Complex-Weighted Convolutional Networks: Provable Expressiveness via Complex Diffusion

Cristina López Amado
IST Austria
cristina.lopezamado@ist.ac.at

Tassilo Schwarz
Mathematical Institute
University of Oxford*
tassilo.schwarz@maths.ox.ac.uk

Yu Tian
Center for Systems Biology Dresden†
yu.tian.research@gmail.com

Renaud Lambiotte
Mathematical Institute
University of Oxford
renaud.lambiotte@maths.ox.ac.uk

Abstract

Graph Neural Networks (GNNs) have achieved remarkable success across diverse applications, yet they remain limited by oversmoothing and poor performance on heterophilic graphs. To address these challenges, we introduce a novel framework that equips graphs with a complex-weighted structure, assigning each edge a complex number to drive a diffusion process that extends random walks into the complex domain. We prove that this diffusion is highly expressive: with appropriately chosen complex weights, any node-classification task can be solved in the steady state of a complex random walk. Building on this insight, we propose the Complex-Weighted Convolutional Network (CWCN), which learns suitable complex-weighted structures directly from data while enriching diffusion with learnable matrices and nonlinear activations. CWCN is simple to implement, requires no additional hyperparameters beyond those of standard GNNs, and achieves competitive performance on benchmark datasets. Our results demonstrate that complex-weighted diffusion provides a principled and general mechanism for enhancing GNN expressiveness, opening new avenues for models that are both theoretically grounded and practically effective.[3]

1 Introduction

Graph Neural Networks (GNNs) [1] have emerged as a powerful class of machine learning models to deal with relational and structured data. They have shown state-of-the-art performance in a wide range of applications, such as recommendation systems [2], molecular property prediction [3], webpage classification [4] or predictions in citation networks [5].

Limitations of GNNs. Most Graph Neural Network (GNN) architectures are built on the message passing paradigm, where each node iteratively aggregates information from its neighbors to update its representation. While this approach has been highly successful, "classical" GNNs usually face two well-documented challenges: (1) poor performance on heterophilic graphs [6] and (2) oversmoothing [7]. Both stem from the inherent difficulty of capturing long-range dependencies. The heterophily problem arises because message passing implicitly assumes homophily [6], i.e. that neighboring nodes tend to share similar features and labels, an assumption that is often not verified in real-world networks

*and Mathematical bioPhysics Group, Max Planck Institute for Multidisciplinary Sciences

†Max Planck Institute of Molecular Cell Biology and Genetics, Max Planck Institute for the Physics of Complex Systems, and Dresden University of Technology

[3]Code: https://github.com/clopezamado/complex-weighted-convolutional-networks.git

Cristina López Amado, Complex-Weighted Convolutional Networks: Provable Expressiveness via Complex Diffusion. *Proceedings of the Fourth Learning on Graphs Conference (LoG 2025)*, PMLR 338, Arizona State University, Phoenix, USA, December 10–12, 2025.

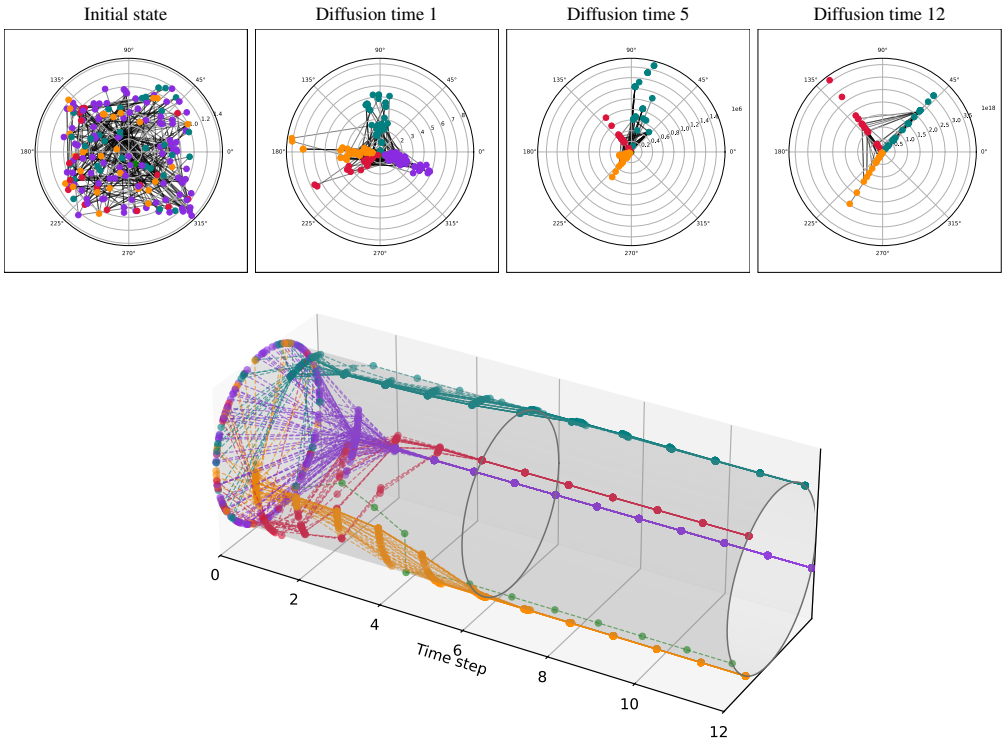

Figure 1: Node features in polar coordinates at selected diffusion times (top) and evolution of node feature phases over time during diffusion (bottom). Complex random walk diffusion progressively separates the classes of a complex-weighted graph: nodes in the same class (same color) converge to complex numbers with the same phase. The figure corresponds to the dataset Texas [15]. From diffusion time 5 onward, the purple class is omitted for better visualization.

[8]. Oversmoothing, on the other hand, refers to the tendency of node representations to become indistinguishable after repeated message passing, eroding their discriminative power. Oversmoothing has been particularly well studied in the context of Graph Convolutional Networks (GCNs). From a diffusion perspective, GCNs can be seen as implementing an augmented heat diffusion process over the graph [9]. In such a process, node features within the same connected component converge to identical values [9], a phenomenon that significantly contributes to oversmoothing in GCNs [10, 11].

Beyond Heat Diffusion. To address these limitations, recent research has focused on developing more expressive diffusion processes. Prominent examples include Sheaf Convolutional Networks (SCNs) [12, 13] and Graph Neural Reaction-Diffusion Networks (GREAD) [14]. Both approaches enrich the underlying graph structure to support more sophisticated forms of diffusion. GREAD has shown strong empirical performance on standard benchmarks, providing a practical and scalable alternative to sheaf-based methods, though it does not come with theoretical guarantees. SCNs, in contrast, offer provable expressiveness, but their guarantees require the sheaf dimension to scale with the number of target classes, a requirement that increases model parameters and, in turn, computational cost.

Contributions. We introduce a new approach that modifies the underlying "geometry" of the graph to define a more expressive diffusion process, forming the foundation for a novel GNN architecture. Concretely, we propose equipping the graph with a complex-weighted structure, where each edge is assigned a complex number. This extension enables a diffusion process that generalizes random walks to the complex domain [16].

Our main contributions are as follows:

- We propose a novel way to enrich graph structure by assigning complex weights to edges, enabling a diffusion process with provable expressiveness guarantees. We show that for any

node-classification task, suitable weights can be chosen such that classification is possible in the steady state of the resulting complex random walk (see Figure 1). To the best of our knowledge, this is the first study establishing a connection between the convergence properties of complex random walks and the expressivity of GNNs for node classification tasks.

- In contrast to sheaf-based diffusion, our theoretical guarantees are independent of the number of target classes, allowing for a simpler model.
- We introduce the *Complex-Weighted Convolutional Network* (CWCN), a GNN that augments complex random-walk diffusion and achieves greater expressiveness than GCNs. CWCN reduces the number of hyperparameters compared to prior methods while maintaining competitive performance on benchmarks.

2 Theoretical Background

Problem Setting. We consider the problem of node classification in undirected graphs. Formally, let $G = (V, E)$ be an undirected graph, with set of nodes $V = \{v_1, \ldots, v_n\}$ and set of edges $E \subset \{\{v_i, v_j\} \mid v_i, v_j \in V\}$. Each node $v_i \in V$ is associated with a feature vector $\mathbf{x}_i \in \mathbb{R}^k$ and belongs to a class $y_i \in \{1, \ldots, C\}$, where C is the number of possible classes. We denote by $\mathbf{A} \in \mathbb{R}^{n \times n}$ the adjacency matrix, $\mathbf{D}$ the degree matrix and $N(v_i)$ the set of neighbours of node v_i. In addition, we group all node feature vectors into a feature matrix $\mathbf{X} \in \mathbb{R}^{n \times k}$.

Given labels for a subset of nodes, the goal is to learn a function $f : V \to \{1, \ldots, C\}$ that predicts the class labels for the remaining nodes in the graph. We focus on the semi-supervised node classification setting, where the complete graph structure – including all node features and edge connections – is available during training, but only a subset of nodes is labelled. In this setting, we denote by $V_{\text{train}}, V_{\text{val}}, V_{\text{test}} \subset V$ the training, validation and test sets, respectively.

2.1 Heat Diffusion and GCNs

The heat diffusion equation in a graph and its unit-timestep Euler discretisation are

$$\dot{\mathbf{X}}(t) = -\triangle_0 \mathbf{X}(t) \quad \rightsquigarrow \quad \mathbf{X}(t+1) = (\mathbf{I} - \triangle_0)\mathbf{X}(t), \tag{1}$$

where $\triangle_0 := \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the normalized graph Laplacian.

Most GNNs follow the message-passing paradigm, in which each node iteratively updates its feature representation by aggregating information from its neighbours. Numerous architectures are built upon this general framework, with Graph Convolutional Networks (GCNs) [17] being one of the most popular examples. GCNs implement message passing at each layer as:

$$\mathbf{H}^{(l)} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l-1)}\mathbf{M}_0^{(l)}) = \sigma((\mathbf{I} - \tilde{\triangle}_0)\mathbf{H}^{(l)}\mathbf{M}_0^{(l)}), \tag{2}$$

where $\mathbf{H}^{(l)} \in \mathbb{R}^{n \times k_l}$ is the matrix of node representations at layer l, $M_0^{(l)} \in \mathbb{R}^{k_{l-1} \times k_l}$ is a learnable weight matrix, and σ denotes a non-linear activation function. $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\tilde{\mathbf{D}} = \mathbf{D} + \mathbf{I}$ are the augmented adjacency and degree matrices, and $\tilde{\triangle}_0 = \mathbf{I} - \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ is the augmented graph Laplacian. Therefore, GCN can be seen as an augmented heat diffusion process with an additional $k_{l-1} \times k_l$ learnable weight matrix $\mathbf{M}_0^{(l)}$ and a non-linearity σ.

Li et al. [9] were the first to highlight the oversmoothing problem, showing that repeatedly applying Equation (1) causes the features of nodes within the same connected component to converge to identical values. Building on this observation, Oono and Suzuki [10] and Cai and Wang [11] demonstrate that, under certain assumptions on the weight matrices $\mathbf{M}_0^{(l)}$, the expressive power of GCNs decays exponentially as the number of layers increases.

2.2 Random Walks on Graphs With Complex Weights

In this section, we present the main concepts regarding complex-weighted graphs, which form the basis for studying the expressive power of the complex-weighted diffusion process, detailed in Section 3.

Definition 1. *A complex-weighted graph is a graph where each edge is assigned a complex number, $G = (V, E, \mathbf{W})$ where $V = \{v_1, \dots, v_n\}$ is the node set, $E \subset \{e_{ij} = (v_i, v_j) : v_i, v_j \in V\}$ is the edge set and $\mathbf{W} = (W_{ij})$ is the complex weight matrix, with $W_{ij} \in \mathbb{C}$ characterizing the edges between nodes.*

Remark 1. *We assume the complex-weighted graphs to be connected and directed*[4]*. In addition, we assume $\mathbf{W}$ to be Hermitian, i.e., $\mathbf{W} = \mathbf{W}^*$.*

Definition 2. *We define the following matrices in a complex-weighted graph:*

1. *The complex degree matrix $\mathbf{Q}$ is the diagonal matrix with elements $q_i = \sum_j |W_{ij}|$.*
2. *The complex transition matrix $\mathbf{P} = \mathbf{Q}^{-1}\mathbf{W}$.*
3. *The complex random walk Laplacian $\mathbf{L}_{rw} = \mathbf{I} - \mathbf{P}$, where $\mathbf{I}$ is the identity matrix.*

We can now use these matrices to define a complex random walk.

Definition 3. *Let $G = (V, E, \mathbf{W})$ be a complex-weighted graph with f feature channels, where node features are represented as a matrix $\mathbf{X} \in \mathbb{C}^{n \times f}$. We define a complex random walk as the diffusion process governed by the following PDE:*

$$\dot{\mathbf{X}}(t) = -\mathbf{L}_{rw}\mathbf{X}(t). \tag{3}$$

Thus, its Euler discretization with a unit step is:

$$\mathbf{X}(t+1) = (\mathbf{I} - \mathbf{L}_{rw})\mathbf{X}(t) = \mathbf{P}\mathbf{X}(t). \tag{4}$$

An important class of complex-weighted graphs is that of structurally balanced graphs, where the sum of the edge phases along any cycle is an integer multiple of 2π [16, 18]. These graphs can be characterized in terms of a partition of the node set, and their asymptotic behavior under a complex random walk has been analyzed in the infinite-time limit [16]. Building on these results, we show in the next section that any node-classification task can be solved in the steady state of a complex random walk, provided the graph is equipped with a suitable complex-weighted structure defined by the task. We further prove that such weights always exist (see Theorem 1), establishing a general expressiveness guarantee for our framework. Additional details on balanced graphs and their properties are provided in Appendix A or can be found in [16].

3 The Expressive Power of Complex-Weighted Diffusion

Bodnar et al. [13] showed that performing diffusion in a real-weighted graph is not sufficient to guarantee expressivity in an arbitrary classification task. In this section, we prove the full expressive power of complex-weighted diffusion, which constitutes the main result of this paper.

Theorem 1. *Let $G = (V, E)$ be an unweighted, undirected graph and $\mathcal{V} = \{V_i\}_{i=1}^{l_p}$ a partition of its nodes. Then, there exists a complex-weighted graph $G' = (V, E, \mathbf{W})$ such that, starting from any initial features $\mathbf{x}(0) = (x_i(0))$, in the steady state of a complex random walk, the features of the nodes belonging to a subset V_l have the same phase, and this phase is different for each subset.*

Proof sketch. The proof builds upon the results on complex-weighted graphs developed by Tian and Lambiotte [16], summarized in Appendix A. According to Proposition 2, the partition $\mathcal{V}$ characterizes a structurally balanced graph satisfying the following properties: (i) any edge within each node subset in $\mathcal{V}$ has phase 0; (ii) any edge between the same pair of node subsets in $\mathcal{V}$ has the same phase; and (iii) if we define the graph $\tilde{G}$ considering each node subset in $\mathcal{V}$ as a super node, the phase of every cycle in $\tilde{G}$ is 0. In addition, Proposition 3 characterizes the steady state of a complex random walk on any structurally balanced graph in terms of its associated partition.

To prove the theorem, we first show that for any given partition $\mathcal{V}$, it is possible to assign complex weights to the edges of the graph G so that conditions (i)–(iii) above are satisfied. The proof of this result proceeds by constructing a super graph $\tilde{G}$ that determines the complex weight corresponding to each pair of subsets of $\mathcal{V}$ such that (iii) holds. Specifically, we first build a cycle basis of $\tilde{G}$ composed of triangles and verify that each element of the basis satisfies (iii). We then extend this property to

[4] Note that we work with undirected unweighted graphs, but directed complex-weighted graphs.

any other triangle and, consequently, to all cycles in $\tilde{G}$. This construction determines the desired complex-weighted graph G'.

Finally, applying Proposition 3 to the obtained graph G' shows that, in the long time limit of a complex random walk, the features of nodes belonging to different subsets of $\mathcal{V}$ converge to complex numbers with different phases. The full proof can be found in Appendix A. □

Hence, as in Figure 1, every graph can be assigned complex weights such that in the steady state of a complex random walk, only nodes belonging to the same class have the same phase. Then, the nodes' features are linearly separable in the asymptotic time limit, which shows that solving any node classification task can be reduced to performing diffusion with the right complex-weighted structure.

4 Complex-Weighted Convolutional Networks

Analogously to how GCN augments heat diffusion, we build a Complex-Weighted Convolutional Network (CWCN) that augments the complex random walk diffusion process. In addition, we propose a method to learn the complex-weighted structure from data.

4.1 Complex-Weighted Convolutions

Let G be a complex-weighted graph with initial feature matrix $\mathbf{X}(0) \in \mathbb{C}^{n\times f}$, complex weight matrix $\mathbf{W} \in \mathbb{C}^{n\times n}$, degree matrix $\mathbf{Q} \in \mathbb{R}^{n\times n}$ and recall $\mathbf{P} = \mathbf{Q}^{-1}\mathbf{W}$, $\mathbf{L}_{rw} = \mathbf{I} - \mathbf{P}$. We equip the diffusion process given by Equation (3) with learnable weight matrices $\mathbf{M}_1 \in \mathbb{C}$, $\mathbf{M}_2 \in \mathbb{C}^{f\times f}$ and a non-linearity σ to arrive at a model whose evolution is governed by:

$$\dot{\mathbf{X}}(t) = -\sigma(\mathbf{L}_{rw}(\mathbf{I}_n \otimes \mathbf{M}_1)\mathbf{X}(t)\mathbf{M}_2) = -\sigma((\mathbf{I} - \mathbf{P})(\mathbf{I}_n \otimes \mathbf{M}_1)\mathbf{X}(t)\mathbf{M}_2), \tag{5}$$

where f is the number of channels and $\otimes$ denotes the Kronecker product. $\mathbf{M}_1$ applies a rotation to the node features in all the channels, while $\mathbf{M}_2$ allows mixing each node's feature channels. Note that by setting $\mathbf{M}_1$, $\mathbf{M}_2$ and σ to the identity, we recover the complex random walk diffusion equation. Therefore, the model is at least as expressive as complex random walk diffusion and can benefit from the linear separation power described in Theorem 1. We focus on the time-discretised version of this model which allows us to use a new set of learnable weight matrices $\mathbf{M}_1^l \in \mathbb{C}, \mathbf{M}_2^l \in \mathbb{C}^{f\times f}$ at each layer $l = 0, \ldots, L-1$:

$$\mathbf{X}(l+1) = \mathbf{X}(l) - \sigma((\mathbf{I} - \mathbf{P})(\mathbf{I}_n \otimes \mathbf{M}_1^l)\mathbf{X}(l)\mathbf{M}_2^l) \in \mathbb{C}^{n\times f}. \tag{6}$$

Since the formulation in Equation (6) requires the initial feature matrix $\mathbf{X}(0) \in \mathbb{C}^{n\times f}$, we first use a multilayer perceptron (MLP) to map the raw node features to a real-valued matrix $\mathbf{X}(0) \in \mathbb{R}^{n\times 2f}$, which we then interpret as a complex matrix in $\mathbb{C}^{n\times f}$. Similarly, we interpret the final representation $\mathbf{X}(L)$ as a real-valued matrix in $\mathbb{R}^{n\times 2f}$ and apply a final MLP to perform node classification.

4.2 Complex Weights Learning

In general, an appropriate complex-weighted structure that guarantees linear separability in the time limit of the diffusion process cannot be determined without full knowledge of the node classes. Therefore, we aim to learn the underlying complex structure from data, which will allow the model to choose the appropriate complex weight matrix $\mathbf{W}$ for each node classification task. Specifically, we define the complex weight matrix $\mathbf{W}$ as a learnable function of the initial node features $\mathbf{X}(0)$:

$$\mathbf{W} = g(G, \mathbf{X}(0); \theta), \tag{7}$$

where θ are learnable parameters. Once we have learned the complex weight matrix $\mathbf{W}$, we can obtain $\mathbf{Q}$ the degree matrix and compute $\mathbf{P} = \mathbf{Q}^{-1}\mathbf{W}$.

The complex weight matrix $\mathbf{W} \in \mathbb{C}^{n\times n}$ is learned using a parametric matrix-valued function. The weight corresponding to an edge (v_i, v_j), W_{ij}, is computed based on the initial features of nodes v_i and v_j, which we denote by $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{C}^f$. Formally, the weight matrix is given by:

$$W_{ij} = \Phi(\mathbf{x}_i, \mathbf{x}_j), \tag{8}$$

where Φ is a learnable function satisfying $\Phi(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_j, \mathbf{x}_i)^*$, since $\mathbf{W}$ must be Hermitian.

In practice, we set Φ to be an MLP and interpret $\mathbf{x}_i$ and $\mathbf{x}_j$ as real vectors in $\mathbb{R}^k$ with $k = 2f$. For brevity, we use the same notation for both complex and real forms, as the intended interpretation will be clear from the context. Given a pair of nodes (v_i, v_j) with corresponding feature vectors $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^k$, we define $\Phi(\mathbf{x}_i, \mathbf{x}_j)$ as follows:

$$\Phi(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} \tilde{\sigma}(\mathbf{V}[\mathbf{x}_i || \mathbf{x}_j]), & \text{if } (v_i, v_j) \in E \text{ and } i \leq j \\ \overline{\tilde{\sigma}(\mathbf{V}[\mathbf{x}_i || \mathbf{x}_j])}, & \text{if } (v_i, v_j) \in E \text{ and } i > j \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Here, $||$ denotes vector concatenation, $\mathbf{V} \in \mathbb{R}^{2 \times 2k}$ is a learnable weight matrix and $\tilde{\sigma}$ is a non-linear function. The overline $\overline{(a, b)}$ denotes the complex conjugate when interpreting $(a, b) \in \mathbb{R}^2$ as a complex number, i.e., $\overline{(a, b)} = (a, -b)$. Finally, we interpret $\Phi(\mathbf{x}_i, \mathbf{x}_j) \in \mathbb{R}^2$ as a complex number.

The following result shows that if the function Φ has enough capacity and the features are diverse enough, we can learn any Hermitian complex weight matrix.

Proposition 1. *Let $G = (V, E)$ be a finite graph with feature matrix $\mathbf{X} \in \mathbb{C}^{n \times f}$. If the node features $(\mathbf{x}_i, \mathbf{x}_j) \neq (\mathbf{x}_k, \mathbf{x}_t)$ for any $(v_i, v_j) \neq (v_k, v_t) \in E$ and Φ defined in (9) is an MLP with sufficient capacity, then Φ can learn any complex-weighted structure $G' = (V, E, \mathbf{W})$.*

Proof sketch. Define the set $S := \{(\mathbf{x}_i, \mathbf{x}_j) : (v_i, v_j) \in E\} \subset \mathbb{C}^{2f}$ and the function $g \colon S \to \mathbb{C}$, $g(\mathbf{x}_i, \mathbf{x}_j) = W_{ij}$. Interpret S as a subset of $\mathbb{R}^{2k}$, with $k = 2f$, and $g \colon S \to \mathbb{R}^2$. First, we show that g can be extended to a smooth function $f \colon \mathbb{R}^{2k} \to \mathbb{R}^2$. Then, f can be approximated by an MLP with sufficient capacity [19, 20] and, thus, so does g. Therefore, Φ defined in Equation (9) can learn any complex weight matrix $\mathbf{W}$ of the graph G. For more details, see Appendix B. □

5 Experiments

Having shown theoretically the expressivity of our model, we evaluate it through both synthetic and real-world experiments. The synthetic experiments show the benefits of using complex-weighted diffusion in controlled heterophilic settings, while the real-world experiments assess performance on benchmark datasets against several baseline models, providing an overview of the model's capabilities.

5.1 Synthetic Experiments

These experiments constitute an ablation study aimed at assessing whether complex-weighted diffusion provides advantages in heterophilic settings compared to its real-weighted counterpart, as suggested by our theoretical analysis. A real random walk can be defined analogously to Definitions 2 and 3 (see [21] for details). While Bodnar et al. [13] demonstrate that real-valued diffusion fails to separate classes in the time limit for certain tasks, Theorem 1 establishes that complex diffusion can achieve linear separation in its steady state. To isolate the effect of complex weights, we evaluate learnable vanilla real and complex random walks by setting $\mathbf{M}_1 = 1$, $\mathbf{M}_2 = \mathbf{I}_f$, and $\sigma = \mathrm{id}$ in Equation (6), so that only the weight matrix $\mathbf{W}$ is trainable. To ensure a fair comparison, we use a single feature channel ($f = 1$) for the complex case and two channels for the real case, ensuring that both models have the same number of learnable parameters. Finally, to analyse the asymptotic behaviour, we set the number of layers to 20.

We first consider a cycle graph containing two nodes per class, where nodes of the same class are placed at opposite positions on the cycle (Figure 5.1, bottom-left) and node features are sampled from distinct class-specific Gaussian distributions. This setup provides a clear example of heterophily: nodes of the same class are distant in the graph topology, and information must diffuse through an increasing number of intermediate nodes, proportional to the number of classes, before reaching another node of the same class.

Next, we design a more challenging heterophilic scenario. We consider a Stochastic Block Model (SBM) with 10 nodes per cluster and 2 clusters per class. The clusters are arranged in a ring topology, where edges are formed only between adjacent clusters, and no intra-cluster connections are present. Clusters positioned opposite to each other on the ring share the same class label and node features are sampled from class-specific Gaussian distributions. Figure 5.1 (bottom-right) illustrates this configuration for the case of 3 nodes per cluster.

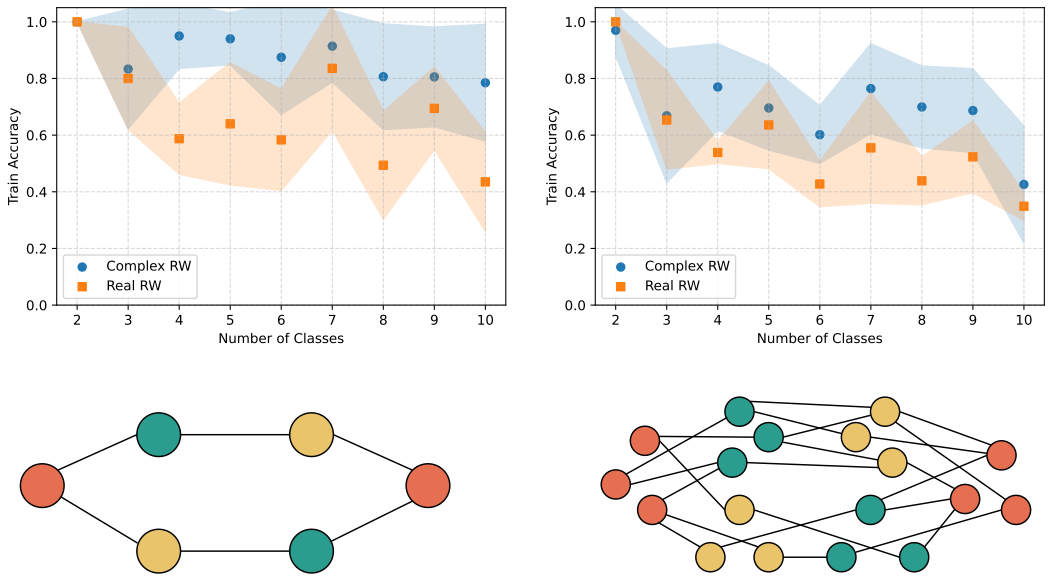

Figure 2: Training accuracy (top) and examples of the processed graphs (bottom) for learnable complex versus real random walks, varying the numbers of classes in two heterophilic settings: a cycle graph with same-class nodes in opposite positions (left) and a ring of clusters with same-class clusters in opposite positions (right). The complex random walk consistently achieves higher mean accuracy (dots) than its real-valued counterpart (shade: standard deviation over 10 random seeds).

Figure 5.1(top) the classification accuracy for both settings as the number of classes increases, averaged over ten random seeds. Consistent with our theoretical findings, the complex random walk consistently outperforms its real-valued counterpart. Notably, in the two-class setting both methods achieve perfect accuracy, which aligns with the observations of Bodnar et al. [13], who showed that real-weighted diffusion is sufficient when only two classes are present. Overall, these experiments empirically confirm the intrinsic advantage of complex-weighted diffusion in heterophilic graphs, in agreement with our theoretical results.

5.2 Real-World Experiments

Given the conceptual similarity between our model and SCN [13], a direct comparison is particularly relevant. To ensure fairness, we follow the same experimental procedure described in [13].

Datasets. We evaluate our model on several real-world graphs [15, 22, 23]. These datasets exhibit varying degrees of edge homophily h, with values ranging from $h = 0.11$ (very heterophilic) to $h = 0.81$ (very homophilic). This diversity allows us to assess our model's robustness under different homophily conditions. Following [13], we evaluate our model using the 10 fixed data splits provided by [15]. For each split, 48% of the nodes in each class are used for training, 32% for validation, and the remaining 20% for testing. We report the mean accuracy and standard deviation across 10 splits.

Baselines. As baselines, we include SCN [13], along with the same selection of Graph Neural Network (GNN) models used in their study. These baselines can be classified into three categories: (1) classical: GCN [17], GAT [24], GraphSAGE [25]; (2) models designed for heterophilic settings: GGCN [26], Geom-GCN [15], H2GCN [8], GPRGNN [27], FAGCN [28], MixHop [29]; and (3) models designed to address oversmoothing: GCNII [30], PairNorm [31]. All baseline results are reported as presented in [13]. For SCN [13], we select O(d)-NSD, the variant that achieves the best average performance. Finally, we include two additional diffusion-based models: BLEND [32] and GREAD [14] (selecting the variant with best average performance). [14].

Results. The results are summarized in Table 1. First, CWCN significantly outperforms classical GNNs on heterophilic datasets, supporting our theoretical claims regarding CWCN's improved expressivity over GCNs and demonstrating that these advantages translate into practical performance

Table 1: Accuracy results (mean test accuracy $\pm$ standard deviation) on node classification datasets, sorted by homophily level. The top four models are highlighted in **First**, **Second**, **Third**, **Fourth**. The background color of the model name: green for models that provide expressive power guarantees for an infinite number of layers, yellow for models that provide them only under certain constraints, and grey for models without expressiveness guarantees. Our model is denoted as CWCN, and the gap to the best model is computed in % as $\frac{\text{Acc}_{\text{bestModel}} - \text{Acc}_{\text{CWCN}}}{\text{Acc}_{\text{bestModel}}} \cdot 100$. Table adapted and modified from [13].

	Texas	Wisconsin	Film	Chameleon	Cornell	Citeseer	Pubmed	Cora	Avg.
Hom level	**0.11**	**0.21**	**0.22**	**0.23**	**0.30**	**0.74**	**0.80**	**0.81**	
#Nodes	183	251	7,600	2,277	183	3,327	18,717	2,708	
#Edges	295	466	26,752	31,421	280	4,676	44,327	5,278	
#Classes	5	5	5	5	5	6	3	7	
CWCN	**84.05**$\pm$6.45	**86.27**$\pm$4.20	36.51 $\pm$1.26	**65.59** $\pm$1.33	83.51$\pm$8.15	76.37$\pm$1.53	89.23 $\pm$0.49	**87.93**$\pm$1.03	**76.18**
Gap to best model (%)	5.48	3.51	3.67	8.11	3.44	2.11	1.11	0.72	3.95
GREAD	**88.92**$\pm$3.72	**89.41**$\pm$3.30	**37.90**$\pm$1.17	**71.38**$\pm$1.31	**86.49**$\pm$7.15	**77.60**$\pm$1.81	**90.23**$\pm$0.55	**88.57**$\pm$0.66	**78.81**
SCN	**85.95**$\pm$6.95	**89.41**$\pm$4.74	**37.81**$\pm$1.15	**68.04**$\pm$1.58	**84.86**$\pm$4.71	76.70$\pm$1.57	**89.49**$\pm$0.40	86.90$\pm$1.13	**77.39**
BLEND	83.24$\pm$4.64	84.12$\pm$3.56	35.63$\pm$1.01	60.11$\pm$2.09	**85.95**$\pm$6.82	76.63$\pm$1.60	89.24$\pm$0.42	86.90$\pm$1.13	75.23
GGCN	**84.86**$\pm$4.55	**86.86**$\pm$3.29	**37.54**$\pm$1.56	**71.14**$\pm$1.84	**85.68**$\pm$6.63	**77.14**$\pm$1.45	89.15$\pm$0.37	**87.95**$\pm$1.05	**77.54**
H2GCN	**84.86**$\pm$7.23	**87.65**$\pm$4.98	35.7$\pm$1.00	60.11$\pm$2.15	82.70$\pm$5.28	77.11$\pm$1.57	88.49$\pm$0.38	87.87$\pm$1.20	75.56
GPRGNN	78.36$\pm$4.31	82.94$\pm$4.21	34.63$\pm$1.22	46.58$\pm$1.71	80.27$\pm$8.11	77.13$\pm$1.67	87.54$\pm$0.38	**87.95**$\pm$1.18	71.92
FAGCN	82.43$\pm$6.89	82.94$\pm$7.95	34.87$\pm$1.25	55.22$\pm$3.19	79.19$\pm$9.79	N/A	N/A	N/A	-
MixHop	77.84$\pm$7.73	75.88$\pm$4.90	32.22$\pm$2.34	60.50$\pm$2.53	73.51$\pm$6.34	76.26$\pm$1.33	85.31$\pm$0.61	87.61$\pm$0.85	71.14
GCNII	77.57$\pm$3.83	80.39$\pm$3.40	**37.44**$\pm$1.30	63.86$\pm$3.04	77.86$\pm$3.79	**77.33**$\pm$1.48	**90.15**$\pm$0.43	**88.37**$\pm$1.25	74.12
Geom-GCN	66.76$\pm$2.72	64.51$\pm$3.66	31.59$\pm$1.15	60.00$\pm$2.81	60.54$\pm$3.67	**78.02**$\pm$1.15	**89.95**$\pm$0.47	85.35$\pm$1.57	67.09
PairNorm	60.27$\pm$4.34	48.43$\pm$6.14	27.40$\pm$1.24	62.74$\pm$2.82	58.92$\pm$3.15	73.59$\pm$1.47	87.53$\pm$0.44	85.79$\pm$1.01	63.08
GraphSAGE	82.43$\pm$6.14	81.18$\pm$5.56	34.23$\pm$0.99	58.73$\pm$1.68	75.95$\pm$5.01	76.04$\pm$1.30	88.45$\pm$0.50	86.90$\pm$1.04	72.99
GCN	55.14$\pm$5.16	51.76$\pm$3.06	27.32$\pm$1.10	64.82$\pm$2.24	60.54$\pm$5.30	76.50$\pm$1.36	88.42$\pm$0.50	86.98$\pm$1.27	63.93
GAT	52.16$\pm$6.63	49.41$\pm$4.09	27.44$\pm$0.89	60.26$\pm$2.50	61.89$\pm$5.05	76.55$\pm$1.23	87.30$\pm$1.10	86.33$\pm$0.48	62.67

gains. On homophilic datasets, our model also generally performs better, although the margins are smaller. Second, CWCN remains competitive across all datasets, with its performance deviating by an average of 3.52% from the best-performing model (4.84% for heterophilic datasets and 1.31% for homophilic datasets). Overall, CWCN ranks 4th in average empirical performance among all evaluated models while providing provable expressiveness guarantees for an infinite number of layers without additional constraints.

6 Discussion and Related Work

Heterophily and Oversmoothing. Heterophilic graphs challenge the homophily assumption underlying many GNNs. To address this, several strategies have been proposed. MixHop [29] aggregates information from higher-order neighbourhoods to capture long-range dependencies; Geom-GCN [15] redefines the notion of neighbourhood; FAGCN [28], H2GCN [8] and GGCN [26] model the relative importance of neighbours during aggregation; and GPRGNN [27] integrates representations from multiple layers to jointly leverage local and global structural information.

To mitigate oversmoothing, a variety of methods have been proposed. Architecture-agnostic techniques include residual or skip connections to preserve information flow across layers [33, 34], normalization methods to limit feature homogenization [31], and graph rewiring to enhance connectivity [35]. Architectures such as GCNII [30] and PairNorm [31] exemplify these approaches.

These methods offer practical mechanisms to address heterophily and oversmoothing. While many include theoretical analyses that highlight the models' advantages, they generally lack theoretical guarantees regarding node features separability as the number of layers increases. In contrast, CWCN not only ensures such guarantees but also achieves superior empirical performance in heterophilic settings compared to most of these models, being only slightly outperformed by GGCN [26].

Diffusion on GNNs. More recently, several approaches jointly address oversmoothing and heterophily by modifying the underlying message-passing dynamics [13, 14, 32, 36]. A notable example is the Sheaf Convolutional Network (SCN), introduced by Hansen and Gebhart [12] and later ex-

tended into a practical learning framework by Bodnar et al. [13]. SCNs increase the expressive power of heat diffusion by equipping the graph with a *cellular sheaf* [37], enabling a diffusion process based on the sheaf Laplacian. In this setup, the sheaf structure is learned from data, and sheaf diffusion is augmented to build a GNN architecture. Another prominent approach is the Graph Neural Reaction-Diffusion Network (GREAD) [14], which models feature propagation through reaction-diffusion equations. Since our method also introduces a novel diffusion process to redefine message passing, it naturally belongs to this family.

While GREAD achieves state-of-the-art results on standard node-classification benchmarks, it does not provide formal theoretical guarantees. SCNs, in contrast, offer provable expressiveness: Bodnar et al. [13] show that, in the time limit of sheaf diffusion, any node classification task can theoretically be solved, provided the graph is equipped with an appropriate sheaf structure. However, these guarantees rely on the sheaf dimension scaling with the number of target classes. Without such scaling, sheaf diffusion only guarantees linear separation power for regular graphs. Importantly, increasing the sheaf dimension introduces additional learnable parameters and enlarges the diffusion matrix, leading to higher computational cost. Furthermore, compared to standard GNNs, GREAD requires multiple additional hyperparameters to parameterize the reaction–diffusion process, while SCNs add only the sheaf dimension as hyperparameter.

Advantages of CWCNs. CWCN achieves competitive performance on node-classification benchmarks, surpassing most existing architectures designed to mitigate heterophily and oversmoothing, while performing only slightly below top-performing diffusion-based models. Importantly, unlike other architectures, CWCN achieves linear separation power in the diffusion time limit using a fixed-size matrix that is independent of the number of classes. In addition, it introduces no extra hyperparameters beyond those of standard GNNs. Thus, CWCN provides a simpler formulation that requires less hyperparameter tuning while offering stronger theoretical guarantees.

Limitations of CWCNs. Despite its theoretical strengths and simplicity, CWCN has two main limitations. First, there remains an empirical performance gap between CWCN and the best-performing methods such as GREAD and SCN. Second, CWCN relies on complex-valued weights, which require additional matrix operations during both training and inference. This introduces overhead in backpropagation and leads to higher computational cost compared to real-valued GNN architectures.

Complex GNNs. Incorporating complex values into graph learning has recently attracted attention from various communities, but the existing approaches differ fundamentally from our framework. For instance, CayleyNets [38] extend the ChebNet paradigm by employing Cayley polynomials as spectral filters, which involve learnable complex coefficients; however, the resulting filters remain real-valued, and signals remain consistently real throughout the process. More broadly, in network science, there has been a growing interest in systematically extending classical concepts to complex-weighted networks, e.g., [39]. This line provides useful foundations but does not design GNN models for downstream applications. Closer to GNNs, MagNet and related works [40, 41] employ a complex Laplacian matrix (e.g., magnetic Laplacian) to capture directed edge information, typically with one or two global phase parameters. In contrast, our framework allows arbitrary complex phases per edge, leading to a richer diffusion process. More importantly, unlike prior studies, we provide theoretical guarantees on the expressivity of our complex-weighted diffusion process. This combination of flexible modeling and rigorous theory distinguishes our contribution.

7 Conclusion and Future Work

In this work, we introduced a novel framework for enhancing the expressive power of GNNs through diffusion on complex-weighted graphs. We first established the full expressive power of complex-weighted diffusion, demonstrating its potential to address two well-known limitations of standard GNN architectures: oversmoothing and poor performance on heterophilic graphs.

Building on this theoretical insight, we proposed CWCN, a GNN architecture that augments complex random-walk diffusion with learnable parameters and nonlinear activations. We further introduced a mechanism to learn the complex-weighted structure directly from data, allowing the model to adaptively capture the most suitable "geometry" for a given task. Compared to prior methods, the resulting framework is simpler—requiring fewer hyperparameters—while being supported by stronger theoretical guarantees.

Empirical evaluations on standard node-classification benchmarks show that CWCN achieves competitive performance, although further work is needed to determine whether our model can be refined to close the gap with top-performing methods. Promising directions for future work include investigating alternative architectures based on complex random walks, exploring other types of diffusion processes on complex-weighted graphs, incorporating a reaction term into the complex-weighted diffusion equation or, interpreting complex multiplication as a rotation in two dimension, considering transformations along the walks in higher dimensions [42]. Additionally, reducing the computational overhead associated with complex weights is an important avenue for improvement.

Overall, our findings suggest that incorporating complex-weighted diffusion provides a powerful approach to designing more expressive GNNs. By enriching the message passing dynamics with a complex-weighted structure, we open new possibilities for designing models that are both theoretically grounded and practically effective. To the best of our knowledge, this is the first work to leverage complex weights to enhance GNN expressiveness, and we hope it inspires further exploration of their potential in graph-based learning tasks.

Acknowledgements

C.L.A. acknowledges that the project that gave rise to these results received the support of a fellowship from the "la Caixa" Foundation (ID 100010434). The fellowship code is LCF/BQ/EU24/12060069. T.S. acknowledges financial support from the Rhodes Trust and the EPSRC Centre for Doctoral Training in Mathematics of Random Systems: Analysis, Modelling and Simulation (ESPRC Grant EP/S023925/1). R.L. acknowledges support from the EPSRC grants EP/V013068/1, EP/V03474X/1 and EP/Y028872/1.

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, 2008. 1

[2] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pages 417–426, 2019. 1

[3] 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. 1

[4] Tao Guo and Baojiang Cui. Web page classification based on graph neural network. In *Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 188–198. Springer International Publishing, 2022. 1

[5] Daniel Cummings and Marcel Nassar. Structured citation trend prediction using graph neural networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3897–3901. IEEE, 2020. 1

[6] Xin Zheng, Yi Wang, Yixin Liu, Ming Li, Miao Zhang, Di Jin, Philip S Yu, and Shirui Pan. Graph neural networks for graphs with heterophily: A survey. *arXiv preprint arXiv:2202.07082*, 2022. 1

[7] T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993*, 2023. 1

[8] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in neural information processing systems*, 33:7793–7804, 2020. 2, 7, 8

[9] 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. 2, 3, 18

[10] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*, 2020. 2, 3

[11] Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020. 2, 3

[12] Jakob Hansen and Thomas Gebhart. Sheaf neural networks. In *NeurIPS 2020 Workshop on Topological Data Analysis and Beyond*, 2020. 2, 8

[13] Cristian Bodnar, Francesco Di Giovanni, Benjamin Chamberlain, Pietro Lio, and Michael Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in gnns. *Advances in Neural Information Processing Systems*, 35:18527–18541, 2022. 2, 4, 6, 7, 8, 9, 17

[14] Jeongwhan Choi, Seoyoung Hong, Noseong Park, and Sung-Bae Cho. Gread: Graph neural reaction-diffusion networks. In *International Conference on Machine Learning*, pages 5722–5747. PMLR, 2023. 2, 7, 8, 9

[15] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020. 2, 7, 8

[16] Yu Tian and Renaud Lambiotte. Structural balance and random walks on complex networks with complex weights. *SIAM Journal on Mathematics of Data Science*, 6(2):372–399, 2024. 2, 4, 13, 14

[17] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 3, 7

[18] C. Lange, S. Liu, N. Peyerimhoff, and O. Post. Frustration index and Cheeger inequalities for discrete and continuous magnetic Laplacians. *Calc. Var.*, 54:4165—-4196, 2015. doi: 10.1007/s00526-015-0935-x. 4

[19] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991. 6, 17

[20] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989. 6, 17

[21] Yu Tian and Renaud Lambiotte. Spreading and structural balance on signed networks. *SIAM Journal on Applied Dynamical Systems*, 23(1):50–80, 2024. 6

[22] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016. 7

[23] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021. 7

[24] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018. accepted as poster. 7

[25] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017. 7

[26] Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 1287–1292. IEEE, 2022. 7, 8

[27] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*, 2021. 7, 8

[28] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 3950–3957, 2021. 7, 8

[29] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 21–29. PMLR, 09–15 Jun 2019. 7, 8

[30] 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. 7, 8

[31] Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in {gnn}s. In *International Conference on Learning Representations*, 2020. 7, 8

[32] Benjamin Chamberlain, James Rowbottom, Davide Eynard, Francesco Di Giovanni, Xiaowen Dong, and Michael Bronstein. Beltrami flow and neural diffusion on graphs. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 1594–1609. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/0cbed40c0d920b94126eaf5e707be1f5-Paper.pdf. 7, 8

[33] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9267–9276, 2019. 8

[34] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, pages 5453–5462. PMLR, 2018. 8

[35] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019. 8

[36] Sohir Maskey, Raffaele Paolino, Aras Bacho, and Gitta Kutyniok. A fractional graph laplacian approach to oversmoothing. *Advances in Neural Information Processing Systems*, 36:13022–13063, 2023. 8

[37] Justin Michael Curry. *Sheaves, cosheaves and applications*. University of Pennsylvania, 2014. 9

[38] 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, 2019. doi: 10.1109/TSP.2018.2879624. 9

[39] Lucas Böttcher and Mason A. Porter. Complex networks with complex weights. *Phys. Rev. E*, 109:024314, Feb 2024. doi: 10.1103/PhysRevE.109.024314. URL https://link.aps.org/doi/10.1103/PhysRevE.109.024314. 9

[40] Xitong Zhang, Yixuan He, Nathan Brugnone, Michael Perlmutter, and Matthew Hirn. Magnet: A neural network for directed graphs. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 27003–27015. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/e32084632d369461572832e6582aac36-Paper.pdf. 9

[41] Dong Xu, Ziye Liu, Fengming Li, and Yulong Meng. Complex graph neural networks for multi-hop propagation. *Neurocomputing*, 644:130364, 2025. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2025.130364. 9

[42] Yu Tian, Sadamori Kojaku, Hiroki Sayama, and Renaud Lambiotte. Matrix-weighted networks for modeling multidimensional dynamics: Theoretical foundations and applications to network coherence. *Physical Review Letters*, 134(23):237401, 2025. 10

[43] Lukas Biewald. Experiment tracking with weights and biases. https://www.wandb.com/, 2020. Software available from wandb.com. 17

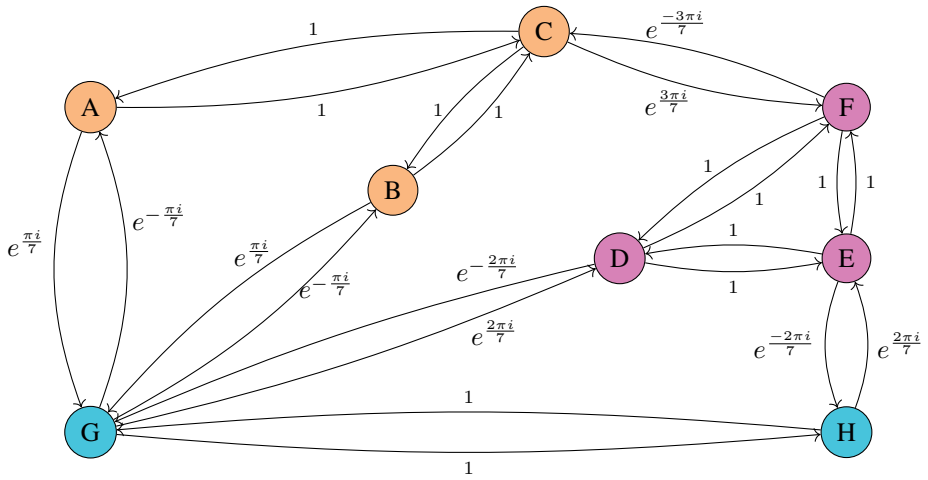

Figure 3: Example of a balanced complex-weighted graph, where the node set is partitioned into three subsets satisfying the conditions of Proposition 2.

A Complex Random Walk Proofs

In this Appendix, we first summarize the main results of [16] on balanced graphs and their behavior under complex random walks. We then use these results to show that any node classification task can be performed in the time limit of a complex random walk, if the graph is equipped with a suitable complex-weighted structure.

A.1 Complex Random Walks on Balanced Graphs

Notation 1. *We express elements in the complex weight matrix in polar coordinates as $W_{ij} = r_{ij}e^{i\varphi_{ij}}$, where $r_{ij} \geq 0$ indicates the magnitude and $\varphi_{ij} \in [0, 2\pi)$ is the phase. Thus, since $\mathbf{W}$ is Hermitian, $r_{ij} = r_{ji}$ and $\varphi_{ij} = -\varphi_{ji} + 2\pi$.*

Next, we define the notion of structural balance in a complex-weighted graph and present necessary and sufficient conditions under which a graph is structurally balanced. To this end, we first introduce the concept of the phase of a path.

Definition 4. *Let $G = (V, E, \mathbf{W})$ be a complex-weighted graph and $P = (e_1, \ldots, e_k), \quad e_i \in E$ be a path, the phase of P is:*

$$\theta(P) := \sum_{i=1}^{k} \theta(e_i) \bmod 2\pi,$$

where $\theta(e)$ returns the phase of edge e.

Definition 5. *A complex-weighted graph $G = (V, E, \mathbf{W})$ is structurally balanced if the phase of every cycle is 0.*

Remark 2. *Since $\mathbf{W}$ is Hermitian, if a cycle has phase θ, the cycle with reverse direction has phase $2\pi - \theta$. Then, a cycle has phase 0 iff its reversed cycle has phase 0, thus structural balance is well-defined.*

Next, we state a characterization of structurally balanced graphs, which is illustrated in Figure 3.

Proposition 2. *A complex-weighted graph G is balanced if and only if there is a partition of the nodes $\mathcal{V} = \{V_i\}_{i=1}^{l_p}$ such that:*

i) Any edges within each node subset in $\mathcal{V}$ have phase 0.

ii) Any edges between the same pair of node subsets in $\mathcal{V}$ have the same phase.

iii) If we define the graph G' considering each node subset in $\mathcal{V}$ as a super node, the phase of every cycle in G' is 0.

Proof. See [16]. □

The following result characterizes the asymptotic behaviour of a complex random walk on a balanced graph within the infinite time limit. Note that we state the proposition for one feature channel for simplicity, and the asymptotic behaviour of $\mathbf{X} \in \mathbb{C}^{n \times f}$ is defined by considering the behaviour of each channel separately.

Proposition 3. *Let G be a balanced and not bipartite complex-weighted graph with associated partition $\mathcal{V} = \{V_i\}_{i=1}^{l_p}$ (Proposition 2). Then, the steady state of a complex random walk is $\hat{\mathbf{x}} = (\hat{x}_j)$, with*

$$\hat{x}_j = e^{i\theta_{1\sigma(j)}} \frac{\mathbf{x}(0)^T \tilde{\mathbf{1}}_1 d_j}{2m}, \tag{10}$$

where:

- $\mathbf{x}(0) = (x_j(0)) \in \mathbb{C}^n$ *is the initial state vector.*
- $2m = \sum_{j=1}^{n} d_j$.
- θ_{hl} *is the phase of a path from nodes in V_h to nodes in V_l.*
- $\sigma(\,\cdot\,)$ *returns the index of the node subset in $\mathcal{V}$ that a node is associated with.*
- $\tilde{\mathbf{1}}_1$ *is the diagonal vector of $\mathbf{S}^*$, where $\mathbf{S}$ is the diagonal matrix whose (i,i) element is $e^{\theta_{1\sigma(i)}i}$.*

Proof. See [16]. □

A.2 The Linear Separation Power of Complex-Weighted Diffusion

Let us first prove that every graph can be endowed with an appropriate complex-weighted structure.

Proposition 4. *Let $G = (V, E)$ be an unweighted, undirected graph and $\mathcal{V} = \{V_i\}_{i=1}^{l_p}$ a partition of its nodes. Then, there exists a complex-weighted graph $G' = (V, E, \mathbf{W})$ such that:*

1. *It satisfies conditions (i), (ii) and (iii) of Proposition 2 for the partition $\{V_i\}_{i=1}^{l_p}$. Therefore, it is a balanced graph.*
2. *All the edges of condition (ii) between the subsets V_1 and V_i have different weight for each i.*

Proof. Let us first consider the graph with l_p nodes resulting from considering each node subset V_i of G as a super node. We denote this graph by $\tilde{G}$ and its nodes by $\tilde{v}_1, \ldots, \tilde{v}_{l_p}$, where each $\tilde{v}_i$ corresponds to the subset V_i. We will show that it is possible to assign weights to $\tilde{G}$ satisfying (iii) and such that the edges between nodes $\tilde{v}_1$ and $\tilde{v}_i$ are different for each i. We denote by $\tilde{\mathbf{W}} = (\tilde{W}_{ij})$ the complex weight matrix of $\tilde{G}$.

We assume w.l.o.g. that $\tilde{G}$ is complete (if condition (iii) holds for $\tilde{G}$ complete, it holds for any graph with l_p nodes, since removing edges does not add any new cycle to the graph).

In addition, for any complete graph it is possible to choose a cycle basis whose elements are all triangles. To see that, note that a cycle basis can be obtained from any spanning tree of $\tilde{G}$ by selecting the cycles formed by combining a path in the tree with a single edge outside the tree. Therefore, we can choose the fundamental cycle basis formed from the spanning tree with edges $\{(\tilde{v}_1, \tilde{v}_2), (\tilde{v}_1, \tilde{v}_3), \ldots, (\tilde{v}_1, \tilde{v}_{l_p})\}$. Denote this fundamental cycle basis by $\{T_1, \ldots, T_m\}$ and note that it contains every triangle of the form $(\tilde{v}_i, \tilde{v}_j, \tilde{v}_1)$, $i \neq j$.

We assign complex weights to $\tilde{G}$ in the following way. First, set $\tilde{W}_{1i} = e^{i\theta_i}$, choosing θ_i such that $e^{i\theta_i} \neq e^{i\theta_j}$ for all $i \neq j$. Then, set $\tilde{W}_{i1} = \overline{\tilde{W}_{i1}} = e^{-i\theta_i}$. Finally, for $k = 1, \ldots, m$, assign the weights $\tilde{W}_{ij}$ and $\tilde{W}_{ji}$ of the remaining edge e_k of each T_k so that the sum of the phases of the cycle T_k is 0. Note that $\tilde{W}_{ji} = \overline{\tilde{W}_{ij}}$ trivially. Note that two triangles of the basis cannot share the same edge e_k because of the way the fundamental basis is built, so this is well-defined.

Since $\tilde{G}$ is complete, every edge in $\tilde{G}$ belongs to some T_k. Therefore, we have assigned weights to every edge in $\tilde{G}$ such that the edges between nodes $\tilde{v}_1$ and $\tilde{v}_j$ are different for each j and (iii) is satisfied for the cycles $T_1, \ldots, T_m$. Next, we will prove that (iii) holds for every other triangle and, finally, that it holds for any cycle in $\tilde{G}$.

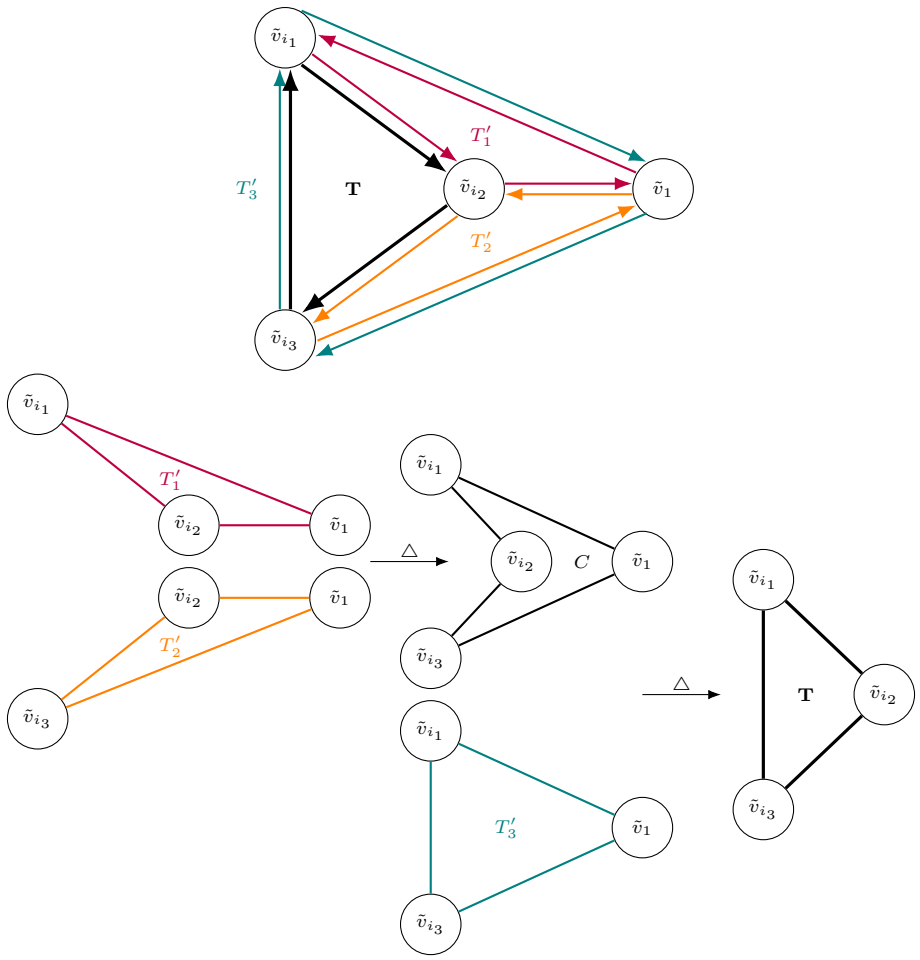

Figure 4: Illustration of how the triangle T in the proof of Proposition 4 can be obtained as $T = T'_1 \triangle T'_2 \triangle T'_3$.

Consider first a triangle $T = (\tilde{v}_{i_1}, \tilde{v}_{i_2}, \tilde{v}_{i_3})$ in $\tilde{G}$, such that $T \notin \{T_1, \ldots, T_m\}$, thus $\tilde{v}_{i_j} \neq \tilde{v}_1$ for $j = 1, 2, 3$. Define the triangles $T'_1 := (\tilde{v}_{i_1}, \tilde{v}_{i_2}, \tilde{v}_1)$, $T'_2 := (\tilde{v}_{i_2}, \tilde{v}_{i_3}, \tilde{v}_1)$, $T'_3 := (\tilde{v}_{i_3}, \tilde{v}_{i_1}, \tilde{v}_1)$ and the cycle $C = (\tilde{v}_{i_1}, \tilde{v}_{i_2}, \tilde{v}_{i_3}, \tilde{v}_1)$. Note that $T'_u \in \{T_1, \ldots, T_m\}$ for $u = 1, 2, 3$ and that:

$$T'_1 \triangle T'_2 = C \qquad C \triangle T'_3 = T,$$

where $\triangle$ is the symmetric difference. Then, T is expressed as a symmetric difference of basis triangles as $T = T'_1 \triangle T'_2 \triangle T'_3$. This is illustrated in Figure 4. Fix the orientation in T given by the order $(\tilde{v}_{i_1}, \tilde{v}_{i_2}, \tilde{v}_{i_3})$ and denote by $\theta_{12}, \theta_{23}, \theta_{31}$ the phases corresponding to the weights of this oriented cycle. Next, set the orientations of T'_1, T'_2 and T'_3 given by the orders $(\tilde{v}_{i_1}, \tilde{v}_{i_2}, \tilde{v}_1)$, $(\tilde{v}_{i_2}, \tilde{v}_{i_3}, \tilde{v}_1)$ and $(\tilde{v}_{i_3}, \tilde{v}_{i_1}, \tilde{v}_1)$, respectively, as illustrated in Figure 4.

Denote by θ_{*j} the phase of the weight $\tilde{W}_{1i_j}$ and θ_{j*} the phase of the weight $\tilde{W}_{i_j 1}$ (then, $\theta_{j*} = -\theta_{*j} \mod 2\pi$). Then, since T'_u are elements of the basis, we have:

$$(\theta_{12} + \theta_{2*} + \theta_{*1}) \bmod 2\pi = 0 \iff (\theta_{12} - \theta_{*2} + \theta_{*1}) \bmod 2\pi = 0$$
$$(\theta_{23} + \theta_{3*} + \theta_{*2}) \bmod 2\pi = 0 \iff (\theta_{23} - \theta_{*3} + \theta_{*2}) \bmod 2\pi = 0$$
$$(\theta_{31} + \theta_{1*} + \theta_{*3}) \bmod 2\pi = 0 \iff (\theta_{31} - \theta_{*1} + \theta_{*3}) \bmod 2\pi = 0$$

Then:

$$0 = (\theta_{12} - \theta_{*2} + \theta_{*1} + \theta_{23} - \theta_{*3} + \theta_{*2} + \theta_{31} - \theta_{*1} + \theta_{*3}) \bmod 2\pi = (\theta_{12} + \theta_{23} + \theta_{31}) \bmod 2\pi$$

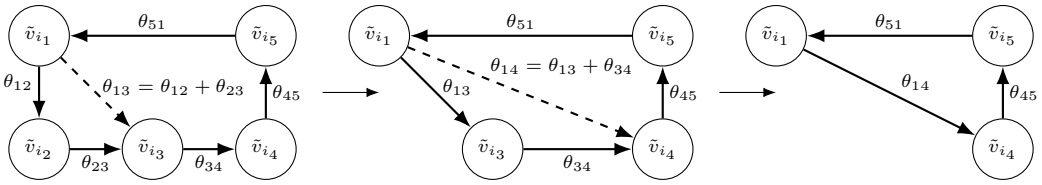

Figure 5: Illustration of the proof of Proposition 4: every cycle satisfies property (iii) of Proposition 2.

Note that a similar argument can be made if we fix the opposite orientation in T. Therefore, we have proven that every triangle in $\tilde{G}$ satisfies (iii).

Finally, let us prove that every cycle in $\tilde{G}$ satisfies (iii). Consider an oriented cycle $(\tilde{v}_{i_1}, \dots, \tilde{v}_{i_r})$ and denote by θ_{kl} the phase corresponding to the weight $\tilde{W}_{i_k i_l}$. Then:

$$\begin{aligned}(\theta_{12} + \theta_{23} + \cdots + \theta_{(r-1)r} + \theta_{r1}) \bmod 2\pi &= (\theta_{13} + \theta_{34} + \cdots + \theta_{(r-1)r} + \theta_{r1}) \bmod 2\pi \\ &= (\theta_{14} + \theta_{45} + \cdots + \theta_{(r-1)r} + \theta_{r1}) \bmod 2\pi \\ &\dots \\ &= (\theta_{1(r-1)} + \theta_{(r-1)r} + \theta_{r1}) \bmod 2\pi = 0,\end{aligned}$$

where we have each equality by substituting the sum of the weights of a 2-length path between two nodes by the weight of the edge joining the nodes (which forms a triangle). This process is illustrated in Figure 5. The last equality holds because $(\tilde{v}_{i_1 i_{r-1}}, \tilde{v}_{i_{r-1} i_r}, \tilde{v}_{i_r i_1})$ is a triangle.

Therefore, we have proven that every cycle in $\tilde{G}$ satisfies (iii).

Now consider $G = (V, E)$, denote its nodes by $v_1, \dots, v_n$ and define the function σ, where $\sigma(j)$ returns the index i of the node subset V_i to which node v_j belongs. Construct the matrix $\mathbf{W} = (W_{ij})$ by assigning weights in the following way:

$$W_{ij} = \begin{cases} 0, & \text{if } (v_i, v_j) \notin E \\ 1, & \text{if } \sigma(i) = \sigma(j) \\ \tilde{W}_{\sigma(i)\sigma(j)} & \text{otherwise} \end{cases}$$

By construction, $G' = (V, E, \mathbf{W})$ satisfies all the conditions of the Proposition. □

Therefore, we have proven that every graph can be assigned complex weights so that the resulting complex-weighted graph satisfies the hypothesis of Proposition 3. Thus, it is possible to describe its asymptotic behaviour in the time limit of a complex random walk.

Theorem 1. *Let $G = (V, E)$ be an unweighted, undirected graph and $\mathcal{V} = \{V_i\}_{i=1}^{l_p}$ a partition of its nodes. Then, there exists a complex-weighted graph $G' = (V, E, \mathbf{W})$ such that, starting from any initial features $\mathbf{x}(0) = (x_i(0))$, in the steady state of a complex random walk, the features of the nodes belonging to a subset V_l have the same phase, and this phase is different for each subset.*

Proof. First, it is possible to assign complex weights to G obtaining a balanced graph $G' = (V, E, \mathbf{W})$ that satisfies the conditions of Proposition 4. In addition, we assign self-loops to every node, $W_{ii} = 1 \quad \forall i = 1, \dots, n$, to ensure that G' is not bipartite.

Then, G' is in the conditions of Proposition 3, so we can obtain its steady state in a complex random walk using Equation (10). Note that the factor $\frac{\mathbf{x}(0)^T \tilde{\mathbf{1}}_1}{2m}$ does not depend on j, so it is common for all nodes. Therefore, two nodes v_j and v_k have different phases iff $e^{i\theta_{1\sigma(j)}} \neq e^{i\theta_{1\sigma(k)}}$.

By construction of G', $\theta_{1\sigma(j)} \mod 2\pi = \theta_{1\sigma(k)} \mod 2\pi$ iff $\sigma(j) = \sigma(k)$. Then, two nodes have the same phase iff they belong to the same subset of $\{V_i\}_{i=1}^{l_p}$. □

B Complex Weights Learning Proof

Proposition 1. *Let $G = (V, E)$ be a finite graph with feature matrix $\mathbf{X} \in \mathbb{C}^{n \times f}$. If the node features $(\mathbf{x}_i, \mathbf{x}_j) \neq (\mathbf{x}_k, \mathbf{x}_t)$ for any $(v_i, v_j) \neq (v_k, v_t) \in E$ and Φ defined in (9) is an MLP with sufficient capacity, then Φ can learn any complex-weighted structure $G' = (V, E, \mathbf{W})$.*

Proof. Let $\mathbf{W} \in \mathbb{C}^{n\times n}$ be a complex weight matrix for G, and denote its elements by $W_{ij} = a_{ij} + ib_{ij}$. Since $\mathbf{W}$ is Hermitian, $a_{ji} = a_{ij}$ and $b_{ji} = -b_{ij}$.

Consider the feature matrix $\mathbf{X}$ as a real-valued matrix $\mathbf{X} \in \mathbb{R}^{n\times k}$, with $k = 2f$.

Define the set $S := \{(\mathbf{x}_i, \mathbf{x}_j) : (v_i, v_j) \in E\} \subset \mathbb{R}^{2k}$. Since each $(\mathbf{x}_i, \mathbf{x}_j)$ is unique, the function $g\colon S \to \mathbb{R}^2$, $g(\mathbf{x}_i, \mathbf{x}_j) = (a_{ij}, b_{ij})$ is well-defined. We now show that g can be extended to a smooth function $f\colon \mathbb{R}^{2k} \to \mathbb{R}^2$.

Since $S = \{s_1, \dots, s_M\}$ is a finite set, for each $s_m \in S$, there exists a sufficiently small neighbourhood $U_m \subset \mathbb{R}^{2k}$ such that $s_m \in U_m$ and $s_p \notin U_p$ for all $m \neq p$. In addition, it is possible to find a smooth bump function $\psi_m\colon \mathbb{R}^{2k} \to \mathbb{R}$ such that $\psi_m(s_m) = 1$ and $\psi_m(x) = 0 \quad \forall x \notin U_m$. Then, the function $f(x) = \sum_{m=1}^{M} g(s_m)\psi_m(x)$ is smooth and $f|_S = g$. Since f is smooth, it can approximated by an MLP [19, 20] and, thus, so does g.

Now it is enough to identify $\mathbb{R}^2 \equiv \mathbb{C}$, with the first coordinate corresponding to the real part and the second coordinate to the imaginary part. Then, interpreting $S \subset \mathbb{C}^{2f}$, we have proven that $g\colon S \to \mathbb{C}$, $g(\mathbf{x}_i, \mathbf{x}_j) = a_{ij} + ib_{ij}$ can be approximated by an MLP with sufficient capacity. Therefore, Φ defined in Equation (9) can learn any complex weight matrix $\mathbf{W}$ of the graph G. □

C Additional implementation details and hyperparameters

Adjusting the activation magnitudes. Following the approach in [13], we found it useful in practice to learn additional parameters $\varepsilon^l = (\varepsilon_1^l, \varepsilon_2^l)$, with $\varepsilon_i^l \in [-1, 1]$, for each layer l. These parameters are used to compute the diffusion step as follows:

$$\mathbf{X}(l+1) = (1+\varepsilon^l)\mathbf{X}(l) - \sigma((\mathbf{I}-\mathbf{P})(\mathbf{I}_n \otimes \mathbf{M}_1^l)\mathbf{X}(l)\mathbf{M}_2^l) \in \mathbb{C}^{n\times f}$$

Here, $(1+\varepsilon^l)\mathbf{X}(l)$ denotes scaling the real part of $\mathbf{X}(l)$ by $\varepsilon_1^l + 1$ and the imaginary part by $\varepsilon_2^l + 1$. This mechanism allows the model to dynamically adjust the relative magnitudes of real and imaginary components at each layer. These learnable scaling parameters are used in all of our experiments.

Batch normalization. Proposition 1 guarantees that any complex weight matrix can be learned, provided the node features are sufficiently diverse. However, in practice, we observed that the initial node features $\mathbf{X}(0)$ tend to be very similar across nodes. This issue arises from the initial MLP, where the bias term often dominates in practice, causing the output features of all nodes to become nearly identical.

To address this problem, we insert a batch normalization layer immediately after the MLP. This normalizes each feature dimension across the batch of nodes, mitigating bias dominance, and promoting feature diversity. Batch normalization is used in all of our experiments.

Real learnable matrices. Complex matrix multiplication is more expensive than its real counterpart. To improve efficiency, we replace the complex matrices $\mathbf{M}_1^l \in \mathbb{C}, \mathbf{M}_2^l \in \mathbb{C}^{f\times f}$ in Equation (6) with real matrices of size 2×2 and $2f \times 2f$, respectively. The feature matrix $\mathbf{X}(l)$ is then treated as a real matrix in $\mathbb{R}^{n\times 2f}$. In practice, we observe no loss in performance with this substitution, and therefore adopt it in all our experiments.

Hyperparameters and Training Procedure. Following [13], we evaluate our model using the hyperparameter ranges listed in Table 2, where dropout has been included as a regularization technique to prevent overfitting. We assign different dropout rates to the initial layer and to the linear layers within the convolutional blocks. We train all models for a fixed maximum number of epochs and perform early stopping when the validation metric has not improved for a pre-specified number of patience epochs. We report the test results at the epoch where the best validation metric was obtained for the model configuration with the best validation score among all models. We use the hyperparameter optimisation tools provided by Weights and Biases [43] for this procedure.

D Additional Synthetic Experiments

In this section, we conduct further synthetic experiments that allow a direct comparison with standard heat diffusion in a controlled setting.

Table 2: Hyperparameter ranges used to evaluate our model in the real-world experiments.

Hyperparameter	**Values**
Hidden channels	(8,16,32)
Layers	2-8
Learning rate	0.02
Activation (regular layers)	ELU
Activation (complex weights learning)	tanh
Weight decay (regular parameters)	Log-uniform $[0.007, 0.2]$
Weight decay (complex weights parameters)	Log-uniform $[0.01, 0.4]$
Input dropout	Uniform $[0, 0.9]$
Layer dropout	Uniform $[0, 0.9]$
Patience (epochs)	1000
Max training epochs	1500
Optimiser	Adam

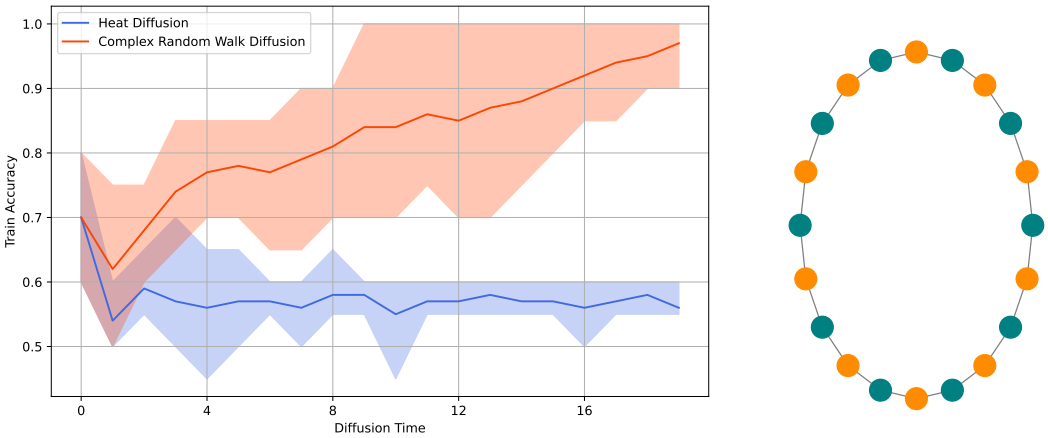

Figure 6: Training accuracy across diffusion layers (left) for heat diffusion and the learned complex random walk on a cycle bipartite graph with 2 classes (right): While the mean accuracy (solid line) for heat diffusion remains low throughout, the complex random walk surpasses 95 % as diffusion progresses. The shaded region (minimum–maximum range across 5 random seeds) further shows that heat diffusion never exceeds 70 %, whereas the complex random walk maintains over 90 % mean accuracy across all seeds.

We consider a bipartite cycle graph with 20 nodes divided into two partitions (Figure 6, right), and assign node features sampled from two overlapping isotropic Gaussian distributions. This setup ensures that the classes are not linearly separable at initialization. As shown by [9], heat diffusion fails to separate the classes in the diffusion limit. In contrast, Theorem 1 demonstrates that a complex random walk can achieve linear separation in its steady state. In this experiment, we study whether a suitable complex weight matrix can be learned directly from data in this simplified setting, using a vanilla random walk diffusion process, i.e., by setting $\mathbf{M}_1 = 1$, $\mathbf{M}_2 = \mathbf{I}_f$ and $\sigma = \mathrm{id}$ in Equation (6).

For heat diffusion, we first learn a transformation of the initial node features $\mathbf{X}(0)$, and then apply the diffusion process. For the complex random walk, we instead learn a complex-valued weight matrix $\mathbf{W}$ as a function of $\mathbf{X}(0)$, and subsequently perform the complex random walk. In both approaches, the model parameters are optimized to produce linearly separable features at the diffusion time limit.

Figure 6 (left) shows the classification results averaged over five random seeds. As expected, at diffusion time zero, a linear classifier fails to distinguish the classes. As diffusion progresses, heat diffusion continues to yield non-separable features, whereas the complex random walk consistently achieves over 90% mean accuracy across seeds. This result highlights the potential of learning an

effective complex-weighted graph structure that enables successful node classification at the diffusion limit.