Graph-adaptive Rectified Linear Unit for Graph Neural Networks

Yifei Zhang
yfzhang@cse.cuhk.edu.hk
The Chinese University of Hong Kong
Hong Kong SAR, China

Hao Zhu
allenhaozhu@anu.edu.au
Australian National University and Data61/CSIRO
Canberra, Australia

Ziqiao Meng
zqmeng@cse.cuhk.edu.hk
The Chinese University of Hong Kong
Hong Kong SAR, China

Piotr Koniusz
piotr.koniusz@data61.csiro.au
Data61/CSIRO and Australian National University
Canberra, Australia

Irwin King
king@cse.cuhk.edu.hk
The Chinese University of Hong Kong
Hong Kong SAR, China

ABSTRACT

Graph Neural Networks (GNNs) have achieved remarkable success by extending traditional convolution to learning on non-Euclidean data. The key to the GNNs is adopting the neural message-passing paradigm with two stages: aggregation and update. The current design of GNNs considers the topology information in the aggregation stage. However, in the updating stage, all nodes share the same updating function. The identical updating function treats each node embedding as i.i.d. random variables and thus ignores the implicit relationships between neighborhoods, which limits the capacity of the GNNs. The updating function is usually implemented with a linear transformation followed by a non-linear activation function. To make the updating function topology-aware, we inject the topological information into the non-linear activation function and propose Graph-adaptive Rectified Linear Unit (GReLU), which is a new parametric activation function incorporating the neighborhood information in a novel and efficient way. The parameters of GReLU are obtained from a hyperfunction based on both node features and the corresponding adjacent matrix. To reduce the risk of overfitting and the computational cost, we decompose the hyperfunction as two independent components for nodes and features respectively. We conduct comprehensive experiments to show that our plug-and-play GReLU method is efficient and effective given different GNN backbones and various downstream tasks.

CCS CONCEPTS

- Information systems → Data mining.

KEYWORDS

Graph Neural Networks, Rectified Linear Unit, Graph Representation Learning

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France.
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9996-5/22/04...
DOI: 10.1145/3485447.3512159

1 INTRODUCTION

Deep learning has revolutionized many machine learning tasks in recent years, ranging from computer vision to speech and natural language understanding [21]. The data in these tasks is typically represented in the Euclidean space. However, there is an increasing number of applications where data is generated from non-Euclidean domains and is represented as graphs with complex relationships between objects [5, 9, 22, 24, 35, 41, 42]. By defining the convolution operators on the graph, graph neural networks (GNNs) extend convolution neural networks (CNNs) from the image domain into the graph domain. Thus, we can still adapt many tools from CNNs to GNNs, i.e., ReLU [28], Dropout [32], or residual link [13].

GNN is now capable of mining the characteristics of the graph by adopting the so-called message passing (MP) framework, which iteratively aggregates and updates the node information following the edge path. Embracing the MP as a modeling tool, recent years witnessed an unprecedented achievement in downstream tasks, such as node/graph classification [30, 31] and link prediction [3, 4]. However, proven by many recent works, an inappropriately designed propagation step in MP violates the homophily assumption held by many real-world graphs, consequently causing critical performance degradation [15, 46].

The aforementioned nuisances motivate us to refine the MP and enhance the generalization ability of GNN. One promising line of exploration is to dynamically consider the contribution of each node during MP. Attention mechanisms are a good pointer as they have been explored in [15, 34]. One of the benefits of attention mechanisms is that they let network focus on the most relevant information to propagate by assigning the weight to different nodes accordingly e.g., works [15, 34] show that propagating information by adaptive transformation yields impressive performance across several inductive benchmarks.

However, the current design of adaptive transformation is inefficient. Our reasoning arises from two critical viewpoints. Firstly,
The main contributions of this work are listed below:

i. We incorporate the topology information in the updating function of GNNs by introducing a topology-aware activation function in the updating stage of message passing, which, to our knowledge, is novel.

ii. We propose a well-designed graph adaptive rectified linear unit for graph data, which can be used as a plug-and-play component in various types of spatial and spectral GNNs.

iii. We evaluate the proposed methods on various downstream tasks. We empirically show that the performance of various types of GNNs can be generally improved with the use of our GReLU.

2 RELATED WORKS

2.1 Graph Neural Networks

GNNs can be categorized into spectral or spatial domain type [37]. The spectral domain is exploring GNNs design from the perspective of graph signal processing i.e., the Fourier transform and the eigen-decomposition of the Laplacian matrix [7] are used to define the convolution step. One can extract any frequency to design a graph filter with the theoretical guarantees. However, these models suffer from three main drawbacks: 1) a large amount of computational burden is induced by the eigen-decomposition of the given graph; 2) spatial non-localization; 3) the filter designed for a given graph cannot be applied to other graphs. Spatial GNNs generalize the architecture of GNNs by introducing the message passing mechanism, which is aggregating neighbor information based on some spatial characteristics of the graph such as adjacency matrix, node features, edge features, etc. [11]. Due to its high efficiency and scalability, spatial GNNs have become much more popular in recent works. The most representative model is graph attention network (GAT) [34], which weights the node features of different neighbors by using the attention mechanism in the aggregation step. However, for each node feature, GAT assigns the node a single weight shared across all channels, which ignores the fact that different channels may exhibit different importance.

2.2 Activation Functions

Activation functions have been long studied over the entire history of neural networks. One of the few important milestones is introducing the Rectified Linear Unit (ReLU) [28] into the neural network. Due to its simplicity and non-linearity, ReLU became the standard setting for many successful deep architectures. A series of research has been proposed to improve ReLU. Two variants of ReLU are proposed which adopt non-zero slopes \( \alpha \) for the input less than zero: Absolute value rectification choose \( \alpha = -1 \) [14]; LeakyReLU fixes \( \alpha \) to a small value [38]. PReLU [12] and [2] takes a further step by making the nonzero slope a trainable parameter. Maxout generalizes ReLU further, by dividing the input into groups and outputting the maximum [10]. Since ReLU is non-smooth, some smooth activation functions have been developed to address this issue, such as soft plus [8], ELU [6], SELU [17], Misc [27], Exp [25] and Sigmoid [20]. These rectifiers are all general functions that can be used in different types of neural networks. None of them has a special design for GNNs by considering the topology information.

2.3 Linear Graph Neural Network

In [33, 36, 43–45], the authors proposed simplified GNNs (SGC and S²GC) by removing the activation function and collapsing the weight matrices between consecutive layers. These works hypothesize that the non-linearity between GCN layers is not critical but that the majority of the benefit arises from the topological smoothing. However, removing non-linearity between layers sacrifices the depth efficiency of GNNs. Moreover, the ReLU removed from SGC is not topology-aware. Hence, the rectifiers play a less important role in GNN. As stated in SGC, the topology information is crucial in the aggregation stage to make GNN work, which also motivates us to incorporate the topology information in the updating stage.
3 PRELIMINARIES

3.1 Notations

In this paper, a graph with node features is denoted as \( G = (X, A) \), where \( V \) is the vertex set, \( E \) is the edge set, and \( X \in \mathbb{R}^{N \times C} \) is the feature matrix where \( N = |V| \) is the number of nodes and \( C \) is the number of channels. \( A \in \{0, 1\}^{N \times N} \) denotes the adjacency matrix of \( G \), i.e., the \((i, j)\)-th entry in \( A \), \( a_{ij} \), is 1 if there is an edge between \( v_i \) and \( v_j \). We denote \( N(i) = \{j | a_{ij} = 1\} \) as the set of indexes of neighborhoods of node \( i \). The degree of \( v_i \), denoted as \( d_i \), is the number of edges incident with \( v_i \). For a \( d \)-dimensional vector \( x \in \mathbb{R}^d \), we use \( x_i \) to denote the \( i \)-th entry of \( x \). We use \( x_i \) to denote the row vector of \( X \) and \( x_{ij} \) for the \((i, j)\)-th entry of \( X \).

3.2 Message Passing (MP)

The success of spatial GNNs results from applying message passing (also called neighbor aggregation) between nodes in the graph. In the forward pass of GNNs, the hidden state of node \( i \) is iteratively updated via aggregating the hidden state of node \( j \) from \( i \)'s neighbors through edge \( e_{ij} \). Suppose there are \( l \) iterations, the message passing paradigm can be formalized as:

\[
m^{(l)}_i = \text{AGGREGATE}^{(l)}(\{h^{(l)}_j, e_{ij} | j \in N(i)\}),
\]

\[
h^{(l+1)}_i = \sigma(\Theta)\{h^{(l)}_i, m^{(l)}_i\},
\]

where \( h_i \) and \( h_j \) are hidden states of nodes \( i \) and \( j \), \( m_{ij} \) denotes the message that the node \( i \) receives from \( j \)'s neighbors through edge \( e_{ij} \). The updating function in (2) is modeled as a linear transformation followed by a non-linear activation function \( \sigma(\cdot) \) where \( \Theta \in \mathbb{R}^{C \times (2C)} \) matrix contains learned parameters for the linear transformation and \([\cdot]\) concatenates vectors.

3.3 Parametric ReLU

The vanilla ReLU is a piecewise linear function, \( y = \max(x, 0) \), where \( \max(\cdot) \) is an element-wise function applied between each channel of the input \( x \) and 0. A more generalized way to extend ReLU is to let each channel enjoy different element-wise function:

\[
y_c = \sigma_c(x_c) = \max_{1 \leq k \leq K} (a^k_c x_c + b^k_c),
\]

where \( x_c \) and \( y_c \) are the \( c \)-th channels of \( x \) and \( y \) respectively, \( k \) is the \( k \)-th segmentation in (3), and \( \{(a^k_c, b^k_c)\}_{k=1}^{K} \) is the set of parameters for the parametric ReLU. Note that when \( a^1 = 0, b^1 = 0, a^2 = 1, b^2 = 0 \), then (3) reduces to ReLU, whereas maxout [10] is a special case if \( K = 2 \). Furthermore, instead of learning parameters \((a^k_c, b^k_c)\) directly, approach [2] proposes the hyperfunction \( \omega(x) \) to estimate sample-specific \((a^k_c, b^k_c)\) for different channels.

4 METHOD

We refer to the proposed adaptive rectified linear unit for GNNs as GReLU. The rest of the sections are organized as follows. Firstly, we introduce the parametric form of GReLU by defining two functions: the hyperfunction that generates parameters and activation function for computing the output. Then, we detail the architecture of hyperfunction of GReLU, and we link GReLU with GAT from the perspective of neighborhood weighting. We finally analyze the time complexity of GReLU and compare it with prior works.
We plug GReLU into the WWW ’22, April 25–29, 2022, Virtual Event, Lyon, France. Yifei Zhang, Hao Zhu, Ziqiao Meng, Piotr Koniusz, and Irwin King

We also notice the proposed method enables GReLU to consider the effect of different value ranges. This further distinguishes the difference between channels. GReLU solves this issue by removing the setting (β = 0 and K = 1) when K = 1, there is only one segment in the piecewise function of GReLU, the non-linear max operation reduces to a linear transformation and thus it can be removed. An illustration is shown in Figure 3. The success of GAT is due to selecting the important node information by weighting them differently during aggregation. While GAT is able to consider the difference of individual nodes, it fails to distinguish the difference between channels. GReLU solves this issue by making the parameter adaptive to both nodes and channels:

\[ h'_n = \max_{1 \leq k \leq K} \{ \alpha^k \cdot h_{nk} + \beta^k \} \]

Figure 3 shows plots to the left and middle side are the piecewise functions in GReLU with K = 3 and K = 2. They cut the input value x in different ranges r1, r2, r3 and re-scale the x w.r.t. r. The plot to the right is the weighted function used in GAT: the input is re-scaled regardless of the value range of x.

### 4.3 Understanding GReLU with Weighted Neighborhood and Attention Mechanism

Below, we analyze GReLU through the lens of Weighted Neighborhood and Attention Mechanism. To this end, we first introduce the so-call masked attention adopted in the graph attention network (GAT) [34], which uses the following form:

\[ h'_n = \alpha_n h_n \quad \text{and} \quad \alpha_n = \text{MaskAttention}(H, A), \]

where \( \alpha_n \) is the attention coefficient assigned to node n. Note that (11) has the linear form of GReLU where the bias parameter and max(·, ·) is omitted by setting \( \beta = 0 \) and \( K = 1 \). When \( K = 1 \), there is only one segment in the piecewise function of GReLU, the non-linear max operation reduces to a linear transformation and thus it can be removed. An illustration is shown in Figure 3. The success of GAT is due to selecting the important node information by weighting them differently during aggregation. While GAT is able to consider the difference of individual nodes, it fails to distinguish the difference between channels. GReLU solves this issue by making the parameter adaptive to both nodes and channels:

\[ h'_n = \max_{1 \leq k \leq K} \{ \alpha^k \cdot h_{nk} + \beta^k \} \]

where \( \alpha^k, \beta^k \in \mathbb{R}^{N \times C} \). Thus, instead of learning a single parameter shared across all channels, GReLU learns multiple weights to evaluate the importance of different channels. Moreover, adding max(·) makes GReLU a piecewise linear function that cuts the input values in different ranges and rescales them differently. This further enables GReLU to consider the effect of different value ranges.

In particular, when \( K = 1 \), GReLU reduces to linear transformations that assign a single weight regardless of the range of input.
GReLU performs similar as attention mechanism described in global attention [23, 34]. In such a special case, these transformations are also related to Linear Graph Networks SGC [36] and S2GC [43].

4.4 Computational Complexity.
Below, we analyze the Computational Complexity of GReLU, which is efficient as we decouple the node- and channel-wise parameters. Moreover, the computation cost of GReLU is better compared to the multi-head attention mechanism (MA) adopted in GAT.

4.4.1 Time Complexity. The complexity of GReLU consists of two parts: (i) the complexity of the hyperfunction $\sigma_{\omega(X,A)}(\cdot)$ which is used to infer the parameters of GReLU, (ii) the complexity of activation function $\sigma_{\omega(X,A)}(\cdot)$ which outputs the activations.

i. Hyperfunction includes two components: (1) computation of the channel-wise parameters which consists of computing the average of diffusion output and MLP layer (with $\tanh(\cdot)$), and (2) the computation of node-wise parameters which includes the cost of diffusion and one-layer MLP with softmax. In the channel-wise block, GReLU spends $O(|E|)$ in the diffusion layer and $O(2K)$ in the MLP. In the node-wise block, GReLU spends $O(|E|)$ in diffusion and $O(|V|)$ in MLP with softmax.

ii. Activation function requires that GReLU is applied per channel of each node, which requires $O(|C|\cdot|V|)$ to compute outputs.

Thus, GReLU has the complexity $O(|E| + |C|\cdot|V|)$. As the computation cost of activation function, $O(|C|\cdot|V|)$, is the same for all rectifiers, the extra cost is mainly introduced by the computation of hyperfunction. Due to the light design of hyperfunction: 1) the averaging of diffusion output results in a single vector $e \in \mathbb{R}^C$, 2) the MLP in node block contains only one output dimension and 3) the choice of $K$ is 2 (discussed in Section 5.4), the computation cost of hyperfunction is negligible compared to cost of GNN.

4.4.2 Complexity of GReLU vs. Multi-head Attention Mechanism. GReLU is faster than the multihead attention mechanism used in GAT. Note that MA needs to compute coefficient for $|E|$ edges of $M$ heads, resulting in $O(M|E|)$, whereas GReLU only computes coefficient for $C$ channel and $|V|$ nodes, resulting in $O(|V| + |C|)$ complexity. As $|V| >> |E|$ and $C$ is a relatively small number compared to both $|V|$ and $|E|$, adopting GReLU with a simple backbone (e.g., GCN) is faster than GAT.

5 EXPERIMENTS
Below, we evaluate the proposed method GReLU on two tasks, node classification in Section 5.1 and graph classification in Section 5.2. We show the effectiveness of GReLU by comparing it with different ReLU variants with several GNN backbones. We show the contribution of different modules of GReLU in an ablation study in Section 5.3. The effect of choosing the hyper-parameter $K$ is discussed in Section 5.4. To verify the GReLU acts adaptively, an analysis of parameters is conducted in Section 5.5. We aim to answer the following Research Questions (RQ):

- **RQ1:** Does applying GReLU in the updating step help improve the capacity of GNNs for different downstream tasks?
- **RQ2:** What kind of factors does GReLU benefit from in its design and how do they contribute to performance?

5.1 Node Classification (RQ1)

Datasets. GReLU is evaluated on five real-world datasets:

- Cora, PubMed, CiteSeer [16] are well-known citation network datasets, where nodes represent papers and edges represent their citations, and the nodes are labeled based on the paper topics.
- Flick [26] is an image and video hosting website, where users interact with each other via photo sharing. It is a social network where nodes represent users and edges represent their relationships, and all nodes are divided into 9 classes according to the interest groups of users.
- BlogCatalog[26] is a social network for bloggers and their social relationships from the BlogCatalog website. Note attributes are constructed by the keywords of user profiles, and the labels represent the topic categories provided by the authors, and all nodes are divided into 6 classes.

Baseline. We compare GReLU with different ReLU variants (as shown in Table 1) and choose several representative GNNs as backbone:

- **Chebyshev** [7] is a spectral GNN model utilizing Chebyshev filters.
- **GCN** [16] is a semi-supervised graph convolutional network model which transforms node representations into the spectral domain using the graph Fourier transform.
- **ARMA** [1] is a spectral GNN utilizing ARMA filters.
- **GraphSage** [11] is a spatial GNN model that learns node representations by aggregating information from neighbors.
- **GAT** [34] is a spatial GNN model using the attention mechanism in the aggregate function.
- **APPNP** [18] is a spectral GNN that propagates the message via a fully personalized PageRank scheme.

Experimental Setting. To evaluate our model and compare it with existing methods, we use the semi-supervised setting in the node classification task [16]. We use the well-studied data splitting schema in the previous work [16, 34] to reduce the risk of overfitting. To get more general results, in Cora, CiteSeer and Pubmed, we do not employ the public train and test split [16]. Instead, for each experiment, we randomly choose 20 nodes from each class for training and randomly select 1000 nodes for testing. All baselines are initialized with the parameters suggested by their papers, and we also further carefully tune the parameters to get the optimal performance. For our model, we choose $K = 2$ for the whole experiment. A detailed analysis of the effect of $K$ is in Section 5.4. We use a 2-layer GNN with 16 hidden units for the citation networks (Cora, PubMed, CiteSeer) and 64 units for social networks (Flick, BlogCatalog). In both networks, dropout rate 0.5 and L2-norm regularization are exploited to prevent overfitting. For each combination of ReLUs and GNNs, we run the experiments 10 times with random partitions and report the average results and the best runs. We use classification accuracy to evaluate the performance of different models.
Table 2: The average node classification accuracy (over 10 runs). The training set and test set are randomly divided. The results in parentheses are the best results achieved among the 10 experiments. We omit the results of Cheb in BlogCatalog and Flickr dataset due to the out-of-memory issue during the training step.

| Dataset   | Model | ReLU | LReLU | ELU | PReLU | Maxout | GReLU |
|-----------|-------|------|-------|-----|-------|--------|-------|
| Cora      | Cheb  | 76.5 | 75.5  | 74.9| 75.6 | 75.4 | 78.3 |
|           | GCN   | 79.2 | 79.4  | 78.8| 78.9 | 78.9 | 81.8 |
|           | SAGE  | 78.6 | 78.7  | 77.5| 77.6 | 77.7 | 79.9 |
|           | GAT   | 81.2 | 80.8  | 80.5| 80.4 | 80.4 | 81.5 |
|           | ARMA  | 79.0 | 79.2  | 79.7| 79.3 | 79.8 | 80.1 |
|           | APPNP | 82.0 | 81.0  | 81.5| 81.4 | 81.4 | 82.5 |
| PuMed     | Cheb  | 68.1 | 70.5  | 68.4| 71.5 | 71.3 | 73.4 |
|           | GCN   | 77.6 | 76.8  | 76.8| 77.3 | 77.3 | 78.9 |
|           | SAGE  | 57.5 | 57.3  | 57.4| 57.6 | 57.6 | 66.2 |
|           | GAT   | 77.2 | 78.7  | 78.9| 78.2 | 78.4 | 76.6 |
|           | ARMA  | 76.9 | 76.5  | 77.3| 76.4 | 77.3 | 77.7 |
|           | APPNP | 78.2 | 78.2  | 78.5| 78.0 | 78.6 | 78.9 |
| CiteSeer  | Cheb  | 67.8 | 67.1  | 67.8| 67.0 | 67.1 | 68.1 |
|           | GCN   | 67.7 | 68.4  | 68.3| 67.3 | 67.5 | 68.5 |
|           | SAGE  | 76.1 | 67.3  | 67.8| 68.2 | 68.5 | 69.3 |
|           | GAT   | 66.8 | 66.8  | 68.4| 68.2 | 68.6 | 69.0 |
|           | ARMA  | 67.7 | 68.6  | 67.9| 66.8 | 68.5 | 70.0 |
|           | APPNP | 68.7 | 69.3  | 69.4| 69.5 | 69.6 | 70.0 |
| BlogCatalog | GCN   | 72.1 | 72.6  | 72.6| 71.4 | 71.4 | 73.7 |
|           | SAGE  | 71.9 | 72.2  | 72.1| 72.0 | 71.6 | 73.3 |
|           | GAT   | 41.7 | 46.6  | 69.5| 67.2 | 54.2 | 67.8 |
|           | ARMA  | 72.5 | 72.5  | 77.2| 79.6 | 84.4 | 85.7 |
|           | APPNP | 71.1 | 72.5  | 72.4| 71.7 | 72.8 | 73.8 |
| Flickr    | GCN   | 50.7 | 51.0  | 52.8| 53.0 | 54.0 | 54.4 |
|           | SAGE  | 49.7 | 50.8  | 52.6| 53.2 | 54.0 | 55.3 |
|           | GAT   | 20.2 | 20.3  | 23.8| 32.8 | 30.0 | 33.7 |
|           | ARMA  | 48.4 | 52.0  | 52.2| 45.9 | 52.7 | 56.6 |
|           | APPNP | 51.6 | 51.6  | 53.0| 53.2 | 53.9 | 54.8 |

Observations. The node classification results are reported in Table 2. We have the following observations:

- Compared with all baselines, the proposed GReLU generally achieves the best performance on all datasets in different GNNs. Especially, Compared with ReLU, GReLU achieves maximum improvements of 13.2% on BlogCatalog and 8.1% on Flickr using ARMA backbone. The results demonstrate the effectiveness of GReLU.
- We particularly notice that GReLU with a simple backbone (i.e., GCN) outperforms the GAT with other ReLUs. This indicates that the mask edge attention adopted in GAT is not the optimal solution to weighting the information from the neighbors. As GReLU is both topology- and feature-aware, it can be regarded as a more effective and efficient way to weigh the information over both effective and efficient ways and consider the connections between nodes and channels.

5.2 Graph Classification( RQ1)

Dataset. In the graph classification task, our proposed GReLU is evaluated on two real-world datasets MUTAG [40] and PROTEINS [40] which are common for graph classification tasks.

Baselines. Similarly to the node classification task, we compare GReLU with ReLU variants (Table 1) with various GNN-based pooling methods:

- GCN [16] is a multi-layer gcnn model followed by mean pooling to produce the graph-level representation.
- GraphSage [11] is a message-passing model. The max aggrigator is adopted to obtain the final graph representation.
- GIN [39] is a graph neural network with MLP, which is designed for graph classification.

Experiment Setting. The graph classification is evaluated on two datasets via 10-fold cross-validation. We reserved one fold as a test set and randomly sampled a validation set from the rest of the nine folds. We use the first eight folds to train the models. Hyper-parameter selection is performed for the number of hidden units [16, 32, 64, 128] and the number of layers [2, 3, 4, 5] w.r.t. the validation set. Similarly to the node classification task, we choose $p = 0.01$ for LeakyReLU and $K = 2$ for our model. We report the accuracy with the optimal parameters setting of each model. The classification accuracy is shown in Table 5.

Observations. We observe that GReLU achieves the best results on both datasets, indicating that the graph-level task can benefit
Table 3: Ablation study: different GReLU variants (ABCD) evaluated on the Flickr dataset, the GCN backbone is used.

| Model    | Type          | K     | Hyperfunction            | Activation                                      | Classification Accuracy |
|----------|---------------|-------|--------------------------|------------------------------------------------|-------------------------|
| GReLU-A  | Non-linear    | 2     | \( \alpha_{nc}^{k}, \beta_{nc}^{k} \leftarrow \omega(X, A) \) | \( \max_{1 \leq k \leq K} \{ \alpha_{nc}^{k} x_n + \beta_{nc}^{k} \} \) | 54.4 ± 1.6              |
| GReLU-B  | Non-linear    | 2     | \( \alpha_{nc}^{k}, \beta_{nc}^{k} \leftarrow \omega(X) \) | \( \max_{1 \leq k \leq K} \{ \alpha_{nc}^{k} x_n + \beta_{nc}^{k} \} \) | 53.4 ± 1.8              |
| GReLU-C  | Linear        | 1     | \( \alpha_{nc}^{k}, \beta_{nc}^{k} \leftarrow \omega(X, A) \) | \( \alpha_{nc}^{k} x_n + \beta_{nc}^{k} \) | 53.7 ± 1.4              |
| GReLU-D  | Linear        | 1     | \( \alpha_{nc}^{k}, \beta_{nc}^{k} \leftarrow \omega(X) \) | \( \alpha_{nc}^{k} x_n + \beta_{nc}^{k} \) | 52.2 ± 1.6              |
| ReLU     | Non-linear    | /     | /                         | \( \max\{x_n, 0\} \)                           | 50.7 ± 2.3              |
| SGC      | Linear        | /     | /                         | \( x_n \)                                       | 48.6 ± 1.3              |

Table 4: Ablation study: different GReLU variants (AEFG) are evaluated on the Flickr dataset, the GCN backbone is used.

| Model    | Parameter          | Hyperfunction            | Activation                                      | Classification Accuracy |
|----------|--------------------|--------------------------|------------------------------------------------|-------------------------|
| GReLU-A  | Node/Channel-wise  | \( \alpha_{nc}^{k}, \beta_{nc}^{k} \leftarrow \omega(X, A) \) | \( \max_{1 \leq k \leq K} \{ \alpha_{nc}^{k} x_n + \beta_{nc}^{k} \} \) | 54.4 ± 1.6              |
| GReLU-E  | Node/Channel-wise  | \( \alpha_{nc}^{k}, \beta_{nc}^{k} \leftarrow \omega(X) \) | \( \max_{1 \leq k \leq K} \{ \alpha_{nc}^{k} x_n + \beta_{nc}^{k} \} \) | 54.0 ± 1.4              |
| GReLU-F  | Channel-wise       | \( \alpha_{nc}^{k}, \beta_{nc}^{k} \leftarrow \omega(X, A) \) | \( \max_{1 \leq k \leq K} \{ \alpha_{nc}^{k} x_n + \beta_{nc}^{k} \} \) | 53.0 ± 1.9              |
| GReLU-G  | Node-wise          | \( f_n \leftarrow \omega(X, A) \) | \( \max_{1 \leq K} \{ f_n^{k} x_n \} \) | 53.8 ± 2.1              |

Table 5: Graph classification results.

| Dataset  | Model | GCN     | LReLU    | ELU     | PReLU    | Maxout | GReLU |
|----------|-------|---------|----------|---------|----------|--------|-------|
| PROTEINS | GCN   | 76.0 ± 3.2 | 75.1 ± 2.2 | 76.3 ± 3.0 | 76.3 ± 3.2 | 76.5 ± 3.2 | 76.7 ± 2.8 |
|          | SAGE  | 75.9 ± 3.2 | 75.7 ± 1.2 | 76.5 ± 3.4 | 75.9 ± 3.4 | 76.6 ± 3.2 | 76.9 ± 1.6 |
|          | GIN   | 76.2 ± 2.8 | 76.1 ± 2.9 | 77.4 ± 2.6 | 76.4 ± 3.8 | 76.8 ± 2.7 | 77.8 ± 3.1 |
| MUTAG    | GCN   | 85.6 ± 5.8 | 86.2 ± 6.8 | 85.7 ± 5.5 | 86.6 ± 5.3 | 85.9 ± 5.8 | 87.2 ± 7.0 |
|          | SAGE  | 85.1 ± 7.6 | 84.5 ± 7.7 | 85.1 ± 7.2 | 85.7 ± 7.4 | 86.0 ± 7.6 | 86.7 ± 6.4 |
|          | GIN   | 89.0 ± 6.0 | 89.2 ± 6.2 | 88.9 ± 5.8 | 89.2 ± 6.1 | 88.7 ± 6.0 | 89.5 ± 7.4 |

from the global information extracted by GReLU. Compared to ReLU which is the default setting for most of CNNs, GReLU obtains the maximum improvement of 1.7% with GCN on PROTEINS. For other backbones, GReLU also improves the performance. Note that the non-linearity choice is orthogonal to the choice of pooling i.e., combining GReLU with high-order pooling [19] may yield further improvements on the MUTAG and PROTEINS datasets.

5.3 Ablation Study (RQ2)

Below, we perform the ablation study (Flicker dataset) on GReLU by removing different modules to validate the effectiveness of incorporating the graph topology into a non-linear activation function and the design of hyperfunction of GReLU. To further show the difference between GReLU and previous works, we also compare the variants with SGC. We define four variants as:

- GReLU-A: GReLU with \( K = 2 \) which is a non-linear function that rescales the input value in two different ranges.
- GReLU-B: No topology information due to omitting the adjacency matrix \( A \) in the hyperfunction.
- GReLU-C: No non-linearity from GReLU-A due to setting \( K = 1 \). In this case, GReLU-C is a linear transformation.
- GReLU-D: Both non-linearity and the topology information are removed.

Table 3 shows the difference of GReLU variants and the classification accuracy evaluated on the Flickr dataset. The results of GReLU-A and GReLU-D are 1% and 1.5% better than GReLU-B and GReLU-D, indicating the topology information introduced by the adjacency matrix \( A \) are useful in both linear and non-linear settings. GReLU-A outperforms the GReLU-C, verifying the necessity of adding the non-linearity. Note that GReLU-B and ReLU use adaptive and non-adaptive rectifiers, respectively. GReLU-B boosts the classification accuracy from 50.7% to 53.4% which implies the adaptive property plays a vital role in GReLU.

Based on these observations, we conclude that the major gain stems from three parts: introducing the topological structure, adding non-linearity, and making the rectifier be adaptive. Also, as SGC is the GNN without non-linear activation, it is somewhat related to the GReLU with \( K = 1 \). The major difference between them lies in the GReLU adopting the topology information to reweight the input \( x \) using learnable parameters, whereas SGC uses an identical mapping. We compare SGC with GReLU-C and GReLU-D. Both GReLU variants with or without the adjacency matrix \( A \) are better than the SGC, which again strengthens our conclusion.

We also study the need for the bias parameter \( \beta \) and the use of node- and channel-wise functions. We further define three variants:

- GReLU-E: The parameters \( \beta \) are dropped.
We check if GReLU is adaptive by examining the input and output which confirms the effectiveness of our design.

\[ \alpha \]

\[ \kappa \]

take a controversial effect on its input. Such an effect is similar to monotonically decreasing piecewise functions indicating sparse selector that filters nodes and channels. Moreover, there are this channel of a node are zero. As a consequence, there is no activation for \( \alpha \) indicates the parameters (\( k \), \( \kappa \)) for a particular channel in node \( n \). This shows that GReLU can work as a sparse selector that filters nodes and channels. Moreover, there are monotonically decreasing piecewise functions indicating \( \alpha \) and \( \alpha_2 \) are negative. This is interesting since it will flip the sign and take a controversial effect on its input. Such an effect is similar to the negative part of the concatenated ReLU (CReLU)[29] defined as \( \text{CReLU}(x) = \text{ReLU}(x) \cdot \text{ReLU}(-x) \). CReLU simply makes an identical copy of the linear responses, negates them, concatenates both parts of the activation, and then applies ReLU on both parts. The extra performance is gained by taking ReLU of the negative input \(-x\) which is similar to setting \( \alpha < 0 \).

5.4 The Effect of Parameter \( K \) (RQ3)

Note that \( K \), which determines the number of segments in the non-linear function, needs to be predefined for GReLU. In this section, we discuss the effect of \( K \) in the GReLU function \( \max_{1 \leq x \leq k} (\alpha \cdot x + \beta) \cdot \rho \). We adopt GCN as the backbone model. To show the effect of \( K \), we evaluate GReLU on the node classification task w.r.t. \( K \). We choose \( K \in \{2, 3, 4, 5, 6, 7\} \). Note that \( K = 2 \) is the default setting for the experiments. Figure 5 shows the fluctuation w.r.t. different \( K \) is small. There is no obvious upward trend and a downward trend for all three datasets. The choice of \( K \) is not a significant factor that influences performance. For the sake of saving computations, \( K = 2 \) is recommended.

5.5 Inspecting GRReLU (RQ4)

We check if GReLU is adaptive by examining the input and output over different node attributes. We randomly choose 1,000 sets of parameters from GReLU on the node classification task with the CiteSeer dataset. We visualize the piecewise functions of the GReLU in Figure 4 which shows that they differ, which indicates GReLU is adaptive to different inputs. Some of the piecewise functions of GReLU are monotonically increasing, indicating both \( \alpha_1 \) and \( \alpha_2 \) are positive, which is consistent with other ReLU variants like PReLU and LeakyReLU. We observe that there are functions that are horizontal lines (\( y = 0 \)) and lines with small slopes in GReLU. This indicates the parameters (\( \alpha_1, \alpha_2, \beta_1, \beta_2 \)) for a particular channel of a node are zero. As a consequence, there is no activation for this channel in node \( n \). This shows that GReLU can work as a sparse selector that filters nodes and channels. Moreover, there are monotonically decreasing piecewise functions indicating \( \alpha_1 \) and \( \alpha_2 \) are negative. This is interesting since it will flip the sign and take a controversial effect on its input. Such an effect is similar to the negative part of the concatenated ReLU (CReLU)[29] defined as \( \text{CReLU}(x) = \text{ReLU}(x) \cdot \text{ReLU}(-x) \). CReLU simply makes an identical copy of the linear responses, negates them, concatenates both parts of the activation, and then applies ReLU on both parts. The extra performance is gained by taking ReLU of the negative input \(-x\) which is similar to setting \( \alpha < 0 \).

5.6 Runtime Comparisons

Below, runtimes of training different ReLU variants with 200 epochs on CiteSeer are reported. As shown in Figure 6, the runtime of the GReLU is shorter than that of the Maxout and slightly larger compared to other variants. The extra runtime is mainly caused by the hyperfunction that infers the parameters of GReLU, whereas other baselines (ReLU, leakyReLU, ELU, PReLU) do not use any hyperfunction. As discussed in Section 4, the hyperfunction of GReLU is simple. GReLU incurs a negligible computation cost.

6 CONCLUSIONS

We have proposed a topology adaptive activation function, GReLU, which incorporates the topological information of the graph and incurs a negligible computational cost. The model complexity is controlled by decoupling the parameters from nodes and channels. By making the activation function adaptive to nodes in GNN, the updating functions are more diverse than the typical counterparts in the message passing frameworks. Therefore, GReLU improves the capacity of the GNNs. Although GReLU is designed for the use in the spatial domain with message passing frameworks, GReLU can be regarded as a plug-and-play component in both spatial and spectral GNNs.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable comments. The work described in this paper was partially supported by the National Key Research and Development Program of China (No. 2018AAA0100204) and CUHK 2300174, Collaborative Research Fund (CRF), No. C5026-18GF. We also acknowledge the in-kind support from the Amazon AWS Machine Learning Research Award (MLRA) 2020.
