Compressing GNNs with Knowledge Distillation

Abhinay Vundyala, Naman Jain, Kalpit Munot

1 Abstract

Knowledge distillation approaches have mainly ignored graph neural networks (GNN), which deal with non-grid data. Knowledge distillation concentrate on convolutional neural networks (CNNs) where the input samples are in a grid domain, such as photos. We have implemented the dedicated strategy of distilling knowledge from a pre-trained GNN model. The proposed method is a local structure preserving module that explicitly accounts for the topological semantics of the teacher GNN in order to facilitate knowledge transfer from the teacher GNN to the student. In this module, both the teacher and the student's local structure information is extracted as distributions, and by computing similarity scores between these distributions, topology-aware knowledge transfer from the teacher is enabled, resulting in a compact yet high-performance student model. We evaluated our model on Protein-Protein Interaction Dataset (PPI), which is commonly used for node classification tasks. We achieved 20x compression with our student model with a minor drop of 2.24% accuracy (from 98.07% to 95.83%)

2 Introduction

Deep learning effectively captures hidden patterns of Euclidean data, but there are an increasing number of applications where graph representations of data are used. An example of this can be an e-commerce application where a graph-based learning system exploits the relations between users, products and builds accurate recommendation systems. Graph Learning systems are also useful in chemistry, where molecules are modeled as nodes, and the chemical bonds between them are modeled as edges that aid in the development of new drugs. In research, articles are linked to each other by citing each other, and they need to be categorized into different groups. In the real world, graph data has become more complex, and existing machine learning algorithms are not able to model them. Since graphs could be irregular, they can contain a variable number of unordered nodes, and nodes from the same graph might have a varying number of neighbors, making some essential operations (such as convolutions) straightforward to compute in the picture domain but challenging to perform in the graph domain. Furthermore, one of the underlying assumptions of existing machine learning methods is that instances are independent of each other. Because each instance (node) is linked to others by various forms of linkages, such as citations, friendships, and relations, this premise no longer holds true for graph data.

The Lottery Ticket Hypothesis postulates that there exists a subnetwork within fully connected networks that are able to outperform its "parent" in training time, performance, and generalisability. Specifically, these subnetworks won the initialization lottery, which is shown to be especially important for their success. Through a process of knowledge distillation, these subnetworks can be uncovered. Our hypothesis is that training a large GNN model, stopping the training early, and then heavily compressing it will have better results than training a small GNN model, stopping training when converged, and compressing it slightly.

Methods based on graph neural networks (GNNs) [5,7,8] have shown their usefulness in identifying node labels, thanks to the success of deep learning. Most GNN models use a message passing strategy [6]: each node aggregates features from its neighborhood, and then the aggregated information is projected using a layer-wise projection function with a non-linear activation. GNNs may use both graph structure and node feature information in their models in this fashion. However, because natural networks are dense and have irregular degrees, using a pre-trained model to properly solve prediction tasks on them can be computed and memory intensive. As a result, it makes sense to do model compression and acceleration in dense networks without sacrificing performance. Deep neural networks have made remarkable development in this field during the last five years. We hope to develop a solution to compress dense Graph Neural Networks in this study. Parameter pruning and quantization, low-rank factorization, transferable convolutional filters, and knowledge distillation are the four types of compression approaches.

Knowledge distillation's effectiveness has been proved in a variety of tasks where the student's performance is practically identical to that of the teacher. Despite the tremendous progress, earlier knowledge distillation methods relied on convolutional neural networks (CNNs), which use grid-based input samples like photos. However, because most real-world data, such as point clouds, is represented by non-grid structures like graphs, graph attention networks (GATs) are required [8,6,7,5]. GATs explicitly probe the data's topological structure by assessing the graph's local and global semantics. As a result, traditional knowledge distillation procedures only account for output or intermediate activation while neglecting the topological context of input data can no longer transmit information entirely.

We present a specific knowledge distillation strategy for GATs in this study. For pre-trained teacher GAT, our objective is to train a student GAT model with fewer layers, lower-dimension feature maps, or even a smaller graph with fewer edges. The ability to store the topological information hidden in the graph, which is missing from previous CNN-based approaches, is at the heart of our GAT distillation. The proposed technique takes into account node properties as well as topological connections between them, providing the student model with deeper and more essential information about the teacher's embedded topological structure.

We demonstrate the workflow of the proposed GAT knowledge distillation technique. To explicitly account for the graphical semantics, we construct a local structure-preserving (LSP) module. The LSP module, given the embedded feature of node and the graph from teacher and student, assesses the topological difference between them and leads the student to a topological embedding that is equivalent to the teacher's. LSP, in particular, creates a distribution for each local structure from both the student and the teacher and then pushes the student to learn a comparable local structure by reducing the gap between the distributions.

To analyze the effectiveness of knowledge distillation, we evaluate our model on node classification task for the Protein-Protein Interaction dataset. Experiments show that this method consistently achieves a good amount of compression with a minor decrease in accuracy.

Our contributions are summarized as follows:

• We implement a knowledge distillation technique for model compression of Graph Attention network.

• We extend the local structure-preserving (LSP) method into the graph neural network literature to measure the similarity of the local topological structures embedded by teacher and student.

• We compare the student model and teacher model with tuning the hyper-parameters of the graph attention network

3 Background

Knowledge distillation was initially proposed as a neural network compression strategy that reduces the KL-divergence between the output logits of teacher and student networks. When compared to discrete labels, the relative probabilities predicted by the teacher network tend to encode semantic commonalities between categories, which are crucial for student network learning. For model compression, knowledge distillation was developed, in which a tiny light-weight student model is taught to imitate the soft predictions of a big teacher model that has been pre-trained. The knowledge from the instructor model will be transmitted to the student model after distillation. The student model can therefore decrease time and spatial complexity while maintaining prediction quality. In the field of computer vision, knowledge distillation is commonly utilized; for example, a deep convolutional neural network (CNN) would be compressed into a shallow one to speed up inference.

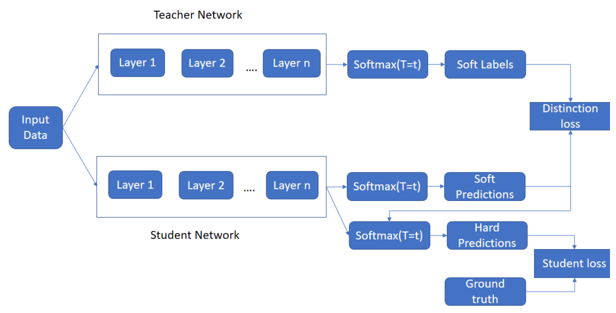


Figure 1: Knowledge Distillation in Deep Neural Networks

Graph Attention Network The Graph Attention Network is a semi-learning approach that learns directly from a node's spatial information. This is in contrast to the Graph Convolutional Network's spectral method, which uses the same basic principles as the Convolutional Neural Net. The Graph Attention Layer is the GAT's fundamental building piece. The relative relevance of neighboring features is determined by their attention coefficients. Multi-head attention is used to increase the learning process's stability. We create a number of distinct attention maps before combining all of the learned representations.

Graph Convolutional Network A Graph Convolutional Network (GCN) is a graph-structured data semi-supervised learning approach. It is based on a speedy convolution neural network variant that works directly with graphs. A local 1st order approximation of the spectral network motivates the use of convolutional architecture.. The model learns hidden layer representations that capture both local graph structure and node properties as the number of network edges increases linearly.

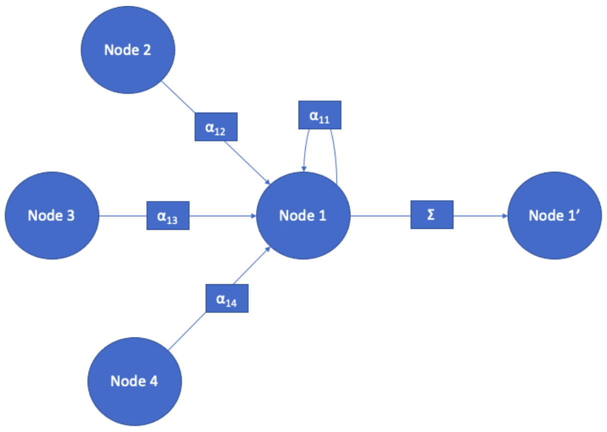


Figure 2: Attention Concatenation in Graph Attention Network

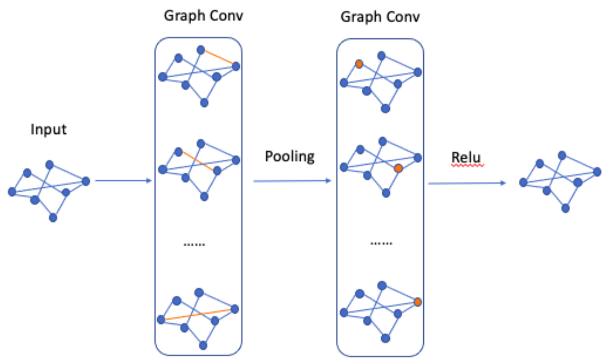


Figure 3: Multi-layer Graph Convolutional Network (GCN) with ReLu Filters

Kernel Functions for GNN In Graph Neural networks, spatial information of each node heavily determines the node features. During the training phase, Node features are determined by aggregating the information passed by its neighbor nodes. Compressing the graph attention network means compressing the pre-trained node feature vectors, and the inference of the compressed model will be heavily compromised if the spatial information of nodes is not preserved. Hence, preserving the local structure of every node is necessary when compressing the node feature vectors. Local structure-preserving (LSP) should be enabled while using any of the compression techniques described above. To determine the effect of each neighbor node, similarity scores can be generated in a pairwise fashion for all nodes in the graph. Kernels are a good choice to compute similarity scores, i.e., if f(a, b) $>$ f(a, c), it implies the distance between a and b in the graph is less compared to the distance between a and c.

4 Design

In this section, we first give a brief description of the aggregation phenomenon that is important for Graph Attention Networks, followed by the motivation for using Knowledge Distillation of this aggregation function. We explain the local structure-preserving model, which is the core formula for our design, and how we integrate it with the Knowledge distillation method.

4.1 Distilling Aggregation Function

Graph Neural Networks usually take non-grid data as input. Non-grid data used for Graph Neural Networks is generally represented as a set of features X=$\{x_1$,x_2,x_3,...$x_n\} \in \mathrm{R}^f$, also including a directed or undirected graph G=$\{V$,$E\}$. For example, in Cora there is single graph in which nodes and edges correspond to documents and citation links, respectively. A sparse feature vector (document keywords) and a class label are associated with each node. Several splits of these datasets are used in the node classification

For given input G and X, the core operation of the Graph Attention Network is

$$x_i^{'} = \Sigma_{k=1}^{k} \sigma(\Sigma_{j \in \varepsilon} \alpha_{ij}^{k} W^k x_j)$$

K refers to the attention heads used in the network, and ε refers to all the edges that node i is connected to.

A novelty that Graph Attention Network brings over Graph Convolution Network is how the information from the one-hop/multi-hop neighborhood is aggregated. In this model, we use multi-attention heads for teacher and student networks. The aggregation function for multi attention heads can follow concatenation or averaging the attentions received. We concatenate the attention in all intermediate layers and average them in the final layer.

The proposed Network is mainly derived from the fundamentals of Graph Attention Network. Aggregation Strategy plays an important role in embedding the features to nodes during the training process. So, using knowledge distillation, we aim to provide the student model, the information about the aggregation function the teacher has already learned. By doing this, we can mimic the training process of the teacher on the student. However, distilling the knowledge that exactly represents the aggregation function and transferring that knowledge to the student is very tedious and compute-intensive. So, instead of distilling the aggregation function, we distill the outcomes of the aggregation function, i.e., the embedded topological structure of the graph around each node. After this step, the student model can be guided by matching the structure embedded by itself and the structure embedded by the teacher model. We will show in the

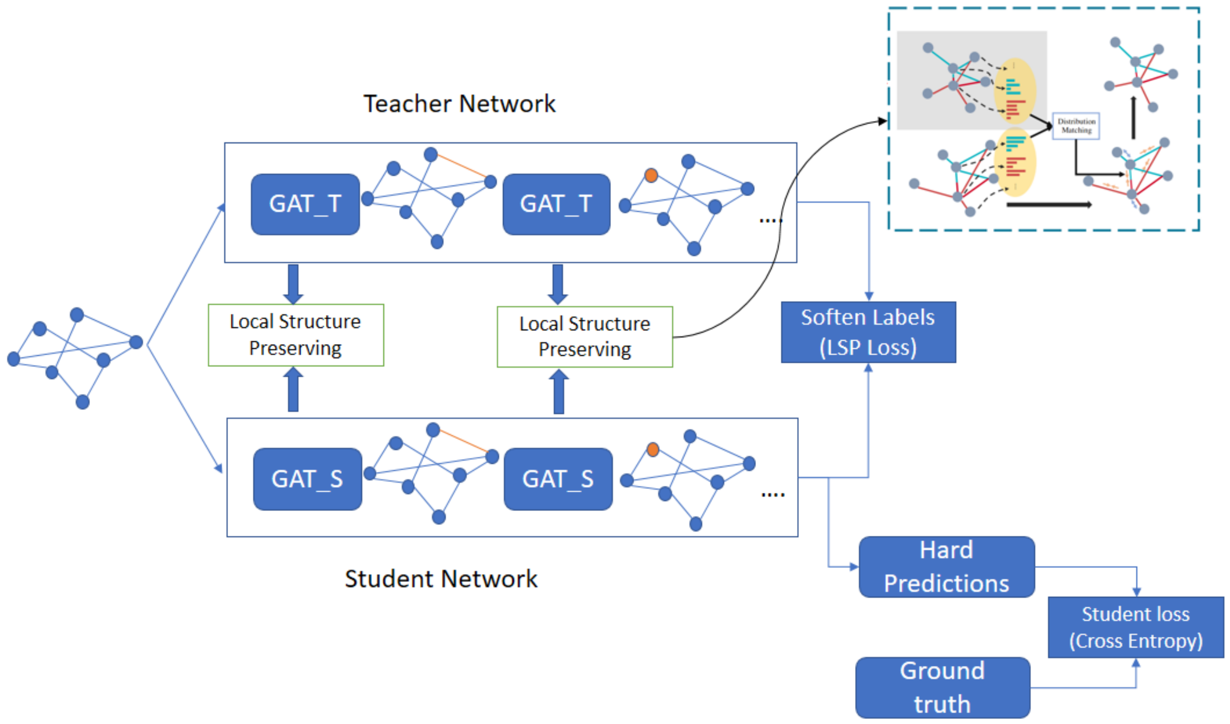


Figure 4: Knowledge Distillation of Aggregation Function Outcomes by matching distributions between Student and Teacher Networks

following section how to secure the topological structure information and distill it to the student.

4.2 Knowledge Distillation with LSP

We have a teacher GAT network as well as a student GAT network in the context of knowledge distillation, with the teacher network being trained and fixed. We first perform a local structure-preserving technique when each of these networks is given the same graph as input but have different layers and dimensions of embedded features.

We may represent the intermediate feature maps of a GAT as a graph with the notation G=$\{V$,$E\}$ and features with P=$\{p_1$,p_2,p_3,...$p_n\}$ where n is the number of nodes. Local Structure can be represented as a set of vectors S=$\{s_1$,s_2,s_3,...$s_n\}$ where the $s_i \in R^d$ and d is the degree.The following formula is used to calculate each element of the vector:

$$S_{i,j} = \frac{e^{\delta(p_i,p_j)}}{\Sigma_{j\in\varepsilon}(e^{\delta(p_i,p_j)})}$$

$$\delta(p_i, p_j) = p_i.p_j$$

where δ is a function that measures the similarity of the given pair of nodes, which can be defined as a dot product between the two features. We use an exponential operation to normalize the values of all nodes pointing to the local structure's center. Using this, for each node i, we can obtain its corresponding local structure S.

We can compute the local structure vectors for both the teacher and student networks, as S^s and S^t, using the intermediate feature maps. The similarity of the local structure between the student's and the teacher's may be computed for each center node i by using Kullback Leibler divergence as follows:

$$S_i = D_{KL}(S_i^s||S_i^t) = \Sigma_{j\in\varepsilon} S_{ij}^s \log\frac{S_{ij}^s}{S_{ij}^t}$$

A smaller S_i indicates that the local structural distribution is more similar. As a result, we compute the similarity of the distributions over all nodes of the provided graph and derive the local structure preserving loss as follows:

$$L_{LSP} = \frac{1}{N}\Sigma_i(S_i)$$

The total loss can be represented as:

$$L = H(p_s, y) + \lambda L_{LSP}$$

where, y is the label, and p_s is the student model's prediction, λ is the hyperparameter used to balance these two losses, and H is the cross-entropy loss function used by many other knowledge distillation methods.

5 Experiments

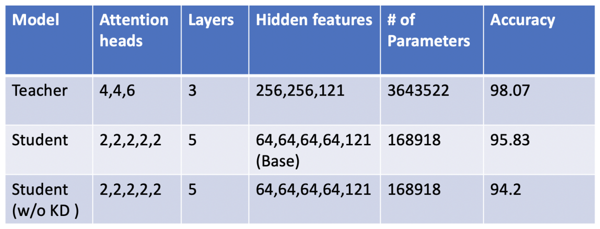
In this section, we first give a brief description of the experimental setup, including the datasets we used and the Graph Attention Network models. Our goal is not to achieve state-of-the-art performance on all the datasets or tasks but to transfer as much knowledge from the teacher model to the student model. A good evaluation of this would be a comparison of student model performance with teacher model performance and also with student GAT without learning.

We evaluate our model using Protein-Protein Interaction Dataset. PPI Dataset contains 24 graphs corresponding to different human tissues. We follow the dataset splitting protocol, wherein 20 graphs are used for training, two graphs are used for testing, and another two graphs are used for validation. Each node can belong to 121 classes, so the output labels for this dataset are 121. The average number of nodes in the subgraph is approximately 2200, and these subgraphs are trained using teacher and Student GAT.

Our task for this project is Node classification. Generally, in the node classification task, input nodes and their features are provided along with the graph. Also, since this is a multi-label task, cross-entropy loss better suits training with ground truth labels.

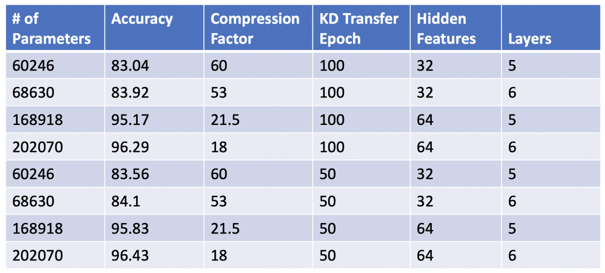
Our model is able to achieve 96% accuracy with a 20x compression factor. There is only a drop of 2% when compared to the teacher model. We also compared our student model performance without knowledge transfer, and the results are in the table.

As shown in Figure 6, we have run experiments with changing hidden units in student GAT layers and changing the number of layers in student GAT. There is a linear relation between accuracy and increasing layers or the number of hidden units. Hidden features are kept constant among all the layers. The experiments are only performed with changing student GAT model, and the pre-trained teacher GAT model is kept constant. KD epoch refers to the number of epoch the knowledge transfer has been performed. We also have run experiments while changing this parameter.



Model	Attention heads	Layers	Hidden features	# of Parameters	Accuracy
Teacher	4,4,6	3	256,256,121	3643522	98.07
Student	2,2,2,2,2	5	64,64,64,64,121 (Base)	168918	95.83
Student (w/o KD)	2,2,2,2,2	5	64,64,64,64,121	168918	94.2

Figure 5: Student Model w/o KD refers to the network which is trained only with ground truth labels



# of Parameters	Accuracy	Compression Factor	KD Transfer Epoch	Hidden Features	Layers
60246	83.04	60	100	32	5
68630	83.92	53	100	32	6
168918	95.17	21.5	100	64	5
202070	96.29	18	100	64	6
60246	83.56	60	50	32	5
68630	84.1	53	50	32	6
168918	95.83	21.5	50	64	5
202070	96.43	18	50	64	6

Figure 6: Experiment Results with Changing Hidden Units per Layer and Changing Hidden Layers

6 Related Work

There has been a lot of work done in Neural networks for compression with techniques like Knowledge Distillation, Pruning, Weight Quantization, Low-rank approximation, and Neural Architecture Search.

Pruning Pruning a trained network involves removing connections between neurons or entire neurons, channels, or filters by dropping out values in its weights matrix or removing groups of weights entirely; for example, to prune a single link from a network, one weight in a weights matrix is set to zero, and to trim a neuron, all attributes of a column in a matrix are set to 0. Pruning is motivated by the fact that networks are often overparametrized, with numerous features encoding roughly the same data. Non - structured pruning removes individual weights or neurons, but structured pruning removes whole channels or filters.

Knowledge Distillation In 2006, Cornell researchers presented the notion of distilling knowledge from a big trained model (or group of models) to a smaller model for deployments by training it to replicate the larger model's output, a method that Hinton et al. generalized in 2014. In brief, knowledge distillation is inspired by the notion that training and inference are two distinct tasks that require separate models.

Weight Quantization While pruning reduces the amount of weights in a model, and quantization reduces the magnitude of the weights that are already present. In general, quantization is the process of transferring values from a big set to values in a smaller set, with the output having a narrower range of potential values than the input, preferably without sacrificing too much information.

Low-rank approximation Low-rank approximation aims to simulate a layer's many redundant filters with a linear combination of lesser filters. Layer compression minimizes the memory footprint of the network as well as the computational complexity of convolutional operations, resulting in considerable speedups.

Neural architecture Search Neural architecture Search is a systematic, automated approach to learning optimum model designs by searching over a collection of decisions that determine the distinct components of a neural network. The goal is to eliminate human bias from the process in order to create unique architectures that outperform those created by humans.

However, none of the above-mentioned methods have been thoroughly explored in the Graph neural network domain due to the constraints which arise from the graphical datasets. There has been some work done in the Knowledge Distillation [1][10] and Pruning[3][4] .The field is still unexplored, considering we have other methods as well which could be used for compression.

7 Future work/ Work in progress

1. We are analyzing the trade off between attention heads and number of hidden layers in Graph Attention Network.

2. We can use set of student models to build a more powerful student model using ensemble learning.

3. Our Experiments currently are performed on Node Classification using Knowledge distillation and local structure preserving. We can analyze our model on graphs containing different relations and perform tasks like link prediction or graph classification.

8 Conclusion

We implemented a compression technique for graph neural network using knowledge distillation. The aggregation approach in GNN is critical for incorporating the node characteristics that are learned during the training process. However, distilling information that accurately describes the aggregation function and transferring it to the student is difficult. This is accomplished by preserving the teacher network local structure during the training process. The local structure of the intermediate feature maps is represented as distributions over the similarities between the local structure's center node and its neighbors, therefore preserving the local structures is equal to matching the distributions. Experiments on the PPI dataset demonstrate that the method yields compression of 20x with a minor drop of 2% accuracy.

References

[1] Sheng Zhou, Yucheng Wang, Defang Chen, Jiawei Chen, Xin Wang, Can Wang, Jiajun Bu. *Distilling Holistic Knowledge With Graph Neural Networks*

[2] Junghun Kim,Jinhong Jung,U. Kang. *Compressing deep graph convolution network with multi-staged knowledge distillation*

[3] Mingyang Zhang, Xinyi Yu, Jingtao Rong, Linlin Ou. *Graph Pruning for Model Compression*

[4] Sixing Yu, Arya Mazaheri, Ali Jannesari. *GNN-RL Compression: Topology-Aware Network Pruning using Multi-stage Graph Embedding and Reinforcement Learning*

[5] Will Hamilton, Zhitao Ying, and Jure Leskovec. *Inductive representation learning on large graphs. In Advances in neural information processing systems. 1024–1034.*

[6] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. *Neural message passing for Quantum chemistry. In Proceedings of the 34th International Conference on Machine Learning-Volume 70. 1263–1272.*

[7] Thomas N Kipf and Max Welling. *Semi-supervised classification with graph convolutional networks. In 5th International Conference on Learning Representations.*

[8] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. *Graph Attention Networks. In International Con- ference on Learning Representations.*

[9] Geoffrey Hinton, Oriol Vinyals, Jeff Dean. *Distilling the Knowledge in a Neural Network*

[10] Tianqi Chen, Ian Goodfellow, Jonathon Shlens. *Net2Net: Accelerating Learning via Knowledge Transfer*

[11] Jian Du, Shanghang Zhang, Guanhang Wu,Jose' MF Moura, and Soummya Kar. *Topology adaptive graph convolutional networks. arXiv preprint arXiv:1710.10370.*

[12] Kim J, Jung J, Kang U. *Compressing deep graph convolution network with multi-staged knowledge distillation. PLOS ONE 16(8): e0256187.*