Graph Neural Convection-Diffusion with Heterophily

Kai Zhao[1] , **Qiyu Kang**[1] , **Yang Song**[2] , **Rui She**[1] , **Sijie Wang**[1] and **Wee Peng Tay**[1]
[1]Nanyang Technological University
[2]C3.AI
{kai.zhao,qiyu.kang}@ntu.edu.sg, yang.song@c3.ai
{rui.she,wang1679,wptay}@ntu.edu.sg

Abstract

Graph neural networks (GNNs) have shown promising results across various graph learning tasks, but they often assume homophily, which can result in poor performance on heterophilic graphs. The connected nodes are likely to be from different classes or have dissimilar features on heterophilic graphs. In this paper, we propose a novel GNN that incorporates the principle of heterophily by modeling the flow of information on nodes using the convection-diffusion equation (CDE). This allows the CDE to take into account both the diffusion of information due to homophily and the "convection" of information due to heterophily. We conduct extensive experiments, which suggest that our framework can achieve competitive performance on node classification tasks for heterophilic graphs, compared to the state-of-the-art methods. The code is available at https://github.com/zknus/Graph-Diffusion-CDE.

1 Introduction

Graph Neural Networks (GNNs) have gained popularity in analyzing complex, structured data collected from various domains including social networks, citation networks, and molecular biology [Veličković *et al.*, 2018; Ji *et al.*, 2023; Kang *et al.*, 2023]. These datasets depict entities as nodes and relationships between entities as edges. Most pioneering GNN works like [Kipf and Welling, 2017; Veličković *et al.*, 2018; Hamilton *et al.*, 2017; Xu *et al.*, 2019; Kipf and Welling, 2016; Klicpera *et al.*, 2019; Lee *et al.*, 2022] assume the graph datasets have strong homophily, where linked nodes often belong to the same class or have similar features. These models fail to optimally utilize information in heterophilic graph datasets, where connected nodes are likely from distinct classes or possess dissimilar features. A study by [Sun *et al.*, 2022] concludes that even simple models that do not take into account the graph structure, such as multilayer perceptrons (MLPs), can surpass several existing GNN models on heterophilic datasets.

To tackle this challenge, several models like [Pei *et al.*, 2019; Zhu *et al.*, 2020; Chien *et al.*, 2021; Zhu *et al.*, 2021; Bo *et al.*, 2021; Yang *et al.*, 2021; Luan *et al.*, 2022; Bodnar *et al.*, 2022; Chanpuriya and Musco, 2022; Sun *et al.*, 2022; Li *et al.*, 2022; Ma *et al.*, 2021; Wang *et al.*, 2022; Du *et al.*, 2022] have been proposed. These works propose various techniques, such as aggregating higher-order neighborhoods, preserving high-frequency input signals, and adopting feature propagation methods, to improve the representation power of GNNs on heterophilic graph datasets.

Recently, works such as [Chamberlain *et al.*, 2021a; Chamberlain *et al.*, 2021b; Song *et al.*, 2022] have incorporated Neural Partial Differential Equations (PDEs) into GNNs, utilizing a variety of diffusion models for message passing on graphs. In this paper, our focus is on specific *diffusion-based* graph PDE models to address graph heterophily, as opposed to more general graph PDEs like those presented in the study by [Rusch *et al.*, 2022]. The latter models nodes in a graph as coupled oscillators and establishes a second-order Ordinary Differential Equation (ODE) for updating node features. In [Chamberlain *et al.*, 2021a], the authors model information propagation as a diffusion process of a substance from regions of higher to lower concentration, such as heat diffusion from a hot object to a cold surface. However, due to the nature of heat diffusion, the features of neighboring nodes tend to become increasingly smooth. In heterophilic graphs, where connected nodes are likely to have dissimilar features, this method of message passing may not be appropriate. The paper [Bodnar *et al.*, 2022] proposes more general sheaf diffusion operators to control the diffusion process and maintain the non-smoothness in heterophily graphs. Furthermore, the study [Zhu *et al.*, 2022] reveals a strong connection between heterophily and the robustness of GNNs. Recent studies [Kang *et al.*, 2021; Song *et al.*, 2021; Song *et al.*, 2022; Wang *et al.*, 2023a] have shown that specially designed neural ODEs may improve model robustness against adversarial attacks. Further research on diffusion models for heterophilic graphs is of particular interest.

In this paper, we propose a novel GNN inspired by the convection-diffusion process. Our proposed model explicitly incorporates the principle of heterophily in its design, which differentiates it from existing graph neural diffusion models. In a convection-diffusion heat transfer process in a fluid or gas, heat is not only transferred through diffusion due to heat concentration gradients but also through the movement of the fluid or gas itself. The velocity term in convection can be thought of as the movement of heat energy from one place to another due to the physical movement of a fluid

or gas. The curvature-preserving diffusion models such as Beltrami and mean-curvature [Chamberlain *et al.*, 2021b; Song *et al.*, 2022] are able to slow down the diffusion at the nodes where the feature difference between neighboring nodes is large, thereby preserving the non-smoothness and network robustness. Our model generalizes these curvature-preserving models by introducing a convection term in the diffusion equation that explicitly controls the propagation velocity at each node.

Main contributions. In this paper, our objective is to develop a general diffusion-based graph PDE framework that is suitable for heterophilic graph datasets. Our main contributions are summarized as follows:

1. Based on the convection-diffusion equation (CDE), we propose a graph neural convection-diffusion model that includes both a diffusion and convection term. The diffusion term aggregates information from homophilic neighbors, while the convection term allows controlled information propagation from heterophilic neighbors.
2. To manage the convection term in our CDE model, we introduce a learnable mechanism that adjusts the propagation rate at every node, allowing for better handling of heterophily.
3. We conduct extensive experiments on benchmark heterophilic datasets and compare with state-of-the-art baselines. Our experiments demonstrate that our model is competitive compared to the baselines, especially on large and more heterophilic datasets.

2 Related Work

In what follows, we briefly review the GNNs that are proposed to handle heterophilic graphs and recent advances in graph neural diffusion models.

2.1 Graph Neural Networks with Heterophily

Several works aim to adapt normal GNNs to heterophilic networks. H2GCN [Zhu *et al.*, 2020] proposes three effective designs to boost the performance of GNNs on heterophilic graphs, including the ego- and neighbor-embedding separation, higher-order neighborhood aggregation, and the combination of intermediate representations. Geom-GCN [Pei *et al.*, 2019] maps the graph to a suitable latent space where it can aggregate immediate neighborhoods and distant nodes to preserve the topology patterns of the graph. GPR-GNN [Chien *et al.*, 2021] assigns each step of feature propagation with a learnable weight, which can be negative or positive, to mitigate graph heterophily and over-smoothing issues. CPGNN [Zhu *et al.*, 2021] incorporates a compatibility matrix into GNNs to learn the likelihood of connections between nodes in different classes, which allows a GNN to capture both heterophily and homophily patterns in the graph. FAGCN [Bo *et al.*, 2021] integrates the low-frequency signals, high-frequency signals, and raw features adaptively through a self-gating mechanism to enhance the expressive power of GNNs on disassortative networks. ACM-GCN [Luan *et al.*, 2022] adaptively learns the local and node-wise information through aggregation, diversification, and identity channels. In this way, it can retain the low-frequency and high-frequency components of the input signal. Instead of using fixed feature propagation step in SGC [Wu *et al.*, 2019], ASGC [Chanpuriya and Musco, 2022] applies a learned polynomial of the normalized adjacency matrix with the input feature to fit a different filter for each feature. Our work is different from these previous works as we mainly focus on the use of graph PDEs to model the information propagation in heterophilic graphs.

2.2 Graph Neural Diffusion

In this subsection, we provide a brief overview of various graph neural diffusion models that have been proposed recently. In [Chamberlain *et al.*, 2021a], the authors modeled information propagation as a diffusion process of a substance from regions of higher to lower concentration. The Beltrami diffusion model was utilized in [Chamberlain *et al.*, 2021b; Song *et al.*, 2022] to enhance rewiring and improve the robustness of the graph. The above works do not specifically target the problem of graph heterophily. To address diffusion in heterophilic graphs, the paper [Bodnar *et al.*, 2022] introduces general sheaf diffusion operators to control the diffusion process and maintain non-smoothness in heterophilic graphs, leading to improved node classification performance. Inspired by the particle reaction-diffusion process, the ACMP[Wang *et al.*, 2023b] models the heterophilic graph by employing both repulsive and attractive force interactions as dual flow directions between nodes. Our approach differs from the aforementioned diffusion models in that we incorporate a convection term that is able to adapt to the complex connections in heterophilic graphs.

3 Preliminaries

In this section, we first introduce the convection-diffusion equation (CDE). We then review graph neural diffusion formulations that are used in our graph CDE model and will be later used in experiments on graph datasets.

3.1 Convection-Diffusion Equation

In [Chamberlain *et al.*, 2021a], the authors unified many popular GNN architectures into a single mathematical framework that is derived from the following (heat) diffusion equation. Let $x(u,t)$ denote a scalar-valued function on $\Omega \times [0,\infty)$, e.g., $x(u,t)$ can be the temperature at point u on the space Ω at time t. The heat diffusion [Grigoryan, 2009] process is formulated as

$$\frac{\partial x}{\partial t} = \mathrm{div}(D\nabla x),\ t > 0, \tag{1}$$

with some initial boundary condition that specifies the temperature distribution at $t = 0$, and div and ∇ are the divergence and concentration gradient operators, respectively. Here D is the thermal diffusivity for heat diffusion, which may vary at different positions in an inhomogeneous material and leads to position-dependent diffusion directions and speeds. In the GNN models [Song *et al.*, 2022; Chamberlain *et al.*, 2021a], the authors design different formulations D to model the information propagation between graph nodes that may depend on the node and its neighbors.

The CDE, on the other hand, describes physical phenomena such as heat transfer within a physical system, resulting from the combination of two processes: diffusion and convection. The CDE is given by

$$\frac{\partial x}{\partial t} = \operatorname{div}(D\nabla x) - \operatorname{div}(\mathbf{v}x) \tag{2}$$

where the first term represents the heat diffusion, same as in (1), and the second term $\operatorname{div}(\mathbf{v}x)$ in the equation represents the convection process. The velocity field $\mathbf{v}$ is a function of both time and space and describes how the quantity is moving. In oceanography, the convection-diffusion equation is used to model the transfer of heat in the ocean. The variable x would represent the concentration of heat in the ocean, and the velocity field $\mathbf{v}$ would represent the ocean currents and water flow as a function of both time and location.

3.2 Common Graph Neural Diffusion Models

Suppose $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ is a graph, where $\mathcal{V} = \{1, \dots, N\}$ is the set of N nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, and $w : \mathcal{E} \to \mathbb{R}^+$ the edge weight function. The features for the vertices at time t are represented by $\mathbf{X}(t) \in \mathbb{R}^{|\mathcal{V}| \times r}$ where r is the node feature dimension. The i-th row of $\mathbf{X}(t)$, denoted as $\mathbf{x}_i^\intercal(t)$, is the feature vector for vertex i at time t.

In the literature, several graph diffusion models [Chamberlain *et al.*, 2021a; Song *et al.*, 2022; Bodnar *et al.*, 2022] have been proposed, which are mostly derived from the heat equation (1) with different diffusivities D learned from the graph datasets. For example, inspired from the heat diffusion equation (1), GRAND [Chamberlain *et al.*, 2021a] defines the following graph version of the gradient ∇ and divergence div operators as

$$(\nabla \mathbf{X}(t))_{(i,j)} = \mathbf{x}_j(t) - \mathbf{x}_i(t),\ \forall\, (i,j) \in \mathcal{E} \tag{3}$$

for each edge $(i, j) \in \mathcal{E}$ in the graph, and

$$(\operatorname{div}(\mathscr{X}))_i = \sum_{j:(i,j)\in\mathcal{E}} \mathscr{X}_{ij}, \tag{4}$$

where $\mathscr{X} = \{\mathscr{X}_{ij}\}_{(i,j)\in\mathcal{E}}$ is the set of all features associated with edges and $\mathscr{X}_{ij} \in \mathbb{R}^r$ is the feature associated with edge (i, j). Furthermore, for graph learning, they implement the following dynamical system:

$$\begin{aligned}\frac{\partial \mathbf{X}(t)}{\partial t} &= \operatorname{div}(D(\mathbf{X}(t), t) \odot \nabla \mathbf{X}(t)) \\ &= (\mathbf{A}(\mathbf{X}(t)) - \mathbf{I})\mathbf{X}(t)\end{aligned} \tag{5}$$

where the initial condition is given by $\mathbf{X}(0) = \left(\mathbf{x}_0^\intercal(0), \cdots, \mathbf{x}_{|\mathcal{V}|}^\intercal(0)\right)^\intercal$, $\odot$ is the element-wise product and the diffusivity D is a $|\mathcal{E}| \times |\mathcal{E}|$ diagonal matrix with elements $\operatorname{diag}(a(\mathbf{x}_i(t), \mathbf{x}_j(t), t))$. Here, $a(\cdot)$ is a similarity function for vertex pairs. The diffusion equation can therefore be reformed as (5) with matrix $\mathbf{A}(\mathbf{X}(t)) = (a(\mathbf{x}_i(t), \mathbf{x}_j(t)))$, which is a learnable attention matrix used to describe the structure of the graph, and $\mathbf{I}$ being an identity matrix.

We refer the readers to the supplementary material for more details about other diffusion variants like the graph Beltrami flow [Song *et al.*, 2022], and graph sheaf diffusion [Bodnar *et al.*, 2022].

3.3 Graph Heterophily

In this section, we introduce the graph homophily measure, which is used in the experiments in Section 5 to quantify the homophily level of a graph dataset.

Definition 1 (Edge homophily ratio [Zhu *et al.*, 2020])**.** The edge homophily ratio is defined as:

$$h_{\text{edge}} = \frac{|\{(u,v) \,:\, (u,v) \in \mathcal{E} \wedge y_u = y_v\}|}{|\mathcal{E}|}. \tag{6}$$

This ratio is the proportion of edges in a graph that connects nodes with the same class label (i.e., intra-class edges). A smaller h_{edge} implies stronger heterophily.

The edge homophily ratio is sensitive to the number of classes and the balance of classes in a graph. To fairly compare the homophily ratio across different datasets with different numbers of classes and class balance, the paper [Platonov *et al.*, 2022] proposes the adjusted homophily ratio as follows.

Definition 2 (Adjusted homophily ratio [Platonov *et al.*, 2022])**.** Suppose there are C classes. The adjusted homophily ratio is given by

$$h_{\text{adj}} = \frac{h_{\text{edge}} - \sum_{k=1}^{C} D_k^2/(2|\mathcal{E}|)^2}{1 - \sum_{k=1}^{C} D_k^2/(2|\mathcal{E}|)^2} \tag{7}$$

where $D_k := \sum_{i:y_i=k} d(i)$, y_i is the label of node i, and $d(i)$ is the degree of node i.

4 Graph Neural Convection-Diffusion

We now consider graph neural convection-diffusion by making use of concepts from Section 3. In homophilic graphs, the edges between nodes tend to be between similar nodes, analogous to homogeneous atoms or molecules in a solid material. In such materials, these connections are relatively stable. The movement of atoms or molecules is relatively limited by the strength of the connections between them, and the heat transfer is mainly described using (1). Analogously, the information aggregation in homophilic graphs can be well-described by the graph heat diffusion equation [Chamberlain *et al.*, 2021c].

In contrast, heterophilic graphs have connections between dissimilar nodes, analogous to having different types of particles in a gas or fluid. These connections are less stable, and the movement of particles is relatively unrestricted. This makes heterophilic graphs well-suited to be regarded as soft materials, where the movement of particles is not limited by the strength of their connections. In the CDE (2), the velocity term (convection) is used to model the movement of the fluid or gas itself. For information propagation in heterophilic graphs, the convection term can help us to assign different information propagation velocities to each node due to the different connections.

The graph analogy of (2) can be formulated as

$$\frac{\partial}{\partial t}\mathbf{X}(t) = \operatorname{div}(D(\mathbf{X}(t), t) \odot \nabla \mathbf{X}(t)) + \operatorname{div}(\mathbf{V}(t) \circ \mathbf{X}(t)) \tag{8}$$

where the first diffusion term $\operatorname{div}(D(\mathbf{X}(t), t) \odot \nabla \mathbf{X}(t))$ could be given by (5) or other variants in [Song *et al.*, 2022; Bodnar

et al., 2022], $\mathbf{V}_{ij}(t) \in \mathbb{R}^r$ is the velocity vector associated with each edge (i, j) at time t, $\mathbf{V}(t) = \{\mathbf{V}_{ij}(t)\}_{(i,j)\in\mathcal{E}}$, and

$$(\mathrm{div}(\mathbf{V}(t) \circ \mathbf{X}(t)))_i \coloneqq \sum_{j:(i,j)\in\mathcal{E}} \mathbf{V}_{ij}(t) \odot \mathbf{x}_j(t) \quad (9)$$

for each node $i \in \mathcal{V}$.

4.1 $\mathbf{V}_{ij}$ for Heterophily

We propose the following simple yet effective velocity formulation:

$$\mathbf{V}_{ij} = \sigma\left(W(\mathbf{x}_j(t) - \mathbf{x}_i(t))\right), \quad (10)$$

where W is a learnable matrix and σ denotes an activation function. In (10), the velocity $\mathbf{V}_{ij}$ determines the direction of preferred information transport from node j to its neighboring node i. The velocity is determined by the difference $\mathbf{x}_j(t) - \mathbf{x}_i(t)$ of the feature representations of the neighboring nodes. Note that the velocity is not always in the same direction as the difference $\mathbf{x}_j(t) - \mathbf{x}_i(t)$ due to the learnable W and activation function. The use of the velocity term in the dot product with the state of the neighboring node in (9) allows for a more flexible and adaptive information flow for heterophilic graphs. This is because it takes into account the dissimilarity of neighboring nodes, which is a unique characteristic of heterophilic graphs. In contrast, traditional neural PDEs [Chamberlain *et al.*, 2021a; Song *et al.*, 2022] utilize only a single diffusion process, which does not consider the dissimilarity of neighboring nodes. The addition of the convection term in the CDE allows for a more accurate representation of the information flow in heterophilic graphs. For a more comprehensive understanding of our model, we direct readers to the supplementary material.

4.2 Model Details

We present our algorithm in Algorithm 1. Our approach is based the graph neural PDE defined in (8). This graph PDE is distinct from the diffusion-based models presented in previous works such as [Chamberlain *et al.*, 2021a; Song *et al.*, 2022; Bodnar *et al.*, 2022]. To solve the neural PDE, we follow the convention in these previous works and use the provided solver from [Chen *et al.*, 2018]. Our main contribution is the introduction of the second term in (8), which utilizes a simple yet effective velocity formulation (10)). The first term in (8) has various options as discussed in [Song *et al.*, 2022; Chamberlain *et al.*, 2021c].

Algorithm 1 Neural CDE Inference

Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, integration time T.
Output: Graph task results

1: Compress the sparse raw input node feature using an MLP and set them as the initial conditions $\{\mathbf{x}_i(0)\}_{i=1}^{|\mathcal{V}|}$.
2: Solve the neural PDE (8) using explicit or implicit solvers from[Chen *et al.*, 2018].
3: Return $\{\mathbf{x}_i(T)\}_{i=1}^{|\mathcal{V}|}$ which is the solution of (8) at time T.
4: Perform further graph tasks, e.g., node classification.

5 Experiments

5.1 Real-World Datasets

The paper [Pei *et al.*, 2019] evaluates the performance of their model on six heterophilic graph datasets: Squirrel, Chameleon, Actor, Texas, Cornell, and Wisconsin. These are widely-used heterophilic benchmarks, and we also use them to evaluate different models' node classification performance.

However, a recent paper [Platonov *et al.*, 2023] has pointed out a train-test data leakage problem in the Squirrel and Chameleon datasets collected by [Rozemberczki *et al.*, 2021], leading to questions on model performance on these two datasets. The datasets Cornell, Texas, and Wisconsin [1] do not have this data leakage issue but are relatively small and have significantly imbalanced classes, which can result in overfitting for diffusion models [Bodnar *et al.*, 2022]. We report the node classification results based on these three datasets in Table 2, where we use a fixed data splitting method as introduced in [Pei *et al.*, 2019]. The experiments where we apply random data splitting are provided in the Appendix, where we also include experiments using Squirrel, Chameleon, and Actor datasets.

To fairly evaluate the model performance on heterophilic graph datasets, we additionally include six new heterophilic datasets proposed in [Platonov *et al.*, 2023]. The proposed datasets come from different domains, such as English Wikipedia, the Amazon product co-purchasing network, and a question-answering platform. All of them have low homophily scores as verified in [Platonov *et al.*, 2023] and diverse structural properties. [Platonov *et al.*, 2023] evaluates a wide range of GNNs, both standard, and heterophily-specific, on these datasets and argues that the progress in graph learning under heterophily is limited since the standard baselines usually outperform heterophily-specific models on the datasets. For these heterophilic datasets, we follow the data splitting in [Platonov *et al.*, 2023], which is 50%, 25%, and 25% for training, validation, and testing. The data statistics are shown in Table 1. We use the ROC-AUC score as the evaluation metric on Minesweeper, Workers, and Questions datasets because they have binary classes. We include the synthetic regular graphs using the algorithm described in [Luan *et al.*, 2022] to control dataset heterophily for further demonstration.

5.2 Implementation and Baselines

We evaluate our neural CDE model on the node classification task against a variety of baseline models, including a graph-agnostic model, ResNet [He *et al.*, 2016], as well as standard GNN models such as GCN [Kipf and Welling, 2017], GAT [Veličković *et al.*, 2018], and GraphSAGE [Hamilton *et al.*, 2017]. Additionally, we compare our model to GNNs specifically designed for heterophilic graphs, including H2GCN [Zhu *et al.*, 2020], CPGNN [Zhu *et al.*, 2021], GPR-GNN [Chien *et al.*, 2021], GloGNN [Li *et al.*, 2022], FAGCN [Bo *et al.*, 2021], GBK-GNN [Du *et al.*, 2022], and ACM-GCN[Luan *et al.*, 2022]. We also include diffusion-based graph PDE models such as GRAND [Chamberlain *et al.*, 2021c], GraphBel [Song

[1] Available in http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/wwkb

Dataset	Nodes	Edges	Classes	Node Features	Edge homophily h_{edge}	Adjusted homophily h_{adj}
Roman-empire	22662	32927	18	300	0.05	-0.05
Wiki-cooc	10000	2243042	5	100	0.34	-0.03
Minesweeper	10000	39402	2	7	0.68	0.01
Questions	48921	153540	2	301	0.84	0.02
Workers	11758	519000	2	10	0.59	0.09
Amaon-ratings	24492	93050	5	300	0.38	0.14
Texas	183	295	5	1703	0.11	0.04
Cornel	183	280	5	1703	0.30	0.04
Wisconsin	251	466	5	1703	0.21	0.07

Table 1: Bechmark's statistics

Method h_{adj}	Roman-empire -0.05	Wiki-cooc -0.03	Minesweeper 0.01	Questions 0.02	Workers 0.09	Amazon-ratings 0.14	Texas 0.04	Cornell 0.04	Wisconsin 0.07
ResNet	65.71±0.44	89.36±0.71	50.95±1.12	70.10±0.75	73.08±1.28	45.70±0.69	80.81±4.75	81.89±6.40	85.29±3.31
GCN	70.51±0.75	91.01±0.69	89.47±0.69	75.97±1.23	83.14±1.11	47.77±0.69	55.14±5.16	60.54±5.30	51.76±3.06
GraphSAGE	85.80±0.69	93.60±0.31	93.53±0.50	76.52±0.93	82.34±0.19	**52.77±0.54**	82.43±6.14	75.95±5.01	81.18±5.56
GAT	81.02±0.46	92.44±0.80	92.10±0.67	**77.42±1.23**	**83.98±0.57**	47.95±0.53	52.16±6.63	61.89±5.05	49.41±4.09
H2GCN	68.09±0.29	89.24±0.32	89.95±0.38	66.66±1.84	81.76±0.68	41.36±0.47	84.86±7.23	82.70±5.28	87.65±4.98
CPGNN	63.78±0.50	84.84±0.66	71.27±1.14	67.09±2.63	72.44±0.80	44.36±0.35	75.68±5.12	70.27±5.12	76.47±6.16
GPR-GNN	73.37±0.68	91.90±0.78	81.79±0.98	73.41±1.24	70.59±1.15	43.90±0.48	81.35±5.32	78.11±6.55	82.55±6.23
GloGNN	63.85±0.49	88.49±0.45	62.53±1.34	67.15±1.92	73.90±0.95	37.28±0.66	84.32±4.15	83.51±4.26	87.06±3.53
FAGCN	70.53±0.99	91.88±0.37	89.69±0.60	77.04±1.56	81.87±0.94	46.32±2.50	82.43±6.89	79.19±9.79	82.94±7.95
GBK-GNN	75.87±0.43	97.81±0.32	83.56±0.84	72.98±1.05	78.06±0.91	43.47±0.51	81.08±4.88	74.27±2.18	84.21±4.33
ACM-GCN	68.35±1.95	87.48±1.06	90.47±0.57	OOM	78.25±0.78	38.51±3.38	**87.84±4.40**	85.14±6.07	88.43±3.22
GRAND	71.60±0.58	92.03±0.46	76.67±0.98	70.67±1.28	75.33±0.84	45.05±0.65	80.27±4.84	82.97±6.63	83.73±3.51
GraphBel	69.47±0.37	90.30±0.50	76.51±1.03	70.79±0.99	73.02±0.92	43.63±0.42	80.27±3.64	83.51±6.45	85.29±4.82
Diag-NSD	77.50±0.67	92.06±0.40	89.59±0.61	69.25±1.15	79.81±0.99	37.96±0.20	85.67±6.95	**86.49±7.35**	**88.63±2.75**
ACMP	71.27±0.59	92.68±0.37	76.15±1.12	71.18±1.03	75.03±0.92	44.76±0.52	86.20±3.0	85.40±7.0	86.10±4.0
CDE-GRAND	**91.64±0.28**	**97.99±0.38**	**95.50±5.23**	75.17±0.99	80.70±1.04	47.63±0.43	86.22±3.30	86.22±5.05	87.45±4.40
CDE-GraphBel	85.39±0.46	97.79±0.40	93.98±0.57	72.11±1.31	81.30±0.43	45.22±0.60	87.57±3.24	85.14±5.95	87.84±4.87

Table 2: Node classification results(%). The best and the second-best result for each criterion are highlighted in **bold** and underlined respectively. The results are accuracy on Wiki-cooc, Roman-empire, Amaon-ratings, Texas, Cornell and Wisconsin datasets and ROC-AUC score on Minesweeper, Workers, and Questions datasets. OOM refers to out-of-memory on NVIDIA RTX A5000 GPU.

et al., 2022], Diag-NSD [Bodnar *et al.*, 2022] and ACMP [Wang *et al.*, 2023b] as baselines.

We have implemented our CDE models using the algorithm outlined in Algorithm 1. In this model, the first diffusion term in (8) is specified by the graph diffusion formulation presented in previous works such as GRAND and GraphBel [Chamberlain *et al.*, 2021a; Song *et al.*, 2022]. We provide detailed descriptions of the specific diffusion formulations used in our implementation in Section 5.6. Specifically, we refer to our CDE model that utilizes the diffusion formulation in (12) as CDE-GRAND-GAT. For simplicity, we may use the abbreviation CDE-GRAND to refer to it in the rest of the paper. CDE-GraphBel refers to the CDE model with GraphBel being the first term. For all these datasets, we use the Adam optimizer [Kingma and Ba, 2014] with a learning rate of 0.01 and weight decay of 0.001. We also apply a dropout rate of 0.2 to prevent overfitting issues. We solve the neural PDE through the methods in [Chen *et al.*, 2018]. The integration time and step size are adjusted according to the performance of validation data. For the Texas, Cornell, and Wisconsin datasets, which have much smaller sizes, we conduct the hyperparameter search on each dataset. We report the best hyperparameters of each dataset in the supplementary material.

5.3 Node Classification Results on Real-World Datasets

From Table 2, we observe that if we augment diffusion models like GRAND and GraphBel with our proposed CDE convection term (9), we can significantly improve the diffusion model performance on the heterophilic graph datasets. For example, on the Roman-empire dataset, CDE-GRAND improves GRAND's performance from 71.60±0.58% to 91.64±0.28%. Similar improvements can be observed on *all of the nine heterophilic graph datasets*, demonstrating the effectiveness of our convection term in addressing the unique challenges of heterophilic graphs. This is because our velocity takes into account the dissimilarity of neighboring nodes in heterophilic graphs. For information propagation in heterophilic graphs, the convection term that includes the dot product between the neighboring node and the velocity helps us to assign flexible and adaptive information propagation flows to each node due to the different connections. We refer the readers to Section 5.6 for further demonstration.

Compared with all the baselines on heterophilic graph datasets, CDE-GRAND achieves the best performance on Wiki-cooc, Roman-empire, and Minesweeper. From Table 1, we observe that these datasets have the lowest adjusted ho-

mophily h_{adj}, indicating they are highly heterophilic. Our results suggest that CDE diffusion is particularly effective on more heterophilic datasets. On the Roman-empire dataset, CDE-GRAND outperforms the second-best model, GraphSAGE, by a significant margin of 5.84%. Notably, GraphSAGE is a standard GNN that is not specifically designed for heterophilic graphs, yet it performs better than all heterophily-specific baselines, highlighting the current limitations in graph learning under heterophily [Platonov *et al.*, 2023]. However, our CDE-GRAND model successfully surpasses GraphSAGE on this heterophilic dataset. A similar trend can be observed in the Minesweeper dataset where GraphSAGE achieves 93.53±0.50% and outperforms all the heterophily-specific baselines. Our CDE-GRAND model, however, can surpass GraphSAGE by a significant margin of 4.14%. On all other heterophilic graph datasets, CDE-GRAND performs comparably to state-of-the-art baselines, with the exception of Amazon-ratings, where all heterophily-specific models perform worse than GraphSAGE. This may be because the Amazon-ratings dataset has less heterophilic structure, as indicated by the largest adjusted homophily h_{adj} score presented in Table 1. As observed in [Platonov *et al.*, 2023] and Table 2, the modified GAT with residual connections outperforms all other heterophily-specific GNN models on the Questions, Workers, and Amazon-ratings datasets, hinting at unique geometries in these datasets that other models fail to capture effectively.

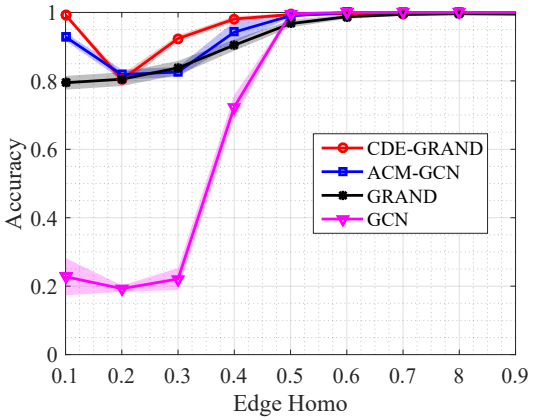

Figure 1: Accuracy versus edge homophily level. A node classification task based on the Cora dataset is used to compare different methods including CDE-GRAND, ACM-GCN, GRAND, and GCN.

5.4 Node Classification Results on Synthetic Datasets

We generate synthetic regular graphs using the algorithm described in [Luan *et al.*, 2022]. The edge homophily levels range from 0.1 to 0.9, with 10 graphs generated for each level. Each graph contains 5 classes, with 400 nodes in each class. For each node, we randomly generate 10 edges to connect to nodes within the same class and $[\frac{10}{H_{\text{edge}}(\mathcal{G})} - 10]$ edges to nodes in other classes. The features of each node are sampled from the corresponding class of the base dataset, which is the Cora dataset in Fig. 1. The nodes are then randomly split into train, validation, and test sets at a ratio of 60%:20%:20%. The node classification results are displayed in Fig. 1.

From Fig. 1, we observe that CDE-GRAND outperforms other methods on classification accuracy. Especially when the edge homophily level is less than 0.5, the CDE-GRAND has advantages obviously. This indicates that our method is more adaptive to the heterophily of graphs than the other methods including ACM-GCN, GRAND, and GCN.

Time	ODE Solver	Roman-empire	Minesweeper
1.0	Euler Solver	87.26±0.46	87.13±1.36
	RK4 Solver	91.55±0.66	93.05±0.48
2.0	Euler Solver	91.55±0.42	90.46±0.66
	RK4 Solver	91.10±1.16	94.64±2.32
3.0	Euler Solver	91.64±0.28	92.00±0.60
	RK4 Solver	90.82±2.18	94.35±3.62
4.0	Euler Solver	91.62±0.34	93.21±0.43
	RK4 Solver	91.00±1.31	97.67±0.22
5.0	Euler Solver	91.16±0.67	93.11±2.36
	RK4 Solver	90.26±2.16	97.10±2.02

Table 3: Node classification accuracy under different integration times of CDE. The ODE solver is used with step size $\tau = 1$. We fixed all other hyperparameters in the model except the integration time T. Unlike the traditional GNNs, the neural diffusion models do not have explicit layers but we may consider the steps taken to solve an ODE as the implicit layers. Hence, the number of implicit layers in neural diffusion models is T/τ for fixed step-size ODE solvers.

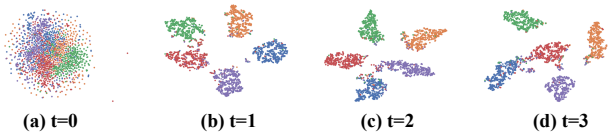

Figure 2: T-sne visualization of the feature representation from CDE-based neural diffusion models at different integration time t on the synthetic dataset described in Section 5.4 . From left to right: $t = 0$, $t = 1$, $t = 2$ and $t = 3$

5.5 Visulization

The visualization of the node features from our CDE model at different integration time t is shown in Fig. 2. From Fig. 2, we observe that the heterophilic graph dataset are classified into different classes by using our CDE models obviously.

5.6 Ablation Studies

Integration Time

We can see from Table 3 that as the integration time of our proposed CDE model increases, the node classification accuracy improves on the Roman-empire and Minesweeper datasets. However, there is a point at which the predicted classification results reach a saturation point. Therefore, it is important to consider the balance between accuracy and computation time when selecting the appropriate integration time for CDE, as a larger integration time results in increased computation time.

Method	Wiki-cooc	Roman-empire	Amazon-ratings	Minesweeper	Workers	Questions
GRAND-LAP	91.58±0.37	69.24±0.53	48.99±0.35	73.25±0.99	75.59±0.86	68.54±1.07
GRAND-GAT	92.03±0.46	71.60±0.58	45.05±0.65	76.67±0.98	75.33±0.84	70.67±1.28
GRAND-TRANS	91.86±0.27	71.18±0.56	45.20±0.52	75.40±1.36	75.13±0.65	69.14±0.97
GraphBel	90.30±0.50	69.47±0.37	43.63±0.42	76.51±1.03	73.02±0.92	70.79±0.99
CDE-GRAND-LAP	98.00±0.20	90.58±0.49	47.43±0.53	90.06±0.60	80.45±0.97	73.78±1.46
CDE-GRAND-GAT	97.99±0.38	91.64±0.28	47.63±0.43	95.50±5.23	80.70±1.04	75.17±0.99
CDE-GRAND-TRANS	98.04±0.35	91.55±0.23	46.91±0.87	91.38±1.92	81.58±0.98	72.92±1.54
CDE-GraphBel	97.79±0.40	85.39±0.46	45.22±0.60	90.79±0.48	81.30±0.43	72.11±1.31

Table 4: Node classification accuracy(%). We evaluate the performance of the vanilla diffusion variants and the inclusion of CDE convection.

PDE Solvers

There are various numerical methods available for solving nonlinear diffusion equations. In this work, we primarily utilize explicit solvers provided in [Chen *et al.*, 2018] for our model. These solvers can be divided into single-step and multi-step schemes, where the latter involves the use of multiple function evaluations at different time steps to compute the next iteration. This results in multi-hop neighboring information aggregation in multi-step explicit solvers. In this work, we use a multi-step solver named RK4 and a single-step explicit solver named Euler. As seen in Table 3, RK4 outperforms Euler, particularly on the Minesweeper dataset. This is likely due to the higher-order neighborhood aggregation used in RK4, which has been shown to be effective for heterophilic graphs [Zhu *et al.*, 2020]. However, the use of an RK4 solver typically results in higher computational complexity. In order to balance performance and computational efficiency, we choose to use the Euler solver for the experiments reported in Table 2, even though better performance may be achieved with the RK4 solver.

Method	Training Time (ms)	Inference Time (ms)
GRAND	9.55	4.66
GraphBel	12.55	6.69
CDE-GRAND	10.43	4.72
CDE-GraphBel	13.91	6.81

Table 5: Training time and inference time of models on the Cora dataset using Euler solver. Neural PDE models are based on the Euler solver with an integral time of 1.0 and a hidden dim of 64. The training and inference time refers to the average time for 100 rounds.

Diffusion Functions

In this section, we show the node classification performance under different diffusion functions compared to the vanilla diffusion variants defined in [Chamberlain *et al.*, 2021a; Song *et al.*, 2022]. The learnable attention matrix $\mathbf{A}(\mathbf{X}(t)) = (a(\mathbf{x}_i(t), \mathbf{x}_j(t)))$ in (5) can be formulated with three versions as used in [Chamberlain *et al.*, 2021a].

GRAND-TRANS uses the scaled dot product attention from the Transformer model [Vaswani *et al.*, 2017]:

$$a(\mathbf{x}_i, \mathbf{x}_j) = \mathrm{softmax}\left(\frac{(\mathbf{W}_K\mathbf{x}_i)^\top \mathbf{W}_Q\mathbf{x}_j}{d_k}\right), \quad (11)$$

where $\mathbf{W}_K$ and $\mathbf{W}_Q$ are learned matrices, and d_k is a hyperparameter determining the dimension of W_k.

GRAND-GAT uses the attention in GAT model [Veličković *et al.*, 2018]:

$$a(\mathbf{x}_i, \mathbf{x}_j) = \frac{\exp\left(\mathrm{LeakyReLU}\left(\mathbf{a}^\intercal\left[\mathbf{W}\mathbf{x}_i \| \mathbf{W}\mathbf{x}_j\right]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\mathrm{LeakyReLU}\left(\mathbf{a}^\intercal\left[\mathbf{W}\mathbf{x}_i \| W\mathbf{x}_k\right]\right)\right)}, \quad (12)$$

where $\mathbf{W}$ and $\mathbf{a}$ are learned and $\|$ is the concatenation operator.

GRAND-LAP models the linear diffusion process, where the $\mathbf{A}(t)$ a constant in the integral, i.e., $\mathbf{A}(\mathbf{X}(t)) = \mathbf{A}$ for a constant $\mathbf{A}$.

GraphBel is inspired by Beltrami diffusion [Sochen *et al.*, 1998]. We refer readers to [Song *et al.*, 2022] for its diffusion formulation.

The results in table 4 show that our CDE framework significantly improves the node classification performance of various diffusion-based GNN models on different datasets. The improvement is particularly noteworthy on the Roman-empire and Minesweeper datasets, with increases of over or near 15%. This highlights the effectiveness of incorporating the convection term into traditional diffusion-based GNNs.

Time Complexity

From Table 5, we observe that introducing CDE for the GNNs does not increase the training and inference time too much. Specifically, around 10% is increased for training time, while, over 1% is increased for inference time. As a result, it is acceptable to use CDE for GNNs to improve the performance w.r.t running complexity.

6 Conclusion

In order to handle the heterophilic graph efficiently, we introduce the convection-diffusion equation into GNNs to represent the information flow w.r.t nodes. Based on this framework, the homophilic information "diffusion" and the heterophilic information "convection" are combined for the graph representation. The extensive experiments show that our framework achieves competitive performance on node classification tasks for heterophilic graphs, compared to the state-of-the-art methods. Therefore, the convection-diffusion equation for the GNNs is beneficial to heterophilic graph learning.

Acknowledgments

This research is supported by A*STAR under its RIE2020 Advanced Manufacturing and Engineering (AME) Industry Alignment Fund – Pre Positioning (IAF-PP) (Grant No. A19D6a0053) and the National Research Foundation, Singapore and Infocomm Media Development Authority under its Future Communications Research and Development Programme. The computational work for this article was partially performed on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg).

Contribution Statement

This work was a collaborative effort of all authors. Kai Zhao and Qiyu Kang, who are recognized for their equal contributions, laid the groundwork by preparing the initial draft of this manuscript. Kai Zhao conceptualized the idea, crafted the design, wrote the experimental code, and performed the experiments. Qiyu Kang offered contributions by introducing the concept of the convection term in neural diffusion and formulating the theoretical framework. Yang Song and Wee Peng Tay guided the paper, helping to refine the research directions and objectives and enhancing the quality of our paper's writing. Rui She and Sijie Wang contributed by conducting the experiments and partaking in the writing process.

References

[Bo *et al.*, 2021] 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*, pages 3950–3957, 2021.

[Bodnar *et al.*, 2022] Cristian Bodnar, Francesco Di Giovanni, Benjamin Paul Chamberlain, Pietro Liò, and Michael M. Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in GNNs. In *Advances in Neural Information Processing Systems*, 2022.

[Chamberlain *et al.*, 2021a] Ben Chamberlain, James Rowbottom, Maria I Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. Grand: Graph neural diffusion. In *International Conference on Machine Learning*, pages 1407–1418. PMLR, 2021.

[Chamberlain *et al.*, 2021b] Benjamin Chamberlain, James Rowbottom, Davide Eynard, Francesco Di Giovanni, Xiaowen Dong, and Michael Bronstein. Beltrami flow and neural diffusion on graphs. *Advances in Neural Information Processing Systems*, 34:1594–1609, 2021.

[Chamberlain *et al.*, 2021c] Benjamin Paul Chamberlain, James Rowbottom, Maria Goronova, Stefan Webb, Emanuele Rossi, and Michael M Bronstein. Grand: Graph neural diffusion. In *Proc. Int. Conf. Mach. Learn.*, 2021.

[Chanpuriya and Musco, 2022] Sudhanshu Chanpuriya and Cameron Musco. Simplified graph convolution with heterophily. *arXiv preprint arXiv:2202.04139*, 2022.

[Chen *et al.*, 2018] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Proc. Advances Neural Inf. Process. Syst.*, 2018.

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

[Du *et al.*, 2022] Lun Du, Xiaozhou Shi, Qiang Fu, Xiaojun Ma, Hengyu Liu, Shi Han, and Dongmei Zhang. Gbk-gnn: Gated bi-kernel graph neural networks for modeling both homophily and heterophily. In *Proceedings of the ACM Web Conference 2022*, pages 1550–1558, 2022.

[Grigoryan, 2009] Alexander Grigoryan. *Heat kernel and analysis on manifolds*. American Mathematical Soc., Providence, 2009.

[Hamilton *et al.*, 2017] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proc. Advances Neural Inf. Process. Syst.*, 2017.

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

[Ji *et al.*, 2023] Feng Ji, See Hian Lee, Hanyang Meng, Kai Zhao, Jielong Yang, and Wee Peng Tay. Leveraging label non-uniformity for node classification in graph neural networks. In *Proc. International Conference on Machine Learning*, Haiwaii, USA, Jul. 2023.

[Kang *et al.*, 2021] Qiyu Kang, Yang Song, Qinxu Ding, and Wee Peng Tay. Stable neural ODE with Lyapunov-stable equilibrium points for defending against adversarial attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, virtual, Dec. 2021.

[Kang *et al.*, 2023] Qiyu Kang, Kai Zhao, Yang Song, Sijie Wang, and Wee Peng Tay. Node embedding from neural Hamiltonian orbits in graph neural networks. In *Proc. International Conference on Machine Learning*, Haiwaii, USA, Jul. 2023.

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

[Kipf and Welling, 2016] Thomas N. Kipf and Max Welling. Variational graph auto-encoders. In *Proc. Advances Neural Inf. Process. Syst. Workshop*, 2016.

[Kipf and Welling, 2017] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proc. Int. Conf. Learn. Representations*, pages 1–14, 2017.

[Klicpera *et al.*, 2019] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *Proc. Int. Conf. Learning Representations*, 2019.

[Lee *et al.*, 2022] See Hian Lee, Feng Ji, and Wee Peng Tay. SGAT: Simplicial graph attention network. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, Vienna, Austria, Jul. 2022.

[Li *et al.*, 2022] Xiang Li, Renyu Zhu, Yao Cheng, Caihua Shan, Siqiang Luo, Dongsheng Li, and Weining Qian. Finding global homophily in graph neural networks when meeting heterophily. *arXiv preprint arXiv:2205.07308*, 2022.

[Luan *et al.*, 2022] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Revisiting heterophily for graph neural networks. *arXiv preprint arXiv:2210.07606*, 2022.

[Ma *et al.*, 2021] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. Is homophily a necessity for graph neural networks? In *International Conference on Learning Representations*, 2021.

[Pei *et al.*, 2019] 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*, 2019.

[Platonov *et al.*, 2022] Oleg Platonov, Denis Kuznedelev, Artem Babenko, and Liudmila Prokhorenkova. Characterizing graph datasets for node classification: Beyond homophily-heterophily dichotomy. *arXiv preprint arXiv:2209.06177*, 2022.

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

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

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

[Sochen *et al.*, 1998] N. Sochen, R. Kimmel, and R. Malladi. A general framework for low level vision. *IEEE Trans. Image Process.*, 7(3):310–318, 1998.

[Song *et al.*, 2021] Yang Song, Qiyu Kang, and Wee Peng Tay. Error-correcting output codes with ensemble diversity for robust learning in neural networks. In *Proc. AAAI Conference on Artificial Intelligence*, virtual, Feb. 2021.

[Song *et al.*, 2022] Yang Song, Qiyu Kang, Sijie Wang, Kai Zhao, and Wee Peng Tay. On the robustness of graph neural diffusion to topology perturbations. In *Advances in Neural Information Processing Systems (NeurIPS)*, New Orleans, USA, Nov. 2022.

[Sun *et al.*, 2022] Yifei Sun, Haoran Deng, Yang Yang, Chunping Wang, Jiarong Xu, Renhong Huang, Linfeng Cao, Yang Wang, and Lei Chen. Beyond homophily: Structure-aware path aggregation graph neural network. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, 2022.

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

[Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *Proc. Int. Conf. Learn. Representations*, pages 1–12, 2018.

[Wang *et al.*, 2022] Tao Wang, Di Jin, Rui Wang, Dongxiao He, and Yuxiao Huang. Powerful graph convolutional networks with adaptive propagation mechanism for homophily and heterophily. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4210–4218, 2022.

[Wang *et al.*, 2023a] Sijie Wang, Qiyu Kang, Rui She, Wee Peng Tay, Andreas Hartmannsgruber, and Diego Navarro Navarro. RobustLoc: Robust camera pose regression in challenging driving environments. In *Proc. AAAI Conference on Artificial Intelligence*, Washington, DC, Feb. 2023.

[Wang *et al.*, 2023b] Yuelin Wang, Kai Yi, Xinliang Liu, Yu Guang Wang, and Shi Jin. ACMP: Allen-cahn message passing with attractive and repulsive forces for graph neural networks. In *The Eleventh International Conference on Learning Representations*, 2023.

[Wu *et al.*, 2019] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

[Xu *et al.*, 2019] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *Proc. Int. Conf. Learn. Representations*, 2019.

[Yang *et al.*, 2021] Liang Yang, Mengzhe Li, Liyang Liu, Chuan Wang, Xiaochun Cao, Yuanfang Guo, et al. Diverse message passing for attribute with heterophily. *Advances in Neural Information Processing Systems*, 34:4751–4763, 2021.

[Zhu *et al.*, 2020] 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.

[Zhu *et al.*, 2021] Jiong Zhu, Ryan A Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K Ahmed, and Danai Koutra. Graph neural networks with heterophily. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 11168–11176, 2021.

[Zhu *et al.*, 2022] Jiong Zhu, Junchen Jin, Donald Loveland, Michael T Schaub, and Danai Koutra. How does heterophily impact the robustness of graph neural networks? theoretical connections and practical implications. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2637–2647, 2022.