Adaptive Graph Convolutional Neural Network and its Biomedical Applications

by

RUOYU LI

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

Dec 2020

To my beloved family.

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to my academic advisor Dr. Junzhou Huang, without whom this thesis would not have been possible. His irreversible fortitude and constant motivation are the true reasons for the successful outcomes of my research on deep graph learning. I sincerely express my gratitude to Dr. Dajiang Zhu, Dr. Vassilis Athitsos and Dr. Yu Lei for their valuable time spent on serving my defense committee.

Novermber 17, 2020

ABSTRACT

Adaptive Graph Convolutional Neural Network and its Biomedical Applications

RUOYU LI, M.S.

The University of Texas at Arlington, 2020

Supervising Professor: Dr. Junzhou Huang

As the rise of graph neural networks, many deep learning frameworks have been extended to graph-structured data. The research in many diverse regimes have been tremendously reshaped, especially in areas like medical image understanding. When input data reach the scale of whole slides images (WSIs), the modeling becomes more challenging and we have to balance the trade-off between performance and efficiency. Furthermore, the theory of existing graph convolution has its own constraints which prevent learning robust graph representation on data that has diverse topological structure and are infeasible for graph sampling or coarsening. To tackle the problems we introduced a series of novel graph neural networks and techniques. For example, Adaptive Graph Convolutional Network (AGCN) [1] combined the graph representation learning with graph learning and empower the network to learn features from hidden substructures on graph. Attentional-AGCN [2] provided a *node-to-graph* attention scheme which does not only improve graph representation trained over large-scale images like WSIs, but also facilitate an intuitive explanation of the effectiveness of Attentional-AGCN. Besides, we introduced an end-to-end survival prediction framework based on WSI input directly, which delivered significant improvement on accuracy of survival analysis. Lastly, to mitigate the randomness introduced by patch sampling, a fast region-of-interest (RoI) search and detection approach is also introduced to images at WSI scale. MROID [3] combined classification with detection into a unified framework to quickly narrow down candidates RoI proposals and gave a coarse-to-fine boundary refinement for generated RoIs. As summary, in this thesis, we reviewed and introduced our contributions to graph neural networks and the applications of our methods on large-scale biomedical data understanding tasks.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

Introduction

As we all know that the Convolutional Neural Networks (CNNs) have been proven tremendously successful on a wide range of machine learning problems. The success of convolutional neural networks is rooted on its biotic simulation of how human eye and brain process optical information. As early as [5], researchers have been attempting to link the convolutional neural network model with the nature of how retina and brain respond to natural optical stimulations. Light comes into eyeball, stimulates optic fiber and generates electrical signals, then the nerve cells aggregates the signal from its connected neurons and pass it to the linked cells of later layer. We have many different kinds of nerve cells to execute different type of signal processing. Further human brain makes sophisticated decision based on aggregated information from bottom neurons (i.e. retina fiber cells).

If we use a neural network to model the entire response of optical signal inside human retina and use forward-pass in neural network to model the electrical signal passing between nerve cells, then, for a classic CNNs shown in Figure 1.1, the input image, decomposed as RGB (red, green, blue) 3 channels, would be the original

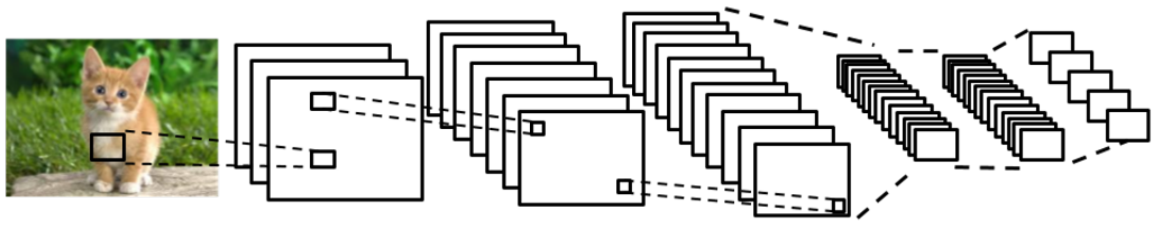


Figure 1.1: A classic Convolutional Neural Network with 2-D image as input.

electrical stimulation given by retina optical fiber. Then, the signals are going to be aggregated by convolution operator defined by filter with fixed shape, 3×3 in example of Figure 1.1. At the end of convolutional layer, each output channel turns out to be the summation of channels of input signals. This setup is to allow convolution operator in CNNs to not only aggregate neighborhood pixels, but also learn dependency from other channels. Back to retina cell layers, there exist similar layer-wise connection between nerve cells at different layers. A nerve cell is to read multiple signals from others before generating its own signal that is going to pass on. If we see CNN as a simulation of this process, then we might also see pixels from feature map as neurons and the convolutional filters as a definition of signal processing pattern among a sort of nerve cells. In real-world neuroscience, the nerve connection and signal processing are sophisticated, while in the configuration of convolutional layers in CNNs, the combination of different aggregation patterns (i.e. convolutional filters) is a simple summation.

There have been a huge amount of publications that elaborate the representation power that CNN has in terms of better performance in downstream use cases, e.g. classification, detection, tracking. We discussed the constraints of existing CNNs in this section.

Figure 1.1 showed an example of convolution on grid and its 3×3 convolutional filter. The training job of network is to learn 3×3 parameter matrices for the kernel and when summing up neighbor features using this kernel, the parameters would be multiplied to corresponding neighbors first. While the training cannot modify the network architecture and configurations. If the initial status of kernel is as large as 3×3, it would stay that shape and size during the entire training time. And this setup would be applied to all feature mapping between channels of input and output. It means, for all the convolutional kernel in a certain layer of CNN, the kernel size is identical. For example, if the input channel cardinality is c_i and output channel count is c_{i+1}, and we define the kernel in this layer is of size 3×3, then the total parameter count of this layer is $c_i \times c_{i+1} \times 3 \times 3$. While, clearly, it does bring extra limitation on how we design networks, but it also releases concern of computational cost by large-scale training data and large batch size. Given fixed, identical kernel size, the back-propagation makes full use of tensor multiplication to accomplish gradient calculation and parameter update within minimal overhead. However, from neuroscience study, the human retina cells have a more complicated and flexible cell connectivity and signal aggregation patterns beside simple summation. Current CNNs are an oversimplified version of retina cell layers, then we argue if this setup lead to efficient learning of information from input. And stacking layers makes network capable of fitting better representation, while this comes at the price of tremendously increasing computational cost and difficulty of training. Apparently, human eyes do not need to stack hundreds of layer of nerve cells just to recognize a flying bird. There must be some better representation could be constructed from flexible kernel and channel connections, while CNNs are unable to model it.

Another constraint comes to the inputs of convolutional layer. The classical CNNs have requirement of regularly shaped tensors as input [5]. For instance, images [6] and videos [7] are respectively modeled as 2-D and 3-D tensors when being fed to a traditional convolutional neural network. As opposite to this stereotype, in many real-world applications, it is more likely to encounter the data deployed on an irregular grid for instead and more generally in non-Euclidean space. Instead of regularly shaped tensors, those data are intuitively more appropriate to be seen as graph-structured, which make it convenient to handle the varying neighborhood vertex connectivity as well as the non-Euclidean metrics. Under these circumstances, the stationarity and the compositionality, which empower convolutional operator to work on multidimensional grid, does no longer exist on graph-structured data. Consequently, it is necessary to reformulate convolutional operator to make it compatible with graphs.

As summary, existing CNNs have following technical constraints which block them from working with graph-structured data:

1. Kernel size and shape are fixed and identical within layers across all channel maps.
2. Input data have to be of identical dimensionality.
3. Data have been deployed on regularly shaped, equally-spaced grid.

Based on above observations, it is necessary to redesign convolutional operator on graph to make convolutional neural networks trained on graph-structured data. Aforementioned constraints of existing CNNs would become the three problems that the proposed new convolutional layer is going to fix:

1. How to design sharable kernel/convolutional filter by all data across channel maps ?
2. How to make network accept input graph data of varying size and shape ?
3. How to control the training complexity (i.e. scale of trainable parameters) of the network ?

Spectral CNN [8] implemented a linear spectral graph kernel together with the eigenvectors after eigen-decomposition of graph Laplacian $L = U\Lambda U^T$. While, the eigen-decomposition takes $\mathcal{O}(N^3)$ time complexity, even though the training complexity (i.e. scale of parameter number) is linear w.r.t. graph size N. As the first successful trial of spectral-based graph convolution kernel, $g_\theta * x = Udiag(U^T\theta)U^T x$, while it comes with three major drawbacks:

1. Graph is fixed. While any change on graph L would make eigenvector U changes accordingly. Consequently, any change on graph will makes the parameters θ useless for a new graph or an unseen graph.

2. If you train the network on a graph structure L, then test data must be deployed on same graph topology. Otherwise it may not work. Which makes the model hard to be generalizable. When you meet a new graph structure after training, spectral CNN cannot work on it. For example, your training data are all graphs of less than 100 nodes, what if when inference, there is a graph has 101 nodes.
3. Parameter complexity, for each graph, we need to learn a new set of parameters, what if we have lots of different graph structures, making training complexity high. Time complexity: the eigen-decomposition is highly expensive. It has cubed N complexity. Makes it not feasible for large graphs. Like social graph.

Following the appearance of Spectral CNN [9], ChevNet [10] and GCN [11] approximated the kernel function as Chebyshev polynomials of diagonal eigenvalue matrix Λ, and trainable parameter θ is assigned to each polynomial item $T_i(\hat{L})$. From the formulations in [10], the eigendecomposition is avoided by converting $T_i(\hat{\Lambda})$ to $T_i(\hat{L})$, based on $L = U\Lambda U^T$. If we assume graph is sparse, then the multiplication of L with node vectors X has less complexity than $\mathbf{O}(N^2)$. It is obvious that ChevNet [10] and its variants have solved problem 2&3, while it does not solve the problem of "fixed graph", which prevent the network to learn from the hidden substructures and cannot deal with the scenarios where graph initialization is poor.

Even though the data has intrinsic graph and it is static, would it be better to make it dynamic or even adaptive with respective to predictive tasks ? And more importantly, the new model has to control the scale of training complexity as training graph is non-trivial with limited number of parameters, free of graph scale. Besides, Spectral CNN[9] indeed has one unique merit that constructs and learns independent filters $\theta_{i,j}$ for each channel pair between input and output. Therefore, it means that each feature channel of output tensor is summed from all input feature channels. Intuitively, spectral CNN and classic CNNs do not only aggregate the features of same channel on neighbor nodes, but also further aggregate other channels, to build cross-channel dependency. However, ChevNet and its variants forget about the advantage of classic CNNs. At ChevNet, each output channel was learned independently from other channels.

For most real-world graph data, their structures vary in terms of both scale and topology. A generalizable convolutional operator on graph is supposed to be compatible with different graph topology. In the article, authors introduced a generalized and flexible GCN framework along with a new spectral graph convolutional layer (SGC-LL) parameterizing distance metric and feature transform. Besides original graph structure, a residual graph is constructed and learned throughout training, Therefore, the introduced Adaptive Graph Convolutional Network (AGCN) is adaptive to graphs of arbitrary topological structure and scale and is also adaptive to various learning tasks easily. Extensive experiments of AGCN and state-of-the-arts on drug property benchmark datasets have demonstrated the superior performance improvement on both convergence speed and predictive accuracy for multi-tasks classification of graph, which becomes mathematical modeling of drug or any chemical compounds.

With the success of graph attention network (GAT), we enabled AGCN to learn graph representation from a bag of patches randomly sampled from large medical images such as Whole-slide-images (WSIs) for sophisticated image understanding tasks, e.g. survival prediction. However, to learn the graph representation that better emphasizes on the patches from lesion and its surroundings instead of the ones from irrelevant regions of WSIs. Because the expensive labeling work by professionals, in many applications, no patch level labels could be provided along the training data, which requires the model itself to be able to learn the weight assigned to each patch during the training towards prediction task, for example the survival risk regression in the thesis.

The DeepGraphSurv is an end-to-end framework that directly predict survival probability from patients' WSI on tissues. Empowered by graph attention, an intuitive annotation of tumor cells is also learned and generated by the model. Extensive experiments of AGCN and state-of-the-arts on drug property benchmark datasets have demonstrated the superior performance improvement on both convergence speed and predictive accuracy for multi-tasks classification of graph (drug). We also executed survival time prediction experiment over the two large WSI datasets of lung cancer patient, i.e. NLST and TCGA, from which DeepGraphSurv indicated significant accuracy lift brought by graph representation learning over patches.

During the study, we realized the importance of patch sampling in final prediction results. Low quality of patches largely affect eventual model accuracy. In other words, the model is not robust enough to challenging practical scenarios where data quality is relatively compromised by either sample contamination or scanning problems. Detecting and localizing pathological region of interest (ROI) over whole slide pathological image (WSI) may mitigate this problem. To reduce computational complexity, we introduced a two-stage superpixel-based ROI detection approach. To efficiently construct superpixels with fine details preserved, we utilized a novel superpixel clustering algorithm which cluster blocks of pixel in a hierarchical fashion. The major reduction of complexity is attributed to the combination of boundary update and coarse-to-fine refinement in superpixel clustering. The former maintains the accuracy of segmentation, meanwhile, avoids most of unnecessary revisit to the 'non-boundary' pixels. The latter reduces the complexity by faster localizing those boundary blocks. Detector of RoI was trained using handcrafted features extracted from super-pixels of labeled WSIs. Extensive experiments indicated that the introduced superpixel clustering algorithm showed lifted accuracy on lung cancer WSI detection at much less cost, compared to other classic superpixel clustering approaches.

Moreover, the clustered superpixels do not only facilitate a fast detection, also deliver a boundary-preserving segmentation of ROI in whole slide images.

As summary, in this thesis we introduced several novel methods in terms of improving the deep graph learning techniques and application of graph neural networks with adaptive kernels on large-scale graph structured data. We have extensively experimented the proposed methods on predictive tasks like drug property prediction, point-cloud classification and survival prediction on WSIs.

CHAPTER 2

Adaptive Graph Convolutional Neural Network

2.1 Introduction

Extending convolutional operator from regularly shaped grids to irregular graphs is not trivial. For the simplicity of constructing convolutional kernel, most early graph neural networks (GNNs) assumed that input data are still low-dimensional, analog to those data that CNNs deal with. Because, in their convolutional layer, the convolver handled a subset of nodes grouped by node degree respectively and independently [8, 9]. More importantly, their convolutional kernel is over-localized and infeasible to learn hierarchical representations from complex graphs with unpredictable and flexible node connectivity, e.g. chemical molecules and social networks [12]. In some applications, e.g. classification of point cloud [13], the topological structure of graph is more informative than the node features or the edge attributes alone. Unfortunately, the existing graph convolution networks [14] cannot thoroughly exploit the geometric property of graph due to the difficulties of implementing a parameterized spatial

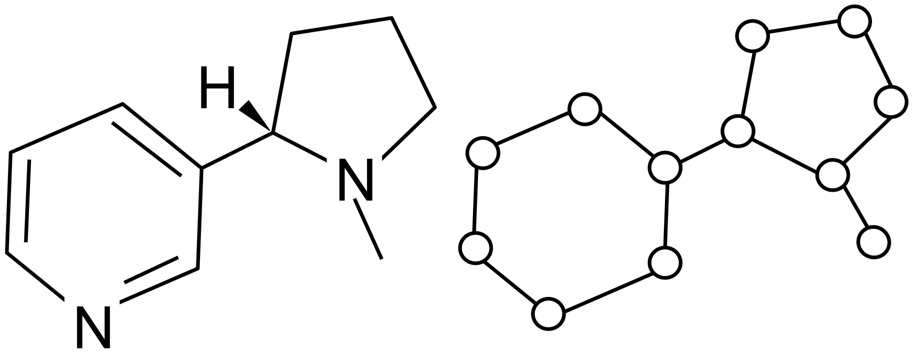


Figure 2.1: Example of Molecular Graph.

kernel compatible with different scenario of node neighborhood. Similar difficulty is also interfering the extension of graph convolutional networks to new applications, e.g. human activity recognition [15] and co-citation networks [16]. Besides, given the flexibility of topology structure of graph and the $\mathcal{O}(N^2)$ scale of parameters to define node-pair connectivity, learning a topology-preserving spatial convolutional kernel for every unique graph data sample is impractical for real-world applications.

Inherited from classical CNNs, a shared convolutional kernel among training samples is one of common assumptions. Consequently, to guarantee a unified dimensionality of input/output for all samples at consecutive layers, the input graph data have to be pruned, which is also a constraint of utilizing traditional networks on graph structured data directly. However, this kind of preprocessing on graph-structured data is going to harm the completeness of graph-oriented information, especially the features derived from its topology. For instance, the coarsening of molecule is hard to be justified chemically, and it is likely that the coarsened graph has lost the key sub-structures that differentiate the molecule from others. In Figure 2.1, it shows the molecular graph of chemical compound Nicotine (C10H14N2, SMILES: CN1CCC[C@H]1c2cccnc2) and its graph structure (omit hydrogen atoms). From the example in Figure 2.1, we can tell that the removal of any Carbon atom from the graph would break the Benzene ring. It would be much better if the graph CNNs could accept and preserve the original graph structures of data. Beyond the spatial graph convolution applied to quasi-grid graphs, spectral-based graph convolutional neural networks, derived from the graph Fourier transform [17], are promising to offer an more flexible convolutional kernel for sophisticated graph-structured data.

Lastly, in this chapter, the graph-structured data used in experiments either have an intrinsic graph structure (e.g. chemical molecules) or have one reconstructed through clustering of vertices (e.g. point-cloud). In existing graph networks [8, 9, 15], the initial graph structures are enforced to remain stable during the training time. While, on the other hand, it is doubtful that the graph topology derived in unsupervised fashion, happen to be optimal for each particular learning task. Although there were pioneering works who included supervised graph reconstructions with fully connected networks [9], the computational complexity from the large number of trainable weights constrains the initialization of graph only feasible to small graph. Furthermore, the graph topology that fits one pre-trained network may not work well with another graph neural network [18], which obviously constrain the generalization of learned graph neural network (GNN) layers.

In this chapter, we are going to introduce a novel spectral-based graph convolution neural network, compatible with graph data of diverse topological structures, e.g. the organic molecules that consist of a different number of atoms. To deal with the fixed graph that may be stale, we choose to grant the network the capability of learning the supplement to graph topological structure. Therefore, different from a parametric kernel formed by a fixed graph Laplacian matrices L [9, 12, 15] , we parametrize the graph Laplacian itself. While, given the goal of preserving topology on each individual graph, we cannot learn the L as trainable parameters directly, let alone the unacceptable computational cost. For instead, we parameterize the distance metric between pairs of node feature vectors as an indirect learning of the self-organized structure of each graph sample. A reasonable assumption is that the distance metric parameters are shared by all samples that belong to the same type, e.g. molecular graphs. Consequently, each individual sample is able to train the network on a unique and adaptive graph Laplacian that preserves its uniqueness and infers any undiscovered task-related substructures. A customized graph Laplacian L will lead to a customized kernel that combines features on neighbor nodes within a flexible coverage. It is interesting to question what exact graph that optimally empower a particular task. For instance, the chemical bonds, found via chemical experiments, naturally build a graph for any compound. However, it is never guaranteed that the convolver that works on intrinsic graph has extracted every meaningful feature. We introduced a so-called residual graph to unveil the substructures which intrinsic graph does not present. Furthermore, in order to guarantee residual graphs to be the optimal supplement to intrinsic graph for the particular task, we learn the residual graph topology along with the rest of network, under the supervision of task-specific annotations.

To implement the learning of adaptive residual graphs, we are faced two major problems:

1. how to efficiently construct residual graph during training;
2. how to preserve unique graph topology in the batch-wise training.

A direct learning of L is of exponential complexity and with $\mathcal{O}(N^2)$ parameters for a $R^{N\times d}$ graph sample. Allowing the topological structures preserved in M training samples means complexity of $\mathcal{O}(MN^2)$, which is unscalable for large graphs. While, an indirect learning of graph structure based on the Mahalanobis distance metric [19] and transformation in feature manifold is able to reduce the complexity scale to $\mathcal{O}(d^2)$, or even $\mathcal{O}(d)$, assuming the metric parameters are shared by all graphs. More importantly, after this, the learning complexity becomes independent of graph size N. When execute the convolutional operator, both intrinsic graph and learned residual graph will be used in the kernel: $\hat{L} = L + L'$. Due to that the proposed layer is another spectral-based graph convolution (SGC) layer but on a learned Laplacian, we name the new layer as spectral-based graph convolution on learned Laplacian, or *SGC-LL*.

In classical CNNs, back-propagation generally updates kernel weights to adjust the relationship between neighboring nodes at each feature dimension individually.

Then it sums up signals from all filters to construct hidden-layer activations. To grant the graph convolutional network a similar capability, we applied a re-parameterization on the output feature using a linear transform weight and bias. Finally, the $\mathcal{O}(d^2)$ training parameters in the proposed graph convolution layer consist of two segments: the Mahalanobis distance metric parameters and the feature transform weight and bias. To facilitate the diverse input graph of varying number of nodes, we need to pad zeros to both feature and adjacency tensor. Therefore, we also modify existing graph pooling and gather layer to recover the original data from padded tensors before layer execution. Because the introduced graph network is capable of being adapted with respect to graph topology, and more importantly the graph being used is also adaptive towards learning task, we name the network as Adaptive Graph Convolutional Network (AGCN) to highlight these valuable features. As first introduced in paper [1], AGCN model was introduced as a new spectral graph convolutional network capable of learning topology-preserving graph representation from graph data of diverse structure and size. Later, the effectiveness of AGCN received confirmation from its application on multiple graph-structured data and prediction tasks ranging from chemical molecules and 3D point cloud generated by LIDAR [14]. AGCN had demonstrated overwhelmingly better accuracy on both graph classification and graph attributes regression.

Some technical advantages of the AGCN architecture are summarized as below:

1. Construct and learn unique graph Laplacian for each individual training graph sample, preserving the completeness of original information, especially in terms of spatial topology.
2. Low computational expense in scale of $\mathcal{O}(d^2)$, independent of graph vertex number N, making the network lightweighted and attractive in prediction tasks on large-scale graphs, such as social graph.

3. Explainable model. Graph neural networks have an intrinsic advantage on interpretability. Discovering the hidden substructures on graph and the representations learned upon those substructure validated the effectiveness of learned residual graphs generated by AGCN.
4. Residual graph is compatible with both spatial and spectral graph convolutional network. We chose SGC [10] as baseline model and build our SGC-LL on top of it, while the idea of residual graph and the learning of graph Laplacian via deep metric learning is trivial to be extended onto other graph neural networks, e.g. GAT [20] and Differentiable Graph Module [21].

2.2 Related Work

Given the fact that graph is a more common and generic topology of data in real-world scenarios, there is an increasing interest of generalizing CNNs to graph data since the significant success on computer vision starting in 2011. The majority of advances in this subarea of GNNs could be well categorized as spatial-based graph convolutional networks (Spatial GCN) and spectral-based graph convolutional networks (Spectral GCN), according to the domain in which the convolution operator executes. Figure 2.2 showed a comparison of different convolutional kernels (filters) on graph. Before the proposal of adaptive graph convolutional network (AGCN), the major topic of developing convolutional operators on graph is to empower a more flexible coverage of neighborhood vertices, while maintaining reasonable parametric complexity.

2.2.1 Spatial-based Graph Convolutional Networks

The first trial of formulating an analogy of CNN on graph was accomplished by [8]. Particularly, the authors proposed a sparse kernel that aggregated the vertex features element-wisely from its neighbors. The finite-size kernel is nonparametric though, and the graph was self-constructed and constricted by data. [22] extended spatial kernel [8] to molecular graph and used dedicated weight matrices for the cluster of nodes of same degree. The drawback of [22] is that the node degree scenario should pre-defined and fixed, otherwise the network cannot be designed and initialized. And the spatial kernel [22, 8] is also over-localized since the adjacency matrix form an undirected graph is merely able to represent the 1-hop connections, therefore, the kernel cannot assign weights to those peripheral nodes not directly linked to the central node. To relieve the constrain, the diffusion-convolutional network (DCNN) [23] builds K transform matrices to handle at most K-hops diffusion of node features on graph, allowing output $x' \in R^{N\times K\times d}$. To tackle the over-localized kernel, DGCN [24] executed two parallel convolution networks on two views of graph data balancing local and global consistency towards a semi-supervised problem.

Previous networks [22, 23, 24] more or less handled the challenge of diverse node degree, while none of them formulated a CNN-alike convolution on graph, and their convolutional layers were without loss of generality able to be approximated as the assembly of a series of fully connected layers. PTCHY-SAN [24] ignored the graph scale and only selected a fixed number of nodes with a fixed receptive filed, following one of graph labeling procedures. The receptive field were picked from its direct neighborhood. And lastly the normalized neighborhood from the receptive field serve the final aggregation operation. [23] proposed LGCL, transforming graph data back to grid-like structure, selects a fixed number of neighbor nodes for each feature dimension, and then then applies a 1D-CNN on top. [25] further discussed the theoretical generalization of CNN from grid to manifold and finally graphs. GraphSAGE [26] argued that above transductive algorithms required the presence of nodes in training and cannot deal with graphs consist of unseen nodes. However, [24, 26] did not utilize

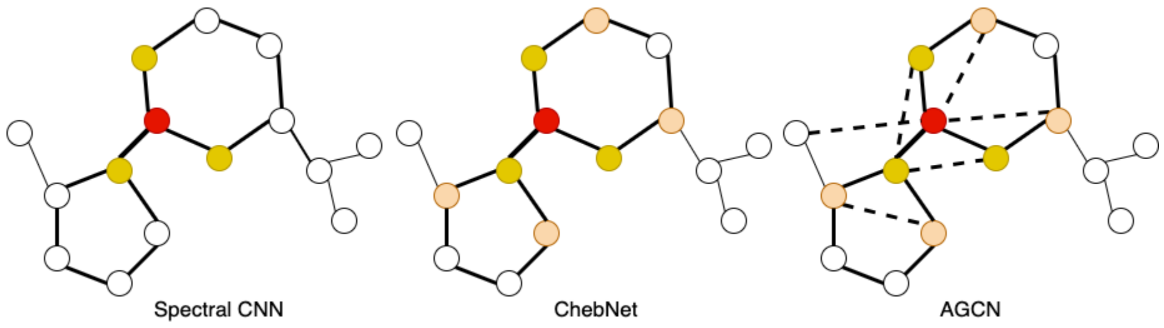


Figure 2.2: Comparison of Spectral Graph Convolutional Kernels.

the entire set of nodes, while, for some scenarios such as drug attribute prediction [27], a selective aggregation of nodes means a damage to local substructures, which have to stay intact in order to learn meaningful hidden representations.

2.2.2 Spectral-based Graph Convolutional Networks

Another category of graph convolutional operator is defined and executed in Fourier domain. [8] first proposed to compute the graph convolution based on the convolution theorem and the eigen-decomposition of normalized Laplacian matrix $L = U\Lambda U^T$. Then, the convolution is rewritten as:

$$g_\theta * x = U g_\theta(\Lambda) U^T x \tag{2.1}$$

, where U is the eigenvectors of the normalized graph Laplacian L: $L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$. D is the degree matrix and A is the adjacency matrix. While, this trivial solution comes with massive computational cost from the eigen-decomposition. More importantly, the spectrum filtering may result in non-locality on spatial domain after applying inverse graph Fourier transform U. [9] attempted to tackle the spatial non-locality after filtering by the nonparametric spectral kernel [8] by parametrizing $g_\theta(\Lambda)$

in terms of $diag(W)$, whose parameters are $W \in R^N$. Furthermore, another non-linear approximation of kernel $g_\theta(\Lambda)$ was proposed as:

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k. \tag{2.2}$$

The kernel Eq 2.2 mitigated the spatial non-locality by smoothing the spectrum filtering, while the computational cost is still high. [32] introduced a truncated Chebyshev expansion of kernel that comprises K items as $T_k(\Lambda)$. A recurring evaluation of $T_k(\Lambda)$, with initial states $T_0(x) = 1$ and $T_1(x) = x$, is formulated as : $T_k(x) = 2T_{k-1}(x) - T_{k-2}(x)$. When convolving with feature x, the $\mathcal{O}(N^2)$ complex multiplication with dense Fourier basis U is replaced with the multiplication with the sparse $\hat{L}$, and it also saves the eigen-decomposition of $\hat{L}$. Consequently, the overall complexity is reduced as $\mathcal{O}(K\|\epsilon\|) \ll \mathcal{O}(N^2)$. $\|\epsilon\|$ is the count of non-zeros in $\hat{L}$ and the number of graph edges. Both [32] and [33] relied on the approximation of spectral convolutional operator finalized as:

$$g_\theta(\Lambda) * x = \sum_{k=0}^{K-1} \theta_k T_k(\hat{L}) x \tag{2.3}$$

, $\hat{L}$ is re-normalized graph Laplacian defined as:

$$\hat{L} = \frac{2}{\lambda_{max}} L - I_N, \tag{2.4}$$

where λ_{max} is the maximum eigenvalue of L. While [11] further simplified the evaluation of Eq 2.3 by setting $K = 1$ and $\lambda_{max} \simeq 2$ to alleviate the overfitting to local structures. Because local neighborhood might deliver a biased representation of graphs especially when the node degree distribution is skewed. Authors argued that a stack of multiple linearly approximated convolutions, i.e. $K = 1$, is also able to recover a similar multi-hop knowledge aggregation as $K > 1$. Given $K = 1$ and

$\lambda_{max} = 2$, and the recurring evaluation equations, the Eq 2.3 was further simplified in [33] as:

$$g_\theta * x \simeq \theta_0 - \theta_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x, \tag{2.5}$$

with layer-wide shared parameters $\{\theta_0, \theta_1\}$. An additional assumption $\theta = \theta_0 = -\theta_1$ resulted in the linear approximation of single-layer spectral-based convolution as shown in Eq 2.3 as:

$$g_\theta * x \simeq \theta\big(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}\big) x. \tag{2.6}$$

However, as expected, a repeated application of Eq 2.6 leads to gradient exploding/vanishing. So, within such layer, a re-normalization of adjacency A is executed by $\hat{A} = A + I_N$ to control the eigenvalues of $\hat{L}$ fall into range $[0, 2]$. The baseline of spectral GCN in the chapter is ChevNet [10] with the K-rank Chebyshev polynomials as an approximation of spectral kernel after eigen-decomposition. The introduced AGCN is also founded on a similar formulation. The Figure 2.2 demonstrated the difference of convolutional filters on several popular GCNs. Note that the red spot is the central node of convolution, the orange nodes represent receptive field. The figure included three common spectral GCNs: (1) Spectral CNN [8]; (2) ChevNet [10]; (3) AGCN [1]. While, Spectral CNN [8] is only able to aggregate first-order neighbor vertices and ChevNet is able to include at most K-hop neighbors within the kernel. The convolutional filter of AGCN is based on both the original graph (solid line) and the learned residual graph (dash lines), that the network inferred from training data. This means that AGCN can learn aggregated features from a kernel that covers K-hop neighbor nodes, but also learn from those nodes that were not even neighbors to center node at original graph.

2.2.3 Graph Neural Network for Biomedical Problems

Biological and chemical research is one of the fields where GCNs have made significant progress in recent years. Given the nature of organic compounds as molecular graph, it is straightforward to formulate chemical compounds as graphs and to perform graph neural network (GNN) on top for representation learning and prediction tasks. Within those, the GCNs, with localized kernels and deep network architecture, have successfully driven big progress on problems, e.g. drug property prediction [28], drug discovery [29, 30], reaction prediction [31, ?]. The early success of GNN on molecular graph was the *NeuralFP* proposed in [22]. While, [22] had a strong assumption on node degree distribution, making the spatial neighborhood aggregation difficult to be generalized to graph of skewed degree distribution. And if designing a different kernel for different feature channel, the massive matrices multiplications are unscalable for large graphs. Consider the real-world scenarios, we have to balance the representational capability and the computational cost. While, [22] groups nodes across graphs in batch according to node degree and lets nodes of same degree share transform parameters, on the direction against over-locality overfitting. But it failed to perform a parametric elastic kernel to aggregate K-hop receptive field as what a CNN does.

Spectral GCNs balance the trade-off between the representational power and the computational cost by reducing the number of parameters and making the kernels shared by entire nodes. In [1], we introduced GCNs to molecular graph classification and drug property prediction tasks. Not doing any node truncation or edge pruning, we devised a novel spectral convolutional layer deployed on full graphs preserving original topology. Besides, we argued that the intrinsic graphs are not optimal for particular tasks. Beyond a kernel designed with fixed structure, the SGC-LL layer makes graph structure trainable along with the rest of network. As opposite to the node ‘Selection, Assembly, Normalization’ procedure by [26] that prunes the graph to fit a pre-defined kernel, we let the kernel be adaptive to different graphs.

2.3 Methodology

In the session, we introduce the spectral convolutional network built on adaptive residual graph, i.e. Adaptive Graph Convolutional Network (AGCN). We first elaborate the construction of residual graphs based on original input graph and node feature vectors. As follows, we introduce the reparameterization operation we applied onto node feature vectors after convolution, in order to reconstruct the channel-wise dependency and correlation that classic CNNs always have in their convolution layers. At last, we illustrated the architecture of AGCN for graph-wise prediction task.

2.3.1 Spectral Graph Convolution on Learned Laplacian

As elaborated in last session, the existing GCNs are tackling two major challenges:

1. how to utilize the complete set of graph nodes and edges without making the model difficult to train;
2. how to balance the localized kernel and the global structures that generalize the network for high-level prediction tasks.

In order to have a kernel that combines both local and non-label features on graph, we insist to borrow the kernel defined on spectrum domain as [10] together with the K-rank Chebyshev expansion as approximation. Through tuning K, the layer has a configurable receptive field as classical CNNs. In practice, we set $K = 2$ in most cases. And we found it was also worth tuning K for best performance.

The root cause why previous GCNs cannot train over entire graphs is that the diversity of graph structures and the skewed node degree distribution makes it so infeasible to directly parameterize node neighborhood in any trivial formulation. Given the $\mathcal{O}(2^{N^2})$ possible node connectivity scenarios on a graph of N nodes, is it technically impractical to search the optimal node connectivity that best serve the training of network. And different from [24] To make graph trainable, we grant data the capability of self-construction of graph structure by training parameterized distance metric of nodes, so that the graph Laplacian itself becomes trainable and adaptive along with data. Given the learned distance metric weights M, we construct the residual graph, as supplement to the original graph, using node features X for each graph sample. Since the new spectral graph convolutional layer was executed on learned graph Laplacian matrice, we name the new layer as Spectral-based Graph Convolution on Learned Laplacian (a.k.a. *SGC-LL* layer). And the convolutional operator of *SGC-LL* layer is written as below :

$$g_\theta(\Lambda) * X = \sum_{k=0}^{K-1} \theta_k T_k((1-\alpha)\zeta(X,\Gamma) + \alpha L)X, \tag{2.7}$$

, where the function $\zeta(X,\Gamma)$ is to generate the new learned graph, in terms of graph Laplacian L'. The model is going to learn L' from the input node features X with the training parameter Γ. And Γ denotes any parameters to form the new graph based on input node features X. α, with range $[0,1]$, is the trade-off coefficient that balance the influence of original graph Laplacian L and the new graph Laplacian from $\zeta(X,\Gamma)$. When $\alpha = 1$, the graph Laplacian and entire convolutional function is regressed to ChevNet [21] as shown in Eq 2.3.

Because the convolution filter with K-hop neighbor coverage is still localized, at least not covering entire graph, and it has to aggregate neighbors attributes following the path defined by edge, either from original graph or the residual graph the network inferred. Therefore, to obtain the high-level representation of graph, we still need to

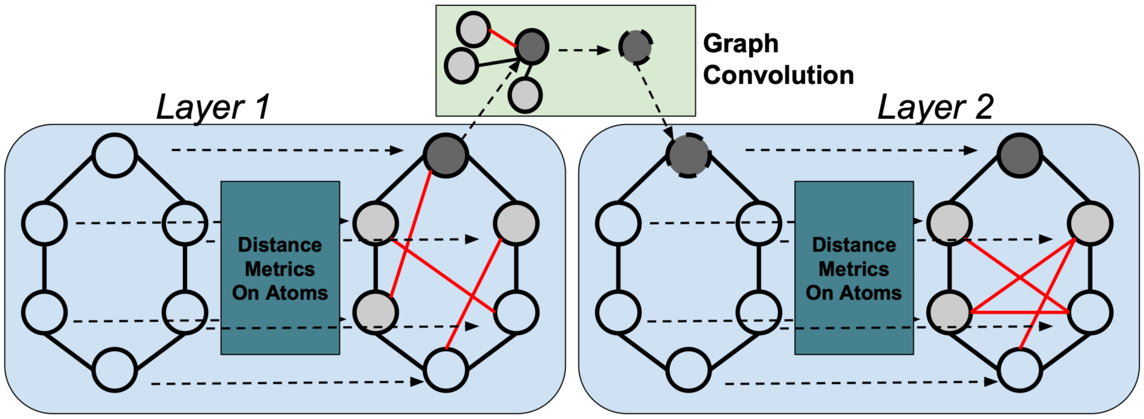


Figure 2.3: Illustration of execution of two consecutive SGC-LL layers.

stack consecutive convolutional layers to enable messages pass along connected edges during the training.

From Figure 2.3, we observed that at the output of first layer, the features would be passed to next layer, but the residual graph stay. And at following layer, the residual graph L', denoted as red lines, would be constructed again with the input features (i.e. the output of last layer). Because we argue that each layer is to only produce representation at certain level of granularity, the residual graph is supposed to discover the hidden structures that only affect learning of graph representations at the same level. They may not work at other layers or may even mislead network fitting. And more importantly, we try to avoid propagating some mistaken edges given by residual graph at earlier layers to later layer, by giving network a chance to fix unsuitable residual graph. Actually, I attempt to use similar architecture proposed in ResNet [32] to bridge previous residual graph to those in later *SGC-LL* layers, but it turns out not helping anything but worsen the performance.

2.3.2 Learning Graph via Metric Learning

As discussed in section 2.3.1, the early baseline Spectral CNN was trained over fixed graph and parameters are graph-related. Therefore, for each individual graph structure, we need a set of parameters trained for it. This results in unacceptable training complexity for graph data, e.g. organic compounds, whose molecular graph are of diverse substructures and no graph sampling or coarsening is suitable. To modeling the graph structure with a reasonably small group of parameters is our objective. Therefore, we propose the use distance metric as trainable parameters and use metrics and node-wise feature vectors to reconstruct graph when necessary during training. This is to avoid building parameter tensor of graph size. In this setup, the learning complexity is only linear to node feature dimensionality. Of couse, it definitely increases computational complexity, since a graph construction step is inserted before executing graph convolution operator. Inspired by how we construct initial graphs for data without intrinsic graph given, we use node-wise distance as indicator function of appearance of edge, to reconstruct graph based on node features and distance metric.

For graph structured data, the Euclidean distance is not necessarily the optimal metric to measure node-wise similarity [26]. In recent articles of metrics learning, the algorithms were divided into supervised and unsupervised learning. The optimal metric obtained in unsupervised fashion is to minimize the intra-cluster distance and also simultaneously maximize the inter-cluster distances. For labeled datasets, the optimal metric is the one that minimizes the learning loss [33]. The generalized Mahalanobis distance between two nodes (x_i, x_j) is formulated as:

$$D_{i,j}\sqrt{(x_i - x_j)^T M(x_i - x_j)}. \tag{2.8}$$

where M is the symmetric positive semi-definite matrix, therefore, it could be decomposed as $M = W_d W_d^T$. And the dense transform matrices $W_d \in R^{d\times d^*}$, converting input features X to the manifold in which the Euclidean distance still serve. And trainable matrices W_d are the *only* parameters to learn at SGC-LL layers. if $M = I$, Eq 2.8 reduces to the Euclidean distance (i.e. L-2) distance. Then, we normalize the distances given in Eq 2.8 via Gaussian kernel:

$$G^D_{i,j} = (\frac{1}{2\pi var(D)})exp(-\frac{D^2_{i,j}}{2var(D)}). \tag{2.9}$$

To make computations efficient, a sparse graph is required. Therefore, thresholding on G^D is to only keep the significant connections in constructed residual graph with adjacency matrix: $A_r = thred(G^D(W_d))$. Therefore, $A_r(W^d)$ is a differentiable function of distance metrics parameter W^d. In training process, the gradients back-propagated from training loss update the distance metrics W^d, and then, in next forward-pass, the residual graphs in batch will be reconstructed using the updated W^d. By doing metric learning before building residual graph, we are supposed to obtain the optimized metric parameters $\hat{W}_d$, applying which onto the node features is able to form the substructures that better serve the fitting of graph node embedding. Finally, the objective of searching optimal graph structure is transfered to a new problem of searching the optimal distance metric where $l2$ similarity of node pairs are measured.

2.3.3 Residual Graph and Original Graph

In real applications, some graph data come with their intrinsic graph topological structures, such as organic molecules. Molecule, when modeled as molecular graph, has its atoms as graph nodes and the chemical bonds as graph edges. If chemical formula indicates an edge connecting two atoms $\{i, j\}$, then on adjacency matric of corresponding graph there would be value 1 at coordinates $\{i, j\}$ indicating the edge. For molecular graphs, all the chemical bonds are justified by chemical experiments or observations, therefore there is no doubt on the value of those structure on learnning graph representation from molecular graph. However, for some other data, on which GCNs is about to perform, there is no intrinsic graph structure derived, e.g. 3D point-cloud data. Under these circumstances, we will need to construct initial graph before feeding the data to any GCNs. Besides above extreme cases, it is mostly likely that the original graphs, either derived from domain knowledge or obtained via graph initialization, may fail to effectively unveil the hidden substructures among the remote nodes on original graphs. The finite, fixed receptive field and the potentially high computational cost prevent one GCN layer from aggregating nodes attributes from entire graph or a flexible neighborhood.

Use chemical compounds and property prediction as example, the initial graph given by SMILES sequence of compound does not disclose anything on toxicity directly. The effective representations of toxicity on the compound are supposed to be composed of those atoms, while not necessarily supported by the bonds from the original graph. If learning on original graphs, it may be difficult to learn the optimal representations. Therefore, we introduce a so-called residual graph, defined as a supplemental graph, patching the substructure missing on original graph that may assists the learning. The residual graphs have the identical node number as original graph. To get rid of the curse of varying graph size, in SGC-LL layer, the distance metrics are the only parameters to learn.

2.3.4 Re-parameterization on Feature Output

At a classical convolution layer, given the input feature as $H^i \times W^i \times d^i$ and the dimensionality of output feature as $H^{i+1}W^{i+1}d^{i+1}$, each dimension of output features is the sum of all feature maps of last layer i, each of which is convolved by a kernel independently. Therefore, each layer has $d^i \times d^{i+1}$ kernels to learn. It means that the resulted features are not only built upon the neighbor vertices, but also depend on the rest of intra-vertex features from input. However, on graph convolution, it is not theoretically explainable to construct and learn a separate topological structure (graph) for each feature dimension. In order to construct the mapping of both intra- and inter-vertex features, at SGC-LL layer, we introduce a linear feature transform matrix and bias vector applied on the output features. The re-parameterization on output feature is formulated as:

$$y = g_\theta(\Lambda) * x \simeq \left(W(\sum_{k=0}^{K-1} \theta_k T_k(\zeta(X, \Gamma) + \alpha L)X) + b \right) \tag{2.10}$$

, in which the linear transformation matrix $W \in R^{d^i \times d^{i+1}}$ and the bias vector $b \in R^{d^{i+1}}$ are trained together with the distance metrics W_d. In total, at each SGC-LL layer, we have parameters up to the scale of $\mathcal{O}(d)$, where $d = \max d^i, d^{i+1}, d^i_m$ and d^i_m is the feature dimension of manifold in which Euclidean distance measured for each node pair. As we note that those feature dimensionalities are all independent of graph node count and node degree. At $(i+1)th$ SGC-LL layer, the spectral filters will be built in another feature manifold with different metrics.

Algorithm 6, represented as Eq 2.10, elaborated the major composition of SGC-LL layer. Using iteration is based on ease of narrative. Besides the Eq 2.8 and Eq

```
input  : Graph Data: input node feature vectors $X = X_i$, graph
         Laplacians $L = L_i$; Parameters: $W_d, W, b, \alpha$
output: Output node features $Y = y_i$
1 for $\{x_i, L_i\} \in \{X, L\}$ do
2     $A_r^i \leftarrow$ Eq 2.8, Eq 2.9;
3     $L_r^i = I_N - D_r^{-\frac{1}{2}} A_r^i D_r^{-\frac{1}{2}}$;
4     $L' = (1-\alpha)L + \alpha L_r^i$;
5     $y^i \leftarrow$ Eq 2.3;
6 end
```

2.9) were not explicitly expressed in batch-mode, while it is trivial to make data processed in terms of batch of graph. When comes to the implementation, the for-loop in Algorithm 6, with no loss of generality, could be replaced using the batch-wise tensor multiplication operators given by mainstream deep learning framework API, e.g. PyTorch and Tensorflow.

2.4 AGCN Architecture

The new graph convolutional network introduced in [1] is named as Adaptive Graph Convolution Network (AGCN), because SGC-LL layers are designed to efficiently learn hidden topological structure, i.e residual graphs, which is adaptive to both data and context of learning task. Besides SGC-LL layer, AGCN also comprises graph pooling layer, to squeeze receptive field, and graph gather layer [34], to aggregating all nodes into single vector as graph representation. In this section, we introduce other layers and motivation of utilizing those graph network layers in the proposed graph-wise classification network (Figure 2.4).

2.4.1 GCN Layers

Besides graph convolutional layers, to build a network that learn graph representation effectively, other sort of layers are also needed for graph data. For classic CNNs, there are pooling layers, fully connected layer, etc. [22, 34, 26, 35] proposed many GNN layers, e.g. graph pooling, graph coarsening, graph gather layers, and proved their effectiveness on learning robust graph representations.

Graph Pooling. The graph pooling operation is conducted feature-wisely. For node feature vector x_i, at the i-th vertex of graph, the pooling operator replaces the j-th feature value, $x_i(j)$, with the maximum at the j-th feature among i-th vertex and its neighbors vertices, $x_i(j) = max_{n \in N_i} x_n(j)$, if the layer is a max-pooling layer. If the layer is for avg-pooling, $x_i(j) = mean_{n \in N_i} x_n(j)$. In AGCN, due to graph structure is adaptive and being updated along training progress, the neighborhood of i-th vertex is as well changing w.r.t the update of graph adjacency A_r.

Graph Gather The graph gather layer sums up all the vertex features along the feature dimension as the final representation of graph data. The output tensor at gather layer for a batch of B graphs is of shape $(B \times d)$, where d is the feature cardinality of vertex representations. It will be used as input for a graph-level classification or regression. Without a graph gather layer, the AGCN is also able to be trained and used for vertex-wise inference tasks. Training is executed with given labels on vertices or in a weakly-supervised fashion by replying on graph-level label alone. The vertex-wise predictions include graph completion and many predictive tasks on social networks.

Bilateral Filter The purpose of using a bilateral filter layer [35] in AGCN is to proactively prevent over-fitting, consider the data scales of graph data are not comparable to other machine learning problems, e.g. ImageNet. Residual graphs definitely push the model to a better fitting to training tasks, while, at the risk of

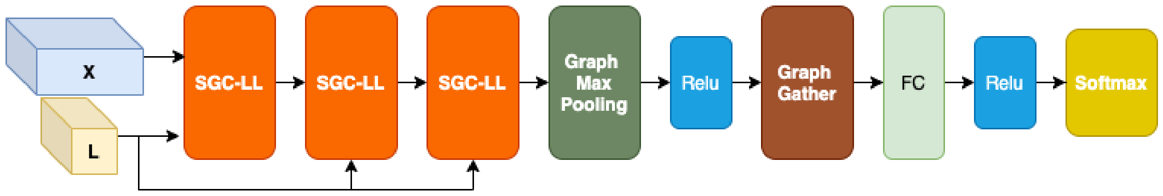


Figure 2.4: AGCN example: graph-wise classification.

over-fitting. To mitigate overfitting, we introduced a revised bilateral filtering layer to regularize the activation from *SGC-LL* layer by augmenting the spatial locality of updated graph Laplacian L'. We also introduced batch normalization layers to avoid gradient explosion or vanishing.

2.4.2 Network Configuration

The AGCN consists of multiple consecutive layer combos, the core layer of which is the SGC-LL layer. See Figure 2.4 as an example of AGCN for graph-wise classification. Besides graph-wise classification, other classification tasks include edge (connection) prediction and graph-node classification. The former is to give a predicted label on each edge of graph, and a good application for such network is prediction of protein interface based on GCNs [36]. The latter is to execute classification for each node on graph, and this is very useful for graph clustering or segmentation [37, 38]. Since a residual graph Laplacian is learned at SGC-LL layer. At the graph pooling layer that follows, the updated graph Laplacian $\hat{L}_i$, of sample i, will replace L_i when finding neighborhood N_i until next SGC-LL layer. As last convolutional layer transformed features, at next SGC-LL layer, the residual graphs have to be reconstructed from the scratch. While, the learned $\hat{L}_i$ will become the 'original' graph Laplacian L at following layers.

Padding Because for data like organic compounds, local sub-structures are decisive on chemical properties, e.g. toxicity. For instance, aromatic hydrocarbon is usually strongly toxic. However, if the hydrogen (H) atom was replaced by methyl group (-CH3), the toxicity of compound would be significantly reduced. Therefore, graph coarsening or feature dropping/averaging will damage the completeness of informative local substructures, resulting in wrong predictions. Therefore, when preparing data, we pad X and graph Laplacian L tensors to as large as that of the maximum node per graph in the dataset. Then, when used at layers, we remove the zeros padded to X, L and execute the calculations, e.g. Algorithm 1, on graphs of original size. By doing this, we maintain a unified batch shape required by deep learning frameworks, e.g. PyTorch, without pruning any decisive local structures on graphs. While, data padding and recovery lead to extra computational cost, which is linear to graph size N.

To make predictions toward particular tasks, a classifier or a regressor need to be added on top of output graph embeddings. We can either simply do logistic regression or insert a fully connected layer before classifier. In experiments, we add one linear layer between last graph layer and the softmax. Besides, adding a linear layer make AGCN adaptable to a multi-task scenario, making the AGCN to deliver multiple predictions for different tasks in one-shot. We will present several experiments about using AGCN in multi-task learning.

2.5 Experiments

2.5.1 Baselines

In experiment session, we compared the introduced AGCN with the state-of-the-art GNNs. [8] that constructs a spectral graph kernel by linear B-spline interpolation, is referred as Spectral CNN. Neural fingerprint [22], known as NeuralFP (neural fingerprint), is the spatial-based graph convolutional neural network particularly designed for molecular graphs. It uses kernel constructed in spatial domain for each node cluster grouped by the same node degree. We refer to the spectral graph convolution equipped with a K-localized spectral filter as GCN [10], in which a Chebyshev approximation is applied for a fast evaluation of consecutive tensor multiplications. Therefore the network proposeed in [10] is also known as ChevNet. Using *ChevNet* as one of baseline is to demonstrate the additional knowledge learned by the residual graph. Graph attentional network (GAT) [20] is also feasible for learning embeddings on molecular graphs, due to its definition that has no prerequisites on graph topology. As opposite, to enable the use of [8, 10] for molecular graphs of different scales and structures, a graph pruning is inevitable, sacrificing performance. Besides, graph isomorphism network (GIN) [39] is introduced as a GNN that learns node embedding via MLP and the aggregated node and edge feature vectors. The graph-level representation of GIN is given by averaging the node embeddings. Those baseline models included for the experiment of molecular graph classification are more or less feasible with diverse graph inputs of varying node count and topological structure. Because the methods, e.g. NeuralFP, GIN and GAT execute the feature aggregation within the first-order neighbors, and parameterized transforms were built and initialized either for entire nodes or for node clusters [22]. Their over-localized, non-parametric kernels excluded most topological structure of molecular graphs from modeling. And that is the reason that AGCN outperformed the baselines on the drug classification benchmarks.

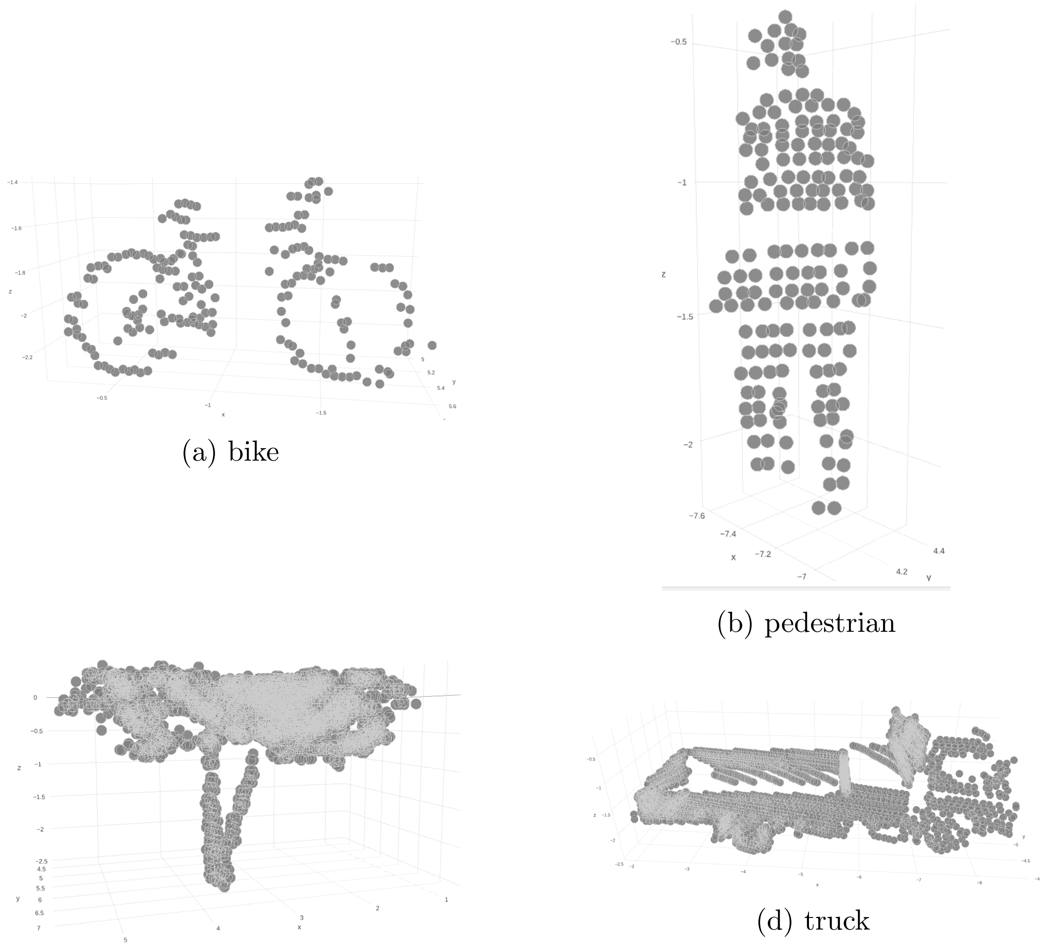
2.5.2 Dataset

2.5.2.1 Molecular Graph

The datasets used in this experiment are all about drug property prediction. Given labels obtained from the extensive lab or clinical experiment over a long list of organic compounds, it is possible to learn some patterns from the compounds related to certain predefined biochemical properties, such as toxicity and solubility. With the open-sourced cheminformatics software RDKit [40], it is straightforward to convert any compound to its corresponding graph representation, molecular graph, which consists of node list, edge list, node features and edge features. The node feature extracted from molecular atoms: atomic number, atom degree, formal charge, chiral tag, number of Hs, hybridization, the Boolean indicator on aromatic and the scaled atom mass. And the edge features include a 4-digit one-hot vector to represent bond type, the indicator on bond aromatic, as well as two Boolean features that determine if bond is conjugated or in-ring.

Downstream task datasets we utilized in experiment are 4 multi-task, binary drug classification dataset from MoleculeNet [41]. They are:

1. Tox21. Toxicity clinical data with labels on 12 tasks corresponding to different biological syndrome. Each label represents an observation toward one property of the compound.
2. oxcast. Another toxicology measurement of drugs collected from the same initiative as Tox21, providing toxicology experimental results for a large library of compounds based on in vitro high-throughput screening. It offers 617 experiments on over 8K compounds.
3. SIDER. A dataset that contains marketed drugs and adverse drug reaction (ADR) with 27 group of organ classes [42].



(a) bike

(b) pedestrian

(c) tree

(d) truck

Figure 2.5: Example of PointCloud data from Sydney urban dataset.

4. ClinTox. Collected data on drugs approved by the FDA and the drugs that failed clinical trials for toxicity reasons. The dataset has two binary labels: 1) clinical trial toxicity pass or fail; 2) FDA approval or not [43].

2.5.2.2 Point Cloud

Point cloud is another important 3D geometric data format widely used in computer graphcis and autopilot system with LiDAR. Since point cloud or mesh are usually not deployed in regular format, previous works simply transform them back to regularly shaped voxel grids [44]. Some of them even sample images as views to represent the 3D mesh. While, the data transformation and sampling definitely introduce extra noise to data itself and render voluminous data, which both harm downtream training tasks. In this experiment, we demonstrated how AGCN works with point cloud as input. And the comparison showed that our method is able to execeed other GCNs on classifying objects represented as point cloud.

The Sydney urban point cloud dataset contains street objects scanned with a Velodyne HDL-64E LIDAR, collected in the CBD of Sydney, Australia. There are 631 individual scans of objects across 26 classes. Due to the actual size and shape of object highly differ, the numbers of received point for different objects also vary (see Figure. 2.5 for illustration). Before feed point sets to previous CNNs, we need to unify the size by downsampling. Coarsened samples must lose part of structural information. While, the AGCN overcomes such drawback by accepting raw point sets of different size. Previous graph convolution share an identical kernel, but, the shared one may mix up features on points regardless of the actual distance. While, the AGCN is able to do convolution exactly according to the spatial relations. The initial graphs of point cloud were constructed by agglomerative clustering. The cutting-edge method on point set recognition, PointNet [44], cannot handle varying sized point cloud data.

2.5.3 Experimental Result

The baseline models and AGCN were trained with the identical training dataset and tested over the same dataset. To measure the classification accuracy, RoC-AUC was chosen as the metric for comparison. Because for the datasets such as Tox21, there are 12 individual classification tasks, the numbers presented are averaged RoC-AUC of each task. To remove randomness, we also applied 4-fold cross-validation and averaged the numbers. From numerical results listed in Table 2.1, it is obvious

	Tox21	Toxcast	ClinTox	SIDER
No. graph	7831	8575	1478	1427
No. task	12	617	2	27
NeuralFP [22]	0.7341	0.6384	0.7469	0.5525
GraphSage [27]	0.7470	0.6335	0.5924	0.6040
ChevNet [10]	0.7481	0.6739	0.7573	0.5914
GAT [20]	0.7540	0.6460	0.5886	0.6090
GIN [39]	0.7480	0.6340	0.5804	0.5730
AGCN [1]	0.8016	0.7033	0.7688	0.5921

Table 2.1: Class-average ROC-AUC on Four Molecular Graph Datasets.

that the AGCN outperformed other four baselines on 3 of 4 datasets. While, at SIDER dataset, GAT gave the best classification accuracy. Considering that GAT is constructed using complete original graph, therefore, there is chance that GAT outperformed AGCN on some tasks. Given that AGCN has more parameters to train, it is likely to have under-fitting issue on a small dataset like SIDER [42]. Recently, researchers also found that GNNs could benefit from a self-supervised pre-training before fine-tuning towards classification tasks [45]. Table 1 also includes RoC-AUC of GAT and GIN tested over the aforementioned 4 molecular graph datasets reported in [45]. It showed that, even with well-designed pretrain task, their performances were still worse than AGCN, who did not experience pretrains, on 3 of 4 tasks. And because of more parameters introduced, AGCN had more significant advantage on relatively larger datasets which the model was able to fit better.

As to the multi-task classification results from Table. 2.1, we notice that the AGCN significantly boosted the accuracy on both small and large datasets, compared other GCNs [22, 10]. For the massive 617 tasks from Toxcast data, the performance still got lifted by around 3% on average. Molecular graph, directly given by chemical formula, is the intrinsic graph for compound data. They come with high variety in both topological structure and graph size. The spectral kernel in *Spectral CNN* [8] can only connect 1-hop neighbor (nearby vertex directly connected by edge), so it is over-localized. This becomes an issue when dealing with molecules, because some important sub-structures of molecular graph are impossible to be covered by over-localized kernels. For example, centered at any carbon atom of Benzene ring (C6H6), the kernel at least needs to cover the vertices of distance $d_{\mathcal{G}}$ <=3, if you want to learn representation from the ring as a whole. The K-localized kernel in GCN [10] is no longer too local, but the kernel is still assumed to be shared among data. It is fine if the molecules in training set share many common sub-structures such as OH (carbonyl group) and C6H6 (Benzene). However, if the molecules are from different classes of compound, GCN may not work well especially when data from some type are short. This is probably why the GCN has similar performance as AGCN on large datasets such as the Sider, but it dramatically worsened on small datasets, e.g Delaney and Clintox.

The AGCN is able to handle molecular data in a better way. The adaptive graph allows input samples to have unique graph Laplacian, so each compound indeed has its unique convolution filter customized according to its unique topological structure. Because of this capability, we can feed the network on the original data (atom/edge features and molecular graph) without any loss of information. Furthermore, our SGC-LL layers train the distance metric minimizing predictive losses of specific tasks together with other transform parameters. Therefore, when it converged, at each SGC-LL, we would find the optimal feature space and distance metric to construct the graph that best serve the task, e.g. toxicity and solubility prediction. This learned graph may contain new edges that did not exist in original molecular graph.

PointNet [44] has overwhelmingly better performance in general on pointcloud classification, while it has a much deeper network architecture of longer training time. In this experiment, the original node feature are coordinates $\{x, y, z\}$ of each point

	All Classes	Building	Traffic Light
Spectral CNN [8]	0.6523	0.6754	0.5197
NeuralFP [22]	0.6350	0.8037	0.5344
ChevNet [10]	0.6657	0.8427	0.7417
AGCN [1]	0.6942	0.9906	0.8556

Table 2.2: Average ROC-AUC on Sydney Urban PointCloud Datasets.

at LiDAR receiver. For baselines, we compared AGCN with other GCNs [8, 22, 10] which are widely used for graph classification tasks. In general, spectral-based GCN outperformed spatial GCN on pointcloud data. This was mainly because the graph was initialized via k-mean clustering is not good enough to represent the intrinsic substructures from pointcloud. Due to the viewpoint of LiDAR with respect to detecting object, some topological close points were somehow separated remotely in original pointcloud data. [22] is only able to aggregate first-order neighbors, therefore, its performance get worsened by the poor graph initialization. Spectral-based methods is able to draw kernel with entire graph included, therefore it is more robust when local substructures get messed up by graph initialization, however, they are unable to correct those mistaken edges, because their graphs are fixed during training. Only AGCN [1] is able to work on learned residual graph which may fix the wrong initial graph. From Table 2.2, AGCN has 3% lift on average ROC-AUC over 26 classes. Especially on class of *building* and *Traffic light*, it showed 15% and 11% gain, compared to the best of other GCN competitors.

2.5.4 Conclusion

We proposed a novel spectral graph convolver (SGC-LL) that work with adaptive graphs. SGC-LL learns the residual graph Laplacian via learning the optimal metric and feature transform. As far as we know, the AGCN is the first graph CNN that accepts data of arbitrary graph structure and size. The supervised training of residual Laplacian drives the model to better fit the prediction task. The extensive multi-task learning experiments on various graph-structured data indicated that the AGCN outperformed the state-of-the-art graph CNN models on various prediction tasks.

CHAPTER 3

Attentional Adaptive Graph Convolutional Network

3.1 Introduction

After transformer and multi-head attention scheme proposed in [46], researcher have achieved some successful trials of introducing attention to graph neural networks (GNNs). GAT [20] is one of them that built graph convolutional kernel based on attention value pairs. GAT is treated as a variant of spatial-based convolutional layer, while it only preseve the first-order neighborhood information on graph. More importantly, the attention head is learned for each neighbor pairs with direction from each neighborhood node to center node. And the attention values is generated after passing a single Fully Connected (FC) Layer with transform weight a and $LeakyReLU$ as activation function. After stacking several GAT layers, the network is capable of learning high-level graph repesentation. However, this attention scheme is still constructed to model the so-called *node-to-node* relationship within on over-localized receptive field. It cannot easily elaborate the *node-to-graph* attentions, which is crucial when aggregating node features to graph level at places like *Graph Gather* layer before softmax. At graph gather layer of AGCN [1], the gathering operation simply sum up the features from entire graph without considering the ordering or the weight allocated on each node. In the chapter, we attempted to propose a *node-to-graph* attention scheme that calculate the task-specific attention value of each node toward the graph. And we named the new model as Attentional AGCN, because it reused the backbone network from AGCN and the attention net, that generate node-wise attetion values, consists of SGC-LL layers [1]. To verify the effectiveness of Attentional

AGCN, we applied it to sophisticated image understanding problem, i.e. whole slides image (WSI) based survival analysis that require the model to be able to extract patch-level local features and also to efficiently rollup them to image level with topological information preserved. From what we know, this is the first trial to predict survival risk based on raw WSIs, and in the experiment section of the chapter we demonstrated the additional performance gain delivered by topological structures of patches alone.

Survival analysis is generally a set of statistical models where the output is the elapsed time until a certain event occurs. The event can range from vehicle part failure to adverse drug reaction. Clinical trials are aimed to assess different treatment regimes with the biological death as primary event of interest to observe. An accurate estimate of survival probability provides invaluable information for clinical interventions. The Cox proportional hazards model [47] is most popular in survival analysis. However, the classical Cox model and its early followers overly simplified the patient's survival probability as linear mapping from covariates. Recently, Katzman designed a fully connected network, DeepSurv [48], to learn the nonlinear survival functions. Although it was showed that neural networks outperformed the linear Cox model [49], it cannot directly learn from pathological images. Along with the success of convolutional neural networks (CNNs) on generic images, pathological image, as well as CT and MRI [50], have become ideal data sources for training DL-based survival models. Among them, whole slides image (WSI) [51] is one of the most valuable data formats due to the massive multi-level pathological information on nidus and its surrounding tissues.

WSISA [52] was the first attempt to predict survival risk of cancer patients purely based on whole slide pathological images (WSIs). To have a efficient approach on WSIs, a patch sampling preprocessing on WSIs is inevitable, and without any validated prior knowledge, the patch sampling is random. However, their DeepConvSurv model was trained on clustered patch samples separately. Consequently, the features extracted were over-localized for WSIs because the receptive field is constrained within physical area corresponding to a single patch (0.063 mm^2). The pathological sections of nidus from patients contain more than the regions of interest (e.g tumor cells), therefore, the representations from random patch may not strongly correspond to the disease. Furthermore, it has been widely recognized that the topological properties of instances on pathological images are crucial in medical tasks, e.g. cell subtype classification and cancer classification. While, WSISA is neither able to learn global topological representations of WSIs nor to construct feature maps upon given topological structures.

Graph is widely employed to represent topological structures. However, modeling a WSI as graph is not straightforward. Cell-graph [53] is infeasible for WSIs due to its huge number of cells and the many possible noisy nodes (isolated cells). The intermediate patch-wise features are a good option to construct graph with, balancing efficiency and granularity. However, applying CNNs on graph-structured data is still difficult.

Overall, in the chapter we introduce an end-to-end graph convolutional neural network (GCN) based survival analysis model (DeepGraphSurv) where global topological features of WSI and local patch feature are naturally integrated via spectral graph convolution operators. The contributions are summarized as:

1. learn both local and global representations of WSIs simultaneously: local patch features are integrated with global topological structures through convolution;
2. task-driven adaptive graphs induce better representations of WSI;
3. introducing graph attention mechanism reduces randomness of patch sampling and therefore increases model robustness.

As far as we learn, DeepGraphSurv is the first end-to-end GCN based survival prediction pipeline. It only utilized WSIs as input data and output survival risk directly. Extensive experiments on large-scale cancer patient WSI datasets demonstrated that our model outperformed the state-of-the-art models, including other GNNs, by providing more accurate survival risk predictions.

3.2 Related Work

3.2.1 Attention on Graph

Inspired by the success of attention-only sequence-mapping networks, e.g. Transformer [46], and a variety of attention mechanism on natural language understanding tasks [54, 55], attention scoring scheme was introduced to GNNs by Graph Attention Network (GAT) [20], leveraging a masked self-attentional layer that allows nodes specifying weights to different neighbors on graph. The mask applied to node selection is where the graph structure introduced, GAT computes attention coefficients $\alpha_{i,j}$, where an edge appears i.e. $A_{i,j} > 0$. And the neighborhood N_i are set of node i's directly linked neighbor nodes. Similar to [11], GAT claimed that a stacking of first-order aggregation renders similar effects.

$$\alpha_{i,j} = softmax_j(e_{i,j}) = \frac{exp(e_{i,j})}{\sum_{j \in N_i} exp(e_{i,k})}. \tag{3.1}$$

Eq 3.1 shows the formulation of a normalized attention coefficient from node i on one of its neighbor j. $e_{i,j}$ is the raw attention coefficient before normalization. To learn the node-wise representations that best computes point-wise attentions $e_{i,j}$, GAT parameterize the process by first applying a linear transformation W onto node feature vector x_i and x_j, and then multiplying the concatenated joint feature vector with attention weight transform a^T. And finally an activation function was applied on each attentional head before normalization. the complete formulation of single attentional layer from GAT [20]:

$$\alpha_{i,j} = \frac{exp(LeakyReLU(a^T[Wx_i||Wx_j]))}{\sum_{j \in N_i} exp(LeakyReLU(a^T[Wx_i||Wx_j]))} \tag{3.2}$$

After obtaining the node-wise normalized attention values, the convolutional operator is nothing but a weighted summation of neighbor features:

$$\hat{x}'_i = \sigma\big(\sum_{j \in (N_i) \cap i} \alpha_{i,j} W x_j\big) \tag{3.3}$$

, where $\hat{x}'_i$ is the output of layer and the summation includes center node i itself. And note that that weight is shared with the attention value calculation Eq 3.2, making attention and aggregation happen in the same manifold.

3.2.2 Deep Learning for Survival Prediction

Survival analysis [56] is a set of statistical inference models where the output is the elapsed time until a pre-defined event occurs. The event can be anything of interest, ranging from vehicle part failures to adverse drug reactions. Clinical trials are aimed to assess different treatment regimens with biological death as the primary event of interest to observe. An accurate estimate of survival probability provides invaluable information for clinical interventions. The Cox proportional hazards model [47] is the most popular model in survival analysis. While, the classical Cox model and its early followers overly simplified the patient's survival probability as linear mapping from covariates. Recently, Katzman et-al, designed a fully connected network (FCN), named as DeepSurv [48], to learn a nonlinear mapping of covariates to the representations in survival prediction. Although it showed that the neural networks [48, 49] outperformed the linear Cox survival model, their networks cannot directly work on pathological images. Along with the success of convolutional neural networks (CNNs) on generic images, pathological images, including CT [57] and MRI [50], have become ideal data sources for training DL-based survival models. Among them, whole slide images (WSIs) are one of the most valuable data formats due to their massive multi-level pathological information on nidus and its surrounding tissues [58, 53].

WSISA [52] was the first success of introducing whole slide pathological images (WSIs) as major data source to survival prediction. Because the data size of single whole slide pathological image is usually at gigabyte level, to have a cost-efficient algorithm, most of existing methods on WSIs, including [58, 52], are based on a set of patches with reasonable size, like 128×128, as inputs. Therefore, a patch sampling on WSIs is required before running the algorithm. However, WSISA model comprises a series of CNNs, each of which was trained with a cluster of similar patch samples collected from all training WSIs, respectively. Therefore, the representations extracted by CNNs were over-localized for WSIs since their receptive field is constrained to be less than a patch's size equivalent to a physical area of $0.063mm^2$. The pathological sections of nidus from patients contain more than the regions of interest (e.g. tumor cells), therefore, the representations drawn from random patches may not strongly correspond to the disease. Furthermore, it has been widely recognized that the topological properties of instances on pathological images are crucial for a wide range of medical tasks, including cell subtype classification and cancer prediction. While, WSISA is neither able to learn any topological representations from WSIs nor to construct feature maps upon given topological structures. Therefore, the proposed GCN based pipeline is attempting to build graph representation with patch-level topological structure preserved.

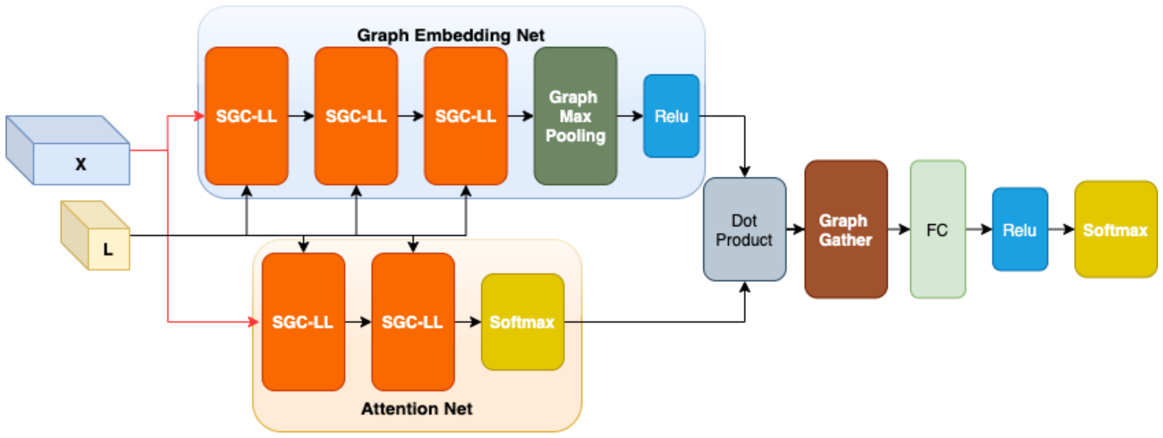


Figure 3.1: Attentional- Adaptive Graph Convolutional Network.

3.3 Methodology

3.3.1 Attentional AGCN

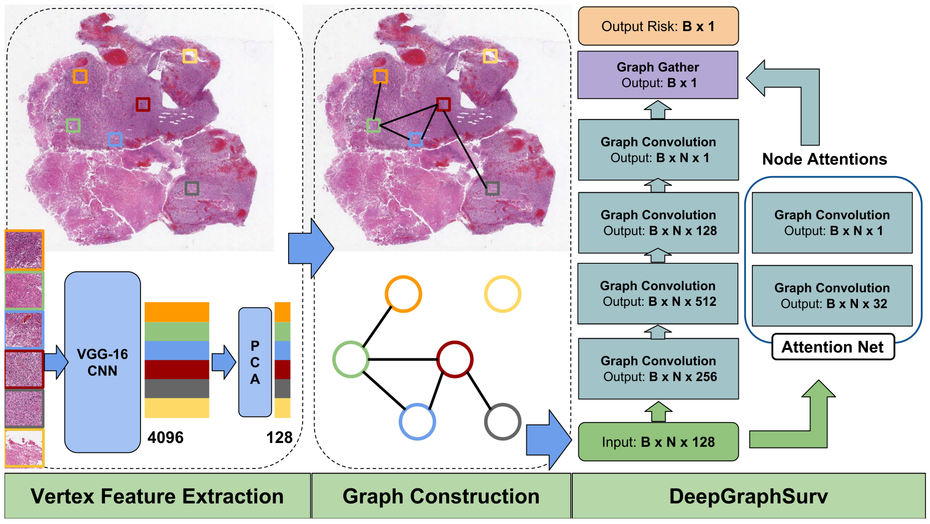
GAT [20] computes pair-wise attention coefficients between a node and its neighborhood using node features and learned transformation $\{W, a\}$ shared across nodes. Given that the parameter scale of attentional layer is irrelevant to graph size and that GAT only consists of attentional layers, similar to AGCN, GAT is also able to accept graph data of diverse structures. However, attentional layers in GAT only aggregate features from attended neighbors, i.e. the first-order vertices. Therefore, the representational capability of GAT is weaker than SGC-LL layers. Because, within a SGC-LL layer there are up to K-hop neighbors included in convolutional kernel, equivalent to a receptive filed of size $(2K + 1) \times (2K + 1)$ on grid. In practice, we set $K = 2$. While, GAT's attentional layers aggregate neighbors with an equivalent 3×3 kernel on grid in practice, which is much smaller than tthe receptive field of SGC-LL. To enable aggregating features from remote nodes to graph level, GAT has to stack more layers, even though each layer GAT has more lightweighted calculation. To utilize the advantage of SGC-LL, we use a stack of SGC-LL layers to construct a fully graph convolutional network as attention network, which is to give a prediction of node-wise attention coefficient $e_i \forall i \in V, G(V,E)$. One simple difference between SGC-LL in attention network and those in AGCN backbone is the output feature dimensionality is 1, instead of d' which is the length of graph representation. At last, softmax operator would normalize the attentional coefficients and have their summation as 1 on each graph.

$$y = \sum_{i \forall V, G(V.E)} \alpha_i \dot{x}'_i \tag{3.4}$$

The Figure 3.1 illustrated an example of attentional AGCN model consists of two parallel components: AGCN and attention network. The former AGCN is to give graph representation and latter is to learn weight on each node. In our setup, we did not squeeze the size of input graph or downsample graph nodes or their neighbors. After both graph representation x'_i and normalized node-wise attention values α_i are both learned, a dot-product would be applied with them two to give a weighted gathering of node features and output a final graph representation y (see Figure 3.4).

3.3.2 End-to-end Survival Framework: DeepGraphSurv

Medical image seems a more direct observation compared to other formats of patient data toward an accurate survival time prediction. While, prior to CNNs, medical imaging analysis is based on handcrafted features, irrelevant to survival. On positive side, it has less chance of overfitting, but its accuracy and robustness are both unsatisfactory. CNNs are proved to be able to generate more comprehensive and generic representations of medical images. However, due to the tremendous data scale of whole slides pathological images (WSIs), no existing CNNs are able to accept the WSIs without down-sampling or cropping. State-of-the-arts of CNN based survival data models were all trained with sampled patches losing the informative



Output Risk: B x 1
Graph Gather Output: B x 1
Graph Convolution Output: B x N x 1
Graph Convolution Output: B x N x 128
Graph Convolution Output: B x N x 512
Graph Convolution Output: B x N x 256
Input: B x N x 128
Node Attentions
Graph Convolution Output: B x N x 1
Graph Convolution Output: B x N x 32
Attention Net
VGG-16 CNN
PCA
4096
128
Vertex Feature Extraction
Graph Construction
DeepGraphSurv

Figure 3.2: DeepGraphSurv: End-to-end Graph Neural Networks (GNNs) based Survival Prediction.

global topological structures among patches, which is crucial for making decision for the entire WSI.

Graph is widely employed to represent topological structures among entities. However, modeling a WSI as graph is not straightforward. Cell-graph [53] is infeasible for tasks on WSIs due to the huge number of cells included and that many of them are possibly noisy nodes, i.e. isolated cells. To control the complexity of overall approach, the granularity of our model is set at patch level, for local substructure smaller than patches, we assumed that the CNN for patch feature extraction is able to represent them and include in patch embedding. To construct graphs for a WSI, patches become graph nodes, and the graph edges were to be built from the scratch. The extracted patch embeddings are, therefore, the original node features, when constructing node-wise connections using methods like clustering. Given a cluster, we set an edge appear on any two nodes belong to the same cluster. Not all sampled patches will be used. For quality assurance purpose, we may have to dump some patches drawn from the marginal areas in which few cells are included. The extensive cleaning preprocessing was done via a visual check by professionals. Therefore, the cardinality of resulted patch samples per graph differs. Namely, the graphs that represent WSIs are of different number of nodes. In our experiment, vertex features are generated by the VGG-16 network pre-trained on ImageNet [59]. Due to the lack of patch labels, we cannot fine-tune the network on WSI patches. We will introduce how the proposed graph CNN model mitigates this deficiency in next session. The graph edges were initialized by thresholding the Euclidean distances between all patch pairs. The distance was calculated using the 128-dimensional node features that were first generated by a VGG-16 pre-trained network and then compressed by principal component analysis (PCA) [60].

After graph is constructed, with minor change on output dimension, the SGC-LL layer is able to generate an attentional mask over graph, equivalent to the importance of each graph node in final graph representations. With a similar architecture as AGCN proposed in [1], we created a graph attention network that comprises a stack of SGC-LL layers, parallel to the AGCN that aimed at graph embedding learning, to learn the normalized node-wise attentional coefficients. The output of the graph attentional mask is about to be applied in the final weighted gather layer, elaborated in Eq 3.4.

As shown in Figure 3.2, it elaborated the proposed end-to-end survival prediction pipeline, named as *DeepGraphSurv* [2]. It includes several components: patch sampling on WSI, feature extraction and compression (PCA), graph construction, graph convolutional network and partial likelihood as loss function. Patch sampling first converts a WSI into a bag of patches, each of which then get represented as a 4096-dimensional embedding from a pre-trained VGG-16 network, after PCA, the node feature vector becomes 128-dimensional vector. Finally, a graph was built using the short node embedding. Different from previous deep learning-based survival models which simply act as feature extractor [52], ignoring patch-wise geographical relationship.

DeepGraphSurv is able to directly generate survival risks. The original proportional hazard function:

$$\lambda(t/y_i) = \lambda_0(t) exp(y_i) \tag{3.5}$$

, where $y_i \in Y = \{y_0, \cdots, y_{M-1}\}$, the M graph samples (WSI) in batch. λ_0 is the base hazard value, while there is no need to figure out the exact value, since the loss function is based on the normalized partial likelihood for one event. And this event in the context of our research is the death of patient i at time t_i, where t_i is the censorship time of patient i when the patient i is still alive at time t_i. Since eventually we focus on predicting the ordering of patients' death risk, therefore the likelihood of risk for patient i:

$$L_i = \frac{\lambda(t_i/y_i)}{\sum_{j \in \{j: t_j \geq t_i\}} \lambda(t_j/y_j)} = \frac{\lambda_0 exp(y_i)}{\sum_{j \in \{j: t_j \geq t_i\}} \lambda_0 exp(y_j)} \tag{3.6}$$

, which is the loss function used in DeepGraphSurv pipeline. The censorship time $t_j \geq t_i$, then patient j's risk becomes part of denominator in Eq 3.6, because we need to minimize the risks from those patients who stay alive after censorship time t_i and maximize the risk of patient i at censorship time t_i. The actual loss function for a batch of patients (WSIs) is the negative Cox log partial likelihood for censored survival data as below [56]:

$$L(Y) = \log \prod_{i=0}^{M-1} L_i = \sum_{i \in \{i, S_i = 1\}} \big(-y_i + \log \sum_{j \in \{j: t_j \geq t_i\}} exp(y_j) \big) \tag{3.7}$$

, where S_i and t_i are respectively the censor status and the survival time measurement of patient i. Eq 3.7 is evaluated in batch-wise, since the ordering has to be evaluated with a batch of samples. $\{y_i, y_j\}$ are the risk values generated by *DeepGraphSurv* for graph sample $\{i, j\}$. The penalty is generated by those sample (patients) whose survival time $t_j \geq t_i$, while his risk y_j is significantly higher than y_i. During the training, the graph embedding x'_i and the survival-related residual graph A' of WSI patches are accessible at each SGC-LL layer. And the later layers usually provide more high-level topology-aware features about WSI. We also visualize the attention coefficients of graph node on the actual coordinates of corresponding patch on WSI in experiment session.

3.4 Experiment

3.4.1 Dataset

As to experimental benchmark dataset, we utilized the whole slide pathological images (WSIs) from a generic cancer patient dataset TCGA [61], which was originally released by The Cancer Genome Atlas project, whose research objective is to discover correlation between genetic errors in DNA and the occurrence of 33 cancer subtypes. We trained and evaluated the baseline models and the introduced DeepGraphSurv over the WSIs associated with two common cancer subtypes from the TCGA dataset: glioblastoma multiforme (GBM) and lung squamous cell carcinoma (LUSC). Besides, NLST (National Lung Screening Trials [62]) is another medical research that employed $53,454$ heavy smokers, whose age 55 to 74 with at least *30*-year smoking history, as the high-risk patient group for lung cancer survival modeling and analysis. We also committed an experiment over the WSIs data of NLST that consists of both squamous-cell carcinoma (SCC) and adenocarcinoma (ADC) patients and their lung tissue images to evaluate the performance of our model on survival prediction for patients of mixed tumor subtype. The numeric facts on the datasets in the experiments are listed in Table 3.1. Some patients have multiple WSIs collected and included in dataset, we executed models on all data and report the average prediction per patient. Because the constructed graphs over patches on WSI is data-specific, not patient-specific instead. Data quality in Table 3.1 is mostly related to image resolution. Average size of single WSI is to emphasize the challenges in experiments, because loading these WSIs of that size to memory is already difficult.

Data Source	Cancer Type	No. Patient	No. WSI	Quality	Avg. Size
TCGA	LUSC	464	535	medium	0.74 Gbyte
TCGA	GBM	365	491	low	0.5 Gbyte
NLST	ADC & SCC	263	425	high	0.74 Gbyte

Table 3.1: The Statistics of Whole Slides Pathological Image (WSI) Datasets.

3.4.2 Baselines

The baseline models included in survival prediction experiments are divided into two categories: classic methods and deep learning-based end-to-end methods. Classic methods, such as LASSO-Cox linear model [47], BoostCI [63] and a multi-task learning framework proposed for Survival Analysis, called MTLSA [64] are not able to directly output survival probability or survival time, and the regression is executed after extracting features from the raw data, no matter the data type is text or image. Therefore, the performances of classic methods largely depend on the quality of extracted hand-crafted features from raw data. Unfortunately, they were entirely not designed for WSI-based survival analysis, which requires extensive calculation. And no hand-crafted features are designed particularly for images at the scale of WSI. Therefore, to have a fair comparison, we first feed those baselines with the selected predefined features extracted over patches by CellProfiler [65], an open-source scientific software for cell image analysis. Patches on WSIs were randomly sampled, and the local features of WSI were the averaged ones of all sampled patches. Furthermore, we also feed the classic survival models with the WSI-level features generated by DeepGraphSurv, in order to demonstrate the unique gain on performance brought by the end-to-end fine-tuned topology-preserved features from the new network.

Besides classic baselines replying on pre-calculated features, we compared DeepGraphSurv with the cutting-edge deep learning-based survival models on WSI. WSISA [52] is the first approach that directly works on WSI. While, training of WSISA is expensive and unscalable. It is required to train a CNN, e.g. VGG-16, for each cluster of patch samples. Therefore, WSISA neglected the wide-existing topological relationships among the patches, which are of great value to survival analysis. As opposite, Graph CNNs are built upon the topological structures and have recognized power of learning structured features on graph-structured data. To demonstrate the capability of graph feature learning over data with no intrinsic graph structure given, we add another GCN as baseline method. We concatenate the graph representation output by ChevNet [10] with the proportional Cox regression function to give risk prediction. While, [10] executed convolution over a pre-defined, fixed spectrum kernel, lack of adaptiveness if the graph initialization is bad.

3.4.3 Result and Discussion

As far as we know, DeepGraphSurv is the first survival model that utilizes graph-based attention scheme. As shown in Figure 3.3, after 40 epochs of training, the regions that comprise the patches of high attentional coefficients have correctly highlighted the most of Regions of Interest (RoIs) corresponding to tumor-related cell clusters. The ground-truth of RoIs were annotated by experts. The embedding learned on those patches of high attentional coefficient will be of higher weights in the final representation of WSI, and then if the higher coefficients predicted by attentional network geographically coincide with the RoIs related to tumor cells, the final graph

Model	LUSC	GBM	NLST
LASSO-Cox [47]	0.5280	0.5280	0.4738
LASSO-Cox + DeepGraphSurv embedding	0.5663	0.5165	0.5663
BoostCI [63]	0.5633	0.5543	0.5705
BoostCI + DeepGraphSurv embedding	0.5800	0.5130	0.5716
EnCox [66]	0.5216	0.5597	0.4883
EnCox + DeepGraphSurv embedding	0.5740	0.5231	0.5742
RSF [67]	0.5066	0.5570	0.5964
RSF + DeepGraphSurv embedding	0.5492	0.5193	0.5491
MTLSA [64]	0.5386	0.5787	0.6042
MTLSA + DeepGraphSurv embedding	0.5247	0.5630	0.5573
WSISA [52]	0.6380	0.5760	0.6539
ChevNet [10] + Cox regression	0.6280	0.5901	0.6845
DeepGraphSurv [2]	0.6606	0.6215	0.7066

Table 3.2: Concordance Index (C-index) of DeepGraphSurv and the State-of-the-art Survival Models.

representation will be consequently more tumor-oriented and more helpful in survival prediction.

The concordance probability (C-index) is the measurement of survival prediction. It is defined as the fraction of all pairs of patients whose predicted survival times/risks are correctly ordered as all censored patients that can be reasonably ordered. Formulating survival order as directed graph G_t where the edge $\epsilon_{i,j}$ implies

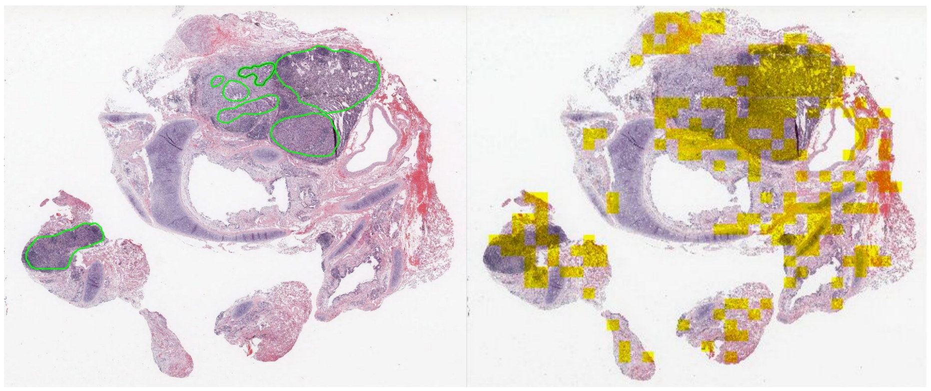


Figure 3.3: Visualization of node-to-graph attention on WSI.

survival time $T_i < T_j$, then C-index of graph G_t, given the risk prediction $f(x)$, is defined as:

is the indicator function formulated as: $1_{a<b} = 1$ if $a < b$, otherwise 0. $f(x)$ is the risk predicted by survival model for WSI x. When a patient has more than one WSI included in dataset, we average the predicted risks for this patient before calculating the C-index. A model who is able to more correctly order the censored patients by the predicted survival time is supposed to have higher C-index. The C-index result of the baseline models and the introduced DeepGraphSurv from three experiments are reported in Table 3.2.

Testing dataset was separated from the training set to avoid data leakage. The classic survival models, e.g. LASSO-Cox [47], failed to deliver compelling prediction accuracy on WSI datasets, some possible explanations:

1. sampled patches are only part of a WSI given the computational constraint;
2. the data quality of patches may vary case by case.

Therefore, the features extracted from randomly sampled patches bring a noisy and biased representation of WSIs. Moreover, the hand-crafted features offered by CellProfiler are the generic descriptors of pathological images, not particularly designed for images at the scale of WSI. Consequently, we believe it is the quality of hand-crafted features that limit the performance of classic survival models. After feeding the topology-aware graph embedding generated by DeepGraphSurv to classic baselines as input, the C-index showed a notable lift as large as 0.04 on average on NLST and LUSC datasets. And the outcome supported our argument that the features fine-tuned with survival labels are better representations of WSIs for survival prediction purpose. However, we also observe that, due to the lower image quality of GBM WSI data, only using the fine-tuned patch features cannot improve prediction accuracy. Because we see that WSISA, a CNN based patch feature extractor, cannot beat MTLSA, which still use hand-crafted features as input. This means that, given a low image quality, CNNs cannot always learn a better representation of image from scratch than heuristic features, even though survival censorship labels were used as penalty in the training of CNNs.

DeepGraphSurv generates predictions by encoding patch features with their topological structure via spectral graph convolutions. When the patch features are not discriminative, the topological structure among the patch instances will play a significant role in recognition of survival patterns of WSI. This explains the additional accuracy given by DeepGraphSurv compared to WSISA on GBM dataset, which is of lower resolution. On other dataset of higher quality, DeepGraphSurv is able to deliver a larger margin of gain on C-index compared to the baselines who cannot learn anything from topological structures. The baseline GCN + Cox is the model in which we use the graph representation learned from ChevNet with LASSO-Cox survival regression. Compared to this baseline, on all 3 datasets DeepGraphSurv still showed a lift, that comes from a better hidden representation disclosed on residual graphs built upon the learned distance metrics. Due to that WSI data do not have any intrinsic graph structures, the initial graphs constructed with the patch embeddings from a VGG-16 network are not guaranteed to be optimal in terms of learning graph representations. While, DeepGraphSurv makes graph structure trainable and adaptive in learning graph representations. Besides, the introduced end-to-end approach is able to directly generate risk prediction, the entire network including the survival regressor are optimized jointly.

3.5 Conclusion

Survival prediction is a useful clinical intervention tool, although it cannot act as expected in many scenarios. Efficient mining of survival-related structured features on whole slide images is a promising solution of boosting survival analysis. In this paper, we suggested to model WSI as graph and proposed DeepGraphSurv to learn global topological representations of WSI. Instead of unsupervised graph, DeepGraph-Surv creatively utilized a survival-specific graph trained under supervision of survival labels. The effectiveness of our model has been confirmed by improved accuracy of risk ranking on multiple cancer patient datasets across carcinoma subtypes.

CHAPTER 4

Fast Regions-of-Interest Detection in Whole Slide Histopathology Images

4.1 Introduction

At our age, many hazardous infectious diseases, e.g. bird flu, and many different kind of cancers, e.g. lung cancer, are still the top threats to our personal health and the public sanitation as well. Automatic searching and localizing Regions of Interest (ROIs) on histopathological images is a crucial intermediate step between large-scale images acquisition and the computer-aided automated diagnosis that we pursue. As the fast development of deep learning techniques and the introduction of neural network models, e.g. convolutional neural networks (CNNs), to medical image understanding area, we are finally able to extend the boundary of modern medical image saliency detection, classification and segmentation [68, 69, 70]. Whole Slide Images (WSIs) are the digitized histopathology images taken over an entire slide of tissue, which retrains as much intact pathological information as possible. Therefore, a typical WSI, that usually has resolution at scale of $10^6 \times 10^6$, is $1.5 \sim 2.0$ Gigabyte large on disk, which is thousands times larger than those images from deep learning benchmark datasets, like MNIST [71] and CIFAR [72]. Therefore, traditional fully convolutional networks, used to work perfectly for medical image segmentation [73], are no longer applicable, because of the parameter scale that may explode and the rising risk of under-fitting along with lack of labeled WSIs for training. We need a brand-new cost-efficient solution designed especially for WSIs to handle such magnificent scale of data without losing too much performance. As far as we know, there is no existing convolutional neural networks who claim themselves to directly work on raw images at WSI scale without any downsampling or patching. The most popular walk-around for extracting features from WSIs is to first sample a bag of patches over WSIs and then train and execute inference on patches respectively. Then, aggregating the prediction from patch level to WSI level is to give final model output. Patch-based network [68] successfully handled classification task on WSIs, [74] enabled survival time inference purely based on tumor tissue WSIs. Although, these models applied to WSIs successfully saved most of computational cost by patching, they also dumped lots of task-relevant information hidden in those patches not being sampled. Besides, losing topological spatial information of patches after being sampled from WSI makes predictor treat patches equally, which is obviously not the optimal strategy.

Considering the practical clinic scenarios for image detection and segmentation techniques applied to CT [73] and MRI [**?**] and the associated pathophysiological procedures, we summarized some challenging but necessary technical requirements for any ROI detection and segmentation solutions for WSIs:

1. high time and energy efficiency. To make it scalable, the ROI detection and localization is supposed to be accomplished within short period of time with high recall and acceptable precision.
2. high fidelity and high trustworthiness on generated ROIs of WSI. We need to quickly and correctly classify if a proposed ROI belongs to, at least partly, ground truth ROIs. Because the ROI prediction may largely affect downstream tasks, e.g. disease diagnosis decisions.

Regions of interest (ROI) could have different definition according to particular scenarios. In this article, we name ROIs as the local regions filled with tumor cell cluster or other cancer-related cells such as lymphocyte. In past related works, ROI detection and segmentation are usually treated separately as two different tasks. The former is to quickly search and localize any suspicious regions on image according to predefined patterns. The output of this task may not have to be fine-detailed at pixel level, due to computational efficiency concern, and sometimes a bounding box that surrounds, at least partly, the ground-truth ROI is enough satisfactory. While, the latter task is to give a pixel-accurate contour of each detected ROIs, which is significantly more expensive. In fact, detection and segmentation are not strictly isolated, and on the opposite, the two tasks could be combined as one under some circumstances. Many CNNs based image segmentation models are indeed end-to-end solutions directly extract and learn hierarchical feature pyramid from raw channels of images to execute pixel-clustering at different level of granularity. Semantic segmentation network [75] is to obtain object detection and segmentation in single forward-pass of network. The advantages of applying deep neural network is from treating the feature design work as an optimization problem, and therefore CNNs are able to discover hidden representations that better serve prediction tasks than handcrafted descriptors, who are either over-localized or not robust. The requirement on high recall rules out patch-based WSI solutions. And patch-based methods obviously cannot handle segmentation of entire WSIs. However, in order to directly work on WSI input, the networks either make the receptive field of convolutional operators large enough to cover any potential region of interest, or stack more layers with relatively small kernel to aggregate local features from entire ROI to form its high-level representations for classification. No matter what architecture is chosen, the total count of parameter in WSI segmentation network is going to be magnificent. Due to the expense of having quality annotation of all ROIs (i.e. tumor cells) on WSIs, the annotated ground-truths of segmentation for training models is quite constrained and very likely not enough to train a wide and deep network as described above that directly works on raw WSIs.

To work around this difficulty, in the method to be introduced, we first chose to still rely on handcrafted features as descriptors of patches to save massive feature aggregation calculations in CNNs, and in the meanwhile we also utilized the hierarchical pyramid structures appear between feature maps of consecutive convolutional layers of CNNs. While, different from what happened in CNNs, in the pyramid of introduced multi-level iterative method, feature vectors of descriptor are not changed along with level, because we did not have gradients back-propagated from loss to update feature formations, while the spatial segmentation did get updated at different granularity of patching. Without having ground-truth of segmentation of ROIs, we introduced superpixel clustering as an unsupervised way to learn spatial segmentation of image, since we do not have gradient to update the assignment of segmentations as well. At different level granularity, we divide the entire WSI into patches of different scale, then the introduced superpixel clustering method [4] is going to cluster patches based on several handcrafted local textual descriptors, preserving both topological consistency and appearance similarity. After superpixel constructed, we run a pre-trained classifier, e.g. SVM or CRF, to classify superpixels represented by the averaged descriptors of patches. Averaging of patch descriptors is to avoid additional difficulty of training a classifier for superpixels of different size and varying shape. This is also the biggest challenge for building an end-to-end fully convolutional network fed with clustered superpixels, since the shape of input tensor to any neural network cannot be undefined.

The main contributions of article is to decouple and reformulate ROI detection and semantic segmentation, that requires dense annotation, into an iterative execution of unsupervised superpixel clustering and classification at coarse-to-fine level of patching granularity. This semi-supervised approach largely replies on quality of superpixel clustering. To obtain better fine-detailed superpixels, we introduced a novel topology-preserved superpixel clustering algorithm to this problem. Besides, the approach introduced is also dependent on accurate classification of superpixels, especially at coarser levels, because any mistaken classification of coarse superpixel cannot be compensated in fine-grained superpixel refinement at next level of granularity. The recall of ROIs will benefit from the increased classification accuracy. Therefore, we trained compact but robust classifier, e.g. SVM, with minimal data requirement. On the other hand, without fine-tune, an improved segmentation of superpixels will automatically boost accuracy of a pre-trained classifier.

4.2 Related Work

Superpixel is a common replacement of pixel with purposes more than saving computational cost. It clusters nearby pixels of similar attributes together as fundamental operational unit in downstream tasks, e.g. object detection, segmentation and even realtime tracking. In this session we introduced the state-of-the-art superpixel clustering algorithms and the combination of superpixel with deep neural networks (DNNs) in medical image understanding.

4.2.1 Superpixel Clustering

One important feature of superpixel construction is that this is a pure unsupervised approach in which there are no annotated ground-truths in any format for guiding the label assignment on pixels. The pixels are clustered purely based on the attributes, such as appearance and physical location, etc. SLIC [76] is an iterative K-mean superpixel clustering that walk through all pixels. It is able to generate almost equally-sized superpixels with outstanding boundary adherence. And the time complexity could be further reduced by limiting search space to a small nearby area.

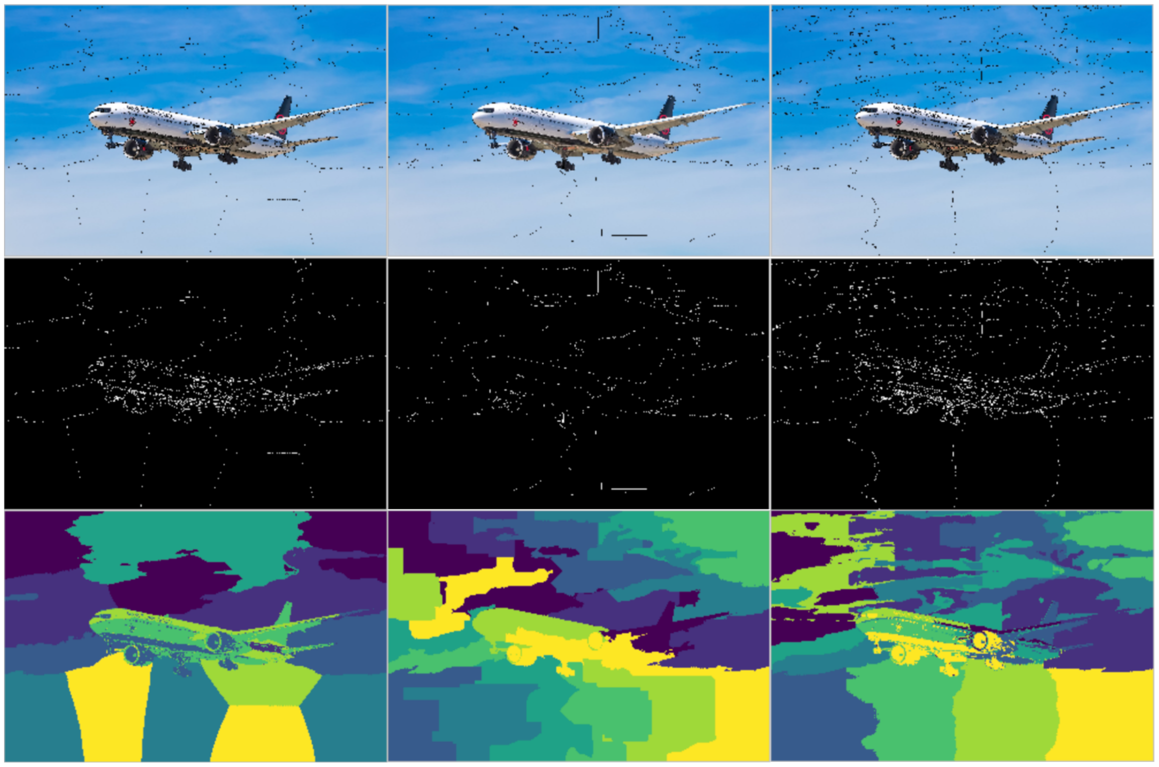


Figure 4.1: Example of superpixel clustering on image with three classic solutions: LSC, SEEDS and SLIC.

While, iterating over entire pixels is still too expensive, stopping SLIC from being applied on large images like WSIs. If compromise part of accuracy, SEEDS [77] , that started from randomly initialized superpixel partitions, focused on updating boundary pixel allocation only and proposed a fast energy function to evaluate each adaption of pixel label assignment by enforcing color homogeneity. Linear spectral clustering, a.k.a. LSC, combined normalized cut and K-mean clustering after discovering optimizing these two objective functions are in fact equivalent on the condition that defines similar function as inner product of feature vectors [78]. LSC also achieved satisfactory boundary adherence and color consistency within segmented superpixels with $\mathcal{O}(N)$ complexity, where N is the pixel number. Compared to SLIC, LSC saved computations from pre-allocation of pixel to large regions by eigenvector-based normalized cuts. And different from the two-stage Ncuts [79], LSC accomplished Ncuts and K-mean in one-stage. Similar to LSC, the computational complexity of SEEDS and SLIC is also approximated as $\mathcal{O}(N)$. Therefore, within visual comparison in Figure 4.1, we did not include expensive solutions such as ERS [80] with $\mathcal{O}(N^2 \log N)$ and EneOpt0 [81] with $\mathcal{O}(N^3)$ complexity. Because we only consider those approaches who are potentially feasible for segmenting whole slide images. Figure 4.1 showed superpixel clustering on image with three classic solutions: LSC [78] (left), SEEDS [77] (middle) and SLIC [76] (right). The upper row is the edges of superpixels displayed on image. The middle row is the contours of superpixels. The bottom row is the segmentation mask filled with different color on different superpixel.

4.2.2 ROI and Superpixel

Regions of interest (ROI) in histopathology whole slide images (WSIs) are usually those disease-related cells or the tissues of specific patterns, but they do not have descriptive definitions to form a category of objects. Due to the magnificent scale of WSI, the major challenge would be the scalability and the memory efficiency of algorithms. [82] relied on cheap segmentation of superpixels on downsampled WSIs to filter out those regions irrelevant to ROIs. However, it did not correctly notice the inevitable influence of wrong classification of coarse superpixels, because the algorithm completely ruled out those regions from later more accurate segmentation and classification. Besides, the classifiers had to be trained multiple times with patches extracted from the superpixels of different magnification to work on different levels of granularity. [83] reduced the workload of labeling and grading by two ways: by excluding the areas of definitely normal tissues within a single specimen or by excluding entire specimens which do not contain any tumor cells. [83] presented a multi-resolution cancer detection algorithm to boost the latter. while it also suffered

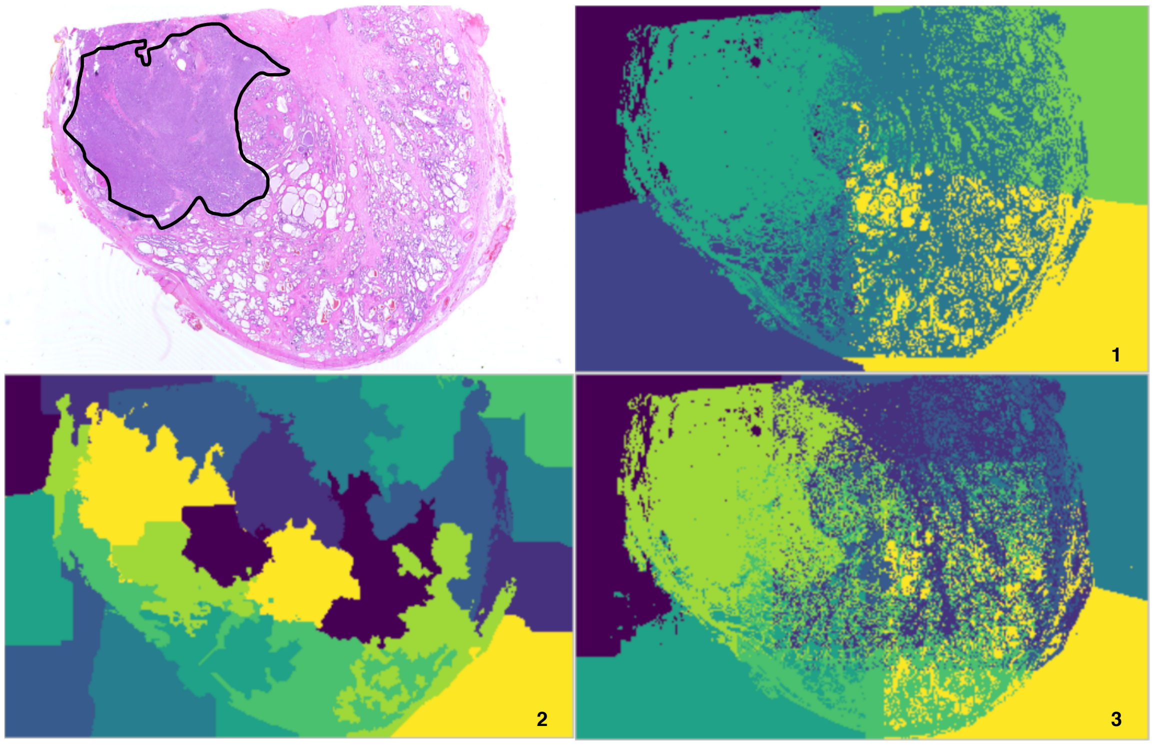


Figure 4.2: Example of pathological whole slide image with ROI annotations and the superpixels.

from the loss of recall as [82]. Another superpixel automated segmentation method is [84], which trained a classifier to predict where mitochondrial boundaries occur using diverse cues from superpixel graph. While, because the selected superpixel clustering approach [76] did not offer satisfactory boundary adherence, the classifier encumber the overall detection performance. As summary, in order to accomplish a quick detection and segmentation of ROIs in WSI, a combination of superpixel clustering and pre-trained classifier seems a popular choice, while the performance bottleneck was the tradeoff between the efficiency and the quality of superpixel clustering, which directly determined classifier accuracy.

To reduce the intense computational cost in superpixel clustering, the algorithm to be introduced creatively combined the coarse-to-fine scheme [85] and the boundary-only update strategy proposed in SPSS [86]. In our method, clustering manipulated the rectangular blocks of pixel as basic unit and a coarse segmentation of superpixel would be constructed before a more fine-detailed refinement got executed. on each level of construction, only boundary blocks or their nearby neighbors got chance of label update. Figure 4.3 illustrated the procedures of introduced superpixel clustering. Furthermore, the introduced boundary-only update strategy on next level would emphasize on differentiating foreground and background blocks, considering the boundaries between superpixels within ROIs are less important. The improvement brought by our algorithm on ROI detection accuracy has been proved and verified in [4, 51], where the method had quantitatively verify the improvement of the accuracy of ROI detection in histopathology images, e.g. lung cancer H&E-stained WSIs. Figure 4.2 shows comparison of classic superpixel methods [78, 77, 76] on cancer patients WSI.

4.2.3 DNNs on Superpixel

As success of deep neural networks in computer vision, many works have extended application of DNNs onto superpixel. [35] introduced a bilateral inceptions module to accelerate convergence of CNNs with superpixel as network input for semantic segmentation. [87] treated superpixels as ”pooling” layer in neural network, but preserving low-level structures.Therefore, their framework trained semantic segmentation network without pixel-level ground-truth. To construct superpixels for small objects of complicated boundaries, [88] introduced a superpixel segmentation based on pixel features trained with affinity loss and segmentation error. In medical images domain, superpixels are also utilized as a topology-preserving simplification of data for deep network. The organ segmentation network in [89] worked on the descriptors extracted from superpixels clustered in CT images. And then CNN simply did a pixel-wise refinement based on the coarse segmentation given by superpixel. Different from previous works who simply utilized superpixels as reduction of image primitives, [90] proposed an end-to-end "Superpixel Sampling Network" (SSN) which contains differentiable superpixel construction together with learning a task-specific prediction.

The rest of article is organized as following: we first introduce the multi-resolution fast superpixel clustering with coarse-to-fine and boundary-only strategy to increase efficiency. Both mathematical explanation and illustrative examples will be given in Section 3. Then we elaborate the numerical results on classification accuracy and visual comparison of superpixels with classic methods on TCGA WSI dataset in Session 4. Lastly, conclusion and future work will be given in Session 5.

4.3 Methodology

The detection framework introduced is not only going to propose bounding box to surround ROIs, but also is going to offer fine-detailed, boundary-adherent superpixel segmentation of them. On the other hand, an improved superpixel construction contributes the differentiation of ROI from background as well. Therefore, the proposed approach comprised two components: fine-detailed superpixel segmentation and superpixel classification. For reduction of computational expense, we chose not to accomplish superpixel segmentation at finest level in one shot. For instead, we first obtain a coarse superpixel segmentation from clustering big pixel blocks (e.g. 500×500). A pre-trained binary classifier then predicts label (ROI v.s. background) of superpixels. Afterwards, those superpixels labeled as ROI along with their neighbors will move to next round of segmentation at finer resolution. The process will be repeated until quality becomes satisfactory. Different from previous superpixel clustering methods [76, 86], the introduced algorithm gave topology-preserving superpixels. A better detection recall is expected as well, since our method did not completely rule out negatively labeled superpixel at coarse stage as [82, 83], and for instead we include negative neighbor superpixels to next level of segmentation.

4.3.1 Superpixel Clustering and Detection

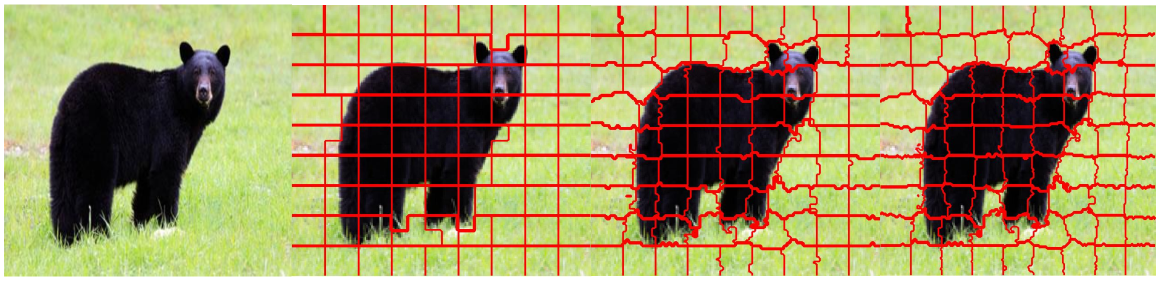


Figure 4.3: An example of the coarse-to-fine/boundary-only update based superpixel segmentation algorithm first presented in [4].

4.3.1.1 Energy Function

Think of superpixels of flexible number of blocks $\mathbf{S} = \{s_0, \cdots, s_{K-1}\}$, and the blocks belong to superpixel $\mathbf{S}_k$ as $\{b_0, \cdots, b_{M-1}\}$, we devised two representations of block: appearance and position. Appearance representation of block is the averaged RGB color over pixels in block as $\mathbf{C}$. Position representation of block is the relative position coordinates at center point of block as $\mathbf{P}$. At superpixel level, $\Theta = (\theta_0, \cdots, \theta_{K-1})$ and $\Xi = (\xi_0, \cdots, \xi_{K-1})$ are the center positions and the mean color vectors of superpixels. The objective function to be minimized consists of a series of energy functions and penalty terms. For appearance, total variance of three color channels are color energy function of superpixel $\mathbf{S}_k$ defined as:

$$E_{col}(\mathbf{S}_k) = \sum_{q=0}^{2} \frac{1}{\|s\|} \sum_{b \in \mathbf{S}_k} (c_b^q - \xi_k^q)^2, \tag{4.1}$$

also known as appearance coherence. For position, the averaged $l2$ distance from block position $\mathbf{P}_b$ to the centre position of its superpixel is the position energy function, $E_{pos}(\mathbf{S}_k) = \frac{1}{\|S_k\|}\sum_{b \in \mathbf{S}_k} \|\mathbf{P}_b - \theta_k\|^2$. This is to ensure clustered blocks are geophysically close. Besides, to avoid seeing any superpixels with sophisticated boundary, we use the total boundary length as boundary penalty function. Furthermore, we constrain the minimal size of finalized superpixel to be at least 25% of initial size. If any update of block's belonging violates this constrain, we give infinity penalty to this update, therefore, the algorithm will reject such label assignment update.

$$P_{size}(\mathbf{S}_k) = \begin{cases} +inf, & size(\mathbf{S}_k) < 0.25 \times initialsize \\ 0, & otherwise. \end{cases} \tag{4.2}$$

Similar penalty would be applied, if the update causes any isolated blocks who are surrounded by blocks from other superpixels. This is to enforce all generated superpixels to be topologically connected.

4.3.1.2 Boundary-Only Update

To define boundary energy function, we need to define boundary block and length. If a block has any neighbor block from other superpixel, then it is a boundary block. The boundary length of block is the number of neighbor blocks that belong to other superpixel.

$$P_b(s) = \sum_{b \in \mathbf{S_k}} \sum_{b_n \in Neighbor(b)} S(\mathbf{S}_k, b_n), \tag{4.3}$$

where $S(\mathbf{S}_k, b_n)$ is the indicator function of superpixel belonging for block, which return 0 if $b_n \in S_k$, otherwise 1. In our algorithm, we first stack entire initial boundary blocks into a queue, then the iterative superpixel clustering algorithm will work on boundary blocks only for consideration of updating label (i.e. superpixel assignment) of block. This is so-called 'boundary-only update'. In other words, the non-boundary blocks will not be considered for label change until they become boundary blocks. When the algorithm decides to update the label of a block, its neighbor will be considered to become new boundary blocks. When using the boundary-only update, there are two things to notice: 1) when update the label of block, it definitely change the list of boundary blocks; 2) we need to append the new boundary block to the end of the list because and follow the FIFO principle when deciding the order of blocks for consideration of changing label, in order to avoid the risk of divergence given by correlated dimensions in coordinate descent optimization. The candidate superpixel labels for a boundary block to swap are limited to its neighbor superpixels, otherwise it will trigger the topology connectivity penalty by having an isolated block. Given a trial of label update, the algorithm compares the objective function values before and after the change to see whether and how much the change is able to drive energy down.

We elaborate objective function each step of updating block-wise superpixel label assignment as below:

$$E(\mathbf{S}) = \sum_{s}(E_{col}(s) + \lambda_{pos}E_{pos}(s) + \lambda_b P_b(s) + P_{topo}(s) + P_{size}(s)), \tag{4.4}$$

where λ_{pos}, λ_b are respectively the tradeoff coefficients for position energy function and boundary length penalty term. In practice, the regularization on superpixel size and topological connectivity will give infinite penalty on those superpixels of over-small size as $P_{size}(S_k) \approx inf$ and those of isolated blocks, i.e. $P_{topo}(S_k) \approx inf$. Therefore, the algorithm will always reject such label proposal that violates topology connectivity and size regularization. When superpixel assignment of a boundary block is updated, the algorithm will add its neighbor blocks to queue, because those non-boundary blocks are now next to other superpixels. The convergence will arrive when the queue is empty.

4.3.1.3 Coarse-to-Fine Detection

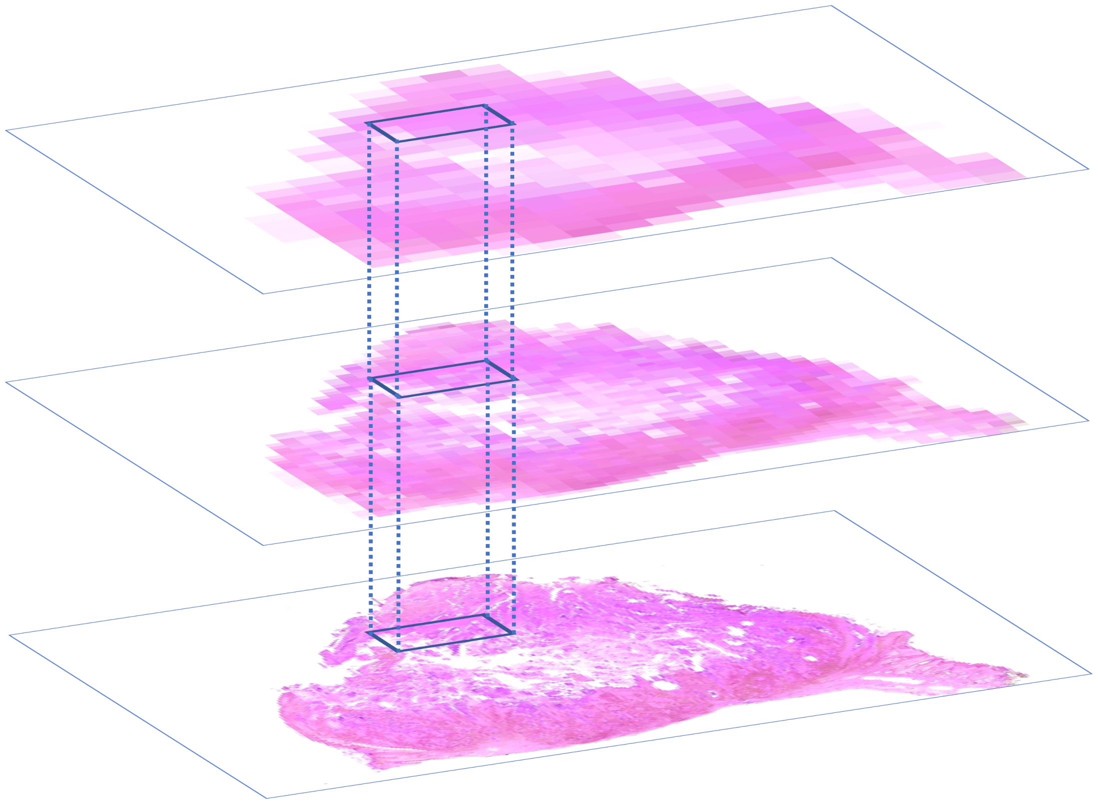


Figure 4.4: An illustration of multi-resolution process of ROI detection on WSI.

Instead of processing WSI at different resolutions [82], we cluster superpixels at coarse-to-fine level of resolution. [4] adopted boundary-only update as well to save unnecessary revisit to non-boundary blocks, while the boundary blocks on WSI may still be too much for extensive iterations. To further reduce the amount of data brought to finer update with more intense computation, we utilized a pre-trained

```
input  : Whole slide image - $W$, superpixel number - $K$
output: Superpixels - $\mathbf{S}$
1  for $l$ = 1 to levelMax do
2      if $l$ = 1 then
3          1. Initialize blocks $\mathbf{B}$ on level $l$ size on entire image $W$;
4          2. Initialize $K$ superpixels $\mathbf{S}$;
5          3. initialize $\Theta, \Xi$
6      end
7      else
8          1. Initialize blocks $\mathbf{B}$ on level $l$ size within positive superpixels and
            their neighbor superpixels $\hat{\mathbf{S}}$;
9          2. Initialize $\Theta, \Xi$ for $\hat{\mathbf{S}}$
10     end
11     Compute the mean color and position in each block;
12     Initialize $L$, the queue of boundary blocks on level $l$;
13     while length(L) $\neq$ 0 do
14         Pop out block $b_i^l$ from the queue ;
15         $E_{before} = E(\mathbf{S})$;
16         for $b_n \in Neighbor(b_i^l)$ do
17             change label of $b_i^l$ to neighbor $b_n$'s label ;
18             $E_{after}(b_n) = E(\mathbf{S})$;
19         end
20         find the $\hat{b}_n = arg\min_{b_n \in Neighbor(b_i^l)} E_{after}(b_n)$;
21         if $E_{after}(b_n) < E_{before}$ then
22             update label of $b_i^l$ to that of $\hat{b}_n$
23         end
24         append $Neighbor(b_i^l)$ to $L$;
25     end
26     run binary classifier on superpixels to predict ROI
```

classifier, e.g. SVM, to predict whether the superpixel belongs part of ROI. For any superpixel moved to finer update, smaller blocks will be initialized within its region. For example, a 10×10 block will be divided into 25 block of size 2×2 arranged at $5{\times}5$ grid. Boundary block queue will be refilled with 2×2 blocks who sit on superpixel boundaries. The classifier was trained using features extracted from patches sampled from ROI and non-ROI regions over annotated WSIs. To deal with different cardinality of patch per superpixel, we use pooling patch features at inference time. Given that we did not downsample images, therefore, the classifier trained on raw WSIs is able to be reused with different level of superpixel. Figure 4.4 showed an example has 3 level of granularity in term of block size. Note that we did not downsample the WSI directly, which dump falsely many local details, and we still include neighbor superpixels close to positive ones at coarse classification to next level. If the bounding box is the ROI (a rough identifier), as resolution goes high, superpixels cover and surround the bounding box will get fine-detailed update.

4.3.2 Complexity Analysis

Pixel-wise superpixel constructions [76, 77] have $\mathcal{O}(N)$ complexity, where N is number of pixel, while it made them infeasible on WSIs of trillions of pixels. The introduced algorithm is able to reduce the complexity to scale of number of block i.e. $\mathcal{O}(\sum_{k=0}^{K-1} \|\mathcal{S}_k\|) \ll \mathcal{O}(N)$. The boundary-only update, first presented in [4], further constrains involved blocks to those boundary blocks. Considering the purpose of clustered superpixel, our algorithm combined detection and superpixel clustering together, and it only executes finer segmentation within those coarse superpixels who were classified as ROI. It saved the calculations wasted on updating the superpixels that do not contribute to ROI detection. Due to the reduced dimensionality, the convergence comes faster than pixel-wise clustering methods. The Figure 4.5 showed

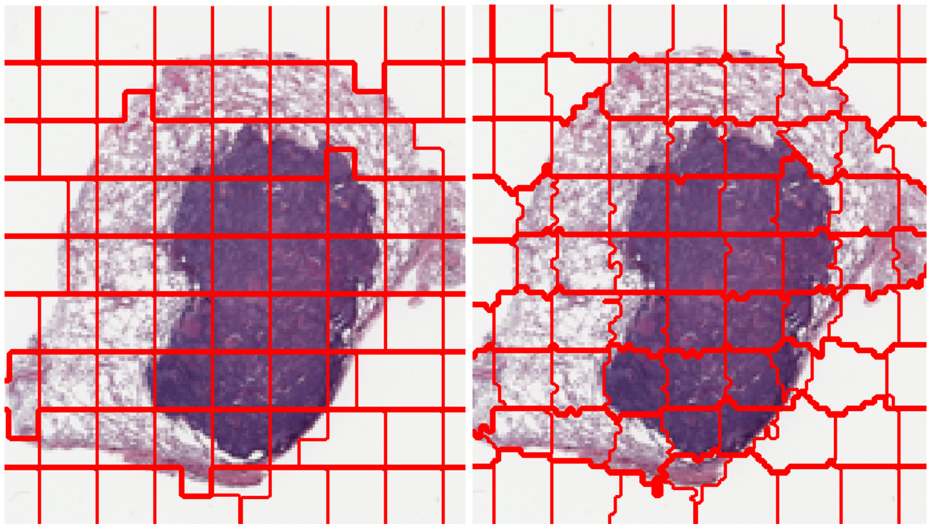


Figure 4.5: A coarse-to-fine superpixel clustering on a lung cancer WSI from NLST.

a coarse-to-fine superpixel clustering on a lung cancer WSI from NLST: 1) Coarse segmentation of superpixels using large blocks (180×180); 2) Refined segmentation with small blocks within selected superpixels.

4.4 Experiments

4.4.1 ROIs in Lung Cancer Histopathology WSI

In histopathology images like lung cancer WSIs, the regions of interest are those areas consist of cancer cells or other tissues that may be related to tumor diagnosis. A fast detection approach of ROIs is to search and localize those regions on image at WSI scale, that usually have trillions of pixels. Traditional pixel-wise methods and neural network cannot directly work on WSI, due to the extraordinary data scale and image dimensionality. Downsampling of WSI reduces complexity but also loses local fine-detailed features. Superpixels first cluster those pixels of similar spacial, color and topological properties as whole, and then in downstream tasks e.g. detection and segmentation, the superpixels will act as minimal manipulatable unit,

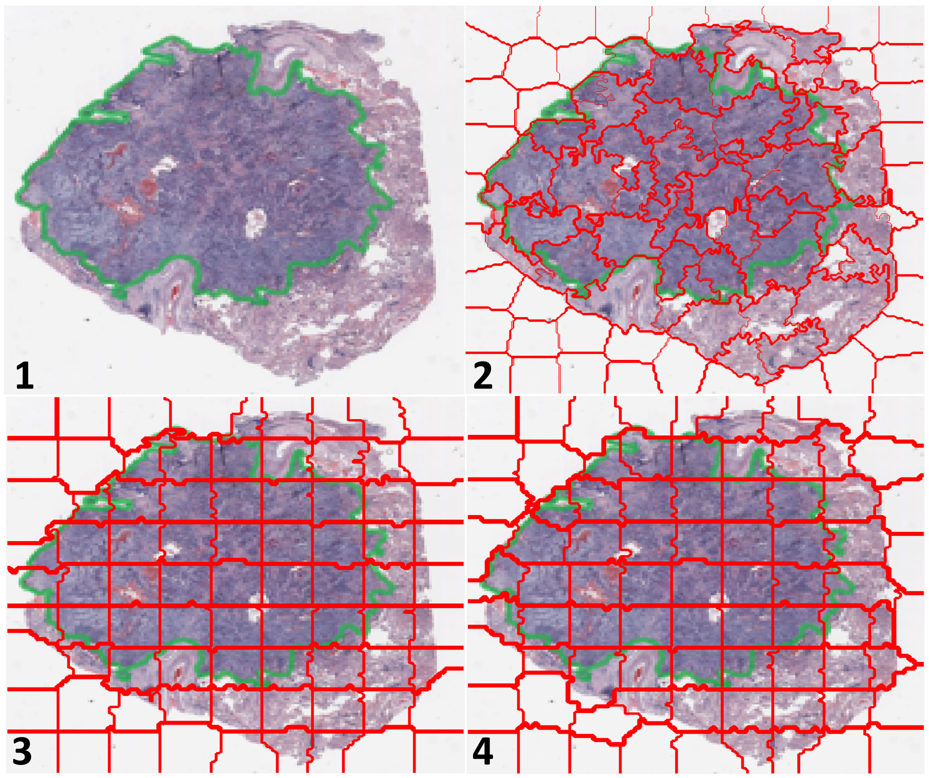


Figure 4.6: The comparison of several superpixel clustering on lung cancer H&E stained WSI.

reducing image primitives and complexity. If superpixels were well constructed, the downstream will not be affected by the sparse representation of image. The tumor cells of lung cancer patients (not only for lung cancer, but also generally appear in other subtypes of cancer) infest as cell mass. If treat the regions where tumor cell mass appears as ROIs, we can easily see that the H&E stained histopathology images that those tumor cells are more deeply colored due to the massive reproduction of genetic materials inside tumor nuclei (See Figure 4.6).

4.4.2 Experimental Setup

In the experimental stage, a random forest and a support-vector-machine (a.k.a. SVM) classifier were trained with local features extracted from regions defined by the superpixels given by Algorithm 1. The total 384 dimensional features include local binary patterns and statistics derived from the histogram of the three-channel HSD color model as well as common texture features, e.g. color SIFT. The introduced method was compared against the superpixels generated by SLIC [76] and tetragonum (i.e. rectangular patches). The experiments used the adenocarcinoma and squamous cell carcinoma lung cancer WSIs from the NLST (National Lung Screening Trial) Data Portal1 [1]. In superpixel classification, we executed feature extraction on the sampled patches (100×100) with 10% overlap with each other within each superpixel, we rule out patches sit across boundary avoiding noise. Lastly, we averaged the feature vectors of patches as representation of superpixel. When deciding ROI belonging for superpixel, if any part of ground-truth ROI fall into a superpixel, it will count as positive. The setup is rooted at the extremely high recall requirement for medical diagnosis. Given this setup, for better detection precision, superpixels should be better boundary adherent and clearly separated from background.

4.4.3 Experimental Results

Classifier	Metric	MROID	SLIC [76]	Tetragonum
Random Forest	Error Rate	**0.1326**	0.1933	0.2047
	Precision	**0.7127**	0.6835	0.6740
	Recall	**0.7333**	0.6108	0.6450
SVM	Error Rate	**0.3011**	0.3343	0.3061
	Precision	**0.6754**	0.6672	0.6723
	Recall	**0.7450**	0.6604	0.6972

Table 4.1: Precision and recall of RoI detection from the MROID, SLIC and tetragonum (non-superpixel).

[1]https://biometry.nci.nih.gov/cdas/studies/nlst/

Due to the overwhelming fidelity of superpixels given by our algorithm, the classifier operated over the regions segmented by superpixels is able to deliver better classification results (See Table 4.1). Since the feature descriptors were built on the patches segmented by contours of superpixels, the better the superpixel adhere to the boundaries, the better differentiability the features have for superpixel classification.

Figure 4.5 demonstrated the introduced multi-resolution coarse-to-fine superpixel segmentation in a lung cancer histopathology images. The algorithm first manipulated large block ($180\times$ 180) to cluster superpixels, then move to finer segmentation with 10×10 blocks on the superpixels selected by the classifier. The recursive refinement continues until the block queue run out, which means energy loss converges. In Table 4.1, we compared the classification recall and precision using superpixels given by SLIC and our method as well as simply patches without any preprocessing like superpixel clustering. Our results showed that, compared to simple patching, utilizing superpixel may not always increase ROI recall but definitely lift precision. Compared to superpixel given by SLIC with sophisticated boundary, out method outperformed on both recall and precision. We also observed that, if superpixels do not adhere to boundary, a detection based on classification of superpixels of low segmentation accuracy leads to worse accuracy than a trivial patch based method. While, our method delivered best results at both recall and precision.

4.5 Conclusion

In the chapter, we presented a novel local feature based solution to fast search and detection of regions of interest (ROI) in whole slide lung cancer histopathology image. For superpixel clustering, we introduced coarse-to-fine multi-resolution segmentation of superpixel by manipulating blocks of different size. Besides, boundary-only update strategy also reduced the computational complexity to the scale of superpixel boundary length, irrelevant of image size.

We creatively embedded the ROI classification into superpixel clustering algorithm. Iteratively executing superpixel construction and ROI detection. A better superpixel will accelerate detection and lift accuracy, while on the other hand, a better classification of ROI on coarse superpixel guides superpixel segmentation at finer level. Our algorithm performed a faster and finer ROI detection and segmentation. The effectiveness and efficiency of our algorithm has been verified on large histopathology WSI database, e.g. NLST.

In future, as the development of neural network capable of flexible input size [1, 91], it is likely to merge superpixel construction and downstream tasks, e.g. semantic segmentation, classification together in neural network architecture, in which superpixels are clustered using hidden features, while superpixels boost feature learning as well.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In the thesis, we reviewed and investigated the current state-of-the-arts GCNs and their applications on biomedical data. From the study, we summarized several technical challenges that prevent large-scale application of GCNs on data of dynamic and sophisticated graph structures. Especially we unveiled the constrained representational capability of convolutional kernel built with fixed graph topology. And from extensive experiments, we demonstrated the consequence of spectral kernel on fixed graph to performance of network on downstream tasks. Therefore, we proposed a the spectral graph convolution on learned Laplacian (SGC-LL) layer that work with both the original graph and the learned graph. Besides, we enhanced the feature extraction of GCN layer by reconstructing the cross-channel dependency, which classic CNNs always have. More importantly, the learned graph, also named as residual graph, were trained together with the rest of network, therefore the proposed AGCN model, equipped with stacked SGC-LL layers, combined graph representation learning and the searching for graph topology. Because we argued that the two processes are strongly correlated and we could not learn good graph representation using a mistakenly initialized graph. And our experiments showed the graph learning were significantly affecting the convergence of regression network: when the graph topological structure that the model attempted to learn became stable, the rest part of AGCN network would turn to convergence quickly. This finding makes us believe the AGCN model is good at discovering hidden substructures on graph that not included on original graph. To avoid introducing too many training parameters, we transfer the problem of searching optimal graph structure to the one of finding the optimal distance metric in which node-wise l-2 similarity is measured. By doing this, we eventually add a graph learning step of minimal training complexity, independent from graph size N, before graph convolutional operator.

After introduction of GAT [20], the spatial-based convolution on graph is reformulated as the problem of learning the *node-to-node* attention coefficients within receptive field. As extension of AGCN for graph-wise representation learning, we want to learn the emphnode-to-graph attentions that illustrate the importance of node with respect to graph-wise predictive tasks. Therefore, we constructed attentional AGCN model with parallel attention network, consist of a stack of SGC-LL layers, generate node-wise attention values at the end. With those attentional coefficients learned, the graph gather operator becomes a weighted summation of node features. Besides the performance gain from attention network, the extra explanability offered by attentional value distribution over graph nodes is more interesting. We already realized the regions of high attentional values coincide with tumor and lymphocyte cell cluster, which is plausibly related to patients' survival risk. Finally, to better utilize the power of attentional AGCN on large graph data and more importantly to extend the application of it to non-graph data, e.g. images and whole slides image (WSI), we built a end-to-end survival risk prediction pipeline, including patch sampling, graph construction, attentional AGCN and prediction. During this process, we realized the randomness introduced from patch sampling would largely affect overall accuracy of prediction. Detecting and localizing pathological region of interest (ROI) over WSI may mitigate this problem. To reduce computational complexity, we introduced a two-stage superpixel-based ROI detection approach. To efficiently construct superpixels with fine details preserved, we utilized a novel superpixel clustering algorithm which cluster blocks of pixel in a hierarchical fashion.

As many following research after [1] pointed out spectral-based GCNs have its own drawbacks on parallelizing the calculation of convolution into different work nodes. While spatial-based GCNs are easily to be decomposed into several independent convolutions over local cluster of nodes using shared kernel. This technique limitation makes spectral -based methods less attractive for large graph, e.g. social connection graph. However, it is not infeasible for spectral-based convolution to be distributed among computer nodes, the major concern is about the efficiency and effectiveness after decomposing the kernel together with graph. Besides, the graph learning and graph representation learning are able to be further decoupled as two independent but competing process. Borrowing the idea from Generative Adversarial Network (GAN), the generator network is to propose new graph structures, the discriminator is to learn the best graph representations on top and therefore boost the performance of regression tasks. Similar to self-supervised learning, graph learning is naturally a self-supervised task, if no annotation given. Those are very interesting research direction for follower to consider and investigate.

REFERENCES

[1] R. Li and S. Wang, “Adaptive graph convolutional neural networks,” in *AAAI Conference on Artificial Intelligence*, 2018.

[2] R. Li, J. Yao, X. Zhu, Y. Li, and J. Huang, “Graph cnn for survival analysis on whole slide pathological images,” in *International Conference on Medical Image Computing and Computer-Assisted Intervention.* Springer, 2018, pp. 174–182.

[3] J. Huang and R. Li, “Fast regions-of-interest detection in whole slide histopathology images,” in *Histopathology and Liquid Biopsy.* IntechOpen, 2020.

[4] J. Yao, M. Boben, S. Fidler, and R. Urtasun, “Real-time coarse-to-fine topologically preserving segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2947–2955.

[5] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning.* MIT press Cambridge, 2016, vol. 1, no. 2.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[7] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, “Large-scale video classification with convolutional neural networks,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[8] J. B. Estrach, W. Zaremba, A. Szlam, and Y. LeCun, “Spectral networks and deep locally connected networks on graphs,” in *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

[9] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163*, 2015.

[10] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, 2016, pp. 3844–3852.

[11] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks."

[12] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE signal processing magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[13] R. Shapovalov, E. Velizhev, and O. Barinova, "Nonassociative markov networks for 3d point cloud classification. the," in *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XXXVIII, Part 3A*. Citeseer, 2010.

[14] Y. Zhang and M. Rabbat, "A graph-cnn for 3d point cloud classification," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 6279–6283.

[15] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," *arXiv preprint arXiv:1801.07455*, 2018.

[16] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs: Graph fourier transform," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 6167–6170.

[17] F. Shoeleh and M. Asadpour, "Graph based skill acquisition and transfer learning for continuous reinforcement learning domains," *Pattern Recognition Letters*, vol. 87, pp. 104–116, 2017.

[18] J. Du, S. Zhang, G. Wu, J. M. Moura, and S. Kar, "Topology adaptive graph convolutional networks," *arXiv preprint arXiv:1710.10370*, 2017.

[19] R. De Maesschalck, D. Jouan-Rimbaud, and D. Massart, "The mahalanobis distance, chemometrics and intelligent laboratory systems," 2000.

[20] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[21] A. Kazi, L. Cosmo, N. Navab, and M. Bronstein, "Differentiable graph module (dgm) graph convolutional networks," *arXiv preprint arXiv:2002.04999*, 2020.

[22] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in neural information processing systems*, 2015, pp. 2224–2232.

[23] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Advances in neural information processing systems*, 2016, pp. 1993–2001.

[24] C. Zhuang and Q. Ma, "Dual graph convolutional networks for graph-based semi-supervised classification," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 499–508.

[25] H. Gao, Z. Wang, and S. Ji, "Large-scale learnable graph convolutional networks," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1416–1424.

[26] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International conference on machine learning*, 2016, pp. 2014–2023.

[27] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns,"

in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5115–5124.

[28] K. Liu, X. Sun, L. Jia, J. Ma, H. Xing, J. Wu, H. Gao, Y. Sun, F. Boulnois, and J. Fan, "Chemi-net: a molecular graph convolutional network for accurate drug property prediction," *International journal of molecular sciences*, vol. 20, no. 14, p. 3389, 2019.

[29] P. C. Rathi, R. F. Ludlow, and M. L. Verdonk, "Practical high-quality electrostatic potential surfaces for drug discovery using a graph-convolutional deep neural network," *Journal of Medicinal Chemistry*, 2019.

[30] S. Ishida, K. Terayama, R. Kojima, K. Takasu, and Y. Okuno, "Prediction and interpretable visualization of retrosynthetic reactions using graph convolutional networks," *Journal of Chemical Information and Modeling*, vol. 59, no. 12, pp. 5026–5033, 2019.

[31] K. Do, T. Tran, and S. Venkatesh, "Graph transformation policy network for chemical reaction prediction," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 750–760.

[32] 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*, 2016, pp. 770–778.

[33] F. Wang and J. Sun, "Survey on distance metric learning and dimensionality reduction in data mining," *Data mining and knowledge discovery*, vol. 29, no. 2, pp. 534–564, 2015.

[34] J. Gomes, B. Ramsundar, E. N. Feinberg, and V. S. Pande, "Atomic convolutional networks for predicting protein-ligand binding affinity," *arXiv*, pp. arXiv–1703, 2017.

[35] R. Gadde, V. Jampani, M. Kiefel, D. Kappler, and P. V. Gehler, "Superpixel convolutional networks using bilateral inceptions," in *European Conference on Computer Vision*. Springer, 2016, pp. 597–613.

[36] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur, "Protein interface prediction using graph convolutional networks," in *Advances in neural information processing systems*, 2017, pp. 6530–6539.

[37] Z. Wang, L. Zheng, Y. Li, and S. Wang, "Linkage based face clustering via graph convolution network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1117–1125.

[38] S. Abu-El-Haija, A. Kapoor, B. Perozzi, and J. Lee, "N-gcn: Multi-scale graph convolution for semi-supervised node classification," in *Uncertainty in Artificial Intelligence*. PMLR, 2020, pp. 841–851.

[39] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *International Conference on Learning Representations*, 2018.

[40] G. Landrum *et al.*, "Rdkit: Open-source cheminformatics," 2006.

[41] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande, "Moleculenet: a benchmark for molecular machine learning," *Chemical science*, vol. 9, no. 2, pp. 513–530, 2018.

[42] H. Altae-Tran, B. Ramsundar, A. S. Pappu, and V. Pande, "Low data drug discovery with one-shot learning," *ACS central science*, vol. 3, no. 4, pp. 283–293, 2017.

[43] K. M. Gayvert, N. S. Madhukar, and O. Elemento, "A data-driven approach to predicting successes and failures of clinical trials," *Cell chemical biology*, vol. 23, no. 10, pp. 1294–1301, 2016.

[44] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[45] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec, "Strategies for pre-training graph neural networks," *arXiv preprint arXiv:1905.12265*, 2019.

[46] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[47] D. R. Cox, "Regression models and life-tables," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 34, no. 2, pp. 187–202, 1972.

[48] J. L. Katzman, U. Shaham, A. Cloninger, J. Bates, T. Jiang, and Y. Kluger, "Deep survival: A deep cox proportional hazards network," *stat*, vol. 1050, no. 2, 2016.

[49] V. S. Dave, M. Al Hasan, B. Zhang, and C. K. Reddy, "Predicting interval time for reciprocal link creation using survival analysis," *Social Network Analysis and Mining*, vol. 8, no. 1, p. 16, 2018.

[50] R. Li, Y. Li, R. Fang, S. Zhang, H. Pan, and J. Huang, "Fast preconditioning for accelerated multi-contrast mri reconstruction," in *International Conference on Medical Image Computing and Computer-Assisted Intervention.* Springer, 2015, pp. 700–707.

[51] ——, "Fast preconditioning for accelerated multi-contrast mri reconstruction," in *International Conference on Medical Image Computing and Computer-Assisted Intervention.* Springer, 2015, pp. 700–707.

[52] X. Zhu, J. Yao, F. Zhu, and J. Huang, "Wsisa: Making survival prediction from whole slide histopathological images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7234–7242.

[53] C. Gunduz, B. Yener, and S. H. Gultekin, "The cell graphs of cancer," *Bioinformatics*, vol. 20, no. suppl_1, pp. i145–i151, 2004.

[54] A. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A decomposable attention model for natural language inference," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 2249–2255.

[55] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," in *International Conference on Learning Representations*, 2018.

[56] R. G. Miller Jr, *Survival analysis.* John Wiley & Sons, 2011, vol. 66.

[57] C. Yan, J. Yao, R. Li, Z. Xu, and J. Huang, "Weakly supervised deep learning for thoracic disease classification and localization on chest x-rays," in *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, 2018, pp. 103–110.

[58] R. Li and J. Huang, "Fast regions-of-interest detection in whole slide histopathology images," in *International Workshop on Patch-based Techniques in Medical Imaging.* Springer, 2015, pp. 120–127.

[59] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition.* Ieee, 2009, pp. 248–255.

[60] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[61] C. Kandoth, M. D. McLellan, F. Vandin, K. Ye, B. Niu, C. Lu, M. Xie, Q. Zhang, J. F. McMichael, M. A. Wyczalkowski, *et al.*, "Mutational landscape and signif-

icance across 12 major cancer types," *Nature*, vol. 502, no. 7471, pp. 333–353, 2013.

[62] B. S. Kramer, C. D. Berg, D. R. Aberle, and P. C. Prorok, "Lung cancer screening with low-dose helical ct: results from the national lung screening trial (nlst)," 2011.

[63] A. Mayr and M. Schmid, "Boosting the concordance index for survival data–a unified framework to derive and evaluate biomarker combinations," *PLOS ONE*, vol. 9, no. 1, pp. 1–10, 2014.

[64] Y. Li, J. Wang, J. Ye, and C. K. Reddy, "A multi-task learning formulation for survival analysis," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1715–1724.

[65] M. R. Lamprecht, D. M. Sabatini, and A. E. Carpenter, "Cellprofiler™: free, versatile software for automated biological image analysis," *Biotechniques*, vol. 42, no. 1, pp. 71–75, 2007.

[66] Y. Yang and H. Zou, "A cocktail algorithm for solving the elastic net penalized cox's regression in high dimensions," *Statistics and its Interface*, vol. 6, no. 2, pp. 167–173, 2013.

[67] J. D. Kalbfleisch and R. L. Prentice, *The statistical analysis of failure time data.* John Wiley & Sons, 2011, vol. 360.

[68] L. Hou, D. Samaras, T. M. Kurc, Y. Gao, J. E. Davis, and J. H. Saltz, "Patch-based convolutional neural network for whole slide tissue image classification," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2016, pp. 2424–2433.

[69] P. Bándi, R. van de Loo, M. Intezar, D. Geijs, F. Ciompi, B. van Ginneken, J. van der Laak, and G. Litjens, "Comparison of different methods for tissue segmentation in histopathological whole-slide images," in *2017 IEEE 14th In-*

ternational Symposium on Biomedical Imaging (ISBI 2017). IEEE, 2017, pp. 591–595.

[70] P. Takács and A. Manno-Kovacs, "Mri brain tumor segmentation combining saliency and convolutional network features," in *2018 International Conference on Content-Based Multimedia Indexing (CBMI)*. IEEE, 2018, pp. 1–6.

[71] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," 2010.

[72] A. Krizhevsky and G. Hinton, "Convolutional deep belief networks on cifar-10," *Unpublished manuscript*, vol. 40, no. 7, pp. 1–9, 2010.

[73] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "Unet++: A nested u-net architecture for medical image segmentation," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*. Springer, 2018, pp. 3–11.

[74] J. Yao, X. Zhu, J. Jonnagaddala, N. Hawkins, and J. Huang, "Whole slide images based cancer survival prediction using attention guided deep multiple instance learning networks," *Medical Image Analysis*, vol. 65, p. 101789, 2020.

[75] L. Chan, M. S. Hosseini, C. Rowsell, K. N. Plataniotis, and S. Damaskinos, "Histosegnet: Semantic segmentation of histological tissue type in whole slide images," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 10 662–10 671.

[76] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[77] M. Van den Bergh, X. Boix, G. Roig, B. de Capitani, and L. Van Gool, "Seeds: Superpixels extracted via energy-driven sampling," in *European conference on computer vision*. Springer, 2012, pp. 13–26.

[78] Z. Li and J. Chen, “Superpixel segmentation using linear spectral clustering,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1356–1363.

[79] X. Ren and J. Malik, “Learning a classification model for segmentation,” in *null*. IEEE, 2003, p. 10.

[80] M.-Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa, “Entropy rate superpixel segmentation,” in *CVPR 2011*. IEEE, 2011, pp. 2097–2104.

[81] O. Veksler, Y. Boykov, and P. Mehrani, “Superpixels and supervoxels in an energy optimization framework,” in *European conference on Computer vision*. Springer, 2010, pp. 211–224.

[82] B. E. Bejnordi, G. Litjens, M. Hermsen, N. Karssemeijer, and J. A. van der Laak, “A multi-scale superpixel classification approach to the detection of regions of interest in whole slide histopathology images,” in *Medical Imaging 2015: Digital Pathology*, vol. 9420. International Society for Optics and Photonics, 2015, p. 94200H.

[83] G. Litjens, B. E. Bejnordi, N. Timofeeva, G. Swadi, I. Kovacs, C. Hulsbergen-van de Kaa, and J. van der Laak, “Automated detection of prostate cancer in digitized whole-slide images of h and e-stained biopsy specimens,” in *Medical Imaging 2015: Digital Pathology*, vol. 9420. International Society for Optics and Photonics, 2015, p. 94200B.

[84] A. Lucchi, K. Smith, R. Achanta, V. Lepetit, and P. Fua, “A fully automated approach to segmentation of irregularly shaped cellular structures in em images,” in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2010, pp. 463–471.

[85] M. Van den Bergh, G. Roig, X. Boix, S. Manen, and L. Van Gool, "Online video seeds for temporal window objectness," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 377–384.

[86] K. Yamaguchi, D. McAllester, and R. Urtasun, "Efficient joint segmentation, occlusion labeling, stereo and flow estimation," in *European Conference on Computer Vision*. Springer, 2014, pp. 756–771.

[87] S. Kwak, S. Hong, B. Han, *et al.*, "Weakly supervised semantic segmentation using superpixel pooling network." in *AAAI*, vol. 1, 2017, p. 2.

[88] W.-C. Tu, M.-Y. Liu, V. Jampani, D. Sun, S.-Y. Chien, M.-H. Yang, and J. Kautz, "Learning superpixels with segmentation-aware affinity loss," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 568–576.

[89] X. Liu, S. Guo, B. Yang, S. Ma, H. Zhang, J. Li, C. Sun, L. Jin, X. Li, Q. Yang, *et al.*, "Automatic organ segmentation for ct scans based on super-pixel and convolutional neural networks," *Journal of digital imaging*, vol. 31, no. 5, pp. 748–760, 2018.

[90] V. Jampani, D. Sun, M.-Y. Liu, M.-H. Yang, and J. Kautz, "Superpixel sampling networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 352–368.

[91] F. Yang, Q. Sun, H. Jin, and Z. Zhou, "Superpixel segmentation with fully convolutional networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 964–13 973.

BIOGRAPHICAL STATEMENT

Ruoyu Li received his B.E. degree from Northwestern Polytechnical University, China, in 2010, and the M.S. degree from Syracuse University, NY, in 2012. After that, he worked as research associate at INFINITUS center of excellence, Nanyang Technologoical University, Singapore from 2012 to 2014. Starting from 2014 Fall, he began to pursue PhD degree under supervision of Dr. Junzhou Huang at the University of Texas at Arlington. His research interests include medical image analysis, computer vision and deep graph learning. He has published several works in top academic conferences e.g. AAAI, MICCAI, ACM-BCB.