

USE OF SUBGRAPH MINING IN HISTOPATHOLOGY IMAGE CLASSIFICATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

By
Bayram Berdiyev
September 2022

USE OF SUBGRAPH MINING IN HISTOPATHOLOGY IMAGE
CLASSIFICATION

By Bayram Berdiyev

September 2022

We certify that we have read this thesis and that in our opinion it is fully adequate,
in scope and in quality, as a thesis for the degree of Master of Science.

Selim Aksoy(Advisor)

Hamdi Dibeklioglu

Ramazan Gökberk Cinbiş

Approved for the Graduate School of Engineering and Science:

Orhan Arıkan
Director of the Graduate School

ABSTRACT

USE OF SUBGRAPH MINING IN HISTOPATHOLOGY IMAGE CLASSIFICATION

Bayram Berdiyev
M.S. in Computer Engineering
Advisor: Selim Aksoy
September 2022

Breast cancer is the most common cancer in women and has a high mortality rate. Computer vision techniques can be used to help experts to analyze the breast cancer biopsy samples better. Graph neural networks (GNN) have been widely used to solve the classification of breast cancer images. Images in this field have varying sizes and GNNs can be applied to varying sized inputs. Graphs can store relations between the vertices of the graph and this is another reason why GNNs are preferred as a solution. We study the use of subgraph mining in classification of regions of interest (ROI) on breast histopathology images.

We represent ROI samples with graphs by using patches sampled on nuclei-rich regions as the vertices of the graph. Both micro and macro level information are essential when classifying histopathology images. The patches are used to model micro-level information. We apply subgraph mining to the resulting graphs to identify frequently occurring subgraphs. Each subgraph is composed of a small number of patches and their relations, which can be used to represent higher level information. We also extract ROI-level features by applying a sliding window mechanism with larger sized patches. The ROI-level features, subgraph features and a third representation obtained from graph convolutional networks are fused to model macro-level information about the ROIs. We also study embedding the subgraphs in the graph representation as additional vertices. The proposed models are evaluated on a challenging breast pathology dataset that includes four diagnostic categories from the full spectrum. The experiments show that embedding the subgraphs in the graph representation improves the classification accuracy and the fused feature representation performs better than the individual representations in an ablation study.

Keywords: Breast histopathology, region of interest classification, graph mining,

subgraph mining.



ÖZET

HİSTOPATOLOJİ GÖRÜNTÜ SINIFLANDIRMASINDA ALT ÇİZGE MADENCİLİĞİ

Bayram Berdiyev
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Danışmanı: Selim Aksoy
Eylül 2022

Meme kanseri kadınlarda en sık görülen kanserdir ve ölüm oranı yüksektir. Uzmanların meme kanseri biyopsi örneklerini daha iyi analiz etmesine yardımcı olmak için bilgisayarla görü teknikleri kullanılabilir. Çizge sinir ağı (ÇSA), meme kanseri görüntülerinin sınıflandırılmasını çözmek için yaygın olarak kullanılmaktadır. Bu alandaki görüntülerin boyutları değişkendir ve ÇSA'lar değişen boyuttaki girdilere uygulanabilir. Çizgeler, çizgenin düğümleri arasındaki ilişkileri saklayabilir ve bu, ÇSA'ların bir çözüm olarak tercih edilmesinin bir başka nedenidir. Bu tezde meme histopatolojisi görüntülerinde ilgi bölgelerinin (İB) sınıflandırılmasında alt çizge madenciliğinin kullanımını inceliyoruz.

Çizgenin düğümleri olarak çekirdek açısından zengin bölgelerde örneklenen pencereleri kullanarak İB örneklerini çizge ile temsil ediyoruz. Histopatoloji görüntülerini sınıflandırırken hem mikro hem de makro düzeyde bilgi gereklidir. Pencereler, mikro düzeydeki bilgileri modellemek için kullanılır. Sıklıkla ortaya çıkan alt çizgeleri belirlemek için elde edilen çizgelere alt çizge madenciliği uyguluyoruz. Her alt çizge, daha üst düzey bilgileri temsil etmek için kullanılacak az sayıda pencereden ve bunların ilişkilerinden oluşur. Ayrıca, daha büyük boyutlu pencereler içeren bir kayar pencere mekanizması uygulayarak İB düzeyindeki özellikleri çıkarıyoruz. İB düzeyindeki özellikler, alt çizge özellikleri ve çizge evrişimli ağlardan elde edilen üçüncü bir gösterim, İB'ler hakkında makro düzeydeki bilgileri modellemek için birleştirilir. Ayrıca alt çizgeleri ek düğüm olarak çizge temsiline yerleştirmeyi de çalışıyoruz. Önerilen modeller, tam spektrumdan dört tanı kategorisini içeren zorlu bir meme patolojisi veri setinde değerlendirilmiştir. Deneyler, alt çizgelerin görüntü gösterimine dahil edilmesinin sınıflandırma doğruluğunu geliştirdiğini ve birleştirilmiş özellik gösteriminin bir ablasyon çalışmasında bireysel gösterimlerden daha iyi performans gösterdiğini göstermektedir.

Anahtar sözcükler: Meme histopatolojisi, ilgi bölgesi sınıflandırması, çizge madenciliği, alt çizge madenciliği.



Acknowledgement

I would like to express my heartfelt thanks to my advisor Prof. Dr. Selim Aksoy for his guidance and support throughout my master's studies. I learned a lot and developed technical and non-technical skills during the master's studies which will help me in my future career. I also want to deeply thank to his patience while I tried to solve the problems I have faced.

I would also like to thank Assist. Prof. Dr. Hamdi Dibeklioglu and Assist. Prof. Dr. Ramazan Gökberk Cinbiş for accepting to be committee members of my thesis. Additionally, I want to thank their feedback on my thesis.

Finally, I would like to express my deepest gratitude, thanks and love to my mother, father and brothers. They have supported me throughout my life and during all of my studies. I would like to extend my thanks to all of my friends who supported me during the master's studies.

Contents

1	Introduction	1
1.1	Problem and our solution	2
1.2	Organization of the thesis	3
2	Related Work	5
2.1	Digital histopathology and GNN	6
2.1.1	Digital histopathology	6
2.1.2	GNN	6
2.1.3	Graph construction in digital histopathology	7
2.2	Subgraph mining	8
3	Methodology	10
3.1	Data set	12
3.2	Patch representation	13
3.2.1	Locating patches	13

3.2.2	Feature extraction	14
3.3	Constructing graph	16
3.4	Subgraph representation	16
3.4.1	Graph mining	17
3.4.2	Subgraph selection	21
3.4.3	Fisher vectors	22
3.5	Graph representation	23
3.6	Combining features	25
3.6.1	Fusion network	25
3.6.2	Embedding subgraphs into graph	27
4	Experiments and Results	31
4.1	Patch and magnification	31
4.2	Patch feature visualization	32
4.3	Patch clustering	33
4.4	Subgraph mining and selection	40
4.5	Combining features	41
4.5.1	Fusion network	44
4.5.2	Embedding subgraphs	47
4.6	Comparison	49

<i>CONTENTS</i>	x
4.6.1 Fusion network vs <i>combineEndGNN</i>	50
4.6.2 Our work vs two stage models	51
5 Conclusion	53



List of Figures

3.1	Fusion network diagram	11
3.2	ROI sample and chosen patches	15
3.3	Long edge removal after applying Voronoi	17
3.4	Illustration of different frequencies	20
3.5	Example for adding a new node from subgraph	28
3.6	Diagram that describes the model <i>combineEnd</i>	30
4.1	t-SNE plots for patches when 5X magnification is used	34
4.2	t-SNE plots for patches when 10X magnification is used	35
4.3	Entropy plots for 5X and 10X magnifications	37
4.4	Sample patches for each cluster when 5X magnification used . . .	38
4.5	Sample patches for each cluster when 10X magnification used . .	39
4.6	Heat map plots of the selected frequent subgraphs for training and validation sets for 10X magnification	42
4.7	All selected frequent subgraphs for 10X magnification	43

List of Tables

3.1	Statistics for number of pixels for each class label	12
3.2	Number of slides and ROIs for each set and class label	13
4.1	Subgraph mining recall scores for the simple classifier mentioned in 3.4.2 with validation set	41
4.2	G-mean, precision, recall and f-measure scores for the fusion net- work for different input sources	46
4.3	G-mean, precision, recall and f-measure scores for models used for embedding subgraph experiments	48
4.4	Confusion matrix for the fusion network when <i>roi-feature</i> , <i>subgraph-feature</i> and <i>graph-feature</i> used to train	49
4.5	Confusion matrix for the model <i>combineEndGNN</i> when trained with <i>new-node-graph</i>	50
4.6	G-mean, precision, recall and f-measure scores for the models from Aygunes's work [1]	52

Chapter 1

Introduction

Breast cancer can lead to a death and it is the most common cancer in women. The cancer can be handled with less damage if it is discovered earlier [2]. Classification of cancer can be used to decrease time consumed to diagnose the samples. For instance, it can be used to eliminate insignificant regions from the slides obtained from the biopsy samples. The samples contain a lot of information and it is harder for humans to process all of the information. To solve this classification task most of the recent works use Neural Networks (NNs) because of its success on image recognition. Specifically, Graph Neural Networks (GNNs) have been widely used to use its power on learning relations between nodes [3].

This thesis introduces use of subgraph mining in classification of breast cancer biopsy samples. Both micro and macro-level information are important features used to classify the biopsy samples. We construct graphs from the samples and nodes of those graphs can represent micro-level information. We use subgraph mining output to represent higher level information about the samples. Additionally, we apply a sliding window and use it as a more higher level information. Fusion network, a model defined to have ablation study, uses all three input sources. Also, we try to embed subgraph features into graph and see that it leads to improved results.

1.1 Problem and our solution

In breast histopathology, biopsy samples are scanned with magnification to get Whole Slide Images (WSI). The experts decide diagnostic labels after investigating WSIs of patients. There are Regions of Interests (ROIs) in WSIs which are informative regions used by the experts. The same diagnostic labels can be used to label ROIs. WSI can contain ROIs from different labels and ROI-level labels play an important role while deciding WSI labels, so the step of classifying ROIs is a necessary step in classification of WSIs. In this thesis, we try to solve classification of ROIs with 4 classes. There are no objective standards to decide the label of the given sample but we know how the experts evaluate it. First they look at the whole image and zoom the parts which contain important information. In other words, both macro-level and micro-level features are used to decide the label of the given sample [4].

Distribution of nucleus is one of the important pieces of information while deciding labels, so we use this information to decide nodes of the graph [5]. First, a probability map for nuclei in each pixel is obtained with a nucleus detector. Then, we put patches on nucleus rich regions of the sample where each patch will be a node of the graph. There are also nucleus free regions which may be an important feature. After locating patches on nucleus rich regions, we locate new patches on nucleus free regions.

Each patch is a window which is a square region that contains a small part of the ROI sample. Using raw RGB data is not a good feature so we have used feature extractor. There are works that use pretrained model as a feature extractor, as well as some of the works train the model with their data set. We use pretrained Resnet50 [6] as a feature extractor and we use it in more than one place. Using pretrained network makes our method easily applicable to a new data set, because training a Deep Neural Network (DNN) is not a straightforward task. To exploit subgraph mining, we want to give labels for each node and in our data set there is only ROI-level label. To get those labels, we use patch features with Kmeans clustering algorithm. As a result, similar patches get the

same label.

Since neural networks are black-box like models, it is better to give additional features to make the learning process easier. Also, features from different levels can help to classify breast histopathology biopsy samples. Because of that we are using features from three different levels with a fusion network that we have defined to do an ablation study. First, the graphs are constructed by using the ROI samples and this input contains features from the lowest level. Second, subgraph mining output that represents a middle level feature. The highest level feature is obtained by applying a sliding window with a higher sized window. In addition to the ablation study, we try to integrate subgraphs into graph. We define some models that process graphs that are integrated with subgraphs. Experiments show that integrating makes the learning process easier.

We have applied frequent subgraph mining after getting graphs from each ROI sample. Then, we found occurrences of frequent subgraphs on graphs which give us subgraph level features. This feature is used alongside graphs while applying GNN, which is our main contribution in this work. First we have tried to use subgraph level features with graph features by concatenating them together. We see an increase in performance when using multiple sources compared to using features from one source. Then, we try to embed the subgraph mining output into a graph. To do that, we add a new node which is connected to subgraph nodes. There are models that process subgraph embedded graphs and perform better than a model that uses them by concatenating.

1.2 Organization of the thesis

The rest of this thesis includes the following chapters.

Chapter 2 describes related works in classification of breast histopathology and works that use graph mining.

Methodology is mentioned in Chapter 3. Data set that we have used is explained in the first section. How samples are collected, difficulties of data set and splitting of the data set into train, validation and test sets are described. Second, representation of the patches i.e. how the patches are located and the process of extracting features from the patches are presented. Then, construction of the graph from patches is explained. After that, representation of frequent sub-graphs with Fisher vectors is demonstrated. Finally, step of getting ROI labels from graphs is explained.

Experiments and discussions of results are in Chapter 4. First, we plotted the patches with t-Distributed Stochastic Neighbor Embedding (t-SNE) [7] for different magnifications, to check if there is significant difference. It is followed by clustering experiments. After that, the effects of graph mining parameters are shown. Then, ablation study on different feature sources is demonstrated and experiments on embedding subgraph mining output into the graph is illustrated with different metrics. Finally, a comparison of our work to related work is presented.

Conclusion and future works are explained in Chapter 5.

Chapter 2

Related Work

In this chapter we are presenting works done related to our work. Success of NNs on image recognition tasks leads to use of GNNs in digital histopathology. Convolutional Neural Networks (CNN) performed well on learning representation from images, but they are not applicable to arbitrary sized images without additional operation. In digital histopathology, input type is also image but with varying sizes. GNN can be applied to graphs with different sizes and this is one of the reasons why GNN is preferred to CNN in digital histopathology tasks. Graphs can encode relations between its nodes and this is another reason why GNNs are preferred [3]. We are also using GNN to solve the classification of cancer on breast histopathology images. Alongside GNN, we have used subgraph mining to exploit the graph representation. In Section 2.1, we present works related to digital pathology and GNN. Works related to subgraph mining are presented in Section 2.2.

2.1 Digital histopathology and GNN

2.1.1 Digital histopathology

In recent years, deep learning helped to develop breast histopathology image analysis [8]. Also, traditional methods such as support vector machines and superpixel clustering are still in use to solve breast histopathology tasks [9]. Breast cancer analysis is one of the applications of digital histopathology, but other cancer analysis can be done in a similar way [1, 10, 11, 12], so we mention works done in digital pathology.

There are different works which try to reduce the effect of lacking task specific labels. Expert knowledge is necessary to label the samples, which leads to scarce of labeled data sets. In our work, we use an unsupervised method and pretrained model to get patch labels, since there is no patch label in our data set. In work [13], they have applied transfer learning from a weakly labeled data set and achieved competitive patch classification results while using less data. Also, transfer learning leads to an increase in classification accuracy on WSIs [14]. Additional works that use transfer learning for classification and segmentation tasks can be found in [15, 16].

2.1.2 GNN

There are several types of GNN such as recurrent GNNs, convolutional GNNs, graph autoencoders and spatial-temporal GNNs [17]. GNNs are used in digital pathology as well as they are used in other fields. For instance, spatial-temporal and recurrent GNNs are used for forecasting tasks [18, 19]. In digital pathology, convolutional GNNs are mainly used GNN type [3].

Most of the convolutional GNN models require node features alongside graph structure. In work [1], they have trained CNN with their data set and used it as a feature extractor. Instead of training model, pretrained CNN models are used

in works [20, 21]. Resnet50, DenseNet [22] and MobileNetV2 [23] are some of the preferred models to extract features in digital pathology. [3].

Graph Convolutional Networks (GCNs) [24] and Graph Attention Networks (GATs) [25] are example convolutional GNNs. Layers in these models change the feature dimension of nodes while graph structure remains unchanged. Node features and neighbor node features are combined to calculate a new feature in GCN layers. Layers in GAT are constructed by adding self-attention mechanisms to the layers in GCN. There are also graph pooling layers that change graph structure, where the number of vertices are decreased. Self-Attention Graph Pooling [26] and Differentiable Graph Pooling [27] are examples for this layer type.

2.1.3 Graph construction in digital histopathology

There are mainly 4 types of graphs constructed in digital histopathology, which are cell-graph, patch-graph, tissue-graph and hierarchical-graph [3]. Vertices of cell-graphs are defined by using nucleus information. Nucleus detector is used to extract nucleus information from the samples and there are studies using pretrained nucleus detectors [28] as well as some works train the model [29]. Patch-graphs are composed of patches which are located uniformly or heuristically [1, 12]. Patches are windows on the input images which may contain many nuclei. Based on the size of the patch it may contain part of the tissue or whole tissue. Some works [30, 31] try to exploit tissue information and used tissues as vertices to the constructed graph which are example to tissue-graph. To use both micro and higher level information hierarchical-graphs are used. For instance, in work [21], they have constructed graph by both cell and tissue nodes where cell nodes and tissue nodes are connected by an edge. To add the edges between the nodes Delauney triangulation [30], distance threshold [1] or Voronoi diagram [32] methods are used.

2.2 Subgraph mining

Subgraph mining is a subsection of graph mining which is used in biology, chemistry, social network, knowledge graph and collaboration network [33]. For example in chemistry, chemical compounds are represented as a graph and subgraph mining is applied to find structural fragments [34]. In a collaboration network, one can mine important collaborations among authors [35]. We have applied subgraph mining and used it alongside graphs. While mining, matching is used and there are two types of matching: exact and approximate. We have used exact matching, since approximate matching is preferred if exact matching is not applicable. In exact matching algorithms, there are pattern-growth approach methods where small subgraphs are extended by adding edges to get bigger frequent subgraphs. Example algorithms for this approach are Molecular Fragments Identification Technique (MoFa) [36], Spanning Tree based Maximal Graph Mining (SPIN) [37] and Graph-Based Substructure Pattern Mining (gSpan) [38]. Another approach is apriori-based where new frequent subgraphs are found by joining two small frequent subgraphs. Edge disjoint path-join algorithm [39] is an example algorithm for this approach. There are also methods which find some of the frequent subgraphs but do not guarantee to find all of the frequent subgraphs which are called incomplete methods. We can see this kind of algorithm in work [40].

The algorithms described above work well on mining subgraphs from a database of graphs. There are also algorithms adapted to mining subgraphs from a single large graph. These types of methods are used in applications of social networks, citation graphs, or protein-protein interactions bioinformatics [41]. In our case, we have a graph for each sample, so we did not consider these types of methods. We have used gSpan as our method because it is widely used in subgraph mining [42].

The metric to count subgraphs in a graph is called support. The widely used metrics are maximum independent sets (MIS) [43], harmful overlap (HO) [44] and minimum image based (MNI) [45]. The metrics differ on how isomorphisms,

which is mapping subgraph to graph, overlap. We have used MNI as a support metric because the computation of MIS and HO are NP-complete. Also MNI mappings contain all of the mappings of both MIS and HO [35].



Chapter 3

Methodology

In this chapter we explain our method to solve the classification of ROI in breast histopathology images. Data set is introduced in Section 3.1. Section 3.2 describes how we locate patches which are nodes of graph that is going to be constructed. Nuclei information is used as a heuristic for patch locations. Features of nodes are obtained from pretrained neural network where input is patch window cropped from ROI image. In Section 3.3 the step of constructing graph from nodes is explained. Voronoi diagram is used to add edges between nodes. There are some undesired edges and how those edges are removed also explained. After constructing graph, subgraph mining is applied which is in Section 3.4. There are alternative algorithms for graph mining. The algorithm that we use and its properties are discussed. Then, frequency of the subgraph, which shows how many times that subgraph occurred on graph is defined. After that, subgraph mining is applied on training set to get frequent subgraphs. Finally, most informative ones are selected by using forward selection. In Section 3.5, graph representation is discussed. Finally, in Section 3.6, we illustrate how subgraph features are used alongside other features to predict diagnostic label of the samples.

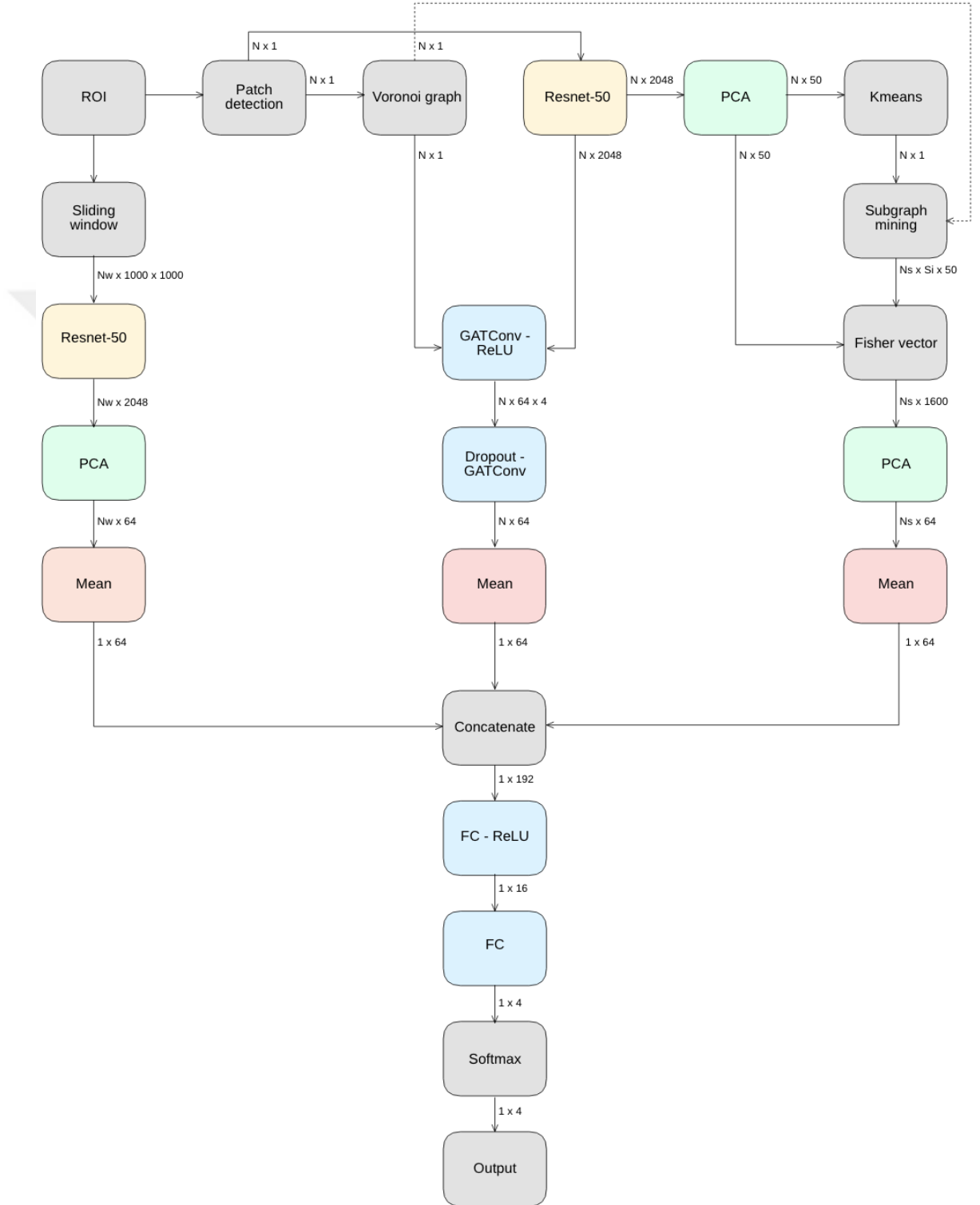


Figure 3.1: Fusion network diagram. The network gets input from three sources: *roi-feature*, *subgraph-feature* and *graph-feature*. Each feature is processed with specific steps and concatenated at the end. Then, two fully connected layers try to learn classification from the concatenated output.

3.1 Data set

The data set we have used is the same data set used in [1] work. It contains 121 WSIs from 99 patients and was collected with the Department of Pathology at Hacettepe University. Olympus slide scanner was used with 40X magnification to scan haematoxylin and eosin stained specimens. Then, ROIs were labeled with 4 class labels by experienced pathologists. ROI is represented as a window on WSI, its mask and class label. The classes are *benign*, *atypia*, *in situ* and *invasive*. We have used the same steps in [1] to get RGB image from hematoxylin and eosin channels which are stain normalization and getting red, green and blue values for each pixel. In Table 3.1 we can see statistics of the number of pixels on each class for samples at 10X magnification. In our experiments we have used 10X and 5X magnifications.

There is a data set which contains the same sized ROIs for all samples [46]. As we can see from Table 3.1, sizes of ROIs vary a lot in our data set. Pathologists themselves decide the size and shape of the samples so we can consider that ROIs have varying sizes. CNNs perform well on image classification tasks, but they require same sized inputs which makes our data set more difficult than same sized data sets.

Table 3.1: Statistics for number of pixels for each class label. Average, standard deviation and maximum to minimum ratio are given.

	Benign	Atypia	In Situ	Invasive
Average	1966K	473K	4227K	13528K
Standard deviation	4207K	687K	8325K	19263K
Max-min ratio	2216.0	704.5	1989.1	762.5

In [1], data set have been split into train, validation and test sets. They have decided to eliminate small ROIs from the data set. In our work, we are using the same train, validation and test sets. Also, we have removed the same small samples from our data set. In Table 3.2, we can see the number of samples for each set and label. It is worth to note that nearly half of the samples belong to

the train set and the other half is distributed to validation and test sets. While splitting the samples, all of the samples belong to one WSI are put in one set. Since WSIs can contain varying numbers of ROIs, it may not be possible to distribute samples perfectly even. That is why the validation set contains 291 samples while the test contains 278 samples.

As can be seen in Table 3.2, there is data imbalance in our data set. Class *atypia* contains 46 samples while class *benign* contains 238 samples for the train set. Difference in the number of samples is very huge, so we have upsampled the samples in the train set to reduce its effect on performance. After upsampling there are 238 samples for each class in the training set. We duplicated the samples without changing anything for the classes that have less number of samples.

Table 3.2: Number of slides and ROIs for each set and class label. Sum of the slides for labels may be greater than the total number of slides, because one slide can contain ROIs with different labels.

		Benign	Atypia	In Situ	Invasive	Total
Slide	Training Set	42	20	22	15	53
	Validation Set	18	11	10	7	23
	Test Set	20	10	11	7	26
	Total	80	41	43	29	102
ROI	Training Set	238	46	173	117	574
	Validation Set	104	26	103	58	291
	Test Set	123	22	77	56	278
	Total	465	94	353	231	1143

3.2 Patch representation

3.2.1 Locating patches

Distribution of the nucleus is one of the important information while deciding cancer labels of breast cancer patients [28]. There are works that use the nucleus as a node of a graph while constructing graphs from breast histopathology images.

Because of that we try to put patches on nucleus rich regions. We have used HoverNet [47] which is widely used in detecting nuclei in digital pathology. Input to the HoverNet is an ROI image and it gives a probability map of the nucleus on each pixel. We can see an example ROI image and output of the model in Figure 3.2b.

After getting probabilities, we try to put a patch on each nucleus. To do that, we need some discrete values for being a nucleus, 0 for no nucleus and 1 for having a nucleus. First, we apply a threshold to get a 0-1 map. Then, a connected components algorithm is applied where each pixel is a node and there is an edge if two pixels are adjacent. Ideally, each connected component should represent one nucleus in a given sample. To find the exact position for the center of the patch we apply centroid operation for the nodes belonging to the component. Mentioned steps give us patches and their locations.

Having one patch for each nucleus will result in overlapping patches. Many of those overlapping patches overlap most of its region with others, because one patch can contain more than a hundred nuclei. To eliminate those overlapping patches we use Non-maxima suppression (NMS). In order to use NMS, we need a value for each patch which shows how important it is. Those values are obtained from summing the nucleus probability of pixels inside the patch. We allow small overlapping between patches by giving IOU-threshold greater than zero. Example patches selected after applying NMS can be seen in Figure 3.2c.

3.2.2 Feature extraction

Before constructing a graph we need to decide how to get features of each patch which is a node of the graph. To do that, we use ImageNet pretrained Resnet50 [6]. First reason why we do not train the model is we do not have patch labels. Also, training DNNs is not a straightforward task, which makes our method easily applicable to a new data set. Our patches are RGB images and the size of patches is equal to input size of Resnet50, which is 224x224. Output of the last average pooling layer on Resnet50, which gives us a 2048 dimensional feature vector, is

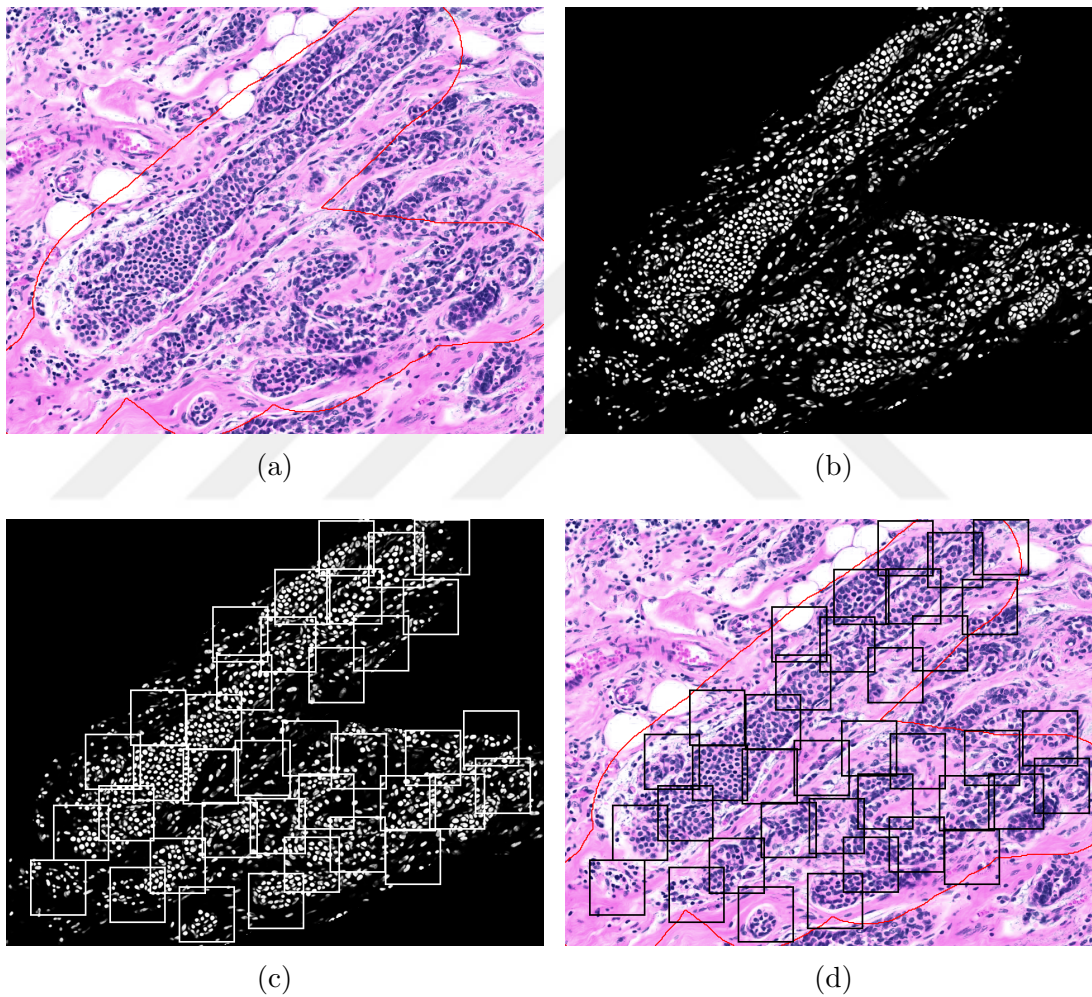


Figure 3.2: ROI sample and chosen patches. In (a), ROI image and boundary drawn by pathologists shown. Output of HoverNet, which is a probability map, is in (b). Followed by (c), which shows patches drawn on the probability map. In (d), the same patches are drawn on ROI image

used as a feature extractor.

We want to give discrete labels for nodes alongside 2048 dimensional vectors. Kmeans is used as a clustering algorithm to get discrete labels from feature vectors. Kmeans uses distance functions to compare two features. A 2048 dimensional vector is not an ideal input to distance functions, so we decrease dimension with PCA. Applying clustering on patches gives us cluster label for each node.

3.3 Constructing graph

To construct graphs we need edges alongside nodes. Voronoi diagram, Delaunay triangulations and distance threshold are used to add edges between nodes in digital pathology [1, 32, 30]. We use the Voronoi diagram for edges and we can see example edges in Figure 3.3a. The main reason why we add edges between patches is to get neighboring information about the patches. As we can see there are edges between nodes which are not close or far away from each other. We decided to remove those long edges and long edge means if it connects nodes whose distance is greater than two times average edge distance in that sample. The example removal can be seen in Figure 3.3b.

3.4 Subgraph representation

After constructing graphs we apply graph mining on constructed graphs. Graph mining is done by finding subgraphs which occurred in many of the graphs which are called frequent subgraphs. These frequent subgraphs are found in a supervised way, so we will get useful information to classify graphs. Additionally we experiment with unsupervised subgraph mining and results can be found in the next chapter. Subgraph mining returns many subgraphs, because of that, selection is done after finding subgraphs.

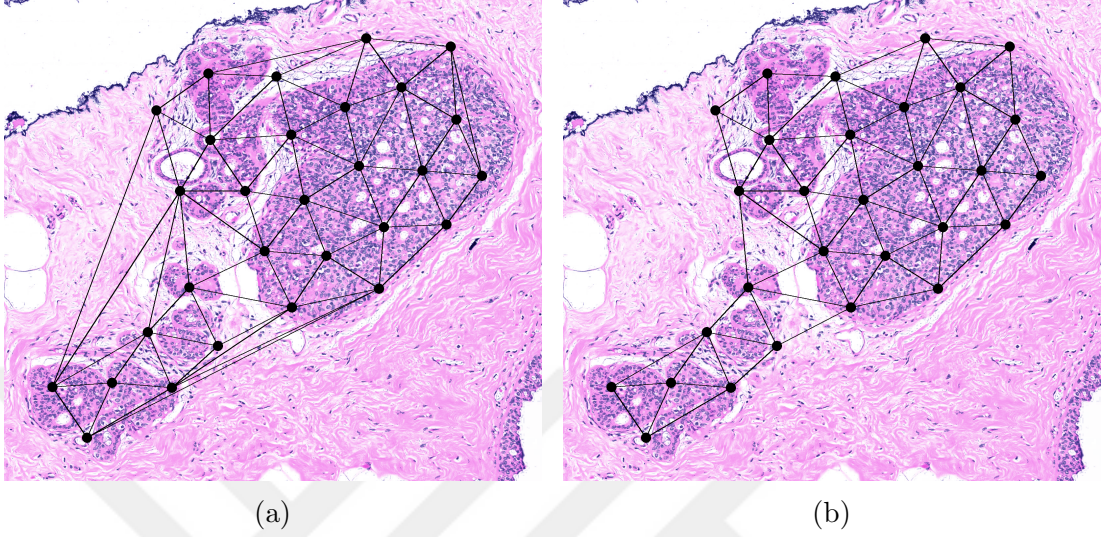


Figure 3.3: Long edge removal after applying Voronoi. In (a), we can see some edges connected even if they are far away from each other. Those long edges are removed and shown in (b).

3.4.1 Graph mining

3.4.1.1 gSpan

gSpan is a pattern or subgraph mining algorithm on a graph data set[38]. Input to the algorithm is $D = \{G_1, G_2, \dots\}$, *support*, *frequency* where D is a data set which contains labeled graphs which is a graph where each node and edge have a label. In literature, the term *support* is used as a metric. In this thesis, we are using it as follows. *Support* is a percentage threshold which decides whether subgraph is frequent if that subgraph occurred at least in $|D| * \text{support}$ of graphs. Subgraph is occurred in graph G_i if there is at least *frequency* mappings from nodes of graph G_i to subgraph. Definition of mapping will be explained in detail in Section 3.4.1.2, but we can think of it as finding nodes and edges in a graph which is equivalent to a given subgraph. Output of the algorithm is frequent subgraphs and their mappings in the graphs. D is combined from graphs that we constructed in Section 3.2. Our graphs do not have edge labels, we can consider them as having the same label for all edges.

The algorithm does not use a graph label, which is the ROI label in our case, so we decided to apply gSpan in a supervised way. First, train data is split into data sets in a way such that each data set contains graphs from only one class and all samples of one class are in one data set. This gives four data sets. Then, graph mining is applied on each data set, so frequent subgraphs give information about the label of the sample. Additionally, we apply subgraph mining in an unsupervised way. In this way, we have only one data set D which contains all of the samples from the training set. Subgraph mining gives us a lot of subgraphs, including similar subgraphs. To eliminate unnecessary subgraphs, selection is done and can be found in Section 3.4.2.

gSpan gives only the way to find frequent subgraphs. To count subgraphs in a given graph is done also by using the same gSpan algorithm. Assume we want to find mappings of subgraph s on a given graph G . To do that, gSpan is applied to a new data set $D = \{G, s\}$ with *support* = 1 and *frequency* = 1. After getting frequent subgraphs, all subgraphs which are not equal to s are removed. Finally, we only have one subgraph which is equal to s and its mappings in G . This can be applied to counting more than one subgraph. For example if we want to find the frequency of two subgraphs s_1 and s_2 on G , the data set will be $D = \{G, s_1, s_2\}$. Then, gSpan is applied with D , *support* = $\frac{2}{3}$ and *frequency* = 1 with the condition that one of the occurrences should be in G otherwise it is not frequent.

3.4.1.2 Subgraph frequency

We need a definition for mapping a subgraph to a graph to define the frequency of the subgraph. In our work there is only a label for vertices of the graph. In this work, graphs are undirected and defined as follows. $G = (V, E, L)$ where V is a set of vertices, E is a set of edges and L is a function which maps vertex to label.

Definition 1: A connected graph $S = (V_S, E_S, L_S)$ is subgraph of graph $G = (V_G, E_G, L_G)$ if and only if $V_S \subseteq V_G$, $E_S \subseteq E_G$ and $L_S(v) = L_G(v) \forall v \in V_S$.

Definition 2: Assume that $S = (V_S, E_S, L_S)$ is subgraph of graph $G = (V_G, E_G, L_G)$, then subgraph isomorphism is an injective function $f : V_S \rightarrow V_G$ such that $L_S(v) = L_G(f(v)) \forall v \in V_S$ and $\exists(f(u), f(v)) \in E_G \forall (u, v) \in E_S$.

Definition 1 gives an explanation for the subgraph relation between two graphs. Definition 2 is about mapping subgraphs to nodes of the graph. There should be at least one subgraph isomorphism from S to G in order to say S is a subgraph of G . Counting the number of isomorphisms can be used as a frequency of subgraph in a graph, but this metric is not anti-monotonic. Anti-monotonicity is a property where if S_1, S_2 are subgraphs of G and S_1 is subgraph of S_2 , then frequency of S_1 should be greater or equal to frequency of S_2 . Having anti-monotone property is important for the algorithms which use prune and search techniques where gSpan is one of them.

There are different metrics used as a frequency of a subgraph. Mainly used frequency metrics are maximum independent set (MIS), harmful overlap (HO) and minimum image based (MNI). Let us assume S is a subgraph, G is a graph and $\{f_1, f_2, \dots, f_m\}$ are all subgraph isomorphisms of S to G . To calculate MIS frequency, first new graph is constructed where nodes are subgraph isomorphisms and there is edge between f_i, f_j if $\{f_i(v) \mid \forall v \in S\} \cap \{f_j(v) \mid \forall v \in S\}$ is not an empty set. Then, a maximum independent set is found on a new graph which gives us MIS frequency. The only different step in calculating HO frequency is adding edges. Edge is added between subgraph isomorphisms f_i, f_j if there exist $v_s \in S$ such that $f_i(v_s) = f_j(v_s)$. In MNI frequency, $F(v)$ is calculated for all nodes of S , where $F(v)$ is a distinct set of $\{f_1(v), f_2(v), \dots, f_m(v)\}$. Then MNI frequency is minimum of $\{|F(v)| \mid \forall v \in S\}$. We can see interpretations of MIS, HO and MNI frequencies in Figure 3.4.

Calculation of MIS and HO frequencies are NP-complete while finding MNI frequency problem is in P. Also, MNI frequency is upper bound for both MIS and HO frequencies [35]. We have used MNI frequency because it can be calculated efficiently and frequency value range is wider than others.

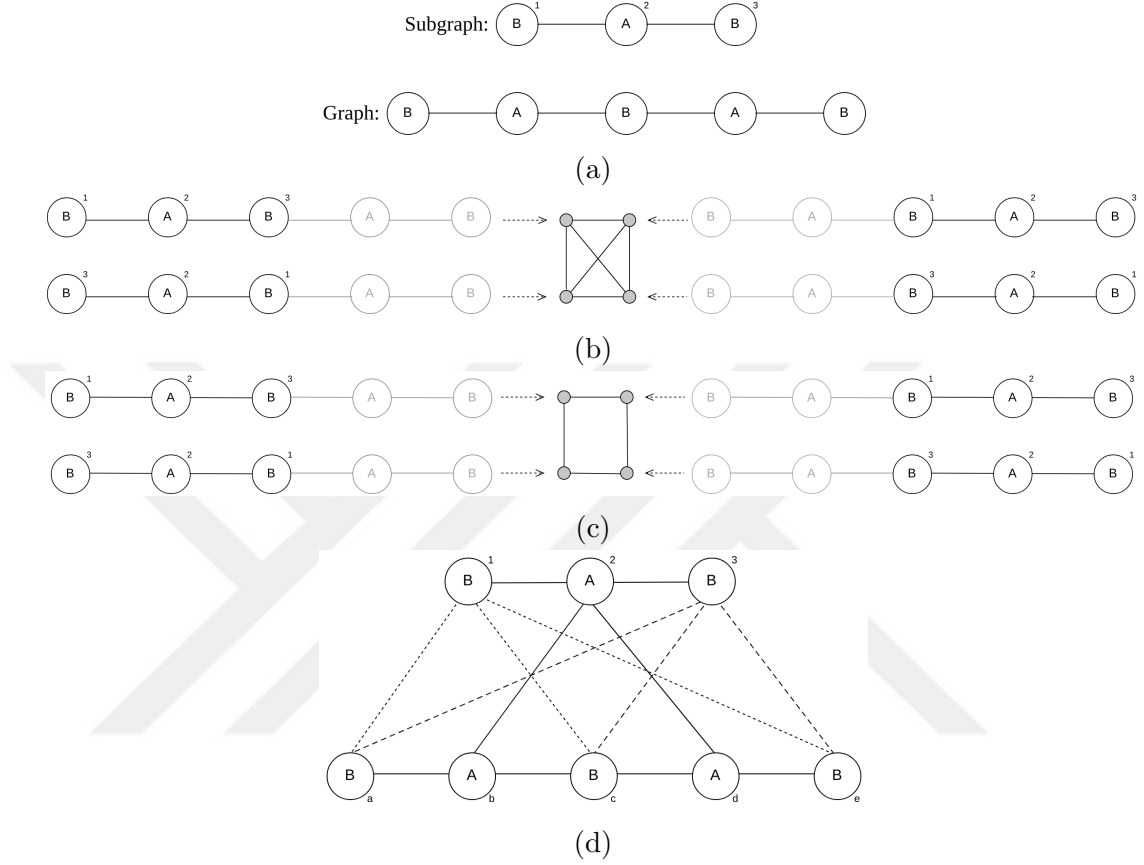


Figure 3.4: Illustration of different frequencies. (a) shows a sample graph and subgraph that is going to be counted with different frequencies. MIS frequency is shown in (b), which is equal to 1. HO frequency is equal to 2 for the given graph and subgraph and can be seen in (c). Differences between HO and MIS frequencies can be seen on the graphs at the middle of (b) and (c). Finally, MNI frequency is in (d) and it is equal to 2. $F(1) = \{a, c, e\}$, $F(2) = \{b, d\}$, $F(3) = \{a, c, e\}$

3.4.1.3 Supervised subgraph mining

We separate train data according to class label and apply subgraph mining. Number of nodes and graphs vary among different class labels. If we give the same support for different data sets we may not get a similar number of frequent subgraphs. To get a similar amount of subgraphs from each class sample, gSpan algorithm is run with different support values. Our method takes a number which is the required number of subgraphs from each class data set. Support starts with high value and decreased by some amount until at least the desired

number of subgraphs is found. If the number of subgraphs is greater than desired number no subgraph is removed, because after this step selection is applied on subgraphs.

3.4.2 Subgraph selection

We have experiments for mining for each class of samples as well as one mining for all samples. There could be subgraphs which are frequent in two different class samples. Also there could be similar subgraphs such as two subgraphs where the only difference is the label of one node. If one subgraph is frequent then it is likely to find similar subgraphs as frequent ones. Another similarity is, a subgraph which is a supergraph for another subgraph. Graph A is a supergraph of another graph B if A can be constructed from adding edges or nodes to the graph B . To eliminate similar and identical subgraphs, we decided to apply selection on frequent subgraphs. Instead of selection, if we mine only a small amount of subgraphs, again we will end up with very similar subgraphs, so the selection step helps us to get informative subgraphs.

As we explained in Section 3.1 we have training, validation and test sets. For each subgraph we define $p(s, c)$ which gives the probability of being from class c for subgraph s . Probability distribution is normalized i.e. sum of $p(s, c_i)$ for all classes is equal to 1. $p(s, c)$ is calculated as

$$p(s, c) = \frac{\text{count}(s, D_c^{val})}{\text{count}(s, D_{all}^{val})} \quad (3.1)$$

where D_c^{val} is data set which contains samples for class c , if $c = all$ then it means all classes, in validation set. $\text{count}(s, D)$ is the number of occurrences of subgraph in $D = \{G_1, G_2, \dots\}$ where occurrence is 1 if there is at least one mapping from s to G_i , 0 otherwise.

We have decided to use a forward selection algorithm because of its simplicity. To apply forward selection, we need a heuristic function which gets selected subgraphs and gives a value that shows performance of classification. $heuristic(S, D)$

is a function which gets selected subgraphs and data set. Equations used to calculate heuristic function are

$$S = \{s_1, s_2, \dots, s_n\}, \text{selected subgraphs} \quad (3.2)$$

$$y_{pred}(G_i) = \begin{cases} c_{not-found} & \text{if } \sum_j^n count(s_j, \{G_i\}) = 0 \\ \operatorname{argmax}_c (\sum_j^n p(s_j, c) \times count(s_j, \{G_i\})) & \text{otherwise} \end{cases} \quad (3.3)$$

$$heuristic(S, D) = recall(y_{pred}, y_{true}) \quad (3.4)$$

where y_{true} is ground truth value for the sample. If no subgraph was found on the given graph then we give the label $c_{not-found}$. There are samples with no frequent subgraph on it, to find value for the $c_{not-found}$ we do experiments and can be found in the next chapter.

3.4.3 Fisher vectors

Until now, we have applied subgraph mining and selection on frequent subgraphs. We want to use subgraphs mining and selection results in a neural network. Subgraphs can have different numbers of nodes. Also, each sample can have varying numbers of subgraphs. Neural network we are going to use requires the same dimensional feature for all samples. Fisher vectors (FV) and bag-of-visual-words (BOV) can be used to extract the same dimensional feature from subgraphs. BOV is used for image recognition and retrieval where the aim is to extract comparable features from a bag of descriptors or words. Here, we can think of a subgraph as a bag and each node is a descriptor or word. FV is an extension of the BOV model and it can be seen as a soft BOV [48]. FV gives richer information and we have used it to extract features from subgraphs since we are going to use it with the neural network which can learn from richer information.

We have 3 options to choose a descriptor for the nodes of the subgraphs: output of Resnet50, output of PCA and Kmeans cluster. [49] states that applying PCA before FV improves accuracy and decreases the memory footprint of the

representation. Because of that, we choose PCA output. We use FV with 16 Gaussians, as a result, $2 \times 16 \times 50 = 1600$ dimensional vector extracted. Also, we decrease the dimension of output of FV due to reasons mentioned in Section 3.6.1 and we achieve that by using PCA. At the end we have one Fisher vector for each subgraph. In order to have a fixed dimensional feature for all samples, we took the mean of the Fisher vectors. There are some samples where no frequent subgraph is found on that sample. For those samples, we have used a random vector with dimension 64. The same random vector has been used for all samples with no frequent subgraph. After all operations there will be one 64 dimensional vector for each sample.

Implementation of the Fisher vector we are using is from VLFeat library and its version is 0.9.21 [50]. We calculate Fisher vectors by enabling *square-root* and *normalized* properties which makes the library to use signed squared-root and to apply normalization respectively.

3.5 Graph representation

Graph representation is used to represent objects and their relations. This representation has been used in other computer vision tasks such as segmentation and human object interaction detection. Our input source, ROI image, is one of the sources that can be converted to graph. We explained the process of building graphs in previous sections and we need to define how to process those graphs.

Most of the GNNs used in digital pathology are composed of traditional NN layers, graph convolution and pooling layers. Input to the graph convolution layer is a graph and it uses both nodes and edges of the graph. In general, both nodes and edges can have features. In our case, we do not have edge features. If we want to represent our graphs with pairs (H^l, A) , where H^l is a matrix of size $N \times D^l$. It represents node features with N nodes where each node has a D^l dimensional feature vector. The matrix A is an adjacency matrix to represent neighboring relationships between nodes. Since we do not have edge features, A

contains only zeros and ones. $A_{i,j}$ contains zero if there is no directed edge from i to j . If there is a directed edge from i to j , then it contains one. Graphs that we have constructed are undirected, which can be represented by two directed edges. In other words, $A_{i,j} = A_{j,i}$ for all (i,j) pairs. Graph convolution layer can be seen as a function f , which gets H^l, A and gives us H^{l+1} matrix. As an example, $f(H^l, A) = AH^lW^l = H^{l+1}$ is a simple graph convolution layer, where $W^l \in D_l \times D_{l+1}$. D_{l+1} is the feature dimension of the new feature. In most of the GNNs, the new feature dimension is smaller than the previous feature dimension.

In our model, we have used GAT layers because of its good performance in our data set. The formula of the GAT layer for one node can be seen in Equation (3.5). Let us assume $h_i^l \in D_l \times 1$ is input feature for node i and $h_i^{l+1} \in (D_{l+1}P) \times 1$ is an output feature for node i . Then, output feature for the node i is calculated as

$$h_i^{l+1} = \parallel_{p=1}^P \left(\sum_{j \in \mathcal{N}(i)} \alpha_{i,j} \Theta h_j^l \right) \quad (3.5)$$

$$\alpha_{i,j} = \frac{\exp(\text{LeakyReLU}(a^T[\Theta h_i^l \parallel \Theta h_j^l]))}{\sum_{k \in \mathcal{N}(i)} \exp(\text{LeakyReLU}(a^T[\Theta h_i^l \parallel \Theta h_k^l]))} \quad (3.6)$$

where symbol $\parallel$ represents concatenation operation. In the GAT layer, there are parallel inner layers and it is defined as heads. In the Equation (3.5) it is defined with P . Result of those parallel inner layers are concatenated as can be seen in the Equation (3.5). Instead of concatenation, mean operation can be used as well. $\mathcal{N}(i)$ gives us a list of vertices which are neighbors to vertex i , including i . Attention coefficient $\alpha_{i,j}$ calculation is shown in Equation (3.6). Learning parameters are $\Theta \in D_{l+1} \times D_l$ and $a \in 2D_{l+1} \times 1$ and each head has its own learning parameters. The Equation (3.5) is applied to all of the nodes in the graph, as a result there is a new feature vector for each node.

The GAT layer changes only the dimension of the node features while graph structure remains unchanged. In our problem, we have graph labels and we need to extract one label from the graph. There are graph pooling layers which change

graph structure. They are used to decrease the number of nodes in the graph. We do not use this kind of layer, instead we took the mean of the node features to decrease the number of the nodes to one.

We have used a dropout layer between GAT layers and it works as follows. First, it chooses some of the dimensions from the feature vector. Then, it sets zero for those dimensions in all of the nodes in one graph. There is a summation operation in the GAT layer, which takes the weighted sum of the neighbor node features. To increase the effect of dropout, it is applied as described and taken from work [51].

3.6 Combining features

Until now, we extracted graphs and frequent subgraphs from the ROI sample. First, we tried to concatenate features from different sources and designed a fusion network. There could be better ways than using concatenated features to exploit the mining output. Additional models are designed to check if embedding frequent subgraph mining output into graph is effective. Design of the parts of the models that process graph is inspired by the work [1].

3.6.1 Fusion network

The Fusion network we have designed combines features from three sources. Since we are concatenating features without any complex operation, we wanted to add more higher level features. To do that, we are applying a sliding window on ROI with window size 1000×1000 . Then, we are using the same pretrained Resnet50 to extract ready to use features. PCA is applied after the Resnet50 extractor to decrease the dimension which decreases to 64 dimensions. To have a fixed dimensional vector for all ROIs, we take the mean of the window features. From now on, we call this source *roi-feature*.

Second source is the graph that we have constructed from each ROI. At the beginning there is one GAT layer with 4 heads and its input, output dimensions are 2048 and 64×4 respectively. Then, activation function and dropout is applied to add non-linearity and to prevent overfitting. After that, there is another GAT layer with one head. Its input is a 256 dimensional vector, since activation function and dropout do not change feature dimension. Output dimension 64, to have equal dimensional vectors from all three sources. We are using the term *graph-feature* to refer to this source.

Third source that we have used in the fusion network is extracted from sub-graphs. The way we extract is explained in Section 3.4.3. The term used for this feature is *subgraph-feature*. Again, to check the effect of each source, we have decided to have the same dimensional vector from all three sources.

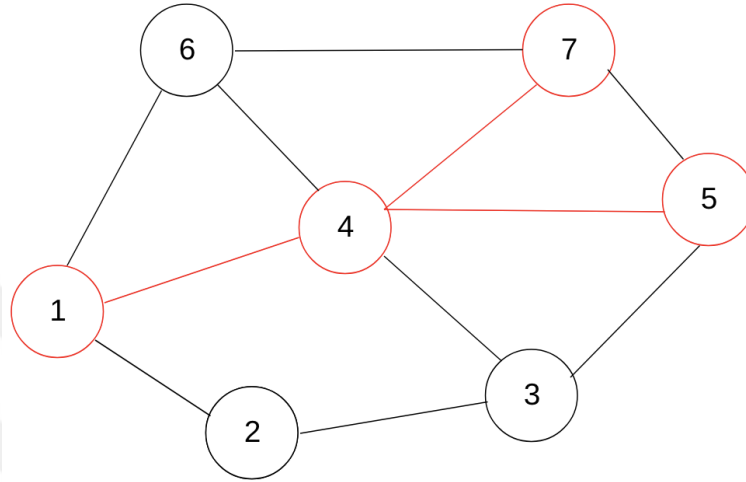
We can see an illustration of the fusion network in Figure 3.1. Concatenation of the *roi-feature*, *graph-feature* and *subgraph-feature* is connected to a fully connected layer. Input dimension of this layer is the sum of the dimensions of the inputs with fixed output dimension, which is 16. Then, ReLU activation function is applied to add non-linearity. It is followed by another fully connected layer with output dimension 4. Finally, a softmax function is added to get the probability vector from the output of the last fully connected layer.

Before deciding the number of layers or feature dimension of input sources, we experimented with a more complex fusion network, such as 3 fully connected layers at the end instead of 2 layers. Also, we experimented when input sources have higher feature dimension. We saw that decreasing complexity and feature dimensions give improved results on classification. Probably, it is because of the smaller number of samples in our data set. Because of improved results, we decided to use a simpler model and smaller feature dimensions.

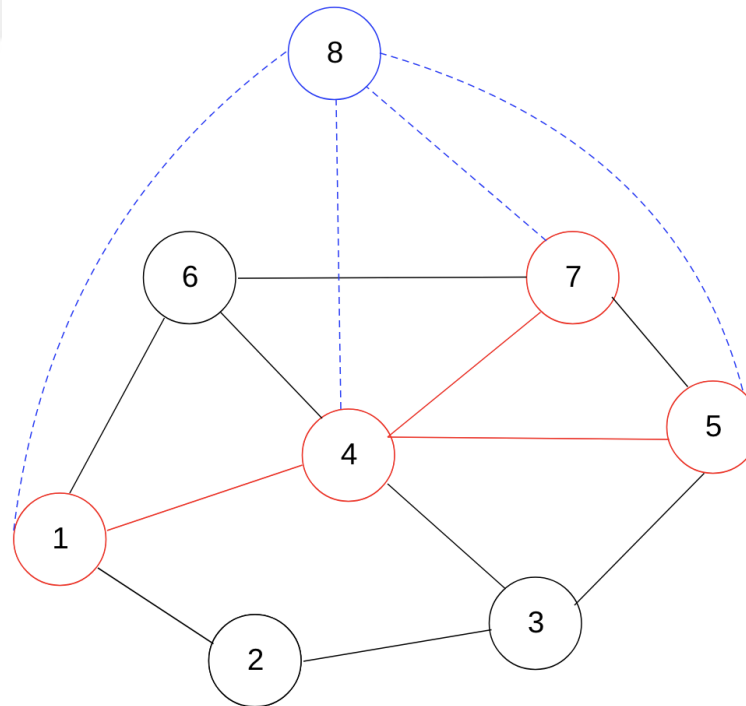
3.6.2 Embedding subgraphs into graph

The Fusion network may not exploit both subgraph and graph features together because of simple concatenation operation. We have tried to embed the subgraph features into graph to check if it helps to learn better. Some of the possibilities to embed the subgraphs are adding edge features, using weights for the subgraph nodes or changing the graph structure. We have changed the graph structure by adding a new node for each subgraph. That new node is connected to the subgraph nodes with undirected edges. An example of adding the new node from the subgraph can be seen in Figure 3.5. Figure 3.5a shows a graph with subgraph nodes and edges labeled with red color. The new node can be seen in Figure 3.5b where added node and edges are shown with blue color. The structure can be counted as hierarchical-graph, because subgraphs represent multiple patches and there are connections between the new nodes and patch nodes. Features of the new node are obtained from averaging subgraph node features. There are graphs with no frequent subgraphs found on it. For those graphs, one new node is added. The node feature vector is the random vector with the same dimension and there is no new edge in this case. The same random vector is used for all graphs with no frequent subgraph. To represent the new graph structure, we are using the term *new-node-graph* in the following and this chapter. To process the *new-node-graph* we have used models which are different from fusion network and the following paragraphs define those models.

First model is a simple model that is used in the fusion network and we call this model *simpleGNN*. It is composed of GAT layer, ReLU, dropout, GAT layer and mean and softmax operation. Here, we have changed GAT layer dimensions and number of heads. In the first GAT layer, there are 8 heads with each head having 64 dimensional output. Output dimension of the second GAT layer is 4 and it has 8 heads. This time output of heads are not concatenated, instead it is averaged. Number of graph convolutional layers defines which nodes can affect node features. In our case there are two of them. That means, features of one node can be affected by nodes within distance two where the distance is the number of the edges between nodes. Addition of new nodes introduces paths



(a)



(b)

Figure 3.5: Example for adding a new node from subgraph. (a) shows a sample graph with subgraph nodes and edges labeled with red color. Added a new node and its edges are shown in (b). New edges are with blue dashed lines and the new node is node 8.

between subgraph nodes with distance two. Those nodes are already connected, but the distance may be more than two. This may help to learn better for node features of the subgraph nodes.

New nodes represent subgraphs and they may be more informative than patch nodes. *sepLayerGNN* is designed to have a separate layer for patch nodes and new nodes. It is obtained by adding two more GAT layers to *simpleGNN* in parallel. GAT layer for patch nodes update features of patch nodes by using features of all nodes. Similarly, the GAT layer for new nodes updates features of new nodes by using features of all nodes. Other parts of the model are exactly that same as in *simpleGNN*.

In most of the graphs the number of subgraphs is smaller than the number of the nodes on that graph. The last operation is mean in previous models and the effect of new nodes on output is less compared to patch nodes. To increase effect, we can take the weighted sum of the node features instead of the mean. But, the number of nodes in the graph is not the same for all graphs. By considering that, we have decided to take the mean of the node features separately for each node type. To do that, *combineEndGNN* is defined by changing *simpleGNN* where the change is on mean operation. There are two mean operations instead of one. First mean operation takes the mean of the patch node features and second mean operation takes the mean of the new node features. Then, the sum of those mean operation results is given to the softmax. We can see the diagram for this model in Figure 3.6. Similarly, *sepLayerCombineEndGNN* is defined in the same way, except the base model is *sepLayerGNN*.

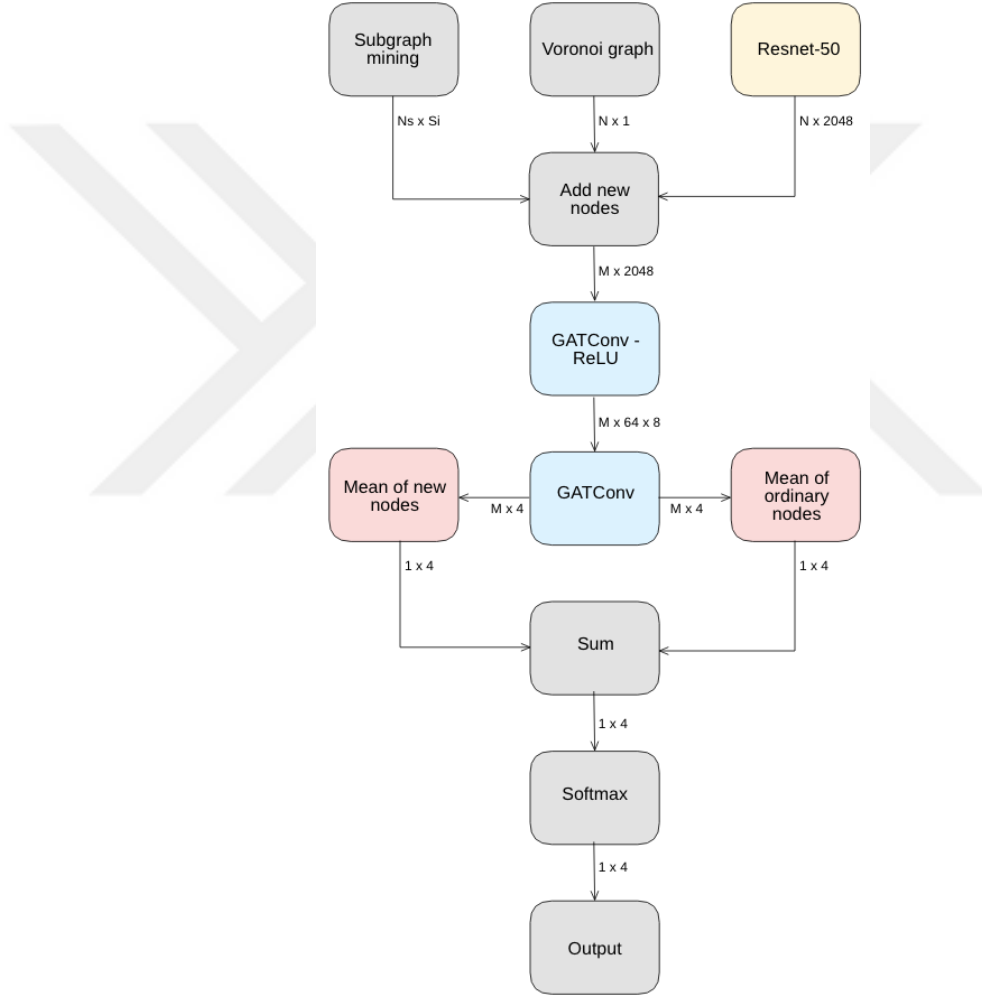


Figure 3.6: Diagram that describes the model *combineEnd*. The block "Add new nodes" adds new nodes as mentioned in Section 3.6.2. The number M is equal to the number of nodes plus number of new nodes. The main difference is, there are separate mean operations for different types of the nodes.

Chapter 4

Experiments and Results

Experiments, results and discussions are presented in this chapter. We use two different magnifications at the beginning. Magnification 10X performs better on subgraph selection results, after that we use only 10X magnification. Patch detection parameters and why we use multiple magnifications is described in the first section. Then, t-SNE patch visualization results are presented for both magnifications. After that, experiments on the number of clusters and how we chose the number of clusters for Kmeans algorithm is mentioned. It is followed by subgraph mining and selection experiments. Then, in Section 4.5, results of ablation study with three different sources are discussed. Additionally, embedding frequent subgraphs into graph results are presented in the same section. Finally, comparison of our work to another work is done in Section 4.6.

4.1 Patch and magnification

In Chapter 3, we explain the process of locating patches. There is one value used as a threshold to convert probability maps into 0-1 values. We have used 0.75 as a threshold value in our experiments. After locating patches NMS is applied to choose more informative patches and it requires an IOU-threshold value. For

this value, 0.1 is used to have patches with small overlapping regions. Size of the patches is 224×224 and it is the same for all magnifications. Having constant patch size for different magnification values is similar to choosing different patch sizes in one magnification. We set patch size constant, because we are using a pretrained model that requires constant sized inputs. Data set we are using has different magnification values. 5X and 10X magnifications are widely used in breast histopathology tasks, so we decided to use both, 5X and 10X, at the beginning. Then we choose one of them when we see there is a significant difference. First, we plot patch features and checked if there is any difference. To do that, we use t-SNE dimensionality reduction technique to convert patch features into two dimensional feature.

4.2 Patch feature visualization

t-SNE is a technique to visualize high-dimensional features and was introduced by Maaten and Hinton. This technique does not guarantee to find the best dimensionality reduction based on its cost function [7], but it is worth checking to see if it helps us or not. We chose this technique to visualize samples from our data set because it is widely used to visualize high-dimensional data.

We can use ROI image and feature vector extracted from Resnet50 as an input to t-SNE. We use the Resnet50 feature since it has less number of dimensions and it is taken from the model which tries to learn representation from images. Decreasing dimensionality is recommended by the library and we use PCA to decrease dimension into 50. There are parameters used by this technique such as *learning-rate*, *perplexity* and *distance-metric*. We experimented with different parameters to find good visualization. In the Figures 4.1 and 4.2 we can see outputs of the t-SNE for the training data set. We use at most 5000 patches for each class. If the number of patches is smaller than 5000, all of the patches have been used to plot. We can see that there is an imbalance because of the diagnostic label *atypia*. To generate those plots we use *learning-rate* = 100, *perplexity* = 30 and squared Euclidean distance as a *distance-metric*. We use the Python library

scikit-learn v0.24.2 [52] to plot the data with t-SNE. Rest of the parameters are default parameters from the scikit-learn library.

Plots of the patches for both 5X and 10X magnifications can be seen in the Figures 4.1 and 4.2 respectively. In both of the Figures, (a) contains patches from *benign* samples, (b) contains patches from *atypia* samples, (c) contains patches from *in situ* samples and *invasive* sample patches are in (d). All of the sub-figures are divided by 5X5 regions. Inside each region, there is a number which shows what proportion of the patches from that class label are in that region. First, we try to find if there is clear separation between patches for different diagnostic labels. There is no clear separation between patches from different labels. If there is, then it means, the problem can be solved by using t-SNE dimensionality reduction. Then, we try to choose magnification based on plots. In both of the magnifications, *invasive* patches are clustered and located away from patches of other class labels. But, for the other 3 diagnostic labels, it is not the same. Because of this we have decided to continue to use both magnifications until we see a significant difference.

4.3 Patch clustering

We are applying the Kmeans clustering algorithm after extracting features with Resnet50. We need to define the number of clusters, but there is no guidance for this in this field. Entropy is one of the methods used to decide the number of clusters alongside the elbow method and silhouette analysis [53]. Entropy is a metric that measures how homogeneous the distribution is. The equations that define the entropy for cluster and class are shown in Equations (4.3) and (4.4). First, we are defining the following variables. We do not have class labels for the patches. We use the class label of the ROI, so patches belonging to samples with class label c get the same label c . First, we define variables h_{ck} , h_c , and $h_{.k}$ which is used in entropy calculation:

- h_{ck} : number of the patches assigned to cluster k where class label of those

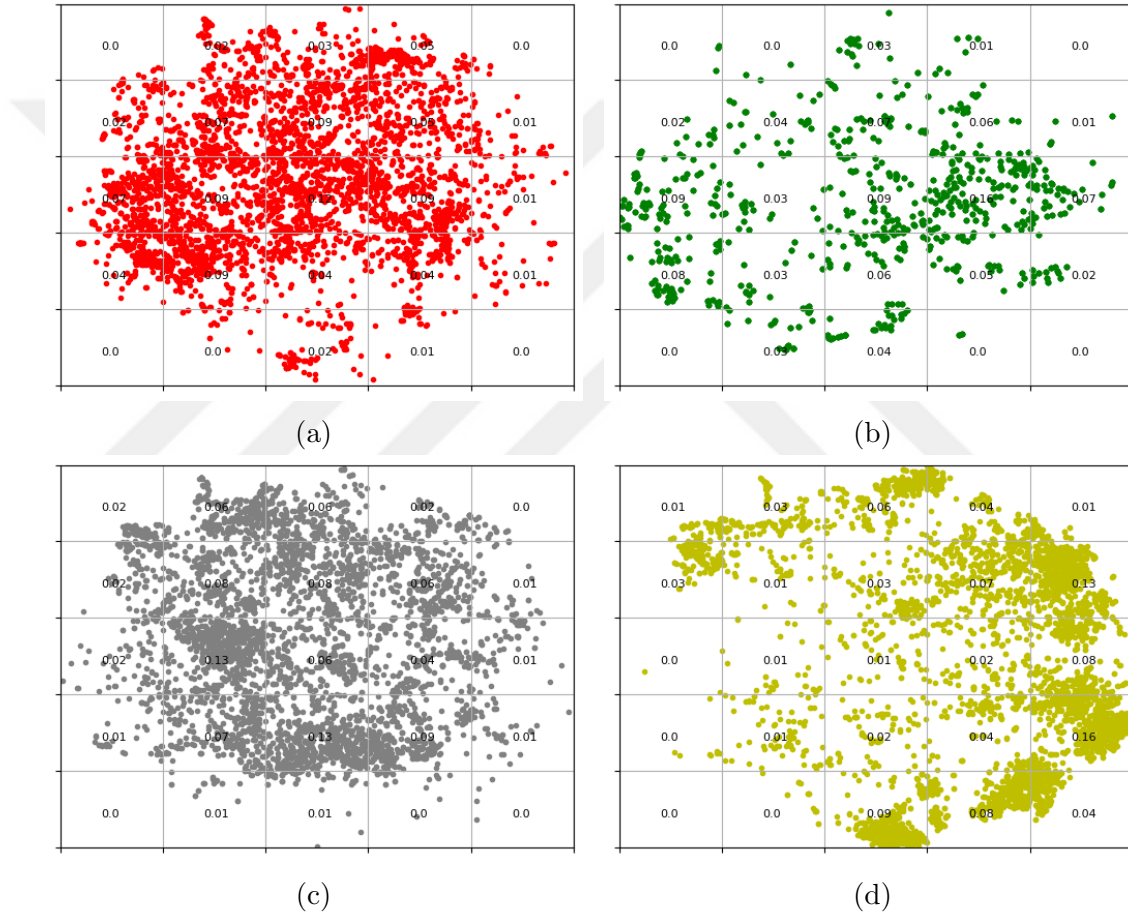


Figure 4.1: t-SNE plots for patches when 5X magnification is used. Patches from *benign*, *atypia*, *in situ* and *invasive* are in (a), (b), (c) and (d) respectively. Each plot is divided into regions to analyze better. Inside of each region there is a number which shows what proportion of the patches from that class label is in that region.

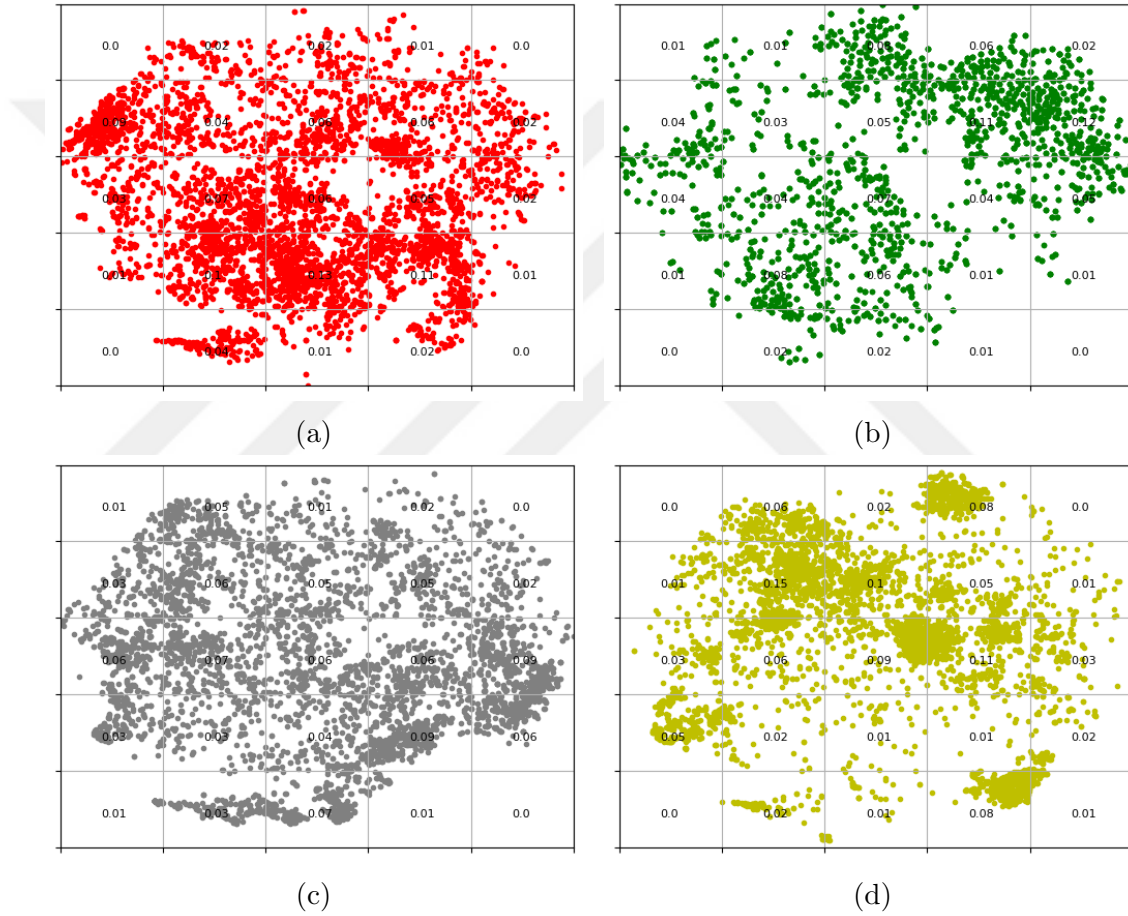


Figure 4.2: t-SNE plots for patches when 10X magnification is used. Patches from *benign*, *atypia*, *in situ* and *invasive* are in (a), (b), (c) and (d) respectively. Each plot is divided into regions to analyze better. Inside of each region there is a number which shows what proportion of the patches from that class label is in that region.

patches is c .

- $h_c = \sum_{k=1}^K h_{ck}$: number of the patches with class label c .
- $h_{.k} = \sum_{c=1}^C h_{ck}$: number of the patches with cluster label k .

Cluster entropy is calculated as

$$E_{cluster} = \frac{1}{\sum_{k=1}^K h_{.k}} \sum_{k=1}^K h_{.k} E_k \quad (4.1)$$

$$E_k = - \sum_{c=1}^C \frac{h_{ck}}{h_{.k}} \log \frac{h_{ck}}{h_{.k}} \quad (4.2)$$

which takes a weighted sum of the entropies for each cluster.

Similarly, entropy for class is calculated as

$$E_{class} = \frac{1}{\sum_{c=1}^C h_c} \sum_{c=1}^C h_c E_c \quad (4.3)$$

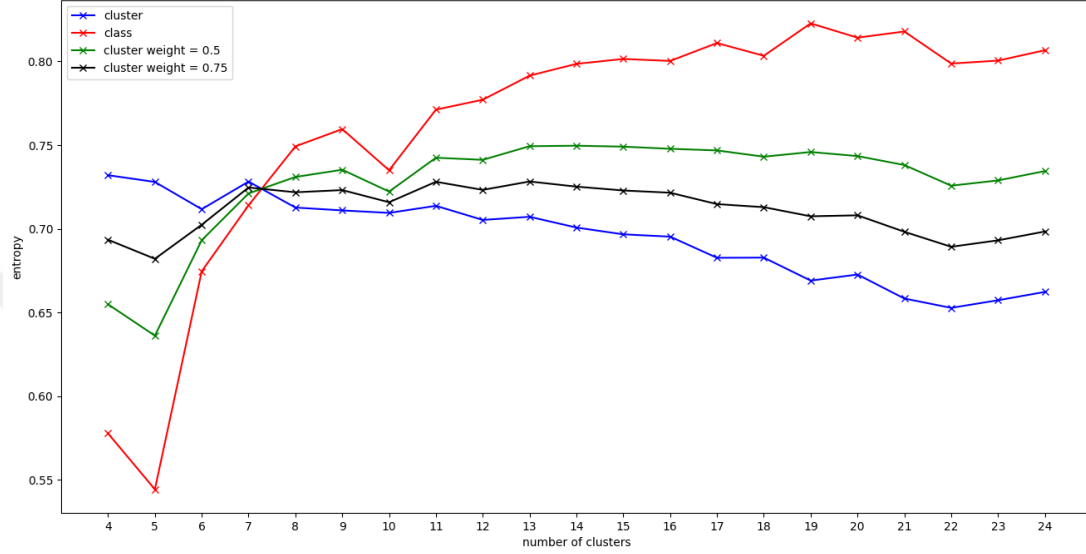
$$E_c = - \sum_{k=1}^K \frac{h_{ck}}{h_c} \log \frac{h_{ck}}{h_c} \quad (4.4)$$

which takes a weighted sum of the entropies for each class. In order to combine cluster entropy and class entropy, their weighted sum is taken as

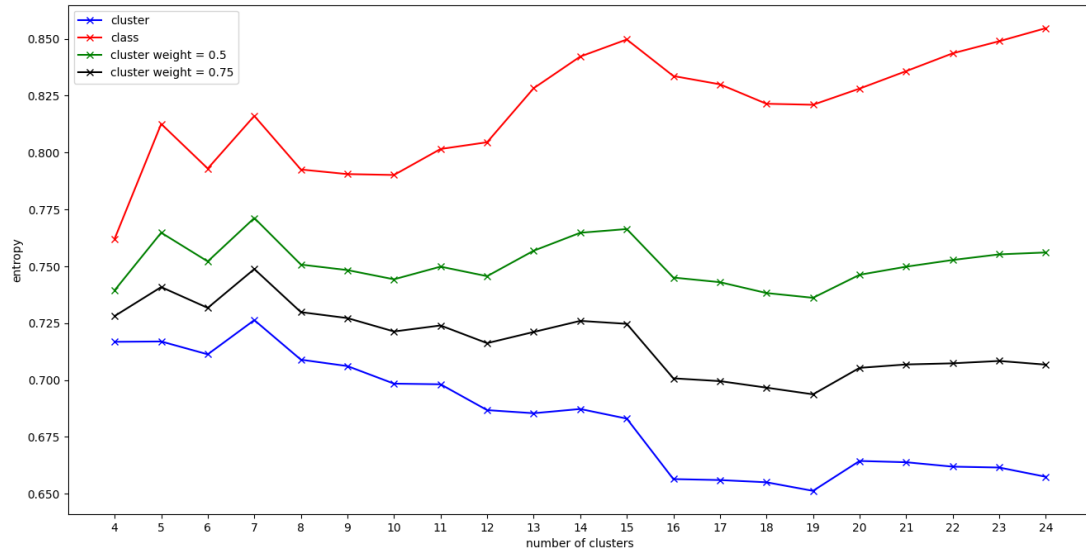
$$E = \beta E_{cluster} + (1 - \beta) E_{class} \quad (4.5)$$

where $\beta \in [0, 1]$ is a weight to adjust the effect of the cluster and class entropies.

In Figure 4.3, we can see plots of the class entropy, cluster entropy and their weighted sum. To get those entropy values, first, Kmeans is trained with the training data set. Then the validation data set is used to calculate entropy values. We calculate entropies when the cluster number gets values between 4 and 24. Entropy value 0 means the distribution is perfectly homogeneous. Entropy value increases when homogeneity is decreased. So, we need to find the number of clusters with the smallest entropy value. $E_{cluster}$ gives us information about how homogenous each cluster is in terms of the class labels. Similarly, E_{class} gives us



(a)



(b)

Figure 4.3: Entropy plots for 5X and 10X magnifications. In both of the (a) and (b), x-axis shows the number of clusters and y-axis shows the entropy value. Lines blue, red, green and black show cluster entropy, class entropy, weighted sum of entropies with 0.5 and weighted sum of entropies with 0.75 respectively. The smallest value weighted sum with 0.75 is 5 for 5X magnification plot. It is 19 for the 10X magnification plot

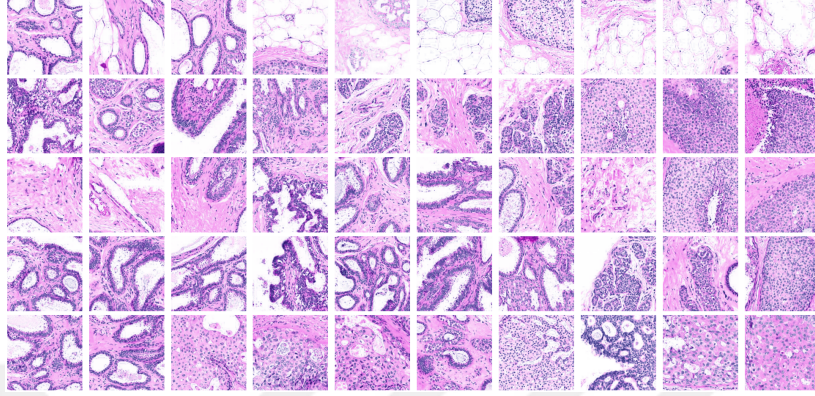


Figure 4.4: Sample patches for each cluster for 5X magnification. Each row belongs to one cluster.

information about how class samples are distributed into clusters. There are 4 lines in the Figure 4.3. Blue and red lines are for $E_{cluster}$ and E_{class} respectively, or it can be seen as $\beta = 1$ and $\beta = 0$ respectively. Green line shows entropy when $\beta = 0.5$. Cluster entropy may be more informative than class entropy, so we have added a line to the plot with $\beta = 0.75$. For the 5X magnification, when the number of clusters is equal to 5, it has the smallest entropy value for the same weighted sum as well as with $\beta = 0.75$. In the case of 10X magnification, 19 clusters gives us the smallest entropy for both the same weighted sum and when $\beta = 0.75$. We use 5 clusters for 5X magnification Kmeans and 19 clusters for 10X magnification Kmeans in all of the experiments below.

In Figure 4.4, we can see sample patches for each cluster. Each row contains patches from one cluster. Since there are only 5 clusters, we expect to see different kinds of patches in one cluster as well as there are similar patches in one cluster. Similarly, patches from 10X magnification are in Figure 4.5. There are clusters that contain specific patches like the first row from top and seventh row from bottom. We do not expect perfect clustering, because we have used pretrained model to extract the features. But, the clustering performs well for some of the clusters. While choosing patches for clusters we take at most one sample from each ROI to see if different ROIs have similar patches.

Number of the clusters for the Kmeans is mentioned above. Parameters other than number of clusters are the same for both magnifications. As an initialization

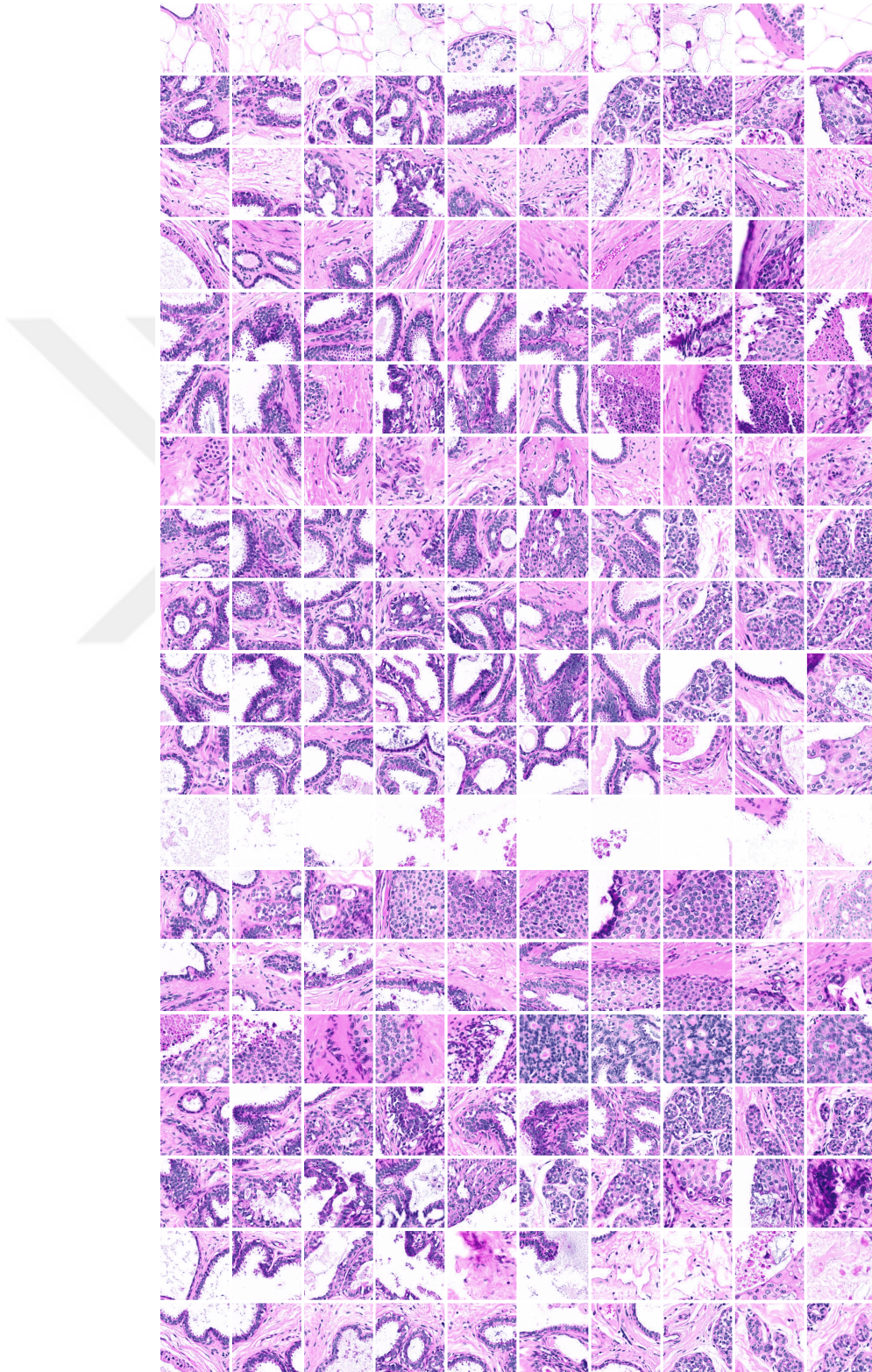


Figure 4.5: Sample patches for each cluster for 10X magnification. Each row belongs to one cluster.

method k-means++ is used. Classical expectation maximization algorithm is used as Kmeans clustering. Again, we have used scikit-learn library and other parameters are default parameters set by that library.

4.4 Subgraph mining and selection

One of the parameters we need to define while applying the graph mining is subgraph limitations. Here, we use the number of nodes as a limitation parameter for subgraphs. Since there are many similar subgraphs, only tree subgraphs have been mined. *Frequency* shows the number of occurrences needed to be counted as found and it is 1 in our experiments. Graph mining continues to run until it finds the most 1000 frequent subgraphs.

After that, subgraph selection is applied to get the most informative ones. Results of the subgraph selection can be found in Table 4.1. While applying subgraph selection it requires a class label which is given to the samples with no frequent subgraphs. The columns in Table 4.1 show those labels. The last column is for the case where a randomly chosen label is assigned to the samples with no frequent subgraphs. The simple classification described in the Section 3.4.2 uses samples to calculate probability and we have used the validation data set for it. The rows of the table get different magnifications, subgraph node size and mining approaches. As described, both 5X and 10X magnifications have been used. For the node size, the following ranges have been used {3-5, 3-7, 5-5, 5-7, 7-7}. There are two different approaches for mining. First approach uses class labels of the samples and applies graph mining for each class samples separately. Second approach applies graph mining for all samples without using class labels of the samples. The best recall score is achieved when magnification is 10X, subgraph node size is between 3 and 5 (inclusive), with no separate mining. The best performance for the experiments with magnification 5X is 0.541. There is a significant difference between best recall score of 10X magnification experiments and best recall for 5X magnification, so we decided to use 10X for the rest of the experiments. The number of selected frequent subgraphs is 32.

Table 4.1: Subgraph mining recall scores on simple classifier mentioned in 3.4.2 with validation set. Column shows a given diagnostic label when there is no frequent subgraph for the sample. There are recall score values for magnifications 5X and 10X, different node sizes and mining approaches.

Mag.	Node size range	Supervised mining	<i>benign</i>	<i>atypia</i>	<i>in situ</i>	<i>invasive</i>	Random label
5X	3-5	✓	0.468	0.504	0.394	0.368	0.400
5X	3-5		0.466	0.504	0.392	0.368	0.419
5X	3-7	✓	0.503	0.522	0.448	0.424	0.490
5X	3-7		0.505	0.505	0.510	0.505	0.505
5X	5-5	✓	0.502	0.541	0.462	0.452	0.485
5X	5-5		0.472	0.511	0.401	0.370	0.421
5X	5-7	✓	0.510	0.529	0.455	0.431	0.502
5X	5-7		0.447	0.461	0.442	0.416	0.446
5X	7-7	✓	0.491	0.510	0.434	0.420	0.443
5X	7-7		0.469	0.500	0.416	0.400	0.432
10X	3-5	✓	0.542	0.596	0.516	0.477	0.455
10X	3-5		0.608	0.641	0.496	0.456	0.485
10X	3-7	✓	0.530	0.590	0.558	0.437	0.413
10X	3-7		0.335	0.359	0.397	0.351	0.356
10X	5-5	✓	0.530	0.609	0.488	0.480	0.441
10X	5-5		0.563	0.609	0.493	0.488	0.519
10X	5-7	✓	0.564	0.622	0.521	0.458	0.419
10X	5-7		0.351	0.357	0.388	0.296	0.348
10X	7-7	✓	0.402	0.445	0.444	0.376	0.346
10X	7-7		0.558	0.566	0.442	0.404	0.430

In Figure 4.6, we can see the heat map plots for selected subgraphs. It contains plots for training and validation sets. The value used for the heat map is the number of frequent subgraphs on the samples. We have clipped the values greater than 20 to make samples with less number of frequent subgraphs more visible. There are more white regions in the training set plot than validation set plot. It is reasonable, because training set have been used to mine the frequent subgraphs. All selected frequent subgraphs can be found in Figure 4.7.

4.5 Combining features

Experiments and results of combining features from different sources are presented in this section. First, fusion network related parameters, results and discussions are mentioned. Then, outcomes of the models for embedding subgraph feature into graph are presented. To measure the performance of the classifiers, we use g-mean, precision, recall and f-measure with macro averaging. Precision

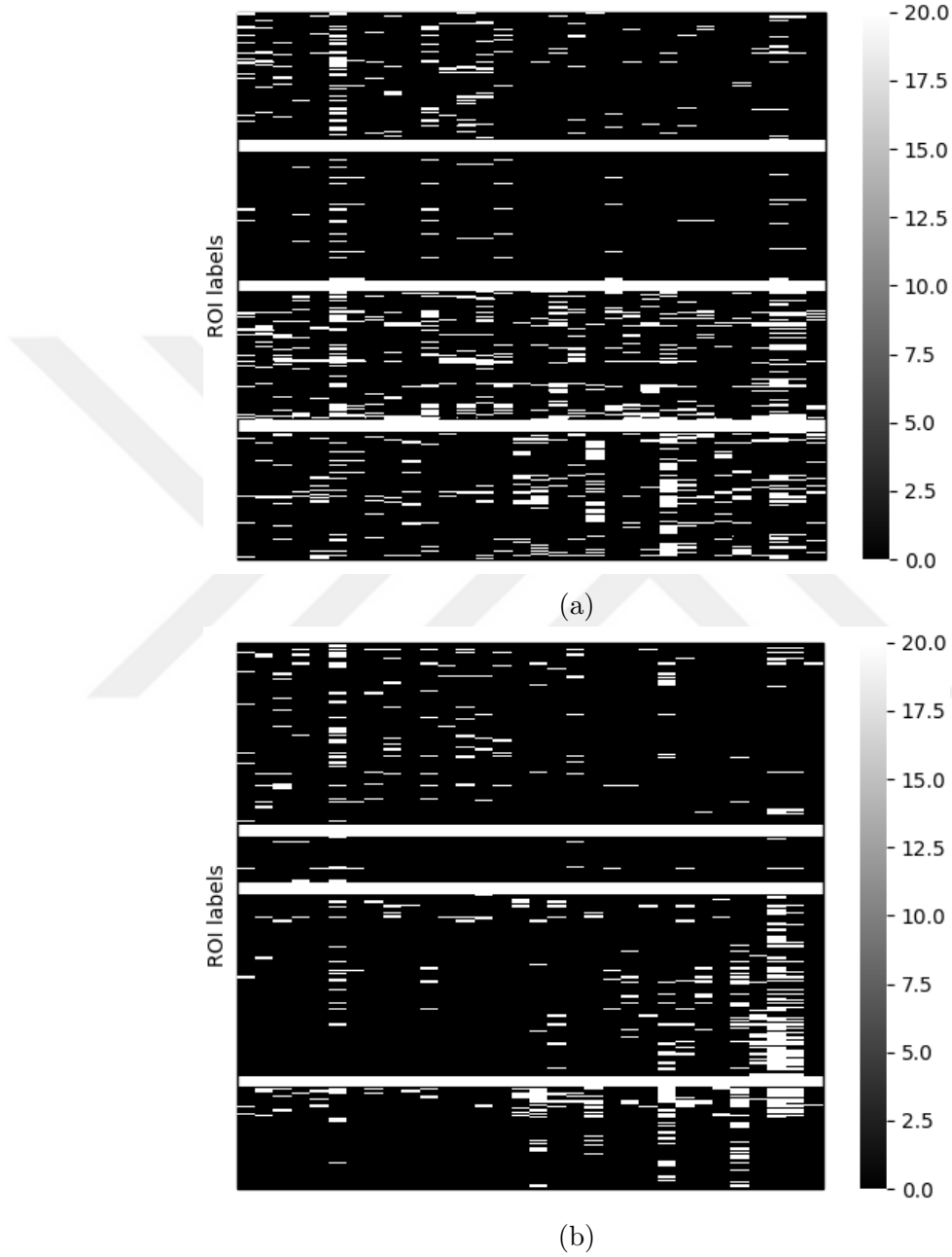


Figure 4.6: Heat map plots of the selected frequent subgraphs for training (a) and validation (b) sets for 10X magnification. Heat map uses the number of frequent subgraphs on the sample as heat value. The values have been clipped if it is higher than 20 to visualize samples with less number of frequent subgraphs better. Each row contains a heat map plot for one sample and there is one column for each frequent subgraph. Samples for each class are shown together and separated by rows which contain white color. In both of the sub-figures groups are *benign*, *atypia*, *in situ* and *invasive* from top to bottom.

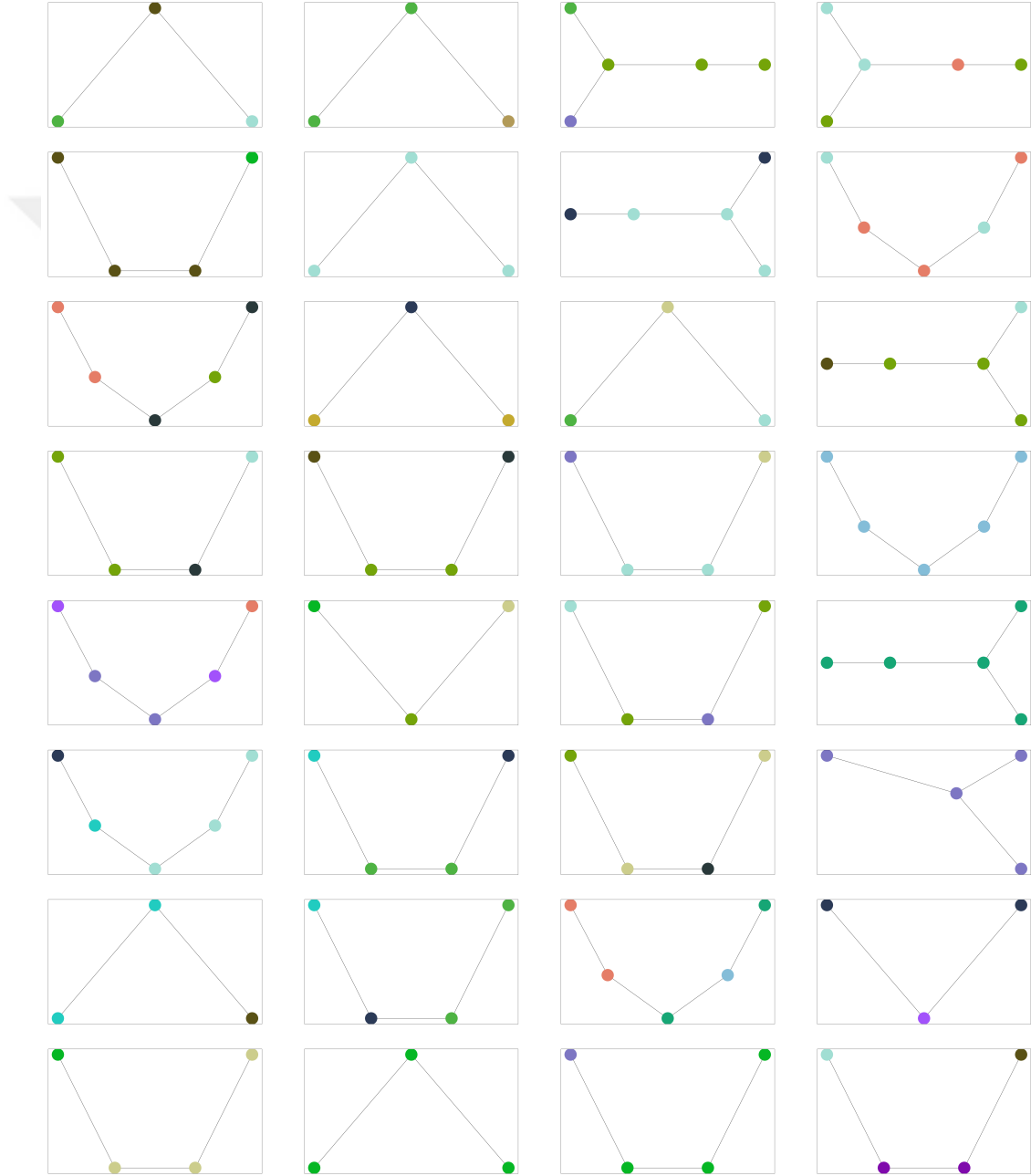


Figure 4.7: All selected frequent subgraphs for 10X magnification.

is calculated as follows. First, precision for each diagnostic label is calculated. Then, average or unweighted mean operation is applied to get precision value for the classifier. Similarly, for recall and f-measure, average of scores for each class is taken as a final value. G-mean score is calculated as the geometric mean of the recalls for each class. To do the experiments of NNs, we have used libraries PyTorch v1.10.0 [54] and PyTorch Geometric v2.0.2 [55].

The common parameters for all of the models are as follows. Training runs for 40 epochs where the best model is chosen based on loss. The model with the smallest loss value on the validation set is defined as the best model. Loss calculation is done after each epoch. We use Adam optimizer with cross entropy loss while training the models. To group the samples into batches, we use a number of nodes instead of a number of samples due to varying sized graphs. Batches are built as follows. The batch takes samples until there are less than b_n nodes in the batch where b_n is a parameter. In the following sections batch size n means the data set is distributed into batches with $b_n = n$. Validation set recall scores have been used to decide hyperparameters of the experiments.

4.5.1 Fusion network

The model fusion network described in Chapter 3 is a model to have ablation study for different sources. The sources are *roi-feature*, *subgraph-feature* and *graph-feature* as described in Chapter 3. We use the same parameters to train when the model does not use *graph-feature* i.e $\{roi-feature\}$, $\{subgraph-feature\}$ and $\{roi-feature, subgraph-feature\}$. The cases when *graph-feature* is used are trained with another set of parameters. Some of the diagnostic classes are easier to classify and to make the learning process better we use weight for each class loss. They are 2, 2, 2 and 1 for *benign*, *atypia*, *in situ* and *invasive* respectively and validation result is used to decide the weights. Parameter specific to each group is defined as follows:

- $\{roi-feature\}$, $\{subgraph-feature\}$ and $\{roi-feature, subgraph-feature\}$: These

experiments are trained with a learning rate value 0.0004 where it is multiplied by 0.990 on each epoch. Batches of these experiments have 32 samples.

- $\{graph\text{-}feature\}$, $\{roi\text{-}feature, graph\text{-}feature\}$, $\{subgraph\text{-}feature, graph\text{-}feature\}$ and $\{roi\text{-}feature, subgraph\text{-}feature, graph\text{-}feature\}$: Learning rate for these experiments is 0.0002 where multiplier for decreasing the learning rate is 0.975. 2048 is used as batch size while training fusion network with given input sources. Fusion network contains dropout in the part which processes *graph-feature* and its probability is 0.9.

In Table 4.2, we can see scores for different metrics. Below there are discussions for one source, two sources and three sources experiments:

- One source: The model with *graph-feature* performed better in terms of precision, recall and f-measure while *roi-feature* is the best on g-means scores. The worst performance for the model with *graph-feature* is obtained on diagnostic class *atypia*. *Roi-feature* is better on classifying *invasive* samples and samples from other 3 classes have f-measure score around 0.300. Model trained with input source *subgraph-feature* performed worse than the other two models. Class recall for *benign* is zero as well as *in situ* class recall is close to zero.
- Two sources: There are three cases when two input sources have been used. The best among them in all of the metrics uses *subgraph-feature* and *graph-feature*. Input *subgraph-feature* performed the worst when it is used individually, but it is better when used alongside *graph-feature*. Performances of other two cases are similar where the model trained with input pair *roi-feature* and *graph-feature* beats with small difference in all of g-mean, precision, recall and f-measure metrics.
- Three sources: The last row in the Table 4.2 shows scores for the model trained with all sources. The lowest value in that row is precision score for *atypia* class. Recall value for the same class is 0.545. The scores for other classes are greater than 0.6, so the model struggles to learn and classify samples from the *atypia* class.

Table 4.2: G-mean, precision, recall and f-measure scores for the fusion network for different input sources. B, A, IS, INV and Avg. are used to represent *benign*, *atypia*, *in situ*, *invasive* and average respectively. Input sources are combinations of *roi-feature*, *subgraph-feature* and *graph-feature*. Precision, recall and f-measure are calculated by macro averaging and g-mean is the geometric mean of class recalls.

ROI	Feature		Gmean	Precision					Recall					F-measure				
	Subgraph	Graph		B	A	IS	INV	Avg.	B	A	IS	INV	Avg.	B	A	IS	INV	Avg.
✓			0.453	0.750	0.136	0.420	0.667	0.493	0.244	0.682	0.273	0.929	0.532	0.368	0.227	0.331	0.776	0.426
	✓		0.000	0.000	0.174	1.000	0.281	0.364	0.000	0.727	0.013	0.929	0.417	0.000	0.281	0.026	0.432	0.184
		✓	0.381	0.841	0.200	0.707	0.576	0.581	0.772	0.045	0.688	0.875	0.595	0.805	0.074	0.597	0.695	0.568
✓	✓		0.524	0.821	0.175	0.517	0.766	0.570	0.187	0.773	0.597	0.875	0.608	0.305	0.286	0.554	0.817	0.490
✓		✓	0.560	0.802	0.286	0.641	0.632	0.590	0.691	0.273	0.532	0.982	0.620	0.742	0.279	0.582	0.769	0.593
	✓	✓	0.648	0.823	0.295	0.603	0.700	0.605	0.642	0.591	0.532	0.875	0.660	0.721	0.394	0.566	0.778	0.615
✓	✓	✓	0.710	0.856	0.293	0.769	0.852	0.693	0.772	0.545	0.649	0.929	0.724	0.812	0.381	0.704	0.889	0.697

4.5.2 Embedding subgraphs

In embedding subgraphs into the graph section, we use two input sources: the graph we generate and *new-node-graph* obtained by adding a new node. The latter is introduced to see if changing graph structure has an effect. The models we are using are *simpleGNN*, *sepLayerGNN*, *combineEndGNN* and *sepLayerCombineEndGNN* which are defined in Chapter 3. Batch size for all of the models is 1024. The models trained with learning rate 0.0002 and learning rate is multiplied with 0.900 after each epoch. Weights used for the loss are 7, 10, 4 and 1 for *benign*, *atypia*, *in situ* and *invasive* respectively. Again, validation set results have been used to define weights. There is dropout after the first GAT layer and its probability is equal to 0.9.

Table 4.3 shows us g-mean, precision, recall and f-measure scores for different models and inputs. Below there are discussions for the models:

- *simpleGNN*: This model trained with two different inputs, graph and *new-node-graph*. Graph is an ordinary graph obtained from ROI samples without any additions. We can see that there is a small increase in the averaged metrics when *new-node-graph* input is used. For the classes *benign* and *invasive* scores of the classifier are very similar while *simpleGNN* with *new-node-graph* has greater values with small difference. We see that there is a small increase when new nodes are added to the graph by using subgraph information, but the table shows that changing the model to process subgraph features differently gives us a significant increase.

Table 4.3: G-mean, precision, recall and f-measure scores for embedding subgraph experiments.. B, A, IS, INV and Avg. are used to represent *benign*, *atypia*, *in situ*, *invasive* and average respectively. Table contains scores for *simpleGNN*-graph, *simpleGNN*, *sepLayerGNN*, *combineEndGNN* and *sepLayerCombineEndGNN* where *simpleGNN*-graph is for experiment when the model *simpleGNN* is trained with graph without new node and other rows are for experiments when models are trained with *new-node-graph*. Precision, recall and f-measure are calculated by macro averaging and g-mean is the geometric mean of class recalls.

Model	Gmean	Precision					Recall					F-measure				
		B	A	IS	INV	Avg.	B	A	IS	INV	Avg.	B	A	IS	INV	Avg.
<i>simpleGNN</i> -graph	0.439	0.810	0.105	0.662	0.680	0.564	0.764	0.091	0.584	0.911	0.588	0.787	0.098	0.621	0.779	0.571
<i>simpleGNN</i>	0.496	0.803	0.188	0.700	0.680	0.593	0.764	0.136	0.636	0.911	0.612	0.783	0.158	0.667	0.779	0.597
<i>sepLayerGNN</i>	0.622	0.804	0.250	0.731	0.637	0.606	0.667	0.500	0.494	0.911	0.643	0.729	0.333	0.589	0.750	0.600
<i>combineEndGNN</i>	0.704	0.861	0.319	0.741	0.708	0.658	0.707	0.682	0.558	0.911	0.715	0.777	0.435	0.637	0.797	0.661
<i>sepLayerCombineEndGNN</i>	0.642	0.821	0.236	0.709	0.685	0.613	0.634	0.591	0.506	0.893	0.656	0.716	0.338	0.591	0.775	0.605

- Models obtained from changing *simpleGNN*: To experiment if handling *new-node-graph* input differently gives us an increase in performance, models *sepLayerGNN*, *combineEnd* and *sepLayerCombineEndGNN* are used. We can see that there is significant increase in the classifiers which handle new node features separately. The model *combineEnd* performed better than *sepLayerGNN* in class specific scores as well as combined scores. Then, we have experimented with *sepLayerCombineEndGNN* which is obtained by mixing *sepLayerGNN* and *sepLayerGNN* models. There are slight increases on combined scores when we compare *sepLayerCombineEndGNN* and *sepLayerGNN*. But, when we compare *sepLayerCombineEndGNN* and *combineEND* models, class specific as well as combined scores are greater in the latter.

4.6 Comparison

In this section we are going to compare methods we have defined as well as method used in the work [1]. First we are comparing the fusion network with *combineEndGNN* trained with *new-node-graph*. Then, a comparison of our model and method from [1] is presented.

Table 4.4: Confusion matrix for the fusion network when *roi-feature*, *subgraph-feature* and *graph-feature* used to train.

		Predicted			
		<i>benign</i>	<i>atypia</i>	<i>in situ</i>	<i>invasive</i>
Reference	<i>benign</i>	95	13	12	3
	<i>atypia</i>	5	12	3	2
	<i>in situ</i>	11	12	50	4
	<i>invasive</i>	0	4	0	52

Table 4.5: Confusion matrix for the model *combineEndGNN* when trained with *new-node-graph*.

		Predicted			
		<i>benign</i>	<i>atypia</i>	<i>in situ</i>	<i>invasive</i>
Reference	<i>benign</i>	87	11	14	1
	<i>atypia</i>	5	15	1	1
	<i>in situ</i>	4	21	43	9
	<i>invasive</i>	5	0	0	51

4.6.1 Fusion network vs *combineEndGNN*

Comparing fusion network trained with three sources and *combineEndGNN* may not be fair, since the former has more input sources. But it can give us an idea about the effect of embedding subgraphs into graph. In all of the combined scores, the fusion network has slightly greater values as expected, since it uses a feature that is not used in the *combineEndGNN* model. If we compare fusion network trained with *subgraph-feature* and *graph-feature*, with the *combineEndGNN* trained with *new-node-graph*, we can see that the latter outperforms in all of the metrics. Both of the mentioned cases use the same features at the beginning. So, we can say that the *combineEndGNN* model successfully processes subgraph embedded graphs. If we try to see where the best models from two different experiments struggled to classify, we can use confusion matrices in Tables 4.4 and 4.5. There we can see that most of the mistakes have been done when models predict the label *atypia*. Due to less number of samples in that class, class precision values have huge differences to precision values of the other classes. To investigate whether models struggle to classify *atypia* samples because of its nature or less number of samples in the data set, future work can be done by collecting more samples from that class and doing the mentioned experiments.

4.6.2 Our work vs two stage models

The model we are comparing is taken from work [1]. They are also constructing graphs from ROI samples where distance threshold is used for graph edges. As a feature extractor, they use efficientnet-b0 [56] architecture and train it with the patches by using ROI-level labels. Then, two graph convolutional layers followed by mean and softmax operations. For the convolutional layers, they use three different types and they are GCN, GAT and Attention-based Graph Neural Network (AGNN) [57] and we refer to them as *2-stage-GCN*, *2-stage-GAT* and *2-stage-AGNN* respectively. The same g-mean, precision, recall and f-measure scores can be found in Table 4.6 for the same data set with the same training, validation and test sets.

As we can see from Table 4.6, for the g-mean, precision and recall there are models better than fusion network in their work. On the other hand, for the f-measure score fusion network outperforms their models. But, the differences are not huge for both cases. So, subgraph mining can be used as a part of the model to classify ROI samples while having similar results to results of the models used to classify ROI samples. As we can see from the same table, precision scores of the class *atypia* is smaller than 0.250 while the *2-stage-GCN* have the smallest recall score for the same class which is 0.591. Also, their models struggle to classify *atypia* samples which confirms the difficulty of classifying class *atypia* by using our data set.

Table 4.6: G-mean, precision, recall and f-measure scores for the models from Aygunes’s work [1]. B, A, IS, INV and Avg. are used to represent *benign*, *atypia*, *in situ*, *invasive* and average respectively. The table contains scores for *2-stage-GCN*, *2-stage-GAT* and *2-stage-AGNN* models. Precision, recall and f-measure are calculated by macro averaging and g-mean is the geometric mean of class recalls.

Model	Gmean	Precision					Recall					F-measure				
		B	A	IS	INV	Avg.	B	A	IS	INV	Avg.	B	A	IS	INV	Avg.
2-stage-GCN	0.694	0.820	0.098	0.639	0.817	0.593	0.582	0.265	0.495	0.875	0.554	0.680	0.173	0.558	0.845	0.556
2-stage-GAT	0.724	0.895	0.233	0.769	0.907	0.701	0.626	0.773	0.649	0.875	0.731	0.737	0.358	0.704	0.891	0.672
2-stage-AGNN	0.711	0.847	0.226	0.837	0.941	0.713	0.585	0.955	0.532	0.857	0.732	0.692	0.365	0.651	0.897	0.651

Chapter 5

Conclusion

We have studied the use of subgraph mining in histopathology image classification. Most of the works in the literature uses graph convolutional networks for the classification tasks of this field due to its powers. We tried to use subgraph mining output alongside the graphs to solve the classification task. We have used the gSpan mining algorithm on our data set to find the frequent subgraphs. First, we tried to combine features from different sources where one of the input sources is frequent subgraph features. Then, we defined new models which process a graph where frequent subgraphs have been embedded into it. We have compared our models to the work that uses the same data set that we have used and saw similar classifying performances. As a result we can say that, applying subgraph mining alongside graph convolutional networks is one of the methods that can be used to classify histopathology images. We saw that our model, as well as models from other work, did not perform well on classifying samples from *atypia* class. It could be because of a smaller number of samples for that class in our data set. As a future work, collecting more samples for that class and doing the experiments again can be done to learn if it is difficult to classify or struggling to classify is related to class imbalance.

Bibliography

- [1] B. Aygüneş, “Weakly supervised approaches for image classification in remote sensing and medical image analysis,” Master’s thesis, Bilkent Üniversitesi (Turkey), 2020.
- [2] Z. Momenimovahed and H. Salehiniya, “Epidemiological characteristics of and risk factors for breast cancer in the world,” *Breast Cancer: Targets and Therapy*, vol. 11, p. 151, 2019.
- [3] D. Ahmedt-Aristizabal, M. A. Armin, S. Denman, C. Fookes, and L. Petersson, “A survey on graph-based deep learning for computational histopathology,” *Computerized Medical Imaging and Graphics*, p. 102027, 2021.
- [4] W. Mi, J. Li, Y. Guo, X. Ren, Z. Liang, T. Zhang, and H. Zou, “Deep learning-based multi-class classification of breast digital pathology images,” *Cancer Management and Research*, vol. 13, p. 4605, 2021.
- [5] Y. Feng, L. Zhang, and Z. Yi, “Breast cancer cell nuclei classification in histopathology images using deep neural networks,” *International Journal of Computer Assisted Radiology and Surgery*, vol. 13, no. 2, pp. 179–191, 2018.
- [6] 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*, pp. 770–778, 2016.
- [7] L. Van der Maaten and G. Hinton, “Visualizing data using t-sne.,” *Journal of Machine Learning Research*, vol. 9, no. 11, 2008.

- [8] R. Krithiga and P. Geetha, “Breast cancer detection, segmentation and classification on histopathology images analysis: a systematic review,” *Archives of Computational Methods in Engineering*, vol. 28, no. 4, pp. 2607–2619, 2021.
- [9] R. Saturi and P. Chand Parvataneni, “Histopathology breast cancer detection and classification using optimized superpixel clustering algorithm and support vector machine,” *Journal of The Institution of Engineers (India): Series B*, pp. 1–15, 2022.
- [10] L. Studer, J. Wallau, H. Dawson, I. Zlobec, and A. Fischer, “Classification of intestinal gland cell-graphs using graph neural networks,” in *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 3636–3643, IEEE, 2021.
- [11] J. Wang, R. J. Chen, M. Y. Lu, A. Baras, and F. Mahmood, “Weakly supervised prostate tma classification via graph convolutional networks,” in *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, pp. 239–243, IEEE, 2020.
- [12] M. Adnan, S. Kalra, and H. R. Tizhoosh, “Representation learning of histopathology images using graph neural networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 988–989, 2020.
- [13] E. W. Teh and G. W. Taylor, “Learning with less data via weakly labeled patch classification in digital pathology,” in *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, pp. 471–475, IEEE, 2020.
- [14] S. Ahmed, A. Shaikh, H. Alshahrani, A. Alghamdi, M. Alrizq, J. Baber, and M. Bakhtyar, “Transfer learning approach for classification of histopathology whole slide images,” *Sensors*, vol. 21, no. 16, p. 5361, 2021.
- [15] J. Fan, J. Lee, and Y. Lee, “A transfer learning architecture based on a support vector machine for histopathology image classification,” *Applied Sciences*, vol. 11, no. 14, p. 6380, 2021.

- [16] E. W. Teh and G. W. Taylor, “Learning with less labels in digital pathology via scribble supervision from natural images,” *arXiv preprint arXiv:2201.02627*, 2022.
- [17] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, “Graph neural networks: A review of methods and applications,” *AI Open*, vol. 1, pp. 57–81, 2020.
- [18] K.-H. N. Bui, J. Cho, and H. Yi, “Spatial-temporal graph neural network for traffic forecasting: An overview and open research issues,” *Applied Intelligence*, pp. 1–12, 2021.
- [19] C. Wang, Z. Lin, X. Yang, J. Sun, M. Yue, and C. Shahabi, “Hagen: Homophily-aware graph convolutional recurrent network for crime forecasting,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 4193–4200, 2022.
- [20] Y. Zheng, B. Jiang, J. Shi, H. Zhang, and F. Xie, “Encoding histopathological wsis using gnn for scalable diagnostically relevant regions retrieval,” in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 550–558, Springer, 2019.
- [21] P. Pati, G. Jaume, L. A. Fernandes, A. Foncubierta-Rodríguez, F. Feroce, A. M. Anniciello, G. Scognamiglio, N. Brancati, D. Riccio, M. D. Bonito, *et al.*, “Hact-net: A hierarchical cell-to-tissue graph neural network for histopathological image classification,” in *Uncertainty for Safe Utilization of Machine Learning in Medical Imaging, and Graphs in Biomedical Image Analysis*, pp. 208–219, Springer, 2020.
- [22] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4700–4708, 2017.
- [23] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

- [24] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *arXiv preprint arXiv:1609.02907*, 2016.
- [25] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” *arXiv preprint arXiv:1710.10903*, 2017.
- [26] J. Lee, I. Lee, and J. Kang, “Self-attention graph pooling,” in *International Conference on Machine Learning*, pp. 3734–3743, PMLR, 2019.
- [27] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, “Hierarchical graph representation learning with differentiable pooling,” *Advances in Neural Information Processing Systems*, vol. 31, 2018.
- [28] G. Jaume, P. Pati, A. Foncubierta-Rodriguez, F. Feroce, G. Scognamiglio, A. M. Anniciello, J.-P. Thiran, O. Goksel, and M. Gabrani, “Towards explainable graph representations in digital pathology,” *arXiv preprint arXiv:2007.00311*, 2020.
- [29] D. Anand, S. Gadiya, and A. Sethi, “Histograms: graphs in histopathology,” in *Medical Imaging 2020: Digital Pathology*, vol. 11320, pp. 150–155, SPIE, 2020.
- [30] W. Lu, S. Graham, M. Bilal, N. Rajpoot, and F. Minhas, “Capturing cellular topology in multi-gigapixel pathology images,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 260–261, 2020.
- [31] A. Raju, J. Yao, M. M. Haq, J. Jonnagaddala, and J. Huang, “Graph attention multi-instance learning for accurate colorectal cancer staging,” in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 529–539, Springer, 2020.
- [32] H. Sharma, N. Zerbe, S. Lohmann, K. Kayser, O. Hellwich, and P. Hufnagl, “A review of graph-based methods for image analysis in digital histopathology,” *Diagnostic pathology*, vol. 1, no. 1, 2015.
- [33] Z. Lou, J. You, C. Wen, A. Canedo, J. Leskovec, *et al.*, “Neural subgraph matching,” *arXiv preprint arXiv:2007.03092*, 2020.

- [34] I. Takigawa and H. Mamitsuka, “Graph mining: procedure, application to drug discovery and recent advances,” *Drug Discovery Today*, vol. 18, no. 1-2, pp. 50–57, 2013.
- [35] M. Elseidy, E. Abdelhamid, S. Skiadopoulos, and P. Kalnis, “Grami: Frequent subgraph and pattern mining in a single large graph,” *Proceedings of the VLDB Endowment*, vol. 7, no. 7, pp. 517–528, 2014.
- [36] C. Borgelt and M. R. Berthold, “Mining molecular fragments: Finding relevant substructures of molecules,” in *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pp. 51–58, IEEE, 2002.
- [37] J. Huan, W. Wang, J. Prins, and J. Yang, “Spin: mining maximal frequent subgraphs from graph databases,” in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 581–586, 2004.
- [38] X. Yan and J. Han, “gspan: Graph-based substructure pattern mining,” in *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pp. 721–724, IEEE, 2002.
- [39] N. Vanetik, E. Gudes, and S. E. Shimony, “Computing frequent graph patterns from semistructured data,” in *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pp. 458–465, IEEE, 2002.
- [40] L. B. Holder, D. J. Cook, S. Djoko, *et al.*, “Substructure discovery in the subdue system,” in *KDD workshop*, pp. 169–180, Washington, DC, USA, 1994.
- [41] F. Qiao, X. Zhang, P. Li, Z. Ding, S. Jia, and H. Wang, “A parallel approach for frequent subgraph mining in a single large graph using spark,” *Applied Sciences*, vol. 8, no. 2, p. 230, 2018.
- [42] C. Jiang, F. Coenen, and M. Zito, “A survey of frequent subgraph mining algorithms,” *The Knowledge Engineering Review*, vol. 28, no. 1, pp. 75–105, 2013.

- [43] M. Kuramochi and G. Karypis, “Finding frequent patterns in a large sparse graph,” *Data Mining and Knowledge Discovery*, vol. 11, no. 3, pp. 243–271, 2005.
- [44] M. Fiedler and C. Borgelt, “Support computation for mining frequent subgraphs in a single graph,” in *MLG*, Citeseer, 2007.
- [45] B. Bringmann and S. Nijssen, “What is frequent in a single graph?,” in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 858–863, Springer, 2008.
- [46] G. Aresta, T. Araújo, S. Kwok, S. S. Chennamsetty, M. Safwan, V. Alex, B. Marami, M. Prastawa, M. Chan, M. Donovan, *et al.*, “Bach: Grand challenge on breast cancer histology images,” *Medical image analysis*, vol. 56, pp. 122–139, 2019.
- [47] S. Graham, Q. D. Vu, S. E. A. Raza, A. Azam, Y. W. Tsang, J. T. Kwak, and N. Rajpoot, “Hover-net: Simultaneous segmentation and classification of nuclei in multi-tissue histology images,” *Medical Image Analysis*, vol. 58, p. 101563, 2019.
- [48] G. Csurka and F. Perronnin, “Fisher vectors: Beyond bag-of-visual-words image representations,” in *International Conference on Computer Vision, Imaging and Computer Graphics*, pp. 28–42, Springer, 2010.
- [49] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman, “The devil is in the details: an evaluation of recent feature encoding methods,” *BMVC*, volume 2, 01 2011.
- [50] A. Vedaldi and B. Fulkerson, “Vlfeat: An open and portable library of computer vision algorithms.” <http://www.vlfeat.org/>, 2008.
- [51] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, “Efficient object localization using convolutional networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 648–656, 2015.
- [52] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos,

- D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [53] P. J. Rousseeuw, “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis,” *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.
- [54] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” in *NIPS-W*, 2017.
- [55] M. Fey and J. E. Lenssen, “Fast graph representation learning with PyTorch Geometric,” in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.
- [56] M. Tan and Q. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” in *International conference on machine learning*, pp. 6105–6114, PMLR, 2019.
- [57] K. K. Thekumparampil, C. Wang, S. Oh, and L.-J. Li, “Attention-based graph neural network for semi-supervised learning,” *arXiv preprint arXiv:1803.03735*, 2018.