A Graph based Calligraphy Similarity Compare Model

Guoyang Pan
School of Artificial Intelligence, University of Chinese Academy of Sciences
Institute of Automation, Chinese Academy of Sciences
Beijing, China, panguoyang2019@ia.ac.cn

Yi Yang
Institute of Automation, Chinese Academy of Sciences
Beijing, China, yangyi@ia.ac.cn

Meng Li
Institute of Automation, Chinese Academy of Sciences
Beijing, China, meng_li@ia.ac.cn

Xueyang Hu
University of Maryland, Maryland, United States, xhu010127@gmail.com

Weixing Huang
Institute of Automation, Chinese Academy of Sciences
Beijing, China,
CASIA-Junsheng (Shenzhen) Intelligent & Big Data Sci-Tech Development Ltd.
Shenzhen, China, weixing.huang@ia.ac.cn

Jian Wang*
Institute of Automation, Chinese Academy of Sciences
Beijing, China, jian.wang@ia.ac.cn

Yun Wang
Institute of Automation, Chinese Academy of Sciences
Beijing, China, y.wang@ia.ac.cn

*Corresponding author: Jian Wang

***Abstract*—Calligraphy is one of the most famous traditional art in China. The Calligraphy copying practice is the inevitable phase when learning Calligraphy. Calligraphy character has structure and stroke attributes, such as length of stroke and the position distribution of subpart, which can identify each certain character. In this paper, we propose a graph neural network-based algorithm which can measure the similarity between two Calligraphy characters according to structure and stoke. Experiment shows that the proposed method gives satisfied results with respect to the similarity measurement for the Calligraphy copying practice.**

Keywords-calligraphy estimation; graph similarity; image process; graph neural network;

I. Introduction

Calligraphy is a famous Chinese traditional art. In recent years, there are many research works focus on the field of digital calligraphy by combining the traditional art and computer technology, such as calligraphy stroke segmentation [1][2][3], calligraphy style generation [4][5], and calligraphy author recognition [6][7].

The area of calligraphy similarity estimation consists of two basic aspects: stroke similarity estimation and structure similarity estimation. The process of stroke similarity starts at segmenting each stroke from the original calligraphy character. After strokes segmentation, it computes the similarity of the two strokes that can be treated as a shape similarity problem. Hence, shape similarity algorithms can be used for addressing this problem.

The structure of a calligraphy character represents quite abstract meaning and is different from character to character. Usually, the skeleton of the calligraphy character is used to represent the structure of character. And the current similarity algorithms between two characters skeletons focus on computing the distance between two points sets of skeleton position. Although such algorithms considered the position of skeleton point as the local information, they took few global information for illustrating the overall characteristics of a calligraphy character. Hence, it can't totally represent the structure similarity of two calligraphy character.

In this paper, we define the skeleton of a Calligraphy character as a graph model. We embed the graph model into a vector space using graph neural network. The embedding model can help to measure the similarity between two given calligraphy images. Experiment shows that the method gives satisfied result in estimating structure similarity.

The paper is organized as follows: The part one is the introduction of the paper. The part two is the related work of the calligraphy estimated. The part three shows the details of the proposed graph neural network. The part four shows comparative experience and result. The last part draws the conclusion of the paper.

II. Related work

A. Similarity of Calligraphy Characters

The similarity of two calligraphy characters contains two important parts: stroke similarity and structure similarity. A Calligraphy character consists of a set of strokes that is the basic component. The similarity of stroke pairs represents the similarity of shape pairs in the abstract level.

B. Graph Similarity Search and Graph Kernels

Graph similarity search has been studied extensively in database and data mining communities[8][9]. The similarity was typically defined by either exact match (full-graph or sub-graph isomorphism)[10] or measuring of structural similarity, e.g. graph edit distances[11]. Most of the approaches proposed in this direction were not learning-based, and focused on efficiency.

Graph kernels are kernels on graphs designed to capture the graph similarity, and can be used in kernel methods for e.g. graph classification[12]. Popular graph kernels included those that measured the similarity between walks or paths on graphs[13], kernels based on limited-sized substructures and kernels based on sub-tree structure[14]. A recent survey on graph kernels can be found in the article [15]. Graph kernels were usually used in models that may have learned components, but the kernels themselves were hand-designed and motivated by graph theory. They can typically be formulated as first computing the feature vectors for each graph (the kernel embedding), and then take inner product between these vectors to compute the kernel value. One exception was [16] where the co-occurrence of graph elements (substructures, walks, etc.) were learned, but the basic elements were still hand-designed. Compared to these approaches, our graph neural network, which is based on similarity learning framework, learns the similarity metric end-to-end.

C. Graph Nural Network and Graph Representation Learning

The history of Graph Neural Networks (GNNs) goes back to at least the early work by Gori and Scarselli [17], who proposed to use a propagation process to learn node representations. These models have been further developed by incorporating modern deep learning components [18]. A separate line of work focused on generalizing convolutions to graphs [19]. Popular graph convolutional networks also computed node updates by aggregating information in local neighborhoods [20], making them the same family of models as GNNs. GNNs have been successfully used in many domains [21]. Most of the previous work on GNNs focused on supervised prediction problems [22]. The graph similarity learning problem we study in this paper and the new graph matching model can be good additions to this family of models. Independently Al-Rfou [23] also proposed a cross graph matching mechanism similar to ours, for the problem of unsupervised graph representation learning.

Recently Xu [24], Morris [25] studied the discriminative power of GNNs and concluded that GNNs are as powerful as the Weisfeiler-Lehman algorithm in terms of distinguishing graphs (isomorphism test). In this paper, however we study the similarity learning problem, i.e. how similar are two graphs, rather than whether two graphs are identical. In this setting, learned models can adapt to the metric we have data for, while hand-coded algorithms cannot easily adapt.

III. GRAPH NEURAL NETWORK MODEL FOR CALLIGRAPHY STRUCTURE SIMILARITY MEASUREMENT

A. Framework

Given a calligraphy character image, denoted as ***P***. First of all, extract the skeleton of the calligraphy image, then propose a method that establish the skeleton to the graph structure ***G***. We propose a mapping function ***F*** that transforms G to the distance space, then a distance function $\boldsymbol{d}(\boldsymbol{G_1},\boldsymbol{G_2})$ in distance space can measure the structure similarity between any graph $\boldsymbol{G_i}$ and $\boldsymbol{G_j}$.

B. Establish Graph Structure From Calligraphy character Skeleton

Skeleton is an important feature of a shape, especially for calligraphy character. We use skeleton extract algorithm [30] to extract skeleton of the calligraphy character which is shown in the Figure 1. The graph has the node set and edge set, so we need to transfer the original skeleton to graph structure.

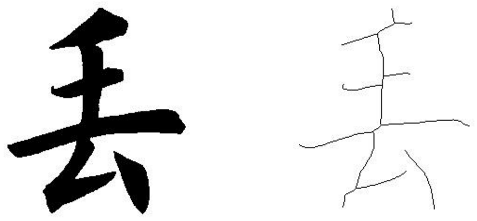

Figure 1 The original calligraphy character(left) and the skeleton of calligraphy character(right)

Intuitively, we can just model each skeleton point as a node in the graph, and if two nodes are 8-neighbor connected, we set an edge between these two nodes. The number of points in the graph is the number points in the skeleton, which is usually more than 300 points in an image with 256×256 pixels. Most of the points are redundant and will increase the complexity of the down-stream algorithm. Furthermore, most of the nodes in the graph only have two edges between other nodes. The adjacent matrix of this graph is quite sparse.

So, in order to improve the representation effectiveness of the calligraphy graph structure, we establish the graph structure through the follow steps:

- Step 1: Finding all terminal points in the skeleton image. Terminal points have the feature with only one neighbor point. The terminal points are usually the terminal of the strokes of character.
- Step 2: Finding all crossing points in the skeleton image. Crossing points have the feature with more than three neighbor points. The crossing points usually are the cross points between two strokes of character.
- Step 3: If the crossing points are connected which is shown in Figure 2, these connected crossing points will be considered as one point in the graph.

- Step 4: Visiting every terminal node. If a terminal node is connected with a crossing point, then add an edge between these two nodes.
- Step 5: Adding inner points as nodes for building the complex structure of stroke for the case that a stoke between two nodes are too long to build a correct structure, which is show in Figure 3.

Eventually, the graph structure has been established, which consider the trade of between node number and represent effectiveness.

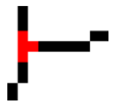
Figure 2 Each square is a pixel of the skeleton, all of the red square has more than three neighbor, these connected red square should be considered as one node in the graph generated by the skeleton.

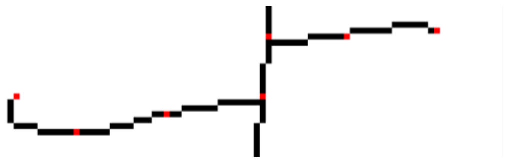
Figure 3 To take more information of the stroke into account, we add inner nodes between terminal node and cross node. The connected crossing points are reduced to one behavior point in this figure.

C. *Graph Embedding Model*

Given two graphs $\boldsymbol{G_1(V_1, E_1)}$ and $\boldsymbol{G_2(V_2, E_2)}$, our model embeds the two graphs into two vectors individually through node embedding, message propagation, aggregation process, which is show in the Figure 4. Then we use similarity metric measure the similarity between graphs.

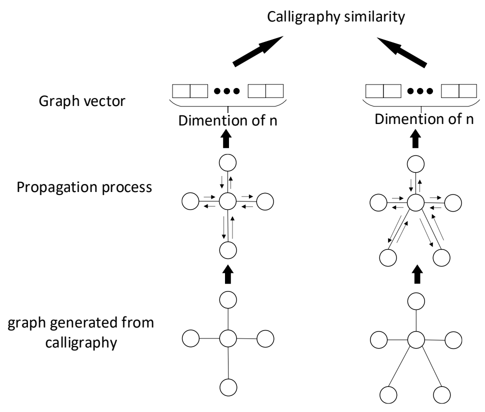

Figure 4. The framework of propoese model to compute similarity of to graph which is generated by the calligraphy skeleton to represent similarity of calligraphy characters.

- Encoding layer

The encoding layer encodes the node feature and edge feature through MLP, as shown in the follow.

$$h_i^{(0)} = MLP_{node}(x_i), \forall i \in V$$
$$e_i^{(0)} = MLP_{edge}(x_{ij}), \forall (i, j) \in E$$

- Message propagation layer

The message propagation layer is the core layer of the graph neural network. Each node collects features from its neighbor nodes and merges the information to the features of itself. Here we use the Graph Attention Network(GAT)[26] as the message propagation module. We can make multilayer GAT module to collect faraway node features of the current nodes.

$$m_{j\to i} = f_{message}(h_i^{(t)}, h_j^{(t)}, e_{ij})$$
$$h_i^{(t+1)} = f_{node}(h_i^{(t)}, \sum_{j:(j,i)\in E} m_{j\to i})$$

- Aggregator layer

After a certain number $\boldsymbol{T}$ rounds of propagations, an aggregator takes all node representations as input, and computes a graph level representation. That means, to aggregate all the node features and embed the whole graph into a vector. We use the proposed aggregation module in [27]. This aggregate module transforms node feature via MLP layer and weight them together with the coefficient from gate MLP layer. Eventually, the last MLP layer transform the vector to the target dimension vector.

$$h_G = MLP_G(\sum_{i\in V} \sigma(MLP_{gate}(h_i^{(T)})) \odot MLP(h_i^{(T)}))$$

D. *Loss Function*

We train our model on a set of triples with an end-to-end framework. The triple data has three graphs, $\boldsymbol{G_1}$, $\boldsymbol{G_2}$ and $\boldsymbol{G_3}$. All of three graphs are from the same calligraphy character. $\boldsymbol{G_1}$ is the reference calligraphy. $\boldsymbol{G_2}$ and $\boldsymbol{G_3}$ are the copying characters. $\boldsymbol{G_2}$ is more similar with the $\boldsymbol{G_1}$ than with $\boldsymbol{G_3}$ and $\boldsymbol{G_3}$ is more dissimilar with G_1 than with $\boldsymbol{G_2}$. We call $\boldsymbol{G_2}$ the positive labeled graph and $\boldsymbol{G_3}$ the negative labeled graph. Our model needs to predict the positive labeled graph more similar than the negative labeled graph. Hence, we optimize the following margin-based triplet loss:

$$L_{triplet} = E_{(G_1,G_2,G_3)}[\max\{0, d(G_1,G_2) - d(G_1,G_3) + \gamma\}]$$

This loss encourages $\boldsymbol{d(G_1, G_2)}$ to be smaller than $\boldsymbol{d(G_1, G_3)}$ by at least a margin $\boldsymbol{\gamma}$.

IV. EXPERIMENT

A. *Dataset*

We collect 200 calligraphy character images which were created by famous Chinese calligrapher such as Yan Zhenqing, Wen Zhengming, and Mi Fu. We label 200 positive images and 200 negative character images from calligraphy learner.

The positive calligraphy images are collected from the calligraphy learner. The positive labeled images do the best to have the excellent similarity performance during their generation process by the learner.

The negative calligraphy images are also collected from the calligraphy learner. Generally, the negative labeled calligraphy character images have the following shortcomings: (1) strokes may be longer or shorter than reference calligraphy character image, (2) position of subpart may deviate from its reference position, (3) stroke may be wider or thinner than the reference calligraphy character image.

As shown in Figure 5, left column is the reference calligraphy images, middle column is the positive labeled calligraphy image and the right column is the negative calligraphy image.

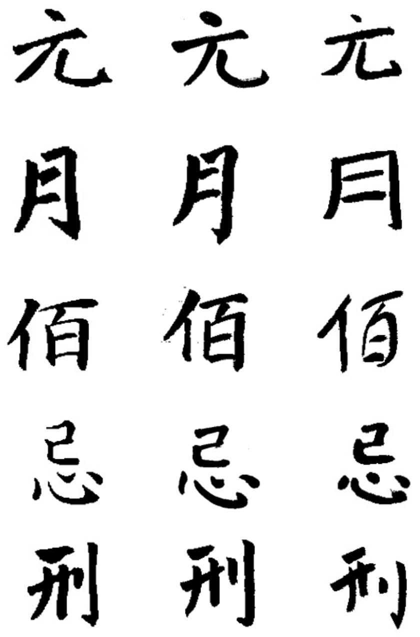

Figure 5 The left column contains the five reference images. The middle column contains the corresponding positive images which is much similar to the reference images. The right column contains the corresponding negative labeled images which is not similar to the reference and usually have some structure problems.

We invited a calligraphy expert and asked him whether the labeled results are correct in the calligraphy domain. And then we rectify incorrect labels in the dataset according to the expert's advice.

B. Node Feature and Edge Feature of Graph With prior Information

Given two graphs $\boldsymbol{G_1}$=($\boldsymbol{V_1}$, $\boldsymbol{E_1}$) and $\boldsymbol{G_2}$=($\boldsymbol{V_2}$, $\boldsymbol{E_2}$), we want a model that produces the similarity score $\boldsymbol{s}(\boldsymbol{G_1}, \boldsymbol{G_2})$ between them. Each graph $\boldsymbol{G}$=($\boldsymbol{V}$, $\boldsymbol{E}$) is represented as sets of nodes $\boldsymbol{V}$ and edges $\boldsymbol{E}$. Optionally each node $\boldsymbol{i} \in \boldsymbol{V}$ can be associated with a feature vector $\boldsymbol{x_i}$, and each edge ($\boldsymbol{i}$, $\boldsymbol{j}$) $\in \boldsymbol{E}$ associated with a feature vector $\boldsymbol{x_{ij}}$. These features can represent, e.g. type of a node and direction of an edge. If a node or an edge does not have any associated features, we set the corresponding vector to a constant vector of 1s.

We train a Resnet50 [29] network to predict the calligraphy character category with dataset which has more than two million images of 3,500 kinds of Chinese common characters. After testing, the model has the predict accuracy of 97% at the worst performance. The convolution layer can extract the feature of the image from shallow level to the deep level. And do prediction based on the aggregation feature. Given a calligraphy character image, we extract the output features from the first and second convolution layers as the node feature in the corresponding position.

In our dataset, we set edge attribute and set node attribute from the middle feature of the pretrained ResNet50 character recognition network as the prior information.

For example, the pixel position of node n in the image is ($\boldsymbol{x}$, $\boldsymbol{y}$). The feature extraction module has the convolutional layer and pooling layer, which usually reduce the image size proportionally. We find the corresponding pixel position ($\boldsymbol{x_1}$, $\boldsymbol{y_1}$) of the source position ($\boldsymbol{x}$, $\boldsymbol{y}$). Then pick out the whole channel in the pixel position in ($\boldsymbol{x_1}$, $\boldsymbol{y_1}$) as the node feature.

The node feature has a dimension number of 256, which is flattened from the channel of node position of the feature.

We use data augmentation to expand the generalization of our model, which includes image scaling and image rotating in small range.

C. Graph network Structure

We have given the overall structure of the proposed network. In the experiment, the concrete parameter is shown as Table 1.

Layer name	**Input Dimension**	**Output Dimension**
Node encoding layer	N × 256	N × 128
Propagation layer 1	N × 128	N × 128
Propagation layer 2	N × 128	N × 128
Propagation layer 3	N × 128	N × 64
Aggregator layer	N × 64	64

TABLE I. INNER PARAMETERS OF PROPOESED MODUEL

We use three propagation layers to update the attribute of each node. All of the three layers use graph attention module as the message aggregation function, while layer 1 and layer 2 with two multi-heads and layer 3 without multi-head.

The $\boldsymbol{N}$ respect to the node number of the input graph. The aggregator layer collects all attribute of nodes, then the input graph is embedded into a vector with 64 dimensions.

D. Result

We compare precision of prediction of our model with the traditional graph kernel framework. The graph kernel framework transforms the input graph into the different vector spaces with different kernel functions. We even could design our own graph kernel method follow graph kernel method.

We train our model on our dataset with the following configuration: Adam optimizer [28] with decay 0.9 and momentum 0.9, initial learning rate 0.002 that decays by 0.97 every 2.4 epochs.

Table 1 shows the prediction result of our model, graph kernel with shortest path kernel and graphlet sampling kernel. The shortest path kernel and graphlet sampling kernel need node attribute optionally and do not need edge attribute. Result shows that our model has better performance than the traditional graph kernel method.

Method	**Prior Information**	**Test Accuracy**
Shortest path kernel	False	60%
graphlet sampling kernel	False	72%
Shortest path kernel	True	63%
graphlet sampling kernel	True	72%
Our model	False	69%
Our model	True	78%

TABLE II. EXPERIMENT RESULT COMPARATION

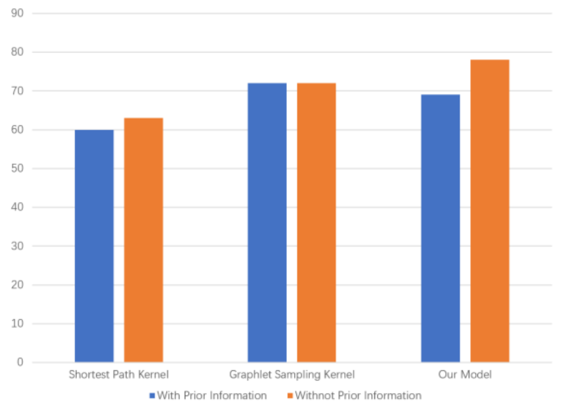

Figure 6 The Histogram figure of the comparing experience result of different method.

The training process learns an embedding network that embedding the input graph in to a vector with the dimension of 32. Therefore, facing with the application scenario that scoring the calligraphy character copying practice from the reference calligraphy reference character, we can just simply deem the normed distance of two embedding vectors as the quality scores of the outcomes of the calligraphy characters copying practice.

V. CONCLUSION

In this paper, we propose a graph similarity model which embed the graph into a vector through graph neural network. Experiment shows that the result outperforms the traditional graph kernel methods in the area of time and space complexity.

The proposed model has following limitations:

- The prediction precision can be largely improved with more efficient network model.
- The model size can be compressed with novel model compressing algorithm. It will reduce the number of model parameters and increase response rate of algorithm.
- From the application point of view, the model does not tell how to rectify the negative labeled calligraphy character image so as to make it more likely to the reference character image.

In the future, we will concentrate on increasing the predict precision on positive labeled calligraphy and negative labeled calligraphy. Furthermore, we will focus on more specific similarity criterion and give the natural languages-based advises to the calligraphy character copying practice.

ACKNOWLEDGMENT

This work was supported by National Key Research and Development Project under Grant No. E0M2040101. We also thank all colleagues and graduate students who helped us for our system and experiments.

REFERENCES

[1] Bi F, Han J, Tian Y, et al. SSGAN: generative adversarial networks for the stroke segmentation of calligraphic characters[J]. The Visual Computer, 2021: 1-10

[2] ZHU X, YANG C. Graph Based Stroke Extraction for Chinese Calligraphy[J]. Software Guide, 2019.

[3] Chen X, Lian Z, Tang Y, et al. A benchmark for stroke extraction of chinese characters[J]. Acta Scientiarum Naturalium Universitatis Pekinensis, 2016, 2(3): 4.

[4] Narasimhan B. Calligraphy Style Transfer using Generative Adversarial Networks[J].

[5] Miao Y, Jia H, Tang K, et al. Chinese Calligraphy Generation Based on Residual Dense Network[C]//Proceedings of the 2019 4th International Conference on Intelligent Information Processing. 2019: 508-512.

[6] Zhang J, Guo M, Fan J. A novel CNN structure for fine-grained classification of Chinese calligraphy styles[J]. International Journal on Document Analysis and Recognition (IJDAR), 2019, 22(2): 177-188.

[7] Zhang J, Wang L, Wen X. Combination of GIST and PHOG Features for Calligraphy Styles Classification[C]//Proceedings of the 2019 4th International Conference on Multimedia Systems and Signal Processing. 2019: 21-24.

[8] Yan X, Yu P S, Han J. Substructure similarity search in graph databases[C]//Proceedings of the 2005 ACM SIGMOD international conference on Management of data. 2005: 766-777.

[9] Dijkman R, Dumas M, García-Bañuelos L. Graph matching algorithms for business process model similarity search[C]//International conference on business process management. Springer, Berlin, Heidelberg, 2009: 48-63. ference on Management of data, pp. 766–777, 2005.

[10] Berretti S, Del Bimbo A, Vicario E. Efficient matching and indexing of graph models in content-based retrieval[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2001, 23(10): 1089-1105.

[11] Willett P, Barnard J M, Downs G M. Chemical similarity searching[J]. Journal of chemical information and computer sciences, 1998, 38(6): 983-996.

[12] Vishwanathan S V N, Schraudolph N N, Kondor R, et al. Graph kernels[J]. Journal of Machine Learning Research, 2010, 11: 1201-1242.

[13] Borgwardt K M, Kriegel H P. Shortest-path kernels on graphs[C]//Fifth IEEE international conference on data mining (ICDM'05). IEEE, 2005: 8 pp.

[14] Shervashidze N, Borgwardt K M. Fast subtree kernels on graphs[C]//NIPS. 2009: 1660-1668.

[15] Kriege N M, Johansson F D, Morris C. A survey on graph kernels[J]. Applied Network Science, 2020, 5(1): 1-42.

[16] Yanardag P, Vishwanathan S V N. Deep graph kernels[C]//Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining. 2015: 1365-1374.

[17] Gori M, Monfardini G, Scarselli F. A new model for learning in graph domains[C]//Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005. IEEE, 2005, 2: 729-734.

[18] Li Y, Tarlow D, Brockschmidt M, et al. Gated graph sequence neural networks[J]. arXiv preprint arXiv:1511.05493, 2015.

[19] Bronstein M M, Bruna J, LeCun Y, et al. Geometric deep learning: going beyond euclidean data[J]. IEEE Signal Processing Magazine, 2017, 34(4): 18-42.

[20] Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks[J]. arXiv preprint arXiv:1609.02907, 2016.

[21] Gilmer J, Schoenholz S S, Riley P F, et al. Neural message passing for quantum chemistry[C]//International conference on machine learning. PMLR, 2017: 1263-1272.

[22] Li Y, Vinyals O, Dyer C, et al. Learning deep generative models of graphs[J]. arXiv preprint arXiv:1803.03324, 2018.

[23] Al-Rfou R, Perozzi B, Zelle D. Ddgk: Learning graph representations for deep divergence graph kernels[C]//The World Wide Web Conference. 2019: 37-48.

[24] Xu K, Hu W, Leskovec J, et al. How powerful are graph neural networks?[J]. arXiv preprint arXiv:1810.00826, 2018.

[25] Morris C, Ritzert M, Fey M, et al. Weisfeiler and leman go neural: Higher-order graph neural networks[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2019, 33(01): 4602-4609.

[26] Veličković P, Cucurull G, Casanova A, et al. Graph attention networks[J]. arXiv preprint arXiv:1710.10903, 2017.

[27] Li Y, Tarlow D, Brockschmidt M, et al. Gated graph sequence neural networks[J]. arXiv preprint arXiv:1511.05493, 2015.

[28] Kingma D P, Ba J. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.

[29] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

[30] Zhang T Y, Suen C Y. A fast parallel algorithm for thinning digital patterns[J]. Communications of the ACM, 1984, 27(3): 236-239.