Classifying the Graph Topology of Non-Hermitian Energy Spectra with Graph Transformer

Xianquan Yan[1,2] **Hakan Akgün**[1] **Kenji Kawaguchi**[2]
N. Duane Loh[3,4*] **Ching Hua Lee**[1†]
[1]Department of Physics, National University of Singapore, Singapore 117551
[2]Department of Computer Science, National University of Singapore, Singapore 117417
[3]Department of Biological Sciences, National University of Singapore, Singapore 117558
[4]NUS Centre for Bioimaging Sciences, National University of Singapore, Singapore 117557

Abstract

Classifying non-Hermitian energy spectra under open boundary conditions is an open challenge in physics. This classification is a critical prerequisite for the rational inverse design of systems exhibiting desired dynamics and topological responses. While graph topology has emerged as a promising approach for characterizing these spectra, systematic methods for distilling non-Hermitian spectra into their corresponding graph representations have been lacking. Moreover, the resulting graphs often exhibit complexities that defy manual classification, necessitating machine learning approaches. In this work, we introduce a two-step framework for classifying non-Hermitian spectra based on their graph topologies. The first step employs `Poly2Graph`, an automated, high-performance pipeline that distills non-Hermitian spectra into *spectral graphs* suitable for graph neural networks (GNNs). The second step involves generating a large dataset of these spectral graphs and training a GNN for classification. We propose `GnLTransformer`, a novel architecture featuring dual channels that leverage line graphs to explicitly capture higher-order topological features. `GnLTransformer` achieves over 99% classification accuracy on our dataset, outperforming standard baselines by 32%. Notably, beyond conventional GNNs, `GnLTransformer` offers inherent explainability regarding higher-order topology. As a further contribution, we release a new *multi-graph* dataset comprising over 117K spectral graphs.

Code — https://github.com/sarinstein-yan/poly2graph

1 Introduction

Topological physics has revolutionized modern physics, traditionally focusing on eigenstate homotopy windings in Hermitian systems. Hermiticity, a core assumption in the Schrödinger equation, ensures probability conservation and the reality of energy eigenvalues.

However, real-world systems are frequently non-Hermitian due to dissipation, gain, or non-conservative forces (Ashida, Gong, and Ueda 2020; Lin et al. 2023). Non-Hermiticity is pervasive across diverse domains, including mechanical materials (Ghatak et al. 2020; Wang, Meng, and Chen 2023), biological systems (Sone et al. 2024; Nelson and Shnerb 1998; Shimokawa et al. 2013; Amir, Hatano, and Nelson 2016), financial markets (Kumar and Wilmott 2024; Gonzalez-Conde et al. 2023), and artificial neural networks (Zdeborová 2020; Kerg et al. 2019).

The confluence of topology and non-Hermiticity has significantly enriched topological physics, unveiling novel phenomena such as exceptional points (spectral winding) (Kawabata et al. 2019; Ding, Fang, and Ma 2022; Kawabata, Bessho, and Sato 2019).

In this study, we investigate a distinct, intricate, and largely unexplored aspect of non-Hermitian topology: the *graph topology of the non-Hermitian energy spectrum*, termed the **spectral graph**. Mathematically, this graph is derived as an algebro-geometric property of the system's **characteristic polynomial**.

Despite the ubiquity and diversity of non-Hermitian spectral graphs, a systematic methodology for studying the relationship between the characteristic polynomial and the resulting spectral graph has remained elusive. Three major challenges have hindered the classification and broader investigation of these graphs:

1. **Exponential complexity with polynomial degree range.** As the degree range[1] of the characteristic polynomial increases, the complexity of the spectral graph grows exponentially. The resulting topologies quickly become intractable and unidentifiable by human inspection (Figure 1a).
2. **The inverse problem is analytically insoluble and has no existing solution.** A key objective is the inverse problem: given a spectral graph, determine the **class** of its parent characteristic polynomial. Due to the irreversible and nonlinear operations involved in the algebro-geometric transformation from polynomial to graph, this problem defies explicit analytical solutions and currently has none.
3. **Identification of essential graph features.** Even when complex spectral graphs can be algorithmically identi-

*duaneloh@nus.edu.sg
†phylch@nus.edu.sg

[1]The degree range is the difference between the maximum and minimum monomial degrees; e.g., $P(z, E) = z^2 + z^{-1}$ has a degree range of three. A larger degree range may arise from denser or longer-range hopping, or more energy bands.

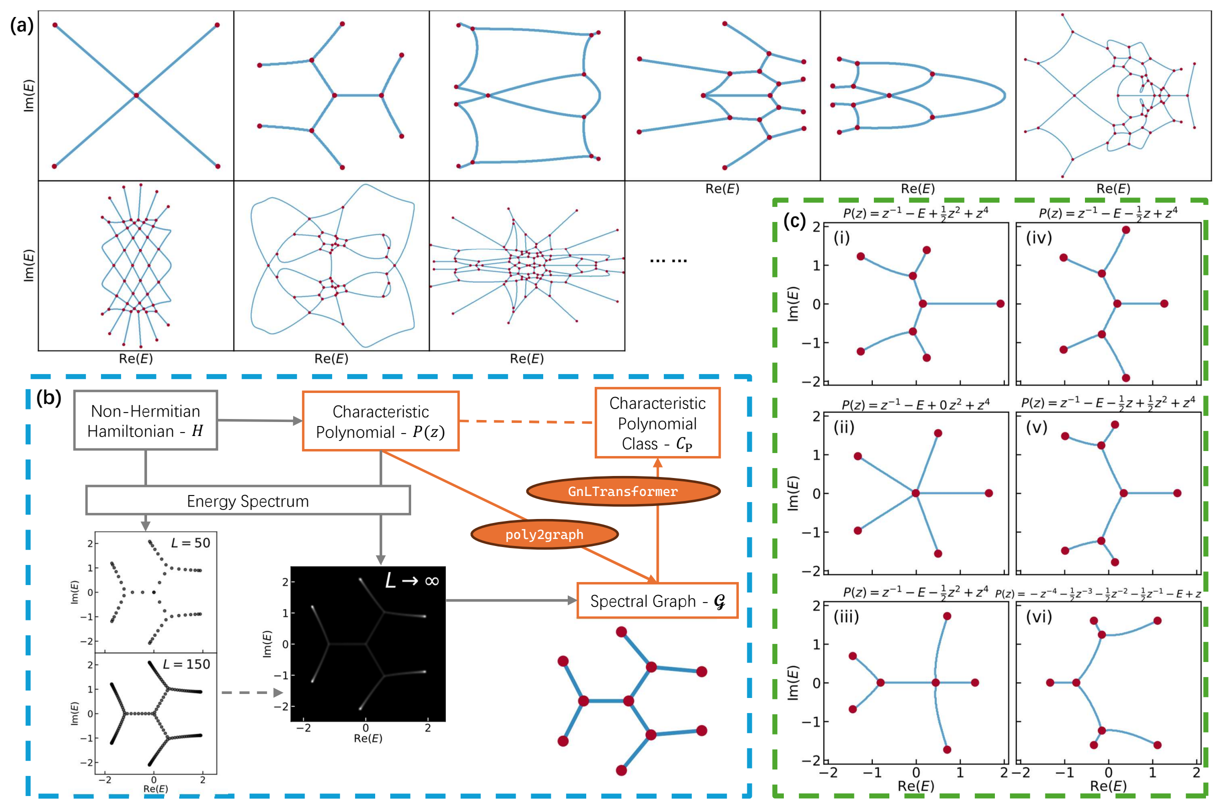
(a)
Im(E)
Re(E)
......
(b)
Non-Hermitian Hamiltonian - H
Characteristic Polynomial - P(z)
Characteristic Polynomial Class - C_P
Energy Spectrum
GnLTransformer
poly2graph
L = 50
L = 150
L → ∞
Spectral Graph - G
(c)
P(z) = z^{-1} − E + ½z² + z⁴ (i)
P(z) = z^{-1} − E − ½z + z⁴ (iv)
P(z) = z^{-1} − E + 0z² + z⁴ (ii)
P(z) = z^{-1} − E − ½z + ½z² + z⁴ (v)
P(z) = z^{-1} − E − ½z² + z⁴ (iii)
P(z) = −z^{-4} − ½z^{-3} − ½z^{-2} − ½z^{-1} − E + z (vi)

Figure 1: **(a)** Exotic geometries of spectral graphs, whose complexity quickly becomes difficult to classify by human inspection. **(b)** A spectral graph ($\mathcal{G}$) emerges from the eigenspectrum of a non-Hermitian Hamiltonian matrix. It becomes rigorously defined only in the large-size limit ($L \to \infty$). This limiting spectrum can be derived from the characteristic polynomial $P(z, E)$ of the Hamiltonian. The polynomials $P(z, E)$ are grouped into classes ($\mathcal{C}_P$) based on their algebraic form. `Poly2Graph` automates the transformation $P(z, E) \to \mathcal{G}$, and `GnLTransformer` solves the inverse classification $\mathcal{G} \to \mathcal{C}_P$. **(c)** [i-iii] The graph topology is sensitive to polynomial coefficients; a single $\mathcal{C}_P$ may encompass multiple distinct topologies. Conversely, [i,iv-vi] the same graph topology may arise from different $\mathcal{C}_P$; nuanced variations in node and edge locations are necessary to uniquely identify the spectral graph with its parent $\mathcal{C}_P$.

fied, determining which *specific patterns* (e.g., loops, branching structures) encode the decisive information about the parent polynomial remains an open question.

We address these challenges with a two-pronged approach. First, we develop `Poly2Graph`, a fast and memory-efficient Python package that automatically constructs spectral graphs from arbitrary characteristic polynomials. This tool enables the systematic exploration of non-Hermitian spectral topologies at scale. By resolving critical computational bottlenecks, `Poly2Graph` achieves a $\mathbf{10^5} \times$ **speedup** over previous methods. As a byproduct, we introduce a large-scale **multi**graph[2] dataset for graph-level classification, along with its generator. This methodology can generate an infinite variety of graph topologies, providing a novel benchmark resource for multigraph algorithms.

Second, we propose a novel explainable graph transformer, `GnLTransformer` (**G**raph a**n**d **L**ine Graph **Transformer**). This architecture introduces dual channels that leverage **line graphs** to explicitly capture higher-order topology beyond standard node-node interactions. Trained on the generated dataset, `GnLTransformer` achieves **99%+ accuracy** in predicting the polynomial class from a given spectral graph, surpassing the best baseline by 32%. This architecture holds potential for representation learning in complex systems exhibiting higher-order phenomena, as well as molecular embedding and drug discovery.

Beyond performance, a primary focus of this work is explainability. `GnLTransformer` transcends conventional graph transformers by being inherently interpretable not only at the node level but also for **higher-order topological structures** such as edges, triplets, and, more generally, *simplices*.

By visualizing attention patterns and similarity matrices across these higher-order components, the model recovers critical features identified in prior domain literature, and

[2] In a multigraph, a pair of nodes can be connected by multiple edges. Conventionally multiedges are aggregated into a vectorized attribute of a simple-edge; however, in spectral graph the multiedges are embedded in a metric space ($\mathbb{C}$-plane), appearing as distinct continuous paths that can not be straightforwardly aggregated.

moreover *uncovers new insights* into the critical spectral features that link a spectral graph to its parent polynomial—*insights elusive to traditional analytical methods*.

Together, these contributions bridge non-Hermitian topology, algebraic geometry, morphological computer vision, and graph representation learning, underscoring the potential of explainable artificial intelligence (XAI) as a powerful tool for scientific discovery.

2 Preliminaries & Setup

In physical sciences, it is customary to represent and study a system through its Hamiltonian. The energy spectrum, comprising the eigenvalues of the Hamiltonian, reveals the energy band structure—a central focus in condensed matter physics. Consider a generic 1D tight-binding Hamiltonian:

$$\boldsymbol{H} = \sum_{x,j} t_j \hat{c}_x^\dagger \hat{c}_{x+j} \tag{1}$$

where x indexes unit cells, j denotes the hopping displacement, and $\hat{c}_x$ ($\hat{c}_x^\dagger$) is the annihilation (creation) operator at the x-th unit cell. The term t_j represents the transition *amplitude* for a particle hopping from site $x+j$ to site x, and the transition *probability* is given by the squared magnitude, $|t_j|^2$.

The matrix representation in *real* space, $\boldsymbol{H}_{\text{real}}$, is a Toeplitz matrix (where each descending diagonal from left to right is constant):

$$\boldsymbol{H}_{\text{real}} = \begin{pmatrix} t_0 & t_1 & t_2 & \cdots & & 0 \\ t_{-1} & t_0 & t_1 & & & \\ t_{-2} & t_{-1} & t_0 & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & & t_2 \\ & & & & t_0 & t_1 \\ 0 & & \cdots & t_{-2} & t_{-1} & t_0 \end{pmatrix} \tag{2}$$

For a system with L sites, $\boldsymbol{H}_{\text{real}} \in \mathbb{C}^{L\times L}$. The hopping terms t_j can generally be complex or matrix-valued (for multi-band systems). In general $t_j \neq t_{-j}$, which *breaks the Hermiticity* of the Hamiltonian $\boldsymbol{H}^\dagger := (\boldsymbol{H}^*)^T \neq \boldsymbol{H}$ and the eigenvalues assume *complex* values.

Diagonalizing $\boldsymbol{H}_{\text{real}}$ yields the eigenvalues. Even in finite-size systems, the skeleton of the **spectral graph** can be discerned; e.g., the finite energy spectrum in figure 1b resembles a 3-Cayley tree. In the thermodynamic limit ($L \to \infty$), the spectral graph becomes rigorously defined (Tai and Lee 2023; Xiong and Hu 2023).

These spectral graphs exhibit a kaleidoscope of pretty geometries, including arcs, loops, and more exotic shapes resembling stars, kites, braids, and even rockets (Figure 1a). These structures represents an uncharted band topology which embeds hidden symmetries and graph topological transitions that lie beyond standard homotopy-based frameworks. In effect, *a new class of topological invariants* appears—those tied to the global geometry of the eigenvalue loci.

However, accurately diagonalizing large non-Hermitian matrices is notoriously challenging, and the infinite-size case is intractable via direct diagonalization (Yang et al. 2020). The standard approach, guided by *non-Bloch band theory*, is to first Fourier transform the real-space Hamiltonian $\boldsymbol{H}_{\text{real}}$ to momentum-space:

$$\boldsymbol{H}(z) = \sum_{j=-p}^{q} t_j\, z^j, \quad z := e^{ik} \tag{3}$$

where $k \in [-\pi, \pi)$ is the quasimomentum. Subsequently, the celebrated *generalized Brillouin zone* (GBZ) theory is applied. We defer mathematical details to appendix A.

The key takeaway is that the spectral graph in the thermodynamic limit can be derived entirely from the *roots* of the **characteristic polynomial**. This polynomial is the determinant of the momentum-space Hamiltonian minus the energy E (i.e., the energy-momentum dispersion):

$$\boxed{P(z,E) := \det\big[\boldsymbol{H}(z) - E\,\boldsymbol{I}\big] = \sum_{n=-p}^{q} a_n(E)\, z^n.} \tag{4}$$

The algebraic form of $P(z,E)$—which monomials z^n are present—plays a crucial role in classifying spectral graph.

Note that the spectral graph is invariant under parity transformation—i.e., spatial inversion about the origin ($x \to -x$), or put differently, flipping the 1D lattice from left to right. In terms of $\boldsymbol{H}_{\text{real}}$, it is equivalent to transpose the matrix ($t_j \to t_{-j}$) which does not change eigenvalues. To account for this invariance, we define the **characteristic polynomial class** $\mathcal{C}_P$. We first define a **binary coefficient vector** $\boldsymbol{b}$ indicating the presence of monomials:

$$\boldsymbol{b} = (b_{-p}, ..., b_q)$$
$$\text{where } b_n := 1 \text{ if } a_n(E) \neq 0; b_n := 0 \text{ otherwise.} \tag{5}$$

The class $\mathcal{C}_P$ is then defined as the set containing $\boldsymbol{b}$ and its reverse $\boldsymbol{b}' = (b_q, ..., b_{-p})$:

$$\mathcal{C}_P = \{\boldsymbol{b}, \boldsymbol{b}'\}. \tag{6}$$

If $\boldsymbol{b}$ is palindromic ($\boldsymbol{b} = \boldsymbol{b}'$), the class contains a single element $\mathcal{C}_P = \{\boldsymbol{b}\}$. We find $\mathcal{C}_P$ is the key criterion that classifies spectral graphs, and thus the target of our inverse classification task.

Example. In figure 1b, the 1D one-band example stems from the Bloch Hamiltonian $H(z) = -z^{-2} - z + z^4$, which is a scalar. Its characteristic polynomial is therefore $P(z,E) = -z^{-2} - E - z + z^4$, belonging to the class $\mathcal{C}_P = \boldsymbol{b} \cup \boldsymbol{b}' = (1,0,1,1,0,0,1) \cup (1,0,0,1,1,0,1)$.

3 Methods

3.1 Poly2Graph

`Poly2Graph` is an optimized, end-to-end computational pipeline designed to automate the extraction of spectral graphs from one-dimensional non-Hermitian Hamiltonians. It operationalizes the theoretical constructions described in appendix A by integrating non-Bloch band theory, algebraic geometry, and morphological image processing. Full algorithmic details are deferred to appendix B.

The realization of this pipeline required overcoming significant computational challenges inherent in calculating the spectrum in the thermodynamic limit. `Poly2Graph` achieves a $\mathbf{10^5}\times$ speedup over prior implementations (benchmarked in appendix B.5), enabling the systematic, large-scale exploration of these topologies for the first time.

The core of the procedure involves calculating the Density of States (DOS), $\rho(E)$, across the complex energy plane $\Omega \subset \mathbb{C}$. The spectral graph is defined by the regions where $\rho(E) > 0$. The DOS is derived from the Laplacian of the spectral potential, $\Phi(E)$:

$$\rho(E) = -\frac{1}{2\pi}\nabla^2\Phi(E), \tag{7}$$

where $\nabla^2 = \partial^2_{\mathrm{Re}E} + \partial^2_{\mathrm{Im}E}$. The spectral potential (also known as the Ronkin function (Wang, Song, and Wang 2024)) itself depends on the roots $\{z_i(E)\}$ of the characteristic polynomial $P(z, E) = 0$, sorted by magnitude $|z_1(E)| \leq \cdots \leq |z_{p+q}(E)|$:

$$\Phi(E) = -\log|a_q(E)| - \sum_{i=p+1}^{p+q} \log\left|z_i(E)\right|, \tag{8}$$

where $a_q(E)$ is the leading coefficient (Tai and Lee 2023; Xiong and Hu 2023; Wang, Song, and Wang 2024).

High-Throughput Root Finding. The primary bottleneck in this process is the need to calculate the roots of $P(z, E) = 0$ across a dense grid Ω (often requiring $> 10^6$ points for graph extraction quality). Prior methods are prohibitively slow at this stage. `Poly2Graph` addresses this by employing a strategy based on Frobenius companion matrices, transforming the root-finding problem into a parallelizable eigenvalue problem. Utilizing optimized eigen-solvers with automatic GPU acceleration, this approach reduces the computation time from hours to milliseconds.

Adaptive Resolution Enhancement. A second challenge involves balancing computational cost with the resolution required to capture fine topological features (small loops, adjacent nodes, etc). Since the spectral graph typically occupies only a small fraction of the domain Ω, uniform high-resolution sampling is inefficient—evident from figure 1b, the graph skeleton only occupies a measure-zero area in the thermodynamic limit (white region), and the majority of the domain is non-informative (black region). `Poly2Graph` hence introduces a two-stage adaptive refinement strategy.

1. **Region Identification:** The DOS is initially computed on a coarse grid (e.g., 256×256). Morphological operations (binarization and dilation) are applied to this coarse image to generate a mask that conservatively covers the spectral graph while excluding 95-99% of the empty domain.
2. **Targeted Refinement:** The spectral potential and DOS are recalculated at a significantly higher resolution (e.g., $4\times$ enhancement) exclusively within the masked region.

This approach achieves high effective resolution (e.g., 1024×1024) with minimal overhead. Efficiency is further improved by exploiting inherent symmetries in the characteristic polynomial (e.g., real coefficients imply symmetry about the real axis), which can reduce computation by up to 50%.

Morphological Graph Extraction. Finally, the high-resolution DOS image is converted into a graph structure. We apply iterative morphological thinning (skeletonization) algorithms (Lee, Kashyap, and Chu 1994) to reduce the binarized DOS to a one-pixel-wide skeleton while preserving topology. This skeleton is then processed to identify nodes (junctions and endpoints) and edges, outputting the result as a `NetworkX MultiGraph` object.

3.2 GnLTransformer

The inverse classification from spectral graph to characteristic polynomial class is analytically intractable, due to the irreversible and nonlinear operations in the forward map from characteristic polynomial to spectral graph. There has been no attempt to solve this problem.

To address this open problem, we propose `GnLTransformer`, a novel graph neural network architecture. GnLTransformer leverages both the original graph structure and its higher-order relationships (captured by line graphs), and the multi-head attention of transformer architecture; hence the name GnLTransformer, as in "**G**raph a**n**d **L**ine graph **Transformer**".

Learning *independent* representations for each higher-order topological component, combined with the powerful attentive convolution layers and self-attention pooling, GnLTransformer effectively captures complicated geometry inherent in spectral graphs. The forward pass is summarized in algorithm 2.

Notably, transcending traditional graph transformers, GnLTransformer is by design explainable not only on nodes but also on higher-order topological components, such as edges, triplets, and beyond. Visualizing the attention patterns across higher-order topology reveals new insights on the essential spectral graph features that identify the spectral graph to its parent polynomial, which are cryptic to traditional methodologies.

Line Graphs and Higher-Order Topology. The line graph of $\mathcal{G}$, denoted as $\mathcal{L} = L(\mathcal{G})$, is constructed such that each node in $\mathcal{L}$ represents an edge in $\mathcal{G}$. Two nodes in $\mathcal{L}$ are connected if and only if their corresponding edges in $\mathcal{G}$ share a common endpoint (illustrated in figure 2c).

The line graph operator $L(\cdot)$ can be nested to capture higher-order topology—e.g., $L(\mathcal{G})$ describes how edges in $\mathcal{G}$ are adjacent, and its edge represents a *triplet*; $L^2(\mathcal{G})$ captures the adjacency of $\mathcal{G}$'s triplets and its edge represents a *3-claw*.

In general[4], $L^i(\mathcal{G})$ captures the interactions within i-simplices in $\mathcal{G}$ (edge is 1-simplex, triangle is 2-simplex, etc.). The augmentation of higher-order relationship overcomes the intrinsic limitations of pairwise interactions; e.g.,

[3]In our experiments, we use only two channels, $\{\mathcal{G}, L(\mathcal{G})\}$, to demonstrate the architecture's effectiveness. The translucent $L^2(\mathcal{G})$ channel and ellipses are meant to illustrate that the channels can be extended indefinitely. Moreover, since $\mathcal{G}$ is planar—i.e., it can be laid out on a 2D plane without edge crossings—adding more channels provides no additional advantage.

[4]$L^i(\mathcal{G}) = L(L^{i-1}(\mathcal{G}))$, $L^0(\mathcal{G}) = \mathcal{G}$

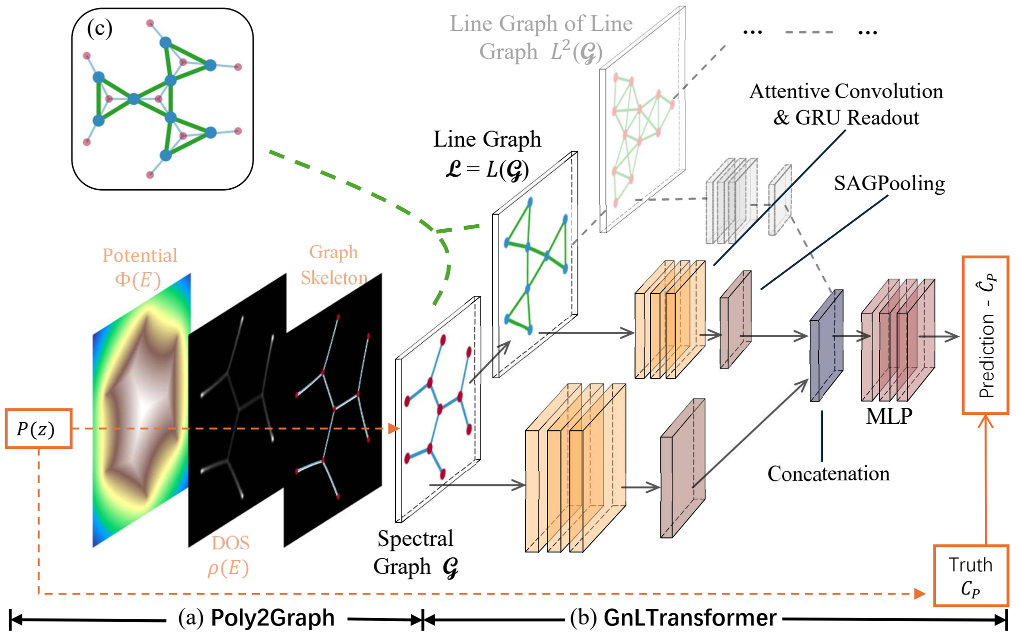

Figure 2: **(a)** `Poly2Graph` transforms input characteristic polynomials to `NetworkX` MultiGraphs via a series of algebro-geometric operations and morphological image processing. **(b)** `GnLTransformer` takes spectral graphs $\{\mathcal{G}\}$ as input, creates line graphs $\{L^i(\mathcal{G}) : i = 1, ..., \ell\}$, and processes them through parallel channels[3]. Each channel comprises attentive convolution blocks and self-attention graph pooling. Their outputs are concatenated to form graph-level embeddings fed into an MLP classifier. **(c)** The line graph (blue vertices, green edges) of a spectral graph (red vertices, blue edges). Each vertex of $L(\mathcal{G})$ represents an edge of $\mathcal{G}$, and two vertices of $L(\mathcal{G})$ are adjacent if the corresponding edges share a common endpoint in $\mathcal{G}$.

it is shown to be responsible for collective behaviors like explosive transitions (Battiston et al. 2021).

Architecture. GnLTransformer takes spectral graphs $\{\mathcal{G}\}$ as input, create ℓ line graphs $\{L^i(\mathcal{G}) : i = 1, ..., \ell\}$, and forward pass them separately through ℓ parallel channels. In figure 2b we demonstrate three channels $\{\mathcal{G}, L(\mathcal{G}), L^2(\mathcal{G})\}$ to indicate the infinite extensibility of the architecture. But in the case of classifying spectral graphs, we only use two channels $\{G, L(G)\}$ (i.e., set $\ell = 1$)—since spectral graphs are planar[5], $L(G)$ already captures edge-edge attention and triplet feature; higher-level attention (e.g., triplet-triplet) is only meaningful for 3D objects like molecules and thus offers diminishing returns in capturing additional information.

Each channel is composed of the following components:

1. **Attentive Convolution Layers:** We adopt two flavors—TransformerConv (Shi et al. 2021) and GATv2Conv (Veličković et al. 2018)—to propagate and refine node features in local neighborhoods.
2. **Gated Recurrent Units (GRUs):** Interleaved with the convolution layers, GRUs (Cho et al. 2014) help integrate information across layers, mitigating oversmoothing and preserving sequential dependencies.
3. **Jumping Knowledge:** After passing through all (convolution, GRU) blocks, we employ jumping knowledge (Xu et al. 2018) to produce the final node embeddings, in order to capture hierarchical information covering all neighborhood scales.
4. **Self-Attention Graph Pooling:** An attentive pooling mechanism that ranks nodes(/edges) and coarse-grain the graphs by keeping the top-k nodes(/edges) with the highest importance (Lee, Lee, and Kang 2019). The pooled embeddings are then concatenated to form the graph-level embedding (Zhang et al. 2018).

Finally, the graph-level embeddings from all channels are concatenated and passed through an multilayer perceptron (MLP) to produce the final classification logits.

3.3 Related Works

GNNs have demonstrated significant success across diverse scientific and social domains, including crystallography (Xie and Grossman 2018; Park and Wolverton 2020), materials discovery (Merchant et al. 2023; Reiser et al. 2022; Schütt et al. 2018), high-energy physics (DeZoort et al. 2023), molecular modeling and drug discovery (Duvenaud et al. 2015; Xiong et al. 2020; Jiang et al. 2021), and social network analysis (Fan et al. 2019).

Several studies have incorporated line graphs into GNN architectures. CL-GNN (Zhu et al. 2019) uses a coarsened line graph for heterogeneous graphs. LgaCL (Wang et al. 2023) employs the original and line graphs as positive pairs in contrastive learning. The most closely related work is ALIGNN (Choudhary and DeCost 2021), which utilizes alternating message passing on the graph and its line graph for

[5]Planar graphs are graphs that can be laid out on a 2D plane without edge crossings.

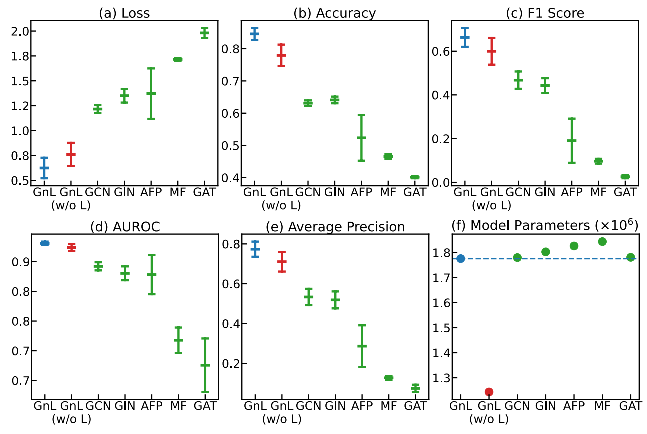

Figure 3: **Ablation study and baseline model comparison results** on a 20% subset of the data. The blue, red, and green markers denote `GnLTransformer`, `GnLTransformer` (without the line graph channel), and the baseline models, respectively. In terms of test accuracy, `GnLTransformer` outperforms the best baseline model, `GIN`, by a relative margin of 32%. Removing the line graph channel decreases the accuracy by 8%. The ablated `GnLTransformer`, despite having fewer parameters, still outperforms the best baseline by 22%. Table 4 lists the exact numerics corresponding to this figure.

molecular property prediction.

`GnLTransformer` differs from ALIGNN in several key aspects: (1) The use of multi-head attention provides inherent explainability; (2) GRU units mitigate oversmoothing; (3) Jumping Knowledge captures hierarchical information; (4) SAGPooling preserves richer graph-level information. Most significantly, our work explores the application of *nested* line graphs and emphasizes the resulting explainability regarding higher-order topology.

4 Experiments

We validate the `GnLTransformer` on the task of inverse classification: predicting the characteristic polynomial class $\mathcal{C}_P$ from a given spectral graph $\mathcal{G}$.

4.1 Dataset Generation

We generated a large dataset, **SpectralGraph-117K**, by systematically sampling the space of characteristic polynomials. We focused on single-band systems with a degree range of 8 ($p = q = 4$):

$$\hat{P}(z, E) = z^{-4} + a_{-3}z^{-3} + a_{-2}z^{-2} + a_{-1}z^{-1} - E + a_1 z + a_2 z^2 + a_3 z^3 + z^4. \quad (9)$$

We sample the 6 variable real coefficients $\{a_{\pm1}, a_{\pm2}, a_{\pm3}\}$ within the 6D hypercube $[-1.2, 1.2]^6$. Each parameter is sampled at 7 equidistant points: $\{-1.2, -0.8, ..., 0.8, 1.2\}$. We apply `Poly2Graph` to each resulting $\hat{P}(z, E)$, generating a dataset $\mathbb{G}$ of $117,658$ pairs of $(\mathcal{G}, \mathcal{C}_P)^6$.

This dataset presents a significant challenge compared to standard graph-level benchmarks (see table 5). It comprises **117K spectral graphs** distributed highly unevenly across **36 $\mathcal{C}_P$ classes**, with class sizes ranging dramatically from 10 to 47K (visualized by the color imbalance in figure 4).

4.2 Input Feature Engineering

The spectral graphs are spatial graphs embedded in the complex energy plane. We design features to capture this geometry explicitly. For the graph $\mathcal{G}$, each node is assigned a $d_v = 2$ feature vector $\boldsymbol{x}^G$ containing its coordinates $(\mathrm{Re}E, \mathrm{Im}E)$. Each edge has a $d_e = 11$ feature vector $\boldsymbol{e}^G$, comprising the edge length and the 2D positions of 5 equidistant points along the edge.

For the line graph $\mathcal{L}$, each node corresponds to an edge in $\mathcal{G}$ and inherits its features: $\boldsymbol{x}^L = \boldsymbol{e}^G$. Each edge in $\mathcal{L}$ represents a triplet (two connected edges) in $\mathcal{G}$. We define a $d_t = 7$ feature vector $\boldsymbol{e}^L$ capturing the triplet's geometry: the 2D position of the triplet center (average of the three nodes' positions) and 5 angles characterizing the branching structure at the joint node. These angles are derived by considering vectors formed by the joint node, the two adjacent nodes, and 2 equidistant points on each connecting edge.

4.3 Performance on Inverse Classification

We trained `GnLTransformer` on the full dataset using a 90-5-5 train-validation-test split, repeating the process over 5 random seeds. Training utilized the AdamW optimizer with a cosine annealing learning rate scheduler. The model state checkpointed at the best validation accuracy was used for final evaluation. Full training details are provided in appendix D, and hyperparameters (derived from the architecture detailed in Appendix C) are listed in table 3.

Metric on $\mathcal{D}_{\text{test}}$	Value
Accuracy	$99.78\% \pm 0.05\%$
F1 (macro)	0.985 ± 0.005
AUROC (macro)	0.99998 ± 0.00001

Table 1: Performance metrics (Accuracy, macro-averaged F1 score, and macro-averaged AUROC) of `GnLTransformer` on the inverse classification task using the full SpectralGraph-117K dataset, averaged over 5 runs.

`GnLTransformer` achieves near-perfect classification results (Table 1). This exceptional performance suggests that while the mapping from $\mathcal{C}_P$ to the graph *topology* (ignoring spatial embedding) is *many-to-many*, the mapping to the specific spectral graph *geometry* (including spatial embedding) is likely *injective*.

The principled sampling in the dataset generation and the deterministic nature of the algebro-geometric transforma-

[6] We slightly augmented the dataset by adding 9 copies of the sole sample belonging to the class $\mathcal{C}_P = \{(1, 0, 0, 0, 1, 0, 0, 0, 1)\}$, which otherwise would have only one instance under this sampling strategy.

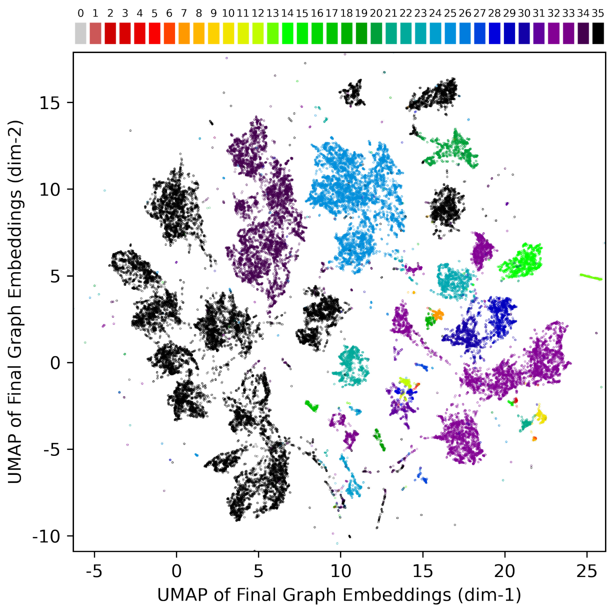

Figure 4: UMAP visualization of the final graph embeddings (logits), colored by the true characteristic polynomial class. Despite severe dataset skewness, `GnLTransformer` clearly separates the spectral graphs into distinct clusters.

tion are likely the key factors enabling the model to effectively distinguish classes based on nuanced geometric features. The UMAP visualization of the final graph embeddings (Figure 4) confirms the model's ability to clearly separate the classes in the latent space, despite the severe class imbalance.

Misclassification Analysis. A closer analysis of the few errors reveals they predominantly arise from *reciprocal* polynomials, where $a_{-n} = a_n$. As detailed in appendix D.1, reciprocity implies that the parent Hamiltonian is Hermitian, ensuring *real* eigenvalues. In the single-band case studied here, all reciprocal polynomials yield an isomorphic graph: two nodes connected by a single edge lying on the real axis, creating degenerate cases that the model finds difficult to distinguish.

Further analysis of the latent space identifies a decision boundary where the difference between the top two prediction probabilities is minimal. All misclassified instances are located near this boundary. Remarkably, *this boundary effectively separates graphs originating from Hermitian Hamiltonians (reciprocal polynomials) from those originating from non-Hermitian ones*, indicating the model has grasped this physical distinction.

Ablation Study & Baseline Comparison. To evaluate the specific contribution of the line graph channel and to compare `GnLTransformer` against established GNN architectures, we conducted an ablation study and baseline comparison. To rigorously test the models' generalization capabilities, these comparisons were performed on a random 20% subset of the data (23.5K graphs) over 10 random splits (details in appendix D).

We compared `GnLTransformer` (GnL) with an ablated version lacking the line graph channel (GnL w/o L), and five popular baselines: GCN (Kipf and Welling 2017), GAT (Veličković et al. 2018), GIN (Xu et al. 2019), Neural Molecular Fingerprints (MF) (Duvenaud et al. 2015), and Attentive Fingerprints (AFP) (Xiong et al. 2020).

The results are summarized in figure 3 (numerical details in table 4). On this challenging subset, `GnLTransformer` achieves a test accuracy of **84.6%**, significantly outperforming the best baseline, GIN (64.1%), by a relative margin of **32%**.

The ablation study demonstrates the critical role of the line graph channel; removing it decreases the accuracy by **8%**. This confirms that explicitly capturing higher-order topology enhances the model's ability to discern complex spectral geometries. Intriguingly, the ablated GnLTransformer (w/o L), despite having fewer trainable parameters than the baselines, still outperforms GIN by **22%**, highlighting the effectiveness of the core architecture.

Given the fact that the baseline models are proven to be powerful in a plethora of real-world applications, GnLTransformer's superior performance on spectral graph suggests that it may also contribute to and push the boundaries in other disciplines, such as bioinformatics and social network analysis, where capturing complicated geometric information and higher-order topological interactions is critical for improving performance.

5 Higher-Order Explainability and Insights

A key advantage of `GnLTransformer` is its inherent explainability, derived from the attentive convolution layers (TransformerConv and GATv2Conv) operating within its parallel-channel architecture. While standard graph transformers provide node-level attention, the inclusion of the line graph channel ($\mathcal{L}$) enables the interpretation of higher-order topological features. For instance, attention weights in the $\mathcal{G}$ channel indicate the significance of neighboring nodes, while attention weights in the $\mathcal{L}$ channel reveal the importance of adjacent edges (representing triplets in $\mathcal{G}$) during message passing.

By visualizing these learned attention weights and analyzing the learned embeddings, we can uncover which specific geometric and topological features the model deems critical for classification—insights often unattainable to traditional analytical methods.

5.1 Attention Visualization: Identifying Critical Features

We visualize the aggregated attention weights (summed across all layers and heads, as detailed in appendix E) for a representative spectral graph in Figure 5b-c. A layer-by-layer breakdown is provided in figure 8. Several key observations emerge across the dataset, revealing which spectral features the model prioritizes:

1. **Focus on Extremal Features.** The model consistently assigns the highest attention weights to the outermost leaf nodes and their connecting edges (Figure 5b-c). This

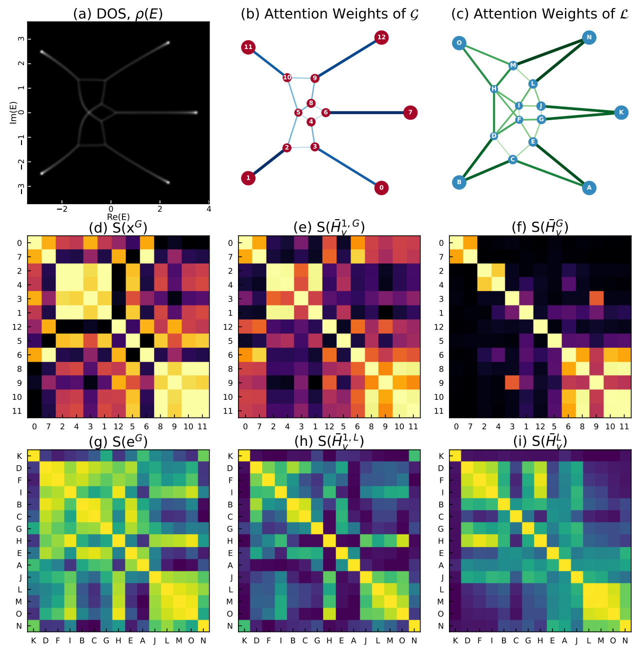

Figure 5: **(a)** Density of states for $\hat{P}(z,E) = z^{-4} + 0.8z^{-3} + 0.4z^{-2} - 1.2z^{-1} - E - 0.4z - 0.8z^2 - 1.2z^3 + z^4$. **(b-c)** Attention visualizations for channels $\mathcal{G}$ and $\mathcal{L}$, where node/edge color, size, and width are scaled by min-max normalized attention weights. The model focuses heavily on leaf nodes and short branches. **(d-i)** Cosine similarity matrices of embeddings at different stages: raw input features (d, g), after the first convolution block (e, h), and final embeddings (f, i). (Upper row: $\mathcal{G}$ nodes; lower row: $\mathcal{L}$ nodes, effectively reflecting edge embeddings in $\mathcal{G}$). The emergence of clear block structures demonstrates the learning of a geometrically aware similarity kernel.

observation aligns remarkably well with theoretical insights from non-Hermitian physics, where leaf nodes are known to encode crucial information about the characteristic polynomial, such as its degree range $p+q$ (Tai and Lee 2023; Lee et al. 2020).

2. **Sensitivity to Topological Transitions.** We observe that short branch edges often receive disproportionately high attention. Short branches typically occur when the system parameters (polynomial coefficients) are near a critical point where the graph topology is about to change (e.g., a loop collapsing or a branch disappearing). The model's focus on these features suggests it learns to recognize these precursors to topological transitions as critical discriminators between classes.

3. **Asymmetric Attention.** Notably, as discussed in appendix E.2, the attention weights are often not distributed symmetrically (e.g., reflection or rotation), even when the graph geometry appears symmetric. This asymmetry indicates that `GnLTransformer` is not merely relying on global topology but is utilizing the precise spatial embedding (geometric information) to distinguish nuanced variations between spectral graphs arising from different polynomial classes.

5.2 Embedding Analysis: Learning Geometric Similarity

To further understand how `GnLTransformer` processes information, we analyze the evolution of the learned representations by examining the cosine similarity matrices of the node and edge embeddings at different stages of the network (Figure 5d-i).

In the initial stage (input features, Figure 5d, g), the similarity reflects only the raw spatial proximity or geometric similarity of the features. However, as the data propagates through the attentive layers, distinct patterns emerge.

In the final embeddings (Figure 5f, i), clear block structures are evident (after appropriate permutation). This demonstrates that `GnLTransformer` effectively clusters nodes (in the $\mathcal{G}$ channel) and edges (in the $\mathcal{L}$ channel) that share similar geometric and topological roles within the spectral graph. This confirms that the architecture successfully learns a robust, geometrically aware similarity kernel during the encoding process, which is essential for the high accuracy. Moreover, it may serve to inspire domain experts to uncover new physical principles governing spectral graph topology.

Conclusion

We have introduced a novel framework for the systematic exploration and classification of non-Hermitian spectral graph topologies derived from characteristic polynomials. This framework addresses significant challenges concerning the complexity of these graphs, the intractability of the inverse mapping, and the identification of essential features. Our contributions include the development of `Poly2Graph`, a highly efficient software package that enables the large-scale generation of spectral graphs, and the release of an extensive spatial multigraph dataset. We proposed `GnLTransformer`, a line graph-enhanced transformer model that achieves over 99% accuracy in identifying the parent polynomial classes. Crucially, the model's inherent explainability regarding higher-order topology provides new insights into the features governing spectral geometry. This work bridges concepts from non-Hermitian physics, algebraic geometry, and graph representation learning, demonstrating the power of explainable AI in deciphering complex physical phenomena and paving the way for future research across diverse scientific domains.

A Mathematical Foundations

This section provides the mathematical underpinning for the concepts introduced in the main text, detailing the emergence of spectral graphs from non-Hermitian Hamiltonians and the theoretical framework used in `Poly2Graph`.

A.1 From Hamiltonian to Characteristic Polynomial

We consider a general 1D tight-binding Hamiltonian describing a system with S internal degrees of freedom (bands) per unit cell:

$$\boldsymbol{H} = \sum_{x,j;\alpha,\beta} (t_j)_{\alpha\beta} \hat{c}^\dagger_{x,\alpha} \hat{c}_{x+j,\beta} \tag{10}$$

where x indexes unit cells, j is the hopping displacement, and $\alpha, \beta \in \{1, \dots, S\}$. The hopping amplitudes t_j are $S \times S$ matrices.

In real space, for a system of size L, the Hamiltonian $\boldsymbol{H}_{\mathrm{real}}$ is an $(LS \times LS)$ block Toeplitz matrix. Non-Hermiticity ($\boldsymbol{H}^\dagger \neq \boldsymbol{H}$) generally leads to complex eigenvalues.

By applying a Fourier transform, we obtain the momentum-space (Bloch) Hamiltonian:

$$\boldsymbol{H}(z) = \sum_j t_j z^j, \quad z := e^{ik} \tag{11}$$

where k is the quasimomentum.

The energy spectrum (dispersion relation) is determined by the roots of the characteristic polynomial:

$$P(z, E) := \det[\boldsymbol{H}(z) - E\,\boldsymbol{I}_S] = \sum_{n=-p}^{q} a_n(E) z^n = 0. \tag{12}$$

Here, we focus on single-band systems ($S = 1$), where p and q correspond directly to the maximum range of backward and forward hopping. The total degree of the polynomial in z (after clearing denominators) is $d_z = p + q$. We sort the roots $\{z_i(E)\}$ of $P(z, E) = 0$ by magnitude: $|z_1| \leq |z_2| \leq \cdots \leq |z_{p+q}|$.

A.2 From PBC to OBC: Spectral Collapse and the GBZ

Under periodic boundary conditions (PBC), the spectrum is obtained by letting k sweep the Brillouin Zone (BZ), i.e., $|z| = 1$. The PBC spectrum typically forms closed loops in the complex energy plane.

However, non-Hermitian systems exhibit extreme sensitivity to boundary conditions. Under Open Boundary Conditions (OBC), eigenstates can become exponentially localized at the boundaries—the non-Hermitian skin effect (NHSE). Consequently, the OBC spectrum differs dramatically from the PBC spectrum; it *collapses* inward from loops to a network of arcs and junctions (figure 6).

The spectral graph $\mathcal{G}$ is defined as the locus of the OBC eigenvalues in the thermodynamic limit (TDL, $L \to \infty$).

To determine this spectrum, non-Bloch band theory is employed. This theory replaces the BZ with the Generalized Brillouin Zone (GBZ). We define the complex momentum $k = k_R + i\kappa$. The imaginary part,

$$\kappa := \mathrm{Im}(k) = -\log|z|, \tag{13}$$

is the inverse localization length (or inverse skin depth) of the eigenstates.

The central result of non-Bloch band theory states that an energy E belongs to the OBC spectrum ($\mathcal{G}$) if and only if there exist two roots (in magnitude) that are equal:

$$|z_p(E)| = |z_{p+1}(E)|, \quad E \in \mathcal{G}. \tag{14}$$

The GBZ is the set of $z(E)$ values satisfying this condition.

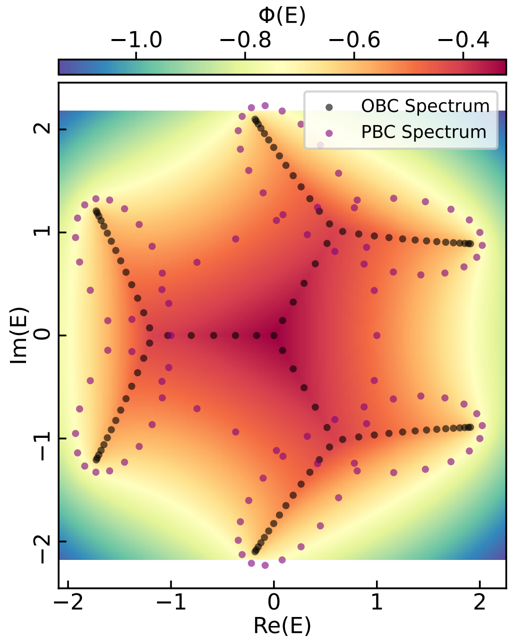

Figure 6: **Spectral Collapse & Spectral Potential.** The PBC spectrum typically appears as loops. Under OBC, the spectrum *collapses* into a graph skeleton. The spectral graph resides on the *ridges* of the spectral potential landscape, $\Phi(E)$.

A.3 The Electrostatic Analogy and Spectral Potential

While equation 14 defines the spectral graph, solving it directly is computationally challenging. A more efficient approach leverages a connection to 2D electrostatics (Yang et al. 2022; Xiong and Hu 2023; Wang, Song, and Wang 2024).

We treat the eigenvalues $\{\epsilon_n\}$ of $\boldsymbol{H}_{\mathrm{real}}$ as localized charges in the complex energy plane. In the TDL, the Density of States (DOS) is $\rho(E)$. We can define a spectral potential (analogous to the Coulomb potential) $\Phi(E)$:

$$\begin{aligned}\Phi(E) &= -\lim_{L\to\infty} \frac{1}{L} \sum_n \log|E - \epsilon_n| \\ &= -\int_{\mathcal{G}} \rho(E') \log|E - E'|\, d^2E'. \end{aligned} \tag{15}$$

The DOS can be recovered from the potential via the Poisson equation:

$$\rho(E) = -\frac{1}{2\pi} \nabla^2 \Phi(E), \tag{16}$$

where $\nabla^2 = \partial^2_{\mathrm{Re}E} + \partial^2_{\mathrm{Im}E}$ is the Laplacian. Geometrically, this implies that the spectral graph $\mathcal{G}$ (where $\rho(E) > 0$) resides on the *ridges* (regions of high curvature) of the potential landscape $\Phi(E)$ (figure 6).

The crucial step is relating $\Phi(E)$ to the characteristic polynomial $P(z, E)$. Note that $\sum \log|E - \epsilon_n| = \log|\det(E\boldsymbol{I} - \boldsymbol{H}_{\mathrm{real}})|$. By applying theorems relating the determinants of large Toeplitz matrices (like $\boldsymbol{H}_{\text{real}}$) to the roots of their symbol (the characteristic polynomial), specifically the Szegő strong limit theorem and its generalizations, $\Phi(E)$ can be expressed directly in terms of the roots $\{z_i(E)\}$ (Xiong and Hu 2023):

$$\Phi(E) = -\log|a_q(E)| - \sum_{i=p+1}^{p+q} \log\big|z_i(E)\big|. \tag{17}$$

This formulation allows the computation of $\Phi(E)$ and subsequently $\rho(E)$ directly from the roots of $P(z,E)$, completely bypassing the need for large-scale matrix diagonalization.

A.4 Characteristic Polynomial Class

The spectral graph $\mathcal{G}$ is determined entirely by the characteristic polynomial $P(z,E)$. To classify these graphs, we define classes based on the algebraic form of $P(z,E)$, characterized by the presence or absence of specific monomials z^n.

Definition 1 (Binary Coefficient Vector)**.** Given $P(z,E) = \sum_{n=-p}^{q} a_n(E)z^n$, the binary coefficient vector $\boldsymbol{b}$ is defined as:

$$\boldsymbol{b} = (b_{-p}, b_{-p+1}, \ldots, b_{q-1}, b_q) \tag{18}$$

$$\text{where } b_n := 1 \text{ if } a_n(E) \neq 0; \quad b_n := 0 \text{ otherwise.} \tag{19}$$

The spectral graph is invariant under spatial inversion (parity transformation) of the lattice. Physically, this corresponds to flipping the 1D lattice left-to-right, which transposes the real-space Hamiltonian $\boldsymbol{H}_{\text{real}}$ ($t_j \to t_{-j}$). Since transposition does not change the eigenvalues, the spectral graph remains the same. In terms of the characteristic polynomial, this corresponds to reversing the order of the coefficients.

Definition 2 (Characteristic Polynomial Class)**.** The characteristic polynomial class $\mathcal{C}_P$ is defined as the set containing the binary coefficient vector $\boldsymbol{b}$ and its reverse $\boldsymbol{b}' = (b_q, \ldots, b_{-p})$:

$$\mathcal{C}_P := \{\boldsymbol{b}, \boldsymbol{b}'\} \tag{20}$$

If $\boldsymbol{b}$ is palindromic ($\boldsymbol{b} = \boldsymbol{b}'$), the class contains a single element $\mathcal{C}_P = \{\boldsymbol{b}\}$.

A.5 The Many-to-Many Mapping

The relationship between the characteristic polynomial class $\mathcal{C}_P$ and the resulting spectral graph topology (i.e., the adjacency structure, ignoring spatial embedding) is complicated and many-to-many.

Many Classes to One Topology. Different polynomial classes can yield isomorphic graph topologies. For example, figure 1c(i,iv-vi) in the main text show instances where distinct $\mathcal{C}_P$ result in the same topological structure.

One Class to Many Topologies. Conversely, the spectral graph topology is sensitive to the specific values of the coefficients $a_n(E)$, not just their presence or absence. Small perturbations in coefficients within the same $\mathcal{C}_P$ can lead to discontinuous topological transitions (e.g., figure 1c(i-iii)).

Because of this many-to-many mapping between $\mathcal{C}_P$ and graph topology, the inverse classification task is challenging. However, when considering the full spectral graph *geometry* (including the spatial embedding of nodes and edges in the complex plane), the mapping from $P(z,E)$ to $\mathcal{G}$ is unique. The nuanced variations in these spatial locations are crucial for distinguishing the parent $\mathcal{C}_P$, yet they elude human inspection, necessitating the use of machine learning approaches like `GnLTransformer`.

B Poly2Graph Algorithm Details

This section adapts material from an anonymized companion work.

Armed with the above theoretical guidance, `Poly2Graph` (Characteristic **Poly**nomial **to** Spectral **Graph**) operationalizes the theoretical framework described in appendix A by integrating numerical methods and morphological image processing techniques (Lee, Kashyap, and Chu 1994; Wang, Yan, and Wei 2018; Nunez-Iglesias et al. 2018). This appendix complements section 3.1 and details the implementation, summarized in algorithm 1.

B.1 Initialization and Input

Poly2Graph accepts diverse input formats for the 1-D tight-binding systems. It can initialize from a Bloch Hamiltonian $\boldsymbol{H}(z)$ or directly from its characteristic polynomial $P(z,E)$. Supported formats for $P(z,E)$ include `sympy.Matrix` (for $\boldsymbol{H}(z)$, $\boldsymbol{H}(k)$), `sympy.Poly`, or a raw string expression of the polynomial. During initialization, Poly2Graph automatically computes a full set of different representations and properties.

The energy domain $\Omega \subset \mathbb{C}$, which must fully contain the spectral graph $\mathcal{G}$, is also required. While users can specify a custom Ω and its discretization, by default Poly2Graph can automatically estimate a suitable region by diagonalizing a small real-space Hamiltonian (typically $L = 40$ unit cells) and applying a small padding.

B.2 Accelerated Root Finding

As detailed in section 3.1, solving for the roots $\{z_i(E)\}$ of $P(z,E) = 0$ for each energy E in the discretized domain Ω is the primary computational bottleneck. To achieve the reported performance gains (five orders of magnitude speedup and higher memory efficiency over previous methods (Tai and Lee 2023) on default settings), we employ a specialized root-finding strategy.

Given the characteristic equation $\sum_{j=-p}^{q} a_j(E)z^j = 0$, we first transform it into a standard polynomial equation by multiplying by z^p: $P'(z,E) = \sum_{k=0}^{d} c_k z^k = 0$, where $d = p+q$ and $c_k = a_{k-p}(E)$. The roots of this polynomial are equivalent to the eigenvalues of its $d \times d$ Frobenius companion matrix $\boldsymbol{F}(E)$:

$$\boldsymbol{F}(E) = \begin{pmatrix} 0 & 0 & \cdots & 0 & -c_0/c_d \\ 1 & 0 & \cdots & 0 & -c_1/c_d \\ 0 & 1 & \cdots & 0 & -c_2/c_d \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -c_{d-1}/c_d \end{pmatrix}. \tag{21}$$

Algorithm 1: `Poly2Graph`: Characteristic **Poly**nomial **to** Spectral **Graph**

```
Input: (1) $\boldsymbol{H}(z)$ or $P(z,E) := \det[\boldsymbol{H}(z) - E\,\boldsymbol{I}]$
 # ↑ Hamiltonian or its characteristic polynomial
Input: (2) Energy Domain: $\Omega \subset \mathbb{C}$ such that $\Omega \supsetneq \mathcal{G}$ (spectral graph)
Output: Spectral Graph: $\mathcal{G} \in$ networkx.MultiGraph

begin
    # Build the characteristic polynomial if only $\boldsymbol{H}(z)$ was given
    if input $\boldsymbol{H}(z)$ then
        $P(z,E) = \det[\boldsymbol{H}(z) - E\,\boldsymbol{I}] = \sum_{n=-p}^{q} a_n(E)\, z^n$

    # Stage 1: Coarse computation over initial energy grid $\Omega$
    (Parallel) for $E \in \Omega$ do
        # Solve roots (via Frobenius companion matrix)
        $\{z_i(E)\} = \mathrm{Sort}[\mathrm{Roots}(P(z,E))]$ s.t. $|z_1| \leq \cdots \leq |z_{p+q}|$
        # Compute spectral potential (Equation 17)
        $\Phi(E) = -\log|a_q(E)| - \sum_{i=p+1}^{p+q} \log\big|z_i(E)\big|$
        # Compute Density of States (DOS) (Equation 16)
        $\rho(E) = -\frac{1}{2\pi}\,\nabla^2\Phi(E)$
    # Identify regions of interest
    coarse_mask $=$ dilate(binarize($\{\rho(E)\}_{E\in\Omega}$))
    # Define refined energy grid
    $\Omega' =$ get_masked_subgrid(coarse_mask)

    # Stage 2: Refined computation within masked regions $\Omega'$
    (Parallel) for $E \in \Omega'$ do
        # Re-solve roots at higher resolution
        $\{z_i(E)\} = \mathrm{Sort}[\mathrm{Roots}(P(z,E))]$
        # Recompute spectral potential
        $\Phi'(E) = -\log|a_q(E)| - \sum_{i=p+1}^{p+q} \log\big|z_i(E)\big|$
        # Recompute DOS
        $\rho'(E) = -\frac{1}{2\pi}\,\nabla^2\Phi'(E)$
    # Combine coarse and refined DOS for full high-resolution image
    $\rho_{\text{final}}(E) =$ combine($\{\rho(E)\}_{E\in\Omega\setminus\Omega'}, \{\rho'(E)\}_{E\in\Omega'}$)
    # Stage 3: Morphological Graph Extraction
    # Binarize high-resolution DOS
    final_binarized_image $=$ binarize($\{\rho_{\text{final}}(E)\}_{E\in\Omega}$)
    # Extract one-pixel-wide skeleton
    graph_skeleton $=$ skeletonize(final_binarized_image)
    # Convert skeleton to graph object
    $\mathcal{G} =$ skeleton2graph(graph_skeleton)
    # Post-processing
    $\mathcal{G} =$ merge_nearby_nodes($\mathcal{G}$, tolerance)
    $\mathcal{G} =$ remove_isolated_nodes($\mathcal{G}$)
```

This formulation holds for complex coefficients $a_j(E) \in \mathbb{C}$.

Poly2Graph constructs a batch of these companion matrices for all $E \in \Omega$. This batch is then processed by a parallelized eigensolver. The implementation leverages `TensorFlow` or `PyTorch` backends for hardware acceleration, including automatic GPU support if available. Calculations are performed using single precision (float32) to optimize memory and computation, which is found sufficient for high-fidelity spectral graph extraction.

B.3 Adaptive Resolution and Image Processing

The adaptive resolution strategy is crucial for computational tractability.

1. **Coarse Identification**: The spectral potential $\Phi(E)$ (equation 17) and DOS $\rho(E)$ (equation 16) are computed on an initial, coarse grid (e.g., 256×256). The DOS image is binarized and morphologically dilated to create a conservative mask covering the potential graph regions.
2. **Refined Calculation**: Within this masked region, the grid is refined (e.g., by a factor of 4 in each dimension), and $\Phi(E)$ and $\rho(E)$ are recomputed at this higher resolution.

This two-stage process achieves high effective resolution (e.g., 1024×1024) while minimizing redundant calculations in empty regions of the complex energy plane.

The resulting high-resolution DOS image is again binarized. We currently employ a *global mean threshold* ($\rho(E) > \langle \rho(E') \rangle_{E' \in \Omega}$) for binarization, as it has empirically outperformed a pool of other common global and adaptive thresholding heuristics, including Otsu, Li, Yen, Triangle, Isodata, local adaptive, and hysteresis variants for our datasets. Subsequently, iterative morphological thinning operations (Lee, Kashyap, and Chu 1994) are applied to reduce the binarized features to a one-pixel-wide skeleton, revealing the graph topology.

B.4 Graph Extraction and Post-processing

The `skeleton2graph` submodule converts the binary skeleton into a graph representation. It identifies pixels as junction nodes (three or more neighbors), leaf nodes (one neighbor), or edge points (two neighbors). The output is a `NetworkX MultiGraph` object, where each edge in particular stores its geometric path as an ordered sequence of $(\mathrm{Re}(E), \mathrm{Im}(E))$ coordinates.

To handle numerical artifacts inherent in the discretization and skeletonization process, two post-processing steps are applied:

1. `merge_nearby_nodes`: Nodes within a predefined Euclidean distance tolerance are merged. This helps consolidate fragmented junctions.
2. `remove_isolated_nodes`: Nodes with no connecting edges after the merging step are removed.

B.5 Benchmark: Poly2Graph Performance

The primary innovation is the *end-to-end automation* of the spectral graph extraction process, which was previously reliant on manual inspection. This automation makes systematic research on spectral graphs feasible for the first time.

Since Poly2Graph is the first fully automated tool for this task, a direct end-to-end comparison with prior work is not possible. We benchmark the core computational bottleneck—the calculation of the spectral potential—against the best available implementation from Ref. (Tai and Lee 2023), which does not include automated graph extraction.

The results, summarized in table 2, demonstrate a speedup exceeding 10^5 across various polynomial degree ranges (covering all realistic scenarios where $p + q \leq 5$). Poly2Graph's time complexity scales polynomially with the degree range ($d_z = p + q$).

Table 2: Performance comparison between Poly2Graph and the implementation from Ref. (Tai and Lee 2023) for the spectral potential calculation bottleneck. We report the time per sample (input $P(z, E)$) and the resulting speedup. Poly2Graph achieves a $> 10^5\times$ speedup.

Degree Range	**Poly2Graph**	**Ref.** (2023)	**Speed-up**
p+q=2	13.1 ± 0.3 ms	3025 s	2.3e5 ×
p+q=3	20.8 ± 0.1 ms	3423 s	1.6e5 ×
p+q=4	28.6 ± 0.3 ms	3921 s	1.4e5 ×
p+q=5	38.8 ± 0.2 ms	5177 s	1.3e5 ×
p+q=6	50.8 ± 0.3 ms	6199 s	1.2e5 ×

C Details of GnLTransformer Architecture

This section details the components of the `GnLTransformer` architecture. The overall forward pass is summarized in algorithm 2.

We denote a spectral graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. Each node $v \in \mathcal{V}$ has a feature vector $\boldsymbol{X}_v \in \mathbb{R}^{d_v}$, forming the node feature matrix $\boldsymbol{X}^G \in \mathbb{R}^{|\mathcal{V}| \times d_v}$. Each edge $(u, v) \in \mathcal{E}$ has a feature vector $\boldsymbol{e}_{uv} \in \mathbb{R}^{d_e}$, forming the edge feature matrix $\boldsymbol{K}^G \in \mathbb{R}^{|\mathcal{E}| \times d_e}$.

C.1 Attentive Convolution Layers

We employ two types of attentive message-passing layers: `TransformerConv` (Shi et al. 2021) and `GATv2Conv` (Brody, Alon, and Yahav 2022). We use a multi-head `TransformerConv` as the first layer due to its high expressivity, followed by several single-head `GATv2Conv` layers.

TransformerConv. For each attention head, the forward pass is defined as:

$$\alpha_{ij} = \mathrm{softmax}_j \left(\frac{(\boldsymbol{W}_q \boldsymbol{X}_i)^\top (\boldsymbol{W}_k \boldsymbol{X}_j + \boldsymbol{W}_e \boldsymbol{e}_{ij})}{\sqrt{d_c}} \right) \quad (22)$$

$$\boldsymbol{H}'_i = \boldsymbol{W}_s \boldsymbol{X}_i + \sum_{j \in \mathcal{N}(i)} \alpha_{ij} \left(\boldsymbol{W}_v \boldsymbol{X}_j + \boldsymbol{W}_e \boldsymbol{e}_{ij} \right) \quad (23)$$

where $\boldsymbol{X}_i$ is the input feature vector of node i, $\mathcal{N}(i)$ denotes the neighbors of node i, $\boldsymbol{W}_{(\cdot)}$ are learnable weight matrices, and d_c is the hidden dimension per head. The term $\boldsymbol{\alpha_{ij}}$ represents the attention weight from node i to node j.

Algorithm 2: Forward Pass of `GnLTransformer`

```
Input: Spectral Graph $\mathcal{G} = \{\boldsymbol{A}^G, \boldsymbol{X}^G, \boldsymbol{K}^G\}$
Output: Logits $\hat{\boldsymbol{H}}_g \in \mathbb{R}^{d_{\text{out}}}$
$\mathcal{L} \leftarrow L(\mathcal{G}) = \{\boldsymbol{A}^L, \boldsymbol{X}^L = \boldsymbol{K}^G, \boldsymbol{K}^L\}$    # Construct line graph
if $\mathcal{E}^L = \emptyset$ then $\tilde{\mathcal{G}} \leftarrow$ AddSelfLoops($\mathcal{G}$), $\mathcal{L} \leftarrow L(\tilde{\mathcal{G}})$
Forward $\mathcal{X} \in \{\mathcal{G}, \mathcal{L}\}$ through parallel channels
    # $\sigma$ denotes an activation function (e.g., ELU, LeakyReLU, ...)
    $\boldsymbol{H}_v^1 \leftarrow \sigma(\text{TransformerConv}(\boldsymbol{X}^{\mathcal{X}}, \boldsymbol{A}^{\mathcal{X}}, \boldsymbol{K}^{\mathcal{X}}))$    # Multi-head Attention
    $\boldsymbol{H}_v^1 \leftarrow \sigma(\text{Linear}(\boldsymbol{H}_v^1))$    # Down projection
    $\bar{\boldsymbol{H}}_v^1 \leftarrow \sigma(\text{GRU}(\boldsymbol{H}_v^1, \boldsymbol{X}^{\mathcal{X}}))$    # GRU update (using initial features as hidden state)
    $\bar{\boldsymbol{H}}_v^1 \leftarrow \sigma(\text{Linear}(\bar{\boldsymbol{H}}_v^1))$
    for $t = 2$ to $n_{conv}$ do
        $\boldsymbol{H}_v^t \leftarrow \sigma(\text{GATv2Conv}(\bar{\boldsymbol{H}}_v^{t-1}, \boldsymbol{A}^{\mathcal{X}}, \boldsymbol{K}^{\mathcal{X}}))$
        $\bar{\boldsymbol{H}}_v^t \leftarrow \sigma(\text{GRU}(\boldsymbol{H}_v^t, \bar{\boldsymbol{H}}_v^{t-1}))$
    $\boldsymbol{H}_v^{\mathcal{X}} \leftarrow \sum_{t=1}^{n_{conv}} \bar{\boldsymbol{H}}_v^t$    # Final Node Embedding (Jumping Knowledge)
    $\boldsymbol{H}_g^{\mathcal{X}} \leftarrow \text{SAGPooling}(\boldsymbol{H}_v^{\mathcal{X}}, \boldsymbol{A}^{\mathcal{X}}, \boldsymbol{K}^{\mathcal{X}})$    # Graph Pooling
$\boldsymbol{H}_g \leftarrow [\, \boldsymbol{H}_g^{\mathcal{G}} \parallel \boldsymbol{H}_g^{\mathcal{L}} \,]$    # Graph-level Embedding
$\hat{\boldsymbol{H}}_g \leftarrow \text{MLP}(\boldsymbol{H}_g)$    # Graph-level Classification
```

The outputs from C parallel attention heads are concatenated and passed through a linear layer with an Exponential Linear Unit (ELU) activation for down-projection:

$$\boldsymbol{H}_i^1 = \left[\, \boldsymbol{H}_i^{\prime(1)} \parallel \boldsymbol{H}_i^{\prime(2)} \parallel \cdots \parallel \boldsymbol{H}_i^{\prime(C)} \,\right] \tag{24}$$

$$\boldsymbol{H}_i^1 = \text{Linear}(\text{ELU}(\boldsymbol{H}_i^1)) \tag{25}$$

GATv2Conv. GATv2 enhances the original graph attention mechanism (Veličković et al. 2018) by allowing dynamic computation of attention coefficients. We disable the node-level self-attention (self-loops) in GATv2Conv to avoid obfuscating spectral graphs that inherently possess self-loops.

The GATv2Conv operation for a node i is defined as:

$$\hat{\alpha}_{ij} = \boldsymbol{A}^\top \text{LeakyReLU}(\boldsymbol{W}_s \boldsymbol{H}_i + \boldsymbol{W}_t \boldsymbol{H}_j + \boldsymbol{W}_e \boldsymbol{e}_{ij}) \tag{26}$$

$$\alpha_{ij} = \frac{\exp(\hat{\alpha}_{ij})}{\sum_{k \in \mathcal{N}(i)} \exp(\hat{\alpha}_{ik})} \tag{27}$$

$$\boldsymbol{H}_i' = \sum_{j \in \mathcal{N}(i)} \alpha_{ij} \boldsymbol{W}_t \boldsymbol{H}_j \tag{28}$$

where $\boldsymbol{H}_{(\cdot)}, \boldsymbol{e}_{(\cdot)}$ are the latent features from the preceding block, $\boldsymbol{A}$ is a learnable attention vector, and $\boldsymbol{W}_{(\cdot)}$ are learnable weight matrices.

C.2 Gated Recurrent Unit (GRU)

A Gated Recurrent Unit (GRU) (Cho et al. 2014) is inserted after each attentive convolution layer to refine node embeddings and integrate information across layers. The t-th GRU takes the embedding $\boldsymbol{H}_i^t$ from the t-th convolution layer and the embedding $\bar{\boldsymbol{H}}_i^{t-1}$ from the $(t-1)$-th GRU cell as input (for $t = 1$, the initial features $\boldsymbol{X}_i$ are used as the initial hidden state), updating the latent embedding as follows:

$$\boldsymbol{r}^t = \text{sigmoid}(\boldsymbol{W}_r \boldsymbol{H}_i^t + \boldsymbol{b}_{w_r} + \boldsymbol{U}_r \bar{\boldsymbol{H}}_i^{t-1} + \boldsymbol{b}_{u_r}) \tag{29}$$

$$\boldsymbol{z}^t = \text{sigmoid}(\boldsymbol{W}_z \boldsymbol{H}_i^t + \boldsymbol{b}_{w_z} + \boldsymbol{U}_z \bar{\boldsymbol{H}}_i^{t-1} + \boldsymbol{b}_{u_z}) \tag{30}$$

$$\boldsymbol{n}^t = \tanh(\boldsymbol{W}_n \boldsymbol{H}_i^t + \boldsymbol{b}_{w_n} + \boldsymbol{U}_n(\boldsymbol{r}^t \odot \bar{\boldsymbol{H}}_i^{t-1} + \boldsymbol{b}_{u_n})) \tag{31}$$

$$\bar{\boldsymbol{H}}^t = (1 - \boldsymbol{z}^t) \odot \boldsymbol{n}^t + \boldsymbol{z}^t \odot \bar{\boldsymbol{H}}_i^{t-1} \tag{32}$$

where $\boldsymbol{z}^t$ and $\boldsymbol{r}^t$ are the update and reset gates, respectively, $\odot$ is the Hadamard product, and $\boldsymbol{W}_{(\cdot)}, \boldsymbol{U}_{(\cdot)}, \boldsymbol{b}_{(\cdot)}$ are learnable parameters. GRUs help capture sequential dependencies across layers and mitigate the oversmoothing effect.

C.3 Jumping Knowledge

We employ Jumping Knowledge (Xu et al. 2018) to produce the final node embeddings by aggregating the outputs from all (convolution, GRU) blocks. This captures hierarchical information across different neighborhood scales. We use summation for aggregation:

$$\boldsymbol{H}_v^{\mathcal{X}} = \sum_{t=1}^{n_{conv}} \bar{\boldsymbol{H}}_v^t \tag{33}$$

Summation is empirically effective and parameter-efficient compared to concatenation (Duvenaud et al. 2015).

C.4 Self-Attention Graph Pooling (SAGPooling)

To generate a graph-level embedding, we use Self-Attention Graph Pooling (SAGPooling) (Lee, Lee, and Kang 2019). SAGPooling uses a trainable graph convolution operator to assign importance scores to nodes, selects the top-k nodes, and uses their embeddings to form a fixed-size graph representation (Zhang et al. 2018).

The procedure for each channel ($\mathcal{X} \in \{\mathcal{G}, \mathcal{L}\}$) is as follows:

1. **Assign attention scores**:
$$\boldsymbol{s}^{\mathcal{X}} = \text{Conv}(\boldsymbol{H}_v^{\mathcal{X}}, \boldsymbol{A}^{\mathcal{X}}, \boldsymbol{K}^{\mathcal{X}}) \tag{34}$$
where $\boldsymbol{H}_v^{\mathcal{X}}$ are the final node embeddings.
2. **Select top-k nodes**:
$$\boldsymbol{i}^{\mathcal{X}} = \text{top}_{(k^{\mathcal{X}})}(\boldsymbol{s}^{\mathcal{X}}) \tag{35}$$
3. **Coarsen graph and obtain pooled embeddings**:
$$\begin{aligned} \tilde{\boldsymbol{H}}_v^{\mathcal{X}} &= \left[\boldsymbol{H}_v^{\mathcal{X}} \odot \tanh(\boldsymbol{s}^{\mathcal{X}})\right]_{(\boldsymbol{i}^{\mathcal{X}})} & (36) \\ &= \left[\boldsymbol{H}_{(1)}^{\mathcal{X}}, \boldsymbol{H}_{(2)}^{\mathcal{X}}, \ldots, \boldsymbol{H}_{(k^{\mathcal{X}})}^{\mathcal{X}}\right]^{\mathrm{T}} \end{aligned}$$
where the subscript $(\cdot)$ denotes the rank.
4. **Form graph-level readout** by concatenating the pooled node embeddings:
$$\boldsymbol{H}_g^{\mathcal{X}} = \left[\, \boldsymbol{H}_{(1)}^{\mathcal{X}} \,\|\, \boldsymbol{H}_{(2)}^{\mathcal{X}} \,\|\, \ldots \,\|\, \boldsymbol{H}_{(k^{\mathcal{X}})}^{\mathcal{X}} \,\right] \tag{37}$$

C.5 Concatenation and Classification

The graph-level embeddings from the $\mathcal{G}$ and $\mathcal{L}$ channels are concatenated and passed through a multi-layer perceptron (MLP) for final classification.

$$\boldsymbol{H}_g = \left[\, \boldsymbol{H}_g^{\mathcal{G}} \,\|\, \boldsymbol{H}_g^{\mathcal{L}} \,\right] \tag{38}$$
$$\hat{\boldsymbol{H}}_g = \text{MLP}(\boldsymbol{H}_g) \tag{39}$$

$\hat{\boldsymbol{H}}_g \in \mathbb{R}^{d_{\text{out}}}$ represents the logits, where d_{out} is the number of target polynomial classes. The model is trained using the cross-entropy loss.

D Training & Evaluation Details

Inverse Classification (Full Dataset). Training was performed on the full SpectralGraph-117K dataset. We used a 90%-5%-5% train-validation-test split. The model state checkpointed at the best validation accuracy was used for evaluation on the test set. This process (random split and training) was repeated 5 times. The model was trained using the AdamW optimizer with a batch size of 512 for 40 epochs. A cosine annealing learning rate scheduler was employed, with no weight decay. Hyperparameters are listed in Table 3. The training dynamics and confusion matrix are shown in Figure 7.

Ablation Study and Baseline Comparison (Subset). For the ablation study and baseline comparison, we used a random 20% subset of the data (23.5K graphs) to challenge the models' generalization capabilities. We generated 10 random splits with a 90%-5%-5% ratio. Training was extended to 150 epochs, with other settings remaining the same as the full dataset run. The results are detailed in Table 4.

Hardware and Experimental Environment. All experiments are run on a single NVIDIA A100 Card (40GB). OS: Red Hat Enterprise Linux 8.4 (Ootpa). GNN models and training code are implemented via PyTorch and PyTorch Geometric.

Hyperparameters		
Node-level	input node features ($\mathcal{G}$)	2
	input edge features ($\mathcal{G}$)	11
	input node features ($\mathcal{L}$)	11
	input edge features ($\mathcal{L}$)	7
	heads of `TransformerConv`	4
	hidden dimension of `TransformerConv`	16
	layers of `GATv2Conv`	3
	hidden dimension of `GATv2Conv`	32
Graph-level	pooling $k^{\mathcal{G}}$	23
	pooling $k^{\mathcal{L}}$	29
	layers of MLP	5
	hidden dimension of MLP	512
	output dimension	36
LR scheduler	$\eta_{\max}$ (Initial LR)	1e-3
	$\eta_{\min}$ (Minimum LR)	5e-5

Table 3: **Hyperparameters of `GnLTransformer`.** The hyperparameters used for the experiments in section 4. Parameters differing between the $\mathcal{G}$ and $\mathcal{L}$ channels are denoted explicitly; otherwise, they are shared.

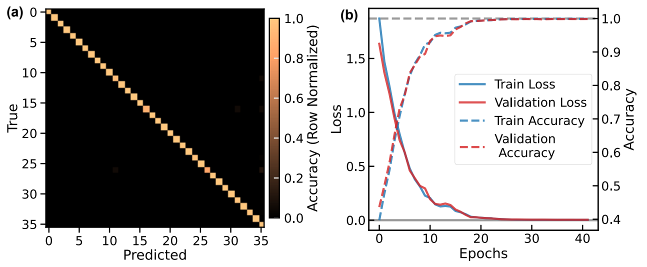

Figure 7: **Training Dynamics and Confusion Matrix. (a)** Confusion matrix of `GnLTransformer` on the test set, normalized by the true class count due to dataset skewness. **(b)** Training and validation loss and accuracy curves. The close tracking of validation and training loss indicates no significant overfitting, likely due to the large size and structured nature of the dataset.

Model	Loss	Accuracy	F1 Score (Macro)	AUROC (Macro)	AP (Macro)	Trainable Parameters
GnL	**0.625 ± 0.103**	**84.59% ± 1.87%**	**0.845 ± 0.019**	**0.980 ± 0.005**	**0.911 ± 0.019**	1.776M
GnL (w/o L)	0.761 ± 0.117	77.94% ± 3.32%	0.778 ± 0.033	0.959 ± 0.012	0.848 ± 0.031	1.243M
GCN	1.217 ± 0.041	63.12% ± 0.84%	0.629 ± 0.008	0.895 ± 0.003	0.683 ± 0.008	1.780M
GAT	1.981 ± 0.051	40.16% ± 0.35%	0.253 ± 0.018	0.629 ± 0.035	0.286 ± 0.021	1.781M
GIN	1.350 ± 0.069	64.10% ± 1.03%	0.637 ± 0.011	0.888 ± 0.006	0.675 ± 0.015	1.803M
MF	1.716 ± 0.013	46.55% ± 0.75%	0.427 ± 0.008	0.783 ± 0.003	0.454 ± 0.006	1.844M
AFP	1.371 ± 0.253	52.34% ± 7.09%	0.486 ± 0.099	0.844 ± 0.058	0.545 ± 0.105	1.827M

Table 4: **Ablation Study and Baseline Comparison Results.** Numerical results corresponding to Figure 3. Experiments were conducted on a 20% subset (23.5K graphs) with 10 random splits (90-5-5 ratio). Metrics are averaged over the 10 test sets. `GnLTransformer` outperforms the best baseline (GIN) by 32% in accuracy. Removing the line graph channel (w/o L) decreases accuracy by 8%.

D.1 Misclassification Analysis

The few misclassified cases predominantly involve reciprocal characteristic polynomials (satisfying $a_{-n} = a_n$). Reciprocity implies that the parent Hamiltonian is Hermitian, resulting in real eigenvalues. In the single-band case studied here, all such polynomials yield an isomorphic spectral graph: two nodes connected by a single edge lying on the real axis. This degeneracy makes these cases difficult for the model to distinguish based purely on the graph structure.

Analysis of the latent space reveals a decision boundary where the difference between the top two prediction probabilities is minimal. All misclassified instances are located near this boundary. Notably, this boundary effectively separates graphs originating from Hermitian Hamiltonians (reciprocal polynomials) from those originating from non-Hermitian ones.

E Explainability: Attention Visualization and Embedding Analysis

The attentive convolution layers (TransformerConv and GATv2Conv) in `GnLTransformer` inherently compute the attention weights ("significance") of neighboring nodes (in the $\mathcal{G}$ channel) and adjacent edges (in the $\mathcal{L}$ channel) during message passing. Visualizing these weights provides insights into the model's focus.

E.1 Attention Visualization Methodology

In Figure 5b-c, we visualize the aggregated attention weights. During message passing, undirected edges are treated as pairs of directed edges. The visualized edge attention weight is the sum of the weights of these two directed edges. A node's attention weight is defined as the sum of the attention weights of all incoming edges. These weights are accumulated across all heads of TransformerConv and all layers of GATv2Conv during the forward pass. The weights are min-max normalized per channel, and visualized such that color intensity, node size, and edge width are proportional to the normalized attention magnitude. Figure 8 shows the detailed breakdown of attention weights per head, layer, and channel.

E.2 Interpretation of Attention Patterns

`GnLTransformer` consistently focuses on the outermost leaf nodes and their adjacent edges. This aligns with theoretical findings that leaf nodes encode crucial information about the characteristic polynomial (Tai and Lee 2023). For example, the number of leaf nodes is closely related to the degree range $p+q$ of the polynomial (Tai and Lee 2023; Lee et al. 2020).

Furthermore, shorter leaf edges tend to receive higher attention. This suggests the model recognizes that short branches are indicators that the graph is near a topological transition point.

The attention weights are generally not distributed symmetrically (e.g., centrosymmetrically or reflection-symmetrically). This asymmetry reflects how `GnLTransformer` balances geometric (location-aware) features and topological (location-ignorant) information during encoding. The ability to integrate both aspects is crucial for distinguishing the nuanced differences between spectral graphs.

E.3 Embedding Similarity Analysis

We analyze the evolution of the learned representations by visualizing the cosine similarity matrices of the node embeddings at different stages (Figure 5d-i and Figure 8). The cosine similarity S_{ij} between embeddings of node i ($\boldsymbol{h}_i$) and node j ($\boldsymbol{h}_j$) is defined as:

$$S_{ij} = \frac{\boldsymbol{h}_i \cdot \boldsymbol{h}_j}{\|\boldsymbol{h}_i\| \|\boldsymbol{h}_j\|} = \cos\theta_{ij} \qquad (40)$$

By appropriate permutation, these matrices reveal clusters of nodes with similar roles. The similarity matrices of the final embeddings show much clearer block structures compared to the input features. This demonstrates that the attentive layers effectively learn a geometrically aware similarity kernel, successfully clustering nodes and edges with similar structural roles within the spectral graph.

Acknowledgments

The computational work for this article was partially performed on resources of the National Supercomputing Cen-

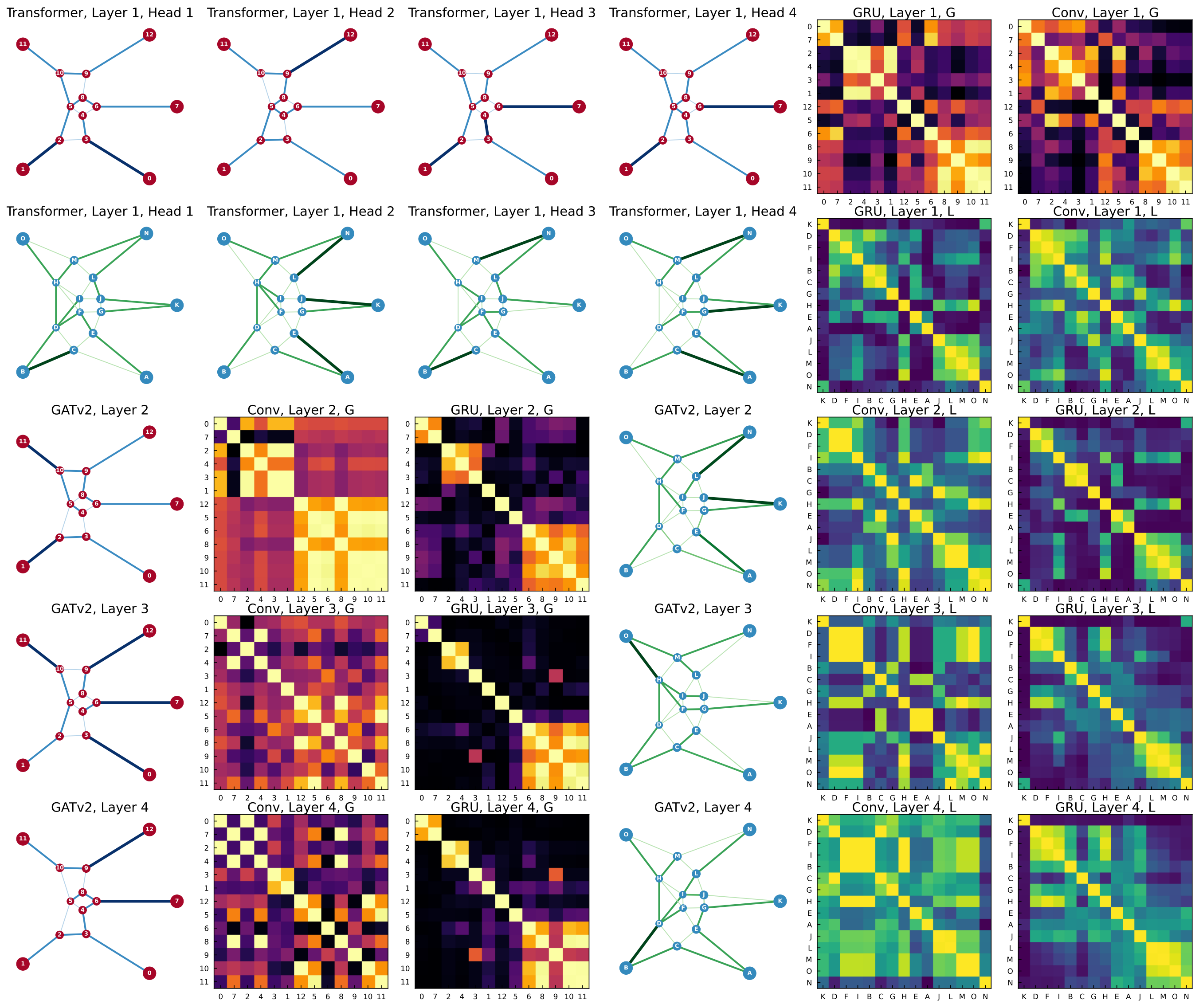

Figure 8: **Detailed Attention Visualization and Node Similarity Matrix per head/layer/channel**. This figure complements Figure 5, showing the full breakdown. Attention weights are visualized for each head of the `TransformerConv` (1st layer) and each `GATv2Conv` layer (layers 2-4) in both $\mathcal{G}$ and $\mathcal{L}$ channels. Cosine similarity matrices are calculated from the output of each attentive convolution layer and the subsequent GRU update.

Dataset	# of Graphs	Target Classes
MUTAG	188	2
ENZYMES	600	6
IMDB-BINARY	1000	2
PROTEINS	1113	2
D&D	1178	2
SIDER	1427	2
ClinTox	1477	2
IMDB-MULTI	1500	3
BACE	1513	2
REDDIT-BINARY	2000	2
BBBP	2039	2
REDDIT-MULTI-5K	4999	5
COLLAB	5000	3
Tox21	7831	2
ToxCast	8576	2
DBLP_v1	19456	2
molhiv	41127	2
REDDIT-MULTI-12K	11929	11
MUV	93087	2
CIFAR10	60000	10
MNIST	70000	10
SpectralGraph-117K	**117658**	**36**
ppa	158100	37
molpcba	437929	2
MALNET	1262024	696

Table 5: Comparison of SpectralGraph-117K with existing graph-level benchmark datasets. At the time of writing, SpectralGraph-117K is among the largest datasets for graph-level tasks and provides a unique benchmark for *spatial multigraphs*. Statistics adapted from Ref. (Yang et al. 2023).

tre, Singapore https://www.nscc.sg. This research is partially supported by the National Research Foundation Singapore under the AI Singapore Programme (AISG Award No: AISG2-TC-2023-010-SGIL) and the Singapore Ministry of Education Academic Research Fund Tier 1 (Award No: T1 251RES2207).

References

Amir, A.; Hatano, N.; and Nelson, D. R. 2016. Non-Hermitian Localization in Biological Networks. *Physical Review E*, 93(4): 042310.

Ashida, Y.; Gong, Z.; and Ueda, M. 2020. Non-Hermitian Physics. *Advances in Physics*, 69(3): 249–435.

Battiston, F.; Amico, E.; Barrat, A.; Bianconi, G.; Ferraz De Arruda, G.; Franceschiello, B.; Iacopini, I.; Kéfi, S.; Latora, V.; Moreno, Y.; Murray, M. M.; Peixoto, T. P.; Vaccarino, F.; and Petri, G. 2021. The Physics of Higher-Order Interactions in Complex Systems. *Nature Physics*, 17(10): 1093–1098.

Brody, S.; Alon, U.; and Yahav, E. 2022. How Attentive Are Graph Attention Networks? arXiv:2105.14491.

Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. arXiv:1406.1078.

Choudhary, K.; and DeCost, B. 2021. Atomistic Line Graph Neural Network for Improved Materials Property Predictions. *npj Computational Materials*, 7(1): 185.

DeZoort, G.; Battaglia, P. W.; Biscarat, C.; and Vlimant, J.-R. 2023. Graph Neural Networks at the Large Hadron Collider. *Nature Reviews Physics*, 5(5): 281–303.

Ding, K.; Fang, C.; and Ma, G. 2022. Non-Hermitian Topology and Exceptional-Point Geometries. *Nature Reviews Physics*, 4(12): 745–760.

Duvenaud, D.; Maclaurin, D.; Aguilera-Iparraguirre, J.; Gómez-Bombarelli, R.; Hirzel, T.; Aspuru-Guzik, A.; and Adams, R. P. 2015. Convolutional Networks on Graphs for Learning Molecular Fingerprints. arXiv:1509.09292.

Fan, W.; Ma, Y.; Li, Q.; He, Y.; Zhao, E.; Tang, J.; and Yin, D. 2019. Graph Neural Networks for Social Recommendation. arXiv:1902.07243.

Ghatak, A.; Brandenbourger, M.; Van Wezel, J.; and Coulais, C. 2020. Observation of Non-Hermitian Topology and Its Bulk–Edge Correspondence in an Active Mechanical Metamaterial. *Proceedings of the National Academy of Sciences*, 117(47): 29561–29568.

Gonzalez-Conde, J.; Rodríguez-Rozas, Á.; Solano, E.; and Sanz, M. 2023. Efficient Hamiltonian Simulation for Solving Option Price Dynamics. *Physical Review Research*, 5(4): 043220.

Jiang, D.; Wu, Z.; Hsieh, C.-Y.; Chen, G.; Liao, B.; Wang, Z.; Shen, C.; Cao, D.; Wu, J.; and Hou, T. 2021. Could Graph Neural Networks Learn Better Molecular Representation for Drug Discovery? A Comparison Study of Descriptor-Based and Graph-Based Models. *Journal of Cheminformatics*, 13(1): 12.

Kawabata, K.; Bessho, T.; and Sato, M. 2019. Classification of Exceptional Points and Non-Hermitian Topological Semimetals. *Physical Review Letters*, 123(6): 066405.

Kawabata, K.; Shiozaki, K.; Ueda, M.; and Sato, M. 2019. Symmetry and Topology in Non-Hermitian Physics. *Physical Review X*, 9(4): 041015.

Kerg, G.; Goyette, K.; Touzel, M. P.; Gidel, G.; Vorontsov, E.; Bengio, Y.; and Lajoie, G. 2019. Non-Normal Recurrent Neural Network (nnRNN): Learning Long Time Dependencies While Improving Expressivity with Transient Dynamics. arXiv:1905.12080.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. arXiv:1609.02907.

Kumar, S.; and Wilmott, C. M. 2024. Simulating the Non-Hermitian Dynamics of Financial Option Pricing with Quantum Computers. arXiv:2407.01147.

Lee, C. H.; Li, L.; Thomale, R.; and Gong, J. 2020. Unraveling Non-Hermitian Pumping: Emergent Spectral Singularities and Anomalous Responses. *Physical Review B*, 102(8): 085151.

Lee, J.; Lee, I.; and Kang, J. 2019. Self-Attention Graph Pooling. arXiv:1904.08082.

Lee, T.; Kashyap, R.; and Chu, C. 1994. Building Skeleton Models via 3-D Medial Surface Axis Thinning Algorithms. *CVGIP: Graphical Models and Image Processing*, 56(6): 462–478.

Lin, R.; Tai, T.; Yang, M.; Li, L.; and Lee, C. H. 2023. Topological Non-Hermitian Skin Effect. *Frontiers of Physics*, 18(5): 53605.

Merchant, A.; Batzner, S.; Schoenholz, S. S.; Aykol, M.; Cheon, G.; and Cubuk, E. D. 2023. Scaling Deep Learning for Materials Discovery. *Nature*, 624(7990): 80–85.

Nelson, D. R.; and Shnerb, N. M. 1998. Non-Hermitian Localization and Population Biology. *Physical Review E*, 58(2): 1383–1403.

Nunez-Iglesias, J.; Blanch, A. J.; Looker, O.; Dixon, M. W.; and Tilley, L. 2018. A New Python Library to Analyse Skeleton Images Confirms Malaria Parasite Remodelling of the Red Blood Cell Membrane Skeleton. *PeerJ*, 6: e4312.

Park, C. W.; and Wolverton, C. 2020. Developing an Improved Crystal Graph Convolutional Neural Network Framework for Accelerated Materials Discovery. *Physical Review Materials*, 4(6): 063801.

Reiser, P.; Neubert, M.; Eberhard, A.; Torresi, L.; Zhou, C.; Shao, C.; Metni, H.; Van Hoesel, C.; Schopmans, H.; Sommer, T.; and Friederich, P. 2022. Graph Neural Networks for Materials Science and Chemistry. *Communications Materials*, 3(1): 93.

Schütt, K. T.; Sauceda, H. E.; Kindermans, P.-J.; Tkatchenko, A.; and Müller, K.-R. 2018. SchNet - a Deep Learning Architecture for Molecules and Materials. *The Journal of Chemical Physics*, 148(24): 241722.

Shi, Y.; Huang, Z.; Feng, S.; Zhong, H.; Wang, W.; and Sun, Y. 2021. Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification. arXiv:2009.03509.

Shimokawa, K.; Ishihara, K.; Grainge, I.; Sherratt, D. J.; and Vazquez, M. 2013. FtsK-dependent XerCD- *Dif* Recombination Unlinks Replication Catenanes in a Stepwise Manner. *Proceedings of the National Academy of Sciences*, 110(52): 20906–20911.

Sone, K.; Yokomizo, K.; Kawaguchi, K.; and Ashida, Y. 2024. Hermitian and Non-Hermitian Topology in Active Matter. arXiv:2407.16143.

Tai, T.; and Lee, C. H. 2023. Zoology of Non-Hermitian Spectra and Their Graph Topology. *Physical Review B*, 107(22): L220301.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. arXiv:1710.10903.

Wang, A.; Meng, Z.; and Chen, C. Q. 2023. Non-Hermitian Topology in Static Mechanical Metamaterials. *Science Advances*, 9(27): eadf7299.

Wang, A.; Yan, X.; and Wei, Z. 2018. ImagePy: an open-source, Python-based and platform-independent software package for bioimage analysis. *Bioinformatics*, 34(18): 3238–3240.

Wang, H.-Y.; Song, F.; and Wang, Z. 2024. Amoeba Formulation of Non-Bloch Band Theory in Arbitrary Dimensions. *Physical Review X*, 14(2): 021011.

Wang, Y.; Wang, H.; Cao, R.; Liu, T.; and Zhang, X. 2023. Graph Contrastive Learning with Line Graph Augmentation.

Xie, T.; and Grossman, J. C. 2018. Crystal Graph Convolutional Neural Networks for an Accurate and Interpretable Prediction of Material Properties. *Physical Review Letters*, 120(14): 145301.

Xiong, Y.; and Hu, H. 2023. Graph Morphology of Non-Hermitian Bands. arXiv:2311.14921.

Xiong, Z.; Wang, D.; Liu, X.; Zhong, F.; Wan, X.; Li, X.; Li, Z.; Luo, X.; Chen, K.; Jiang, H.; and Zheng, M. 2020. Pushing the Boundaries of Molecular Representation for Drug Discovery with the Graph Attention Mechanism. *Journal of Medicinal Chemistry*, 63(16): 8749–8760.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful Are Graph Neural Networks? arXiv:1810.00826.

Xu, K.; Li, C.; Tian, Y.; Sonobe, T.; Kawarabayashi, K.-i.; and Jegelka, S. 2018. Representation Learning on Graphs with Jumping Knowledge Networks. arXiv:1806.03536.

Yang, R.; Tan, J. W.; Tai, T.; Koh, J. M.; Li, L.; Longhi, S.; and Lee, C. H. 2022. Designing Non-Hermitian Real Spectra through Electrostatics. *Science Bulletin*, 67(18): 1865–1873.

Yang, Z.; Zhang, G.; Wu, J.; Yang, J.; Sheng, Q. Z.; Xue, S.; Zhou, C.; Aggarwal, C.; Peng, H.; Hu, W.; Hancock, E.; and Liò, P. 2023. State of the Art and Potentialities of Graph-level Learning. arXiv:2301.05860.

Yang, Z.; Zhang, K.; Fang, C.; and Hu, J. 2020. Non-Hermitian Bulk-Boundary Correspondence and Auxiliary Generalized Brillouin Zone Theory. *Physical Review Letters*, 125(22): 226402.

Zdeborová, L. 2020. Understanding Deep Learning Is Also a Job for Physicists. *Nature Physics*, 16(6): 602–604.

Zhang, M.; Cui, Z.; Neumann, M.; and Chen, Y. 2018. An End-to-End Deep Learning Architecture for Graph Classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Zhu, S.; Zhou, C.; Pan, S.; Zhu, X.; and Wang, B. 2019. Relation Structure-Aware Heterogeneous Graph Neural Network. In *2019 IEEE International Conference on Data Mining (ICDM)*, 1534–1539. Beijing, China: IEEE. ISBN 978-1-7281-4604-1.