $f_1, \ldots, f_K$ which are ϵ-commuting. For example, one can plug each node feature $q(n)$ into a simple MLP or a linear transformation Θ, to obtain $f(n) = \Theta(q(n))$. The transformation Θ is optimized with respect to the following target.

Definition 13 (Position-momentum optimization (PMO))**.** *Given a signal $q \in \mathbb{R}^{N\times M}$ and $\lambda > 0$, the linear transformation $T \in \mathbb{R}^{M\times K}$, mapping q to $f = qT \in \mathbb{R}^{N\times K}$, is optimized with respect to*

$$\min_{T\in\mathbb{R}^{M\times K}} \sum_{i\neq j}^{K} \|[\nabla^2_{f_j}, X_{f_i}]\|^2_{\text{op}} + \lambda \sum_{k=1}^{K} \left(\|\nabla_{f_k}\|_\infty - 1\right)^2. \tag{13}$$

3.8 Improving signal routing through modulation

In typical situations, if the goal is to transmit the real-valued signal $g^{(0)}$ towards a feature location r, modulating the signal helps. Namely, given feature locations h, f and time t, there is a nonzero choice of the modulation parameter θ such that the signal routing measure $\mathcal{P}_{X_f}(g^{(0)}, \mathcal{S}[t, f]D[\theta h]g^{(0)}, r)$ is lower than $\mathcal{P}_{X_f}(g^{(0)}, \mathcal{S}[t, f]g^{(0)}, r)$. We propose an analysis supporting this in Appendix C.7. Hence, since we assume that in order to solve the task there are special pairs of graph regions than need to communicate, modulations are an important ingredient that promote such long term interactions when propagating the signal via Schrödinger operators.

In Figure 2 we give an example of a graph, initial signal $g^{(0)}$ with $\mathcal{E}_{X_f}(g) = -0.98$, modulating feature $h = f$, and desired location value $r = 1$. We show that choosing an appropriate modulation θ and propagating the signal using the Schrödinger operator to time $t = 0.1$ improves the signal routing measure with respect to not modulating. For further details of this example, refer to Appendix E.2.

3.9 Schrödinger signal processing and GNNs

Schrödinger signal processing. We define Schrödinger filters by considering linear combinations of the evolutions of the modulated signal with different modulations and times. Let $f : V \to \mathbb{R}^K$ be location features and $D \in \mathbb{N}$ be the output feature dimension. To use linear algebra notations, let us now treat signals and location features as vectors in $\mathbb{C}^{N\times J}$ and $\mathbb{R}^{N\times K}$ respectively. A Schrödinger filter Ψ is parameterized by $(t_m \in \mathbb{R}, \theta_m \in \mathbb{R}, \mathbf{W}^{(m)} \in \mathbb{C}^{J\times D}, \mathbf{T}^{(m)} \in \mathbb{R}^{K\times 1})_{m\in[M]}$, and maps signals $g \in \mathbb{C}^{N\times J}$ to

$$\Psi(g) = \sum_{m=1}^{M} \mathcal{S}[t_m, f]D[\theta_m f \cdot \mathbf{T}^{(m)}]g \cdot \mathbf{W}^{(m)}. \tag{14}$$

Schrödinger GNNs. The application of a Schrödinger GNN is a two-step procedure. First, the input signal is optimized via Position-Momentum Optimization (PMO) (Definition 13) to obtain the location features f. Then a layered architecture with Schrödinger filters and activation functions is applied on the locations features and the input signal. For nonlinear activations, we apply standard activations



Figure 3: Signal propagation on a ring graph. From left to right: the noisy input Gaussian signal, the target signal obtained by translating the Gaussian by a fixed angular displacement, and the outputs predicted by GCN, GAT, the real-valued Schrödinger GNN ablation, and the full Schrödinger GNN. Only the full Schrödinger GNN accurately transports the input signal to the target location.

(e.g., ReLU) separately to the real and imaginary parts: $\sigma(z) = \mathrm{ReLU}(\mathrm{Re}(z)) + i \cdot \mathrm{ReLU}(\mathrm{Im}(z))$ or we use the absolute value $\sigma(z) = |z|$. See Appendix B for implementation details and for computational complexity analysis.

Sensitivity analysis. In oversquashing analysis, one approach for modeling the sensitivity of the GNN in a given vertex to the signal received from another vertex is via the gradients of the MPNN [23–25]. In Appendix C.8 we conduct a similar analysis for our setting. We focus on a single Schrödinger GNN layer. We first suppose that the modulation feature h, phase θ, and time t were chosen so that each signal chunk is transmitted to a meaningful destination location r (appropriate for the task). Under this assumption, one would like to show that the derivative of the output signal about the location r with respect to the location of the input chunk is not small, as r is seen as the "right" destination for solving the task. We show that this derivative is exactly 1. This shows that all of the signal information reached its destination. In contrast to oversquashing analyses, we do not want high sensitivity of the signal at locations other than the destination r with respect to the input chunk, as transmitting information between such location pairs is not seen as informative for the task.

4 Experiments

We evaluate Schrödinger GNN on a range of graph learning tasks and investigate its signal propagation properties. We first show the signal propagation experiment on a ring graph, followed by evaluations on graph classification, heterophilous node classification, and long-range graph benchmarks. We further examine the learned signal propagation through a window-based analysis of signal flow. Additional experiments in Appendix E include ablation studies of unitary Schrödinger propagation, complex modulation, and PMO, as well as a sensitivity analysis of the learned feature locations. Code available at: https://github.com/edenagar/SGCN.

4.1 Toy example: signal propagation on a ring

We first demonstrate the ability of Schrödinger GNN to direct the propagation of the signal through a regression task on a ring graph. Consider a ring graph discretizing the unit circle, and the location feature $x = \cos(\theta)$, where θ is the angle. Each input signal is a noisy Gaussian on the ring with random mean μ and variance σ^2, and the target is obtained by translating the same Gaussian by a prescribed angular displacement d. The task is to map the input signal to the target signal. Figure 3 compares the predicted signal produced by Schrödinger GNN against GCN [2], GAT [38], and a real-valued Schrödinger GNN. Table 1 reports the corresponding test losses. These results show that only the Schrödinger GNN, with a modulated input signal, accurately learns the prescribed transport.

Table 1: Test loss for the ring signal transport task. The Schrödinger GNN achieves the lowest error.

Model	Test Loss (↓)
GCN	0.6644±0.0720
GAT	0.6050±0.0052
Schrödinger real	0.9334±0.0514
Schrödinger	**3e-04**±2e-04

4.2 Graph classification on TU dataset under architecture-matched evaluation

We next evaluate our approach on graph classification tasks from the TU dataset [39], using ENZYMES, IMDB, MUTAG, and PROTEINS. To isolate the effect of the propagation mechanism from differences in model depth or capacity, we use an architecture-matched evaluation protocol.

We compare Schrödinger GNN with GIN [40], GCN [2], GAT [38], UniGCN [41], adaptive unitary, Schrödinger, and their PMO variants. In this setting, every method is implemented with the same macro-architecture: six graph convolution or message-passing layers followed by the same graph-level readout and a final linear prediction layer. We match the number of trainable parameters across methods. The results are reported in Table 2. We see that our method achieves the best performance on all four datasets, suggesting that the Schrödinger propagation is effective and PMO further improves its performance.

Table 2: Architecture- and parameter-matched graph classification results on TU datasets.

	ENZYMES	IMDB	MUTAG	PROTEINS
GIN	31.93±3.16	69.22±3.14	78.19±5.57	71.88±3.08
GCN	31.66±5.35	50.60±4.10	73.24±6.27	71.41±3.04
GAT	31.13±3.48	49.54±2.54	75.21±6.41	72.31±3.28
UniGCN	40.30±6.63	65.42±2.80	75.74±6.67	69.19±3.01
Adaptive Unitary	41.60±5.18	65.46±2.48	75.53±5.95	71.79±3.33
Adaptive Unitary PMO	41.83±4.44	66.27±3.01	75.62±6.24	71.77±2.84
Schrödinger	43.50±4.89	65.86±2.83	75.42±6.11	71.57±2.56
Schrödinger PMO	**43.70**±3.37	**69.60**±2.85	**79.25**±6.19	**72.68**±3.05

4.3 Heterophilous node classification

We also assess Schrödinger GNN on heterophilous node classification benchmarks from [42], including Roman-empire, Amazon-ratings, Minesweeper, Tolokers, and Questions. The competing baselines include GCN [2], SAGE [10], GAT [38], GT [43], UniGCN [41], and Lie UniGCN. Table 3 shows the classification results. Schrödinger GNN achieves the best performance on three out of five datasets, and on the remaining two, it obtains the second best, indicating that Schrödinger propagation is also well-suited for heterophilous graphs.

Table 3: Heterophilous node classification performance. Results are reported as mean ± standard deviation. Results of competing methods marked with † are taken from [42, 41].

	Roman-Empire	Amazon-Rating	Minesweeper	Tolokers	Questions
	Test AP ↑	Test AP ↑	ROC AUC ↑	ROC AUC ↑	ROC AUC ↑
GCN†	73.69±0.74	48.70±0.63	89.75±0.52	83.64±0.67	76.09±1.27
SAGE†	85.74±0.67	53.63±0.39	93.51±0.57	82.43±0.44	76.44±0.62
GAT†	80.87±0.30	49.09±0.63	92.01±0.68	83.70±0.47	77.43±1.20
GT†	86.51±0.73	51.17±0.66	91.85±0.76	83.23±0.64	77.95±0.68
UniGCN†	87.21±0.76	**55.34±0.74**	94.27±0.58	84.83±0.68	79.21±0.79
Lie UniGCN†	85.50±0.22	52.35±0.26	96.11±0.10	**85.18±0.43**	80.01±0.43
Schrödinger	**88.56±0.71**	54.26±0.40	**96.31±0.49**	85.09±0.17	**80.14±0.20**

4.4 Long range graph benchmark

Finally, we evaluate our approach on the long range graph benchmark (LRGB) [44]. We consider four tasks: (i) Peptides-Func, a graph classification task, (ii) Peptides-Struct, a graph regression task, and (iii) PascalVOC-SP and (iv) COCO-SP, which formulate semantic image segmentation as a node-classification task on superpixel graphs. These benchmarks are designed to test whether a model can capture long-range dependencies beyond local neighborhood aggregation. We compare our method with GCN [2], GINE [40], GatedGCN [45], GUMP [46], GPS [47], DRew [48], Exphormer [49], GRIT [50], Graph ViT [51], CRAWL [52], UniGCN [41], and Lie UniGCN. The results are reported in Table 4. Schrödinger GNN performs strongly across LRGB tasks, where it outperforms all the baselines on the Peptides-Func dataset, and it achieves the second-best among the other three tasks. This suggests that Schrödinger propagation is effective at learning long-range signals.

4.5 Diagnosis of signal flow with windows

Table 4: Comparison of Schrödinger GNN with competing GNN architectures on the LRGB dataset. Results of competing methods marked with † are taken from [53, 41].

	Peptides-Func	Peptides-Struct	PascalVOC-SP	COCO-SP
	Test AP ↑	Test MAE ↓	Test F1 ↑	Test F1 ↑
GCN†	0.6860±0.0050	0.2460±0.0007	0.1338±0.0007	0.2078±0.0031
GINE†	0.6621±0.0067	0.2473±0.0017	0.2125±0.0009	0.2718±0.0054
GatedGCN†	0.6765±0.0047	0.2477±0.0009	0.2922±0.0018	0.3880±0.0040
GUMP†	0.6843±0.0037	0.2564±0.0023	-	-
GPS†	0.6534±0.0091	0.2509±0.0014	**0.3884±0.0055**	0.4440±0.0065
DRew†	0.7150±0.0044	0.2536±0.0015	-	0.3314±0.0024
Exphormer†	0.6527±0.0043	0.2481±0.0007	0.3430±0.0008	0.3960±0.0027
GRIT†	0.6988±0.0082	0.2460±0.0012	-	-
Graph ViT†	0.6942±0.0075	0.2449±0.0016	-	-
CRAWL†	0.7074±0.0032	0.2506±0.0022	-	**0.4588±0.0079**
UniGCN†	0.7072±0.0035	**0.2425±0.0009**	0.2852±0.0016	0.3516±0.0070
Lie UniGCN†	0.7173±0.0061	0.2460±0.0011	0.3153±0.0035	0.4005±0.0067
Schrödinger	**0.7207±0.0099**	0.2439±0.0011	0.3507±0.0019	0.4259±0.0034

To directly diagnose whether a trained model transports information across the graph, we measure whether the trained layer moves the signal along the learned location features. For each graph, we split the node signal into localized windows according to the values of the location features. Each window isolates the part of the signal concentrated around a small region in feature space. We then pass each windowed signal through one trained layer and compare its average location before and after the layer. The shift of the window is measured as the change in its average location along the corresponding feature coordinate. To make this quantity comparable across graphs and features, we normalize the shift by the standard deviation of that location feature. We compute this relative shift for every layer, average it over the PascalVOC-SP dataset [44], and repeat the same process for evaluating our Schrödinger GNN, GCN [2], and unitary GCN [41]. Figure 4 shows the mean relative shift across layers. Our method demonstrates a clear and sustained shift throughout the network. The Schrödinger GNN, with unitary propagation and feature-dependent modulation, enables the network to create nonzero effective momentum and route signal mass across the graph. Thus, it confirms that our model learns layer-wise signal flow.

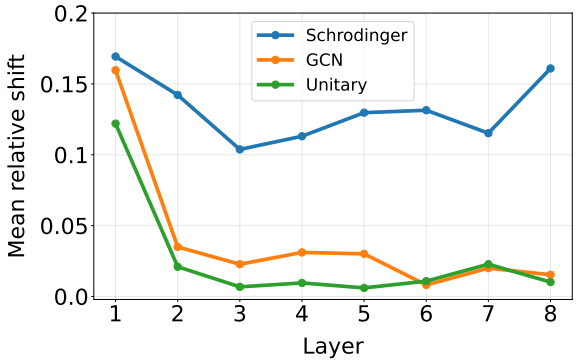

Figure 4: Diagnosis of signal flow on PascalVOC-SP. The Schrödinger GNN maintains a large relative shift across layers, indicating that signal transportation is achieved by the modulation in Schrödinger layers.

5 Summary

We presented a new approach for defining and analyzing signal propagation across graphs. The approach directly models where the information of the signal is, how well concentrated it is, and how well it is routed between regions in the graph. We presented Schrödinger GNN, a graph neural network that is able to route the information of the signal along any direction in the graph. We showed that standard GNNs do not have this capability. One limitation of Schrödinger filters with respect to simple polynomial filters is that applying the Schrödinger operator on a signal involves approximating the exponential of the GSO, which involves applying the GSO several times.

Acknowledgments

RL was supported by a grant from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel, and the United States National Science Foundation (NSF), (NSF-BSF, grant No. 2024660), and by the Israel Science Foundation (ISF grant No. 1937/23). YEL acknowledges funding from the Alexander von Humboldt Foundation, the Munich Center for Machine Learning (MCML), the Schmidt Futures Israeli Women's Postdoctoral Award, and the Viterbi Fellowship from the Faculty of Electrical and Computer Engineering at the Technion.

References

[1] 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.

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

[3] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[4] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

[5] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.

[6] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016.

[7] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *Advances in neural information processing systems*, 30, 2017.

[8] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016.

[9] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pages 8459–8468. PMLR, 2020.

[10] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[11] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[12] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, pages 165–174, 2019.

[13] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 639–648, 2020.

[14] Francho Nerín-Fonz and Zoe Cournia. Machine learning approaches in predicting allosteric sites. *Current Opinion in Structural Biology*, 85:102774, 2024.

[15] Hao Tian, Sian Xiao, Xi Jiang, and Peng Tao. PASSer: fast and accurate prediction of protein allosteric sites. *Nucleic Acids Research*, 51(W1):W427–W431, 2023.

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

[17] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*, 2020.

[18] Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in GNNs. In *International Conference on Learning Representations*, 2020.

[19] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020.

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

[21] Jhony H Giraldo, Konstantinos Skianis, Thierry Bouwmans, and Fragkiskos D Malliaros. On the trade-off between over-smoothing and over-squashing in deep graph neural networks. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, pages 566–576, 2023.

[22] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021.

[23] 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.

[24] Mitchell Black, Zhengchao Wan, Amir Nayyeri, and Yusu Wang. Understanding oversquashing in GNNs through the lens of effective resistance. In *International Conference on Machine Learning*, pages 2528–2547. PMLR, 2023.

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

[26] Alessio Gravina, Moshe Eliasof, Claudio Gallicchio, Davide Bacciu, and Carola-Bibiane Schönlieb. On oversquashing in graph neural networks through the lens of dynamical systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 16906–16914, 2025.

[27] Pradeep Kr Banerjee, Kedar Karhadkar, Yu Guang Wang, Uri Alon, and Guido Montúfar. Oversquashing in GNNs through the lens of information contraction and graph expansion. In *2022 58th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1–8. IEEE, 2022.

[28] Yonatan Sverdlov, Yair Davidson, Nadav Dym, and Tal Amir. Fsw-gnn: A bi-lipschitz wl-equivalent graph neural network. *arXiv preprint arXiv:2410.09118*, 2024.

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

[30] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98, 2013.

[31] Aliaksei Sandryhaila and Jose MF Moura. Discrete signal processing on graphs: Frequency analysis. *IEEE Transactions on signal processing*, 62(12):3042–3054, 2014.

[32] Pradeep Kr Banerjee, Kedar Karhadkar, Yu Guang Wang, Uri Alon, and Guido Montúfar. Oversquashing in gnns through the lens of information contraction and graph expansion. In *2022 58th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1–8. IEEE, 2022.

[33] Ron Levie, H-G Stark, Florian Lieb, and Nir Sochen. Adjoint translation, adjoint observable and uncertainty principles. *Advances in computational mathematics*, 40(3):609–627, 2014.

[34] Ron Levie and Nir Sochen. Uncertainty principles and optimally sparse wavelet transforms. *Applied and Computational Harmonic Analysis*, 48(3):811–867, 2020.

[35] Ron Levie and Nir Sochen. A wavelet plancherel theory with application to multipliers and sparse approximations. *Numerical Functional Analysis and Optimization*, 43(11):1303–1400, 2022.

[36] Simon Halvdansson, Jan-Fredrik Olsen, Nir Sochen, and Ron Levie. Existence of uncertainty minimizers for the continuous wavelet transform. *Mathematische Nachrichten*, 296(3):1156–1172, January 2023.

[37] Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010.

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

[39] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. TUDataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

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

[41] Bobak Kiani, Lukas Fesser, and Melanie Weber. Unitary convolutions for learning on graphs and groups. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[42] Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *International Conference on Learning Representations*, 2023.

[43] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

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

[45] Xavier Bresson and Thomas Laurent. Residual gated graph convnets, 2018.

[46] Haiquan Qiu, Yatao Bian, and Quanming Yao. Graph unitary message passing. *arXiv preprint arXiv:2403.11199*, 2024.

[47] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

[48] 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.

[49] 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.

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

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

[52] Jan Tönshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. Walking out of the weisfeiler leman hierarchy: Graph learning beyond message passing. *arXiv preprint arXiv:2102.08786*, 2021.

[53] Jan Tönshoff, Martin Ritzert, Eran Rosenbluth, and Martin Grohe. Where did the gap go? reassessing the long-range graph benchmark. *Transactions on Machine Learning Research*, 2024.

[54] Ronald Newbold Bracewell and Ronald N Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-hill New York, 1986.

[55] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in Neural Information Processing Systems*, 29, 2016.

[56] Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1):97–109, 2018.

[57] Ya-Wei Eileen Lin, Ronen Talmon, and Ron Levie. Equivariant machine learning on graphs with nonlinear spectral filters. *Advances in Neural Information Processing Systems*, 37:128182–128226, 2024.

[58] Ya-Wei Eileen Lin and Ron Levie. Adaptive canonicalization with application to invariant anisotropic geometric networks. In *The Fourteenth International Conference on Learning Representations*, 2026.

[59] Ritchie Zhao, Yuwei Hu, Jordan Dotzel, Christopher De Sa, and Zhiru Zhang. Building efficient deep neural networks with unitary group convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11303–11312, 2019.

[60] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018.

[61] Caiwen Ding, Siyu Liao, Yanzhi Wang, Zhe Li, Ning Liu, Youwei Zhuo, Chao Wang, Xuehai Qian, Yu Bai, Geng Yuan, et al. Circnn: accelerating and compressing deep neural networks using block-circulant weight matrices. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 395–408, 2017.

[62] Guillaume Verdon, Trevor McCourt, Enxhell Luzhnica, Vikash Singh, Stefan Leichenauer, and Jack Hidary. Quantum graph neural networks. *arXiv preprint arXiv:1909.12264*, 2019.

[63] Cenk Tüysüz, Carla Rieger, Kristiane Novotny, Bilge Demirköz, Daniel Dobos, Karolos Potamianos, Sofia Vallecorsa, Jean-Roch Vlimant, and Richard Forster. Hybrid quantum classical graph neural networks for particle track reconstruction. *Quantum Machine Intelligence*, 3(2):29, 2021.

[64] Edward Farhi and Sam Gutmann. Quantum computation and decision trees. *Physical Review A*, 58(2):915, 1998.

[65] Dorit Aharonov, Andris Ambainis, Julia Kempe, and Umesh Vazirani. Quantum walks on graphs. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 50–59, 2001.

[66] Julia Kempe. Quantum random walks: an introductory overview. *Contemporary Physics*, 44(4):307–327, 2003.

[67] Salvador Elías Venegas-Andraca. Quantum walks: a comprehensive review. *Quantum Information Processing*, 11(5):1015–1106, 2012.

[68] Lu Bai, Yuhang Jiao, Lixin Cui, Luca Rossi, Yue Wang, Philip S Yu, and Edwin R Hancock. Learning graph convolutional networks based on quantum vertex information propagation. *IEEE Transactions on Knowledge and Data Engineering*, 35(2):1747–1760, 2021.

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

[70] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

[71] 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.

[72] Joshua Bassey, Lijun Qian, and Xianfang Li. A survey of complex-valued neural networks. *arXiv preprint arXiv:2101.12249*, 2021.

[73] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[74] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The MNIST database of handwritten digits, 1998.

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

[76] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[77] Khang Nguyen, Nong Minh Hieu, Vinh Duc Nguyen, Nhat Ho, Stanley Osher, and Tan Minh Nguyen. Revisiting over-smoothing and over-squashing using Ollivier-Ricci curvature. In *International Conference on Machine Learning*, pages 25956–25979. PMLR, 2023.

[78] Alessio Gravina, Moshe Eliasof, Claudio Gallicchio, Davide Bacciu, and Carola-Bibiane Schönlieb. On oversquashing in graph neural networks through the lens of dynamical systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 16906–16914, 2025.

[79] Ali Hariri, Alvaro Arroyo, Alessio Gravina, Moshe Eliasof, Carola-Bibiane Schönlieb, Davide Bacciu, Xiaowen Dong, Kamyar Azizzadenesheli, and Pierre Vandergheynst. Return of chebnet: Understanding and improving an overlooked GNN on long range tasks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Appendix

A Extended background and related work **17**

A.1 Spectral graph neural networks 17

A.2 Comparison with related unitary neural architectures 18

A.3 Relation to quantum and quantum-inspired graph neural networks 20

A.4 Sparse signal routing versus dense attention 21

B Computational realization of Schrödinger GNNs **21**

B.1 Approximating the unitary propagation 22

B.2 Layer-level realization of the Schrödinger GNN 22

B.3 Complex input modulation 22

B.4 Position-momentum optimization 23

B.5 Nonlinearities and dropout 23

B.6 Computational complexity 23

B.7 Propagation time initialization 23

C Theoretical results and proofs **23**

C.1 Background in functional calculus 24

C.2 Proof of Theorem 5 24

C.3 Proof of Theorem 7 24

C.4 Dynamics of the variance 26

C.5 Proof of Theorem 10 27

C.6 Proof of Theorem 11 27

C.7 Improving signal routing through modulation 29

C.8 Sensitivity analysis of signal routing via Schrödinger operators 30

D Tutorial: Schrödinger dynamics in classical quantum mechanics **31**

E Experimental details and additional experiments **33**

E.1 Toy experiment: recovering orthogonal directions on a grid 33

E.2 Optimizing signal transport via modulation 33

E.3 Gaussian-translate toy experiment on a ring 34

E.4 MNIST graph classification 35

E.5 Graph classification on TU dataset 35

E.6 Heterophilous node classification 37

E.7 Long range graph benchmark 38

E.8 Ablation study on unitary Schrödinger propagation, complex modulation and PMO 39

E.9 Sensitivity to the learned feature locations 39

E.10 Relation between the core mechanisms, PMO, commutation, transport, and performance 40

A Extended background and related work

A.1 Spectral graph neural networks

The convolution layers in many graph neural architectures can be viewed as instances of spectral graph filtering, where the convolution on a graph is defined through the eigendecomposition of the graph shift operator. Namely, the graph convolution uses the eigenvectors of a graph shift operator as graph Fourier modes. Our method belongs to this broad family of spectral GNN, since the operators used by our method can be interpreted as learnable frequency responses acting on graph signals.

Spectral graph filters. A graph shift operator (GSO) is a self-adjoint matrix that reflects the graph's connectivity, such as a normalized graph Laplacian, an unnormalized Laplacian, or a symmetrized adjacency matrix. Let $\mathcal{L}$ be such a GSO with eigenpairs $\{(\lambda_i, v_i)\}_{i=1}^N$, and denote $\mathcal{L} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top$, where $\boldsymbol{V} = [v_1, \ldots, v_N]$ and $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$. The graph Fourier transform of a node signal $\boldsymbol{X} \in \mathbb{R}^{N\times T}$ is given by $\boldsymbol{V}^\top \boldsymbol{X}$, and the inverse transform is given by multiplication by $\boldsymbol{V}$.

For a T-channel node signal $\boldsymbol{X} \in \mathbb{R}^{N\times T}$ and a matrix-valued frequency response $q : \mathbb{R} \to \mathbb{R}^{d'\times T}$, and the corresponding spectral graph filter is given by

$$q(\mathcal{L})\boldsymbol{X} := \sum_{i=1}^{N} v_i v_i^\top \boldsymbol{X}\, q(\lambda_i)^\top.$$

This applies the graph analog of the convolution theorem [54]: each graph Fourier mode v_i is preserved in the node domain, while the channel features associated with that mode are mixed by the matrix $q(\lambda_i)$ in the spectral domain. Thus, $q(\lambda_i)$ determines how information at graph frequency λ_i is amplified, suppressed, or mixed across channels. In the scalar case, where $T = d' = 1$ and $f : \mathbb{R} \to \mathbb{R}$, the filter reduces to the usual functional-calculus operator

$$f(\mathcal{L})\boldsymbol{X} = \sum_{i=1}^{N} f(\lambda_i)\, v_i v_i^\top \boldsymbol{X} = \boldsymbol{V} f(\Lambda)\boldsymbol{V}^\top \boldsymbol{X}.$$

Hence, a scalar spectral filter acts diagonally in the graph Fourier basis: it multiplies the coefficient of the i-th graph Fourier mode by $f(\lambda_i)$.

Parameterized spectral filters. In a neural network, the frequency response q is usually not fixed and it can be parameterized by learnable weights. A common choice is a polynomial filter $q_\theta(\lambda) = \sum_{k=0}^{K} \Theta_k \lambda^k$, where $\Theta_k \in \mathbb{R}^{d'\times T}$ are trainable channel-mixing matrices. Then, it gives the node-domain expression $q_\theta(\mathcal{L})\boldsymbol{X} = \sum_{k=0}^{K} \mathcal{L}^k \boldsymbol{X}\Theta_k^\top$.

This form avoids explicitly computing the eigendecomposition of $\mathcal{L}$ and also gives a locality interpretation, i.e., if $\mathcal{L}$ is sparse and respects graph connectivity, then $\mathcal{L}^k\boldsymbol{X}$ aggregates information along paths of length up to k. Chebyshev spectral networks [55] use a stable polynomial parameterization based on Chebyshev polynomials, while the graph convolutional network of [2] can be viewed as a first-order spectral approximation with a particular normalization. More expressive spectral families, such as rational or Cayley filters, replace polynomial responses by richer frequency responses while retaining the same spectral-filtering principle [56]. Nonlinear spectral filters [57, 58] were further introduced to be equivariant to graph functional shifts.

Spectral GNN layers. A spectral graph neural network composes spectral filters with pointwise nonlinearities. Given hidden node features $\boldsymbol{H}^{(\ell)} \in \mathbb{R}^{N\times d_\ell}$ at layer ℓ, a spectral GNN layer is given by $\boldsymbol{H}^{(\ell+1)} = \sigma\left(q_{\theta_\ell}(\mathcal{L})\boldsymbol{H}^{(\ell)} + \mathbf{1}b_\ell^\top\right)$, where $q_{\theta_\ell} : \mathbb{R} \to \mathbb{R}^{d_{\ell+1}\times d_\ell}$ is a trainable matrix-valued frequency response, $b_\ell \in \mathbb{R}^{d_{\ell+1}}$ is a bias vector, and σ is applied pointwise. Stacking these layers then produces a nonlinear architecture whose linear components are graph-frequency filters.

This perspective places our method within the broad family of spectral GNNs: the learnable operators can be interpreted as frequency-selective transformations of graph signals, with the GSO determining the graph Fourier basis and the parameterized response determining how each spectral component is transformed. Consequently, the architecture inherits the standard spectral interpretation of graph convolution while allowing flexible channel mixing and nonlinear feature extraction across layers.

A.2 Comparison with related unitary neural architectures

In this section, we position our Schrödinger graph signal processing framework in relation to recent unitary neural architectures, including unitary convolutions for graphs and groups [41], graph unitary message passing [46], and unitary group convolutions [59], and clarify how our approach differs from these methods.

The main distinguishing difference of our method is that it is unitary. Rather, our contribution lies in the formulation of feature observables, feature-derivative Schrödinger dynamics, phase-induced feature momentum, and routing measure. Existing unitary methods mainly use unitarity to preserve norms, stabilize gradients, avoid oversmoothing or oversquashing, or improve efficient channel mixing. Our method uses unitarity as part of a broader physical and geometric framework for measuring and controlling signal localization on graphs.

A.2.1 Comparison with unitary convolutions on graphs and groups

Our Schrödinger graph signal processing framework is related to, but different from, the unitary convolution framework [41] but they use different components, optimize for different notions of signal propagation, and provide different theoretical guarantees.

Common principle: unitary propagation. [41] propose replacing standard graph convolutions by unitary or orthogonal convolutions. Their separable unitary graph convolution is $F_{\mathrm{UniConv}}(\boldsymbol{X}) = \exp(it\boldsymbol{A})\boldsymbol{X}\boldsymbol{U}$ with $\boldsymbol{U}\boldsymbol{U}^\dagger = \boldsymbol{I}$, where $\boldsymbol{A}$ is the graph adjacency matrix, $t \in \mathbb{R}$ is a learnable or tunable propagation time, and $\boldsymbol{U}$ is a unitary channel-mixing matrix. They also introduce a Lie unitary/orthogonal graph convolution: $F_{\mathrm{Lie}}(\boldsymbol{X}) = \exp(g_{\mathrm{conv}})(\boldsymbol{X})$ with $g_{\mathrm{conv}}(\boldsymbol{X}) = \boldsymbol{A}\boldsymbol{X}\boldsymbol{W}$ and $\boldsymbol{W} + \boldsymbol{W}^\dagger = 0$ so that the vectorized generator belongs to a Lie algebra and its exponential is unitary or orthogonal.

Our method also uses unitary evolution, but the unitary operator is not generated directly by the adjacency matrix. Instead, we construct feature-dependent graph derivatives $(\nabla_{f_k})_{n,m} = a_{n,m}(f_k(n) - f_k(m))$, define the Schrödinger Laplacian $\Delta_f = -\sum_{k=1}^{K} \nabla_{f_k}^2$, and propagate signals by $\mathcal{S}[t, f] = \exp(-it\Delta_f)$.

Thus, while both methods use exponentials of structure-dependent generators, our framework builds on a second-order feature-derivative operator built from learned or optimized feature locations.

Main conceptual difference. Unitary convolutions [41] are primarily a stability mechanism for graph and group convolutions. Their goal is to make deep equivariant networks more stable by using norm-preserving and invertible layers. Specifically, their theory focuses on avoiding oversmoothing and avoiding vanishing or exploding gradients.

By contrast, our method analyzes information flow in GNNs beyond oversquashing. It introduces an observable-based framework for measuring, controlling, and routing signal localization. We define feature-location observables $X_{f_k} = \mathrm{diag}(f_k)$ and and track the expected feature location and variance of a signal: $\mathcal{E}_{X_{f_k}}(g) = \langle X_{f_k} g, g\rangle$ and $\mathcal{V}_{X_{f_k}}(g) = \|(X_{f_k} - \mathcal{E}_{X_{f_k}}(g)I)g\|_2^2$. This leads to the signal routing measure $\mathcal{P}_M(g^{(0)}, g^{(t)}, r) = \frac{\langle (M-rI)^2 g^{(t)}, g^{(t)}\rangle}{\mathcal{V}_M(g^{(0)})}$, which explicitly quantifies whether the final signal is concentrated near a target feature value r. This target-directed localization objective is absent from unitary convolutions.

Role of complex phases. In [41], complexification is used to make operators such as $\exp(it\boldsymbol{A})$ unitary and wave-like. In our framework, complex phases serve an additional geometric role: they create momentum in feature space. For a real signal g, the expected feature momentum $mathcalE_{i\nabla_f}(g) = \langle i\nabla_f g, g\rangle$ is zero. We therefore introduce feature modulation $D[\theta h] = \mathrm{diag}(e^{i\theta h})$, which can create nonzero expected momentum. Thus, our method enables directed transport toward a target feature location.

A.2.2 Comparison with graph unitary message passing

Graph unitary message passing (GUMP) [46] proposes to improve graph learning by making the adjacency operator used in message passing unitary. Although both GUMP and our Schrödinger

graph signal processing framework use unitarity, they use it in different places and for different purposes.

GUMP: unitary adjacency for stable message passing. GUMP starts from the observation that standard message passing repeatedly applies powers of a graph adjacency operator: in $\boldsymbol{H}^{(k)} = \sigma(\boldsymbol{A}\boldsymbol{H}^{(k-1)}\boldsymbol{W}_k)$, the dependence of $\boldsymbol{H}^{(L)}$ on the input contains powers of $\boldsymbol{A}$. If the eigenvalues of $\boldsymbol{A}$ have magnitude smaller or larger than one, these powers may decay or grow exponentially with depth. GUMP therefore replaces the standard adjacency matrix by a unitary adjacency matrix $\boldsymbol{U}[\boldsymbol{A}_w]$, giving $\boldsymbol{H}^{(k)} = \sigma\left(\boldsymbol{U}[\boldsymbol{A}_w]\boldsymbol{H}^{(k-1)}\boldsymbol{W}_k\right)$ and $\boldsymbol{U}[\boldsymbol{A}_w]^\dagger\boldsymbol{U}[\boldsymbol{A}_w] = \boldsymbol{I}$. Their main objective is to avoid graph-induced training instability by preventing the graph propagation operator from causing exponential contraction or expansion of the Jacobian.

In contrast, our method does not make the adjacency matrix itself unitary. We keep the graph adjacency as a structural object and construct feature-dependent derivative operators, then define the Schrödinger Laplacian and propagate by the unitary Schrödinger operator.

Therefore, in GUMP, the unitary object is an adjacency-like message-passing matrix. In our method, the unitary object is a Schrödinger evolution generated by a self-adjoint feature-derivative Laplacian.

Graph transformation vs. feature-geometry construction. One technical challenge in GUMP is that a general graph does not naturally have a sparse unitary adjacency matrix with the same support as the original adjacency. GUMP addresses this by transforming the graph. For an undirected graph $G = (V, E)$, it first constructs a directed graph G' by replacing each undirected edge with two directed edges, and then passes to the line graph $G \longmapsto G' \longmapsto L(G')$. The transformed line graph $L(G')$ has $2|E|$ vertices, corresponding to directed edges of the original graph. GUMP then computes a weighted adjacency matrix A_w on $L(G')$, projects it to a unitary matrix $\boldsymbol{U}[\boldsymbol{A}_w]$, performs message passing on $L(G')$.

Our framework does not require this line-graph transformation. The signal remains on the original vertex set V, and the graph operator is constructed directly from the original adjacency weights and the learned feature locations. Therefore, our method keeps the signal domain fixed.

Unitary projection vs. exponential of a self-adjoint operator. GUMP obtains a unitary adjacency matrix by projecting a learned weighted adjacency matrix onto the unitary group. In its polar-projection form, this can be written as $\boldsymbol{U}[A_w] = \arg\min_{\boldsymbol{U}^\dagger\boldsymbol{U}=\boldsymbol{I}} \|\boldsymbol{A}_w - \boldsymbol{U}\|_F^2 = \boldsymbol{A}_w(\boldsymbol{A}_w^\dagger\boldsymbol{A}_w)^{-1/2}$.

Our method does not project an arbitrary learned adjacency matrix to the unitary group. Instead, unitarity comes from the spectral theorem: since Δ_f is self-adjoint, $\mathcal{S}[t, f] = e^{-it\Delta_f}$ is unitary. We believe this distinction is important as GUMP enforces unitarity by projecting a graph message-passing matrix, and our method derives unitarity from a Schrödinger-type Hamiltonian constructed from feature derivatives.

A.2.3 Comparison with unitary group convolutions

We next compare our method with the unitary group convolution framework [59]. Unitary group convolutions are designed for efficient convolutional neural networks, especially CNNs with grouped channel convolutions. Our method is designed for graph signal propagation and feature-space routing.

UGConv: unitary transforms around channel group convolutions. A convolutional layer with M input channels and N output channels can be represented by $y^{(j)} = \sum_{i=1}^{M} x^{(i)} * W^{(i,j)}$ for $1 \le j \le N$. A group convolution divides channels into G disjoint groups $\widetilde{y}^{(g,j)} = \sum_{i=1}^{M/G} \widetilde{x}^{(g,i)} * \widetilde{W}^{(g,i,j)}$ for $1 \le j \le N/G$. This reduces parameters and floating-point operations, but it also removes many cross-group channel connections. UGConv addresses this by sandwiching the group convolution between two unitary channel transforms P and Q:

$$\begin{aligned} \widetilde{X}_k &= PX_k, \qquad \forall k, \\ \widetilde{y}^{(g,j)} &= \sum_{i=1}^{M/G} \widetilde{x}^{(g,i)} * \widetilde{W}^{(g,i,j)}, \\ Y_\ell &= Q\widetilde{Y}_\ell, \qquad \forall \ell. \end{aligned}$$

Here $P \in \mathbb{C}^{M\times M}$ and $Q \in \mathbb{C}^{N\times N}$ are unitary transforms applied across channels.

Our Schrödinger method uses unitarity in a different domain. The unitary operator $\mathcal{S}[t, f] = e^{-it\Delta_f}$ acts across graph nodes, and Δ_f is built from the graph structure and feature locations. Channel mixing is handled separately by matrices $\boldsymbol{W}^{(m)}$, but the main unitary propagation is a graph-domain evolution, not a channel-domain basis change.

Different meaning of "group convolution." In [59], "group convolution" refers to grouped channel convolution in CNNs: the input and output channels are partitioned into groups, and convolution is performed independently in each group. It is not a graph convolution and does not use a graph adjacency matrix. It is also not primarily about group-equivariant convolution over symmetry groups.

Efficiency-oriented unitarity versus physics-inspired unitarity. UGConv uses unitary transforms because they preserve inner products and allow information to mix across channel groups without destroying gradient magnitudes. The paper shows that ShuffleNet [60] can be interpreted as a UGConv with a permutation matrix, and block-circulant networks (e.g., [61]) can be interpreted as UGConvs with Fourier transforms. It then proposes Hadamard transforms as an efficient dense channel-mixing alternative, where Hadamard matrices have ± 1 entries, can be computed using additions and subtractions, and avoid the complex arithmetic of Fourier transforms.

Our use of unitarity is closer to Schrödinger dynamics. Since Δ_f is self-adjoint, $\mathcal{S}[t, f]$ is unitary. Moreover, our complex phases are not introduced for computationally efficient channel mixing, they are introduced to create momentum along feature coordinates $D[\theta h]g = e^{i\theta h} \odot g$. Thus, UGConv uses unitary transforms as efficient channel-basis changes, while our method uses unitary evolution and modulation as interpretable graph-signal transport mechanisms.

A.3 Relation to quantum and quantum-inspired graph neural networks

Our use of the term "Schrödinger" is mathematical rather than hardware-oriented. The proposed model is a classical graph signal processing and graph neural network architecture whose propagation operator is motivated by the Schrödinger equation. It is important to highlight this distinction as the phrase "quantum graph neural network" has been used in the literature for several different families of methods. In short, prior quantum and quantum-inspired GNNs primarily use quantum circuits, quantum walks, or tensor-network parameterizations to define new graph learning architectures. Our work instead develops a classical Schrödinger graph signal processing framework in which graph signals are routed through learned feature-location coordinates using observables, momentum, commutators, and phase modulation. This distinction clarifies both the relationship to the existing literature and the technical contribution of the proposed Schrödinger GNN. We next position our method relative to these families and clarify the novelty of our construction.

Quantum-circuit graph neural networks. A first line of work studies quantum graph neural networks as quantum neural-network ansätze implemented by parameterized quantum circuits [62, 63]. For example, [62] introduces graph-structured quantum neural network ansätze designed to represent quantum processes with graph structure, including quantum graph recurrent neural networks and quantum graph convolutional neural networks. In their setting, the graph typically specifies interactions among quantum subsystems or the structure of a quantum computation, and node or edge information is encoded into qubits, transformed by trainable quantum gates, and read out through quantum measurements. The central design questions in this line of work are quantum-circuit expressivity, quantum state preparation, measurement, and compatibility with near-term or distributed quantum hardware.

Our method is different in both computational substrate and mathematical objective. We do not encode graph data into qubits, do not use parameterized quantum gates, and do not perform quantum measurements. All operations in our framework are standard finite-dimensional linear-algebra operations on classical node features. The graph signal $g : V \to \mathbb{C}^J$ is represented explicitly as a complex-valued array over the vertices, and the unitary operator $\mathcal{S}[t, f] = \exp(-it\Delta_f)$ is evaluated as a spectral graph filter. Thus, unitarity in our model is not used to obtain quantum computation, but to obtain norm-preserving, phase-sensitive signal propagation on a graph. The relevant inductive bias is signal transport in learned feature-location coordinates.

Quantum-walk and quantum-propagation graph convolutions. Another related line of work uses quantum walks to define graph convolution operators [64–67]. A quantum walk is the quantum analog of a classical random walk: instead of evolving a real-valued probability distribution by a stochastic transition matrix, it evolves a complex-valued amplitude vector by a unitary operator. The probability of visiting a vertex is then obtained from the squared modulus of the corresponding complex amplitude.

Several graph learning methods use such quantum-walk dynamics to construct feature aggregation weights. For example, quantum spatial graph convolutional networks [68] compute an average mixing matrix $\boldsymbol{Q}$ from a continuous-time quantum walk and use it as the propagation matrix in a graph convolution $\boldsymbol{Z} = \sigma(\boldsymbol{QXW})$, where $\boldsymbol{X}$ is the node feature matrix and $\boldsymbol{W}$ is a trainable channel-mixing matrix. Thus, the quantum-walk component primarily replaces the usual adjacency- or Laplacian-based propagation matrix by a quantum-walk-derived visiting probability matrix.

Our method is related to this family only at the broad level that both use unitary dynamics as inspiration for graph propagation. The construction and purpose are different. Quantum-walk-based graph convolutions usually derive a real-valued propagation kernel from the visiting probabilities of a walk generated by a fixed graph operator such as $\boldsymbol{A}$ or $\boldsymbol{L}$. In contrast, we construct a feature-dependent Schrödinger Laplacian $\Delta_f = -\sum_{k=1}^{K} \nabla_{f_k}^2$, $(\nabla_{f_k})_{n,m} = a_{n,m}(f_k(n) - f_k(m))$, and propagate complex graph signals by $\mathcal{S}[t, f] = e^{-it\Delta_f}$. Moreover, our architecture includes feature modulation $D[\theta h] = \mathrm{diag}(e^{i\theta h})$, which creates controllable feature momentum, and our analysis is formulated in terms of feature-location observables X_{f_k}, momentum operators $i\nabla_{f_k}$, variances, routing measures, and commutators. These elements are not present in quantum-walk graph convolutions, whose main role is to define an alternative graph aggregation kernel.

It is important to note that the components proposed by our framework, including learned feature-location coordinates using observables, momentum, commutators, and phase modulation, are not present in previous QGNN or tensor-network GNN constructions. Its role is to make multidimensional feature-location propagation more interpretable by reducing cross-direction interference.

A.4 Sparse signal routing versus dense attention

Transformers provide a natural alternative mechanism for long-range communication. In a standard self-attention layer, every node or token can attend to every other node or token. Thus, in principle, a Transformer can route information from a source region to a specific distant destination in a single layer, since the attention weights define a task-dependent soft communication pattern between all pairs of locations [69]. Thus, Transformer-based graph models are attractive for tasks requiring long-range interactions [70, 71, 47]. However, dense self-attention achieves this flexibility by explicitly considering all node pairs. For a graph with N nodes, full attention has quadratic time and memory complexity in the number of nodes, namely $O(N^2)$ pairwise attention scores per layer, up to channel and head dimensions. This all-pairs communication bypasses graph bottlenecks, but it is also expensive and can be unnecessary when the relevant information flow is structured by the graph structure or by a small number of meaningful feature directions. Schrödinger GNNs address the same conceptual problem, routing information between distant regions, from a different perspective. Rather than constructing an explicit all-pairs communication matrix, our method defines feature-location observables and uses complex modulation to induce momentum along learned formal directions. The Schrödinger operator then propagates the signal coherently along these directions. Thus, routing is represented as a geometry-preserving flow of localized signal content, instead of as a dense set of pairwise attention coefficients. In addition, our method is $O(|E|)$ for fixed architectural hyperparameters. This contrasts with the $O(N^2)$ pairwise cost of dense Transformer attention.

B Computational realization of Schrödinger GNNs

In this appendix, we explain how the Schrödinger graph signal processing introduced in Section 3 is instantiated in practice. One important note is that the model does not require explicit construction of a dense matrix exponential. Instead, it applies sparse feature-dependent operators to node features and uses a truncated series to approximate the action of the Schrödinger propagator.

B.1 Approximating the unitary propagation

In the implementation, we approximate $\mathcal{S}[t, f] = e^{-it\Delta_f}$ on feature tensors rather than forming the dense matrix exponential explicitly. For any operator B, we have

$$e^B = \sum_{r=0}^{\infty} \frac{B^r}{r!},$$

thus, the order-R truncated Taylor approximation is given by

$$e^{-it\Delta_f} X \approx \sum_{r=0}^{R} \frac{(-it\Delta_f)^r}{r!} X.$$

A practical advantage of this formulation is that one can apply Δ_f without explicitly constructing the matrix Δ_f. Specifically, for the multi-feature operator, we have

$$\Delta_f X = -\sum_{k=1}^{K} \nabla_{f_k} \big(\nabla_{f_k} X\big).$$

Each application of ∇_{f_k} is an edge-local sparse operation, so one evaluates $\Delta_f X$ by a sequence of sparse passes over the graph. This is more efficient for the actual implementation than a dense matrix-exponential routine.

Approximation error introduced by the truncated Taylor expansion. We remark that the implementation applies the action of $\mathcal{S}[t, f] = e^{-it\Delta_f}$ through a truncated Taylor series, and it does not construct a dense exponential. We evaluate the approximation on trained propagation operators at the truncation orders and find that: (i) with the order-10 approximation $\tilde{\mathcal{S}}_{10}$ of the unitary Schrödinger operator $\mathcal{S}$, the Taylor remainder satisfies $||\tilde{\mathcal{S}}_{10} - \mathcal{S}||_{\text{op}} \leq 4.44 \times 10^{-3}$, and $||\tilde{\mathcal{S}}_{10}^* \tilde{\mathcal{S}}_{10} - I||_{\text{op}} \leq 8.90 \times 10^{-3}$, where the singular values of the approximate propagation operator lie in [0.99999794, 1.00000206], and (ii) with the order-15 approximation $\tilde{\mathcal{S}}_{15}$, the Taylor remainder satisfies $||\tilde{\mathcal{S}}_{15} - S||_{\text{op}} \leq 2.06 \times 10^{-6}$, and $||\tilde{\mathcal{S}}_{15}^* \tilde{\mathcal{S}}_{15} - I||_{\text{op}} \leq 4.12 \times 10^{-6}$, where the singular values of the approximate propagation operator lie in [0.99999794, 1.00000206]. Hence, near-unitarity, routing observables, and propagation stability are preserved to within the reported numerical tolerances after Taylor truncation.

B.2 Layer-level realization of the Schrödinger GNN

Let $\boldsymbol{F} \in \mathbb{R}^{N \times K_f}$ be the matrix of formal-location features, with $\boldsymbol{F}_{:,k} = f_k$, and let $\boldsymbol{X} \in \mathbb{C}^{N \times J}$ be the current complex signal matrix. A Schrödinger filter with M filter terms outputs $\boldsymbol{Y} \in \mathbb{C}^{N \times D}$ by

$$\boldsymbol{Y} = \sum_{m=1}^{M} \mathcal{S}[t_m, \mathbf{f}]\, D[\theta_m \boldsymbol{F}\boldsymbol{T}^{(m)}]\, \boldsymbol{X}\, \boldsymbol{W}^{(m)}.$$

Here $t_m, \theta_m \in \mathbb{R}$ are scalar propagation and modulation parameters, $\boldsymbol{T}^{(m)} \in \mathbb{R}^{K_f \times 1}$ chooses a direction in the formal-location feature space, $D[\theta_m \boldsymbol{F}\boldsymbol{T}^{(m)}]$ is the diagonal phase-modulation operator with diagonal entries $\exp\big(i\theta_m (\boldsymbol{F}\boldsymbol{T}^{(m)})_n\big)$ for $n \in [N]$, and $\boldsymbol{W}^{(m)} \in \mathbb{C}^{J \times D}$ mixes feature channels. The order of operations is: phase-modulate the signal, propagate it with the unitary Schrödinger operator, and then mix channels. In such construction, the modulation creates directional momentum, while $\mathcal{S}[t_m, \mathbf{f}]$ transports the signal along the formal locations.

B.3 Complex input modulation

The raw input node-feature matrix is denoted by $\boldsymbol{Q} \in \mathbb{R}^{N \times d_{\text{in}}}$. Before applying Schrödinger propagation, we map $\boldsymbol{Q}$ to a complex signal matrix by an amplitude–phase factorization,

$$\widetilde{\boldsymbol{X}} = (\boldsymbol{Q}\boldsymbol{B}) \odot \exp\big(i\, \boldsymbol{Q}\boldsymbol{P}\big) \in \mathbb{C}^{N \times d},$$

where $\boldsymbol{B}, \boldsymbol{P} \in \mathbb{R}^{d_{\text{in}} \times d}$ are learned linear maps and $\odot$ denotes entrywise multiplication. The amplitude term $\boldsymbol{Q}\boldsymbol{B}$ carries the real-valued content, while the phase term $\exp(i\boldsymbol{Q}\boldsymbol{P})$ creates oscillations that can induce nonzero expected momentum under the Schrödinger dynamics. This conversion is not

a dimensionality-reducing projection by itself, but it represents each learned channel by a complex amplitude whose modulus and phase are both trainable functions of the original features.

B.4 Position-momentum optimization

When Position-momentum optimization (PMO) is used, it is run before training the main prediction network as a preprocessing step.

Given the raw candidate formal-location features $\boldsymbol{Q} \in \mathbb{R}^{N\times M}$, PMO learns a linear map $\boldsymbol{T} \in \mathbb{R}^{M\times K_f}$ and defines $\boldsymbol{F} = \boldsymbol{Q}\boldsymbol{T}$. The objective in Definition 13 penalizes cross-commutators such as $[\nabla^2_{f_j}, X_{f_i}]$ and includes the stated regularization terms. In practice, we optimize $\boldsymbol{T}$ with Adam over the training graphs, using mini-batches when the dataset contains multiple graphs. After this preprocessing stage, $\boldsymbol{T}$ is fixed, and the resulting formal-location matrix $\boldsymbol{F}$ is used by the Schrödinger GNN during training and inference. Thus, PMO is an additional preprocessing cost, not an additional matrix-exponential cost at every layer.

B.5 Nonlinearities and dropout

After the modulation layer, the network operates on complex-valued node features. We apply the nonlinearities componentwise [72], given by

$$\sigma(z) = \mathrm{ReLU}(\mathrm{Re}(z)) + i\,\mathrm{ReLU}(\mathrm{Im}(z)),$$

and we found that it works well in practice. We also explore magnitude $\sigma(z) = |z|$. The componentwise choice keeps the representation in the complex domain, and the magnitude is used at the final stage when the downstream task requires real-valued outputs.

B.6 Computational complexity

Let R denote the Taylor truncation order, let K be the number of feature-location directions used to build Δ_f, and let C be the hidden feature dimension. One application of ∇_{f_k} to a feature tensor costs $O(|E|C)$, since it is an edge-local sparse operation. Therefore, one application of $\nabla^2_{f_k}$ also costs $O(|E|C)$ up to a constant factor, and one application of $\Delta_f = -\sum_{k=1}^{K} \nabla^2_{f_k}$ costs $O(K|E|C)$. An order-R truncated Taylor approximation costs $O(RK|E|C)$ per layer. For fixed R and K, this remains linear in the number of edges.

The memory cost is dominated by the feature tensors, the sparse graph structure, and the stored feature-location channels. Note that we do not compute the dense matrix exponential, which keeps memory usage comparable to other sparse message-passing architectures. In our experiments, we found that values around $R \approx 10$–15 gave a good balance between accuracy and cost. The memory complexity is $O((|V| + |E|)C)$, similar to standard GNNs, as we do not explicitly construct the dense matrix exponential.

B.7 Propagation time initialization

Each output channel has a learnable real propagation-time parameter. If a layer has C output channels, we initialize

$$t_j \sim \mathrm{Uniform}(0, 1.5), \qquad j = 1, \ldots, C.$$

Small values of t_j produce local filters, while larger values allow a channel to aggregate information from farther regions through the Schrödinger propagation. This initialization gives different channels different initial transport ranges. When learnable propagation times are disabled, we use a fixed scalar time value instead.

C Theoretical results and proofs

We present the theoretical results and proof for Schrödinger graph signal processing in Section 3. To support these proofs, we also include several additional propositions and lemmas, which are numbered separately for clarity. We restate the claim of each statement for convenience.

C.1 Background in functional calculus

We recall the finite-dimensional functional calculus for self-adjoint operators. Let $\boldsymbol{M} \in \mathbb{C}^{N\times N}$ be self-adjoint and let $\boldsymbol{M} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^*$ be its eigendecomposition, where $\boldsymbol{V}$ is a unitary matrix and $\boldsymbol{\Lambda}$ is a diagonal matrix with elements $\mathrm{diag}(\lambda_1, \ldots, \lambda_N)$.

For any function $y : \mathbb{R} \to \mathbb{C}$ defined on the spectrum $\{\lambda_1, \ldots, \lambda_N\}$, we have $y(\boldsymbol{M}) = \boldsymbol{V}y(\boldsymbol{\Lambda})\boldsymbol{V}^*$, where $y(\boldsymbol{\Lambda}) = \mathrm{diag}(y(\lambda_1), \ldots, y(\lambda_N))$. If $\boldsymbol{v}_j$ is an orthonormal eigenbasis of $\boldsymbol{M}$ with $\boldsymbol{M}\boldsymbol{v}_j = \lambda_j\boldsymbol{v}_j$, then $y(\boldsymbol{M})\boldsymbol{v}_j = y(\lambda_j)\boldsymbol{v}_j$. Thus, $y(\boldsymbol{M})$ is diagonal in an orthonormal basis, and hence is a normal operator. In addition, if y maps to $\mathbb{R}$ into the unit circle $\{z \in \mathbb{C} : |z| = 1\}$, then $y(\boldsymbol{M})$ is unitary.

Lemma 14. *Let $\boldsymbol{M}, \boldsymbol{R} \in \mathbb{C}^{N\times N}$ be commuting self-adjoint operators (i.e., $\boldsymbol{M}\boldsymbol{R} = \boldsymbol{R}\boldsymbol{M}$). Then, for every function $f : \mathbb{R} \to \mathbb{C}$, M commutes with $y(\boldsymbol{R})$.*

C.2 Proof of Theorem 5

Theorem 5 (Constant expected momentum)**.** *Let $g^{(0)} : V \to \mathbb{C}$ be a normalized signal and $f : V \to \mathbb{R}$ a feature location. Then, for every $t \in \mathbb{R}$,*

$$\mathcal{E}_{i\nabla_f}(g^{(t)}) = \mathcal{E}_{i\nabla_f}(g^{(0)}).$$

Proof. Let $\Delta_f = -\nabla_f^2$ and $\mathcal{S}[t, f] = e^{-it\Delta_f}$. Since Δ_f is self-adjoint, and $e^{-it(\cdot)}$ maps to the complex unit circle, $\mathcal{S}[t, f]$ is unitary. Moreover, since Δ_f commutes with ∇_f, by Lemma 14 ∇_f commutes with $\mathcal{S}[t, f]$. Hence,

$$\begin{aligned}
\mathcal{E}_{i\nabla_f}(g^{(t)}) &= \left\langle i\nabla_f\mathcal{S}[t, f]g^{(0)}, \mathcal{S}[t, f]g^{(0)} \right\rangle \\
&= \left\langle \mathcal{S}[t, f]i\nabla_f g^{(0)}, \mathcal{S}[t, f]g^{(0)} \right\rangle \\
&= \left\langle i\nabla_f g^{(0)}, g^{(0)} \right\rangle = \mathcal{E}_{i\nabla_f}(g^{(0)}).
\end{aligned}$$

□

C.3 Proof of Theorem 7

Before proving Theorem 7, we develop some analysis for the smoothing operator.

Lemma 15 (Smoothing operator as commutator)**.**

$$W_f = [X_f, \nabla_f] \;=\; X_f\nabla_f - \nabla_f X_f \;=\; -[\nabla_f, X_f].$$

Proof. For any signal g and vertex v, we have

$$\begin{aligned}
([X_f, \nabla_f]g)(v) &= (X_f\nabla_f g)(v) - (\nabla_f X_f g)(v) \\
&= f(v)\sum_{w\in V} a_{v,w}(f(v) - f(w))g(w) - \sum_{w\in V} a_{v,w}(f(v) - f(w))f(w)g(w) \\
&= \sum_{w\in V} a_{v,w}(f(v) - f(w))^2 g(w) \\
&= (W_f g)(v).
\end{aligned}$$

Hence, $[X_f, \nabla_f] = W_f$, and equivalently $[\nabla_f, X_f] = -W_f$.

□

Lemma 16 (Commutator of Schrödinger Laplacian and location observable)**.** *For the Schrödinger Laplacian $\Delta = -\nabla_f^2$ and location observable X_f, we have*

$$[\Delta, X_f] = \nabla_f W_f + W_f\nabla_f$$

where $W_f = [\nabla_f, X_f]$ is the f-smoothing operator.

Proof. Using the product rule for commutators $[AB, C] = A[B, C] + [A, C]B$, and by Lemma 15, we have

$$\begin{aligned}[\Delta, X_f] &= [-\nabla_f^2, X_f] \\ &= -[\nabla_f \nabla_f, X_f] \\ &= -\nabla_f[\nabla_f, X_f] - [\nabla_f, X_f]\nabla_f \\ &= -\nabla_f(-W_f) - (-W_f)\nabla_f \\ &= \nabla_f W_f + W_f \nabla_f\end{aligned}$$

□

Now we prove Theorem 7.

Theorem 7 (Expected feature location derivative under Schrödinger dynamics)**.** *Let $g^{(0)} : V \to \mathbb{C}$ be a normalized signal and $f : V \to \mathbb{R}$ a feature location. Let $g^{(t)} = \mathcal{S}[t, f]g^{(0)}$ for every $t \in \mathbb{R}$. Then,*

$$\frac{\partial}{\partial t}\mathcal{E}_{X_f}(g^{(t)}) = 2\mathrm{Re}\left(\langle i\nabla_f g^{(t)}, W_f g^{(t)}\rangle\right).$$

Proof. Note that by definition of $\mathcal{E}_{X_f}(g^{(t)}) = \left\langle X_f g^{(t)}, g^{(t)}\right\rangle$, we have

$$\frac{\partial}{\partial t}\mathcal{E}_{X_f}(g^{(t)}) = \frac{\partial}{\partial t}\left\langle X_f g^{(t)}, g^{(t)}\right\rangle$$

and since $g^{(t)} = \mathcal{S}[t, f]g^{(0)}$, by the chain rule , we have

$$\begin{aligned}\frac{\partial}{\partial t}\mathcal{E}_{X_f}(g^{(t)}) &= \left\langle X_f(-i\Delta_f)g^{(t)}, g^{(t)}\right\rangle + \left\langle X_f g^{(t)}, (-i\Delta_f)g^{(t)}\right\rangle \\ &= \left\langle i[\Delta_f, X_f]g^{(t)}, g^{(t)}\right\rangle\end{aligned}$$

Substituting $\Delta_f = -\nabla_f^2$ and using Lemma 16 yields

$$\begin{aligned}\frac{\partial}{\partial t}\mathcal{E}_{X_f}(g^{(t)}) &= \langle i\nabla_f W_f g^{(t)}, g^{(t)}\rangle + \langle W_f i\nabla_f g^{(t)}, g^{(t)}\rangle \\ &= \langle W_f g_t, i\nabla_f g_t\rangle + \langle i\nabla_f g^{(t)}, W_f g^{(t)}\rangle \\ &= \overline{\langle i\nabla_f g^{(t)}, W_f g^{(t)}\rangle} + \langle i\nabla_f g^{(t)}, W_f g^{(t)}\rangle \\ &= 2\,\mathrm{Re}\left(\langle i\nabla_f g^{(t)}, W_f g^{(t)}\rangle\right).\end{aligned}$$

□

The previous theorem still contains the smoothed signal $W_f g^{(t)}$. The next definition isolates the regime in which $W_f g^{(t)}$ is close to $g^{(t)}$, so that the derivative of the expected feature is close to the exact momentum.

Definition 17 (ϵ-f regular signal)**.** *Let $\mathcal{G} = (V, E)$ be a graph, $f : V \to \mathbb{R}$ be a feature location, and W_f the corresponding f-smoothing operator. A signal $g : V \to \mathbb{C}$ is called ϵ-f regular if there exists a signal $e_g : V \to \mathbb{R}$ such that $W_f g = g + e_g$ with $\|e_g\|_2 \leq \epsilon$.*

The next theorem quantifies the extent to which the the exact expected momentum approximates the rate of change of the expected feature location, when the signal is ϵ-f regular.

Theorem 18 (Expected feature location dynamics for ϵ-f regular signals)**.** *Let $g^{(0)} : V \to \mathbb{C}$ be normalized and $g^{(t)} = \mathcal{S}[t, f]g^{(0)}$. If $g^{(t)}$ is ϵ-f regular, then*

$$\left|\frac{\partial}{\partial t}\mathcal{E}_{X_f}(g^{(t)}) - 2\mathcal{E}_{i\nabla_f}(g^{(0)})\right| \leq 2\epsilon\|\nabla_f\|_{\mathrm{op}}.$$

Proof. Recall from Theorem 7 that

$$\frac{\partial}{\partial t}\mathcal{E}_{X_f}(g^{(t)}) = 2\mathrm{Re}\left(\langle i\nabla_f g^{(t)}, W_f g^{(t)}\rangle\right). \tag{15}$$

By the ϵ–f regularity assumption there exists $e_{g^{(t)}}$ with $\|e_{g^{(t)}}\|_2 \leq \epsilon$ such that $W_f g^{(t)} = g^{(t)} + e_{g^{(t)}}$. Using this identity together with (15) gives

$$\begin{aligned}\left|\frac{\partial}{\partial t}\mathcal{E}_{X_f}(g^{(t)}) - 2\,\mathcal{E}_{i\nabla_f}(g^{(t)})\right| &= \left|2\,\mathrm{Re}\Big(\langle i\nabla_f g^{(t)}, e_{g^{(t)}}\rangle\Big)\right| \\ &\leq 2\,\|i\nabla_f g^{(t)}\|_2\,\|e_{g^{(t)}}\|_2 \\ &\leq 2\epsilon\,\|\nabla_f\|_{\mathrm{op}}\,\|g^{(t)}\|_2 \\ &= 2\epsilon\,\|\nabla_f\|_{\mathrm{op}}.\end{aligned}$$

Hence, by Theorem 5, we have

$$\left|\frac{\partial}{\partial t}\mathcal{E}_{X_f}(g^{(t)}) - 2\mathcal{E}_{i\nabla_f}(g^{(0)})\right| \leq 2\epsilon\|\nabla_f\|_{\mathrm{op}}.$$

□

C.4 Dynamics of the variance

Next, we derive the dynamics of the variance.

Theorem 19 (Time derivative of variance)**.** *Let $g : V \to \mathbb{C}$ be a signal and $f : V \to \mathbb{R}$ a feature location, and $\Delta_f = -\nabla_f^2$. The first-order derivative of variance with respect to time $t \in \mathbb{R}$ is*

$$\frac{\partial}{\partial t}\mathcal{V}_{X_f}(g^{(t)}) = \mathcal{E}_{i[\Delta_f, X_f^2]}(g^{(t)}) - 4\mathcal{E}_{X_f}(g^{(t)})\mathrm{Re}\big(\langle i\nabla_f g^{(t)}, W_f g^{(t)}\rangle\big).$$

This mirrors the classical Schrödinger equation dynamics where variance evolution depends on both the commutator $[\Delta, X^2]$ and the coupling between position and momentum. See Appendix D for the classical correspondence.

Proof of Theorem 19. By the definition of variance, we have

$$\mathcal{V}_{X_f}(g^{(t)}) = \mathcal{E}_{X_f^2}(g^{(t)}) - \mathcal{E}_{X_f}(g^{(t)})^2.$$

Differentiating with respect to t then gives

$$\frac{\partial}{\partial t}\mathcal{V}_{X_f}(g^{(t)}) = \frac{\partial}{\partial t}\mathcal{E}_{X_f^2}(g^{(t)}) - 2\mathcal{E}_{X_f}(g^{(t)})\frac{\partial}{\partial t}\mathcal{E}_{X_f}(g^{(t)}).$$

From the time evolution of the expected feature for every observable, and by definition of $\mathcal{E}_{X_f^2}(g^{(t)}) = \left\langle X_f^2 g^{(t)}, g^{(t)}\right\rangle$, we have $\frac{\partial}{\partial t}\mathcal{E}_{X_f^2}(g^{(t)}) = \frac{\partial}{\partial t}\left\langle X_f^2 g^{(t)}, g^{(t)}\right\rangle$. Now, by the chain rule and the fact that $g^{(t)} = \mathcal{S}[t, f]g^{(0)}$, we have

$$\begin{aligned}\frac{\partial}{\partial t}\mathcal{E}_{X_f^2}(g^{(t)}) &= \left\langle X_f^2(-i\Delta_f)g^{(t)}, g^{(t)}\right\rangle + \left\langle X_f g^{(t)}, (-i\Delta_f)g^{(t)}\right\rangle \\ &= \left\langle i[\Delta_f, X_f^2]g^{(t)}, g^{(t)}\right\rangle \\ &= \mathcal{E}_{i[\Delta_f, X_f^2]}(g^{(t)})\end{aligned}$$

In addition, by Theorem 7, we have

$$\frac{\partial}{\partial t}\mathcal{E}_{X_f}(g^{(t)}) = 2\,\mathrm{Re}\left(\langle i\nabla_f g^{(t)}, W_f g^{(t)}\rangle\right).$$

By substituting both expressions, we obtain

$$\frac{\partial}{\partial t}\mathcal{V}_{X_f}(g^{(t)}) = \mathcal{E}_{i[\Delta_f, X_f^2]}(g^{(t)}) - 4\mathcal{E}_{X_f}(g^{(t)})\mathrm{Re}\big(\langle i\nabla_f g^{(t)}, W_f g^{(t)}\rangle\big).$$

□

C.5 Proof of Theorem 10

Theorem 10 (Expected momentum of modulated signal)**.** *Given a normalized signal $g : V \to \mathbb{R}$, feature locations $f, h : V \to \mathbb{R}$, and a phase $\theta \in \mathbb{R}$, the expected momentum of $D[\theta h]g$ satisfies*

$$\mathcal{E}_{i\nabla_f}(D[\theta h]g) = \sum_{(m,n)\in E} a_{m,n} g(m)g(n)(f(n) - f(m)) \sin(\theta(h(n) - h(m))).$$

Proof. For the modulated signal $D_{\theta h}g(v) = g(v)e^{i\theta h(v)}$, we have

$$(\nabla_f D_{\theta h}g)(m) = \sum_{n\in V} a_{m,n}\big(f(m) - f(n)\big)g(n)e^{i\theta h(n)}.$$

Therefore

$$\begin{aligned}\mathcal{E}_{i\nabla_f}(D_{\theta h}g) &= \langle i\nabla_f D_{\theta h}g, D_{\theta h}g\rangle \\ &= i\sum_{m\in V}\sum_{n\in V} a_{m,n} g(m)g(n)e^{i\theta(h(n)-h(m))}(f(m) - f(n)).\end{aligned}$$

Because the graph is undirected, we may group the two orientations of each edge. Namely,

$$\begin{aligned}&\mathcal{E}_{i\nabla_f}(D_{\theta h}g) \\ &= \frac{i}{2}\sum_{(m,n)\in E} a_{m,n} g(m)g(n)\left[e^{i\theta(h(n)-h(m))}(f(m) - f(n)) + e^{i\theta(h(m)-h(n))}(f(n) - f(m))\right] \\ &= \frac{i}{2}\sum_{(m,n)\in E} a_{m,n} g(m)g(n)(f(n) - f(m))\left[e^{-i\theta(h(n)-h(m))} - e^{i\theta(h(n)-h(m))}\right] \\ &= \frac{i}{2}\sum_{(m,n)\in E} a_{m,n} g(m)g(n)\big(f(n) - f(m)\big)\big(-2i\big)\sin(\theta(h(n) - h(m))) \\ &= \sum_{(m,n)\in E} a_{m,n} g(m)g(n)(f(n) - f(m))\sin(\theta(h(n) - h(m))).\end{aligned}$$

□

C.6 Proof of Theorem 11

Theorem 11 (Expected multi-feature derivative)**.** *Given the Schrödinger Laplacian $\Delta_f = -\sum_{k\in[K]} \nabla^2_{f_k}$, a normalized signal $g^{(0)}$, and $g^{(t)} = \mathcal{S}[t, f]g^{(0)}$, we have*

$$\frac{\partial}{\partial t}\mathcal{E}_{X_{f_k}}(g^{(t)}) = 2\mathrm{Re}\left(\langle i\nabla_{f_k} g^{(t)}, W_{f_k} g^{(t)}\rangle\right) - \sum_{j\neq k}\left\langle [i\nabla^2_{f_j}, X_{f_k}]g^{(t)}, g^{(t)}\right\rangle.$$

Proof. By definition of $\mathcal{E}_{X_{f_k}}(g^{(t)}) = \left\langle X_{f_k} g^{(t)}, g^{(t)}\right\rangle$, we have

$$\frac{\partial}{\partial t}\mathcal{E}_{X_{f_k}}(g^{(t)}) = \frac{\partial}{\partial t}\left\langle X_f g^{(t)}, g^{(t)}\right\rangle$$

and since $g^{(t)} = \mathcal{S}[t, f]g^{(0)}$, by the chain rule, we have

$$\begin{aligned}\frac{\partial}{\partial t}\mathcal{E}_{X_{f_k}}(g^{(t)}) &= \left\langle X_{f_k}(-i\Delta_f)g^{(t)}, g^{(t)}\right\rangle + \left\langle X_{f_k} g^{(t)}, (-i\Delta_f)g^{(t)}\right\rangle \\ &= \left\langle i[\Delta_f, X_{f_k}]g^{(t)}, g^{(t)}\right\rangle.\end{aligned}$$

Since $\Delta_f = -\sum_{j=1}^K \nabla^2_{f_j}$, we have

$$\frac{\partial}{\partial t}\mathcal{E}_{X_{f_k}}(g^{(t)}) = -\sum_{j=1}^K \langle [i\nabla^2_{f_j}, X_{f_k}]g^{(t)}, g^{(t)}\rangle.$$

We split the sum into the $j = k$ term and the cross terms. When $j = k$, using Lemma 16 gives

$$\begin{aligned}\frac{\partial}{\partial t}\mathcal{E}_{X_{f_k}}(g^{(t)}) &= \langle i\nabla_{f_k} W_f g^{(t)}, g^{(t)}\rangle + \langle W_{f_k} i\nabla_{f_k} g^{(t)}, g^{(t)}\rangle \\ &= \langle W_{f_k} g_t, i\nabla_{f_k} g_t\rangle + \langle i\nabla_{f_k} g^{(t)}, W_{f_k} g^{(t)}\rangle \\ &= \overline{\langle i\nabla_{f_k} g^{(t)}, W_{f_k} g^{(t)}\rangle} + \langle i\nabla_{f_k} g^{(t)}, W_{f_k} g^{(t)}\rangle \\ &= 2\,\mathrm{Re}\left(\langle i\nabla_{f_k} g^{(t)}, W_{f_k} g^{(t)}\rangle\right).\end{aligned}$$

Therefore, we have

$$\left\langle i[\Delta_{f_k}, X_{f_k}] g^{(t)}, g^{(t)}\right\rangle = 2\mathrm{Re}\left(\langle i\nabla_{f_k} g^{(t)}, W_{f_k} g^{(t)}\rangle\right).$$

Keeping the remaining indices explicit gives

$$\frac{\partial}{\partial t}\mathcal{E}_{X_{f_k}}(g^{(t)}) = 2\mathrm{Re}\left(\langle i\nabla_{f_k} g^{(t)}, W_{f_k} g^{(t)}\rangle\right) - \sum_{j\neq k}\left\langle [i\nabla_{f_j}^2, X_{f_k}] g^{(t)}, g^{(t)}\right\rangle.$$

□

The cross terms in Theorem 11 vanish when $i\nabla_{f_j}^2$ commute with X_{f_k} for $j \neq k$. The next result quantifies the deviation from the single feature dynamics when commutation holds only approximately and the signal is ϵ-f_k regular.

Theorem 20 (Dynamics of multi-channel expected feature location for ϵ-f regular signals)**.** *Let $\Delta_f = -\sum_{j\in[K]} \nabla_{f_j}^2$, $g^{(t)} = \mathcal{S}[t, f]g = e^{-it\Delta_f} g^{(0)}$ with $\|g^{(0)}\|_2 = 1$, and $k \in [K]$. Suppose that $g^{(t)}$ is ϵ-f_k regular and that the feature locations $\{f_1, \ldots, f_K\}$ form a δ-commuting in the sense of Definition 12. Then,*

$$\left|\frac{\partial}{\partial t}\mathcal{E}_{X_{f_k}}(g^{(t)}) - 2\mathcal{E}_{i\nabla_{f_k}}(g^{(t)})\right| \leq 2\epsilon\|\nabla_{f_k}\|_{\mathrm{op}} + (K-1)\delta.$$

Proof. By Theorem 11, we have

$$\frac{\partial}{\partial t}\mathcal{E}_{X_{f_k}}(g^{(t)}) = 2\mathrm{Re}\left(\langle i\nabla_{f_k} g^{(t)}, W_{f_k} g^{(t)}\rangle\right) - \sum_{j\neq k}\left\langle [i\nabla_{f_j}^2, X_{f_k}] g^{(t)}, g^{(t)}\right\rangle.$$

Let $A_k(t) := 2\mathrm{Re}\big(\langle i\nabla_{f_k} g^{(t)}, W_{f_k} g^{(t)}\rangle\big)$ and $C_{k,j}(t) := \left\langle [i\nabla_{f_j}^2, X_{f_k}] g^{(t)}, g^{(t)}\right\rangle$ for $j \neq k$, we have

$$\frac{\partial}{\partial t}\mathcal{E}_{X_{f_k}}(g^{(t)}) = A_k(t) - \sum_{j\neq k} C_{k,j}(t),$$

and therefore,

$$\left|\frac{\partial}{\partial t}\mathcal{E}_{X_{f_k}}(g^{(t)}) - 2\mathcal{E}_{i\nabla_{f_k}}(g^{(t)})\right| \leq \left|A_k(t) - 2\mathcal{E}_{i\nabla_{f_k}}(g^{(t)})\right| + \sum_{j\neq k}|C_{k,j}(t)|.$$

Because $g^{(t)}$ is ϵ-f_k regular, there exists a signal e_t such that $W_{f_k} g^{(t)} = g^{(t)} + e_t$ and $\|e_t\|_2 \leq \epsilon$. Since $i\nabla_{f_k}$ is self-adjoint and $\mathcal{E}_{i\nabla_{f_k}}(g^{(t)}) = \langle i\nabla_{f_k} g^{(t)}, g^{(t)}\rangle$, we have

$$\begin{aligned}A_k(t) - 2\mathcal{E}_{i\nabla_{f_k}}(g^{(t)}) &= 2\mathrm{Re}\big(\langle i\nabla_{f_k} g^{(t)}, W_{f_k} g^{(t)}\rangle\big) - 2\langle i\nabla_{f_k} g^{(t)}, g^{(t)}\rangle \\ &= 2\mathrm{Re}\big(\langle i\nabla_{f_k} g^{(t)}, W_{f_k} g^{(t)} - g^{(t)}\rangle\big) \\ &= 2\mathrm{Re}\big(\langle i\nabla_{f_k} g^{(t)}, e_t\rangle\big).\end{aligned}$$

Therefore,

$$\left|A_k(t) - 2\mathcal{E}_{i\nabla_{f_k}}(g^{(t)})\right| \leq 2\|i\nabla_{f_k} g^{(t)}\|_2\,\|e_t\|_2 \leq 2\epsilon\|\nabla_{f_k}\|_{\mathrm{op}}.$$

For each $j \neq k$, the δ-commuting assumption gives $\|[\nabla_{f_j}^2, X_{f_k}]\|_{op} \leq \delta$. Hence,

$$|C_{k,j}(t)| \leq \|[i\nabla_{f_j}^2, X_{f_k}]\|_{\mathrm{op}}\,\|g^{(t)}\|_2^2 = \|[\nabla_{f_j}^2, X_{f_k}]\|_{\mathrm{op}} \leq \delta.$$

By summing over $j \neq k$, then we have $\sum_{j\neq k}|C_{k,j}(t)| \leq (K-1)\delta$. □

C.7 Improving signal routing through modulation

Here, we show that in typical situations modulating real-valued signals improves their signal routing measure. Consider the following setting. We have a multilayer network where at each layer l we have a real-valued signal $g^{(l)}$ that we are allowed to modulate by choosing the free parameter $\theta_l \in \mathbb{R}$ in $D[\theta_l h]g^{(l)}$. We then propagate the signal via $\mathcal{S}[dt, f]D[\theta_l h]g^{(l)}$ for some small time step dt, and lastly apply a modulus nonlinearity to define the signal at the next layer $g^{(l+1)} = \left|\mathcal{S}[dt, f]D[\theta_l h]g^{(l)}\right|$. Here, we can interpret $g^{(l)}$ as the signal at time ldt, and the input to the network $g^{(0)}$ as the signal at time 0.

Suppose that we would like to route the signal to the feature location r, i.e., we would like $\mathcal{P}_{X_f}(g^{(0)}, D[\theta_l h]g^{(l)}, r)$ to decrease in l by choosing appropriate θ_l. In this setting, since dt is small, we can linearize the propagation of $g^{(l)}$ about $t = 0$, and obtain

$$\mathcal{P}_{X_f}(g^{(0)}, g^{(l+1)}, r) = \mathcal{P}_{X_f}(g^{(0)}, g^{(l)}, r) + \frac{\partial}{\partial t}\mathcal{P}_{X_f}(g^{(0)}, \mathcal{S}[t, f]D[\theta_l h]g^{(l)}, r)\Big|_{t=0} dt + O(dt^2),$$

where the first term in the right-hand-side follows the fact that $\mathcal{P}_{X_f}(g^{(0)}, D[\theta_l h]g^{(l)}, r)$ does not depend on θ_l. We would now like to know if modulating the signal at layer l improves the routing measure at layer $l + 1$. For that, it is enough to show that the derivative of $\mathcal{P}_{X_f}(g^{(0)}, g^{(l+1)}, r)$ with respect to θ_l is nonzero at $\theta_l = 0$. Observe that

$$\frac{\partial}{\partial \theta_l}\mathcal{P}_{X_f}(g^{(0)}, g^{(l+1)}, r) = \frac{\partial}{\partial \theta_l}\frac{\partial}{\partial t}\mathcal{P}_{X_f}(g^{(0)}, \mathcal{S}[t, f]D[\theta_l h]g^{(l)}, r)\Big|_{t=0} + O(dt^2).$$

Hence, our goal is to show that

$$\mathcal{D} := \frac{\partial}{\partial \theta_l}\frac{\partial}{\partial t}\mathcal{P}_{X_f}(g^{(0)}, \mathcal{S}[t, f]D[\theta_l h]g^{(l)}, r)\Big|_{t,\theta_l=0}$$

is nonzero in general. As long as this is true, $\theta_l = 0$ is not the minimizer of $\mathcal{P}_{X_f}(g^{(0)}, g^{(l+1)}, r)$, so one can always choose a better modulation than $\theta_l = 0$.

We now simplify the notation and give a formula for

$$\mathcal{D} := \frac{\partial}{\partial \theta}\frac{\partial}{\partial t}\mathcal{P}_{X_f}(g, \mathcal{S}[t, f]D[\theta h]g, r)\Big|_{t=\theta=0}.$$

Claim 21 (Mixed derivative of the signal routing measure)**.**

$$\frac{\partial}{\partial \theta}\frac{\partial}{\partial t}\mathcal{P}_{X_f}(g, \mathcal{S}[t, f]D[\theta h]g, r)\Big|_{t=\theta=0} = \frac{\langle [X_h, [\Delta, X_f^2]]g, g\rangle \; - \; 4r \operatorname{Re}\langle [X_h, W_f \nabla_f]g, g\rangle}{\mathcal{V}_{X_f}(g)}$$

We see that when h is constant, i.e., there is no modulation, $\mathcal{D}$ is zero.

Proof. Let $\phi_\theta = D[\theta h]g = e^{i\theta X_h}g$. Since both $D[\theta h]$ and X_f are diagonal, they commute. Consequently, $\mathcal{E}_{X_f}(\phi_\theta) = \mathcal{E}_{X_f}(g)$ and $\mathcal{V}_{X_f}(\phi_\theta) = \mathcal{V}_{X_f}(g)$ for every θ. Using Lemma 3, Theorem 7, and Theorem 19, we obtain

$$\frac{\partial}{\partial t}\mathcal{P}_{X_f}(g, \mathcal{S}[t, f]\phi_\theta, r)\Big|_{t=0} = \frac{\mathcal{E}_{i[\Delta_f, X_f^2]}(\phi_\theta) - 4r \operatorname{Re}\big(\langle i\nabla_f \phi_\theta, W_f \phi_\theta\rangle\big)}{\mathcal{V}_{X_f}(g)}.$$

We now differentiate with respect to θ. For any operator A and $\phi_\theta = e^{i\theta X_h}g$, we have

$$\frac{d}{d\theta}\langle A\phi_\theta, \phi_\theta\rangle = i\langle [A, X_h]\phi_\theta, \phi_\theta\rangle.$$

By applying this with $A = i[\Delta_f, X_f^2]$ and evaluating at $\theta = 0$, we obtain

$$\frac{d}{d\theta}\mathcal{E}_{i[\Delta_f, X_f^2]}(\phi_\theta)\Big|_{\theta=0} = \langle [X_h, [\Delta_f, X_f^2]]g, g\rangle.$$

For the second term, we can write

$$\langle i\nabla_f \phi_\theta, W_f \phi_\theta\rangle = \langle iW_f \nabla_f \phi_\theta, \phi_\theta\rangle.$$

We can use the same differentiation rule with $A = iW_f\nabla_f$ and get

$$\frac{d}{d\theta}\mathrm{Re}\big(\langle i\nabla_f\phi_\theta, W_f\phi_\theta\rangle\big)\Big|_{\theta=0} = \mathrm{Re}\left(i\langle[iW_f\nabla_f, X_h]g, g\rangle\right)$$
$$= \mathrm{Re}\big(\langle[X_h, W_f\nabla_f]g, g\rangle\big).$$

Substituting the two derivatives into the previous formula completes the proof. □

C.8 Sensitivity analysis of signal routing via Schrödinger operators

In this section, we propose a simple sensitivity analysis for Schrödinger Operators in our observable setting, to mirror the sensitivity analysis presented in oversquashing papers [23–25].

Suppose we have an input nonnegative-valued signal at some layer, which is not localized on one graph region, and we partition it into chunks. To keep the notations simple, we focus on one chunk, and denote it by $g^{(0)}$. Since the original signal has roughly the same amount of energy at all locations (it is not localized), we can roughly think about $g^{(0)}$ both as the signal's chunk and as a localization window about $\mathcal{E}_{X_{f_k}}(g^{(0)})$, with variance $\mathcal{V}_{X_{f_k}}(g^{(0)})$.

Suppose that we modulate the signal to $g_\theta^{(0)} = D[\theta h]g^{(0)}$, apply a Schödinger operator $\mathcal{S}[t, f]$ to obtain the output signal $\Phi(g^{(0)}) = \mathcal{S}[t, f]g_\theta^{(0)}$. We lastly apply a modulus nonlinearity, to obtain $\Psi(g^{(0)}) = \big|\Phi(g^{(0)})\big|$. Similarly to the input signal, we think of $\Psi(g^{(0)})$ both as the output signal and as a localization window about $\mathcal{E}_{X_{f_k}}(\Psi(g^{(0)}))$, with variance $\mathcal{V}_{X_{f_k}}(\Psi(g^{(0)}))$. For this interpretation, we suppose that the variance of $\Psi(g^{(0)})$ is not too high.

We would like to know how much the values of $\Psi(g^{(0)})$ in the graph region about $\mathcal{E}_{X_{f_k}}\big(\Phi(g^{(0)})\big)$ are sensitive to changes in the input values $g^{(0)}$ about $\mathcal{E}_{X_{f_k}}(g^{(0)})$. If the sensitivity is high, it indicates that much of the information from region $\mathcal{E}_{X_{f_k}}(g^{(0)})$ reached region $\mathcal{E}_{X_{f_k}}\big(\Psi(g^{(0)})\big)$. We formalize this sensitivity via the differential of Φ, which is also the standard approach in oversquashing analyses [23–25].

Differentials. To keep the analysis self-contained, we recall basic definitions related to differentials.

Given a function $\psi : \mathbb{R}^D \to \mathbb{R}^K$, and a point in the domain $x \in \mathbb{R}^D$, if there exists a linear operator $\mathcal{D}_x[\psi] : \mathbb{R}^D \to \mathbb{R}^K$ which satisfies for every $y \in \mathbb{R}^D$

$$\psi(x + dy) = \psi(x) + d\mathcal{D}_x[\psi]y + o(d),$$

this operator is called the *differential* of ψ at x. Here, the limit in the o notation is over $d > 0$. In view of the above formula, the function $y \mapsto \psi(x) + \mathcal{D}_x[\psi]y$ is seen as the linearization of ψ about x.

In typical calculus formulations, we compute partial derivatives. These are connected to differentials as follows. Denote the n'th standard basis element by $\delta^n = (0, \ldots, 0, 1, 0 \ldots, 0)$, i.e., δ^n has 1 at entry n and 0 elsewhere. Given $m \in [D]$, consider the linear functional $\lceil\delta^m\rceil : \mathbb{R}^D \to \mathbb{R}$ which operates on input vectors $v = (v_d)_{d=1}^D$ via $\lceil\delta^m\rceil(v) = v_m$. Now, the partial derivative of the m'th output coordinate of ψ with respect to the n'th input coordinate is exactly $\lceil\delta^m\rceil\mathcal{D}_x[\psi](\delta^n)$.

In general, we need not restrict ourselves to probing the differential using only standard basis elements. Instead, given a general vector $v \in \mathbb{R}^K$ one can consider a general linear functional $\lceil v\rceil$ defined by

$$\lceil v\rceil(w) = \langle w, v\rangle$$

for probing the output of the differential. Moreover, one can consider a general vector $w \in \mathbb{R}^D$ for probing the input. Now, the quantity $\lceil v\rceil\mathcal{D}_x[\psi](w)$ is interpreted as the partial derivative of the v component of the output of ψ with respect to the w component of the input.

Derivation of the routing sensitivity In our formulation, we are not interested in showing that the sensitivity between all pairs of nodes is high. Rather, we just want to show that the sensitivity between the starting region and the destination region is high. Indeed, our premise is that good GNNs should have the ability to route the signal information between each region of the graph to a *specific* corresponding destination region, relevant for the task. We suppose that the destination of region $\mathcal{E}_{X_{f_k}}(g^{(0)})$ is $\mathcal{E}_{X_{f_k}}\big(\Psi(g^{(0)})\big)$. Namely, we suppose that the correct modulation feature and time

t were already chosen for the task of interest. This greatly simplifies the analysis with respect to oversquashing analyses, and gives us exact sensitivity results rather than loose inequalities.

By linearity of Φ, we have

$$\lceil \Phi(g^{(0)}) \rceil \mathcal{D}_{g^{(0)}}[\Phi](g^{(0)}) = 1.$$

In words, scaling up the input of a linear operator scales up the output by the same value.

Now, by the fact that $\Psi = |\Phi|$ we also have

$$\lceil \Psi(g^{(0)}) \rceil \mathcal{D}_{g^{(0)}}[\Psi](g^{(0)}) = 1.$$

Indeed, denote $\nu = \Psi(g^{(0)})$, and observe that

$$\langle |\nu|, |\nu| \rangle = \sum_n |\nu_n| \, |\nu_n| = \sum_n \nu_n \overline{\nu_n} = \langle \nu, \nu \rangle .$$

As a result, the sensitivity of Ψ probed at $\Psi(g^{(0)})$ is equal to the sensitivity of Φ probed at $\Phi(g^{(0)})$. This simple observation shows that all of the energy in the window $g^{(0)}$ is mapped to the window $\Psi(g^{(0)})$ under Ψ.

D Tutorial: Schrödinger dynamics in classical quantum mechanics

Our method is inspired by classical quantum mechanics. In this section, we briefly review it. Throughout this subsection we take $f(x) = x$, let X_f be the position operator $(X_f g)(x) = x\, g(x)$, and choose the continuum analogue

$$\nabla_f = -\frac{\partial}{\partial x}, \quad \Delta = -\nabla_f^2 = -\frac{\partial^2}{\partial x^2},$$

With this convention, we have

$$\mathcal{E}_{i\nabla_f}(g) = \left\langle i\left(-\frac{\partial}{\partial x}\right) g,\ g \right\rangle = -i \int_{\mathbb{R}} g'(x) \overline{g(x)} dx.$$

Note that if g is real-valued, then $\mathcal{E}_{i\nabla_f}(g) = 0$. The free Schrödinger evolution is $g^{(t)} = e^{-it\Delta} g$.

Theorem 22 (Heisenberg equation of motion for expected values)**.** *For every observable A, we have*

$$\frac{\partial}{\partial t} \mathcal{E}_A(g^{(t)}) = i\langle [\Delta, A] g^{(t)}, g^{(t)} \rangle.$$

Proof. By definition, we have

$$\begin{aligned} \frac{\partial}{\partial t} \mathcal{E}_A(g^{(t)}) &= \frac{\partial}{\partial t} \langle A g^{(t)}, g^{(t)} \rangle \\ &= \langle A(-i\Delta) g^{(t)}, g^{(t)} \rangle + \langle A g^{(t)}, (-i\Delta) g^{(t)} \rangle \\ &= i\langle [\Delta, A] g^{(t)}, g^{(t)} \rangle. \end{aligned}$$

□

Theorem 23 (Expected position evolution in the classical case)**.** *Let X be multiplication by x and let $\Delta = -\partial_x^2$. Then,*

$$\frac{\partial}{\partial t} \mathcal{E}_X(g^{(t)}) = 2\, \mathcal{E}_{i\nabla}(g^{(t)}).$$

Moreover, $\mathcal{E}_{i\nabla}(g_t)$ is constant in t, and therefore

$$\mathcal{E}_X(g^{(t)}) = \mathcal{E}_X(g^{(0)}) + 2t\, \mathcal{E}_{i\nabla}(g^{(0)}).$$

Proof. By Theorem 22, we have

$$\frac{\partial}{\partial t} \mathcal{E}_X(g_t) = i\langle [\Delta, X] g_t, g_t \rangle.$$

For any smooth test function u, we have

$$[\Delta, X] u = -\partial_x^2 (xu) + x\partial_x^2 u = -2\, \partial_x u = 2i(i\partial_x u).$$

Therefore, $\frac{\partial}{\partial t} \mathcal{E}_X(g^{(t)}) = 2\, \mathcal{E}_{i\nabla}(g^{(t)})$. Similarly, since

$$[\Delta, i\nabla] = [-\partial_x^2, i\partial_x] = 0,$$

we have $\mathcal{E}_{i\nabla}(g^{(t)})$ is conserved by Theorem 22. Integrating in time gives the stated linear evolution of $\mathcal{E}_X(g^{(t)})$. □

For real-valued signals, the expected location remains constant under Schrödinger evolution, which motivates the need for modulation to achieve directional transport.

Theorem 24 (Modulation generates classical transport)**.** *Let $g, \widetilde{g} : \mathbb{R} \to \mathbb{R}$ be real-valued and define $g^{(0)}(x) = e^{i\theta\widetilde{g}(x)}g(x)$. Then*

$$\mathcal{E}_{i\nabla}(g^{(0)}) = -\theta \int \widetilde{g}'(x)\,|g(x)|^2\,dx,$$

and

$$\mathcal{E}_X(g^{(t)}) = \mathcal{E}_X(g) - 2t\theta \int \widetilde{g}'(x)\,|g(x)|^2\,dx.$$

Proof. We can directly compute

$$\begin{aligned}\mathcal{E}_{i\nabla}(g^{(0)}) &= \int i\partial_x(e^{i\theta\widetilde{g}(x)}g(x))\,\overline{e^{i\theta\widetilde{g}(x)}g(x)}\,dx \\ &= \int (ig'(x) - \theta\widetilde{g}'(x)g(x))g(x)\,dx \\ &= -\theta\int \widetilde{g}'(x)|g(x)|^2\,dx,\end{aligned}$$

since $\int ig'(x)g(x)\,dx = 0$ for real-valued g. The second identity then follows from Theorem 23. □

Theorem 25 (Time derivative of the position variance in the free Schrödinger case)**.** *Let $g^{(t)} = e^{-it\Delta}g$ with $\Delta = -\partial_x^2$. Then*

$$\frac{\partial}{\partial t}\mathcal{V}_X(g^{(t)}) = \mathcal{E}_{i[\Delta,X^2]}(g^{(t)}) - 4\,\mathcal{E}_X(g^{(t)})\,\mathcal{E}_{i\nabla}(g^{(t)}).$$

Since $\mathcal{E}_{i\nabla}(g^{(t)})$ is constant, it can also be written as

$$\frac{\partial}{\partial t}\mathcal{V}_X(g^{(t)}) = \mathcal{E}_{i[\Delta,X^2]}(g^{(t)}) - 4(\mathcal{E}_X(g^{(0)}) + 2t\,\mathcal{E}_{i\nabla}(g^{(0)}))\mathcal{E}_{i\nabla}(g^{(0)}).$$

Proof. By definition, we have

$$\mathcal{V}_X(g^{(t)}) = \mathcal{E}_{X^2}(g^{(t)}) - \mathcal{E}_X(g^{(t)})^2.$$

By differentiating and using Theorem 22 with $A = X^2$, we obtain

$$\frac{\partial}{\partial t}\mathcal{V}_X(g^{(t)}) = \mathcal{E}_{i[\Delta,X^2]}(g^{(t)}) - 2\mathcal{E}_X(g^{(t)})\frac{\partial}{\partial t}\mathcal{E}_X(g^{(t)}).$$

By Theorem 23, we have $\frac{\partial}{\partial t}\mathcal{E}_X(g^{(t)}) = 2\mathcal{E}_{i\nabla}(g^{(t)})$, which gives the first formula. The second is obtained by substituting the linear evolution of $\mathcal{E}_X(g^{(t)})$ and the conservation of $\mathcal{E}_{i\nabla}(g^{(t)})$. □

Relation of standard quantum-mechanical identities from our graph-specific constructions. Note that our contribution is to adapt the established formalism from quantum mechanic to graph signal processing and graph machine learning and to derive the resulting graph-specific observables, derivatives, Hamiltonians, commutator terms, and routing guarantees. Below, we distinguish standard quantum-mechanical identities from our graph-specific constructions.

Specifically, Theorem 7 is obtained by applying the standard Heisenberg equation for expectation values, also known as the general Ehrenfest relation, $\frac{d}{dt}\langle A\rangle_t = i\langle[H,A]\rangle_t$ in units $\hbar = 1$, to the graph Hamiltonian $H = \Delta_f$ and the feature-location observable $A = X_f$. Our Lemmas 15-16 then evaluate this commutator for the proposed feature derivative and obtain $\frac{d}{dt}\mathcal{E}_{X_f}(g^{(t)}) = 2\,\mathrm{Re}\langle i\nabla_f g^{(t)}, W_f g^{(t)}\rangle$. In the continuum free-particle setting $f(x) = x, \nabla_f = -\partial_x$, and $W_f[X_f, \nabla_f] = I$. The formula reduces to $\frac{d}{dt}\langle X\rangle_t = 2\langle P\rangle_t$, which is the free-particle Ehrenfest equation under the normalization $H = P^2$, corresponding to $\hbar = 1$ and $m = \frac{1}{2}$. Importantly, we remark that:

- Theorem 5 is a graph free-momentum conservation, which is an instance of the Heisenberg/Ehrenfest equation when the momentum observable commutes with the Hamiltonian

- Theorem 7 is a graph-specific application of the Heisenberg equation for expectation values and reduces to the standard Ehrenfest theorem for expected position in the continuum free-particle setting.
- Theorem 10 is a graph analogue of the momentum shift induced by a spatially varying phase.
- Theorem 11 is a multidimensional graph Ehrenfest relation whose additional cross terms arise because the learned graph directions do not satisfy the exact Cartesian commutation relations.
- Theorem 19 follows from applying the Heisenberg equation to the second position moment and differentiating the corresponding variance.
- Theorems 18 and 20 are graph-specific approximation bounds quantifying the deviations caused by graph smoothing and cross-feature non-commutation.

E Experimental details and additional experiments

All experimental runs were conducted on individual GPUs, specifically utilizing an NVIDIA L40S hardware.

E.1 Toy experiment: recovering orthogonal directions on a grid

To illustrate the effectiveness of position-momentum optimization (PMO), we consider a two-dimensional grid graph whose original node features are the Cartesian coordinates (x, y). We then replace this orthogonal pair by the correlated pair $(x, x + y)$ and run PMO on the resulting two channels. Since $(x, x + y)$ spans the same two-dimensional feature subspace as (x, y), the optimization should recover two approximately orthogonal directions. Figure 5 compares the correlated input features with the transformed features returned by PMO.

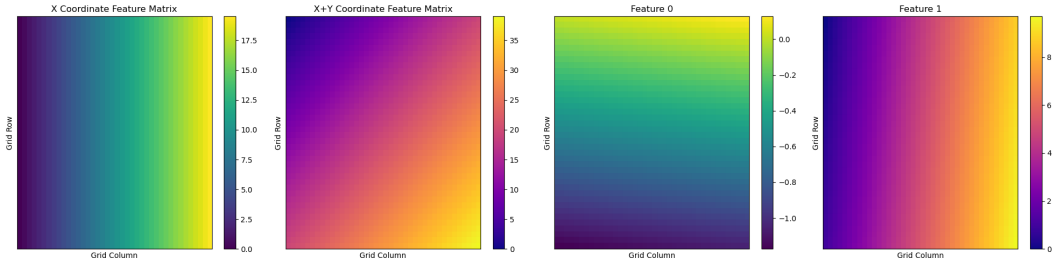

Figure 5: Grid-orthogonality toy experiment. The two left panels show the input features x and $x + y$. The two right panels show the transformed features after PMO. The recovered directions align with two approximately orthogonal axes on the grid.

E.2 Optimizing signal transport via modulation

In this subsection, we illustrate the effectiveness of signal transport under our modulation formulation. Specifically, we use a synthetic two-cluster graph to demonstrate that phase modulation can improve directed transport. We sample $N = 60$ nodes from two Gaussian clouds in $\mathbb{R}^2$: 30 nodes around $(-1, 0)$ and 30 nodes around $(1, 0)$, each with standard deviation 0.5 in both coordinates. Two nodes are connected by an undirected, unweighted edge whenever their Euclidean distance is smaller than 1.5. The feature location is the x-coordinate, denoted by f, and we use the same feature for modulation, namely $h = f$.

We choose a nonnegative initial signal that is concentrated on the left cluster and normalize it to unit ℓ_2 norm. The resulting signal is denoted by g. In the realization shown in Figure 2, this normalized signal satisfies $\mathcal{E}_{X_f}(g) = -0.98$, and the target location is set to $r = 1$.



Figure 6: Gaussian-translate learning curves. Lower is better. The modulated Schrödinger model converges to the lowest error among the compared models, outperforming the real-valued baselines and the non-modulated Schrödinger variant. The dashed line denotes the trivial predictor.

For each modulation parameter $\theta \in [-5, 5]$, we apply the modulation $D[\theta f]$, propagate by the Schrödinger operator $\mathcal{S}[0.1, f]$, and repeat this propagation three times. We track

$$\mathcal{E}_{X_f}(g) = \sum_{n=1}^{N} f(n)|g(n)|^2,$$
$$\mathcal{V}_{X_f}(g) = \mathcal{E}_{X_f^2}(g) - \mathcal{E}_{X_f}(g)^2,$$
$$\mathcal{P}_{X_f}(g^{(0)}, g^{(t)}, r) = \frac{\mathcal{V}_{X_f}(g^{(t)}) + \left(r - \mathcal{E}_{X_f}(g^{(t)})\right)^2}{\mathcal{V}_{X_f}(g^{(0)})}.$$

We then evaluate $\mathcal{E}_{X_f}(g_\theta^{(0.1)})$ and $\mathcal{P}_{X_f}(g, g_\theta^{(0.1)}, r)$. For visualization only, we renormalize $g_\theta^{(0.1)}$ to unit ℓ_2 norm and plot its magnitude. Figure 2 shows that an appropriate modulation parameter θ moves the signal toward the target location and lowers the routing measure relative to the unmodulated case.

E.3 Gaussian-translate toy experiment on a ring

We study a synthetic translation task on a ring graph. Given an input signal x on the ring, the model needs to predict the translated signal $y = S_d x$, where S_d denotes circular shift by d nodes. This task stresses whether a graph model can implement phase consistent transport on a simple topology.

E.3.1 Data

We use a ring graph with $N = 100$ nodes. Let $\theta_n = -\pi + 2\pi n/N$ denote the angular coordinate of node n. Each sample is generated by drawing a variance $\sigma^2 \sim \mathcal{U}[0.5, 1.5]$, sampling a Gaussian signal on the ring, adding Gaussian noise with standard deviation 10^{-3}, applying a random circular shift, and normalizing the result to unit ℓ_2 norm. The target is $y = S_d x$ with a fixed shift $d = 35$. We use an 80/10/10 train/validation/test split and batch size 32.

E.3.2 Models

We compare GCN and GAT baselines with Schrödinger variants that approximate a unitary graph flow through a truncated exponential. Let A be the aggregation operator on the cycle and define the skew-Hermitian generator $H = i\,A$. Each Schrödinger layer applies a learnable linear map W followed by the truncated series $z \leftarrow \sum_{k=0}^{T} \frac{(\delta \mathcal{H})^k}{k!} Wz$ with $T = 15$, where δ is a learned per-channel scale. All Schrödinger models use feature normalization after each layer and a magnitude nonlinearity. The modulated variant additionally learns a phase direction $m = \mathrm{Linear}([x, \theta])$ and multiplies the features by $e^{i\epsilon m}$ with $\epsilon = 25$.

E.3.3 Training

For a sample (x_i, y_i), we minimize the L_2 loss $\|\hat{y}_i - y_i\|_2$. We train for 250 epochs with Adam [73], using a base learning rate of 0.1, a $10\times$ larger learning rate for the modulation parameters, and a ReduceLROnPlateau scheduler with factor 0.7 and patience 10. Figure 6 reports smoothed test curves for readability and the dashed line denotes a naive baseline. In Table 5, we report the parameter counts for the Schrödinger GNN and the competing methods.

Table 5: Parameter counts for the Gaussian-translate models on a ring with $N = 100$ and shift $d = 35$. This table reports model size only, not predictive performance.

Model	Params
GCN	2,136
GAT	6,193
Schrödinger non modulated	4,273
Schrödinger	4,275

E.4 MNIST graph classification

We conduct an experiment on the classical MNIST dataset [74] to evaluate our model's performance on a standard image classification task formulated as a graph problem.

Dataset construction. We convert each 28×28 MNIST image into a graph with $N = 784$ nodes, one node per pixel. We add undirected edges using 8-neighbor grid connectivity (Chebyshev radius 1). Each node v_i carries the feature vector $\mathbf{x}_i = [x_{\text{norm}}, y_{\text{norm}}, I]$, where $x_{\text{norm}}, y_{\text{norm}} \in [0, 1]$ are normalized pixel coordinates and $I \in [0, 1]$ is the pixel intensity. We use the standard MNIST split of 60,000 training images and 10,000 test images [74], and we report averages over five random seeds.

Baselines. We compare against five standard GNN baselines: GCN [2], GAT [38], GIN [40], MPNN [75], and ChebConv [55]. As an additional non-graph reference, we train a classical 2D CNN [76] directly on the raw 28×28 images. For comparability, the CNN uses the same optimizer, learning rate, dropout, and overall width/depth scale whenever applicable.

Training hyperparameters. All graph models use hidden dimension 64, depth 3, batch size 16, dropout 0.1, and global mean pooling. We train for 200 epochs with Adam and learning rate $\alpha = 3 \times 10^{-4}$.

Classification performance. Table 6 reports the classification performance. We can see that our Schrödinger model achieves competitive performance.

Table 6: MNIST classification performance.

Model	Test Accuracy (↑)
GCN	$92.09_{\pm 0.28}$
ChebConv	$95.72_{\pm 0.74}$
GAT	$95.94_{\pm 0.71}$
GIN	$98.33_{\pm 0.11}$
MPNN	$98.95_{\pm 0.06}$
CNN	$99.07_{\pm 0.07}$
Schrödinger	$99.13_{\pm 0.04}$

E.5 Graph classification on TU dataset

In this subsection, we present the experimental setup for the architecture-matched comparison on ENZYMES, IMDB-BINARY, MUTAG, and PROTEINS from the TU dataset collection [39]. The

dataset is split into 50% for training, 25% for validation and 25% for testing. The corresponding results are reported in Table 2.

Table 7: Statistics of the TU graph-classification datasets used in the architecture-matched comparison.

	Enzymes	IMDB	MUTAG	PROTEINS
# Graphs	600	1000	188	1113
# Nodes (range)	2 - 126	12 - 136	10 - 28	4 - 620
# Edges (range)	2 - 298	52 - 2498	20 - 66	10 - 2098
Avg # Nodes	32.63	19.77	17.93	39.06
Avg # Edges	124.27	193.06	39.58	145.63
# Classes	6	2	2	2
Directed	False	False	False	False
ORC Mean	0.13	0.58	-0.27	0.17
ORC Std	0.15	0.19	0.05	0.20

Architectural constraints. To compare architectures under a common capacity budget, we use a standardized backbone with six graph convolution layers followed by one linear classifier. We first define a reference parameter budget using the Unitary (UniGCN) architecture with a hidden dimension of 128. For the remaining models (GAT, GCN, GIN, Adaptive Unitary, Schrödinger, and Schrödinger PMO), we then adjust the hidden dimension so that the total number of trainable parameters falls within 0.6% of this reference budget. This isolates differences in the propagation operator as much as possible. For complex-valued models, each complex parameter is counted as two real parameters.

Hyperparameter search. For each model–dataset pair, we search over learning rate $\{0.0005, 0.001, 0.005, 0.01\}$ and dropout $\{0, 0.25, 0.5\}$. The search space follows the settings used in prior evaluations [41, 77]. The chosen configuration is the one with the best mean validation accuracy over 100 runs. The chosen values are summarized in Table 8.

Table 8: Hyperparameters for the architecture-matched comparison.

Model	Hyperparameter	ENZYMES	IMDB	MUTAG	PROTEINS
GCN	Learning Rate	0.005	0.005	0.005	0.001
	Dropout	0	0	0	0
	Hidden Dimension	190	190	190	190
GAT	Learning Rate	0.001	0.001	0.0005	0.005
	Dropout	0	0.5	0	0.5
	Hidden Dimension	189	189	189	189
Schrödinger	Learning Rate	0.005	0.0005	0.005	0.005
	Dropout	0.25	0	0.25	0
	Hidden Dimension	117	170	170	170
Schrödinger PMO	Learning Rate	0.005	0.001	0.01	0.005
	Dropout	0	0	0	0
	Hidden Dimension	117	117	117	117
Unitary	Learning Rate	0.001	0.001	0.001	0.0005
	Dropout	0	0	0	0
	Hidden Dimension	128	128	128	128
Adaptive Unitary	Learning Rate	0.005	0.0005	0.005	0.001
	Dropout	0	0	0	0
	Hidden Dimension	127	127	127	127
Adaptive Unitary PMO	Learning Rate	0.001	0.01	0.001	0.001
	Dropout	0.25	0	0	0
	Hidden Dimension	127	127	127	127
GIN	Learning Rate	0.001	0.005	0.01	0.0005
	Dropout	0	0	0	0
	Hidden Dimension	190	190	190	190

Runtime comparison. Table 9 reports the mean and standard deviation of the training time per epoch for each model on the TU datasets.

Table 9: Training time per epoch on TU datasets (seconds, mean $\pm$ std).

Model	ENZYMES	IMDB	MUTAG	PROTEINS
GCN	33.4 ± 7.85	27.5 ± 5.66	19.0 ± 4.30	33.9 ± 2.02
GAT	61.5 ± 15.51	42.6 ± 0.52	14.3 ± 2.83	65.7 ± 8.13
GIN	39.8 ± 5.41	32.9 ± 9.12	9.1 ± 1.52	43.2 ± 10.40
Unitary	216.7 ± 6.25	261.9 ± 72.57	60.4 ± 14.47	189.3 ± 6.79
Adaptive Unitary	200.1 ± 27.03	285.5 ± 48.41	47.5 ± 12.85	202.2 ± 36.36
Adaptive Unitary PMO	84.3 ± 15.91	142.9 ± 7.83	45.0 ± 0.67	158.0 ± 0.82
Schrödinger	172.8 ± 12.89	255.9 ± 55.36	44.4 ± 13.06	247.6 ± 47.71
Schrödinger PMO	173.5 ± 25.11	279.1 ± 67.42	68.6 ± 8.54	258.5 ± 30.49

Ablation study on ENZYMES dataset We use the ENZYMES dataset to isolate the contribution of three ingredients in the Schrödinger framework: learnable propagation time, phase modulation, and PMO preprocessing. All models share the same backbone with hidden dimension 128, learning rate 0.005, and 300 training epochs. Results are averaged over 100 independent runs. The sequence in Table 10 reports a cumulative ablation from the UniGCN baseline to the full Schrödinger PMO model.

Table 10: Ablation study on ENZYMES (test accuracy $\pm$ std). All models share the same backbone architecture and optimization settings.

Model	Test Accuracy
Unitary (UniGCN)	37.33 ± 8.25
Adaptive Unitary	41.56 ± 5.67
Schrödinger	43.61 ± 4.58
Schrödinger PMO	44.83 ± 4.03

Evaluation under 10-fold cross-validation. In Section 4, we provide a controlled architecture-matching setup that holds depth and capacity fixed while changing the propagation operator. Here, we included the 10-fold cross-validation for TUDataset with layer $L \in \{2, 3, 4, 5, 6\}$ for GCN, GAT, GIN, UniGCN, and Schrödinger GNN. We report the best performance of each method. These TU results in Table 11 support the competitiveness of Schrödinger propagation under a standard cross-validation protocol.

Table 11: Classification accuracy (%) on TU datasets under 10-fold cross-validation.

	ENZYMES	IMDB	MUTAG	PROTEINS
GIN	53.28 ± 5.31 ($L = 4$)	73.34 ± 4.02 ($L = 3$)	86.49 ± 4.95 ($L = 3$)	74.69 ± 3.64 ($L = 4$)
GCN	48.03 ± 3.92 ($L = 3$)	71.29 ± 4.86 ($L = 2$)	83.54 ± 5.67 ($L = 3$)	75.31 ± 5.81 ($L = 3$)
GAT	47.65 ± 5.08 ($L = 2$)	68.76 ± 3.42 ($L = 2$)	79.70 ± 9.45 ($L = 2$)	76.14 ± 3.92 ($L = 2$)
UniGCN	59.97 ± 7.25 ($L = 6$)	72.32 ± 3.64 ($L = 4$)	80.63 ± 9.96 ($L = 4$)	72.82 ± 4.33 ($L = 6$)
Schrödinger GNN (ours)	63.87 ± 5.11 ($L = 5$)	74.45 ± 3.85 ($L = 3$)	88.24 ± 6.24 ($L = 3$)	77.56 ± 4.02 ($L = 5$)

E.6 Heterophilous node classification

Dataset statistics. Table 12 summarizes the statistics of the heterophilous node-classification datasets.

Architecture and hyperparameters. We follow the experimental setup as in [41]. For our Schrödinger model, we first run PMO on the input features for a 50 batch size with learning rate 10^{-3}, and 2000 epochs. We set the number of layers as $\{4, 6, 8, 10\}$ with hidden dimension 512. The dropout is set as $\{0.2, 0.5\}$. A final linear layer maps the node embeddings to class logits. The performance is reported with the best results.

Table 12: Statistics of the heterophilous node-classification datasets from [42].

	ROMAN-EMPIRE	AMAZON-RATINGS	MINESWEEPER	TOLOKERS	QUESTIONS
#Nodes	22,662	24,492	10,000	11,758	48,921
#Edges	32,927	93,050	39,402	519,000	153,540
#Classes	18	5	2	2	2
Homophily	0.05	0.38	0.68	0.59	0.84
Metric	AP	AP	ROC AUC	ROC AUC	ROC AUC

E.7 Long range graph benchmark

We follow the experimental setup and training scheme as in [53, 41].

E.7.1 Peptides

Edge features. The current unitary Schrödinger block does not natively consume edge attributes. For edge-attributed molecular graphs, we use an edge-feature adapter before the Schrödinger propagation. In the Peptides runs, this adapter is GINE [40].

Resources, parameter count, and search space. In our implementation, Peptides epochs typically took less than 15 seconds, and the dataset storage footprint was about 1GB. Following the LRGB parameter-budget convention, models are kept around roughly 500K trainable real-valued parameters. We also follow the hyperparameter settings as in [52, 41]. Each complex parameter is counted as two real-valued parameters. We use Adam [73] with an initial learning rate of 0.001 and a cosine learning-rate scheduler. The sweep for the basic model varies the number of layers in $\{6, 8, 10, 12, 14\}$, dropout in $\{0.1, 0.15, 0.2\}$, batch size 200, with maximal 4000 epochs, and hidden dimension in $\{135, 155, 175, 195, 200\}$.

E.7.2 PascalVOC-SP and COCO-SP

We evaluate on the PascalVOC-SP and COCO-SP datasets from the Long Range Graph Benchmark (LRGB) [44]. Both datasets are superpixel-based computer vision benchmarks formulated as node classification tasks, and performance is reported using the official F1 metric. We follow the experimental protocol of [41] for these two datasets. We train with the Adam optimizer, set the initial learning rate to 10^{-3}, use a cosine learning-rate scheduler, and a batch size of 50.

For models that do not directly incorporate edge attributes in the message-passing layer, we follow [41] and prepend a single GatedGCN layer as an edge-feature aggregator, which maps edge information into the node representations before applying the main network. We use the same positional encoding as in the reference setup: no additional PE/SE is used for COCO-SP, while random-walk structural encodings (RWSE) [47] are used for PascalVOC-SP.

Hyperparameters were tuned by sweeping the dropout rate over $\{0.1, 0.15, 0.2\}$ and the number of layers over $\{6, 8, 10, 12, 14\}$. For each depth, the hidden dimension was selected so that the model satisfied the LRGB parameter budget of approximately 500K trainable parameters. When applicable, complex-valued parameters were counted as two real-valued parameters. We trained COCO-SP models for up to 750 epochs and PascalVOC-SP models for up to 1000 epochs. Hyperparameters were selected based on validation performance, and we report the test-set F1 score of the selected configuration.

E.7.3 Signal-flow diagnosis

We describe the diagnostic used in Figure 4. The purpose of this experiment is to measure whether the hidden signal of a trained model is transported across the graph by individual layers. All models are first trained on PascalVOC-SP using the standard training split and selected by validation performance. After training, all parameters are frozen and the diagnostic is run in evaluation mode, with dropout disabled. The diagnostic uses only hidden activations and location features.

Let $\mathcal{G} = (V, E)$ be a PascalVOC-SP superpixel graph with $|V| = N$ nodes. Let $f = (f_1, \dots, f_K) : V \to \mathbb{R}^K$ denote the location features used for the diagnostic. In the PascalVOC-SP experiments,

we use the two superpixel-center coordinates, $K = 2$, corresponding to the normalized horizontal and vertical coordinates. The same location features are used for all compared models. For a hidden representation $H \in \mathbb{C}^{N\times d}$, or $H \in \mathbb{R}^{N\times d}$ for real-valued models, we compute its node energy and for each location coordinate f_k, we compute the expected location of H. This graph-dependent standard deviation is used only for normalization of the reported shift.

Window construction. To measure local signal transport, we split the hidden signal into windows according to the location features. For each coordinate f_k, we construct a collection of nonnegative windows $\{w_{b,k} : V \to [0,1]\}_{b=1}^{B}$ localized at different values of f_k. These windows are normalized to form a partition of unity along each coordinate $\sum_{b=1}^{B} w_{b,k}(v) = 1$ for all $v \in V$. For two-dimensional PascalVOC-SP coordinates, a window is indexed by $\alpha = (b_1, b_2)$ and is defined by the product window $w_\alpha(v) = w_{b_1,1}(v)w_{b_2,2}(v)$. Given a hidden representation H, the corresponding windowed hidden signal is $H_\alpha(v,:) = \sqrt{w_\alpha(v)}\, H(v,:)$ and we normalize it.

Layerwise relative shift. Let Φ_ℓ denote the ℓ-th trained layer of a model, including its graph propagation, channel mixing, normalization, and activation operations. For a model with hidden states $H^{(0)}, H^{(1)}, \ldots, H^{(L)}$, we apply the ℓ-th layer to each windowed input signal $\widetilde{H}_\alpha^{(\ell)} = \Phi_\ell\Big(H_\alpha^{(\ell-1)}\Big)$ then compute the relative shift. The quantity plotted in Figure 4 is the average of the relative shifts over all nonempty windows and all graphs in the PascalVOC-SP evaluation split.

E.7.4 Additional competing methods

we additionally add comparisons with SWAN [78] and Stable-ChebNet [79] to the two LRGB peptide tasks, Peptides-Func and Peptides-Struct. We see that our Schrödinger GNN obtains better results on both tasks compared to both mehtods.

Table 13: Performance comparison on the Peptides benchmarks.

	Peptides-Func (↑)	Peptides-Struct (↓)
SWAN	0.6715 ± 0.0039	0.2485 ± 0.0009
Stable-ChebNet	0.7032 ± 0.0026	0.2542 ± 0.0030
Schrödinger GNN (ours)	0.7207 ± 0.0099	0.2439 ± 0.0011

E.8 Ablation study on unitary Schrödinger propagation, complex modulation and PMO

Here, we conduct a systematic $2 \times 2 \times 2$ factorial ablation on PascalVOC-SP and Roman-Empire for analyzing the effect of unitary Schrödinger propagation, complex modulation and PMO.

Table 14 shows that the each component of our model contributed to the performance. Specifically, modulation has the larger effect on PascalVOC-SP, which is in line with the ring experiment and Figure 4 that modulation is the principal mechanism producing directional transport. In addition, while PMO provides a consistent additional improvement, the model containing unitary propagation and complex modulation but without PMO obtains strong performance. Thus, we interpret PMO as a useful conditioning and refinement component.

This factorial ablation supports three conclusions: (i) unitary propagation and complex modulation each provide an independent benefit, (ii) their combination provides an additional complementary gain, and (iii) the core model remains effective under a single shared hyperparameter configuration even without PMO.

E.9 Sensitivity to the learned feature locations

Here, we evaluate the sensitivity to the learned feature locations on PascalVOC-SP and Roman-Empire by measuring a normalized non-commutation score $\widetilde{\varepsilon} = \frac{1}{K(K-1)} \sum_{i\neq j} \frac{||\nabla^2_{f_j}, X_{f_i}||_{\mathrm{op}}}{||\nabla^2_{f_j}||_{\mathrm{op}}||X_{f_i} - \bar{f}_i I||_{\mathrm{op}} + 1e^{-12}}$, the dynamics residual $R_{\mathrm{dyn}} = |\frac{d}{dt}\mathcal{E}_{X_{f_k}}(g(t)) - 2\mathcal{E}_{i\nabla_{f_k}}(g(t))|$, the relative transport shift S_{rel}, variance growth $V_{\mathrm{var}} = \frac{V_{out}}{V_{in}}$ where V_{out} and V_{in} are the variance of the output and the input from the trained layers, and the variance for a normalized signal is $V_{X_{f_k}}(g)$, and final task performance.

Table 14: Ablation study on the effects of unitary Schrödinger propagation, complex modulation, and PMO.

Unitary Schrödinger propagation	Complex modulation	PMO	PascalVOC-SP	Roman-Empire
×	×	×	0.2803	86.67
×	×	✓	0.2887	86.80
×	✓	×	0.3042	87.35
×	✓	✓	0.3129	87.63
✓	×	×	0.3095	87.40
✓	×	✓	0.3142	87.45
✓	✓	×	0.3507	88.56
✓	✓	✓	0.3532	88.62

Table 15 shows that non-commutation is strongly positively correlated with the dynamics residual, which is consistent with Theorem 20. Higher non-commutation is also associated with smaller relative shifts and greater variance growth. Thus, lower non-commutation corresponds to signal transport that moves farther along the intended feature direction while remaining more concentrated. Non-commutation is moderately negatively correlated with final performance while relative shift is positively correlated with performance. The weaker relationship with task performance is expected as predictive accuracy also depends on the informativeness of the locations and the downstream model. The consistent signs across both datasets show that commutation is an empirically measurable indicator of accurate dynamics and coherent signal transport.

Table 15: Correlation analysis on PascalVOC-SP and Roman-Empire.

Correlation	PascalVOC-SP	Roman-Empire
$\rho(\widetilde{\varepsilon}, R_{\mathrm{dyn}})$	$+0.86$	$+0.73$
$\rho(\widetilde{\varepsilon}, S_{\mathrm{rel}})$	-0.61	-0.46
$\rho(\widetilde{\varepsilon}, V_{\mathrm{var}})$	$+0.58$	$+0.42$
$\rho(\widetilde{\varepsilon}, \mathrm{performance})$	-0.42	-0.34
$\rho(S_{\mathrm{rel}}, \mathrm{performance})$	$+0.59$	$+0.45$

E.10 Relation between the core mechanisms, PMO, commutation, transport, and performance

We remark that we can distinguish our main mechanisms into four different roles, organized into two levels: (i) core transport mechanisms and (ii) supporting coordinate-conditioning analysis:

1. Complex modulation creates nonzero feature momentum and makes directed transport possible.
2. Unitary Schrödinger propagation transports the modulated signal while avoiding the contraction associated with diffusive propagation.
3. Approximate commutation makes the dynamics along different learned feature coordinates more nearly decoupled and therefore makes the coordinate-wise theoretical description more accurate.
4. PMO conditions the feature-location system by reducing cross-coordinate interference. It does not create momentum or directly optimize the downstream task loss.

The first two are the core architectural mechanisms. Approximate commutation is instead a sufficient condition for clean multi-feature dynamics, and PMO is one possible preprocessing procedure for improving that condition. This hierarchy is consistent with the paper's intended claim, which identifies the unitary GSO and complex modulation as the two main components of the Schrödinger GNN.

Indeed, good communication is strongly correlated with how clean the transport dynamics are. It is also positively correlated with performance, but not as strongly. This is natural. Our theory supports commutation being an important component contributing to clean dynamics. Clean dynamics is related to coherent transportation of the signal between graph regions (transportation without much distortion). Since in long range problems, transportation between graph regions is important, commutation is correlated with accuracy, but less so. The reason there is a drop in correlation is that high accuracy can

be achieved in a number of ways. First, it may be the case that in some tasks, even those categorized as long range tasks, there are workarounds that allow standard local GNNs to also find a good or reasonable solution. Hence, long range transport capabilities are not the only component responsible for performance. For this reason, we propose the synthetic experiment of signal propagation on a ring (see Section 4.1), where long range propagation of the signal is more clearly important for the task. Secondly, Schrödinger GNNs with poor commutation can still transport the signal across the graph, but the signal may "lose its coherence" when transported, i.e., it may be much distorted. Still, the ability to transport the signal across graph regions, even without keeping the signal coherent, may be a "good enough capability" for roughly solving long range problems. Hence, some of the performance is attributed to local interactions, some to long range "incoherent" interactions, and the rest of the performance is due to the coherence of the transportation. For this reason, the correlation between commutation and performance is not as high as the correlation between commutation and coherence of the dynamics.

We additionally include a matched analysis for every configuration in the factorial experiment from Appendix E.8, where in each configuration, we report the non-commutation score $\widetilde{\varepsilon}$, the relative transport shift S_{rel}, variance growth V_{var}, feature-probe performance (fixed MLP predictor trained only on the frozen feature-location representation), and final task performance. For the configurations using Schrödinger propagation, we additionally report two separate dynamics quantities: (i) dimensionless normalized cross-coordinate residual $R_{\mathrm{cross}} = \frac{\sum_k \sum_{j\neq k} \left|\left\langle [i\nabla^2_{f_j}, X_{f_k}]g, g\right\rangle\right|}{\sum_k \left(\left|2\,\mathrm{Re}\left\langle i\nabla_{f_k} g, W_{f_k} g\right\rangle\right| + \eta\right)}$, and (ii) the same-coordinate regularity residual $\delta_{\mathrm{reg}} = \frac{1}{K}\sum_k \frac{||W_{f_k} g - g||_2}{||g||_2}$.

Table 16: Ablation analysis and diagnostic metrics on PascalVOC-SP.

Unitary	Modulation	PMO	$\widetilde{\varepsilon}$ (↓)	R_{cross} (↓)	δ_{reg} (↓)	S_{rel} (↑)	V_{var} (↓)	Feature probe (↑)	Performance (↑)
×	×	×	0.245	-	-	0.018	0.62	0.190	0.2803
×	×	✓	0.102	-	-	0.020	0.56	0.191	0.2887
×	✓	×	0.249	-	-	0.057	0.53	0.190	0.3042
×	✓	✓	0.118	-	-	0.064	0.46	0.192	0.3129
✓	×	×	0.257	0.190	0.198	0.032	0.23	0.191	0.3095
✓	×	✓	0.121	0.082	0.191	0.035	0.20	0.191	0.3142
✓	✓	×	0.262	0.221	0.232	0.136	0.16	0.190	0.3507
✓	✓	✓	0.104	0.091	0.225	0.144	0.13	0.191	0.3532

Table 17: Ablation analysis and diagnostic metrics on Roman-Empire.

Unitary	Modulation	PMO	$\widetilde{\varepsilon}$ (↓)	R_{cross} (↓)	δ_{reg} (↓)	S_{rel} (↑)	V_{var} (↓)	Feature probe (↑)	Performance (↑)
×	×	×	0.218	-	-	0.012	0.46	65.8	86.67
×	×	✓	0.112	-	-	0.013	0.42	66.1	86.80
×	✓	×	0.220	-	-	0.034	0.39	65.7	87.35
×	✓	✓	0.123	-	-	0.038	0.35	66.3	87.63
✓	×	×	0.224	0.171	0.185	0.025	0.21	65.9	87.40
✓	×	✓	0.116	0.091	0.179	0.027	0.19	66.4	87.45
✓	✓	×	0.235	0.193	0.214	0.069	0.15	66.2	88.56
✓	✓	✓	0.119	0.101	0.207	0.074	0.13	66.8	88.62

Tables 16-17 shows the results of the diagnostics for PascalVOC-SP and Roman-Empire. We see that PMO reduces normalized non-commutation in these comparisons, and thus, PMO consistently performs its intended geometric role across both datasets and across all settings of the other two components. In addition, we notice that the reduction in $\widetilde{\varepsilon}$ is accompanied by a comparably large reduction in R_{cross}. As such, the close correspondence between the reductions in $\widetilde{\varepsilon}$ and R_{cross} supports the interpretation of approximate commutation as a meaningful predictor of cross-coordinate interference in the realized dynamics. Further, we observe that the regularity residual changes much less, and PMO substantially reduces the cross-coordinate part of the approximation error while leaving the separate same-coordinate regularity contribution largely unchanged. Interestingly, PMO improves the transport metrics, although the changes are more modest than the changes in non-commutation, and PMO consistently produces greater displacement with less spreading, which supports that lower non-commutation is associated with more coherent transport, in line with the correlations reported in Appendix E.9. Finally, we note that the feature-probe results show a smaller and dataset-dependent effect. Specifically, PMO can yield a modest improvement in feature informativeness, but this

change remains much smaller than the performance difference attributable to modulation and unitary propagation in the factorial experiment. These results support that geometric cleanliness and feature informativeness are related but distinct. The observed feature-probe means increase modestly on Roman-Empire and remain nearly unchanged on PascalVOC-SP.

It is important to note that the sensitivity experiment provides a complementary result. Non-commutation is strongly correlated with the dynamics residual, but only moderately correlated with final performance. This difference is expected for several reasons:

1. Theorem 20 is a dynamical approximation bound, not a task-loss bound. It states that small cross-commutators with signal regularity are sufficient for the expected location along each coordinate to approximately follow its corresponding momentum. It does not state that minimizing the commutator directly maximizes classification performance, nor that larger commutators necessarily imply poor prediction.
2. PMO substantially reduces only one of the two theoretical error sources. In the complete PascalVOC-SP configuration, it mainly reduces R_{cross}, but only marginally reduces δ_{reg}. The analogous reductions on Roman-Empire appear similarly. Thus, PMO cleans the cross-coordinate geometry while leaving another source of approximation error almost unchanged.
3. PMO does not create task-directed momentum. The modulation parameters determine whether the current signal acquires momentum in a useful direction, and the unitary propagator determines whether this momentum produces coherent transport. PMO only reduces interference among the coordinates used to describe that motion. This explains why modulation and unitary propagation have substantially larger effects on final performance.
4. The factorial results suggest diminishing returns and PMO produces its largest PascalVOC-SP gains when one or both core mechanisms are absent. Once modulation and unitary propagation have already established effective transport, further cleaning of the feature-coordinate geometry yields only a small additional predictive gain.

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: We presented a new approach for defining and analyzing signal propagation across graphs. The approach directly models where the information of the signal is, how well concentrated it is, and how well it is routed between regions in the graph. We presented Schrödinger GNN, a graph neural network that is able to route the information of the signal along any direction in the graph. The main contributions are summarized in §1.

   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 discuss the limitation of our method in Section 5.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: The proof of our theoretical results is in Appendix C. We also include several additional propositions and lemmas.

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: Appendix E presents detailed methodological descriptions that enable re-implementation of the proposed method and reproduction of the reported results.

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 detailed descriptions of our method's implementation and hyperparameters in Appendix E. Our source code is at: https://github.com/edenagar/SGCN

   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 provide detailed information on experimental setup, dataset, architecture construction, and additional empirical results in §E.

   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: We report the mean and the stadard deviation of each experiments in Section 4 and Appendix E.

   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: Detailed descriptions of the type of computer resources and time of execution are provided in Appendix E.

   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: We respect the NeurIPS code of Ethics.

   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: [N/A]

    Justification: This paper presents work whose goal is to advance the field of graph neural networks. The paper has no foreseeable societal impact.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All external datasets, code repositories, and pretrained models used in this work are properly credited to their respective original papers and authors.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: The new assets introduced in the paper are well documented. Detailed descriptions of our method's implementation are in Appendix E.

    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: No crowdsourcing nor research with human subjects was involved in this work.

    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 nor research with human subjects was involved in this work.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: LLMs were used only for editing purposes, such as for grammar, spelling, and word choice.

    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.