Inflation Improves Graph Learning

Dongxiao He
Tianjin University
Tianjin, China
hedongxiao@tju.edu.cn

Rui Guo
Tianjin University
Tianjin, China
961941663@qq.com

Xiaobao Wang*
Tianjin University
Tianjin, China
wxbxmt@tju.edu.cn

Di Jin
Tianjin University
Tianjin, China
jindi@tju.edu.cn

Yuxiao Huang
George Washington University
United States
yuxiaohuang@gwu.edu

Wenjun Wang
Tianjin University
Tianjin, China
wjwang@tju.edu.cn

ABSTRACT

Graph neural networks (GNNs) have gained significant success in graph representation learning and have emerged as the go-to approach for many graph-based tasks such as node classification, link prediction, and node clustering. Despite their effectiveness, the performance of graph neural networks (GNNs) is known to decline gradually as the number of layers increases. This attenuation is partly caused by over-smoothing, in which repetitive graph convolution eventually renders node representations identical, and partly by noise propagation, in which poor homogeneity nodes absorb noise when their features are aggregated. In this paper, we find that the feature propagation process of GNNs could be seen as as a Markov chain, analyze the inevitability of the over-smoothing problem, and then use the idea of Markov clustering to propose a novel and general solution, called a graph inflation layer, to simultaneously addressing the above issue by preventing local noise from propagating to the global as depth increases, while retaining the uniqueness of local homogeneity characteristics. By applying the additional inflation layer mentioned, various variants of GCN and other GCN-based models could also be improved. Besides, our method is completely suitable for both graphs with or without features. We evaluated our method on node classification over several real networks. Results show that our model can significantly outperform other methods and have a stable performance with the depth increases.

CCS CONCEPTS

• **Networks** → **Network architectures**.

KEYWORDS

graph neural networks, over-smoothing, inflation

ACM Reference Format:
Dongxiao He, Rui Guo, Xiaobao Wang, Di Jin, Yuxiao Huang, and Wenjun Wang. 2018. Inflation Improves Graph Learning. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/1122445.1122456

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

A large amount of data exists in the form of graph structures, in which a given number of nodes are irregularly connected by edges. Social networks [8], biology networks [15], and molecular [1] are a few examples. Learning on graphs is important not just for analyzing graph data, but also for generic data forms, because graphs provide strong inductive biases that enable relational reasoning and combinatorial generalization [18].

Graph Neural Networks (GNNs), a form of deep neural network that works with graph-structured data and borrows from the convolution operation of Convolutional Neural Networks (CNNs) [9], have gained significant traction in recent years. This heightened attention may be ascribed to two primary factors: the rising prevalence of non-Euclidean data in real-world applications, and the restricted performance of CNNs when dealing with such data.

Advances in the capacity to effectively train very deep neural networks have been critical in boosting the performance in image recognition, language modeling, and a variety of other fields [6, 10]. While GCNs are extremely effective in a wide range of tasks [25], deep GCNs that even perform beyond just a few (2–4) layers, have not typically outperformed shallower GNNs [13]. This flaw is caused by a number of factors, which can be broken down into two primary aspects: **Noise aspect**: when two connected nodes have low homogeneity, noise aggregates into the features of these two nodes. The noise will continue to propagate to the representations of other nodes as the network deepens. Even if the noise appear to be weak, the speed of superimposition increases exponentially as the number of network layers increases. When the network is deep enough, its real features can be distorted. **Oversmoothing aspect**: we found that deep propagation process of the network progressively approaches the smooth distribution of the random walk process, that is, the problem of over-smoothing [13], where a GNN's latent node representations become increasing similar over successive steps of message passing. Once these representations are over-smoothed, adding more steps does not increase expressive capacity, and hence performance does not improve.

Training deep GNNs has been studied extensively and many techniques have been suggested to overcome it [2, 7, 13, 22]. Such as, Rong *et al.* [16] proposed to alleviate oversmoothing through message passing reduction via removing a certain fraction of edges at random from the input graph. Li *et al.* [11] used ResNet concepts

to train deep GCNs using residual and dense connections. However, these studies employ customized solutions that incorporate extra parameters and/or a different network architecture, and they do not take noise into account, and thus do not demonstrate consistent advantages as depth increases.

In this work, we try to train a deep GNNs from the perspective of graph theory. Specifically, we regard the feature propagation process as a Markov chain, analyze the inevitability of the oversmoothing problem, and then use the idea of Markov clustering to propose the inflation layer. To lessen the impact of noise, $\text{GCN}_{\text{w/ inflation}}$ introduces a new convolution strategy that adds an inflation operation in-between intermediate layers. Such an inflation process has the effect of preventing local noise from propagating to the global as depth increases, while retaining the uniqueness of local homogeneity characteristics. The following is a summary of our major contributions.

- We introduce an inflation layer, a novel convolution approach that makes GNNs much more resistant to oversmoothing and noise confusion, allowing deeper models to be trained without compromising performance. Our suggested method is based on an understanding that most GNNs execute a specific type of Markov chain propagation, which makes node features progressively similar to one another. The key idea is to keep local homogeneity characteristics and reduce noise transmission, resulting in distant pairs having less identical features, and lowering noise confusion.
- Inflation is relatively simple to implement and introduces no new parameters. It is simply applied to the output features of each layer, which consist of simple operations, most notably re-sharpening. Inflation, being a basic convolution step between layers, is not exclusive to any GNNs but rather applies widely.
- Through extensive experiments, we show that $\text{GCN}_{\text{w/ inflation}}$, which employs our inflation operation significantly outperforms GCN on the classification task. More importantly, $\text{GCN}_{\text{w/ inflation}}$ avoids performance from degrading dramatically as the number of layers increases, especially on graphs without attributes.

The rest of the paper is organized as follows: Section 2 introduces the preliminaries, followed by the proposed framework in detail in Section 3. In Section 4, we evaluates its performance on real-world datasets. Section 5 gives related works. Finally, we conclude the paper in Section 6.

2 PRELIMINARIES

In this section, we formally define the notion of graph representation that appear later in the paper and the message passing mode of graph convolutional network (GCN).

2.1 Notations

The input graph $\mathcal{G}$ is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes ($|\mathcal{V}| = N$, N is the number of the nodes) and $\mathcal{E}$ is the set of edge. Each node $v_i \in \mathcal{V}$ is represented by a d dimensional feature vector $x_i \in \mathbb{R}^d$ and matrix $X \in \mathbb{R}^{N\times d}$ denotes all node features in which each row represents a corresponding feature vector. The edge set $\mathcal{E}$ can also be represented by an adjacency matrix $A \in \mathbb{R}^{N\times N}$.

Table 1: Notations and Explanations.

Notations	Explanations
$\mathcal{V}$	The set of nodes
$\mathcal{E}$	The set of edges
$\mathcal{N}$	The number of nodes
v_i	The node i
X	The nodes features
A	The adjacency matrix
H	The hidden node representation
$\text{ReLU}(\cdot)$	The Rectified Linear Unit activation functio
$\text{Softmax}(\cdot)$	The normalized exponential function

$A_{i,j}$ is equal to 1 if there is an edge between nodes v_i and v_j or 0 if no edge exists between these two nodes. Thus, a graph can also be represented as $\mathcal{G} = (A, X)$. The detailed table of notations is listed in Table 1.

Moreover, our design and discussion involve the use of some base non-linear functions in neural networks, so we denote the Rectified Linear Unit activation function that outputs the input directly if it is positive and zero else as $\text{ReLU}(\cdot)$ and the softmax function as $\text{Softmax}(\cdot)$, which is showed as:

$$\begin{aligned} \text{ReLU}(\cdot) &= \max(0, \cdot) \\ \text{Softmax}(x_i) &= \frac{\exp(x_i)}{\sum_{i\in\mathcal{V}} \exp(x_i)} \end{aligned} \tag{1}$$

2.2 Graph Convolutional Networks

In recent years, Graph Neural Networks (GNN) has been developed rapidly and has become an effective method for solving graph structured data. Most of the GNN works follow a fixed message passing mode which use a combination of aggregation operation and combination operation to capture the structural information from neighborhoods of node v_i as $N(v_i)$. Generally, we formulate the representation of v_i at the t-th layer as:

$$\begin{aligned} z_i^{(t)} &= \text{COMBINE}^{(t)}(z_i^{(t-1)}, a_i^{(t)}), \\ a_i^{(t)} &= \text{AGGREGATE}^{(t)}(\{z_j^{(t-1)} : v_j \in \mathcal{N}(v_i)\}), \end{aligned} \tag{2}$$

where $z_i^{(t)}$ is the representation of node v_i after the iterative neighborhood aggregation of t layers, $z_i^{(0)} = x_i$, $\text{COMBINE}(\cdot)$ and $\text{AGGREGATE}(\cdot)$ are component functions with learnable parameters. More specifically, the GCN[9] is a representative work at present, which can be formulated as:

$$X^{(t)} = \sigma\left(\hat{A}X^{(t-1)}\Theta^t\right) \tag{3}$$

where Θ^t is a matrix of parameters at layer t, and σ as the non-linear activation function which usually selected as $\text{ReLU}(\cdot)$. $\hat{A}$ is the normalized laplacian matrix which can be calculated as $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$. $\tilde{A} = A + I$ is the adjacency matrix A with adding self-loops and $\tilde{D}_{(i,i)} = \sum_j \tilde{A}_{(i,j)}$ is the diagonal node degree matrix. GCN follows this strategy to pass message among neighbors so that each node can learn a good representation for various tasks. Thanks to the propagation, we can learn graphs in a semi-supervised way

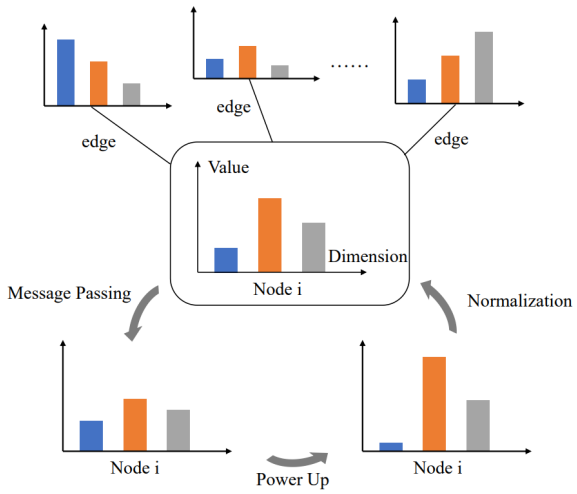

Figure 1: Representation of the embedding change of node i during a message passing procedure with our proposed inflation layer. We first perform a convolution operation on the node i and its neighbors $N(v_i)$ so that embedding accepts neighbor information in all dimensions and becomes smooth. Then the inflation layer consist of power and normalization operation resharps the information in each dimension to gather neighbor information and alleviate the smoothing problem at the same time.

that unlabeled test set recieve useful representation messages from training set with labels.

3 METHODS

In this section, we propose a new convolution strategy, which adds an inflation layer after the original message passing layer, so that the important information in the embedding of each node will not be smoothed with the process of message passing. In this resharp process, the power operation can make the key attributes account for a larger proportion of the entire embedding, and make the insignificant information for graph learning have a smaller value in the embedding. Moreover, we discussed the problem of over-smoothing in message passing from the perspective of embedding, and attributed it to the negative impact of convolution operations on features. Finally, for different GNN model settings, we propose two extensive variants of designing modes.

3.1 Inflation Layer

The inflation layer that we proposed is a simple graph neural network design but has the ability of re-sharpening the embedding of nodes at the same time. The new methods is designed to make the embedding more expressive by operating between message passing layers. The simple idea is to perform the power operation to the embedding vector on each node during the procedure of the network forward propagation. The multiplication in each dimension will make the original larger value more prominent than the representation of other dimensions after the operation. On the contrary, the smaller value will get a weaker expression after going through this inflation layer. Thus, our design can obtain embedding with better expressive ability than plain methods by adding an inflation layer with power operation as the core of design. The choice of the exponent of power should be greater than 1 (usually the choice of 2 can achieve good results). The process of node representation of an inflation layer combined with message passing operation can be represented visually as Figure 1.

For the parameters of the neural network are initialized randomly, when we choose an even number as the exponential parameter of the power operation, the dimensional information of the negative output in the neural network layer will be lost. This will cause the output of the layer not only to lose a lot of expressed information, but also to show completely opposite signals. To solve this problem, we add the softmax non-linear operation to the inflation layer to map the values from negative infinity to positive infinity to the range between 0 and 1. It can keep the relative size trend between the numbers in the original vector and maintain the normalization of embedding in dimensions.

Moreover, the disadvantage of the power operation is that only the relative relationship between the values is considered. This will cause the absolute value between the embedding dimensions of the inflation layer to be too small even if the relative gap between the dimensions is widened. The operation of computers on extremely small number are very rough and therefore the neural network is not sensitive to the embedding after the multiple power up operation, which will increase the difficulty of learning. Thus, we add a simple normalization operation after the above steps to make our layer learning easier. The entire inflation layer can be formalized as:

$$\text{INFLATION}(z_i, e) = \text{Normal}\left(\text{Pow}\left(\text{Softmax}\left(z_i\right)\right), e\right) \quad (4)$$

where $\text{Pow}(\cdot, e)$ means function that multiply an embedding z_i to the power of e, and e is the exponent parameter. What we consider is the normalization between dimensions, so each value is divided by the sum of the values on the current embedding of this node in all dimensions, which is expressed as $\text{Normal}(\cdot)$. In general, by combining the above components to form an inflation layer, we can re-sharp the embedding between the convolutional layers to enhance representation learning of graph neural networks.

3.2 Model Architecture

Based on the inflation layer we proposed above, we can use it as an enhancement layer to improve the capabilities of the original GNN-based model, as long as it uses the message passing mechanism for learning the nodes on the graph. Plain graph neural networks learn useful representations about nodes by convolution on the neighbors in the graph, while this may learn a lot of noise information and affect the prediction of the model, and as the number of layers deepens, there may be the problem that the embedding tends to be smooth. Inflation layer take the simplest and effective way for solving this situation. The designs aiming at embedding can highlight the dimensions that carry important information, and weaken the interference of noise information on model prediction. The GNN model combined with the inflation layer is shown in Figure 2.

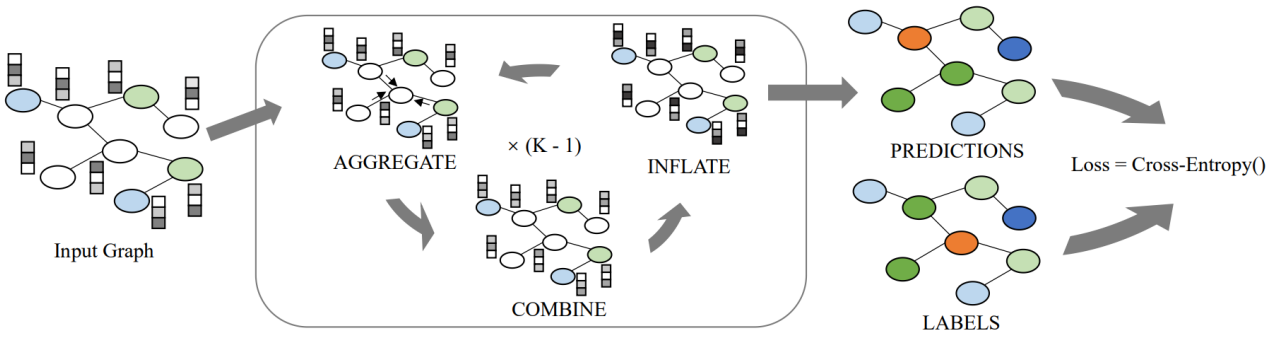

Figure 2: Pure-inflation mode that employ our proposed infaltion layer after each convolution. Each iteration process goes through the aggregation, combination and inflation in sequence. After $K-1$ times of convolution, we predict the output graph and employ the Cross-Entropy as the loss objective.

Different from existing GNN encoder which is formulated in Equation. 2, our proposed stronger GNN model designed to be combined with the inflation layer have three steps: aggregation, combination and inflation. The model with the ability of re-sharpening embeddings can be formulated as:

$$\begin{aligned} a_i^{(t)} &= \text{AGGREGATE}^{(t)}(\{z_j^{(t-1)} : v_j \in \mathcal{N}(v_i)\}), \\ r_i^{(t)} &= \text{COMBINE}^{(t)}(z_i^{(t-1)}, a_i^{(t)}), \\ z_i^{(t)} &= \text{INFLATION}^{(t)}(r_i^{(t)}, e^{(t)}), \end{aligned} \tag{5}$$

In our design of model with inflation layers, in each iteration, every node firstly receives the message from its neighbors in the aggregation stage. Then the combine operations perform the linear transformation on the received information and get the embedding of message passing. Unlike most graph neural network model that enter the next cycle here, our work adds an additional inflation layer to benefit more from each iteration. The model carries out the inflation operation and performs the re-sharping on the obtained output of layers, which not only makes the obtained intermediate representation more expressive but also alleviates the over-smoothing problem. After $K-1$ times iteration of the above steps, the model uses a GNN layer to predict the labels of all nodes, and at the end, we adopt the cross-entropy loss to evaluate over all labeled nodes to update parameters.

It is worth noting that no non-linear activation function layer is used in our model. The subsequent ablation experiment in Section 4.4 will prove that the softmax in our inflation layer does not help the graph learning at all, and may even crash the GCN learning. We use it as a tool to keep our input of layers a constant positive value and the relative relation. The real core design of the our inflation layer is the power funciton.

3.3 Convolution and Inflation

In our opinion, we attribute the problem of over-smoothing to the convolution operation of the graph. In a common graph neural network, we can regard message passing as a convolution process in the spatial domain: neighboring nodes exchange features (outputs of layers) with each other. However, all of these are based on the assumption of homogeneity, that is, adjacent nodes in a graph tend to have the same labels or similar features compared to other nodes that are far away. However, if there is noisy data, which often appear in graph structured datasets from the real world, as the nodes converge with each other, the data in certain dimensions will become smooth due to the presence of noise and the spatial convolution operation of GNNs. If we analyze the node classification task from the perspective of probability discipline, the features and embeddings can be seens as set vectors of probabilities to be classified into each dimension. What is spread between nodes is the probability distribution of classification result. So, when over-smoothing happens, all nodes share the same probability distribution and the classification becomes meaningless. From a high perspective of the model, the result is that more complex parameters and larger neighbor reception range lead to worse classification.

If the output $z^{(t)}$ of the graph neural network after message passing is regarded as a feature of a normal fully connected neural network, the rule is that as the number of network layers increases, the effect tends to gradually get better. However, as the depth of the graph neural network increases, this phenomenon does not appear, indicating that the problem occurs in the convolution operation of feature propagation. We use the idea of random walk to separate the steps of message passing, where the output feature of layer l can be formulated as:

$$H^l = \underbrace{\hat{A}\hat{A}...\hat{A}}_{l} H^0 \tag{6}$$

where H^l represents all nodes feature h_i^l on the graph in iteration l. From right to left of the Equation 6, it can be unterstood as the procedure that we first get the features of the first aggregation, and then use the same topology to propagate which is same as most GNNs. While from left to right, we can obtain the l-step transition probability matrix by calculate A to the power of l and then multiply the attribute matrix as the aggregated feature. Thus, it is easy to find that as the random walk deepens, the transition matrix of layer

l will converge to a fixed distribution because of the calculation of the limit. And if the feature multiplies this transition matrix as topology, then the output result obtained is not discriminative, for the value of each node on the transition matrix is the same. Li [13] thinks this is a special case of laplace smoothing, and no matter which graph laplacian matrix is used for propagation, the deepening of the graph neural network is a process in which the features become uniform. In general, we want to solve the over-smoothing problem from the perspective of embeddings caused by convolution and, furthermore, improve the expressive ability of the model.

Thus, we want to be able to perform deconvolution operations between message passing layers. In the spatial domain, this can be understood as when the convolution operation gathers a lot of neighbor information for a node, the useful information of itself may be affected by the noise and thus be weakened. Deconvolution is the completely opposite process that is expected to enable nodes to enhance their useful information when faced with these noisy information. From another perspective, it means that when a convolution layer smooth the classification probability distribution of the nodes on the graph, the deconvolution can do something that resharp the distribution as a counterwork to message passing. Taking the transition matrix of the random walk into account, the deconvolution operation changes the smoothing process of the continuous self-multiplication of the adjacency matrix to an iterative process consisting of a squaring operation and a probability resharping, which avoids obtaining a stable operation based on the limit when l becomes larger.

We believe that the proposed inflation layer can achieve the effect of deconvolution, for the Sofmax(·) and power operation can make the smoothed embedding sharp again. Inflation layers ensure that the probability distribution of nodes after deep graph neural networks processing can still remain distinguishable clearly, that is, non-smooth.

3.4 Variant of Inflation Modes

It is well established that a resharp mechanism is very important for the graph neural networks. Although our simple inflation layer design which contains Softmax(·) and power up operations could achieve this procedure that is opposite to the convolution, in actual, the model may face some difficulties in gradient descent. Thus we proposed an extended method for designing model called tail-inflation mode.

Pure-Inflation Mode. We name the previously mentioned convolution model which is shown in Figure 2 the pure-inflation mode. Thanks to the resharp mechanism for improving convolution, the pure-inflation model composed of the message passing layer and the inflation layer intersecting each other one by one has stronger expressive ability. While this kind of model uses a large number of nonlinear methods for the realization of the resharp procedure, which cause great obstacles to the parameter learning of the previous layer. Thus, for exploring the over-smoothing problem of deep GCN, Pure-Inflation mode will not play a good role, but will encounter bigger problems than the over-smoothing, which has been shown in the ablation study in Section 4.4. For the shallow and commonly used GNNs, the Pure-Inflation method will not encounter problems, and will take advantage of its expressive ability from re-sharpening to achieve better results.

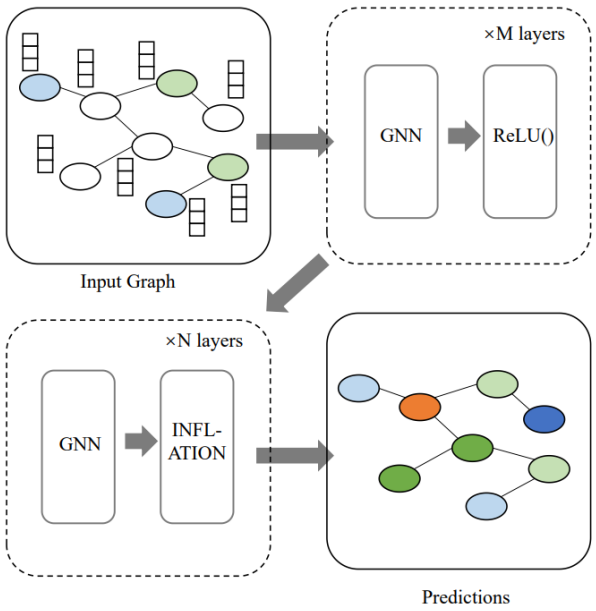

Figure 3: Our proposed tail-inflation mode that use the inflation layer to strengthen the tail of the deep GNN models. The selection of the first M GNN layers can avoid the affect of the non-linear operation of inflation on model learning and the N layer inflation-enhanced GNN can alleviate the problem of overfitting.

Tail-Inflation Mode. Aiming at the shortcomings that cannot be used in deeper networks, we propose an alternative resharp mode to explore over-smoothing problems, called Tail-Inflation mode. We limit the number and position of the inflation layer to control the affect of nonlinear operations on the gradient. As shown in Figure 3, we proposed the mix mode that have both ordinary message passing process and inflation message passing process. Models that first go through the M ordinary GNN layers and then the N resharp layers has no danger that the gradient cannot successfully go backwards for N usually takes 1 or 2 in our experiment. Moreover, the deeper the cumulative number of GNN layers, the more severe the over-smoothing may be. We directly add inflation layers between the tail (deepest) message passing layers, which is the most effective and easiest way from an intuitive point of view.

4 EXPERIMENTS

In this section, we empirically evaluate models combined with our proposed graph inflation layer on the node classification task under different graph data situations (with or without features). We give an introduction about the datasets we used and the experimental settings we apply at the beginning. And then, we list detailed experimental results and analysis based on it.

Table 2: Summary of performance on node classification of graph with features in terms of accuracy in percentage. The better performance of models with different depth is highlighted in boldface.

Dataset	Method	Layers					
		2	3	4	5	6	7
Cora	GCN	**81.50**	**79.90**	78.50	70.90	66.70	51.30
	GCNw/ inflation	81.20	78.80	**81.30**	**76.60**	**70.70**	**63.90**
Citeseer	GCN	70.30	66.10	65.70	57.70	53.80	54.80
	GCNw/ inflation	**71.90**	**66.80**	**67.60**	**61.50**	**55.80**	**55.10**
Pubmed	GCN	79.00	77.60	75.30	75.00	72.30	70.40
	GCNw/ inflation	**79.50**	**78.00**	**77.30**	**76.20**	**75.10**	**71.20**

Table 3: Statistics of datasets used in experiments

Datasets	#Nodes	#Edges	#Features	#Classes
Cora[1]	2,708	5,429	1,433	7
Citeseer[2]	3,327	4,732	3,703	6
Pubmed[3]	19,717	44,338	500	3
Brazil-Airports[4]	131	1,074	-	4
Europe-Airports[4]	399	5,995	-	4
USA-Airports[4]	1,190	13,599	-	4

[1] https://github.com/kimiyoung/planetoid/tree/master/data/ind.cora
[2] https://github.com/kimiyoung/planetoid/tree/master/data/ind.citeseer
[3] https://github.com/kimiyoung/planetoid/tree/master/data/ind.pubmed
[4] https://github.com/snap-stanford/distance-encoding/tree/master/data/node_classification

4.1 Datasets and Experimental Settings

4.1.1 Datasets. To analyze the effectiveness of the inflation layer, we choose three widely used graph learning datasets for the node classification task with graph features, and three datasets for the tasks without features. The datasets are collected from real world networks in different fields and the detailed statistics information is summarized in Table 3.

Citation Networks. We use three citation datasets (Cora, Citeseer and Pubmed) to verify the effectiveness of our inflation layer. Each node in the graph represents a paper and each paper is represented by a vector to characterize the feature of it which only uses 0 and 1 to indicate whether the paper has the word from the dictionary. The existence of edges means that there is an relationship between the paper such as reference and common author. We use the fixed split of training/validating/testing parts of datasets which is first proposed in [23] and the three datasets are used for the supervised node classification.

Airports Networks. We also use three Airports Networks (Europe-Airports, USA-Airports, Brazil-Airports) to observe the performance of our inflation layer on tasks without features. The node represents the airport, and the edge represents the flight route. Labels indicates the status of the current node in the entire network, which is determined by the flow of people at the airport. We use the 1-dimension degree of nodes as the structural feature and the training/validating/testing set is randomly splited as 80%/10%/10%. These three datasets are used only for node classification with supervision.

4.1.2 Experimental Settings. For every experiment, we adopt the torch_geometric code implementation of GCN. All models are implemented by Pytorch 1.7.0 and Pytorch Geometric 1.7.0, and datasets are summarized in the datasets library of Pytorch Geometric. During our experiments, we use a computer server with a 16GB memory NVIDIA P100 GPU as the accelerator.

4.2 Node Classification on Attributed Graphs

In order to show that our inflation layer can really enhance the expression ability of GCN and to a certain extent alleviate the problem of over-smoothing, we design our contrastive experiments where we run the original GCN and the GCN combined with the inflation layer (marked as $\text{GCN}_{\text{w/ inflation}}$ in tables) on three citation graphs with feature. We observe the changes in the accuracy of the two methods by varying the depth of layers. For a fair comparison, we use exactly the same settings for the two methods. Considering the impact of the number of layers on training which will increase the difficulty of learning and make the training time longer, we fix 1000 epochs for training which is much more than the original work, and stop the learning when we observe that the model has stable preformance with recording the stable accuracy. The results on three citation featured graphs are shown in Table 2, and for the choice of design mode, the $\text{GCN}_{\text{w/ inflation}}$ with less than 4 layers choose the pure-inflation mode while the rest four froups take the tail-inflation mode.

We make observations from the results as follows. First, for the three datasets, the accuracy of the models decreases to varying degrees when the number of layers increases. It shows that the over-smoothing problem does exist in GNN models and it is more serious on Cora and Citeseer. Secondly, we can see that our proposed method shows strong performance over GCNs on almost all layers settings. Moreover, the results of 2 and 3 layers shows that even if the number of layers is relatively shallow, the results of using only $\text{GCN}_{\text{w/ inflation}}$ in pure-inflation is advantageous compared to GCN on the Citeseer and Pubmed datasets. This means that the inflation layer we proposed does play a role in graph learning and the convolution model with resharping operation has a stronger expressive ability for features. Furthermore, as the number of layers increases, the over-smoothing problems starts to affect the accuracy of classification. The most obvious exemple is the Cora dataset, with

Table 4: Summary of performance on node classification of graph without features in terms of accuracy in percentage. The better performance of models with different depth is highlighted in boldface.

Layers	Methods	Brazil-Airports	Europe-Airports	USA-Airports
2	GCN	62.86±5.98	54.72±2.23	54.70±3.39
	GCNw/ inflation	66.07±3.49	60.63±6.23	55.18±2.97
3	GCN	63.27±7.63	54.78±4.05	53.97±2.77
	GCNw/ inflation	64.29±5.05	60.31±3.88	54.53±2.73

the classification accuracy of GCN dropped from 81.5% to 51.3%. Although our tail-inflation mode cannot completely solve the over-smoothing problem, it can greatly alleviate the side effects caused by graph convolution which increase by more than 10% on the basis of the original GCN. Thus, the idea of resharping on the output of the intermediate layers can indeed alleviate the over-smoothing graph learning problem to a certain extent.

4.3 Node Classification on Graph Without Attributes

In this section, we perform the models with our inflation layer on graphs without features in the semi-supervised learning pattern. The degrees of nodes are set as the initial structural features of the graph. We continue to use the experimental methods on the three citation networks to see the enhancement effect of inflation on the message transmission of structural features on the network, and we follow the experiment scheme as introduced in [12]. Different from the results in 4.2, we do not observed the occurrence of over-smoothing problems during our experiments on these three airports datasets where the depth of encoder start from two layers and gradually deepen as long as we give enough time and epoch to train. So we only publish the results of two and three layers of GCN and $\text{GCN}_{\text{w/ inflation}}$ for comparison. Results are listed in Table 4.

Here we make our observation on the results that classification without graph features. First of all, the graph learning process that only uses structural information does not show the same phenomenon that as the depth of model layer increases the side effects appear in our experiments. We believe that when the initial information of the node is less, for example, predicting according to the degree of the node, the embeddings will not easily be over-smoothing. Our prediction target and the degree of the nodes are directly related, which is also an important reason for the stable training results regardless of the number of layers. Secondly, although our inflation layer does not shown to alleviate the over-smoothing problem in this part, the resharping ability it possesses has obviously improved the GCN-based model. Operations that make the dimensional gap of the embedding more obvious can indeed improve the learning process of the network with thinking of degrees as attributes, which is proved by the average accuracy advantage of about 3%, which is in general a big improvement on GCNs. More generally, we think that networks with smaller attribute dimensions can benefit more from our resharp mechanism.

Table 5: Summary of performance on node classification of GCN combined with ReLU(), Softmax() and infaltion layer to verify the effectiveness of our design.

Method	GCN+ReLU	GCN+Softmax	GCN+Inflation
Cora	81.50	53.30	81.20
Citeseer	70.30	28.50	71.90
Pubmed	79.00	56.30	79.50
Brazil-Airports	62.86	58.04	66.07
Europe-Airports	54.72	50.41	60.63
USA-Airports	54.70	53.22	55.18

4.4 Ablation Study

We perform ablation studies on our inflation layers and its extensive variant mode across all six datasets.

The Choice of Activation function. Unlike other common graph models, our model does not have a non-linear activation function layer. Our inflation layer can not only play the role of resharping on embeddings during training to alleviate the over-smoothing problem and enhance the expressiveness of the GNN-based model, but also take into account the work of the activation function. Although Softmax($\cdot$) is a kind of activation function and as part of the inflation layer, it cannot improve the ability of GNN as other normal activation function. On the contrary, too many exponential operations will increase the complexity and reduce the efficiency of gradient back-propagation, which is especially obvious on the Citeseer dataset with a decrease of accuracy about 40%.

As shown in Table 5, we manage exepriments on exploring the combined effects of GCN and other auxiliary layers. It can be easily shown that the combination effect of softmax is particularly poor on citation networks that trained with features. But in our work inflation layer, we can properly apply it as a resharp tool and achieve better results. In our proposed design, combined with the power up operation and the normalization step, the inflation layer can replace the non-linear activation function to increase the ability of expressiveness and achieve stronger model performance.

The Choice of Inflation Mode. In this section, we evaluate our proposed variant tail-inflation mode and the original pure-inflation mode to verify our inference just mentioned in 3.4. In order to express more clearly, we pick the deepest and shallowest layers in our experiment. We choose 2-layer model and 7-layer model as a comparison because the deeper layer can show the ability of our mode to alleviate over-smoothing and a relatively shallow model will show the ability of model expression between modes. The result is listed in Table 6. For shallow models, both of the mode

Table 6: Summary of performance on node classification of GCN with different layer design modes to verify to support our analysis in the method section.

Layers	Mode	Cora	Citeseer	Pubmed
2	Pure-Inflation	81.2	71.9	79.5
	Tail-Inflation	77.90	63.40	76.40
7	Pure-Inflation	19.90	21.10	41.50
	Tail-Inflation	63.9	55.1	71.2

perform well and Pure-Inflation model has a little advantage for it has an additional inflation layers to make the model have more discrimination ability. While for the deeper model, Pure-inflation lost the ability to learn parameters because of its complex calculations and power operation design. In general, the commonly used GNN shallow models can achieve competitive performance under the design of the two modes, but for the deep model that needs to mine high-order neighbor information, we need to carefully use the tail-inflation mode for design

5 REALTED WORK

5.1 Graph Neural Networks

The concept of graph neural network (GNN) [17]was first proposed by Scarselli in 2009, which extends the existing neural network for processing data represented in graphs. The goal of graph neural network is to learn the low-dimensional vector representation h_v of each node v. As a powerful graph representation technology based on deep learning, it shows excellent performance and attracts extensive research interest. It can be used for many downstream tasks, such as node classification, node clustering, and link prediction. The rationale behind this is that each node is naturally defined by its own characteristics and its neighborhood. Based on this idea, Defferrard et al. proposed a CNN formula ChebNet[3] based on spectrum theory, which used Chebyshev polynomials to filter the graph signal in the Fourier domain of the graph, providing necessary mathematical background and effective numerical scheme for designing fast local convolution filters on the graph. This technique provides the same linear computational complexity and constant learning complexity as classical CNN, and is applicable to any graph structure. In addition, GCN[9] proposed by Thomas N. Kipf et al., is also a very influential model. This model is based on an effective variant of the convolutional neural network that operates directly on the graph. The problem of over-fitting is alleviated and the performance is improved effectively. Each layer of the above model requires the entire graph as input and has the disadvantages of poor scalability and generalization. GraphSAGE [5] is an important space-based GNN framework that effectively generates node embedding for previously unseen data using node feature information. Transformer[20] network architecture adopts RNN model, which is completely based on attention mechanism and avoids repetition and convolution. For graph structured data, GAT[21] uses the masked self-attention layer to solve the problems existing in the previous graph convolution model. GGNN[14] uses gate cycle unit and modern optimization technology to extend graph neural network of output sequence. GNN mentioned above is effective, but it has significant limitations: excessive smoothness, stability and efficiency can be further improved.

5.2 Oversmoothing Problem

Convolutional neural network (CNN) fails to properly solve the problem of non-Euclidean data, so graph convolutional network (GCNs) is used to construct a graph to represent non-Euclidean data. However, stacking more layers in GCN will lead to the common problem of vanishing gradient, which indicates that back propagation through the network will lead to excessive smoothing. Therefore, the author put the idea of Resnet/Densenet into GCN[11], and also introduced the idea of empty convolution to increase the receptive field, and left a small probability to randomly select aggregation points during training. ZHAO formally explained why over-smooth occurs, suggesting that the convolution of each layer is equivalent to having node representations converge, and proposed a novel normalization layer called PairNorm[24] to address this problem by preventing node representations from becoming too similar. A new model, CS-GNN[7], improves the use of graph information based on the smoothness value of graph, and proposes two smoothness measures to measure the quantity and quality of information obtained from graph data. Kenta generalized the forward propagation of GCN to a specific dynamical system, linking the expressiveness of graph convolutional networks with the topology information inherent in graphs. Rong proposes a DropEdge[16] approach to reduce over-fitting and over-smoothing problems, enabling graph convolutional networks to be more in-depth.

5.3 Other Related Graph Algorithms

The traditional clustering algorithm initializes the clustering and then iterates, but only when the initialization is appropriate, the optimal solution can be obtained. Affinity Propagation (AP)[4] is a new clustering algorithm proposed in Science. The input is the similarity between N data points, which forms the similarity matrix S of N*N (N is the number of data objects). The matrix is used for automatic iterative calculation. Real-value information is exchanged between data points until a high-quality paradigm and corresponding clustering is gradually formed. Its advantage lies in its fast computing speed when dealing with a large number of classes, but it has not yet solved the problem of how to find the optimal clustering result. Markov clustering algorithm (MCL algorithm)[19] provides an interface to algebraic processes defined on random matrices, known as MCI processes. In the calculation process, expansion and inflation are carried out alternately. Expansion strengthens the connection between different regions, while inflation continuously separates the connection between various points. After several iterations, the aggregation phenomenon will gradually appear, so as to achieve the effect of clustering.

6 CONCLUSION

We analyzed the limitations of deep GNN training,regard the feature propagation process of GNN as a Markov chain and suggested the Inflation operation, a novel convolution strategy that improves deep GNN robustness against oversmoothing and noise propagation. Inflation layers ensure that the representations of nodes after deep graph neural networks processing can still remain distinguishable

clearly. Inflation does not need any changes to network architecture or additional parameters, and it could be used in any GNN models. Experiments on real-world classification problems demonstrated the effectiveness of Inflation, where it improves performance when the task benefits from additional layers. More interestingly, networks with smaller attribute dimensions can benefit more from our resharping mechanism. Realizing the benefits of very deep GNNs offers up new possibilities in graph and other structured prediction challenges.

REFERENCES

[1] Gonen Ashkenasy, Reshma Jagasia, Maneesh Yadav, and M Reza Ghadiri. 2004. Design of a directed molecular network. *Proceedings of the National Academy of Sciences* 101, 30 (2004), 10872–10877.

[2] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2020. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 3438–3445.

[3] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems* 29 (2016), 3844–3852.

[4] Brendan J Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *science* 315, 5814 (2007), 972–976.

[5] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.

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

[7] Yifan Hou, Jian Zhang, James Cheng, Kaili Ma, Richard TB Ma, Hongzhi Chen, and Ming-Chang Yang. 2019. Measuring and improving the use of graph information in graph neural networks. In *International Conference on Learning Representations*.

[8] Di Jin, Ziyang Liu, Weihao Li, Dongxiao He, and Weixiong Zhang. 2019. Graph convolutional networks meet markov random fields: Semi-supervised community detection in attribute networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 152–159.

[9] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436–444.

[11] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. 2019. Deepgcns: Can gcns go as deep as cnns?. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9267–9276.

[12] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. 2020. Distance Encoding: Design Provably More Powerful Neural Networks for Graph Representation Learning. arXiv:2009.00142 [cs.LG]

[13] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*.

[14] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493* (2015).

[15] Oliver Mason and Mark Verwoerd. 2007. Graph theory and networks in biology. *IET systems biology* 1, 2 (2007), 89–119.

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

[17] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE transactions on neural networks* 20, 1 (2008), 61–80.

[18] Saul Stahl. 1980. Permutation-partition pairs: a combinatorial generalization of graph embeddings. *Trans. Amer. Math. Soc.* 259, 1 (1980), 129–145.

[19] Stijn vanDongen. 2000. A cluster algorithm for graphs. *Information Systems [INS]* R 0010 (2000).

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[21] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[22] Chaoqi Yang, Ruijie Wang, Shuochao Yao, Shengzhong Liu, and Tarek Abdelzaher. 2020. Revisiting Over-smoothing in Deep GCNs. *arXiv preprint arXiv:2003.13663* (2020).

[23] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. *ArXiv* abs/1603.08861 (2016).

[24] Lingxiao Zhao and Leman Akoglu. 2019. Pairnorm: Tackling oversmoothing in gnns. *arXiv preprint arXiv:1909.12223* (2019).

[25] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81.