Ladder-GNN: Hop-Aware Representation Learning for Graph Neural Networks

Ailing Zeng, Minhao Liu, Zhiwei Liu, Ruiyuan Gao, Qiang Xu
{alzeng,mhliu,rygao,qxu}@cse.cuhk.edu.hk, zliu213@uic.edu

ABSTRACT
In the representation learning of Graph Neural Networks (GNNs), as the messages passed among nodes contain both information and noise, it is critical to retrieve information effectively while suppressing noise. Generally speaking, interactions with distant nodes introduce more noise for a particular node than those with close neighbours. However, in most existing works, the messages being passed among nodes are mingled together, which is inefficient from a communication perspective.

Motivated by the above, we propose a simple yet effective hop-aware aggregation scheme, resulting in a ladder-style GNN architecture, namely Ladder-GNN. Specifically, we separate messages from different hops, assign different dimensions for them, and then concatenate them to obtain the node representation. Such disentangled representations facilitate improving the information-to-noise ratio of messages passed from different hops. To explore an effective hop-dimension relationship, we propose a conditionally progressive neural architecture search strategy. The resulting hop-aware representations generally contain more dimensions for low-order neighbours and fewer dimensions for high-order neighbours, leading to a ladder-style architecture. This observation motivates us to introduce an efficient approximate hop-dimension relation function in Ladder-GNN design. We verify the proposed Ladder-GNN on seven semi-supervised node classification datasets, including both homogeneous and heterogeneous graphs. Experimental results show that the proposed simple hop-aware representation learning solution outperforms existing techniques.

CCS CONCEPTS
• Computing methodologies → Graph Neural networks.

KEYWORDS
Graph Neural Network, Representation Learning, Hop-Aware Aggregation

1 INTRODUCTION
Recently, a large number of research efforts have been dedicated to applying deep learning methods to graphs, known as graph neural networks (GNNs) [14, 35], achieving great success in modelling non-structured data, e.g., social networks [24] and recommendation systems [25].

Learning an effective low-dimensional embedding to represent each node in the graph is arguably the most important task for GNN learning, wherein the node embedding is obtained by aggregating information with its direct and indirect neighbouring nodes passed through GNN layers [11]. Earlier GNN works usually aggregate with neighboring nodes that are within short range. For many graphs, this may cause the so-called under-reaching issue [3].

To verify the effectiveness of the proposed simple hop-aware representation learning solution, we demonstrate it on seven semi-supervised node classification datasets for both homogeneous and heterogeneous graphs. Experimental results show that the proposed Ladder-GNN solution can achieve state-of-the-art performance on most of them.

• We take a communication perspective on GNN message passing. That is, we regard the target node for representation learning as the receiver and group the set of neighbouring nodes with the same distance to it as a transmitter that carries both information and noise. The dimension of the message can be regarded as the capacity of the communication channel. Then, aggregating neighbouring information from multiple hops becomes a multi-source communication problem with multiple transmitters over the communication channel.

• To simplify node representation learning, we propose to separate the messages from different transmitters (i.e., Hop-k neighbours), each occupying a proportion of the communication channel (i.e., disjoint message dimensions). As the information-to-noise ratio from high-order neighbours is usually lower than low-order neighbours, the resulting hop-aware representation is usually unbalanced with more dimensions allocated to low-order neighbours, leading to a ladder-style aggregation scheme.

• To explore the dimension allocation for neighbouring nodes from different hops effectively, we propose a conditionally progressive neural architecture search (NAS) strategy. Motivated by the search results, we introduce an approximate hop-dimension relation function, which can generate close results to the NAS solution without applying compute-expensive NAS.
GNNs adopt message passing to learn node embeddings, which involves two steps for each node: neighbour aggregation and linear transformation [11]. The following formula presents the mathematical form of message passing in a graph convolutional network (GCN) [14]. Given an undirected graph $G = (V, E)$ with $N$ nodes and adjacency matrix $A$, we can aggregate node features at the $l$-th layer $H^{(l)} \in \mathbb{R}^{N \times C^{(l)}}$ as:

$$H^{(l+1)} = \sigma(\hat{A}H^{(l)}W^{(l)})$$

(1)

where $\hat{A} = D^{-1/2}(A + I_N)D^{-1/2}$ is the augmented normalized adjacency matrix of $G$, $I_N$ is the identity matrix and $D_{ii} = \sum_j (A + I_N)_{ij}$. $W^{(l)} \in \mathbb{R}^{C^{(l)} \times C^{(l-1)}}$ is the trainable weight matrix at the $l$-th layer used to update node embeddings. $C^{(l)}$ and $C^{(l-1)}$ are the channel size of the input and hidden layer, respectively. $\sigma(\cdot)$ is an activation function.

By taking the importance of different neighbours into consideration, graph attention network (GAT) [35] applies a multi-head self-attention mechanism during aggregation and achieves higher performance than GCN in many datasets. Recent GNN works improve GCN or GAT from two aspects: (i). Some make changes to the input graphs [7, 26, 31, 44, 49]. For example, [26] proposes to drop some task-irrelevant or noisy edges to achieve high generalization capability. (ii). Without changing the graph, some GNN works try to further extract relevant information from high-order neighbours. Our work belongs to this category, and it can be easily combined with those GNN works in the first category that modifies the adjacency matrix $A$.

In order to aggregate information from high-order neighbours, some earlier works [18, 42, 43] simply stack deeper networks to retrieve such information recursively. To mitigate the possible overfitting issues (due to model complexity), SGC [39] removes the nonlinear operations and directly aggregates node features from multiple hops. To relieve the potential over-smoothing problem that results in less discriminative node representations (due to over-mixing) [19], various hop-aware aggregation solutions are proposed. Some of them (e.g. HighOrder Aggregation [28], MixHop [2], N-GCN [1], GB-GNN [29]) employ multiple convolutional branches to aggregate neighbors from different hops. Others (e.g. AM-GCN [38], HWGCN [22], MultiHop [52]) try to learn adaptive attention scores when aggregating neighboring nodes from different hops.

In Fig. 1, we plot the homophily ratio for nodes in the Pubmed dataset. As can be observed from the figure, with the increase of hop distance, the percentage of neighbouring nodes with the same label decreases, indicating a diminishing information-to-noise ratio for messages ranging from low-order neighbours to high-order neighbours. Therefore, the critical issue in GNN message passing is how to retrieve information effectively while suppressing noise simultaneously. However, all the existing hop-aware aggregation schemes do not explicitly consider this issue, making discriminative feature extraction challenging. This observation motivates us to propose our LADDER-GNN architecture, as detailed in the following section.

### 3 METHOD
In Sec. 3.1, we take a communication perspective on GNN message passing and representation learning. Then, we give an overview of the proposed LADDER-GNN framework in Sec. 3.2. Next, we explore the dimensions of different hops with an RL-based NAS strategy in Sec. 3.3 and then introduce the approximate hop-dimension relation function in Sec. 3.4.

#### 3.1 GNN Representation Learning from a Communication Perspective
In GNN representation learning, messages are passed from neighbouring nodes to the target node and updated its embedding. Fig. 2 presents a communication perspective on GNN message passing, wherein we regard the target node as the receiver. Considering neighbouring nodes from different hops tend to contribute unequally (see Figure 1), we group the set of neighbouring nodes with the same distance as one transmitter, and hence we have $K$ transmitters if we would like to aggregate up to $K$ hops. The dimension of the message can be regarded as the communication channel capacity. Then, GNN message passing becomes a multi-source communication problem.

Some existing GNN message-passing schemes (e.g., SGC [39], JKNet [43], and $S^2$GC [51]) aggregate neighboring nodes before transmission, as shown in Figure 2(b), which mix clean information source and noisy information source directly. The other hop-aware GNN message-passing schemes (e.g., AMGNC [38], MultiHop [52], and MixHop [2]) as shown in Figure 2(c) first conduct aggregation within each hop (i.e., using separate weight matrix) before transmission over the communication channel, but they are again mixed afterward.
Figure 2: An illustration of GNN message passing and representation learning from a communication perspective. (a) A communication system contains transmitters that encode source information, communication channel, and receivers that decode the original information; (b) GNN representation learning with existing node aggregation scheme; (c) GNN representation learning with existing hop-aware aggregation scheme; (d) GNN representation learning with the proposed ladder-style aggregation scheme.

Different from a conventional communication system that employs a well-developed encoder for the information source, one of the primary tasks in GNN representation learning is to learn an effective encoder that extracts useful information with the help of supervision. Consequently, mixing clean information sources (mostly low-order neighbours) and noisy information sources (mostly high-order neighbours) makes the extraction of discriminative features challenging.

The above motivates us to perform GNN message passing without mixing up messages from different hops, as shown in Figure 2(d). At the receiver, we concatenate the messages from various hops, and such disentangled representations facilitate extracting useful information from various hops with little impact on each other. Moreover, dimensionality significantly impacts any neural networks’ generalization and representation capabilities [3, 4, 21, 33], as it controls the amount of quality information learned from data. In GNN message passing, the information-to-noise ratio of low-order neighbours is usually higher than that of high-order neighbours. Therefore, we tend to allocate more dimensions to close neighbours than distant ones, leading to a ladder-style aggregation scheme.

### 3.2 Ladder-Aggregation Framework

With the above, Figure 3 shows the node representation update procedure in the proposed Ladder-GNN architecture. For a particular target node (the center node in the figure), we first aggregate node within each hop, which can be conducted by existing node-wise aggregation methods (e.g., GCN or GAT). Next, we determine the dimensions for the aggregated messages from different hops and then concatenate them, instead of mixing them up, for inter-hop aggregation. Finally, we perform a linear transformation to generate the updated node representation.

Specifically, given the graph $G = (V, E)$ for representation learning, $K$ is the maximum number of neighboring hops for node aggregation. For each group of neighbouring nodes at Hop-$k$, we determine their respective optimal dimensions and then concatenate their embeddings into $H$ as follows:

$$H = \|_{k \in [1, K]} (\tilde{A}_k X W_k),$$

where $\tilde{A}_k$ is the normalized adjacency matrix of the $k_{th}$ hop and $X$ is the input feature. A learnable matrix $W_k \in \mathbb{R}^{C_i \times C_o}$ controls the output dimension of the $k_{th}$ hop as $C_o \parallel k$ means concatenation. Encoding messages from different hops with distinct $W_k$ avoids the over-mixing of neighbours, thereby alleviating the impact of noisy information sources on clean information sources during GNN message passing. Accordingly, $H$ is a hop-aware disentangled representation of the target node. Then, with the classifier $f$ after the linear layer $W_U$, we have:

$$\tilde{Y} = f(HW_U),$$

where $\tilde{Y}$ is the output softmax values. Given the supervision $Y$ of some nodes, we can use a cross-entropy loss to calculate gradients and optimize the above weights in an end-to-end manner.

With the above, if the adjacency matrix $A$ are the same as the original GCN architecture, the resulting GNN architecture with our ladder-aggregation framework is namely Ladder-GCN. Similarly, when we employ a self-attention scheme within hops to obtain the attention-based adjacency matrix $\tilde{A}$ as in the original GAT architecture, the resulting GNN architecture is namely Ladder-GAT. Please note, our proposed ladder-aggregation scheme could also be integrated into other GNN architectures (e.g., [26]).

### 3.3 Hop-Aware Dimension Search

Allocating different dimensions for messages from different hops is the key in LADDER-GNN design. As there are numerous hop-dimension allocation possibilities, determining an appropriate allocation is a non-trivial task. In recent years, neural architecture search (NAS) has been extensively researched, which automatically designs deep neural networks with comparable or even higher performance than manual designs by experts (e.g., [5, 20, 23, 34, 53]). Existing NAS works in GNNs [10, 32, 50] search the graph architectures (e.g., $t$-hop aggregators, activation function, aggregation type, attention type, etc) and hyper-parameters to reach better performance. However, they ignore to aggregate multi-hop neighbours, let alone the dimensionality of each hop. In the following, we introduce our proposed NAS solution.
Search Space: Different from previous works in GNNs [10, 46, 50], our search space focuses on the dimension of each hop, called hop-dimension combinations. To limit the possible search space $O$ for hop-dimension combinations, we apply exponential sampling $2^0, 2^1, 2^2, \ldots, C_i, 2^{(n-1)}, 2^{(n)}$ strategies for dimensions. $n$ are hyper-parameters, representing the index and sampling granularity to cover the possible dimensions. For each strategy, the search space should also cover the dimension of initial input feature $C_i$.

Basic Search Algorithm: Given the search space $O$, we target finding the best model $M^* \in M$ to maximize the expected validation accuracy. We choose the reinforcement learning strategy since its reward is easy to customize for our problem. As shown in Figure 4, a LSTM controller based on the parameters $\theta$ generates a sequence of actions $a_{i,k}$ with length $K$, where each hop dimension $C_k (1 \leq k \leq K)$ is sampled from the search space mentioned above. Then, we can build a model $M$ mentioned in Sec. 3.2, and train it with a cross-entropy loss function. Then, we test it on the validation set $D$ to get an accuracy $R_D(M)$. Next, we can use the accuracy as a reward signal and perform a policy gradient algorithm to update the parameters $\theta$ so that the controller can generate better hop-dimension combinations iteratively. The objective function of the model is shown in:

$$M^* = \arg \max_M \mathbb{E}_{P(a_{i,k}, \theta)} [R_D(M)]. \quad (4)$$

Conditionally Progressive Search Algorithm: Considering the extremely large search space with the basic search algorithm, e.g., the search space size will be $(n + 1)^k$ for the exponential sampling with $k$ hops. This makes it more challenging to search for the optimal combinations with limited computational resources. Moreover, we find that there are a large number of redundant actions in our search space. To improve the efficiency and effectiveness of the search procedure, we are inspired to propose a conditionally progressive search algorithm.

That is, instead of searching the entire space all at once, we divide the searching process into multiple phases, starting with a relatively small number of hops, e.g., $K = 3$. After obtaining their results, we only keep those hop-dimension combinations that are promising, where they are regarded as the conditional search space, with high $R_D(M)$.

Next, we conduct the hop-dimension search for the $(K+1)_k$ hop based on the conditional search space filtered from the last step, and again, keep those combinations with high $R_D(M)$. This procedure is conducted progressively until aggregating more hops cannot boost performance. With this algorithm, we can largely reduce the redundant search space to enhance search efficiency.

3.4 Hop-Dimension Relation Function

The computational resources required to conduct NAS are extremely expensive for large graphs, even with the proposed progressive search algorithm. Therefore, it is essential to have an efficient method to determine the dimensionality of every hop in practical applications. From our NAS experimental results (Sec. 4.1), we observe that the low-order neighbours within $L$ hops are usually directly aggregated with the original feature dimensions while high-order neighbours are associated with an approximately exponentially decreasing dimensions.

This motivates us to propose a simple yet effective hop-dim relation function to approximate the NAS solutions. The output dimension of $k_{th}$ hop is:

$$C_{o,k} = d^{\max (k-L,0)} \cdot C_i, \quad (5)$$

where $0 < d < 1$ is the dimension compression ratio, and $C_i$ is the dimension of the input feature. With such an approximate function, there is only one hyper-parameter to determine, significantly reducing the computational cost.
4 EXPERIMENT

In this section, we validate the effectiveness of Ladder-GNN on seven widely-used semi-supervised node classification datasets. We first analyze the NAS results in Sec. 4.1. Then, we combine the proposed hop-aware aggregation scheme with the approximate function with existing GNNs in Sec. 4.2. Moreover, as a new hop-aware aggregation scheme, in Sec. 4.3, we quantitatively compare with exiting works. Furthermore, we conduct experiments on heterogeneous graphs in Sec. 4.4. Last, we show an ablation study on the proposed hop-dim relation function in Sec. 4.5.

Data description: For the semi-supervised node classification task on homogeneous graphs, we evaluate our method on five datasets: Cora [45], Citeseer [45], Pubmed [45], OGB-Arxiv [13] and OGB-Products [13]. We split the training, validation and test set following earlier works [2, 14, 35, 38]. Furthermore, on heterogeneous graphs, we verify the methods on two datasets: ACM and IMDB [37]. Due to page limits, more details about dataset descriptions, data pre-processing procedure and more comparison with existing methods are listed in the Appendix.

Experimental Settings: Our experiments are conducted on a machine with an NVIDIA Tesla V100-SXM2-32GB GPU card. Most experiments take up a few hundred MBs memory.

For the NAS experiment, the controller is a one-layer LSTM with 100 hidden units. We train it with Adam optimizer. The learning rate is 3.5 × 10⁻⁴, mini-batch is 128, and the weights are randomly initialized, which is the same as GraphNAS [10] for a fair comparison. We train each sampled model M with 150 epochs, and the controller update step is 50. For the conditionally progressive search algorithm, we filter hop-dimension combinations with 𝑅(𝑀) set as 0.8, 0.7, 0.78 for Cora, Citeseer, Pubmed datasets, respectively. Each model costs about 25 seconds to get its reward 𝑅(𝑀) and hence it takes about 15 days to get 50,000 models.

Considering computational efficiency, NAS may not be practical for graphs with a large amount of nodes and edges [13]. Therefore, except for the results shown in Sec. 4.1, all the other experiments are conducted with the proposed approximate hop-dimension relation function in Sec. 3.4. In these experiments, our training settings are the same as SGC [39] with a fixed random seed. The learning rate is 0.2 with Adam optimizer and the total number of epochs is 200. All the hyper-parameters are listed in the scripts of our Code.

4.1 Results from Neural Architecture Search

To study the impact of the dimensions among hops, we conduct the NAS on different datasets to find out the optimal hop dimension combinations. There exist a number of NAS approaches for GNN models, including random search (e.g., AGNN [50]), reinforcement learning-based (RL) solution (e.g., GraphNAS [10] and AGNN [50]) and evolutionary algorithm (e.g., Genetic-GNN [32]), wherein the RL-based solutions are more effective than others. Thus, in this work, we follow the same strategy as RL-based solutions to search for appropriate dimensions allocated to each hop.

In particular, we search the hop-dimension combinations of 10 hops on Cora, Citeseer, and Pubmed datasets and show experimental results in Table 1. Compared with existing NAS methods, our NAS method achieves better results with conditional progressive search algorithm on Citeseer and Pubmed datasets, improving over Genetic-GNN by 1.4% and 2.1%, respectively. Meanwhile, we achieve comparable accuracy in the Cora dataset only by considering the hop-dimension combinations. Moreover, compared with w/o cond., we can find 2.6% improvements on conditional progressive search, indicating the effectiveness of this strategy to search optimal hop dimension combinations under a limited search resource. Moreover, the Approx. method show competitive results with NAS-based results, especially on Cora and Citeseer datasets.

Specially, we demonstrate the histogram of the possible dimension assignment for different hops in Figure 5. We can obtain two observations: (i) for low-order neighbors, i.e., when hop is less than 3 in this case, most of the sorted solutions with high accuracy keep the initial feature dimension; (ii) most of the possible dimensions of the hop are only in single digits, which verify the necessity of the proposed conditional strategy to reduce the search space greatly; (iii) The dimensionality tends to be reduced for high-order...
neighbours, and approximating it with exponentially decreasing dimensions occupies a relatively large proportion of the solutions.

Last, the above results serve two purposes: (i) they facilitate and support the design of the proposed approximate hop-dimension relation function; (ii) they validate the effectiveness of the proposed approximate hop-dimension relation function. Accordingly, we could use the approximate relation function with only one parameter to search for proper hop dimensions with comparable performance to the NAS solution.

4.2 Results of High-order Aggregation

We take the combinations of two aggregation framework among nodes with our proposed Ladder-GNN as examples.

For GAT, we explore whether GAT itself can aggregate higher-order neighbors effectively. Hence, we use eight multi-heads and four kinds of channel sizes \{1, 4, 8, 16\} for each head in their self-attention scheme. To aggregate \(K\) hop neighbors, we set two kinds of baselines with a deeper structure (stacking \(K\) layers), named as D-GAT in blue lines and a wider (one layer computes and aggregates multiple-order neighbors) network W-GAT in orange lines. In Figure 6(a), we demonstrate the accuracy of D-GAT and W-GAT with ours in purple line. As we can observe, the D-GAT will suffer from over-smoothing problems suddenly (\(K=4\)), especially for the larger channel size. Moreover, the performance of W-GAT will degrade gradually, due to the over-fitting problems with more parameters. Thus, both of the two aggregations drop their performance as hops increase. On the contrary, the proposed Ladder-GAT is robust to the increasing of hops since the proposed Ladder-GNN can relieve the above problems when aggregating high-order neighbors.

In Figure 6(b), we compare the original GCN and GAT methods with the proposed Ladder-GCN and Ladder-GAT on Citeseer dataset. We can obtain similar observations. As the high-order neighbors are aggregated, GCN and GAT encounter performance degradation, while our framework can boost the performance and relieve the potential over-smoothing issue.

In Figure 6(c), we compare GCN, SGC with Ladder-GCN, where the low-order \(L\) hops are aggregated without dimension reduction and compress the dimensions of hop \(k\) (\(L < k \leq K\)). The purple line is set as an interesting setting: only compress the last \(K_h\) hop to a lower dimension 32 and \(L=K-1\). Under the same horizontal coordinate \(K\) hops, Ladder-GCN achieves consistent improvement compared with both GCN and SGC (in orange line). The reasons behind this result is that the parameters updated in SGC are affected by (i) the decrease in information-to-noise ratios within distant hops and (ii) over-squashing phenomena [3] – information from the exponentially-growing receptive field is compressed by fixed-length node vectors, and causing it is difficult to make \(k_h\) hop neighbors play a role. This observation suggests that compressing the dimension on the last hop can mitigate the over-squashing problem in SGC, which consistently improves the performance of high-order aggregation.

Quantitative Results: We further demonstrate the accuracy among general GNNs, like GCN, GAT and GraphSage on five popular datasets in Table 2. As an effective hop-aware aggregation, we take GCN and GAT as examples to integrate them into our aggregation framework. The results show our methods can boost the performance of GCN and GAT by 4.7% at most, indicating the proposed aggregation is beneficial and robust on different dataset.

![Figure 6: Accuracy comparison of different hop-level aggregation methods of different datasets.](image)

| Method          | Cora | Citeseer | Pubmed | Arxiv | Products |
|-----------------|------|----------|--------|-------|----------|
| GCN [14]        | 81.5 | 70.3     | 79.0   | 71.7  | 75.6     |
| GraphSage [12]  | 81.3 | 70.6     | 75.2   | 71.5  | 78.3     |
| GAT [35]        | 78.9 | 71.2     | 79.0   | 73.6  | 79.5     |
| SGC [39]        | 81.9 | 71.9     | 78.9   | 68.9  | 68.9     |
| Ladder-GCN      | 83.3 | 74.7     | 80.0   | 72.1  | 78.7     |
| Ladder-GAT      | 82.6 | 73.8     | 80.6   | 73.9  | 80.8     |

Efficiency Exploration: A comparison of our Ladder-GCN model with other related works (GCN, SGC, GAT, and MixHop) is shown in the following Table 3. Experimental results are performed on a TITAN-Xp machine. We average ten runs on the Citeseer dataset to obtain the average training time for each epoch, the average test time, and the model accuracy. The settings of the other methods follow the respective papers, and we set \(K = 5\) and \(d = 0.0625\) in Ladder-GCN.

As can be seen from Table 3, the computational time and memory costs of Ladder-GCN are moderately lower than SGC, with higher model accuracy.

4.3 Comparison with Hop-aware GNNs

Our proposed solution focuses on improving the hop-aware GNN representation learning capability when aggregating high-order
neighbors during message passing. Table 4 presents the compare with other hop-aware solutions.

As can be observed, in terms of Top-1 accuracy (%) on {Cora, Citeseer, Pubmed} datasets, the best existing hop-aware models for the three datasets are different, while our simple hop-dimension relation function consistently outperforms them on all datasets, indicating the effectiveness of the proposed solution.

Specifically, our method shows more improvements on Citeseer (improvement by 2.2%). Unlike Cora and Pubmed, Citeseer has a lower graph homophily rate [30], making high-order neighbours more noisy and hard to extract discriminative information. Consequently, the fact that our method can boost relatively more on Citeseer proves our method’s effectiveness in handling noisy graphs.

Table 3: The comparison of efficiency on Citeseer dataset.

| Methods     | GCN       | GAT       | SGC       | MixHop    | Ours     |
|-------------|-----------|-----------|-----------|-----------|----------|
| Accuracy (%)| 70.31±0.17| 71.20±0.72| 71.93±0.53| 73.02±0.06| 74.70±0.34|
| Params.(K)  | 118.73    | 237.52    | 22.22     | 177.89    | 107.55   |
| Test Time (ms)| 5.91±1.02| 525.00±5.54| 1.22±0.40| 4.85±0.30| 2.03±0.51|
| Test Time (ms)| 9.82±0.80| 181.41±0.50| 4.55±0.30| 10.52±2.43| 6.20±0.91|
| Memory Cost (MiB)| 723.00 | 10,565.00 | 753.00 | 789.00 | 779.00 |

Table 4: Comparison with Hop-aware GNNs.

| Method      | Cora | Citeseer | Pubmed |
|-------------|------|----------|--------|
| HighOrder [28] | 76.6 | 64.2 | 75.0 |
| MixHop [2] | 80.5 | 69.8 | 79.3 |
| GB-GNN [29] | 80.8 | 70.8 | 79.2 |
| HWGCN [22] | 81.7 | 70.7 | 79.3 |
| MultiHop [52] | 82.4 | 71.5 | 79.4 |
| AM-GCN [38] | 82.6 | 71.5 | 79.3 |
| N-GCN [1] | 83.0 | 73.1 | 77.6 |
| Ladder-GCN | 83.3 | 74.7 | 80.0 |

4.4 Heterogeneous Graph Representation Learning

Heterogeneous graphs that consist of different types of entities (i.e., nodes) and relations are ubiquitous. In heterogeneous graph representation learning, meta-path-based solutions are proposed to model the semantics of different relations among entities, e.g., Movie-Actor-Movie and Movie-Director-Movie in the IMDB dataset.

In this section, we apply our method to heterogeneous semi-supervised classification on two popular datasets: ACM and IMDB extracted by HAN [37]. For a fair comparison, we follow the experimental settings of HAN [37].

Meta-path based baseline: Heterogeneous graph attention network (HAN) introduces hierarchical attention, including node-level and semantic-level attention. It can learn the importance between a node and its meta-path-based neighbours and the importance of different meta-paths for heterogeneous graph representation learning.

Ladder Aggregation at Semantic Level: Since distinct meta-paths have different contributions to the target node representation, we propose to use Ladder-GNN for semantic-level aggregation. In particular, with the prior knowledge of the ordinal importance of meta-paths (e.g., MDM is more important than MAM for movie type), we can allocate dimensions for them accordingly. Therefore, the hop-dimension relation function is used here for semantics-dimension relations, wherein more/less relevant semantic embeddings get higher/lower dimensions. Again, the only hyper-parameter in our method here is the compression ratio $d$.

Compare Ladder-GNN with Attention-based Aggregation: We use eight kinds of meta-path combinations and compare with GAT$^2$ and HAN. Experimental results on Top-1 accuracy are shown in Table 5. From the results we can observe: (i) distinguishing heterogeneous paths is essential as the performance of GAT is always the worst; (ii) by allocating distinct dimensions for different semantic embeddings, the proposed Ladder-GNN produces better results than state-of-the-art solution (i.e., HAN).

Furthermore, we analyze the semantic attention scores learned in HAN. In Figure 7, we demonstrate three kinds of semantic attention scores between two meta-paths in the ACM dataset from different self-attention channels from node level, e.g., {4, 8, 16} of each head and the head is 4 here. We find that (i) although these three models obtain similar accuracy, the patterns of their attention scores are quite different, and the scores of the best accuracy are also different, for instance, $[PAP=0.823, PSP=0.177]$ for the first model while $[PAP=0.609, PSP=0.391]$ for the second model; (ii) the learnable attention scores are changed by epochs, which can be vulnerable with training strategies; (iii) if only a score is multiplied over the whole semantic features, then both useful and useless information of the features will be scaled simultaneously. Therefore, the information-to-noise ratio will not be changed. Thus, the attention-based methods tend to obtain worse performance. Compared with these methods, our method shows consistent improvements and proves its effectiveness.

4.5 Ablation Study

To analyze the impact of the hop-dimension function in Eq. (5), we conduct experiments by varying two dominant hyper-parameters: the furthest hop $K$, the dimension compression rate $d$ and the aggregation methods among hops. We present the results on Citeseer. In Table 6, the compression rate $d$ varies from 2 to 0.03125 (1/32) as comparison. We can observe that (i) by increasing the furthest

---

1With GAT, we treat the heterogeneous meta-paths as homogeneous edges by summing up the adjacency matrices of different attributes.
which validates the effectiveness of the basic principle of dimension compression when $d < 0.0625$, the reduced dimension is too low to preserve the overall structural information, leading to worse performance in most cases. (iii) the effective rate $d$ is mainly on $0.125, 0.0625$, which can achieve better results for most $K$. If $K = 5$ and $d = 0.0625$, we obtain the best accuracy of $74.7\%$. (iv) note that there are significant improvements with dimension compression comparing to dimension increase ($d = 2$), which validates the effectiveness of the basic principle of dimension compression.

Table 5: Comparison of the accuracy (%) of the two methods on two datasets (Higher values are better). GAT and HAN are re-implemented by us to keep all hyper-parameters as the same. All uses all meta-paths. IMP. is an abbreviation of improvement.

| Dataset | ACM | IMDB |
|---------|-----|------|
| Method  | PAP&PSP | PAP&PTP | PSP&PTP | All | MAM&MMDM | MAM&MYM | MDM&MMDM | All |
| GAT     | 83.5 | 79.7 | 79.7 | 79.7 | 53.0 | 40.9 | 48.9 | 48.2 |
| HAN     | 87.9 | 84.9 | 82.1 | 88.7 | 52.4 | 48.6 | 51.8 | 53.4 |
| Ours    | 89.2 | 86.2 | 84.7 | 89.6 | 58.4 | 50.4 | 53.1 | 55.9 |
| IMP (%) | 1.48 | 1.53 | 3.17 | 1.01 | 11.5 | 3.57 | 2.45 | 4.12 |

Table 6: Comparison of different compression rate $d$ under different furthest hop $K$ of the proposed LADDER-GNN.

| $K$ | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|-----|---|---|---|---|---|---|---|---|
| $d=2$ | 60.7 | 67.6 | 64.8 | 63.0 | 61.2 | 58.9 | 55.8 | 50.2 |
| $d=0.5$ | 65.5 | 71.5 | 72.3 | 72.8 | 73.0 | 73.1 | 73.2 | 73.7 |
| $d=0.25$ | 68.8 | 72.9 | 73.3 | 73.9 | 73.6 | 73.5 | 73.4 | 73.0 |
| $d=0.125$ | 71.0 | 73.5 | 74.1 | 74.0 | 74.2 | 74.3 | 74.0 | 72.8 |
| $d=0.0625$ | 69.3 | 73.1 | 73.6 | 74.7 | 74.3 | 74.3 | 73.8 | 74.2 |
| $d=0.00625$ | 67.2 | 71.4 | 72.7 | 73.0 | 73.2 | 73.8 | 73.5 | 73.3 |

For aggregation method among hops, most existing solutions (e.g., GCN, SGC and GAT) mix the information from multiple hops, while Ladder-GNN disentangles and concatenates the information from multiple hops. To further demonstrate the effectiveness of the proposed method, we conduct an experiment to substitute concatenation with addition in Ladder-GNN. To accommodate the dimensional differences between features from different hops, we use zero paddings to fill those vacant positions before addition. As can be seen from Table 7, concatenating features show consistently better results.

Table 7: Comparison of different compression rate $d$ under different furthest hop $K$ of the proposed LADDER-GNN.

| Methods | Citeseer | Cora | Pubmed |
|---------|----------|------|--------|
| Concatenation | 74.70±0.34 | 83.34±0.38 | 80.08±0.45 |
| Addition | 73.15±0.40 | 80.06±0.51 | 76.20±0.46 |

5 CONCLUSION

In this work, we propose a simple yet effective ladder-style GNN aggregation scheme, namely Ladder-GNN. To be specific, we take a communication perspective for the GNN representation learning problem, which motivates us to separate messages from different hops and assign different dimensions for them before concatenating them to obtain the node representation. The resulted representation facilitates extracting discriminative features effectively compared to exiting solutions. Experimental results on various semi-supervised node classification tasks show that the proposed simple LADDER-GNN solution can achieve state-of-the-art performance on most datasets.
6 APPENDICES
In the appendix, we first detail our experimental settings, including datasets, our preprocessing procedure and other settings in Appendix A. Then, we apply the proposed LADDER-GNN on heterogeneous graph representation learning problem in Appendix B. Next, we discuss the limitations of our work in Appendix D. Finally, we present the broader impact of this work in Appendix E.

A HOMOGENEOUS DATA DESCRIPTION
We use six datasets on semi-supervised node classification: Cora, Citeseer, Pubmed, OGB-Arxiv, and OGB-Products. The specifics of these datasets are listed in Table 8.

Table 8: The statistics of the homogeneous datasets.

| Dataset        | Node | Edge | Feature Size | Train/Valid/Test | Class |
|----------------|------|------|--------------|------------------|-------|
| Cora           | 2708 | 2429 | 1433         | 20 per class/500/1000 | 6     |
| Citeseer       | 3327 | 4712 | 3709         | 6                |       |
| Pubmed         | 19717| 4433 | 500          | 3                |       |
| Flicks         | 7935 | 2197 | 3566         | 180/360/540      | 0     |
| OGB-Arxiv      | 107943| 116623 | 125         | 9                | 40    |
| OGB-Products   | 2449029| 6159140 | 100  | 85/22/90        | 47    |

Cora, Citeseer, Pubmed\textsuperscript{2} [45] are citation network datasets, where the nodes are papers and the edges are citation links. The node attributes are the bag-of-words features of papers, and nodes are divided into different areas. We follow the data preprocessing by SGC\textsuperscript{3}.

OGB-Arxiv\textsuperscript{4} [13] is a citation network between all Computer Science (CS) arXiv papers indexed by MAG [36], where nodes are arxiv papers and edges are citation links. The node feature comes from a 128-dimensional feature vector generated by averaging the embeddings of words in its title and abstract. We follow the data preprocessing by AGD\textsuperscript{5}.

OGB-Products\textsuperscript{6} [13] is an Amazon product co-purchasing network. Nodes represent products sold on Amazon, and edges between two products indicate that the products are purchased together. We follow previous works [8, 13] to process node features and target categories. Specifically, node features are obtained by extracting bag-of-words features from the product descriptions followed by a Principal Component Analysis, and its dimension is reduced to 100. We follow the data preprocessing by SAGN\textsuperscript{7}.

B HETEROGENEOUS DATA DESCRIPTION
The specifics of the heterogeneous datasets are listed in Table 9.

ACM\textsuperscript{8} [37] is sourced from the ACM database, where the nodes are papers (published in KDD, SIGMOD, SIGCOMM, MobifiCOMM, and VLDB). The heterogeneous graph consists of three kinds of meta-path sets \{PAP, PSP, PTP\}. Each node represents the bag-of-words features of keywords, and these nodes are to be categorized as three classes (Database, Wireless Communication, Data Mining). We follow the data preprocessing by HAN\textsuperscript{9}.

IMDB\textsuperscript{10} [37] is a subset of the online IMDB database extracted by HAN [37], where the nodes are movies classified into three types (Action, Comedy, Drama). The heterogeneous graph consists of three kinds of meta-path sets \{MAM, MDM, MYM\}. The features of movies correspond to the elements of plots. We follow the data preprocessing by HAN\textsuperscript{11}.

C COMPARISON WITH EXISTING WORK
Table 10 shows results on LADDER-GNN with other two groups of existing methods, such as general GNNs and GNNs with modified graph structures, in terms of Top-1 accuracy (%) on the most-used datasets. Meanwhile, Table 11 compares LADDER-GNN with existing methods on larger graphs. As a new hop-aware aggregation, our method can achieve the state-of-the-art performance in most cases by improving the hop-aware aggregation method of GCN and GAT, and the proposed method can improve GCN and GAT consistently. Specifically, Ladder-GCN surpasses 1.5% with GNN on Citeseer, 0.5% with DisenGCN on Pubmed, respectively. Our method can still show superiority on Citeseer, showing its effectiveness to handle those graphs with lower information-to-noise ratio. Although there are kinds of variants on GNNs, we verify the potential of a better hop-aware aggregation method. Hence, applying the proposed method to other structure modification and aggregation within one hop will be a future direction.

D LIMITATIONS AND FUTURE WORK
The proposed LADDER-GNN focuses on hop-level aggregation and the dimensionality of high-order neighbors is reduced with our hopdim relation function. The obtain high-quality node representations, LADDER-GNN relies on the quality of the given graphs for learning. That is, for node classification problems, the homogeneity ratio of low-order neighbors is higher than that of high-order neighbors for LADDER-GNN to perform well. While this is true in most cases, it would be better to jointly optimize the structure of the graph and LADDER-GNN, especially for large graphs with a huge amount of nodes and complex relations. We plan to investigate this joint optimization problem in our future work.

E BROADER IMPACT
Graph neural networks (GNNs) have been widely used in various real-world applications ranging from chemo- and bioinformatics to recommendation systems and social network analysis. Although our method can boost the development of the GNNs, we also notice that there should be a negative impact of this technique on protecting personal privacy and security. For instance, customers’ behaviour may be easily predicted based on their historical relevant hobbies, which provides a convenient way for unscrupulous
Table 9: The statistics of the heterogeneous datasets

| Dataset | Relations (A-B) | Node A | Node B | Edge (A-B) | Feature | Training | Validation | Test | Class | Meta Path |
|---------|----------------|--------|--------|------------|---------|----------|------------|------|-------|-----------|
| ACM     | Paper-Author   | 7027   | 56     | 10153      | 1106893 | 476809   | 600        | 300  | 2125  | PAP       |
|         | Paper-Subject  | 4780   | 5841   | 51395      | 12899   | 49316    | 300        | 300  | 2687  | PTP       |
|         | Paper-Year     | 4780   | 5841   | 51395      | 12899   | 49316    | 300        | 300  | 2687  | PTP       |

Table 10: The comparison of General GNNs.

| Method           | Cora | Citeseer | Pubmed |
|------------------|------|----------|--------|
| ChebNet [9]      | 81.2 | 69.8     | 74.4   |
| GCN [14]         | 81.5 | 70.3     | 79.0   |
| GraphSage [12]   | 81.3 | 70.6     | 75.2   |
| GAT* [35]        | 78.9 | 71.2     | 79.0   |
| GIN [42]         | 77.6 | 66.1     | 77.0   |
| JKNet [43]       | 80.2 | 67.6     | 78.1   |
| SGC [39]         | 81.0 | 71.9     | 78.9   |
| APPNP [15]       | 81.8 | 72.6     | 79.8   |
| ALaGCN [40]      | 82.9 | 70.9     | 79.6   |
| GraphHeat [41]   | 83.7 | 72.5     | 80.5   |
| MCN [16]         | 83.5 | 73.3     | 79.3   |
| DisenGCN [27]    | 83.7 | 73.4     | 80.5   |
| FAGCN [6]        | 84.1 | 72.7     | 79.4   |
| SGC [51]         | 83.5 | 73.6     | 80.2   |

Table 11: The comparison with existing methods on Flicker, OGB-Arxiv and OGB-Products datasets.

| Method            | Flicker | OGB-Arxiv | OGB-Products |
|-------------------|---------|-----------|--------------|
| GCN [14]          | 41.1    | 71.7      | 75.6         |
| GraphSage [12]    | 57.4    | 71.5      | 78.3         |
| GAT* [35]         | 46.9    | 73.6      | 79.5         |
| JKNet [43]        | 56.7    | 72.2      | -            |
| SGC [39]          | 67.3    | 68.9      | 68.9         |
| APPNP [15]        | 71.4    | -         | -            |
| MixHop [2]        | 39.6    | -         | -            |
| SGC [51]          | -       | 72.0      | 76.8         |
| Ladder-GCN        | 73.4    | 72.1      | 78.7         |
| Ladder-GAT        | 71.4    | 73.9      | 80.8         |

business people to send spam information. Therefore, we call on researchers to cultivate a responsible AI-ready culture in technology development and avoid the abuse of technologies.