PIC2GRAPH: GRAPH-BASED IMAGE CLASSIFICATION OF BREAST CARCINOMA SUBTYPES

Shaan Gill, George Ye & Sichen Zhu
Group 12
Georgia Institute of Technology
Atlanta, GA 30332, USA
{sgill36,gye31,sichenzhu}@gatech.edu

ABSTRACT

Recent advances in machine learning have facilitated quick development in digital pathology. Traditional machine learning tools like residual neural network in computer vision have been successfully adopted to improve cancer diagnosis in a pixelwise-based manner. However, in any machine learning applications in the medical field, interpretability is always the top priority. In order to better understand pathology images from a whole-image point of view, Graph Neural Network is introduced into the field of digital pathology due to its power to capture underlying structures behind an image. In this project, we explore the potential of graph neural network in classifying breast cancer types from H&E stained histology images, and compare them with traditional machine learning methods in computer vision. Through this adventure, we hope to gain empirical understandings of different graph-based models and test their power in real-world application. Although graph-based models did not outperform other deep learning methods, their comparable performance shows that it is still promising for graph neural network to provide us with a more interpretable way of applying machine learning tools to the medical field, given their ability of extracting and analyzing features based on the whole image rather than pixels.

1 INTRODUCTION

The problem our group aims to tackle is classification of breast tissue images for breast carcinoma using a graphical representation for the images and GNN models for prediction. Specifically, our group will focus on identifying different sub-types of atypical lesions that are usually an early indication into malignant breast cancer. Accurate classification of these lesion images is crucial for early diagnosis and can lead to different actions that patients can take to significantly improve their outcomes.

Recent advances in machine learning have been tailored to digital pathology for faster and better detection and diagnosis of diseases. Traditionally, Convolutional Neural Networks (CNNs) have been an effective tool in tackling image classification. Despite of its success in applications, CNNs are limited by their fixed connectivity and inefficiency in capturing neighborhood information Ahmedt-Aristizabal et al. (2021). Recent development in Graph Neural Networks (GNNs) enables us to capture the potential structures and correlations behind a image and free us from the constraints of Cartesian grid and image pixels.

In our project, we convert H&E histology images to graph representations and use GNNs as a potential alternative to image classification. Graph offers a flexible and efficient data structure, and by converting images to graphs, we may be able to capture unique features and patterns that would have been difficult to catch using traditional methods. Furthermore, GNNs have had great success in working with graph-based data such as social networks and drug discovery, we would like to explore their potential for more accurate and/or efficient models with image classification.

In our experiment, though graph-based algorithms did not outperform traditional tools in computer vision, their performance is at least comparable to pixelwise machine learning models. We still hold promises that under fine tuning of model architecture and hyperparameters, graph neural network is a more interpretable tool for medical image classification.

2 Related Works

Graphs offer a flexible and efficient way to model data. The nodes of a graph can represent key features in the data and an adjacency matrix can represent edges along with weights for relationship between different features. A very common application for GNNs is graph classification, where given graph $\mathcal{G}$ and its label y, the model learns to classify different graph structures Zhang et al. (2018). In this project, we test the following prevalent graph models on graph classification task:

2.1 Graph Convolutional Network (GCN)

The idea of a graph neural network is to compress a set of messages coming from neighborhood into a single vector, and pass this aggregated message to the next node. The idea of graph convolutional operator is firstly proposed in paper ”Semi-supervised Classification with Graph Convolutional Networks” Kipf and Welling (2016). Given an undirected graph $\mathcal{G}$ with N nodes $v_i \in \mathcal{V}$, a node feature matrix X, edges $(v_i, v_j) \in \mathcal{E}$, an adjacency matrix $A \in \mathbb{R}^{N \times N}$ (either binary or weighted), and a degree matrix $D_{ii} = \sum_j A_{ij}$, layer-wise propagation is defined as:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}) \tag{1}$$

where $\tilde{A} = A + I_N$ is the adjacency matrix added with self-loops, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ is the degree of graph $\mathcal{G}$ with self-loops, $W^{(l)}$ is the weight matrix in the l^{th} layer, $H^{(l)} \in \mathbb{R}^{N \times D}$ is the matrix of features in the l^{th} layer, $H^{(0)} = X$.

It is worth mentioning that any graph neural network must be equivariant to the permutation of adjacency matrix as well as invariant to the ordering of input nodes. Thus, the aggregation of messages from each node is usually operation that is not sensitive to the node ordering, such as mean, max, min, etc. In GCN Kipf and Welling (2016), the aggregation is simply the sum over all the messages from neighbors and apply activation function (as we could see from $H^{(l)}W^{(l)}$ in equation 1).

By message passing algorithms defined as above, we could get updated node embeddings that encode underlying graph structure. Then we use a global aggregation function to gather all node embeddings to generate a graph-level output. This aggregation function also needs to satisfy the property of being invariant and equivariant.

2.2 Graph Attention Network (GAT)

Attention model has achieved great success in natural language processing. One benefits of attention mechanism is that they are able to deal with different sizes of input Veličković et al. (2017). GAT, or graph attention networks, is introduced by Velickovic *et al.* in their paper ”Graph Attention Networks”. It uses attention to learn the representation of the graph. As an initial step, a shared linear transformation parametrized by weight matrix W is applied to all node. A shared operator a computes attention coefficients (a scalar) that indicate the importance of node j's feature to node i:

$$e_{ij} = a(W\vec{h}_i, W\vec{h}_j) \tag{2}$$

where $\vec{h}_i$ and $\vec{h}_j$ are node features for node i and node j, respectively.

The coefficients are normalized across all choices of node j using softmax function:

$$\alpha_{ij} = \text{softmax}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})} \tag{3}$$

The operator a could be parametrized using a single layer of neural network $\vec{a}$ with LeakyReLU ($\sigma(\cdot)$) activation function ($\alpha = 0.2$). Plugging the new notation into Eq.3:

$$\alpha_{ij} = \frac{\exp\left(\sigma\left(\vec{a} \cdot \text{CONCAT}(W\vec{h}_i, W\vec{h}_j)\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\sigma\left(\vec{a} \cdot \text{CONCAT}(W\vec{h}_i, W\vec{h}_k)\right)\right)} \tag{4}$$

Final output for updated node feature $\vec{h}_i'$ is:

$$\vec{h}_i' = \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij} W \vec{h}_j\right) \tag{5}$$

For multi-head attention, the output of feature representation is simply the concatenated output from all heads. Then we can apply graph level readout to perform the graph classification on.

2.3 GraphSAGE

GraphSAGE provides an inductive way to learn node embeddings for graphs. In particular, GraphSAGE learns a node embedding function that iteratively aggregates messages from a node local neighborhood as well as previous messages on the node itself. The authors tested different aggregator functions for the model, and the model will iteratively update and learn the weight matrices and parameters for the aggregator function that will produce the best embeddings. For each iteration, the node will gather more and more information about its neighbors. Because GraphSAGE focuses on learning the topological features of a graph, it can generalize very well to create embeddings for unseen graphs or unseen nodes Hamilton et al. (2018). For the purposes of graph classification, we added a readout function to obtain an overall graph-level representation and add MLP classifier after the embedding layers for classification.

2.4 Graph Isomorphism Network (GIN)

GIN or a Graph Isomorphism Network is a way to extract features from a graph. They are inspired by the Weisfeiler-Lehman (WL) test and are at least as powerful as the WL test in terms of distinguishing non-isomorphic graphs Xu et al. (2019). The main idea of the algorithm is to use node and edge embeddings and transform them into graph-level embeddings. This is done by first, using a message-passing framework in which neighboring nodes apply an aggregation function of their neighbors to update their representation. Then the model propagates this information across the graph using an MLP to get each node representation. This MLP in propagation makes the aggregation scheme in GIN injective, which means it would generate unique representations for different neighborhoods in the graph. This makes GIN powerful than other GNN models using a fixed, predefined aggregator function. Finally, a readout function can be used to obtain some kind of graph level representation Xu et al. (2019). The architecture also allows for different graph input sizes which is helpful for comparing many graphs as they tend not to be the same shape.

3 Dataset Description

For our implementation, we will leverage the BReAst Carcinoma Subtyping (BRACS) dataset Brancati et al. (2021). The dataset contains 387 Whole Slide Images (WSI) of breast tissues collected from 151 patients. Of the 387 WSIs, there are 4539 Regions of Interest (RoIs) which are annotated with seven different classes. Of these 7 different classes, there are 6 different atypical lesion subtypes along with images representing normal tissue samples as well. Specifically, we have the following lesion subtype breakdown:

Normal	Benign Tumors	Atypical Tumors	Malignant Tumors
Normal (N)	Pathological Benign (PB)	Flat Epithelial Atypia (FEA)	Ductal Carcinoma in Situ (DCIS)
	Usual Ductal Hyperplasia (UDH)	Atypical Ductal Hyperplasia (ADH)	Invasive Carcinoma (IC)

Table 1: 7 Classes in BRACS dataset

More specific details about our usage of this dataset is covered in section 5.

4 Methods

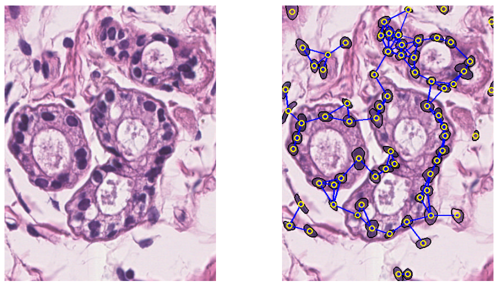
4.1 Traditional Way of Image Classification

For traditional image classification, we used prior successful pre-trained CNNs models that have been used for image classification and had good results. We used RESNET-18 He et al. (2015), RESNET-50, and EfficentNET Tan and Le (2019) as our baseline models in this project. To be consistent with the models we choose to train these CNNs only using files that were less than 10MB in size. The models were tested using a variety of hyperparameters such as LR, batch size, optimizer, lrschduler, and weight decay. The best hyperparameters were kept and used as the baseline. We trained the models by searching the hyperparameter space for all 3 of the given models. This procedure was then carried out for binary classification and the best model was used as the baseline.

4.2 Image Classification through Graphs

4.2.1 Image to Cell Graph Conversion

In order to apply graph machine learning techniques, we need to transform histology images into a graph representation. In general, node in those graphs can be defined by biologically meaningful entities like cell nuclei, tissue regions, patches surrounding the region of interest, etc. In this project, we build cell-graphs from Region of Interests (ROI) images in BRACS dataset, where each cell is a node in the cell-graph. Cells are first detected from those histology images by applying machine learning tools. We used Hover-Net in this project, which is a pre-trained residual network specialized in nuclei segmentation Graham et al. (2019). It contains a pre-activated residual unit and three branches of dense unit for different tasks.



(a) Original Image (b) Cell Graph

Figure 1: Example of cell graph constructed from histology image. All nuclei is circled out by black, while yellow circles are the "centroids" of nuclei. Blue edges are the connectivities detected by kNN algorithm.

In a cell-graph, node features can be hand-crafted features including morphological and topological properties of a cell such as size, shape, and nuclei intensity Ahmedt-Aristizabal et al. (2021). However, in order to leverage the power of neural networks, we followed procedures in this paper Jaume et al. (2021) and used a resnet34 model from torchvision.models to extract visual attributes from images as our node features. After we had cells at hand, k-nearest neighbors algorithm (k=5) was performed to detect cell clusters, draw edges between cells that satisfy the distance threshold, and compute corresponding adjacency matrix. Each node feature has a dimension of 514. At this point, we are able to apply suitable graph neural networks to classify our cell-graphs based on node features. An example of cell graph is shown in Fig.1

In order to further capture the spatial information provided by images, we defined edge features as the inverse of Euclidean distance between cells (edge feature is only defined for those edges identified by kNN algorithm). Mathematically, the edge weight between cell c_i and cell c_j is calculated by $\|c_i - c_j\|_2^{-1}$, where c_i and c_j are the cell locations in the image. We hope that under this circumstance, edges in our cell-graph could encode the proximity between neighbor cells, carrying information about cell group structure that might be useful for tumor classification. We performed graph classification on cell-graphs both with and without edge weight. Detailed results are shown in section 5.

4.3 Graph Classification

Image classification falls into the category of graph-level prediction in GNN, which gives one output for an entire graph. We tested four prevalent GNN models in this project: GCN Such et al. (2017), GAT Veličković et al. (2017), GraphSAGE Hamilton et al. (2018),GIN Xu et al. (2018a), in the hope of comparing their performance to find out the most suitable architecture for our graph classification task. Apart from different models, we also tested the influence of whether edge weights is defined in cell-graphs, as well as different aggregation functions for graph readout.

4.3.1 GCN, GAT and GraphSAGE

We used a basic architecture for graph classification task, which is composed of three parts: 1) node embedding; 2) graph readout; 3) classification layers. Node embedding involves two graph convolutional layers (GraphConv/GATConv/SAGEConv from torch_geometric.nn for GCN, GAT, GraphSAGE respectively). Between two graph convolutional layers, there is an activation function (ReLU() for GCN and GraphSAGE; ELU() for GAT) and dropout with probability 0.5. The GAT convolutional layer has 7 heads with a smaller hidden dimension each head, in order to keep total hidden dimensions similar size as GraphConv and SAGEConv. Graph readout is extracted by global_mean_pool() function from torch_geometric.nn. Final classifier based on graph readout contains 2 linear layers. The dropout rate in between is also 0.5. Training was done by torch.optim.Adam with a 0.001 learning rate without any weight decay. Loss function for training is cross entropy loss. Each model was trained for 50 epochs. We kept all the other settings and parameters exactly the same in order for comparison. After the training, we picked the model with lowest validation error for downstream testing.

4.3.2 GIN

As shown in related works, Graph Isomorphism Network (GIN) provide another powerful method for obtaining graph-level representation of node features. We will follow a similar model to the GIN in Xu et al. (2019) using GINConv from torch geometric for our layers. We used 2 GINConv layers and a combination of Linear, BatchNorm, and ReLU layers for our MLP within the GINConv layer. Once the embeddings are obtained, we used a graph readout extracted using global_mean_pool, global_add_pool, or global_max_pool. Since our goal is graph classification, we added an additional two layer MLP with dropouts after obtaining the readouts, for classifying the graph for either binary or multi classification (7 class).

In addition, we also tested out using Jumping Knowledge (JK) with GINs as well. In the above work, we took the embedding of the last layer with all the embedded information and used that for the graph level read out and classification. The idea behind Jumping Knowledge is to concatenate embeddings from all (or at least some) previous layers in hopes that information from previous layers can improve on the structural information for each node Xu et al. (2018b). In our GIN model, we concatenated the previous layers together into a single embedding to pass into the graph readout for the graph classification.

5 Experiments & Results

We set up two sets of experiments given the BRACS dataset: 1) a binary classification: including images from normal and benign (N + PB + UDH) as one class, and malignant tumors (DCIS + IC) as another class; 2) a 7-class classification involving all classes in the dataset. Image sizes in this dataset have a large variance, ranging from hundreds KB to 200+ MB. We limited our images in training, validation and testing to those below 10MB due to limited space quota. The owners of this dataset have already split it into train/validation/test folds based on their own criterion, thus we did not combine or re-split those three folds in our own experiments. Numbers of images (< 10 MB) in each class is summarized in Table 2. Our experiments are aimed to compare multiple GNN models to the state-of-the-art baseline image classification models based on their classification accuracy.

Train/Validation/Test Splits (< 10MB)				
Category	Train	Validation	Test	Total
N	206	30	65	301
PB	333	33	53	419
UDH	316	41	67	424
FEA	594	37	79	710
ADH	325	37	67	429
DCIS	338	11	59	408
IC	132	1	32	165
Total	2,244	190	422	2856

Table 2: Number of Images (< 10MB) in Each Classes

For our baseline, we ran the following models and hyperparameters over both experiments. The best run of these hyperparameters are noted in the tables below. All models were trained using the PyTorch library and utilizing the models along with the pre-trained weights from them.
The hyperparameters used in the Table 3 were: Loss Function: Cross Entropy Loss, Optimizer: Adam, LR: .001, Scheduler step size: 5, Scheduler gamma: .75, batch size: 128, epochs: 30.

Model	**Train** (%)	**Test** (%)	**Validation** (%)
EfficientNET	99.77	87.68	91.38
RESNET-18	99.02	86.96	90.52
RESNET-50	99.77	87.68	91.38

Table 3: Best baseline results for Binary Classification

The hyperparameters used in Table 4 were: Loss Function: Cross Entropy Loss, Optimizer: Adam, LR: .003, Scheduler step size: 5, Scheduler gamma: .75, batch size: 128, epochs: 25.

Model	**Train** (%)	**Test** (%)	**Validation** (%)
Efficientnet	88.32	55.82	45.79
RESNET-18	66.34	51.07	43.68
RESNET-50	75.75	52.49	51.58

Table 4: Best baseline results for 7-class Classification

Here are the GNN models and their best performances on the binary classification task. We also tested different graph readout functions (global_max_pool, global_add_pool, global_mean_pool) and picked the one with the best performance:

Model	**Train** (%)	**Test** (%)	**Validation** (%)	**Graph Readout**
GCN	95.25	83.33	95.69	mean
GCN with edge weight	96.15	82.61	94.83	mean
GAT	94.94	82.97	93.10	mean
GraphSAGE	95.47	83.07	94.83	mean
GIN	97.36	83.33	93.97	mean
GIN w/ JK	99.55	82.25	94.83	mean

Table 5: Best results using Graph for Binary Classification

Similarly, here are the GNN models and their best performances and readout functions on the 7-class multi-classification task:

Model	**Train** (%)	**Test** (%)	**Validation** (%)	**Graph Readout**
GCN	68.76	44.55	57.89	mean
GCN with edge weight	78.97	46.92	56.84	mean
GAT	60.92	38.15	42.63	mean
GraphSAGE	79.68	47.16	53.68	mean
GIN	85.52	42.42	47.37	mean
GIN w/ JK	96.57	41.94	43.68	mean

Table 6: Best results using Graph for 7-class Classification

6 Discussion

In terms of test accuracies, all of the baselines outperformed all of the graph models we tested. However, some of the models beat out the baselines in terms of validation accuracy however the margins were small. Among all the graph models, GraphSAGE had the best performance for 7-class classification while GCN had the best for binary classification. Something interesting to look at is the training accuracies for all the models. Overall CNN models are able to overfit on the training set much more than the graph models used. This shows that image nets are very easily able to overfit on the training set. Modifying the hyperparams such as weight decay/batch size to combat this had little to no change in reducing this overfitting. Some of the graph models were also overfitting however the gap performance gap between train and val/test is very large and perhaps using more methods to bridge this gap would allow the models to perform better.

Some experiments were also carried out on the image CNNs and we found that using all images (not limiting file size) had little to no difference in performance. Additionally, the differences in hyperparameters did not seem to affect performance by more than 5 points. The overall run-time of the CNNs after restricting the file size to 10MB had drastically dropped by close to tenfold (depending on the model used).

After limiting image size in the training/validation/test dataset, we found that we lost most of the images in IC class. This dataset also had large gaps between test and validation accuracy which is uncommon and shows this dataset may not be of the highest quality. This is probably a reason for models not performing well in 7-classes classification. Another thing to consider about this dataset is that when doing 7-class classification some of the categories are quite similar since each of the categories are part of a group (benign: N + PB + UDH, atypical: FEA + ADH, and malignant: DCIS + IC). For this dataset it might be worth using a custom loss function that gives a lower loss if the model guesses inside the group. This could help increase the model's performance since it will get a higher loss when its prediction is out of the group and lower when it is.

7 Conclusion

In this paper, we developed and tested a methodology to perform image classification using GNNs. We used resnet34 getting important attributes in the image and k-nearest neighbors to create cell clusters to represent as our node features. We also used the inverse Euclidean distance between cells for edge features to get graphical representations of the images. We tested multiple GNNs on classifying these graphs using different graph readout functions and found that the current state-of-the-art image classification models outperformed our GNNs, however the margin was not large. While not beating the state-of-the-art, GNNs still perform well and provide some useful utility such as not being able to take a variety of image sizes into the same model.

Some future work that can be done in this area:

1. Improving the image to graph conversion:
   (a) We can look to learn on more representative edge features other than Euclidean distance.
   (b) Running resnet and k-nearest neighbors can you computationally expensive; we can look to explore more efficient ways without sacrificing performance.
2. Improving the GNN models:
   (a) While we tested multiple models and different readout functions, we can look to try deeper models and different hyperparameters such as learning rate.
   (b) There are also more state-of-the-art GNN models that can be tested with the similar pipeline such as ChebNet.
3. While we found that the current state-of-the-art image classification models perform better than GNNs, we can look to compare the efficiency in between GNNs and CNNs based on number of parameters and runtime.

8 CONTRIBUTIONS

Each group member contributed equal amounts of work for researching into possible implementations and related works for graphical representations of images and GNN classification models, developing the ML pipeline for pre-processing the data, training the model, and testing the classification accuracy, as well as writing this final report.

9 DATA & CODE AVAILABILITY

The BRACS dataset could be accessed through official website: https://www.bracs.icar.cnr.it/. All the code are publicly available at https://github.com/george-ye45/CSE8803_Final_Project.

REFERENCES

David Ahmedt-Aristizabal, Mohammad Ali Armin, Simon Denman, Clinton Fookes, and Lars Petersson. 2021. A Survey on Graph-Based Deep Learning for Computational Histopathology. *CoRR* abs/2107.00272 (2021). arXiv:2107.00272 https://arxiv.org/abs/2107.00272

Nadia Brancati, Anna Maria Anniciello, Pushpak Pati, Daniel Riccio, Giosuè Scognamiglio, Guillaume Jaume, Giuseppe De Pietro, Maurizio Di Bonito, Antonio Foncubierta, Gerardo Botti, Maria Gabrani, Florinda Feroce, and Maria Frucci. 2021. BRACS: A Dataset for BReAst Carcinoma Subtyping in HE Histology Images. arXiv:2111.04740 [q-bio.QM]

Simon Graham, Quoc Dang Vu, Shan E Ahmed Raza, Ayesha Azam, Yee Wah Tsang, Jin Tae Kwak, and Nasir Rajpoot. 2019. Hover-net: Simultaneous segmentation and classification of nuclei in multi-tissue histology images. *Medical Image Analysis* 58 (2019), 101563.

William L. Hamilton, Rex Ying, and Jure Leskovec. 2018. Inductive Representation Learning on Large Graphs. arXiv:1706.02216 [cs.SI]

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385 (2015). arXiv:1512.03385 http://arxiv.org/abs/1512.03385

Guillaume Jaume, Pushpak Pati, Behzad Bozorgtabar, Antonio Foncubierta, Anna Maria Anniciello, Florinda Feroce, Tilman Rau, Jean-Philippe Thiran, Maria Gabrani, and Orcun Goksel. 2021. Quantifying explainers of graph neural networks in computational pathology. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 8106–8116.

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

Felipe Petroski Such, Shagan Sah, Miguel Alexander Dominguez, Suhas Pillai, Chao Zhang, Andrew Michael, Nathan D Cahill, and Raymond Ptucha. 2017. Robust spatial filtering with graph convolutional neural networks. *IEEE Journal of Selected Topics in Signal Processing* 11, 6 (2017), 884–896.

Mingxing Tan and Quoc V. Le. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *CoRR* abs/1905.11946 (2019). arXiv:1905.11946 http://arxiv.org/abs/1905.11946

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

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018a. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks? arXiv:1810.00826 [cs.LG]

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018b. Representation Learning on Graphs with Jumping Knowledge Networks. *CoRR* abs/1806.03536 (2018). arXiv:1806.03536 http://arxiv.org/abs/1806.03536

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An End-to-End Deep Learning Architecture for Graph Classification. *Proceedings of the AAAI Conference on Artificial Intelligence* 32, 1 (Apr. 2018). https://doi.org/10.1609/aaai.v32i1.11782