ResGCN:A method to train deep graph convolutional network

Mingkai Wang[1], **Yiwei Ma**[1], **Ming Li**[1]
[1]Xiamen University
Fujian, China 361005
[1]23020201153854,31520201153913,23020201153844

Abstract

Convolutional Neural Networks (CNN) have achieved impressive performance in a wide range of fields such as image classification, semantic segmentation or machine translation, where the underlying data representation has a grid-like structure. When very deep CNN models can be trained reliably, their success is due to large-scale push. Despite its advantages, CNN cannot correctly solve the problem of non-Euclidean data. In order to overcome this challenge, graph convolutional networks (GCN) can construct graphs to represent non-Euclidean data, draw on the concept of CNN, and apply it to training. GCN can handle many interesting tasks involve data that can not be represented in a grid-like structure and that instead lies in an irregular domain. This is the case of 3D meshes, social networks, telecommunication networks, biological networks or brain connectomes. GCN shows good results, but since the vanishing gradient problem appears, they are usually limited to very shallow models. In this paper, we propose a method to train deeper GCN. We do this by borrowing concepts from the residual connections of Resnet. Experiments have shown the positive effects of these GCN added with residual connections.

Introduction

Many interesting tasks involve data that can not be represented in a grid-like structure and that instead lies in an irregular domain. This is the case of 3D meshes, social networks, telecommunication networks, biological networks or brain connectomes. Such data can usually be represented in the form of graphs.The rise of availability of non-Euclidean data has recently shed interest into the topic of Graph Convolutional Networks (GCNs). GCNs provide powerful deep learning architectures for unstructured data, like point clouds and graphs. GCNs have already proven to be valuable in several applications including predicting individual relations in social networks [Tang and Liu, 2009], modelling proteins for drug discovery [Zitnik and Leskovec, 2017, Wale et al., 2008], enhancing predictions of recommendation engines [Monti et al., 2017b, Ying et al., 2018], and efficiently segmenting large point clouds [Wang et al., 2018].

Learning with graph structured data, such as molecules, social, biological, and financial networks, requires effective representation of their graph structure (Hamilton et al., 2017b). Recently, there has been a surge of interest in Graph Neural Network (GNN) approaches for representation learning of graphs (Li et al., 2016; Hamilton et al., 2017a; Kipf & Welling, 2017; Velickovic et al., 2018; Xu et al., 2018). GNNs broadly follow a recursive neighborhood aggregation (or message passing) scheme, where each node aggregates feature vectors of its neighbors to compute its new feature vector (Xu et al., 2018; Gilmer et al., 2017). After k iterations of aggregation, a node is represented by its transformed feature vector, which captures the structural information within the node's k-hop neighborhood.

Convolutional Neural Networks (CNNs) have been successfully applied to tackle problems such as image classification (He et al., 2016), semantic segmentation (Jegou et al., 2017) or machine translation (Gehring et al., 2016), where the underlying data representation has a grid-like structure. These architectures efficiently reuse their local filters, with learnable parameters, by applying them to all the input positions.

A key reason behind the success of CNN is the ability to design and reliably train very deep CNN models. In contrast, GCN is not able to be designed very deep. Stacking more layers into GCN will cause the vanishing gradient problem. This means that back-propagation through these networks will lead to over-smoothing, which will eventually cause the features of the graph vertices to converge to the same value.

In the CNN world, the disappearing gradient is not an alien phenomenon. This also limits the deeper development of such networks. When ResNet introduces residual connections between the input and output layers, it has taken a big step in the pursuit of a very deep CNN. These connections greatly alleviate the disappearing gradient problem.

In this work,to address the above issues by adapting concepts that were successful in training deep CNN. To showcase these layer adaptations, we apply them to the popular task. We show that adding residual connections enables successful training of GCNs up to more layers.

We summarize our contributions as followed:

- We propose a new structure Deeper GCN by adapting residual connections into the graph structure. Graph convolutional network layer that combines with residual con-

nections will go deeper. Thus, we solve the gradient explosion and vanishing problem of GCN.

- We conduct several extensive experiments on a number of datasets and use the new GCN models called ResGCN to these experiments and show availability and effectiveness of the new models.
- We find a new way to design graph convolutional network with CNN concepts. In the future, we will try to design graph convolutional network structure with other concepts and we believe it will bring inspiration to design the structure.

Related work

We first review previous work on Graph convolutional Network and ResNetThe residual block structure is shown in Figure 1, followed by a review of recent developments in training deep GCN.

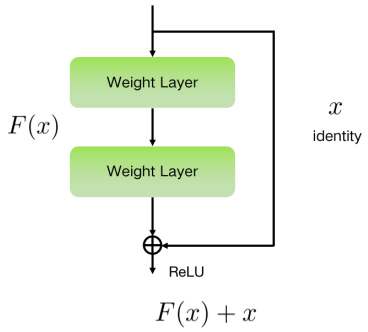

Figure 1: Residual block structure: The output of the residual structure is expressed as F(x) + x. Compared with using multiple stacked nonlinear layers to directly learn the identity mapping F(x) = x, the residual structure directly learns F(x) = 0 so that training is easier.

Graph Convolutional Networks. Kipf et al. propose the Graph Convolutional Networks (GCNs) to define convolutions on the non-grid structures. GCN is a method based on deep learning that can perform convolution operations on the graph. Compared with traditional CNN, GCN has a unique convolution operator for irregular data structures. Generally, GCN can be divided into two categories: spectrum-based GCN and non-spectrum-based GCN. The latter attempts to expand the spatial definition of convolution by rearranging the vertices of the graph into a specific grid form so as to be directly applicable to traditional convolution operations, while the former uses Fourier transform to perform the convolution process. Generally, spectral GCN can handle graphs with fixed topology well, while non-spectral GCN can handle graphs with topological changes. Graph Convolutional Networks (GCNs). Current GCN algorithms can be divided into two categories: spectral-based and spatial-based. Based on spectral graph theory, Bruna et al. [2013] firstly developed graph convolutions using the Fourier basis of a given graph in the spectral domain. Later, many methods are proposed to apply improvements, extensions, and approximations on spectral-based GCNs [Kipf and Welling, 2016, Defferrard et al., 2016, Henaff et al., 2015, Levie et al., 2018, Li et al., 2018]. On the other hand, spatial-based GCNs [Hamilton et al., 2017, Monti et al., 2017a, Niepert et al., 2016, Gao et al., 2018, Xu et al., 2019b] define graph convolution operations directly on the graph, by aggregating the information from neighbor nodes. To address the scalability issue of GCNs on large-scale graphs, there are mainly two categories of scalable GCN training algorithms: sampling-based [Hamilton et al., 2017, Chen et al., 2018a, Li et al., 2018, Chen et al., 2018b, Zeng et al., 2020] and clustering-based [Chiang et al., 2019].

ResNet. Nowadays, CCN faces many difficulties (e.g. vanishing gradient and limited receptive field). We bridged this gap and showed that most of these drawbacks can be compensated by borrowing a technique from CNN. Introducing ResNet, the performance of deep CNN is greatly improved. By adding residual connections between the input and output of the layer, ResNet tends to eliminate the vanishing gradient problem. ResNet's residual block adds a shortcut connection between one or more convolutional layers. This shortcut connection will act as an identity mapping to solve the vanishing gradient and exploding gradient. ResNetV2 performs a ReLU operation after the addition to achieve nonlinear activation. Without this ReLU operation, the residual block output is always non-negative, which restricts the expressive ability of the model. ResNeXt adopts the idea of grouped convolution and controls the number of groups through variable cardinality, making the network structure more efficient. SKNet proposes the idea that different images can get convolution kernels of different importance. Therefore, SKNet uses different convolution kernel weights for different images, that is, a dynamic generation of convolution kernels for images of different scales. ResNeSt proposes a modular Split-Attention block, which can disperse attention to several feature map groups, which can be used directly for downstream tasks without adding additional calculations. DenseNet proposes a more radical dense connection mechanism, which connects all layers to each other to achieve feature reuse and improve efficiency.

Training Deep GCNs. Despite the rapid and fruitful progress of GCNs, most previous art employ shallow GCNs. Several works attempt different ways of training deeper GCNs [Hamilton et al., 2017, Armeni et al., 2017, Rahimi et al., 2018, Xu et al., 2018]. All these works are however limited to 10 layers of depth before GCN performance would degrade. Inspired by the benefit of training deep CNN-based networks [He et al., 2016a, Huang et al., 2017, Yu and Koltun, 2016], many researches seek to solve the vanishing problem of deepGCN. A further obstacle to train deeper GCNs is over-smoothing, Recent works focus on addressing this phenomenon [Klicpera et al., 2019, Rong et al., 2020, Zhao and Akoglu, 2020]. Klicpera et al. [2019] proposes a PageRank-based message passing mechanism, involving the root node in the loop. Alternatively, DropEdge [Rong et al., 2020] proposes randomly removing edges from the graph, and PairNorm [Zhao and Akoglu, 2020] develops a normalization layer.

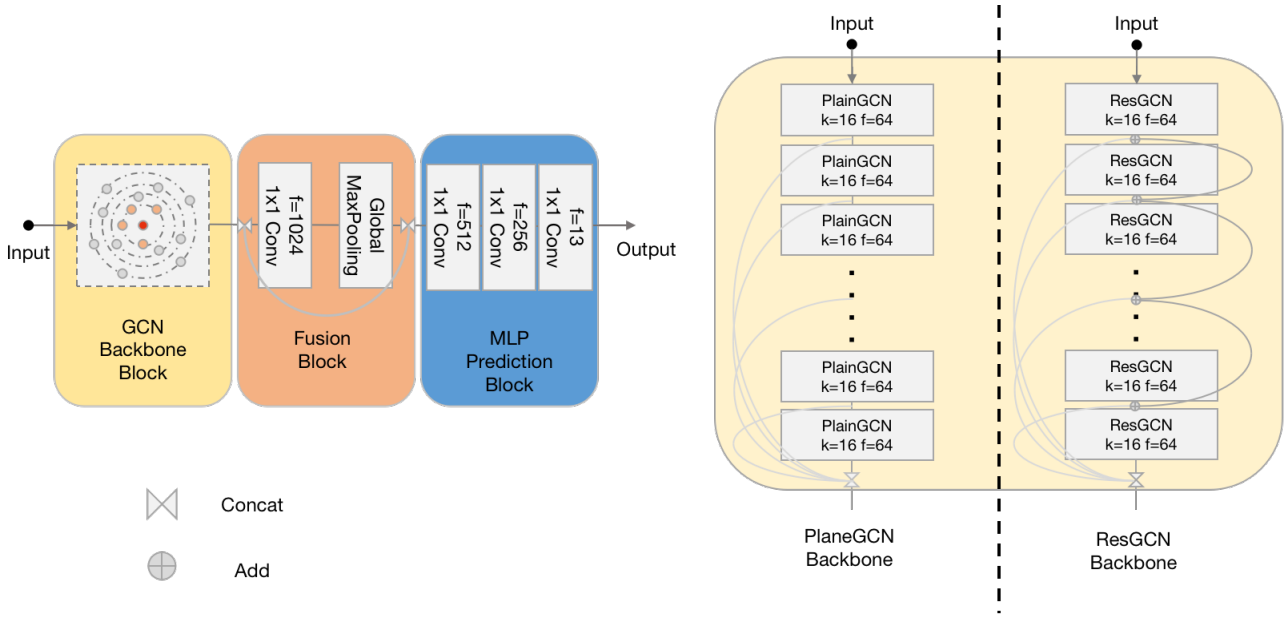

Figure 2: (left) Our framework consists of three blocks: a GCN Backbone Block , a Fusion Block, and an MLP Predic- tion Block . (right) Two types of GCN Backbone Block

Proposed Solution

Graph Definition

A Graph $\mathcal{G}$ could be represented by a tuple with a set of unordered vertices and a set of edges: $(\mathcal{G} = \mathcal{V}, \mathcal{E})$. We denote vertices v_i and v_j are connected when $e_{i,j} \in \epsilon$.

Graph Convolution Network

GCNs also extract high-level features at a vertex by fusing features of vertices from its neighborhood, like CNNs. Given a feature vector $h_c \in \mathcal{R}^D$ for a vertex v, we can regard the graph G as the concatenation of features of all the unordered vertices, i.e. $\mathbf{h}_\mathcal{G} = [\mathbf{h}_{v_1}, \mathbf{h}_{v_2}, \ldots, \mathbf{h}_{v_N}]^\top \in R^{N \times D}$, where N is the number of set $\mathcal{V}$ and D is the feature dimension. So a standard graph convolution could be formulated as the combination of aggregation and update operations like this:

$$\mathcal{G}_{l+1} = \mathcal{F}(\mathcal{G}_l, \mathcal{W}_l) = Update\left(Agg(\mathcal{G}_l, \mathcal{W}_l^{agg}), \mathcal{W}_l^{update}\right)$$

$\mathcal{G}_l$ and $\mathcal{G}_{l+1}$ are the input graph and output graph at layer i, W_l^{agg}, W_l^{agg} are the learnable weights. In most GCN networks, update functions means a non-linear transform on the aggregated information, which used to calculate new vertex representations, while aggregation functions are used to fuse features from the neighborhood of vertices.

There are different variants of those two functions, e.g. mean aggregator, max-pooling, attention aggregator or an LSTM aggregator for aggregation function, and MLP, gated network for opdate function.

For all $v_{l+1} \in \mathcal{V}_{l+1}$, the representation of vertices is calculated by aggregating features from its neighbor vertices at each layer:

$$\mathbf{h}_{v_{l+1}} = \phi\left(\mathbf{h}_{v_l}, \rho\left(\{\mathbf{h}_{u_l} \mid u_l \in \mathcal{N}(v_l)\}, \mathbf{h}_{v_l}, \mathcal{W}_\rho\right), \mathcal{W}_\phi\right)$$

where ρ is the aggregation function for feature aggregation and ϕ is the function to update vertex features at layer l-th and $l+1$-th. $\mathcal{N}(v_l)$ and h_{u_l} are the set of neighbor vertices of v and neighbor vertices parametrized by $\mathcal{W}_p$, respectively. $\mathcal{W}_\phi$ contains the weights of these functions. Without generality, we can use max-pooling as the feature aggregation function, to pool the difference of features between vertex v_l and all of its neighbors: $\rho(.) = \max\left(\mathbf{h}_{u_l} - \mathbf{h}_{v_l} \mid u_l \in \mathcal{N}(v_l)\right)$. We model the feature update operation ϕ as a MLP with BatchNorm and ReLU, then its input is formed by aggregating features from $\rho(.)$.

Dtnamic Edges

Recent work shows that dynamic graph can learn better graph representations compare to GCNs with fixed structure, which only update the vertex features at each iteration. For example, ECC(Edge-Conditioned Convolution) uses dynamic ECC to learn edge-specific weight matrix. EdgeConv reconstruct the graph after every EdgeConv layer by finding the nearest neighbors Graph-Convolution GAN applies k-NN graphs for learning point clounds by constructing the neighbourhood of each vertex in each layer.

Our experiments shows that dynamically changing neighbors in GCNs could help alleviate the over-smoothing and thus results in an effective and larger receptive field when GCNs going deeper. In our framework, we recompute edges between vertices to increase the receptive field, via a Dilated kNN function.In the following content, we provide a detailed description of the operation that can enable deeper GCN training: residual connections.

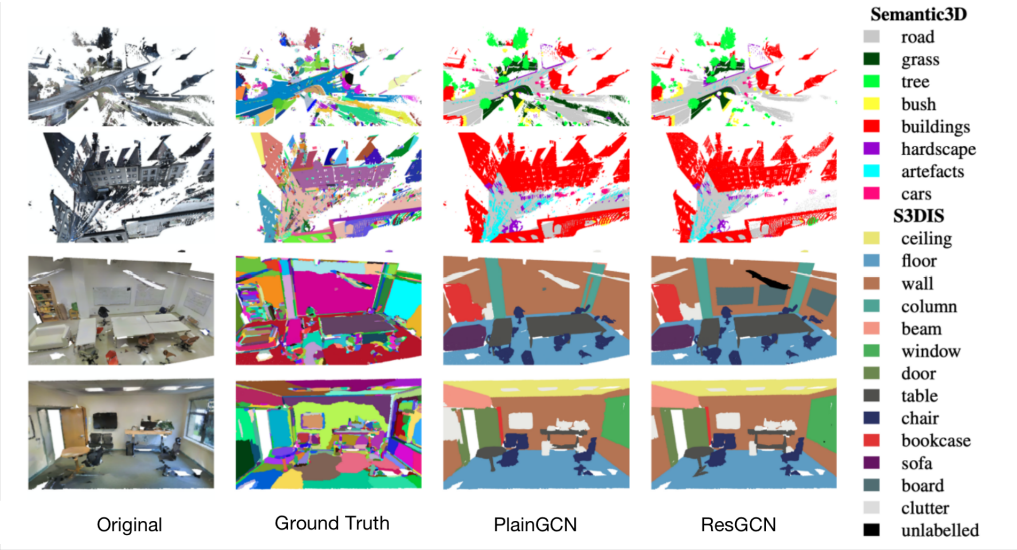

Figure 3: Experiment Result

Residual Learning for GCNs

How to design a deeper GCN architecture is also an open problem. Recent work shows that GCNs do not perform well on deep architectures, one reason might be stacking multiple layers of graph convolutions would result in extremely high complexity in back-propagation. As a result, there are no more than 3 layers deep in most state-of-art models. Inspired by the success of ResNet, we use residual structure to GCNs to improve its ability of learning representation. This could make GCNs deeper and reliably converge in training, thus achieve superior performance in inference.

In the original graph learning framework, we need to learn the underlying mapping $\mathcal{F}$. Here, we learn an underlying mapping $\mathcal{H}$ by fitting another mapping $\mathcal{F}$. Vertex-wise addition is applied to obtain $\mathcal{G}_{l+1}$ after transforming G_l by $\mathcal{F}$. The residual mapping $\mathcal{F}$ learns to obtain the redisual output graph representation $\mathcal{G}^{res}_{l+1}$ by taking a graph as input, where $\mathcal{W}_l$ is the set of weights at l-th layer, we refer our model as ResGCN:

$$\mathcal{G}_{l+1} = \mathcal{H}\left(\mathcal{G}_l, \mathcal{W}_l\right) = \mathcal{F}\left(\mathcal{G}_l, \mathcal{W}_l\right) + \mathcal{G}_l = \mathcal{G}^{res}_{l+1} + \mathcal{G}_l$$

Experiments

We propose ResGCN to handle the vanishing gradient problem of GCNs. To evaluate our framework, we conduct experiments on the task of point cloud segmentation and demonstrate that our methods can improve performance.

Graph Learning on 3D Point Clouds Point cloud segmentation is a challenging task because of the unordered and irregular structure of 3D point clouds. Normally, each point in a point cloud is represented by its 3D spatial coordinates and possibly auxiliary features such as color and surface normal. We treat each point as a vertex v in a directed graph G and we use k-NN to construct the directed dynamic edges between points at every GCN layer. In the first layer, we construct the input graph $\mathcal{G}_0$ by executing a k-NN search to find the nearest neighbor in 3D coordinate space. At subsequent layers, we dynamically build the edges using k-NN in feature space. For the segmentation task, we predict the categories of all the vertices at the output layer.

Network Architectures As shown in Figure 2, all the network architectures in our experiments have three blocks: a GCN backbone block, a fusion block and an MLP prediction block. The GCN backbone block is the only part that differs between experiments. For example, the only difference between PlainGCN and ResGCN is the use of residual skip connections for all GCN layers in ResGCN. Both have the same number of parameters. For fair comparison, we keep the fusion and MLP prediction blocks the same for all architectures. In the S3DIS semantic segmentation task, the GCN backbone block takes as input a point cloud with 4096 points, extracts features by applying consecutive GCN layers to aggregate local information, and outputs a learned graph representation with 4096 vertices. The fusion block is used to fuse the global and multi-scale local features. It takes as input the extracted vertex features from the GCN backbone block at every GCN layer and concatenates those features, then passes them through a 1×1 convolution layer followed by max pooling. The latter layer aggregates the vertex features of the whole graph into a single global feature vector, which in return is concatenated with the feature of each vertex from all previous GCN layers (fusion of global and local information). The MLP prediction block applies three MLP layers to the fused features of each vertex/point to predict its category.

- PlainGCN: This baseline model consists of a PlainGCN backbone block, a fusion block, and a MLP prediction block. No skip connections are used here.
- ResGCN: We construct ResGCN by adding dynamic k-NN and residual graph connections to PlainGCN. These

Method	OA	mIOU	ceiling	floor	wall	beam	column	window	door	table	chair	sofa	bookcase	board	clutter
PointNet	78.5	47.6	88.0	88.7	69.3	42.4	23.1	47.5	51.6	54.1	42.0	9.6	38.2	29.4	35.2
PointNet++	-	53.2	90.2	91.7	73.1	**42.7**	21.2	49.7	42.3	62.7	**59.0**	19.6	45.8	48.2	45.6
DGCNN	84.1	56.1	-	-	-	-	-	-	-	-	-	-	-	-	-
ResGCN-28 (*Ours*)	85.9	**60.0**	**93.1**	**95.3**	**78.2**	33.9	**37.4**	**56.1**	**68.2**	**64.9**	51.0	**34.6**	**51.5**	**51.1**	**54.4**

Table 1: **Comparison of ResGCN-28 with other Semantic Segmentation methods on S3DIS.** We compare ResGCN-28 with other Semantic Segmentation methods on S3DIS. It shows that out model performs better than other methods. The metrics shown are overall point accuracy (OA) and mean IoU (mIoU).

connections between all GCN layers in the GCN backbone block do not increase the number of parameters.

Implementation We implement all our models using Tensorflow. For fair comparison, we use the Adam optimizer with the same initial learning rate 0.001 and the same learning rate schedule;the learning rate decays 50% every 3×10^5 gradient decent steps. The batch size is set to 8 for each GPU. Batch Normalization is applied to every layer. Dropout with a rate of 0.3 is used at the second MLP layer of the MLP prediction block. We train our models end-to-end from scratch.

Result Figure 3 shows results on S3DIS. As expected from the results in Table 1,Our ResGCN-28 perform well on these situation. Rows 1-4 clearly show how ResGCN-28 are able to segment the board, beam, bookcase and door respectively, while PlainGCN-28 fails.

Conclution

In this paper, we investigate how to expand GCNs with the residual connections and expore how to make GCNs be deeper.Then, extensive experiments show that by adding skip connections to GCNs, we can alleviate the difficulty of training, which is the primary problem impeding GCNs to go deeper. Finally, draw a conclusion that by using residual block structure, we can get a deeper GCN mdoel.

References.

[1] L. Tang and H. Liu. Relational learning via latent social dimensions. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 817–826. ACM, 2009. 1, 2

[2] M. Zitnik and J. Leskovec. Predicting multicellular function through multi-layer tissue networks. Bioinformatics, 33(14):i190–i198, 2017. 1, 2

[3] N. Wale, I. A. Watson, and G. Karypis. Comparison of descriptor spaces for chemical compound retrieval and classifi- cation. Knowledge and Information Systems, 14(3):347–375, 2008. 1, 2

[4] F. Monti, M. Bronstein, and X. Bresson. Geometric matrix completion with recurrent multi-graph neural networks. In Advances in Neural Information Processing Systems, pages 3697–3707, 2017. 1, 2

[5] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 974–983. ACM, 2018. 1, 2

[6] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell. Understanding convolution for semantic segmentation. In 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), pages 1451–1460. IEEE, 2018.8

[7] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. IEEE Data Engineering Bulletin, 40(3):52–74, 2017b.

[8] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In International Conference on Learning Representations (ICLR), 2016.

[9] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Advances in Neural Information Processing Systems (NIPS), pp. 1025–1035, 2017a.

[10] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In International Conference on Learning Representations (ICLR), 2017.

[11] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In International Conference on Learning Representations (ICLR), 2018.

[12] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In International Conference on Machine Learning (ICML), pp. 5453–5462, 2018.

[13] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In International Conference on Machine Learning (ICML), pp. 1273–1272, 2017.

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

[15] Simon Jegou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In Workshop on Computer Vision in Vehicle Technology CVPRW, 2017.

[16] Jonas Gehring, Michael Auli, David Grangier, and Yann N. Dauphin. A convolutional encoder model for neural machine translation. CoRR, abs/1611.02344, 2016. URL http://arxiv.org/abs/1611.02344.

[17] Michael Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In Advances in Neural Information Processing Systems, pp. 3844–3852, 2016.

[18] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. arXiv preprint arXiv:1506.05163, 2015.

[19] Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. IEEE Transactions on Signal Processing, 67(1):97–109, 2017.

[20] F. Monti, M. Bronstein, and X. Bresson. Geometric matrix completion with recurrent multi-graph neural networks. In Advances in Neural Information Processing Systems, pages 3697–3707, 2017. 1, 2

[21] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In International conference on machine learning, pages 2014–2023, 2016.

[22] Yilun Chen, Zhicheng Wang, Yuxiang Peng, Zhiqiang Zhang, Gang Yu, and Jian Sun. Cascaded pyramid network for multi-person pose estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 7103–7112, 2018.

[23] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese. Joint 2D-3D-Semantic Data for Indoor Scene Understanding. ArXiv e-prints, Feb. 2017. 2, 7

[24] A. Rahimi, T. Cohn, and T. Baldwin. Semi-supervised user geolocation via graph convolutional networks. arXiv preprint arXiv:1804.08049, 2018. 2

[25] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? arXiv preprint arXiv:1810.00826, 2018. 11, 12

[26] 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, pages 770–778, 2016. 1, 2, 4, 6