Graph Convolutional Networks Guided by Explicitly Estimated Homophily and Heterophily Degree

Rui Zhang and Xin Li

Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200062, China
* Correspondence: xinli@sei.ecnu.edu.cn

Abstract: Graph convolutional networks (GCNs) have been successfully applied to learning tasks on graph-structured data. However, most traditional GCNs based on graph convolutions assume homophily in graphs, which leads to a poor performance when dealing with heterophilic graphs. Although many novel methods have recently been proposed to deal with heterophily, the effect of homophily and heterophily on classifying node pairs is not clearly separated in existing approaches and inevitably influences each other. To deal with various types of graphs more accurately, in this work we propose a new GCN-based model that leverages the explicitly estimated homophily and heterophily degree between node pairs and adaptively guides the propagation and aggregation of signed messages. We also design a pre-training process to learn homophily and heterophily degree from both original node attributes that are graph-agnostic and the localized graph structure information by using Deepwalk that reflects graph topology. Extensive experiments on eight real-world benchmarks demonstrate that the new approach achieves state-of-the-art results on three homophilic graph datasets and outperforms baselines on five heterophilic graph datasets.

Keywords: graph convolutional networks; heterophilic graphs; node classification

1. Introduction

Graph convolutional networks (GCNs) [1], and their variants, such as GAT [2], GraphSage [3] and GIN [4], have made great progress in network analysis tasks [5] and have been widely-used in many fields, such as natural language processing [6, 7], recommendation systems [8, 9], bioinformatics [10] and so on.

The design of traditional GCN-based methods, such as GAT, GraphSAGE and GIN, is based on the homophily assumption, which means that the connected nodes in a graph more likely belong to the same category or have similar features [1, 11]. The traditional GCN-based methods reflect the assumption of a message-passing mechanism [5, 12], where connected nodes propagate features to each other as messages, and each node aggregates the node features of neighbors and then combines the aggregated message and ego features to update the node representation [1, 2, 4, 5]. For each node, the aggregation and update process can be viewed as a weighted sum of ego features and all neighbor’s features, and the neighbor’s features are always assigned positive coefficients when calculating the weighted sum. From the perspective of spectral approaches [5, 13, 14], the results in [15, 16] show that the typical message-passing mechanism of the GCNs above actually acts as a low-pass filter, which filters out high-pass signals and retains the low-pass signals. It causes the representations of connected nodes to tend to be similar and easier to be classified as the same category. Under this assumption, the traditional GCN-based methods have shown good predictive accuracy of node classification tasks on homophilic graphs.

However, there are many graphs in the real-world where a large number of neighbouring nodes belong to different categories, also known as heterophilic graphs. For example, on e-commerce websites, a fraudster is more inclined to interact with ordinary users than with another fraudster [17]. In dating networks, users tend to establish relationships with
the opposite gender. Studies have shown that many traditional GCN models are ineffective on heterophilic graphs, especially for semi-supervised node classification tasks [18,19]. As shown in Figure 1a, consider updating the node feature \(h_1\) of \(v_1\) that has two neighbours \(v_2\) and \(v_3\). By applying the traditional message-passing method, the updated representation of node \(v_1\) is obtained by adding the node features of \(v_2\) and \(v_3\) to its self-representation, respectively. After updating, the node feature of \(v_1\) becomes closer to that of both \(v_2\) and \(v_3\), which makes it difficult to distinguish between nodes \(v_1\) and \(v_2\) that belong to different categories.

\[
\begin{align*}
\text{Aggregation} & \quad \text{Update} \\
\end{align*}
\]

(a) The traditional message-passing process.

Figure 1. An illustration of two message-passing processes for updating the node feature of \(v_1\), where \(h_1, h_2, h_3\) are mutually perpendicular three-dimensional feature vectors of nodes \(v_1, v_2, v_3\), and the node color indicates their category, and \(h'_1\) is the updated node feature of \(v_1\) after aggregating \(h_2, h_3\) and combining with \(h_1\).

Recently, many novel GCN-based methods have been proposed to deal with heterophily. Notably, FAGCN [20] takes the perspective of graph signal processing [21] and explores both low-pass filters and high-pass filters in the model design. By connecting to the spatial domain that the process of low-pass filtering represents, the sum of node features with neighbours and the process of high-pass filtering represent the difference of node features between neighbouring nodes; it uses a self-gating mechanism to learn the signed coefficients of the messages. HOG-GCN [22] and BM-GCN [23] explore the estimated homophily degree between node pairs to guide the propagation of positive messages but do not consider how to guide the propagation of negative messages, which is often important in heterophilic graphs. Consider again the example in Figure 1b, if we introduce signed messages during the message-passing process, the updated representation of node \(v_1\) is closer to that of its homophilic neighbor \(v_3\) and away from that of its heterophilic neighbor \(v_2\) as expected. We also observe that the above methods have a common limitation in that the effects of homophily and heterophily on classifying node pairs are not clearly separated in their learning approaches and inevitably influence each other.

In this paper, we aim to design a new model that works well on either homophilic or heterophilic graphs for semi-supervised node classification tasks. Comparing the existing approaches and inspired by the above models, especially FAGCN and HOG-GCN, we would like to leverage explicitly the estimated homophily and heterophily degree between node pairs to adaptively guide the propagation and aggregation of signed messages, respectively. In particular, positive messages are aggregated from homophilic neighbors, and negative messages are aggregated from heterophilic neighbors. To overcome the limitation that the categories of most nodes are unknown in semi-supervised node classification tasks, we pre-train estimators to calculate both the homophily and the heterophily degree of node pairs, which reflect the probability that neighbouring nodes belong to the same category and different categories, respectively. We also propose to learn estimators from both original node attributes that are graph-agnostic and the localized graph structure information by using Deepwalk [24] that reflects graph topology. These two estimators are jointly used in our learning process. We adopt a two-stage training method to train our proposed model, in which the estimators are pre-trained in the first stage, and the GCN
module together with the pre-trained estimators are jointly trained in the second stage. That is, the estimators are fine-tuned when training the GCN module.

To summarize, the main contributions of this paper are as follows:

1. We propose an improved GCN-based message-passing mechanism which explores the explicitly estimated homophily and heterophily degree to adaptively guide the passing of positive and negative messages, respectively. The improved mechanism makes the learned representations of nodes which belong to different categories easier to be distinguished in both homophilic and heterophilic graphs.

2. We propose extracting localized structure information in graphs by using Deepwalk which is explored to pre-train an estimator. The estimator as well as the pre-trained estimator learned from original node attributes are jointly used and give us an approach to estimate appropriate homophily and heterophily degree of node pairs.

3. We then design a new Graph Convolutional Network Guided by Explicitly-Estimated Homophily and Heterophily Degree called GCN-EHHD based on the above components. Extensive experiments on eight real-world benchmarks demonstrate the effectiveness of our model. Specifically, GCN-EHHD achieves state-of-the-art results on three homophilic graph datasets and outperforms baselines on five heterophilic graph datasets.

The rest of the paper is organized as follows. In Section 2, we briefly introduce the related work on tackling semi-supervised node classification tasks on heterophilic graphs. In Section 3, we define the semi-supervised node classification tasks and describe the details of our proposed GCN-EHHD. In Section 4, we demonstrate the effectiveness of our proposed approach by comprehensive experiments and an ablation study on real-world datasets. Finally, we conclude with future work in Section 5.

2. Related Work

There are two major GCN-based approaches for tackling learning problems on heterophilic graphs. One is to mine more supplemental nodes when updating a node feature in the message-passing process, which act as those directly-connected neighbors for propagating messages to the node and participate in aggregation and updating of node representation. Geom-GCN [18] maps each node to the latent space and takes the nodes whose distance from the node is less than the setting value in the latent space as the supplemental nodes. NL-GNN [25] utilizes an attention-guided sorting method to choose the supplemental nodes. It defines a learnable calibration vector and adopts an attention mechanism to calculate the attention scores of each node representation and the calibration vector. All nodes are sorted according to their attention scores. For each node, the other nodes whose attention scores are similar are chosen as the supplemental nodes. In addition to the above methods, there are some works that choose the higher-order neighbors in the graph as supplemental nodes. They are based on the observation that for nodes in heterophilic graphs, the first-order neighbors are often heterophilic, but the higher-order neighbors are likely to be homophilic [26]. MixHop [19] chooses two-hop neighbors as the supplemental nodes for message passing. In the layers of MixHop, the messages from different hops neighbors are linearly transformed by different weights and then concatenated. H2GCN [26] chooses higher-order neighbors as the supplemental nodes for message passing. It cancels the linear transformation and concatenates the messages aggregated from neighbors of different hops in the hidden layer, and the aggregated messages at different hidden layers are concatenated and linearly transformed in the output layer. Considering that the number of two-hop neighbors will grow exponentially with the growth of network scale, UGCN [27] limits the two-hop neighbors that have at least two different paths to the node as the supplemental nodes. Apart from the two-hop neighbors, UGCN also utilizes the cosine similarity to choose $k$-nearest neighbor neighbors as the supplemental nodes. However, there is a shortcoming for these methods in that mixing the messages obtained from connected nodes and supplemental nodes will damage the graph structure information which is often important in graph learning tasks.
The other approach is to design a more reasonable message-passing mechanism. FAGCN [20] demonstrates the importance of high-pass signals on heterophilic graphs through experiments and illustrates that high-pass signals can be retained by propagating negative messages between the connected nodes. It uses a self-gating attention mechanism to learn the signed coefficients of neighbors for propagation, and the coefficients represent the proportions of low-pass signals and high-pass signals. However, the effects of homophily and heterophily on the learning of signed coefficients are not clearly distinguished and treated in a mixed and implicit way. GGCN [28] uses the cosine similarity function to calculate the signed coefficients of neighbors for propagation between connected nodes. CPGNN [29] defines a parameterized and learnable compatibility matrix which is used to model the probability of node pairs that belong to each category. It also adopts a pre-trained estimator to calculate the prior belief of class labels for nodes and propagates the prior belief by the compatibility matrix. HOG-GCN [22] combines the estimated homophily degree and the learnable homophily degree of node pairs to guide the positive message passing between nodes and their multi-hop neighbors. Since HOG-GCN models the homophily degree between node pairs by learnable parameters, the parameter quantity of HOG-GCN can be quite big for a large graph. BM-GCN [23] adopts the estimated soft labels to calculate a block similarity matrix and combines the estimated soft labels and block similarity matrix to guide the positive message passing. HOG-GCN and BM-GCN do not consider the passing of signed messages, which is often important in heterophilic graphs. In a word, the existing works do not consider utilizing the explicitly estimated homophily and heterophily degree to guide the passing of signed messages.

3. The Proposed Method

3.1. Problem Definition and Overview of the Proposed Method

Consider a graph \( G = (\mathcal{V}, \mathcal{E}, \mathbf{A}) \), where \( \mathcal{V} = \{v_1, v_2, \ldots, v_N\} \) represents a set of \( N \) nodes, \( \mathcal{E} = \{e_{ij}\} \subseteq \mathcal{V} \times \mathcal{V} \) represents a set of edges, and \( \mathbf{A} \in \{0, 1\}^{N \times N} \) is an adjacency matrix which can describe the topology of the graph \( G \). \( \mathbf{A}[i,j] = 1 \) if there is an edge between \( v_i \) and \( v_j \), or 0 otherwise. In this paper, we focus on the semi-supervised node classification task on graphs. For each graph, there is a feature matrix \( \mathbf{X} \). Let \( x_i \) be the \( i \)-th column of \( \mathbf{X} \), which represents the feature vector associated with node \( v_i \). Each node belongs to one of \( C \) classes. There is a set of nodes \( \mathcal{V}_L \) (\(|\mathcal{V}_L| << N\)), and each node of \( \mathcal{V}_L \) is assigned a label \( y_i \in \{1, 2, \ldots, C\} \). The labels of all nodes in set \( \mathcal{V}_L \) form a set \( \mathcal{Y}_L \). The task is to learn a function \( f : (G, \mathbf{X}, \mathcal{Y}_L) \rightarrow \mathcal{Y}_U \) and predict the labels \( \mathcal{Y}_U \) of unlabeled nodes.

There are two major components in GCN-EHHD. One is the MLP module for pre-training estimators. Here we choose the multi-layer perceptron (MLP) as the estimator to explicitly estimate the homophily and heterophily degree between neighbouring nodes. The other is the GCN module that leverages the estimated homophily and heterophily degree to adaptively guide signed-message passing. In the improved message-passing mechanism, when updating the representation of each node in the graph, we use the homophily degree to guide the passing of a positive message and the heterophily degree to guide the passing of a negative message, respectively, and these signed messages are aggregated with the node self-representation. We also adopt a two-stage training process to train GCN-EHHD, where the MLP module is pre-trained first for obtaining estimators and then is jointly trained with the GCN module in the second step. An overview of GCN-EHHD is shown in Figure 2.
3.2. The Pre-Training Module for Estimators

In order to make full use of the graph information, we pre-train two MLPs by learning from two MLPs which use the node attribute information following [22] and localized structure information independently. We choose Deepwalk [24] to mine localized structure information. Deepwalk [24] applies a group of random walk generators to generate node sequences from a graph. For each node \( v_i \) in the graph, the random walk generator takes \( v_i \) as the first node to visit, samples uniformly from neighbors of \( v_i \) and then designates the sampled neighbor as the next node to visit. The generator repeats the sampling process until the maximum length \( T \) of the sequence is reached and the sampled nodes form a sequence in order. Then, Skip-gram [30] is used to generate embeddings \( x_i^E \) for each node \( v_i \) from random walk sequences. For each node in a random walk sequence, Skip-gram maximizes the probability of the neighbors that co-occur with the node in a window. It iterates over the random walk set and uses gradient descent to train the embedding of nodes. Let \( X^E \) be the learned matrix generated by Deepwalk in graph \( G \), and let \( x_i^E \) be the \( i \)-th column of \( X^E \). \( X^E \) will be used to train an estimator.

Then, we define two MLPs, called MLP\(_1\) and MLP\(_2\), respectively. The \( l \)-th layers of the two MLPs can be defined as:

\[
\begin{align*}
    p_i^{(l)} &= \sigma\left(W_1^{(l)} \cdot p_i^{(l-1)} + b_1^{(l)}\right) \quad (1) \\
    q_i^{(l)} &= \sigma\left(W_2^{(l)} \cdot q_i^{(l-1)} + b_2^{(l)}\right) \quad (2)
\end{align*}
\]

where \( p_i^{(0)} = x_i \) and \( q_i^{(0)} = x_i^E \) are the original feature vector and embedding vector of node \( v_i \) respectively, \( W_1^{(l)} \) and \( W_2^{(l)} \) are the learnable weight matrices, \( b_1^{(l)} \) and \( b_2^{(l)} \) are the learnable bias vectors, and \( \sigma \) is an activation function.

Through MLP estimators, we can obtain the soft labels of all the nodes:

\[
\begin{align*}
    s_i &= \text{softmax}(p_i^{(L)}) \quad (3) \\
    t_i &= \text{softmax}(q_i^{(L)}) \quad (4)
\end{align*}
\]

where \( p_i^{(L)}, q_i^{(L)} \) are the outputs of the last layers of MLP\(_1\) and MLP\(_2\) for node \( v_i \), respectively, and \( s_i, t_i \in \mathbb{R}^C \) denote the probabilities that node \( v_i \) belongs to class \( j \) as estimated by MLP\(_1\) and MLP\(_2\), respectively.
We pre-train estimators by the nodes with known labels and obtain the optimal parameters of MLP$_1$ and MLP$_2$ by minimizing the following two loss functions:

$$L_{mlp1} = \sum_{v_i \in V_l} J(s_i, y_i)$$

$$L_{mlp2} = \sum_{v_i \in V_l} J(t_i, y_i)$$

where $J$ represents the cross entropy.

After the above pre-training process, we obtain the estimators MLP$_1$ and MLP$_2$, and we will use them to estimate the homophily and heterophily degree of the neighboring nodes.

### 3.3. The GCN Module Guided by Explicitly Estimated Homophily and Heterophily Degree

For any directly connected nodes $v_i$ and $v_j$ in the graph, we use pre-trained estimators to calculate the homophily degree of these two nodes, which is the estimated probability that two nodes belong to the same category:

$$u_{i,j}^{homo} = s_i^\top s_j$$

$$z_{i,j}^{homo} = t_i^\top t_j$$

We take a weighted sum of the two estimated homophily degrees:

$$d_{i,j}^{homo} = \theta \cdot u_{i,j}^{homo} + (1 - \theta) \cdot z_{i,j}^{homo}$$

The hyperparameter $\theta$ controls the proportion of original node attribute information and localized structure information used in estimation. Meanwhile, we can obtain the estimated heterophily degree between node pairs, which is the estimated probability that two nodes belong to different categories:

$$d_{i,j}^{hete} = 1 - d_{i,j}^{homo}$$

Following the settings in BM-GCN [23], for a graph $G$, we add multiple self-loops to it:

$$\tilde{A} = A + \delta I$$

where $I \in \mathbb{R}^{n \times n}$ is the identity matrix, and $\delta$ is the self-loop parameter which is used to control the proportion contributed by the node itself when calculating the weights of messages. We use $\mathcal{N}(v_i)$ to denote the set of neighbors of node $v_i$ in the graph. We use a softmax operation to normalize the homophily and heterophily degrees between node $v_i$ and all its neighbors to obtain the weights of positive and negative messages:

$$w_{i,j}^{pos} = \frac{\exp(d_{i,j}^{homo})}{\sum_{v_p \in \mathcal{N}(v_i)} \exp(d_{i,p}^{homo})}$$

$$w_{i,j}^{neg} = \frac{\exp(d_{i,j}^{hete})}{\sum_{v_p \in \mathcal{N}(v_i)} \exp(d_{i,p}^{hete})}$$

Then, we use the above weights to guide the message passing in the graph convolutional layer. The $l$-th graph convolutional layer in GCN-EHHD can be defined by:

$$m_i^{(l)} = \alpha \sum_{v_j \in \mathcal{N}(v_i)} w_{i,j}^{pos} w_{i,j}^{pos} h_j^{(l-1)} - (1 - \alpha) \sum_{v_j \in \mathcal{N}(v_i)} w_{i,j}^{neg} w_{i,j}^{neg} h_j^{(l-1)}$$
\textbf{3.4. Optimization Objective}

Similar to the loss function in (5), we can calculate the loss of the GCN module by

\[ L_{gcn} = \sum_{v_i \in V} J(r_i, y_i) \] (17)

When training the GCN module of GCN-EHHD, we fine-tune the estimators at the same time. That is, we add the loss of MLP\(_1\) described in Equation (5), the loss of MLP\(_2\) described in Equation (6) and the loss described in Equation (17) as the final loss:

\[ L_{gcn-EHHD} = L_{mlp1} + L_{mlp2} + L_{gcn} \] (18)

We will jointly train the MLP module and the GCN module by minimizing the above loss function.

\textbf{4. Experiments}

\textbf{4.1. Experimental Settings}

\textbf{Datasets.} We chose eight widely-adopted datasets. Following [26], we used the homophily ratio \( h = \frac{|\{(u,v) \in E : y_u = y_v\}|}{|E|} \) to describe the homophily level for a graph. We chose three homophilic graph datasets: Cora, Citeseer and Pubmed [31,32], and five heterophilic graph datasets: Texas, Wisconsin, Squirrel, Chameleon and Cornell [18]. The statistics of these datasets are listed in Table 1, where \( h \) represents the homophily ratio of a graph, and \( d_{avg} = \frac{|E|}{|V|} \) represents the average degree of nodes in a graph.

\textbf{Baseline Methods.} We compared our proposed model with the following baselines: (1) MLP which only uses original node attributes information; (2) Deepwalk [24] which only uses graph topology information; (3) GCN-based models under the assumption of homophily: GCN [1] and GAT [2]; and (4) GCN-based models that deal with heterophilic graphs: MixHop [19], Geom-GCN [18], FAGCN [20], H2GCN [26], HOG-GCN [22] and BM-GCN [23].

\textbf{Implementation.} The baselines MLP and Deepwalk were the same methods with the MLP\(_1\) and MLP\(_2\) in GCN-EHHD, respectively, and the parameter settings and training procedures of baselines MLP and Deepwalk were the same as those of the pre-trained MLP\(_1\) and MLP\(_2\) in GCN-EHHD, respectively. For GCN and GAT, the dimension of hidden layers was set as 64, the number of layers was set as 2, the Adam optimizer [33] was used as optimizer, the learning rate was set as 0.001, and the weight decay was set as 5E-4. The number of attention heads in GAT was set as five. The number of training epochs of GCN and GAT was set as 1600 with the early stopping strategy of 100 patience epochs. For other
baselines, the optimal parameters and the training procedures were taken as reported in their original papers. We implemented our proposed GCN-EHHD based on PyTorch [34]. We utilized the Adam optimizer [33] to train the GCN-EHHD. In the process of generating embedding $X^E$ using Deepwalk, the number of random walk generators was set as 80, the maximum length of random walk was set as 10, and the window size was set as 5. For each dataset, the embedding dimension of $X^E$ was the same as the original features $X$. For both the MLP module and the GCN module, the number of layers was searched in $\{2, 3\}$, and the dimension of hidden layer was searched in $\{32, 64\}$. The learning rate was set in $\{0.001, 0.005\}$, and the weight decay was set in $\{0.02, 5E-4, 5E-5\}$. For different datasets, The parameter $\delta$ in Equation (11) for controlling the contribution of self-loops was set in $\{0, 1, 2\}$, the hyperparameter $\alpha$ in Equation (14) was set in $\{0, 0.4, 0.5\}$, and the hyperparameter $\theta$ in Equation (9) was set in $\{0.1, 0.4, 0.6, 1\}$. The impact of the parameter $\alpha$ and $\theta$ is analyzed in Section 4.4. To train GCN-EHHD, the number of pre-training epochs was set as 400, the number of jointly training epochs was set as 1600 with the early stopping strategy with 100 patience epochs following the training procedure of [23].

| Datasets | Nodes | Edges | Features | Classes | $d_{avg}$ | $h$ |
|----------|-------|-------|----------|--------|----------|-----|
| Texas    | 183   | 309   | 1703     | 5      | 1.68     | 0.09|
| Wisconsin| 251   | 499   | 1703     | 5      | 1.98     | 0.19|
| Squirrel | 5201  | 198,493 | 2089    | 5      | 38.16    | 0.22|
| Chameleon| 2277  | 31,421 | 2325    | 5      | 13.79    | 0.23|
| Cornell  | 183   | 309   | 1703     | 5      | 1.68     | 0.30|
| Citeseer | 3327  | 4732  | 3703     | 6      | 1.42     | 0.74|
| Pubmed   | 19,717| 44,338| 500      | 3      | 2.24     | 0.80|
| Cora     | 2708  | 5429  | 1433     | 7      | 2.00     | 0.81|

4.2. Experimental Results

For all eight datasets, we adopted the test approach of [18] and randomly split the data ten times according to the division ratio 48%/32%/20% to generate the training/validation/test sets. We used the mean accuracy on the test set of the ten splits as the result. We report the experimental results of three homophilic graphs in Table 2, and the results of five heterophilic graphs in Table 3.

Table 2. Node classification accuracy on homophilic graphs (in % ± Standard Error). The best results are in bold, and the second-best results are underlined.

| Model   | Cora       | Citeseer   | Pubmed     |
|---------|------------|------------|------------|
| MLP     | 72.55 ± 1.66 | 70.80 ± 2.82 | 85.92 ± 0.67 |
| Deepwalk| 80.96 ± 1.46 | 61.20 ± 2.12 | 82.02 ± 0.46 |
| GCN     | 86.53 ± 1.51 | 75.00 ± 2.12 | 87.31 ± 0.73 |
| GAT     | 86.70 ± 1.28 | 74.95 ± 2.46 | 86.42 ± 0.81 |
| MixHop  | 86.65 ± 1.62 | 74.12 ± 1.64 | 89.09 ± 0.60 |
| Geom-GCN| 85.05 ± 1.50 | 75.16 ± 1.57 | 88.07 ± 0.56 |
| FAGCN   | 87.10 ± 1.81 | 75.77 ± 1.43 | 88.67 ± 0.67 |
| H2GCN   | 85.39 ± 1.14 | 73.72 ± 1.95 | 84.25 ± 0.61 |
| HOG-GCN | 86.01 ± 1.44 | 75.25 ± 2.37 | 88.50 ± 0.61 |
| BM-GCN  | 87.60 ± 1.32 | 76.17 ± 1.49 | 89.32 ± 0.76 |
| GCN-EHHD| 87.20 ± 1.36 | 76.34 ± 1.86 | 90.01 ± 0.60 |

By inspecting Tables 2 and 3, we have the following observations. On one hand, for homophilic graphs, the performances of GCN-based models under the homophily assumption were similar to that of GCN-based models tailored for dealing with heterophily. This shows that the emerging methods for tackling heterophily have good compatibility for homophily. For heterophilic graphs, these methods tailored for heterophily significantly
outperformed traditional GCNs. On the other hand, for homophilic graphs, our GCN-EHHD outperformed all the baselines on Citeseer and Pubmed and achieved the second best on Cora. For heterophilic graphs, GCN-EHHD outperformed all the baselines on the five datasets. Especially, GCN-EHHD achieved 4.83%, 3.17% and 2.01% improvement on Texas, Squirrel and Cornell, respectively. Therefore, our proposed model can achieve the best or comparable performance on both homophilic and heterophilic graphs.

Table 3. Node classification accuracy on heterophilic graphs (in % ± Standard Error). The best results are in bold, and the second-best results are underlined.

| Model       | Texas   | Wisconsin | Squirrel | Chameleon | Cornell |
|-------------|---------|-----------|----------|-----------|---------|
| MLP         | 78.37 ± 4.93 | 84.90 ± 4.62 | 33.02 ± 1.38 | 47.49 ± 2.63 | 80.05 ± 6.67 |
| Deepwalk    | 55.40 ± 4.80 | 48.23 ± 6.00 | 47.58 ± 1.36 | 61.27 ± 2.74 | 50.81 ± 5.22 |
| GCN         | 54.32 ± 8.58 | 46.86 ± 4.37 | 38.58 ± 2.02 | 54.97 ± 2.60 | 50.81 ± 7.40 |
| GAT         | 55.67 ± 5.72 | 50.19 ± 4.73 | 40.46 ± 2.28 | 61.64 ± 1.30 | 55.94 ± 7.85 |
| MixHop      | 71.62 ± 6.14 | 81.76 ± 4.98 | 30.22 ± 1.30 | 38.72 ± 2.12 | 72.43 ± 6.47 |
| Geom-GCN    | 67.83 ± 4.08 | 64.50 ± 4.06 | 38.30 ± 1.05 | 60.72 ± 2.54 | 60.54 ± 3.00 |
| FAGCN       | 72.70 ± 6.16 | 80.19 ± 5.58 | 42.82 ± 2.80 | 61.79 ± 4.06 | 71.89 ± 7.22 |
| H2GCN       | 78.91 ± 6.08 | 85.68 ± 5.31 | 34.63 ± 1.58 | 46.86 ± 2.47 | 79.45 ± 6.26 |
| HOG-GCN     | 81.35 ± 5.16 | 84.31 ± 5.46 | 37.19 ± 1.54 | 53.59 ± 2.56 | 80.27 ± 6.74 |
| BM-GCN      | 83.78 ± 6.49 | 77.84 ± 6.13 | 52.68 ± 1.79 | 68.85 ± 2.84 | 71.08 ± 6.11 |
| GCN-EHHD    | 87.83 ± 3.87 | 86.27 ± 3.81 | 54.35 ± 1.36 | 69.32 ± 3.23 | 81.89 ± 3.83 |

The accuracies of MLP1 and MLP2 of the GCN-EHHD are also meaningful for analyzing the performance of GCN-EHHD. We show the accuracies of pre-trained MLP1, MLP2 and final MLP1, MLP2 after fine-tuning in Table 4. In our experiments, in general, the better the MLPs accuracies were, the better the accuracy of the whole GCN-EHHD was, especially for the large graph. We discuss the effect of pre-trained estimators in Section 4.4.

Table 4. Node classification accuracies of MLP1 and MLP2 of the GCN-EHHD (in % ± Standard Error).

| Dataset      | Pre-Trained MLP1 | Pre-Trained MLP2 | Fine-Tuned MLP1 | Fine-Tuned MLP2 |
|--------------|------------------|------------------|----------------|----------------|
| Cora         | 72.55 ± 1.66     | 80.96 ± 1.46     | 71.24 ± 2.81   | 80.74 ± 1.74   |
| Citeseer     | 70.80 ± 2.82     | 61.20 ± 2.12     | 70.49 ± 2.71   | 61.03 ± 1.74   |
| Pubmed       | 85.92 ± 0.67     | 82.02 ± 0.46     | 87.66 ± 0.72   | 81.45 ± 0.48   |
| Texas        | 78.37 ± 4.93     | 55.40 ± 4.80     | 78.16 ± 5.72   | 51.10 ± 6.47   |
| Wisconsin    | 84.90 ± 4.62     | 48.23 ± 6.00     | 83.68 ± 3.70   | 43.78 ± 3.77   |
| Cornell      | 80.05 ± 6.67     | 50.81 ± 5.22     | 79.68 ± 3.70   | 46.75 ± 3.67   |
| Chameleon    | 47.49 ± 2.63     | 61.27 ± 2.74     | 45.06 ± 2.40   | 61.51 ± 2.70   |
| Squirrel     | 33.02 ± 1.38     | 47.58 ± 1.36     | 31.75 ± 1.29   | 48.58 ± 1.76   |

Next, we discuss the computation complexity and model complexity compared with three representative and competitive models, including FAGCN, HOG-GCN and BM-GCN. To facilitate comparison, we assume that for each model, the dimensions of the input layer, the output layer and all hidden layers are equal and denoted by \( d \). Let \( K \) be the number of network layers and \( L \) be the number of layers of MLP for pre-training the estimators, let \( |V| \) be the number of nodes and \( |E| \) be the number of edges in a graph. We also take the time cost and the numbers of parameters of FAGCN, HOG-GCN, BM-GCN and GCN-EHHD when conducting experiments on the Cora datasets as examples. Note that the time cost and the numbers of parameters of the three models for comparison are calculated according to their source code implementation.

The computation complexity of a single layer in GCN-EHHD is \( O(3d^2|V|) \), and they are \( O(d(|V| + |E|)) \), \( O(2d^2|V|) \), \( O(2d^2|V|) \) for FAGCN, HOG-GCN and BM-GCN, respectively. Note that for HOG-GCN, BM-GCN and GCN-EHHD, the estimated results are calculated at the beginning of model reasoning and are used at each layer of the model.
The coefficients of messages must be calculated at each layer for FAGCN, and the linear transformation of features is only used in the first and the last layers in FAGCN but must be used in each layer in HOG-GCN, BM-GCN and GCN-EHHD. So, the computation time of FAGCN grows with the increase of the number of edges of the graph, and the computation time of HOG-GCN, BM-GCN and GCN-EHHD grows in a square number with the increase of the representation dimension of the hidden layer. In our experiments on the Cora dataset, the time cost of one-pass forward reasoning process in FAGCN, HOG-GCN, BM-GCN and GCN-EHHD were 2.00 ms, 361.03 ms, 1.99 ms and 1.99 ms, respectively. It can be seen that our proposed GCN-EHHD consumes relatively less time.

The complexity of parameters in our GCN-EHHD is \( O((3K + 2L)d^2) \), and they are \( O(2d^2 + Kd) \), \( O((2K + L)d^2 + K|V|^2) \) and \( O((2K + L)d^2) \) in FAGCN, HOG-GCN and BM-GCN, respectively. Note that the complexity of parameters in HOG-GCN is able to become \( O((2K + L)d^2 + K|E|) \) by modeling the edge weights rather than the homophily degree matrix and overlooking the message passing from high-order neighborhoods. Among these models, the number of parameters in FAGCN is generally the smallest, and the number of parameters in HOG-GCN is generally the largest, since the number of nodes and the number of edges in the graph are generally much larger than the hidden layer dimension of the model. Taking the parameters used in the four models when conducting experiments on the Cora dataset as an example, the numbers of parameters of FAGCN, HOG-GCN, BM-GCN and GCN-EHHD were 92.49 k, 26,154.75 k, 336.46 k and 553.36 k, respectively. It can be seen that, our proposed GCN-EHHD achieves an improvement in the experimental performance without too much increase in the number of parameters.

### 4.3. Visualization

In order to more visually observe the performance of our proposed model and the compared baselines, we performed the visualization tasks for GCN, FAGCN, BM-GCN and GCN-EHHD on Chameleon that is a heterophilic graph. We extracted the output embedding of the last layer of the four models and used t-SNE [35] to project them to a 2D space. The projected result can be seen in Figure 3 where we can find some meaningful phenomena. First, for GCN, many nodes with different categories are mixed together, which once again shows the defect of GCN for learning tasks on heterophilic graphs. Next, for FAGCN, most nodes with the same category are basically clustered together, but the boundaries between clusters are not obvious. For BM-GCN, the classification result was further improved, but there is still no clear boundary between two clusters (i.e., the blue cluster and the purple cluster) in Figure 3c. Finally, for our proposed GCN-EHHD, the boundaries between clusters are the clearest compared with the other three models. This visualization result once again demonstrates the effectiveness of our proposed model.

![Figure 3](image-url). The visualization of the output embedding of the last layer in GCN, FAGCN, BM-GCN and GCN-EHHD on the Chameleon dataset by t-SNE with nodes colored according to their class labels.

### 4.4. Ablation Study

In order to explore the role of negative message passing guided by heterophily degree and estimator generated by Deepwalk, we conducted an ablation study. Recall that the hyperparameter \( \alpha \) in Equation (14) controls the proportion of aggregated positive and negative messages from the neighbors, and the hyperparameter \( \theta \) in Equation (9) controls the
proportion of node attributes and graph structure information when estimating homophily degree and heterophily degree. We therefore perform an ablation study by comparing different settings of parameters $\alpha$ and $\theta$, and report their effects on the predictive accuracy of node classification tasks.

We chose eleven values of $\alpha$ in the interval $[0, 1]$. When $\alpha = 1$, it amounts to discarding the part of negative message passing and only aggregating positive messages from neighbors. When $\alpha = 0$, the reverse is true. For each setting of $\alpha$, we chose eleven values of $\theta$ ranging from 0 to 1. When $\theta = 0$, we only used the pre-trained estimator generated based on graph structure information extracted by Deepwalk. When $\theta = 1$, we only considered node attributes for estimating the homophily and heterophily degree. For each setting of $\alpha$ and $\theta$, we conducted node classification experiments as described in Section 4.2 on the aforementioned eight datasets in Section 4.1 and report the experimental results as given in Figures 4–7.

**Figure 4.** Analysis results of the ablation study on $\alpha$ for homophilic graphs. The x-axis is different settings of $\alpha$. The y-axis of figures on the left shows the average accuracy under eleven different settings of $\theta$, and the y-axis of figures on the right shows the accuracy for $\theta \in \{0, 0.5, 1\}$, respectively.
Figure 5. Cont.
Figure 5. Analysis results of the ablation study on $\alpha$ for heterophilic graphs. The x-axis is different settings of $\alpha$. The y-axis of figures on the left shows the average accuracy under eleven different settings of $\theta$, and the y-axis of figures on the right shows the accuracy for $\theta \in \{0, 0.5, 1\}$, respectively.

Figure 6. Analysis results of the ablation study on $\theta$ for homophilic graphs. The x-axis is different settings of $\theta$. The y-axis of figures on the left shows the average accuracy under eleven different settings of $\alpha$, and the y-axis of figures on the right shows the accuracy for $\alpha \in \{0, 0.5, 1\}$, respectively.
Figure 7. Cont.
Figure 7. Analysis results of the ablation study on $\theta$ for heterophilic graphs. The x-axis is different settings of $\theta$. The y-axis of figures on the left shows the average accuracy under eleven different settings of $\alpha$, and the y-axis of figures on the right shows the accuracy for $\alpha \in \{0, 0.5, 1\}$, respectively.

The analysis results of the ablation study on $\alpha$ are shown in Figures 4 and 5. For each sub-figure, the x-axis is different settings of $\alpha$, and under each setting of $\alpha$, the y-axis coordinate value of the figures on the left shows the average accuracy given the aforementioned eleven different settings of $\theta$, and the y-axis coordinate value of the figures on the right shows the classification accuracy for $\theta \in \{0, 0.5, 1\}$, respectively. As shown, there are three main situations. For the first situation, the influence of $\alpha$ on the model accuracy in average is small for graphs such as Cora, Citeseer and Pubmed. For the second situation, the plots are characterized by sort of “reverse U-shape”, where the accuracy values for $\alpha = 0$ and $\alpha = 1$ are close to each other, and the best results are obtained in the vicinity of $\alpha = 0.5$. The analysis results for Wisconsin, Cornell and Texas are this case. In the third situation, the plots for Chameleon and Squirrel are characterized by sort of “U-shape”. When $\alpha = 1$, that is, the negative message passing is cancelled, the performance is not the best in the eight datasets, and in particular is poor on Cornell, Wisconsin and Texas. This validates the effectiveness of negative message passing. For Cora, Citeseer, Pubmed, Wisconsin, Cornell and Texas, when the value of $\alpha$ is set to be approximately 0.5, the performance is the best. Consider that the ablated GCN-EHHD models with $\alpha \in \{0, 1\}$ are similar, so the accuracy for $\alpha = 0$ and $\alpha = 1$ is close. For Chameleon and Squirrel, the performance is the best when $\alpha = 0$; that is, we only use negative message passing, and the performance is the worst in the vicinity of $\alpha = 0.5$. We think the difference is related to the average degree of nodes in graphs. As shown in Table 1, the average degrees of nodes in Chameleon and Squirrel are both much larger than the other datasets. For graphs with a small average degree, the influence of homophilic neighbors and heterophilic neighbors can both play important roles in message passing on heterophilic graphs. This may be the reason why the best performance is obtained in the vicinity of $\alpha = 0.5$ for Wisconsin, Cornell and Texas. Therefore, while for most graphs we can obtain a good performance by mixing positive messages with negative messages, for some heterophilic graphs with large average node degrees, it may be better to keep using negative messages alone.

The analysis results of the ablation study on $\theta$ are shown in Figures 6 and 7. For each sub-figure, the x-axis is different settings of $\theta$, and under each setting of $\theta$, the y-axis coordinate value of the figures on the left shows the average accuracy given the aforementioned eleven different settings of $\alpha$. The y-axis coordinate value of the figures on the right shows the classification accuracy for $\alpha \in \{0, 0.5, 1\}$, respectively. In addition, we also annotate each sub-figure with the accuracy of two pre-trained MLPs. As shown, there are three main situations. For the first situation, the better accuracy of GCN-EHHD is obtained for the values of $\theta$ that give more importance to the pre-trained MLP with best performances on graphs such as Chameleon and Squirrel as shown in Figure 7e,h and Figure 7i,j, and Pubmed as shown in Figure 6e and Figure 6f, respectively. These three graphs are relatively large compared with other datasets. In particular, the average
degree of nodes in Chameleon and Squirrel is quite large. For the second situation, the influence of $\theta$ on the accuracy of GCN-EHHD is small for graphs such as Cora and Citeseer as shown in Figure 6a,b and Figure 6c,d, respectively. The difference in accuracy between the two pre-trained MLPs used for analyzing Cora and Citeseer is about 10% and not as significant as for other datasets. For the third situation, the accuracy of GCN-EHHD fluctuates with the change of $\theta$ on graphs such as Texas, Cornell and Wisconsin as shown in Figure 7c–f and Figure 7a,b, respectively. The special performance in the third situation may mainly be because the three graphs are all relatively small which causes instability. The results indicate that the pre-trained estimator using Deepwalk is useful in measuring the homophily and heterophily degree on some large graphs, especially for those with large node degrees.

5. Conclusions

We proposed a new model called GCN-EHHD in order to better generalize GCN-based methods to deal with heterophilic graphs. In GCN-EHHD, we pre-trained estimators on homophily degree and heterophily degree between neighbouring nodes and leveraged the information to adaptively guide propagation/aggregation of the positive messages and negative messages. Since both node attributes and graph topology matter to the prediction accuracy, we propose applying Deepwalk to extract the structure information of graphs for training an estimator that is combined with the estimator learned from original node attributes to estimate accurate homophily and heterophily degree between node pairs. Our experiments with eight datasets demonstrate the effectiveness of the proposed approach. The proposed new approach achieved state-of-the-art results on three homophilic graphs and outperformed baselines on five heterophilic graphs. Our extensive experiments show that the explicit propagation of signed messages adaptively guided by heterophily degree is important in the message-passing process of heterophilic graphs. The exploration of graph structure information by using Deepwalk turned out to be useful for large heterophilic graphs such as Chameleon and Squirrel, for which the graph topology seemed to play a more decisive role than node attributes as shown by the accuracy of pre-trained estimators. In future work, it would be interesting to explore models other than MLPs to train more effective estimators and study more large graphs with a variety of statistical properties.

Author Contributions: Conceptualization, R.Z. and X.L.; methodology, R.Z. and X.L.; software, R.Z.; validation, X.L.; formal analysis, R.Z. and X.L.; funding acquisition, X.L.; investigation, R.Z.; resources, X.L.; data curation, R.Z. and X.L.; writing—original draft preparation, R.Z. and X.L.; writing—review and editing, R.Z. and X.L.; visualization, R.Z.; supervision, X.L.; project administration, X.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Ministry of Science and Technology of China under Grant No. 2018YFC0830400, National Natural Science Foundation of China under Grant No. 61802126.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data analysed in this study can be accessed at https://github.com/xtyi1997/geom-gcn (accessed on 20 August 2022).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
2. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph attention networks. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
3. Hamilton, W.; Ying, Z.; Leskovec, J. Inductive representation learning on large graphs. Adv. Neural Inf. Process. Syst. 2017, 30. [CrossRef]
4. Leskovec, K.X.W.H.J.; Jegelka, S. How powerful are graph neural networks? In Proceedings of the International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 6–9 May 2019.

5. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Philip, S.Y. A comprehensive survey on graph neural networks. IEEE Trans. Neural Netw. Learn. Syst. 2020, 32, 4–24. [CrossRef] [PubMed]

6. Yao, L.; Mao, C.; Luo, Y. Graph convolutional networks for text classification. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27–28 January 2019; Volume 33, pp. 7370–7377.

7. Marcheggiani, D.; Titov, I. Encoding sentences with graph convolutional networks for semantic role labeling. In Proceedings of the EMNLP, Copenhagen, Denmark, 7–11 September 2017.

8. Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W.L.; Leskovec, J. Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 974–983.

9. He, X.; Deng, K.; Wang, X.; Li, Y.; Zhang, Y.; Wang, M. Lightgcn: Simplifying and powering graph convolution network for recommendation. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Xi’an, China, 25–30 July 2020; pp. 639–648.

10. Yan, Y.; Zhu, J.; Duda, M.; Solarz, E.; Sripada, C.; Koutra, D. Groupinn: Grouping-based interpretable neural network for classification of limited, noisy brain data. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 772–782.

11. Zheng, X.; Liu, Y.; Pan, S.; Zhang, M.; Jin, D.; Yu, P.S. Graph neural networks for graphs with heterophily: A survey. arXiv 2022, arXiv:2202.07082.

12. Gilmer, J.; Schoenholz, S.S.; Riley, P.F.; Vinyals, O.; Dahl, G.E. Neural message passing for quantum chemistry. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 1263–1272.

13. Chung, F.R.; Graham, F.C. Spectral Graph Theory; American Mathematical Society: Providence, RI, USA, 1997; Volume 92.

14. Bruna, J.; Zaremba, W.; Szlam, A.; LeCun, Y. Spectral networks and deep locally connected networks on graphs. In Proceedings of the 2nd International Conference on Learning Representations (ICLR 2014), Banff, AB, Canada, 14–16 April 2014.

15. Wu, F.; Souza, A.; Zhang, T.; Fifty, C.; Yu, T.; Weinberger, K. Simplifying graph convolutional networks. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 6861–6871.

16. Nt, H.; Machara, T. Revisiting graph neural networks: All we have is low-pass filters. arXiv 2019, arXiv:1905.09550.

17. Pandit, S.; Chau, D.H.; Wang, S.; Faloutsos, C. Netprobe: A fast and scalable system for fraud detection in online auction networks. In Proceedings of the 16th international conference on World Wide Web, Banff, AB, Canada, 8–12 May 2007; pp. 201–210.

18. Pei, H.; Wei, B.; Chang, K.C.C.; Lei, Y.; Yang, B. Geom-gcn: Geometric graph convolutional networks. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.

19. Abu-El-Haija, S.; Perozzi, B.; Kapoor, A.; Alipourfard, N.; Lerman, K.; Harutyunyan, H.; Ver Steeg, G.; Galstyan, A. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 21–29.

20. Bo, D.; Wang, X.; Shi, C.; Shen, H. Beyond low-frequency information in graph convolutional networks. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 2–9 February 2021; Volume 35, pp. 3950–3957.

21. Shuman, D.I.; Narang, S.K.; Frossard, P.; Ortega, A.; Vanderheynst, P. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. IEEE Signal Process. Mag. 2013, 30, 83–98. [CrossRef]

22. Wang, T.; Jin, D.; Wang, R.; He, D.; Huang, Y. Powerful Graph Convolutional Networks with Adaptive Propagation Mechanism for Homophily and Heterophily. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 6–10 November 2022; Volume 36, pp. 4210–4218.

23. He, D.; Liang, C.; Liu, H.; Wen, M.; Jiao, P.; Feng, Z. Block modeling-guided graph convolutional neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 6–10 November 2022; Volume 36, pp. 4022–4029.

24. Perozzi, B.; Al-Rfou, R.; Skiena, S. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014; pp. 701–710.

25. Liu, M.; Wang, Z.; Ji, S. Non-local graph neural networks. IEEE Trans. Pattern Anal. Mach. Intell. 2021. [CrossRef] [PubMed]

26. Zhu, J.; Yan, Y.; Zhao, L.; Heimann, M.; Akoglu, L.; Koutra, D. Beyond homophily in graph neural networks: Current limitations and effective designs. Adv. Neural Inf. Process. Syst. 2020, 33, 7793–7804.

27. Jin, D.; Yu, Z.; Huan, C.; Wang, R.; Wang, X.; He, D.; Han, J. Universal Graph Convolutional Networks. 34. In Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS 2021), online, 6–14 December 2021.

28. Yan, Y.; Hashemi, M.; Swersky, K.; Yang, Y.; Koutra, D. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. arXiv 2021, arXiv:2102.06462.

29. Zhu, J.; Rossi, R.A.; Rao, A.; Mai, T.; Lipka, N.; Ahmad, N.K.; Koutra, D. Graph neural networks with heterophily. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 2–9 February 2021; Volume 35, pp. 11168–11176.

30. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. arXiv 2013, arXiv:1301.3781.

31. Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; Eliassi-Rad, T. Collective classification in network data. AI Mag. 2008, 29, 93. [CrossRef]
32. Namata, G.; London, B.; Getoor, L.; Huang, B.; Edu, U. Query-driven active surveying for collective classification. In Proceedings of the 10th International Workshop on Mining and Learning with Graphs, Scotland, UK, 1 July 2012; Volume 8, p. 1.

33. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the ICLR (Poster), San Diego, CA, USA, 7–9 May 2015.

34. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.* 2019, 32.

35. Van der Maaten, L.; Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* 2008, 9, 2579–2605.