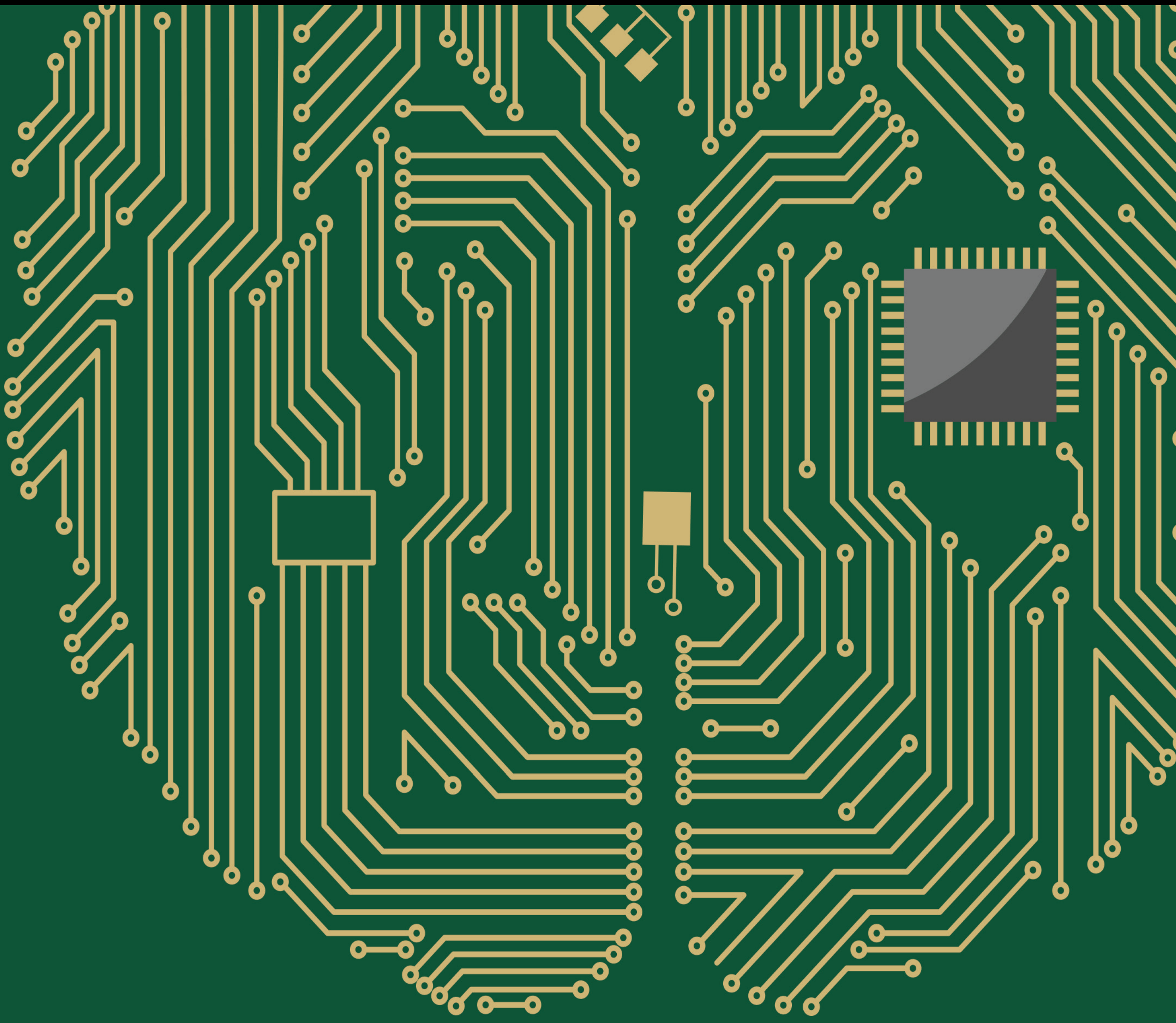
Computational Intelligence and Neuroscience

Using Neuroevolution to Design Neural Networks

Lead Guest Editor: Mauro Castelli
Guest Editors: Eric Medvet, Leonardo Trujillo, and Luca Manzoni



Using Neuroevolution to Design Neural Networks

Computational Intelligence and Neuroscience

Using Neuroevolution to Design Neural Networks

Lead Guest Editor: Mauro Castelli
Guest Editors: Eric Medvet, Leonardo Trujillo, and Luca Manzoni

Contents

Hindawi
Computational Intelligence and Neuroscience
Volume 2020, Article ID 6430627, 11 pages
https://doi.org/10.1155/2020/6430627

Research Article

Change Detection of Remote Sensing Images Based on Attention Mechanism

Long Chen,[1,2] Dezheng Zhang,[1,2] Peng Li,[1,2] and Peng Lv[1,2]

[1]*School of Computer and Communication Engineering, University of Science and Technology Beijing (USTB), No. 30 Xueyuan Road, Haidian District, Beijing 100083, China*
[2]*Beijing Key Laboratory of Knowledge Engineering for Materials Science, Beijing, China*

Correspondence should be addressed to Peng Lv; b1901775@ustb.edu.cn

Received 13 December 2019; Revised 13 February 2020; Accepted 2 June 2020; Published 25 August 2020

Academic Editor: Luca Manzoni

In recent years, image processing methods based on convolutional neural networks (CNNs) have achieved very good results. At the same time, many branch techniques have been proposed to improve accuracy. Aiming at the change detection task of remote sensing images, we propose a new network based on U-Net in this paper. The attention mechanism is cleverly applied in the change detection task, and the data-dependent upsampling (DUpsampling) method is used at the same time, so that the network shows improvement in accuracy, and the calculation amount is greatly reduced. The experimental results show that, in the two-phase images of Yinchuan City, the proposed network has a better antinoise ability and can avoid false detection to a certain extent.

1. Introduction

Change detection in remote sensing images is a critical and challenging task, and its specific work refers to the quantitative analysis of multiple temporal remote sensing images for the same target area, determining the features and scope of surface changes and detecting the changed and unchanged parts [1]. Remote sensing image change detection is utilized to detect illegal buildings, water area supervision, natural disaster assessment, urban planning expansion research, and military reconnaissance [2].

Because of the increasing amount of data from remote sensing images and the increasing demand in this direction, manual comparison and analysis of the change area appear time-consuming and laborious. Due to factors such as seasons and solar illumination, imaging styles of different phases have huge differences [3], which make it difficult to solve the change detection task by computer vision.

Change detection is a unique task for remote sensing image processing, which can be regarded as a dichotomy problem of a region changing or not, as shown in Figure 1. Figure 1(a) shows the remote sensing images of a certain region of Yinchuan City in 2015, Figure 1(b) shows the remote sensing images of this region in 2017, and Figure 1(c) shows the change label of this region, where black indicates that the location has not changed and white indicates that the location has changed. The task of change detection is to identify the changing areas in different phases.

General change detection methods are mainly based on autoencoders and feature extraction through the full connection between neurons [4]. But in fact, change detection can flexibly apply the method of semantic segmentation and extract features by convolution. CNNs have led the field in image processing since AlexNet [5] won the championship in 2012. With the advent of networks such as FCN [6], U-Net [7], and SegNet [8], the baseline effect in this field is getting better and better. However, due to the characteristics of the change detection task, the above-mentioned excellent networks often cannot exert the best results.

In this paper, our task is to solve the change detection in three districts of Yinchuan City. After a detailed analysis of this task, we found that the volume of the changed part is much smaller than that of the unchanged part. There is a greater degree of positive and negative sample imbalance

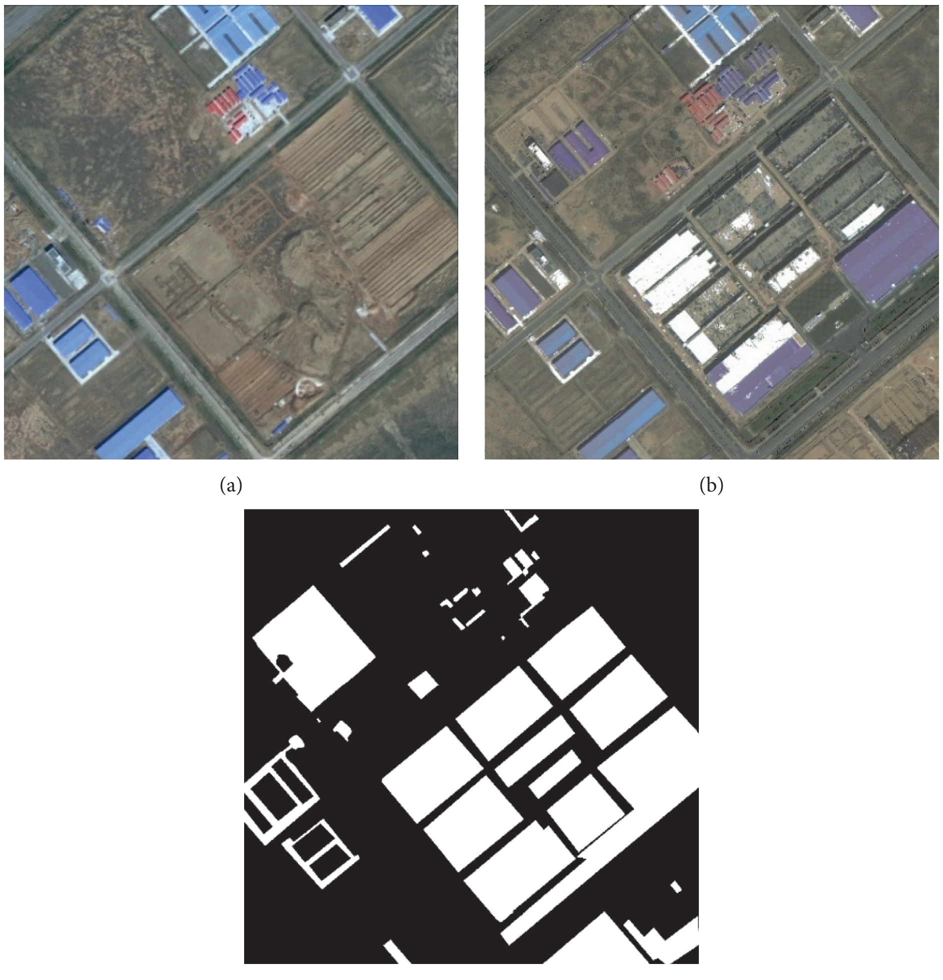

(a) (b) (c)

Figure 1: Example of remote sensing image change detection.

problem. To this end, we propose a new network based on U-Net. The network consists of encoder and decoder. Based on the residual attention model [9], we proposed a new attention mask structure for feature extraction, and a new encoder structure is also proposed in order to better perceive the changing area. By generating an attention mask, the model can pay more attention to the regions with obvious changes and improve the antinoise ability of the model. In the decoder stage, a data-dependent upsampling method (DUpsampling) [10] is used to replace the general upsampling method. The new upsampling method can be applied to smaller-resolution feature maps, which greatly reduces the computational complexity. At the same time, we propose a new loss function for the network, and the initial values of different loss functions are used to balance its impact on network training and reduce the impact of sample imbalance.

2. Related Work

2.1. Convolutional Neural Network. The achievement of today's success in image processing largely depends on the CNNs [11]. The essence of CNNs is a multilayer perceptron. The network structure includes a convolutional layer, a downsampling layer, and a fully connected layer. The reason for its success is local connection and weight sharing method [12]. Reducing the number of weight makes the network easy to optimize and reduces the complexity of the model, which reduces the risk of overfitting. The earliest CNNs are time-delayed neural networks [13] and Lenet-5 [14]. After AlexNet won the champion of ILSVRC [15] in 2012, thanks to the support of GPU computing cluster, deep CNNs such as ZFNet [16], VGGNet [17], and GoogLeNet [18] became the winning algorithm of ILSVRC for many times. But at the same time, CNNs fail to converge with the deepening of network layers. ResNet [19] proposes the mechanism of residual learning, making the network easy to converge while getting deeper. However, the original CNN receptive field is small, and it cannot sense the neighborhood information well. Enlarging the receptive field will lead to a large increase in computing resources. At the same time, CNN's fully connected mode is too redundant and inefficient.

2.2. Attention Mechanism. Mnih et al. [20] confirmed the effectiveness of attention mechanism. Attention is generally classified into two types: one is top-down conscious attention, called focus attention. The other is bottom-up unconscious attention, called saliency-based attention. Focus attention refers to the attention that has a predetermined purpose and focuses on a certain object actively and consciously [21, 22]. Saliency-based attention is also called stimulation-based attention [23]. Wang et al. [9] proposed a method to solve the problem of image classification by using attention residual learning.

2.3. Encoder-Decoder Architectures. Since FCN was proposed, people have been trying to use FCN to improve the accuracy of pixel-level prediction. On the one hand, people start with the atrous convolution [24, 25], which needs more complex operations, and on the other hand, people use encoder-decoder architectures. The most significant feature of encoder-decoder architectures is the ability to complete end-to-end learning. U-Net improves on the FCN framework by connecting codecs with skip connections to improve the effect. SegNet records the location of the maximum value during the maximum pooling operation of the encoder part and then realizes nonlinear upsampling through the corresponding pooling index in the decoder. DeepLab V3 [26] uses the ASPP structure to expand the receptive field, mining context information, and the improved Xception module to reduce the number of parameters and achieve the best effect of the current semantic segmentation network. Tian et al. [27] proposed a data-dependent upsampling method, which enables the encoder to sample down to the bottom layer and improve the accuracy by fusing features of different layers.

2.4. Change Detection. Remote sensing image change detection can greatly improve land utilization and contribute to urban planning and expansion. In the first decade of this century, CNN was rarely used in the field of change detection. Carincotte et al. [28] used a fuzzy hidden Markov chain algorithm to avoid a large number of false changes and missed detections caused by threshold segmentation. Liu et al. [29] used the stacked restricted Boltzmann machine to analyze the differential images between multiphase SAR images and classified the neighborhood features of the two-phase images. By using the deep learning algorithm, the images were classified pixel by pixel to achieve the purpose of change detection. In recent years, CNNs began to be implemented in this field. Desclee et al. [30] proposed an object-oriented forest vegetation change detection method, which firstly segments multitemporal high-resolution remote sensing images and then, based on the hypothesis chi-square test, identifies outliers of statistical differences in reflectivity and marks corresponding objects as changes. Qing et al. [31] applied Faster R-CNN to this field, greatly reducing the false changes of detection results. Ma et al. proposed a network based on multigrained cascade forest and multiscale fusion, so that the network can select image blocks of different sizes as input, thereby learning more image features [32]. Dong et al. designed a “Siamese samples” convolutional neural network to learn the semantic difference between changed and unchanged pixels [33].

The change detection in Yinchuan area includes many different landforms. Considering the particularity of this region, our method adds the attention module in the feature extraction stage and uses U-Net’s skip connection to further reduce the loss of information from upsampling and downsampling, it also uses DUpsampling to accelerate the upsampling process, while facilitating subsequent feature fusion. As a result, our model can avoid many false detections, while ensuring accuracy without occupying too many computing resources.

3. Our Approach

In this section, we first introduce the network we proposed and then elaborate on each functional module of the network, and we also propose a new loss function based on the problem we meet to improve the accuracy of the model. It is noteworthy that, different from the current mainstream of change detection method based on the convolution neural network, we improve the residual attention mechanism, and we proposed a new way to generate attention mask and apply the mask to change detection task. At the same time, the DUpsampling method is used to reduce the loss of computing resources. Finally, we noticed the impact of the initial size of the loss function on the network performance and proposed a new loss function, which achieved good results.

3.1. Network Architecture. In this paper, we propose a new network based on U-Net, as shown in Figure 2. Like U-Net, the network is divided into encoder and decoder. The encoder downsamples the input to extract the features, while the decoder upsamples the input to restore the resolution. The network uses a six-channel matrix superimposed on two pictures of different phases in the same area as input to the encoder. The encoder takes ResNet50 as the trunk and adds the attention modules. The attention module consists of trunk branch and mask branch, which perform feature extraction and mask generation on the input, respectively. The attention mask subtracts the three channels of two RGB images as the input and outputs a feature map with attention weight to highlight the key areas of feature extraction. Then residuals are performed on feature maps of mask branch and trunk branch. The DUpsampling method is used to replace the conventional bilinear upsampling on the decoder, which avoids the computation and memory footprint caused by reducing step size (such as DeepLab V3) of the decoder. The decoder maintains the original network structure of U-Net. There are skip connections between the encoder and the decoder to combine the features of the corresponding encoding during decoding.

3.2. Functional Modules

3.2.1. Encoder. In order to improve the accuracy of the network, different from the traditional autoencoder network, we choose to deepen the network depth, thus

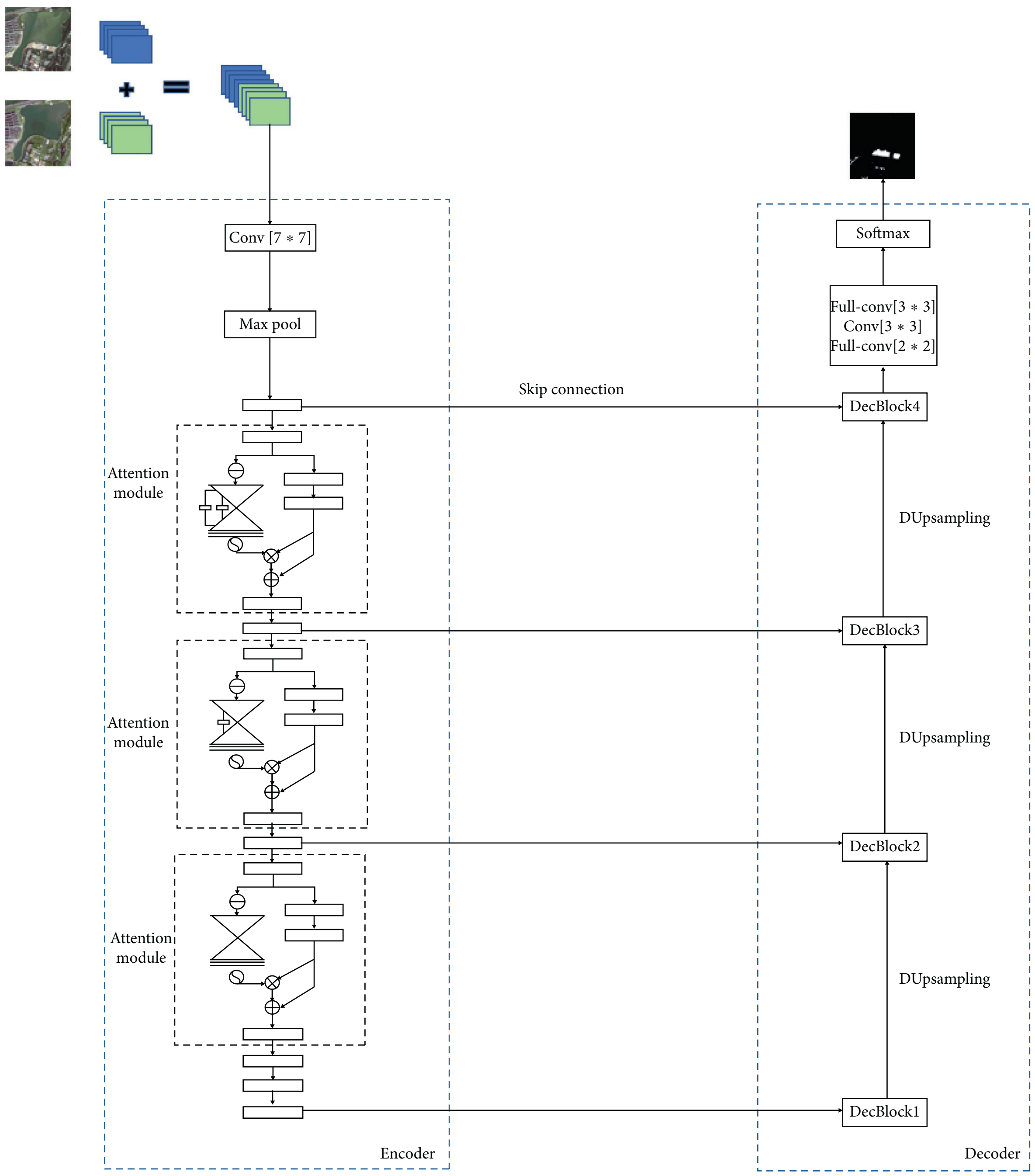


Figure 2: The framework of our proposed network.

introducing ResNet50. Its unique residual learning method can avoid the problem of unable to converge due to gradient explosion or gradient disappearance in deep network. In the encoder stage, ResNet50 as the backbone, and the residual attention mechanism is added to form a new encoder structure to guide the network to focus on areas with significant changes and improve the network's antinoise ability. The encoder includes three attention modules as shown in Figure 3. Each attention module is divided into mask branch and trunk branch. Trunk branch performs feature extraction just like normal convolutional neural networks. Mask branch is responsible for generating attention weights for input features, and finally, residuals are performed on feature maps of mask branch and trunk branch. The formula is as follows:

$$H_{i,c}(x) = \left(1 + M_{i,c}(x)\right) * F_{i,c}(x), \qquad (1)$$

where $F_{i,c}(x)$ represents the features generated by the deep convolutional network, $M_{i,c}(x)$ represents the output of the mask branch, and the value range of $M_{i,c}(x)$ is $[0, 1]$. When it approaches 0, $H_{i,c}(x)$is approximately the original feature$F_{i,c}(x)$. The detailed structure is illustrated in Table 1.

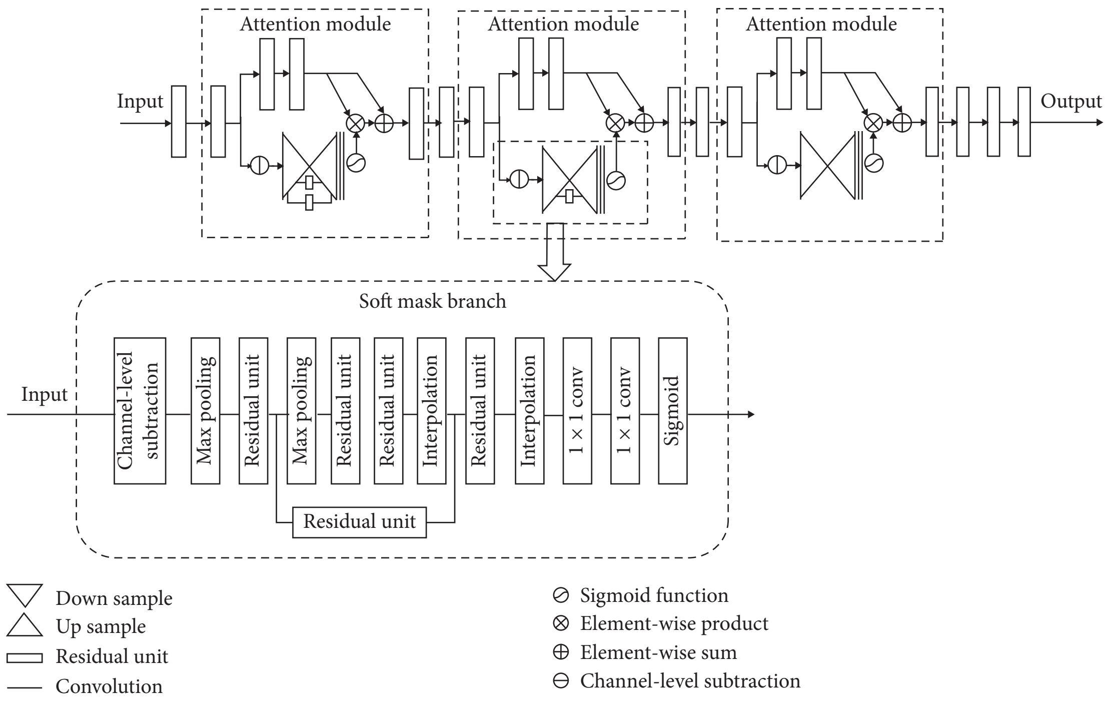


FIGURE 3: The framework with our proposed encoder which includes three attention modules and some residual units.

3.2.2. Attention Module. Because change detection requires comparing information of different phases in the same region, we propose a new architecture of the attention modules, which is different from semantic segmentation task when generating attention weight. In the mask branch, we subtract each channel of the original two feature maps to generate a new three-channel feature map. This step is called the channel-level subtraction. The new feature map is used to generate the attention mask and then put it into the mask branch for upsampling and downsampling and convolution, and the purpose of this step is to downsample to low resolution so that we can get strong semantic information. Each mask branch has a different number of residual units between upsampling and downsampling as skip connections to capture attention information at different scales. Each time pass the residual unit connected to the attention modules, and the size of the feature map is reduced. The architecture is displayed in Table 1.

TABLE 1: The architecture details for our encoder.

Layer	Output size	Encoder
Conv1	512×512	7×7, 64, stride 2
Max pooling	256×256	3×3 stride 2
Residual unit	256×256	$\begin{pmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{pmatrix} \times 1$
Attention module	256×256	Attention $\times 1$
Residual unit	128×128	$\begin{pmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{pmatrix} \times 1$
Attention module	128×128	Attention $\times 1$
Residual unit	64×64	$\begin{pmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{pmatrix} \times 1$
Attention module	64×64	Attention $\times 1$
Residual unit	32×32	$\begin{pmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{pmatrix} \times 3$

3.2.3. Decoder. We keep the decoder structure of U-Net. The proposed network uses DUpsampling in the upsampling phase to replace the original bilinear upsampling procedure. The bilinear upsampling method does not take into account the correlation between the predicted pixels. DUsampling uses the redundancy of segmentation labels to produce accurate segmentation results through the rough features generated by the encoder. And the encoder structure does not need to continue to excessively reduce the resolution of the feature map, thereby reducing the calculation time and memory usage. An important discovery is that the label of images is not independently and uniformly distributed, and the structural information it contains is related, so the label can be compressed without causing too much loss. So we compress the label first and split label into multiple grids, and each grid size is $t * t$ (t is the image size ratio, such as 16 and 32), and then we reshape the content of each grid into a vector v, then compress v into x, and stack x to get compressed labels. Formally, we have

$$x = Pv, \tag{2}$$

$$\widetilde{v} = Wx. \tag{3}$$

Linearly map v to x through P, and W is the inverse mapping matrix, which is the reconstruction matrix. Through the following formula to minimize the reconstruction error and optimize through SGD iteration, PCA

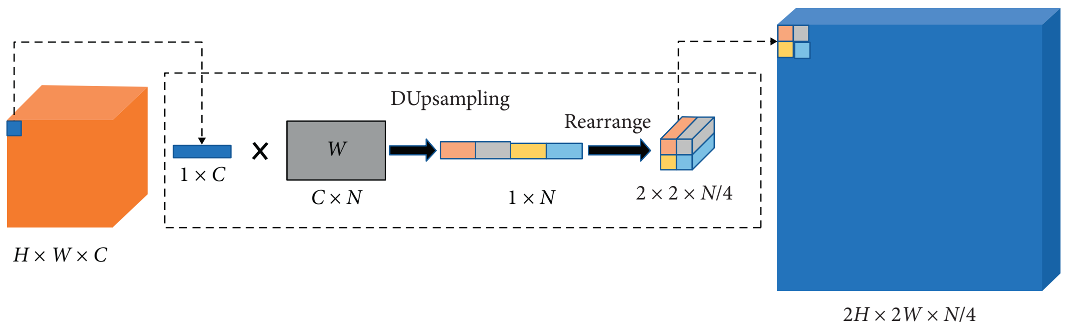


Figure 4: The specific process of DUpsampling. W is the reconstruction matrix, and there are three DUpsamplings in our network.

[34] can be used to obtain the closed solutions P and W. Formally,

$$P^*, W^* = \underset{P,W}{\arg\min} \sum_{v} v - \tilde{v}^2 = \underset{P,W}{\arg\min} \sum_{v} v - WPv^2. \quad (4)$$

When we get the reconstructed matrix W, W is the parameter of the convolution kernel, which can complete the upsampling procedure, as shown in Figure 4.

At the same time, adaptive temperature softmax is introduced because the DUpsampling method may be calculated based on the one-hot label, so that the probability distribution is relatively smooth, resulting in too slow or even difficult convergence of loss in training [35], as shown in the following equation:

$$\text{sotfmax}\, z_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_i/T)}. \quad (5)$$

T can be learned automatically by the backpropagation algorithm without tuning. There is a skip connection between the encoder and the decoder so that the decoder can obtain the feature information in the encoder and reduce the loss of information in the decoding procedure. The decoder part contains four decoders, whose input includes the encoder map from skipping connection and the output of the previous layer. Through the upsampling procedure, the size of the feature map is doubled. Finally, two full convolutional layers and one convolutional layer are added, so that the size is enlarged again to achieve the effect of end-to-end training, as shown in Figure 2.

3.2.4. Loss. The two images of the change detection mission in the three districts of Yinchuan were taken in 2015 and 2017. When we look at the two phases of a total of 924 images and found that the volume of the changed part is much smaller than that of the unchanged part, change detection is essentially a binary classification problem, namely, one regional changes or not. There is a greater degree of positive and negative samples by imbalance problem. Therefore, we introduced the focal loss function [36], which can effectively solve the imbalance of positive and negative samples, as part of the loss function. Formally,

$$FL(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log(p_t), \quad (6)$$

where p_t is the classification probability of different categories and γ and α_t are fixed values. Here, $\gamma = 2$ and $\alpha_t = 0.25$. Combined with the binary crossover entropy loss function L_{BCE}, which is commonly used in this task, combining the above two loss functions, we proposed a new loss function. First of all, we noticed that different types of loss functions have different initial values. Therefore, if we simply weighted the two loss functions, the loss function with a large initial value would dominate the loss function with a small initial value. We use the initial values of the two loss functions obtained in the first iteration to balance:

$$\text{loss} = \frac{c_1}{c_1 + c_2} FL(p_t) + \frac{c_2}{c_1 + c_2} L_{BCE}, \quad (7)$$

where c_1 is the initial value of L_{BCE} and c_2 is the initial value of focal loss. Experiments show that c_1 is much larger than c_2, often several times larger than c_2, so that different loss functions will not distinguish the primary and secondary relationship due to different sizes. And then we add the two loss functions together to get the final loss, and with α approximating 0.3, our network will achieve the best results. The loss function can be formulated as follows:

$$\text{loss} = \alpha \frac{c_1}{c_1 + c_2} FL(p_t) + (1 - \alpha) \frac{c_2}{c_1 + c_2} L_{BCE}. \quad (8)$$

4. Experiments

In this section, we first explain the datasets we used and the criteria we used to evaluate the effect of the method and then introduce the relevant experimental details. At the same time, we performed ablation experiments on the proposed loss function, verified the effectiveness of our loss function, and found the optimal value of α. Finally, we compare the effect of the current mainstream network with our network.

4.1. Datasets. In our experiment, the datasets came from Gaofen-2 (Gf-2) satellite including Xixia District, Xingqing District, and Jinfeng District of Yinchuan City, known as Yinchuan three districts. The ground resolution of Gf-2 is 1 meter, indicating that per pixel represents one square meter. The image has four channels, namely, RGB channels and near-infrared channel. Since the year 2015 image has only RGB channels, we take RGB channels in both phases. The images we used have been irradiated and registered in absolute terms, but not in relative terms. Therefore, we first performed histogram matching on the image to reduce the radiation difference between the two images, as shown in Figure 5.

Due to the particularity of change detection, we need to select the original datasets and eliminate the regions with no change or those with little change. Finally, 924 images from Gf-2 satellite were selected, covering different areas such as towns, agriculture, and industrial areas. Each image size was $512 * 512$.The datasets were divided into training set, verification set, and test set at 7 : 2 : 1, with 647, 184, and 92 pictures, respectively. Because of the small amount of dataset, U-Net, which has relatively loose requirements on dataset size, was chosen as the basic framework at the beginning of designing the network.

4.2. Evaluation Criteria. In our experiment, accuracy and $F1$ value are used as the accuracy evaluation standard, and FLOPS is used as the efficiency evaluation standard. The calculation formula of accuracy is as follows:

$$\text{accuracy} = \frac{(\text{TP} + \text{TN})}{(\text{P} + \text{N})}, \tag{9}$$

where TP is the number of pixels with a positive detection number representing actual changes that are correctly detected and FP is the number of pixels with false detection number representing actual unchanged that are erroneously detected as changes. TP and FP correspond to TN and FN. TN is the number of pixels that have not actually changed but detected as changed and FN is the number of pixels that have actually changed but detected as not changed. P = TP + FN and N=FP + TN. The calculation formulas of precision and recall are as follows:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{10}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{11}$$

The precision represents the proportion of correctly detected change areas in the predicted results. The higher the precision, the less the false changes and noise, and recall represents the proportion of correctly detected change areas in the actual change areas. The higher the recall, the better the coverage of change detection results to the true change results. On the basis of these two important indicators, we calculated the weighted harmonic average of precision and recall. As a comprehensive indicator, $F1$ value is used here:

$$F1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \tag{12}$$

FLOPS is an abbreviation of floating-point operations per second. It is often used to estimate the performance of a network.

4.3. Experimental Details. There are a number of areas in our datasets that have not changed, so we filter the datasets. The size of each picture is $512 * 512$, which is about 250,000 pixels. We consider pictures with a total number of changed pixels less than 1000 as unchanged and removed. In order to enhance the generalization of the network, after histogram matching of the images and reducing the imaging difference, we carry out random geometric transformation and color adjustment of the images in a small range and normalize each channel value of the processed remote sensing images, so that the network could rapidly converge. We use the Adam optimizer [37] to train 1000 epochs on the network, and the initial learning rate is designed to be $1e-3$. After each epoch, the learning rate dropped and α in the loss function is set to 0.3. We observe that our networks generally converged after 200 epochs.

4.4. Loss Function Ablation Experiments. To prove the effectiveness of our loss function, we performed two experiments in this section. First, use formulas (7) and (13) as loss functions, respectively, to compare the effect of initial value balance on experimental results:

$$\text{loss} = FL(p_t) + L_{BCE}, \tag{13}$$

As mentioned above, the initial value of L_{BCE} is much larger than the focal loss, so during the training process, L_{BCE} will occupy the dominant position. The datasets are mentioned in Section 4.1. The experimental details are mentioned in Section 4.3. The initial learning rate is set to $1e$–3. The Adam optimizer is used to train 1000 epochs on multiple networks. The experimental results are shown in Table 2. At the same time, the network has significantly improved the convergence speed after the initial value balance.

After confirming the effectiveness of the initial value balance, we began to explore the optimal value of α. When $\alpha = 0$, it means that only L_{BCE} works, and when $\alpha = 1$, only focal loss works. The results of the comparative experiments are as follows.

It can be seen from Figure 6 and Table 3 that, for our problem, when $\alpha = 0.3$, the network has the best performance. The reason may be that the imbalance of positive and negative samples was improved after the introduction of residual attention mechanism, so the effect of focal loss was not optimal. In the following experiments, we set α to 0.3 by default and compared it with other networks.

4.5. Experimental Results. We select three mainstream networks to compare with our network on our datasets (mentioned in Section 4.1), including the stack autoencoder based on deep confidence network [38], U-Net, and PSPNet [39]. All networks are trained and tested on our datasets without using pretrained models. As shown in Figure 7, from left to right are the year 2015 images, the year 2017 images, attention mask, labels, and the results of different networks. The images on display include different scenes of cities, towns, fields, and bare grounds. As shown in the figure, our model is superior to other models in most cases. Table 4 shows the performance of the above networks and ours on $F1$ value, recall, and accuracy. As shown in the table,

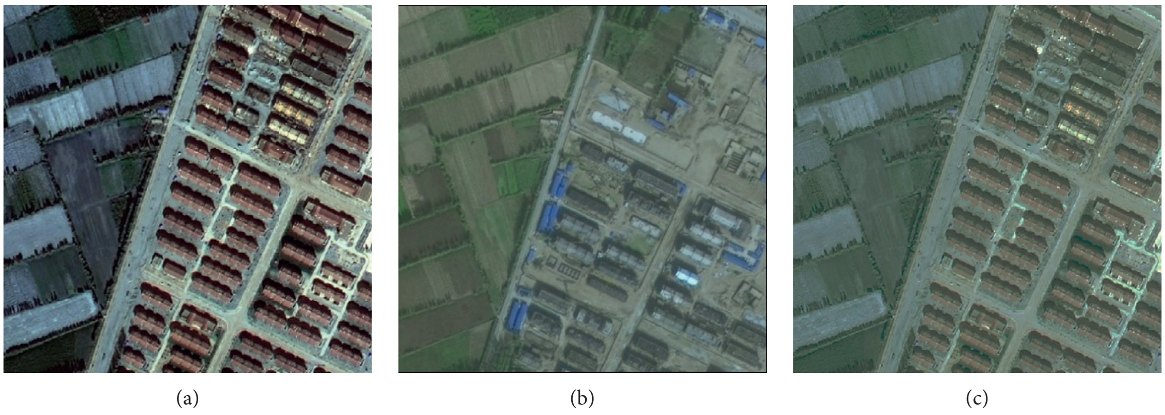


Figure 5: Effect before and after histogram matching. (a) The original images of 2017, (b) the original images of 2015, and (c) the processed image of 2017.

Table 2: Comparison of the effects of two loss functions on the network.

Method	$F1$	Recall	Pixel accuracy	Convergence epoch
Formula (7)	**0.615**	**0.608**	**0.693**	**340**
Formula (13)	0.575	0.554	0.574	570

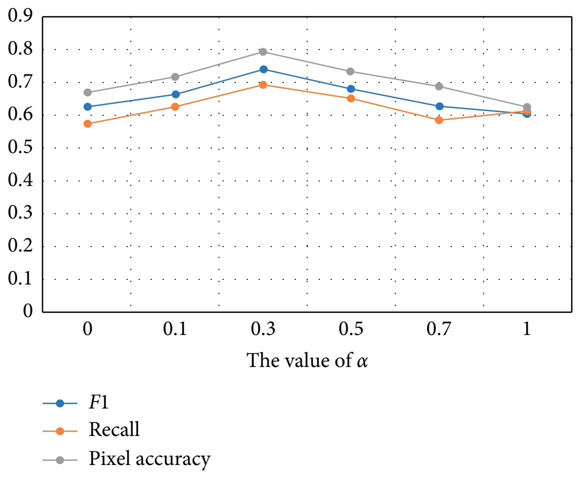


Figure 6: The effect of different values of α on the experimental results. It proves that our method has an optimal solution when $\alpha = 0.3$.

Table 3: Comparison of $F1$, recall values, and pixel accuracy values with different values of α.

α	$F1$	Recall	Pixel accuracy
0	0.628	0.576	0.672
0.1	0.666	0.629	0.717
0.3	**0.743**	**0.697**	**0.796**
0.5	0.686	0.653	0.735
0.7	0.626	0.584	0.694
1	0.607	0.617	0.627

the networks based on the convolution are superior to the stack autoencoder in our datasets, and PSPNet has a strong ability of context information so that it performs better than ordinary U-Net, and our network is 1.9% higher than PSPNet in $F1$ and 5.1% higher in accuracy. The performance of $F1$ and accuracy verifies that our network and the proposed loss functions are effective. At the same time, the FLOPS value of our network is 4.5×10^9, which is far lower than PSPNet and U-Net. It shows that our method has high efficiency while ensuring accuracy and reduces waste of computing resources. It is worth mentioning that in the attention modules, our network will subtract the corresponding channels of input to generate the attention mask. In the figure, we will visualize the attention mask, and the attention mask will optimize the network training process, thus greatly improving the antinoise ability of our network. Thanks to the attention mask, our network can avoid many false detections, such as the third set of pictures. We analyzed in detail that our task for two-phase imaging style difference is huge, so we need to do histogram match in the pretreatment stage, but after the match, color distortion will happen, for example, the third group pictures had dark green fields into a dark purple, which greatly affected the judgment of the network, excellent network like PSPNet is also unable to avoid this problem. In the fifth group pictures, we can see that the attention mask divides the left variable area into two parts, and our network can make correct judgments, while most other networks are unable to. We also observe some disadvantages from the experiment, such as the fourth group experiment, mask only focuses on the middle of the road section and ignores the rest of the change, so the module cannot detect the change of the upper left corner buildings like PSPNet; hence, generating accurate attention mask is the key to the success of our network. At the same time, because we adopt the DUpsampling method, different from the ordinary method, the training speed will be very fast. This experiment only took about 30 hours to train on the latest GPU. At the same time, it is also convenient for feature fusion of different output layers [26] to improve the accuracy. Experiments show that the accuracy of our proposed network is greatly improved compared with the baseline. In complex scenes such as densely built towns, thanks to the attention mechanism, our network can reduce false detection and missed detection and show a better effect.

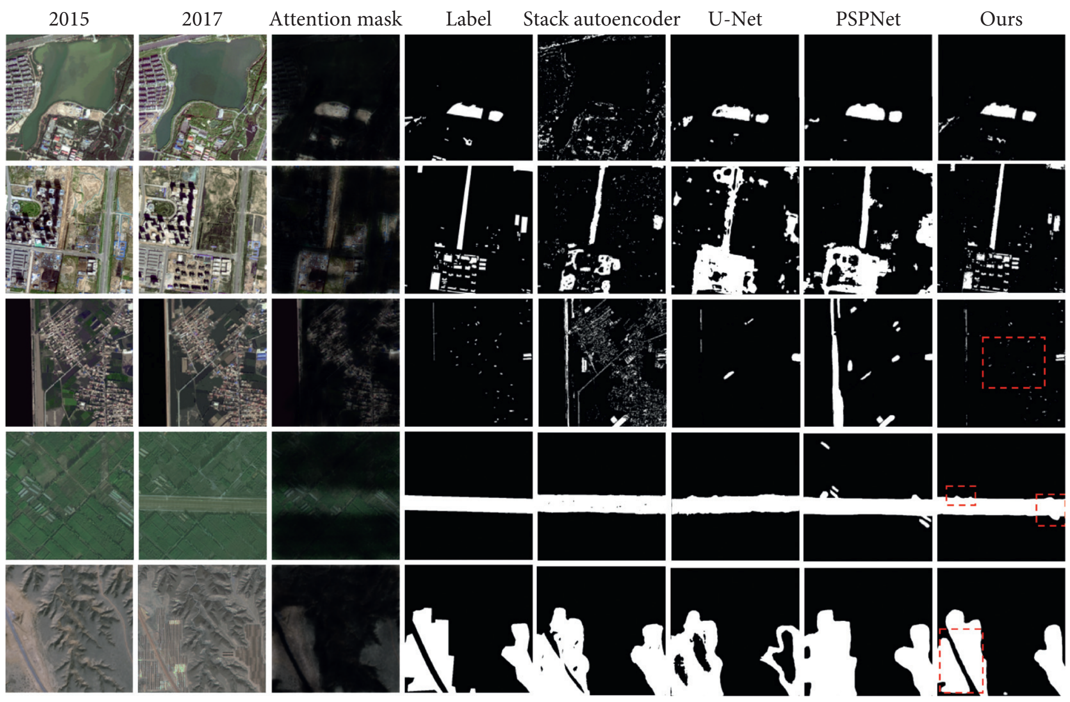


Figure 7: The attention mask and extraction results of the stack autoencoder, U-Net, PSPNet, and ours.

Table 4: Comparison of $F1$, recall values, and FLOPS values with the stack autoencoder, U-Net, PSPNet, and ours on the test datasets.

Module	$F1$	Recall	Pixel accuracy	FLOPS $\times 10^9$
Stack autoencoder	0.463	0.391	0.568	3.2
U-Net	0.689	0.691	0.687	5.6
PSPNet	0.724	0.704	0.745	11.2
Ours	0.743	0.697	0.796	4.5

5. Conclusion

In this paper, we propose a new network based on U-Net by combining the skip connection structure with the advanced residual attention mechanism and the DUpsampling method, and we also propose a new loss function suitable for application scenarios. Our network is applied to Yinchuan change detection task, and experiments show that compared with the current change detection methods, our network and loss function have improved accuracy and $F1$ value without wasting excessive computing resources. At the same time, our method has a strong antinoise ability and certain robustness which can better solve the change detection task of Yinchuan City.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request, but for study only, not for commercial use.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Authors' Contributions

This work was completed in cooperation with Long Chen, Peng Li, Dezheng Zhang, and Peng Lv. Among them, Long Chen and Peng Li designed and proposed a new type of network structure. Dezheng Zhang provided innovative thinking for the attention module so that the method can be used in the direction of change detection. Peng Lv proposed constructive opinions on the final loss function to achieve better results, while optimizing the upsampling structure makes it possible to quickly achieve end-to-end output.

Acknowledgments

This work was supported by Ningxia Hui Autonomous Region Key Research and Development Program of Ningxia Hui Autonomous Region under Grant 2019BFG02009 (Key Technologies for Intelligent Monitoring of Spatial Planning Based on High-Resolution Remote Sensing).

References

[1] L. I. Deren, *Change Detection from Remote Sensing Images*, Editorial Board of Geomatics & Information Science of Wuhan University, Wuhan, China, 2003.

[2] C. Zhang, S. Wei, S. Ji, and M. Lu, "Detecting large-scale urban land cover changes from very high resolution remote sensing images using CNN-based classification," *ISPRS International Journal of Geo-Information*, vol. 8, no. 4, p. 189, 2019.

[3] W. Ma, Y. Xiong, Y. Wu, H. Yang, X. Zhang, and L. Jiao, "Change detection in remote sensing images based on image mapping and a deep capsule network," *Remote Sensing*, vol. 11, no. 6, p. 626, 2019.

[4] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[5] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the Conference on Neural Information Processing Systems NIPS*, Curran Associates Inc, Red Hook, NY, USA, December 2012.

[6] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 39, no. 4, pp. 640–651, 2014.

[7] O. Ronneberger, P. Fischer, and T. Brox, "U-net: convolutional networks for biomedical image segmentation," in *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, Munich, Germany, October 2015.

[8] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2495–2481, 2015.

[9] F. Wang, M. Jiang, C. Qian et al., "Residual attention network for image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3156–3164, Honolulu, HI, USA, July 2017.

[10] Z. Tian, T. He, C. Shen, and Y. Yan, "Decoders Matter for Semantic Segmentation: Data-dependent Decoding Enables Flexible Feature Aggregation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, January2019.

[11] Y. Lecun, J. S. B. Boser, R. E. HowardHenderson, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[12] Z. Fei-Yan, J. Lin-Peng, and D. Jun, "Review of convolutional neural network," *Chinese Journal of Computers*, vol. 40, no. 1, pp. 1–23, 2017, in Chinese.

[13] A. Waibel, G. T. Hanazawa, and K. J. LangShikano, "Phoneme recognition using time-delay neural networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 3, pp. 328–339, 1989.

[14] Y. Lecun, Y. L. Bottou, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[15] O. Russakovsky, H. J. Deng, H. Su et al., "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[16] M. D. Ma and R. Fergus, "Visualizing and Understanding Convolutional Networks," in *Computer Vision – ECCV 2014*, Springer, Berlin Germany, 2013.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the International Conference on Learning Representations (ICLR)*, pp. 1409–1556, San Diego, CA, USA, May 2015.

[18] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, June 2014.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of theIEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, June 2015.

[20] V. Mnih, N. Heess, A. Graves et al., "Recurrent models of visual attention," in *Proceedings of the Conference on Neural Information Processing SystemsNIPS*, Vancouver, Canada, December 2014.

[21] K. Xu, "Show, attend and tell: neural image caption generation with visual attention," vol. 3, p. 5, 2015, https://arxiv.org/abs/1502.03044.

[22] J. Lu, C. Xiong, D. Parikh, and R. Socher, "Knowing when to look: adaptive attention via a visual sentinel for image captioning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, November 2017.

[23] P. Anderson, X. He, C. Buehler et al., "Bottom-up and top-down attention for image captioning and visual question answering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6077–6086, Salt Lake City, UT, USA, June 2018.

[24] L. C. Chen, G. Papandreou, I. Kokkinos et al., "Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[25] C. Peng, X. Zhang, G. Yu et al., "Large kernel matters--improve semantic segmentation by global convolutional network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4353–4361, Honolulu, HI, USA, June 2017.

[26] L. C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017, https://arxiv.org/abs/1706.05587.

[27] Z. Tian, T. He, C. Shen, and Y. Yin, "Decoders matter for semantic segmentation: data-dependent decoding enables flexible feature aggregation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3126–3135, Long Beach, CA, USA, June 2019.

[28] C. Carincotte, S. Derrode, and S. Bourennane, "Unsupervised change detection on SAR images using fuzzy hidden Markov chains," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 44, no. 2, pp. 432–441, 2006.

[29] J. Liu, M. Gong, J. Zhao, H. Li, and L. Jiao, "Difference representation learning using stacked restricted Boltzmann machines for change detection in SAR images," *Soft Computing*, vol. 20, no. 12, pp. 4645–4657, 2016.

[30] B. Desclée, P. Bogaert, and P. Defourny, "Forest change detection by statistical object-based method," *Remote Sensing of Environment*, vol. 102, no. 1-2, pp. 1–11, 2006.

[31] W. Qing, Z. Xiaodong, C. Guanzhou, F. Dai, Y. Gong, and K. Zhu, "Change detection based on Faster R-CNN for high-resolution remote sensing images," *Remote Sensing Letters*, vol. 9, no. 10, pp. 923–932, 2018.

[32] W. Ma, H. Yang, Y. Wu et al., "Change detection based on multi-grained cascade forest and multi-scale fusion for SAR images," *Remote Sensing*, vol. 11, no. 2, p. 142, 2019.

[33] H. Dong, W. Ma, Y. Wu et al., "Local descriptor learning for change detection in synthetic aperture radar images via convolutional neural networks," *IEEE Access*, vol. 7, pp. 15389–15403, 2019.

[34] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[35] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, https://arxiv.org/abs/1503.02531.

[36] T. Y. Lin, P. Goyal, R. Girshick et al., "Focal loss for dense object detection," in *Proceedings of the IEEE International*

Conference on Computer Vision, pp. 2980–2988, Venice, Italy, December 2017.

[37] D. P. Kingma and J. Ba, “Adam: a method for stochastic optimization,” 2014, https://arxiv.org/abs/1412.6980.

[38] G. E. Hinton, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[39] H. Zhao, J. Shi, X. Qi et al., “Pyramid scene parsing network,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2881–2890, Honolulu, HI, USA, November 2017.

Hindawi
Computational Intelligence and Neuroscience
Volume 2020, Article ID 3283890, 9 pages
https://doi.org/10.1155/2020/3283890

Research Article

Hybrid Low-Order and Higher-Order Graph Convolutional Networks

Fangyuan Lei, [1,2] Xun Liu, [2] Qingyun Dai, [1] Bingo Wing-Kuen Ling, [3] Huimin Zhao, [4] and Yan Liu [2]

[1] *Guangdong Province Key Laboratory of Intellectual Property and Big Data, Guangzhou 510665, China*
[2] *School of Electronic and Information, Guangdong Polytechnic Normal University, Guangdong, Guangzhou 510665, China*
[3] *School of Information Engineering, Guangdong University of Technology, Guangdong, Guangzhou, China*
[4] *School of Computer Sciences, Guangdong Polytechnic Normal University, Guangdong, Guangzhou 510665, China*

Correspondence should be addressed to Qingyun Dai; 1144295091@qq.com

Received 15 November 2019; Revised 26 April 2020; Accepted 23 May 2020; Published 23 June 2020

Academic Editor: Luca Manzoni

With the higher-order neighborhood information of a graph network, the accuracy of graph representation learning classification can be significantly improved. However, the current higher-order graph convolutional networks have a large number of parameters and high computational complexity. Therefore, we propose a hybrid lower-order and higher-order graph convolutional network (HLHG) learning model, which uses a weight sharing mechanism to reduce the number of network parameters. To reduce the computational complexity, we propose a novel information fusion pooling layer to combine the high-order and low-order neighborhood matrix information. We theoretically compare the computational complexity and the number of parameters of the proposed model with those of the other state-of-the-art models. Experimentally, we verify the proposed model on large-scale text network datasets using supervised learning and on citation network datasets using semisupervised learning. The experimental results show that the proposed model achieves higher classification accuracy with a small set of trainable weight parameters.

1. Introduction

Convolutional neural networks (CNNs) have achieved great success in grid structured data such as images and videos [1, 2]. It is attributed to a series of filters of convolutional layers from the CNNs that can obtain local invariant features. Compared to a regularized network, the number of neighbors of a node in a graph network may be different. Therefore, it is difficult to directly implement the filter operator in an irregular network structure [3].

In the graph network, the nodes and the connecting edges between them contain abundant network characteristic information. A graph convolutional network (GCN) aggregates the neighborhood nodes to realize continuous information transmission based on a graph network. By making full use of this information, a GCN can effectively achieve tasks such as classification, prediction, and recommendation.

A graph convolutional network (GCN) generalizes traditional convolutional neural networks (CNNs) to the graph domain. The GCN methods are mainly divided into two categories [3], the frequency domain-based methods [4–6] and the spatial domain-based methods [7, 8].

In the spatial domain, to simulate the convolution operation of the traditional CNN on an image, the convolution operation aggregates the information of the neighborhood nodes [7–10]. Henaff et al. [11] proposed a smoothed parametric spectral filter to realize localization and to preserve the parameters of filters independent of the input dimension. One of the key challenges is that the number of neighborhood nodes in the network irregularly changes.

In the frequency domain, Bruna et al. [5] were the first ones to extend CNN-type architectures to graphs. Cao et al. [12] applied a generalized convolutional network to the graph frequency domain using the Fourier transform. In this method, eigenvalue decomposition is performed on the neighborhood matrix. To reduce the computational complexity, Defferrard et al. [13] proposed the Chebyshev polynomial of the eigenvalues of the graph Laplacian to achieve efficient and localized graph convolutional operation filters. Kipf and Welling [6] proposed a classical GCN, which was approximated by a first-order Chebyshev polynomial. This approach reduces the computational complexity but introduces truncation errors. This introduction results in the inability to capture high-level interaction information between the nodes in the graph, and it also limits the capabilities of the model. The information propagation process in the graph is related not only to its first-order neighborhood but also to its higher-order neighborhood.

Abu-El-Haija et al. [14, 15] proposed the high-order convolutional network layer on a graph that used linear combination of the high-order neighborhood basis of the GCN [6]. Tiao et al. [16] proposed a Bayesian estimation approach via the stochastic variational inference in the adjacency matrix of the graph. Levie et al. [17] proposed Cayley polynomials to compute the localized regular filters of the interest frequency bands of graphs. Therefore, the rational use of second-order neighborhoods, third-order neighborhoods, and other high-order neighborhood information will be beneficial to classification prediction accuracy [14–16, 18–20].

Based on the classical GCN [6], to make full use of the high-order and low-order neighborhood information, we propose a novel hybrid low-order and higher-order graph convolutional network (HLHG). As shown in Figure 1, the graph convolutional layer of our model is simple and effective at capturing the high-order neighborhood information, nonlinearly combining the different order neighborhood information. The contributions are summarized as follows:

(1) We propose a new fusion pooling layer to achieve high-order neighborhood fusion with the low-order neighborhood of graph networks

(2) We propose a low-order neighborhood and high-order neighborhood weight sharing mechanism to reduce the computational complexity and number of parameters of the model

(3) The experimental results show that our HLHG achieves state-of-the-art performance in both the text network classification with supervised learning and the citation network with semisupervised learning

The rest of the paper is organized as follows. In Section 2, the related theoretical basis such as the graph convolution and the high-order graph convolution are introduced. In Section 3, the general information fusion pooling for the high-order neighborhood is presented. Then, the proposed model and its variant are presented. The computational complexity and parameter quantity of the proposed model are also theoretically analyzed. In Section 4, our proposed model is verified and the corresponding analysis are presented. Finally, Section 5 concludes the paper.

2. Related Theoretical Background

In this section, the related theoretical basis will be introduced, including the graph convolutional network (GCN).

2.1. Graph. Given a graph G, its nodes set V, and its edges E, the graph is represented as $G = (V, E)$. If nodes V_i and V_j are connected, then $E_{ij} = 1$; otherwise, $E_{ij} = 0$. The information in the graph propagates along with the edge E. It also applies when considering the network node self-loop, which means that $E_{ii} = 1$. Assuming that the information that is propagated by each node in the graph network is $x \in R^r$, the information matrix in the graph is $X \in R^{n\times r}$, where n is the total number of nodes in the graph network and r is the dimension of the information feature. It assumes that if the loop graph network G is represented as $\widetilde{G}$, then the adjacency matrix of the graph network $\widetilde{G}$ is represented as $\widetilde{A} = (A + I)$. The degree matrix of $\widetilde{A}$ in the graph network $\widetilde{G}$ is the diagonal matrix, $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$.

2.2. Graph Convolutional Network. In the given graph G, there are two signals $f = (f_1, \ldots, f_n)^T$ and $g = (g_1, \ldots, g_n)^T$. The graph's Fourier transforms are defined as $\widehat{f} = \Phi^T f$ and $\widehat{g} = \Phi^T g$, where Φ is the orthonormal eigenvalues of the graph Laplacian of graph G. The same as in Euclidean space, the spectral graph convolution operation of f and g is given as an elementwise product as follows:

$$g * f = \Phi\left(\left(\Phi^T g\right) \circ \left(\Phi^T f\right)\right) = \Phi\widehat{G}\Phi^T f, \tag{1}$$

where $\widehat{G} = \mathrm{diag}(\widehat{g}_1, \ldots, \widehat{g}_n)$ represents the diagonal matrix of $\widehat{g}$.

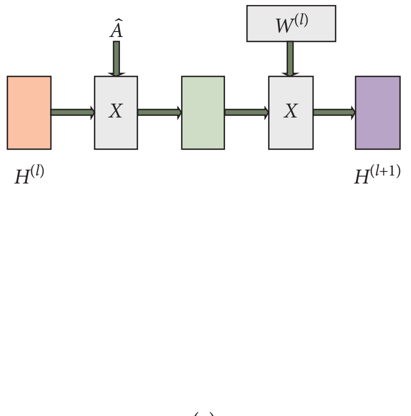
Defferrard et al. [13] utilized the k-th order polynomial filters based on Chebyshev to represent the graph convolutional operation of Laplacian $\widehat{G} = \sum_i \alpha_i \Lambda^i$, where α_i denotes the coefficients andΛ represents the eigenvalues of the Laplacian.

Kipf and Welling [6] propose the classical graph convolutional neural network model based on the Fourier transform, $g * f = \alpha\widetilde{A} f$. The GCN model approximates the model using a first-order Chebyshev polynomial. The propagation model in the graph network is as follows:

$$H^{(l+1)} = \sigma\left(\widetilde{D}^{-(1/2)}\widetilde{A}\widetilde{D}^{-(1/2)}H^{(l)}W^{(l)}\right), \tag{2}$$

where $H^{(l)}$ denotes the information propagation matrix; $W^{(l)}$ represents the trainable weight of layer l; when $l = 0$, $H^{(0)} = X \in R^{n\times r}$, which represents the initial input value of the GCN; $\sigma(.)$ denotes the activation function. To reduce the computational complexity, the convolution operator in the graph is defined by a simple neighborhood average. However, the convolutional filters are too simple to capture the high-level interaction information between the nodes in the graph. Therefore, the classification accuracy on citation network datasets is low.

Abu-El-Haija et al. [14, 15] propose a high-order graph convolutional layer model based on the GCN for semi-supervised node classification. The propagation model of the



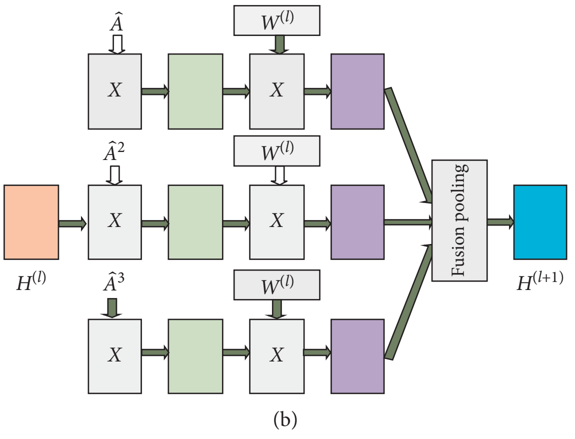


Figure 1: The graph convolutional layer of our model. (a) First-order graph convolutional layer of the Kipf and Welling [6] model. The input is $H^{(l-1)}$, the output is $H^{(l)}$, and the trainable parameter is $W^{(l)}$. (b) The 3rd order graph convolutional layer of our HLHG model. Different order neighborhood matrices share the trainable weight.

high-order graph convolution is as shown in formula (3). In this model, the transfer function of the $(l+1)$-th layer is a column concatenation from the first order to the p order in the l-th layer, which is the linear combination of the high-order neighborhood. In the propagation model, the different order neighborhoods of the same layer use different weight parameters:

$$H^{(l+1)} = \sigma\left(B^{(0)}H^{(l)}W_0^{(l)} \,\middle|\, \dots \,\middle|\, B^{(p)}H^{(l)}W_p^{(l)}\right), \quad (3)$$

where $B = \widetilde{D}^{-(1/2)}\widetilde{A}\widetilde{D}^{-(1/2)}$. However, as the network layers deepen, the dimensions of $H^{(l+1)}$ will increase and propagate between layers. Therefore, the number of trainable weight parameters will be more, and the training resource will also be increased to learn the optimized dimension of the weight.

3. Method

When the message passes through the graph network, the nodes will receive latent representations from their first-hop nodes and from their N-hop neighbors every time. In this section, we propose a model to nonlinearly aggregate the trainable parameters, which can choose how to mix latent messages from various hop nodes.

3.1. General Information Fusion Pooling.

3.1. General Information Fusion Pooling. The information propagation of the graph network is passed along the edges between the vertices in the graph. It assumes that the graph network $G = (V, E)$ is an undirected graph. The general procedure of fusion pooling is described as follows. It assumes that the k-th order neighborhood matrix is $A^{(k)} = [a_{ij}^{(k)}]$, and the result after the fusion pooling operator is $\text{Pmax}(A^{(0)}, \dots, A^{(k)}) = Z^{(k)} = [z_{ij}^{(k)}]$, where $z_{ij}^{(k)} = \max(a_{ij}^{(1)}, a_{ij}^{(2)}, \dots, a_{ij}^{(k)})$ and k represents the hop from the given node.

Here, is an example to show how to fuse the different order neighborhoods. For a given adjacency matrix $\widehat{A}$, assume that h_1 denotes the first-order neighborhood and h_2 denotes the second-order neighborhood.

If $h_1 = \widehat{A}XW_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ and $h_2 = \widehat{A}^2XW_1 = \begin{bmatrix} -1 & 0 \\ 2 & 1 \end{bmatrix}$, then $\text{Pmax}(h_1, h_2) = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}$.

In the information dissemination and fusion process, both the first-order neighborhood features and the high-order neighborhood features are fully considered. Therefore, the classification accuracy should be improved.

3.2. Our Proposed Model. In Figure 2, we propose the high-order graph convolutional network model to fuse the high-order messages that pass through the graph network. The model consists of an input layer, two graph convolutional layers, and an information fusion pooling layer that is connected to the graph convolutional layer. The softmax function is used for the multiclassification output.

The proposed model extends the classical GCN model [6] to the graph neural network of higher-order neighborhoods. Each node in the model can get its representation from its neighborhood and integrate messages. The system model is as follows:

$$Y = \mathcal{F}\left(Pm\left(\widehat{A}\sigma\left(H^{(l+1)}\right)W_{l+1}, \cdots, \widehat{A}^{(p)}\sigma\left(H^{(l+1)}\right)W_{l+1}\right)\right), \quad (4)$$

where p is the order of the neighborhoods, $\widehat{A}^{(p)} = \widehat{A}^{(p-1)}\widehat{A}$, $\sigma(.)$ is the activation function, function $\mathcal{F}(.)$ denotes the softmax function. Parameter W_{l+1} is the trainable weight parameter of layer $(l+1)$ in the graph network, and function $Pm(.)$ represents $\text{Pmax}(.)$, which denotes the hybrid high-order and low-order of the information fusion. When parameter l is equal to 0, $H^{(1)} = P\max(\hat{A}H^{(0)}W_0, \cdots, \widehat{A}^{(p)}H^{(0)}W_0)$, which is the output of the first convolutional layer of the graph propagation model. In addition, $H^{(0)} = X \in R^{n\times r}$, which represents the initial input of our model.

In the preliminary experiment, we found that the two-layer high- and low-order mixed graph convolution is better than the one-level high- and low-order mixed graph convolution, and stacking more layers does not significantly

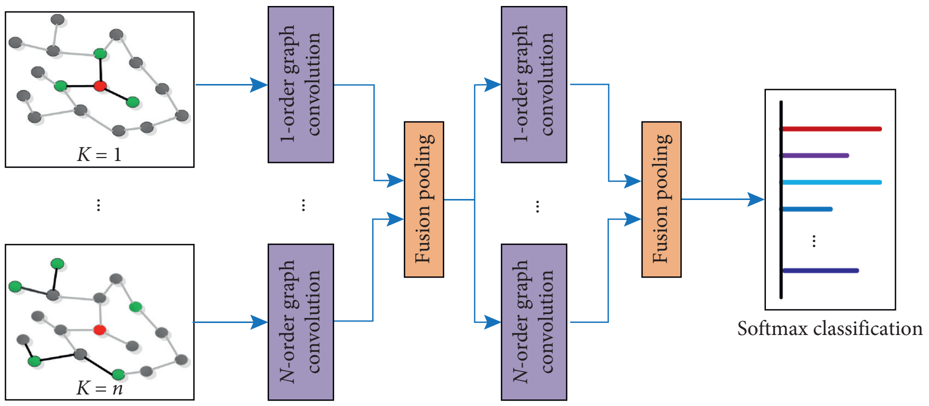

K = 1
K = n
1-order graph convolution
N-order graph convolution
Fusion pooling
1-order graph convolution
N-order graph convolution
Fusion pooling
Softmax classification

Figure 2: HLHG mode. The graph convolutional network layer of the HLHG model consists of two convolutional layers and information fusion pooling. The input parameters are from the first-order to the n-th order neighborhoods. When $n = 1$, the model degenerates into a classical graph convolution GCN model. When the neighborhood order is $n = 2$, it is called the HLHG-2 model, and its input parameters are the 1st order neighborhood and the 2nd order neighborhood. When the neighborhood order is $n = 3$, it is called the HLHG-3 model, and its input parameters are the 1st order neighborhood, the 2nd order neighborhood, and the 3rd order neighborhood.

improve the accuracy of the graph recognition task. Therefore, this paper uses a 2-layer graph convolution layer. In further experiments, we validate $p = 2$ and $p = 3$ in equation (4) for our HLHG models. In the supervised learning and unsupervised learning classification tasks, our HLHG models show very good performance and achieve a good balance between the classification accuracy and computational complexity. We also validate that at $p = 4$ and $p > 4$, the classification accuracy is not significantly improved. Therefore, we only analyze and implement our model for $p = 2$ and $p = 3$ in the following sections.

In equation (4), the model with $p = 2$, that is, the hybrid model of the 1st and 2nd order neighborhoods, is called the HLHG-2 model. The model with $p = 3$, that is, the hybrid model of the 1st, 2nd, and 3rd order neighborhoods, is called the HLHG-3 model.

In the HLHG-2 model, it assumes that the graph convolutional network has 2 convolutional layers and the activation function is Relu. Then, the output Y of the HLHG-2 model can be expressed as follows:

$$Y = \mathscr{F}\left[Pm\left[\widehat{A}(\mathrm{Relu}(M2))W_2, \widehat{A}^2(\mathrm{Relu}(M2))W_2\right]\right], \tag{5}$$

where $M2 = P\max(\widehat{A}XW_1, \widehat{A}^2XW_1)$ and Pm denotes the fusion pooling $P\max$.

The same as with the HLHG-2 model, the output Y of the HLHG-3 model can be expressed as follows:

$$Y = \mathscr{F}\left[Pm\left[\widehat{A}\mathscr{T}, \widehat{A}^2\mathscr{T}, \widehat{A}^3\mathscr{T}\right]\right], \tag{6}$$

where $\mathscr{T} = (\mathrm{Relu}(M3))W_2$ and $M3 = P\max(\widehat{A}XW_1, \widehat{A}^2XW_1, \widehat{A}^3XW_1)$.

For a large-scale graph network, it is unacceptable to directly calculate $\widehat{A}^3 = \widehat{A}^{(2)}\widehat{A} = \widehat{A}\widehat{A}\widehat{A}$. Therefore, we calculate $\widehat{A}^3XW_1 = \widehat{A}(\widehat{A}(\widehat{A}X))W_1$. In general, the dimension of $\widehat{A}X$ is less than $\widehat{A}$, and this procedure avoids large-scale matrix multiplication operations.

Therefore, our HLHG model has a 2-layer graph network, and the iterative expression of the 2nd order neighborhood is as follows:

$$Y = \mathrm{softmax}\left(\widehat{A}\mathrm{Relu}(H)W_2, \widehat{A}^2\mathrm{Relu}(H)W_2\right), \tag{7}$$

where $H = P\max(\widehat{A}XW_1, \widehat{A}^2XW_1)$. We use $P\max$ as our fusion pooling operator, which assumes the maximum value in the corresponding element. Algorithm 1 shows how to fuse the different order neighbors.

We use the multiclassified cross entropy as the loss function of our HLHG model, $L = -\sum_i \widetilde{y}_i \log(q_i)$, where $\widetilde{Y}$ is the labeled samples. The graph neural network trainable weights W_1 and W_2 are trained using gradient descent. In each training iteration, we perform the batch gradient descent.

3.3. Computational Complexity and Parameter Quantity.

In the large-scale graph network, the adjacency matrix is $\widehat{A} \in R^{n\times n}$. It is difficult to directly calculate $\widehat{A}^{(p)}$. To reduce the computational complexity, we iteratively calculate $\widehat{A}^{(p)}$. For higher orders, the right to left iterative multiplication procedure is $\widehat{A}^{(p)}H^{(l)}W_l = (\widehat{A}^{(p)}H^{(l)})W_l = \widehat{A}(\widehat{A}^{(p-1)}H^{(l)})W_l$. For example, when $p = 1$, $\widehat{A}^{(1)}H^{(0)} = \widehat{A}X \in R^{n\times r}$. When $p = 2$, $\widehat{A}^{(2)}H^{(1)} = \widehat{A}(\widehat{A}X) \in R^{n\times r}$.

In the proposed model, the input feature of the graph network is $X \in R^{n\times r}$. The weight of the first convolutional layer is $W_1 \in R^{r\times r_1}$, and the weight of the second layer is $W_2 \in R^{r_1\times r_2}$. Then, the input of the first convolutional layer is $H^{(0)} = X \in R^{n\times r}$ where the parameter r represents the dimension of the input feature. For example, r_1 denotes the number of hidden neurons in the first convolutional layer and r_2 denotes the number of hidden neurons in the second convolutional layer. In our HLHG model, the trainable weight parameters are shared in the same convolutional layer. Therefore, in the first convolutional layer, the output dimension after the convolutional operator is the same. That

```
(1) Inputs: $X$, $\widehat{A}$, and the other parameters.
    $N$ (number of hidden units), dr (dropout rate),
    L2 (L2 regularization), es (early stopping),
    epochs and lr (learning rate).
    Output: weight parameters $W_1$ and $W_2$.
(2) Randomly generate the trainable weights $W_1$ and $W_2$;
(3) Iteratively calculate the forward output value
    (1) $h_1 = \widehat{A}XW_1$, $h_2 = \widehat{A}^2XW_1 = \widehat{A}h_1$
    (2) $h_3 = P\max(h_1, h_2)$
    (3) $h_4 = \mathrm{Relu}(h_3)$;
    (4) $h_5 = \widehat{A}h_4W_2$, $h_6 = \widehat{A}^2h_5$
    (5) $h_7 = P\max(h_5, h_6)$
    (6) $Y = \mathrm{softmax}(h_7)$
    (4) Calculate the cross entropy $L = -\sum_i \widetilde{y}_i \log(q_i)$
```

Algorithm 1: Iterative calculation for HLHG-2.

is, $\widehat{A}XW_1 \in R^{n\times r_1}$, $\widehat{A}^{(2)}XW_1 \in R^{n\times r_1}$, and $\widehat{A}^{(k)}XW_1 \in R^{n\times r_1}$, where k is the order of the adjacency matrix $\widehat{A}$.

In the l-th convolutional layer, $\widehat{A}^{(k)}H^{(l)}W_l \in R^{n\times r_l}$, where r_l denotes the number of hidden neurons in the l-th convolutional layer. It assumes that $\widehat{A}$ is a sparse matrix with m nonzero elements. For the l-th convolutional layer of our HLHG, the computational complexity is $O(r_l\times k\times m\times r_{l-1})$ and the quantity of trainable weight is $O(r_l\times r_{l-1})$.

The total computational complexity of our HLHG model is $O(\sum_l^j (r_l\times k\times m\times r_{l-1}))$, and the total number of trainable parameters is $O(\sum_l^j (r_l\times r_{l-1}))$, where parameter j denotes the total number of convolutional layers and l denotes the l-th convolutional layer. When $l = 1$, r_0 represents the feature dimensions of the datasets and r_l represents the number of hidden neurons in the l-th convolutional layer. For all the datasets, $r_0 \gg r_l$; therefore, we only consider the first convolutional layer when we compare the computational complexity and number of parameters.

Compared to [14], we set fewer filters to maintain a similar computational complexity and the number of parameters is less via weight sharing for both the lower-order and higher-order convolutions.

4. Experiments

We conduct experiments in order to verify that our HLHG model can be applied to supervised learning and semisupervised learning. On the text network datasets, we compare our model with the state-of-the-art methods using supervised learning. On the citation network datasets, we compare our model with the state-of-the-art methods using semisupervised learning. For all experiments, we construct a 2-layer graph convolutional network of our model using TensorFlow. The code and data are available on GitHub.

4.1. Supervised Text Network Classification. We conduct supervised learning on five benchmark text graph datasets to compare the classification accuracy of HLHG with the graph convolutional neural network and other deep learning approaches.

4.1.1. Datasets. In our supervised experiments, the 20-Newsgroups (20NG), Ohsumed, R52 and R8 of Reuters 21578, and Movie Review (MR) are used to verify the proposed models. These datasets are publicly available on the web and are widely used as test-verified datasets. The summary statistic features of the text network are shown in Table 1.

These benchmark text datasets were processed by Yao et al. [21], who converted the text datasets into graph network structures. Then, they used preprocessing to construct the adjacency matrix of the graph network input and input parameters. The dataset is divided into a training dataset and a test dataset in the same way.

4.1.2. Baselines and Experimental Setting. We compare our HLHG with the following approaches: the convolutional neural network with pretrained vectors (CNN-rand) [22], the LSTM model with pretrained vectors (LSTM-pre) [23], the predictive text embedding for text classification (PTE) [24], the fast text classifier (fastText) [25], the simple word embedding model with simple pooling strategies (SWEM) [26], the label-embedding attentive model for text classification (LEAM) [27], the graph CNN model with the Chebyshev filter (GCN-C) [13], the graph CNN model with the spline filter (GCN-S) [5], the graph CNN model with the Fourier filter (GCN-F) [11], and the graph convolutional network for text classification (text GCN) [21]. The baseline models were tested by Yao et al. [21].

In our HLHG-2 model, we set the dropout rate = 0.2. The learning rate is updated from Adam [28] during the training process. In our model, we set the L2 loss weight as 0, and we adopt early stopping. We set the learning rate to 0.02 for the R8 dataset, and the learning rates of the remaining datasets are all set to 0.01. We set different epochs for different datasets. The number of epochs in the R52 dataset is 350. The number of epochs in the OH and 20NG datasets is 200, and the number in the R8 and MR datasets is 60. In the HLHG-2 model, we set the number of hidden neurons in the 1st convolutional layer as 128 for all datasets.

Except for the parameters in Table 2, the other parameters are the same as in the HLHG-2 model.

For our HLHG-3, we set the number of hidden neurons in the first convolutional layer to 128 except for the MR dataset, which is set to 64. To obtain better training results, we separately set different hyperparameters such as the dropout rate, learning rate, and number of epochs for different datasets (see Table 2). In addition, the other parameters of HLHG-3 are the same as those in HLHG-2.

We construct the graph network for our HLHG-2 and HLHG-3 models, and the feature matrix and other parameters are the same as those by Yao et al. [21].

4.1.3. Results. We show supervised text classification accuracies for the five datasets in Table 3. We demonstrate how

Table 1: Text network datasets.

Datasets	C	D	Tr	Te	N
R52	52	9,100	6,532	2,568	17,992
OH	23	7,400	3,357	4,043	21,557
20NG	20	18,846	11,314	7,532	61,603
R8	8	7,674	5,485	2,189	15,362
MR	2	10,662	7,108	3,554	29,426

C indicates the category, D is the total number of texts, Tr is the training set, Te is the test set, and N is the number of vertices of the graph network.

Table 2: The hyperparameters in our HLHG-3 model.

Datasets	Dropout	Learning rate	Epochs
R52	0.6	0.005	950
OH	0.2	0.01	230
20NG	0.0	0.01	210
R8	0.2	0.005	300
MR	0.1	0.01	80

Table 3: Text network classification accuracy.

Methods	R52	OH	20NG	R8	MR
CNN-rand [22]	87.59	58.44	82.15	95.71	**77.75**
LSTM [23]	85.54	41.13	65.71	93.68	75.06
LSTM-pre [23]	90.48	51.10	75.43	96.09	77.33
PTE [24]	90.71	53.58	76.74	96.69	70.23
fastText [25]	92.81	57.70	79.38	96.13	75.14
SWEM [26]	92.94	63.12	85.16	95.32	76.65
LEAM [27]	91.84	58.58	81.91	93.31	76.95
GCN-C [13]	92.75	63.86	81.42	96.99	77.22
GCN-S [5]	92.74	62.82	—	96.80	76.99
GCN-F [11]	93.20	63.04	—	96.89	76.74
Text GCN [21]	93.56	68.36	86.34	97.07	76.74
HLHG-2 (ours)	94.21 ± 0.14	69.16 ± 0.19	$\mathbf{86.57 \pm 0.08}$	$\mathbf{97.25 \pm 0.10}$	75.95 ± 0.14
HLHG-3 (ours)	$\mathbf{94.33 \pm 0.16}$	$\mathbf{69.36 \pm 0.24}$	86.35 ± 0.24	97.25 ± 0.12	76.49 ± 0.32

our model performs on common splits that were taken from Yao et al.'s study [21].

Table 3 presents the classification accuracies and standard deviations of our models and the benchmark on the text network data. In general, our HLHG-2 and HLHG-3 achieve high levels of performance. Specifically, they achieve the best performances on R52, OH, 20NG, and R8. Compared to the best performing approach, the proposed models yield worse accuracies on the MR dataset. In general, the HLHG-3 and HLHG-2 models perform equally well. More specifically, the 3rd order HLHG has slightly better classification accuracy than the 2nd order HLHG on most datasets. However, the performance difference is not very large. Overall, the proposed architecture with hybrid high- and low-order neighborhoods has good classification performance, which indicates that it effectively preserves the topological information of the graph, and it also obtains a high-quality representation of the nodes.

The benchmark test results are copied from [8]. The mean standard deviation of our model is the average of 100 runs.

Table 4 shows the comparison of the network complexity and the number of parameters with the Text GCN [21]. Our HLHG can match the Text GCN with respect to computational complexity while requiring fewer parameters than the Text GCN. As described in Section 3.3, the number of features in the dataset is much larger than the number of neurons in the hidden convolutional layer. Therefore, we only compare the computational complexity and number of parameters of the first convolutional layer in our HLHG model. In Table 4, Comp. and Params represent the computational complexity and the number of parameters in the first layer of the graph convolutional network, respectively. In the computational complexity results, the first constant denotes the number of neurons in the first convolutional layer and the second constant denotes the order of the adjacency matrix. The parameter m denotes the number of nonzero entries of the sparse regularization adjacency matrix. The parameter r denotes the feature dimension of the nodes in the graph network.

In the Text GCN [21], the number of hidden neurons in the first convolutional layer is 200; therefore, the complexity and params are 200. In our HLHG-2 model, 128 denotes the number of hidden neurons in the first convolutional layer and 2 represents the highest order of HLHG-2. In our HLHG-3 model, 128 and 64 denote the number of hidden

Table 4: Comparison of network computational complexity and the number of parameters.

Approaches	Comp.	Params
Text GCN [21]	O (**200** × **1** × m × r)	O (200 × r)
HLHG-2 (ours)	O (128 × 2 × m × r)	O (**128** × r)
HLHG-3 (ours)	O (**64** × **3** × m × r) (MR dataset) O (128 × 3 × m×r) (other datasets)	O (**64** × r) (MR dataset) O (**128** × r) (other datasets)

neurons in the first convolutional layer and 3 represents the highest order of the corresponding model. The result in Table 4 shows that our HLHG-3 model has better computational complexity for the MR dataset. Because of the weight sharing in the different order neighborhoods, our HLHG models require fewer trainable weight parameters. Especially on the MR dataset, the number of parameters is only 1/3 of that of the Text GCN [21].

4.2. Semisupervised Node Classification. We conduct semisupervised learning on three benchmark citation network datasets to compare the node classification accuracy of HLHG with some classical approaches and with some graph convolutional neural network approaches. The graph semisupervised learning corresponds to the process of “label” spreading on citation networks.

4.2.1. Datasets. In semisupervised node classification, we use the CiteSeer, Cora, and PubMed citation network datasets [29]. In these citation datasets, the nodes represent the articles that were published in the corresponding journal. The edges between the two nodes represent references from one article to another, and the tags represent the topics of the articles. The citation link constructs an adjacency matrix. Those datasets have low label rates. The summary statistic features of the citation graph are shown in Table 5.

4.2.2. Baselines and Experimental Setting. We compare our HLHG with the same baseline methods as by Abu-El-Haija et al. [15] and Yang et al. [30]. The baselines are as follows: manifold regularization (ManiReg) [31], semisupervised embedding (SemiEmb) [32], label propagation (LP) [33], skip-gram-based graph embeddings (DeepWalk) [34], the iterative classification algorithm (ICA) [35], Planetoid [30], HO [14], and MixHop [15].

For the HLHG-2 model, we use the following parameters for the citation datasets (Cora, CiteSeer, and PubMed): 16 (number of hidden units), 0.5 (dropout rate), 0.0005 (L2 regularization), 10 (early stopping), 300 (number of epochs), and 0.01 (learning rate).

For tthe HLHG-3 model, we set different numbers of hidden neurons for the different datasets. We set 8 hidden neurons for the CiteSeer dataset to reduce the computational complexity and the number of parameters, and set 10 hidden neurons for the Cora and PubMed datasets to capture richer features. The hyperparameters of the HLHG-3 are set as shown in Table 6.

Table 5: Citation network datasets.

Datasets	N	E	F	L	C
Cora	2708	5429	1433	0.052	7
CiteSeer	3327	4732	3703	0.036	6
PubMed	19717	44338	500	0.003	3

N means the number of nodes of citations, E means the number of edges between citations, F means the number of features of the nodes, L denotes the labeling rate, and C denotes the number of classes.

Table 6: The hyperparameters of HLHG-3.

Datasets	Dropout	Learning rate	Early stopping	Epochs
Cora	0.5	0.01	No	500
CiteSeer	0.5	0.005	5	500
PubMed	0.6	0.01	1	200

Table 7: Citation network classification test accuracy.

Approaches	Cora	CiteSeer	PubMed
ManiReg [31]	59.5	60.1	70.7
SemiEmb [32]	59.0	59.6	71.1
LP [33]	68.0	45.3	63.0
DeepWalk [34]	67.2	43.2	65.3
ICA [35]	75.1	69.1	73.9
Planetoid [30]	75.7	64.7	77.2
GCN [6]	81.5	70.3	79.0
HO-3 [14]	81.6 ± 0.47	71.2 ± 0.94	80.0 ± 0.64
HO-4 [14]	81.6 ± 0.63	71.2 ± 0.84	80.1 ± 0.65
MixHop [15]	81.8 ± 0.62	71.4 ± 0.81	80.0 ± 1.10
MixHop (learned) [15]	81.9 ± 0.40	71.4 ± 0.81	*80.8 ± 0.58*
HLHG-2 (ours)	*82.7 ± 0.28*	*71.5 ± 0.22*	79.1 ± 0.18
HLHG-3 (ours)	82.7 ± 0.29	71.5 ± 0.39	79.3 ± 0.15

4.2.3. Results. In the semisupervised experiments, we train and test our models on those citation network datasets following the methodology that was proposed by Yang et al. [30]. The classification accuracy is the average of 100 runs with random weight initializations.

The benchmark test results were copied from [15, 30]. The mean standard deviation of our model is the average of 100 runs.

In Table 7, the node classification accuracies that are above the line are copied from Abu-El-Haija [14, 15] and Yang et al. [30]. The values below the line are our HLHG models. ± represents the standard deviation of 100 runs with different random initializations. These splits utilize only 20 labeled nodes per class during training. We achieve the best test accuracies of 82.7% and 71.5% on the Cora and CiteSeer datasets, respectively. Compared with other high-order graph convolutional neural networks [14, 15] on the same datasets, they get the high-order information using linear combinations of features from farther distances. Our HLHG model acts nonlinearly to get the high-order neighborhood information.

In Table 8, we compare the network complexity and the number of parameters with the other high-order graph convolutional networks and the classic GCN. The result shows that our model has the same computational complexity as other approaches. With respect to the number of

Table 8: Comparison of network complexity and number of parameters.

Methods	Comp.	Params
GCN [6]	$O\ (\mathbf{16} \times m \times r)$	$O\ (16 \times r)$
HO-3 [14]	$O\ (10 \times 3 \times m \times r)$	$O\ (10 \times 3 \times r)$
HO-4 [14]	$O\ (10 \times 4 \times m \times r)$	$O\ (10 \times 4 \times r\mathrm{r})$
MixHop [15]	$O\ (20 \times 2 \times m \times \mathrm{r})$	$O\ (20 \times 3 \times r)$
MixHop (learned) [15]	$O\ (20 \times 2 \times m \times r)$	$O\ (60 \times r)$
HLHG-2 (ours)	$O\ (16 \times 2 \times m \times r)$	$O\ (16 \times r)$
HLHG-3 (ours)	$O\ (8 \times 3 \times m \times r)$ (CiteSeer)	$O\ (\mathbf{8} \times r)$ (CiteSeer)
	$O\ (10 \times 3 \times m \times r)$ (Cora, PubMed)	$O\ (\mathbf{10} \times r)$ (Cora, PubMed)

parameters, our HLHG-3 model has fewer parameters than the GCN [6]. The reason is that our model shares the weights in the same layer among the different order neighborhood matrixes.

5. Conclusion

In this paper, we propose a hybrid lower-order and higher-order GCN model for the supervised classification of text network datasets and for semisupervised classification in a citation network. In our model, we propose a novel nonlinear information fusion layer to combine the low- and higher-order neighborhoods. To reduce the number of parameters, we propose sharing the weights in the same convolutional layer with different order neighborhoods. Experiments on the two network datasets suggest that HLHG has the capability to fuse higher-order neighborhoods for supervised classification and semisupervised classification. Our model significantly outperforms the benchmarks. We also find that the computational complexity and the number of parameters are less than those of the high-order method. In order to obtain more neighborhood information, we could use more higher-order adjacency matrix. However, the direct use of higher orders may lead to oversmoothing problems. Therefore, in future research work, we will extend our HLHG models to fuse graph attention networks [36] to develop a deeper graph convolutional network.

Data Availability

The Supervised Text Network Classification data used to support the findings of this study have been deposited in the repository DOI:10.1609/aaai.v33i01.33017370. The Semi-supervised Node Classification data used to support the findings of this study have been deposited in the repository DOI:10.1609/aimag.v29i3.2157

Disclosure

The funding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; nor in the decision to publish the results.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grants U1701266, 61571141, 61702120, and 61672008; Guangdong Province Key Laboratory of Intellectual Property and Big Data under Grant 2018B030322016; Scientific and Technological Projects of Guangdong Province under Grant 2019A070701013; and Qingyuan Science and Technology Plan Project under Grants 170809111721249 and 170802171710591.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 1097–1105, Lake Tahoe, NV, USA, December 2012.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, Las Vegas, NV, USA, December 2016.

[3] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond Euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[4] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," 2017, https://arxiv.org/abs/1706.02216.

[5] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," 2013, https://arxiv.org/abs/1312.6203.

[6] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, https://arxiv.org/abs/1609.02907.

[7] F. P. Such, S. Sah, M. A. Dominguez et al., "Robust spatial filtering with graph convolutional neural networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 884–896, 2017.

[8] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," 2016, https://arxiv.org/abs/1605.05273.

[9] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5115–5124, Honolulu, HI, USA, July 2017.

[10] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning*, pp. 1263–1272, Sydney, Australia, August 2017.

[11] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," 2015, https://arxiv.org/abs/1506.05163.

[12] S. Cao, W. Lu, and Q. Xu, "Grarep: learning graph representations with global structural information," in *Proceedings of the 24th ACM International on Conference on Information*

and Knowledge Management, pp. 891–900, Melbourne, Australia, October 2015.

[13] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," 2016, https://arxiv.org/abs/1606.09375.

[14] S. Abu-El-Haija, N. Alipourfard, H. Harutyunyan, A. Kapoor, and B. Perozzi, "A higher-order graph convolutional layer," in *Proceedings of the 32nd Conference on Neural Information Processing Systems (NIPS 2018)*, NIPS, Montreal, Canada, December 2018.

[15] S. Abu-El-Haija, B. Perozzi, A. Kapoor et al., "MixHop: higher-order graph convolution architectures via sparsified neighborhood mixing," 2019, https://arxiv.org/abs/1905.00067.

[16] L. Tiao, P. Elinas, H. Nguyen, and E. V. Bonilla, "Variational spectral graph convolutional networks," 2019, https://arxiv.org/abs/1906.01852.

[17] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "CayleyNets: graph convolutional neural networks with complex rational spectral filters," *IEEE Transactions on Signal Processing*, vol. 67, pp. 97–109, 2018.

[18] G. Ma, N. K. Ahmed, T. Willke et al., "Similarity learning with higher-order graph convolutions for brain network analysis," 2019, https://arxiv.org/abs/1811.02662.

[19] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," 2019, https://arxiv.org/abs/1810.00826.

[20] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," *Advances in Neural Information Processing System*, vol. s, pp. 1993–2001, 2016, https://arxiv.org/abs/1511.02136.

[21] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," 2018, https://arxiv.org/abs/1809.05679.

[22] Y. Kim, "Convolutional neural networks for sentence classification," 2014, https://arxiv.org/abs/1408.5882.

[23] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," 2016, https://arxiv.org/abs/1605.05101.

[24] J. Tang, M. Qu, and Q. Mei, "PTE: predictive text embedding through large-scale heterogeneous text networks," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1165–1174, Sydney, Australia, August 2015.

[25] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," 2016, https://arxiv.org/abs/1607.01759.

[26] D. Shen, G. Wang, W. Wang et al., "Baseline needs more love: on simple word-embedding-based models and associated pooling mechanisms," 2018, https://arxiv.org/abs/1805.09843.

[27] G. Wang, C. Li, W. Wang et al., "Joint embedding of words and labels for text classification," 2018, https://arxiv.org/pdf/1805.04174.pdf.

[28] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014, https://arxiv.org/pdf/1412.6980.pdf,.

[29] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective classification in network data,," *AI Magazine*, vol. 29, no. 3, p. 93, 2008.

[30] Z. Yang, W. W. Cohen, and R. R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *Proceedings of the ICML*, New York, NY, USA, June 2016.

[31] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: a geometric framework for learning from labeled and unlabeled examples," *Journal of Machine Learning Research*, vol. 7, pp. 2399–2434, 2006.

[32] J. Weston, F. D. R. Ratle, H. Mobahi, and R. Collobert, "Deep learning via semi-supervised embedding," in *Neural Networks: Tricks of the Trade*, Springer, Berlin, Germany, 2012.

[33] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using Gaussian fields and harmonic functions," in *Proceedings of the ICML*, Washington, DC, USA, August 2003.

[34] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining—KDD'14*, New York, NY, USA, August, 2014.

[35] Q. Lu and L. Getoor, "Link-based classification," in *Proceedings of the ICML*, Washington, DC, USA, August 2003.

[36] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2017, https://arxiv.org/abs/1710.10903.