Ensemble Graph Neural Networks on Melanoma and Cervical Cancer Screening Datasets using SLIC Superpixels

Negin Ashouri

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Computer Science at

Concordia University

Montréal, Québec, Canada

June 2021

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Negin Ashouri**

Entitled: **Ensemble Graph Neural Networks**
on Melanoma and Cervical Cancer Screening Datasets
using SLIC Superpixels

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

TRISTAN GLATARD ______________ Chair
Dr. Tristan Glatard

TRISTAN GLATARD ______________ Examiner
Dr. Tristan Glatard

MARTA KERSTEN-OERTEL ______________ Examiner
Dr. Marta Kersten-Oertel

THOMAS FEVENS ______________ Supervisor
Dr. Thomas Fevens

Approved by LATA NARAYANAN ______________
Dr. Lata Narayanan, Chair
Department of Computer Science and Software Engineering

24/06/ 2021

MOURAD DEBBABI ______________
Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

Ensemble Graph Neural Networks
on Melanoma and Cervical Cancer Screening Datasets
using SLIC Superpixels

Negin Ashouri

Graph neural networks (GNNs) have become the standard procedure to deal with graph-structured data and data in non-Euclidean spaces. Since 2017, numerous researchers have been using GNN models in their experiments. However, despite GNNs' recent rapid growth, there are not yet many real-world applications that benefit from these models. In this thesis, we use GNNs as image classifiers. To improve the efficiency and reduce the complexity of the models, we first generate graphs from the images by creating superpixels of the images and use them as our graph nodes instead of individual pixels. Then, we define edges on these nodes based on the distance of each superpixel to their closest ones. We propose two ensemble frameworks containing a pre-trained ResNet18 and two Graph Neural Network (GNN) models (GAT and GIN). We call these frameworks GATRes and GATGIN, respectively.

We test these frameworks on two real-world medical applications: Cervical Cancer Screening and Melanoma detection. Cervical cancer is among the top four common cancer in women worldwide. Cervical cancer can be easily prevented if caught in its pre-cancerous stage. Determining the appropriate treatment method depends on patients' physiological differences. A treatment that works effectively for one woman may obscure future cancerous growth in another woman due to differences in the type of their cervix. In this thesis, we experiment with multiple GNNs on this dataset to distinguish the cervix types and examine if these models enhance detection performance and accuracy.

The other problem we consider is Melanoma skin cancer. Melanoma is the most lethal skin cancer. If melanoma gets diagnosed in the early stages, the patients' survival rate will increase significantly. This research applies GNN models on the Melanoma dataset to discriminate between melanoma and benign skin lesions.

We show that our models' sensitivity and accuracy outperform the individual models in our classification tasks. Our GATRes model also outperforms the accuracies achieved by previously published papers on the Cervical Cancer Screening dataset.

Acknowledgments

First of all, I would like to take the opportunity to convey my sincere appreciation towards my supervisor Dr. Thomas Fevens who really guide me in the field of AI applications in Medicine. I Truly appreciate his constant support and patience during my master's study both towards my thesis and my startup journey. I have been very fortunate to have a great supervisor like him.

Secondly, I would like to thank all the people who helped and supported me during my master's studies. My teammates in my lab and my colleagues at my startup, FemTherapeutics.

Next, I would like to thank the Surgical Innovation program, the cross-disciplinary graduate program from McGill, ETS and Concordia funded by NSERC CREATE. It provided me with a unique opportunity to learn about the surgical field as well as business and start my own startup that gave me a very broad and unique insight and experience in my life. It also provided me with financial support which without, I couldn't finish my masters.

Furthermore, I want to extend my thanks to my friends for their supports, my sister, Negar, and my parents, for their love and encouragement and finally my better-half, Emad, for his sacrifices, selflessness and encouragements through years of my master's studies.

I also would like to thank the Examining Committee for reviewing the previous edition of the thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we will give a brief introduction to this thesis. First, we will describe the motivation of the thesis and the contributions, and then we give the outline of this thesis.[1]

1.1 Introduction and Motivation

Graph Representation Learning is a fairly new domain in machine learning that has become popularly in use since 2017. It helps to understand and solve machine learning problems on data from not-Euclidean spaces and is graph-structured in nature. The most common tasks on graphs are node classification and visualization, link prediction and graph classification. Although much research is being conducted on this domain, not many graph networks are implemented in real-world applications. Developing powerful and theoretically expressive Graph Neural Network (GNN) architectures is a key concern towards practical applications and real-world adoption of graph machine learning. However, tracking recent progress has been challenging as most models are evaluated on small datasets[7]. In order to tackle this problem, Dwivedi *et al.* newly published [7] in June 2020. This paper benchmarks GNNs to identify and quantify what types of architectures, first principles or mechanisms are universal, generalizable, and scalable when moving to larger and more challenging datasets. One of their experiments includes using graphs as image classifiers on MNIST and

[1] Warning: this paper has graphics containing blood that some people might find disturbing.

CIFAR10. They provide a reproducible GNN benchmarking framework [2], with the facility for researchers to add new models for arbitrary datasets. In this thesis, with leveraging from their framework, we implement GNN models on our two medical applications. Cervical Cancer Screening (hereafter ”Cervix” dataset) and Melanoma Skin Cancer (hereafter ”Melanoma” dataset) obtained from Cervical Cancer Screening Challenge [23] and the balanced version of dermoscopic images from the International Skin Imaging Collaboration (ISIC) archive [3] [24], respectively.

We firstly implemented simple linear iterative clustering (SLIC) superpixel algorithm[1] on our original images. Super-pixels represent small regions of homogeneous intensity in images, and can be extracted with the SLIC technique[1][7]. Superpixel algorithms cluster a group of pixels together in order to increase computation efficiency. Then, we generate graphs from these obtained superpixels. The nodes in these graphs will be the center coordinates of the superpixels, and each node's features are the superpixel coordinates and intensity. The edges will be the distance of neighboring superpixels. Using superpixels instead of individual pixels decreases the computational expense. Afterward, we implemented GNN models on these graphs. The models used are GraphSage, Gated GCN, Graph Attention (GAT) and Graph Isomorphism Network (GIN)). GAT and GIN had shown better performance on accuracy.

In the second step, we propose two ensemble frameworks, with the first one being a combination of a pre-trained Residual Network ResNet18 (with conducting transfer learning on ResNet18) and GAT (we call it GATRes network) and the second one being the combination of the GAT and GIN networks, called GATGIN. We concatenate the feature extractions done by each model in this framework and then feed them into a fully connected classifier for the final classification. Our ensemble models outperform all the individual models on these classification tasks. Also, our GATRes model outperforms all the previous published accuracies on the Cervix dataset.

1.2 Contribution

The objectives and contributions of this work are twofold that include:

1) Generation of a superpixel graph representation of images in two real-world medical datasets:

[2] https://github.com/graphdeeplearning/benchmarking-gnns

[3] https://www.kaggle.com/fanconic/skin-cancer-malignant-vs-benign

Melanoma and Cervix and implementation of four GNN models on them. In one, GNN models have never been implemented.

2) Proposing two ensemble framework containing:

    (a) Two graph neural networks (GAT and GIN) and named it GATGIN

    (b) A pre-trained ResNet18 and a graph neural network (GAT) and named it GATRes

Our framework also prepares any color images to be fed into graph networks.

We transformed these dataset images into a graph of superpixels and then fed them to our ensemble models.

1.3 Thesis Outline

This thesis is organized as follows: A comprehensive literature review on image classification methods, superpixel image segmentation, and graphs for classification are presented in Chapter 2. Chapter 3 will present the methodology and the framework that has been proposed in the experiments. Chapter 4 will present the setup, experiments performed, and corresponding results. In Chapter 5, we will conclude this thesis and discuss some future work and research directions.

Chapter 2

Background

In this chapter, we will cover related background works of this thesis. First of all, we will review the image classification methods currently being used in medical applications such as ResNet. Then we will explain superpixel methods and how they are used to improve the outcome of classification tasks and highlighting the superpixel method used in this thesis (SLIC). After that, we will review graph representation learning and algorithms and how they can be used in image classification tasks. Also, we will review ensemble methods and finally, we will explain the Graph models being used in this thesis.

2.1 Image Classification

Image classification, one of the most fundamental tasks in computer vision and pattern recognition, assigns one or more labels to an image. Classic machine learning approaches for image classification use low-level or mid-level features to represent the image, and then the label will be assigned by a trainable classifier. The deep learning approach combines feature extraction and classification tasks and estimates high-level features, which significantly outperforms hand-crafted low-level and mid-level features. These techniques have also been applied to medical image classification and computer-aided diagnosis [65].

2.1.1 Medical Image Classification

Skilled medical professionals use various imaging techniques to capture anomalies of the human body. These images will be used for diagnosis, prognosis and treatment planning. Captured anomalies are usually understood and annotated by these clinicians. However, the lack of availability of human experts and the their exhaustion along with the difficulty involved with this procedure limit the effectiveness of image understanding[50].

Medical image classification aims to help medical professionals with diagnosis, analysis and educational purposes. The primary objective of medical image classification is to achieve good accuracy and understand which parts of the anatomy are affected by the disease to help clinicians in early diagnosis and in learning the progression of a disease[30].

Convolutional neural networks (CNNs) have reached state-of-the-art performance in image understanding. In many cases, they even outperformed medical professionals.[50]. Convolutional Neural Network (CNN) model automatically learns the needed features and extracts them for medical image understanding. The CNN model comprises convolutional filters whose primary function is to learn and extract necessary features for efficient medical image understanding. CNN started gaining popularity in 2012 due to AlexNet [29], which outperformed all the others models in the imageNet challenge 2012. Later on, GoogleNet reached an accuracy of 89% in cancer detection while human pathologists could only achieve the accuracy of 70%[50, 44]. The following paragraph describes how CNN works.

CNN

CNNs consist of a series of neurons that have learnable weights and biases. Each neuron receives inputs, performs a dot product and will be followed with a non-linearity. It consists of three layers:

- **Convolution Layer (Kernel)** is used to extract features.The Convolution operation is used to extracts the high-level features from the input image. The first Convolution layer is responsible for understanding the Low-Level features such as edges, colors and many more. With layers being added, the architecture starts to understand the High-Level features, and it ends

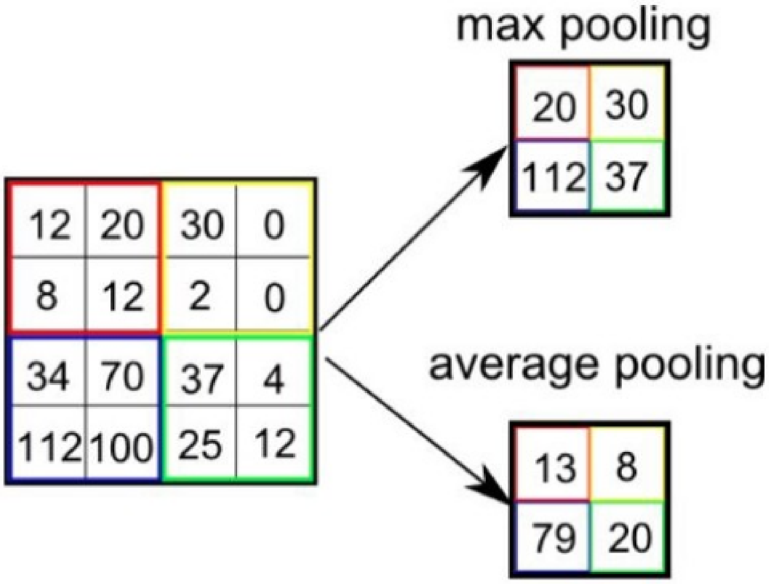


Figure 2.1: Types of pooling [48]

up with a broad comprehension of the whole image.

- **Pooling Layer**. The Pooling layer reduces the dimensionality to optimize the computational power. It achieves this goal by decreasing the size of the obtained features from its previous layer. Two types of Pooling consist of: Max Pooling and Average Pooling. Max Pooling calculates the maximum value from the chunk of the convolved features, and Average Pooling calculates the average of all the values from that chunk. (Figure 2.1).

  The Convolutional Layer and the Pooling Layer will be stacked together to form any multi-layer CNN we want to create. The final step is to flatten the output and feed it to a fully connected network for the final classification.

- **Classification — Fully Connected Layer (FC Layer).** The FC Layer will learn the non-linear combination of the captured high-level features from the convolution layer. (Figure 2.2).

In the following paragraph, we will review ResNet (Residual Network), one of the most popular models of CNN, that has been used in the previous works on the datasets we will be using in our

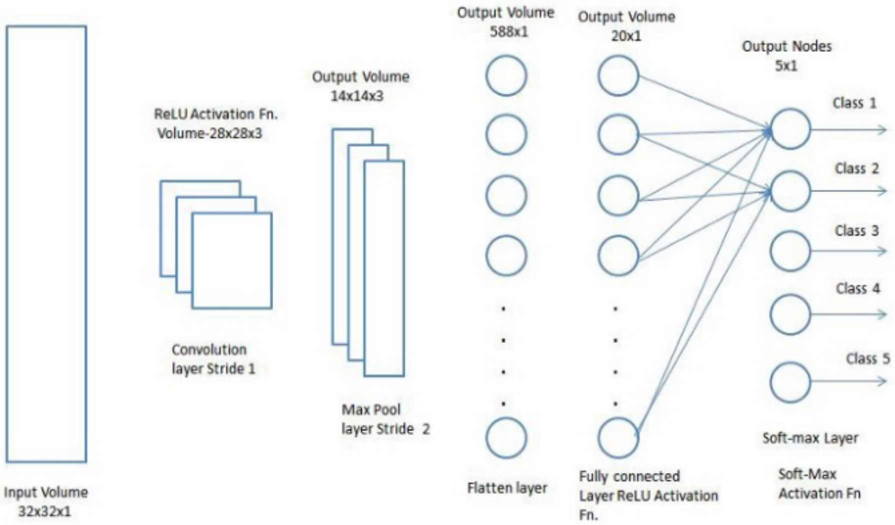


Figure 2.2: Fully Connected Layer [48]

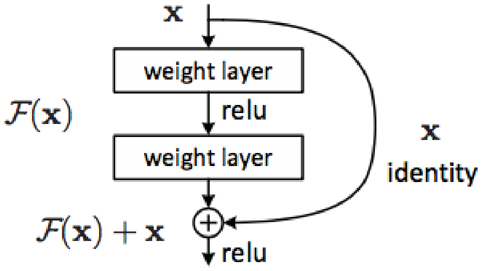

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.3: A building block of Residual network [18].

thesis.

ResNet(Residual Network)[18]

In Figure 2.3 you can see a residual block. The residual network consists of multiple of these residual blocks stacked together. You can see a Shortcut connection [5, 45, 61] in the picture. These shortcut connections skip one or multiple layers showed by F(x)+x, which in this case will be an identity mapping, and their outputs are added to the outputs of the stacked layers (Fig. 2.3). This architecture prevents the gradient from vanishing. In deep neural network architectures, the gradient usually becomes very small as the gradient gets back-propagated to earlier layers. ResNet solves this problem with its shortcut connection architecture.

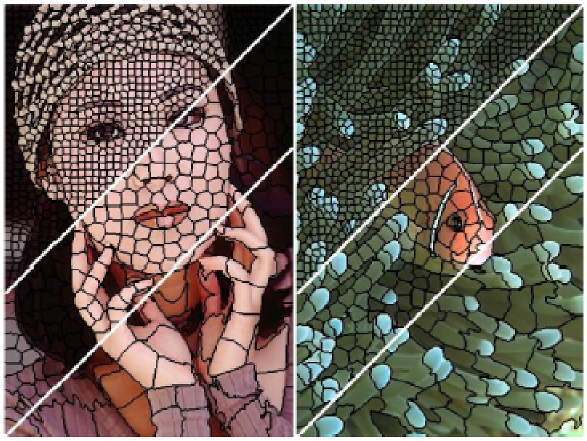


Figure 2.4: Images segmented using SLIC into superpixels of size 64, 256, and 1,024 pixels (approximately) [1].

2.2 Superpixel Image Segmentation

Medical image segmentation is one of the critical steps in medical image processing and analysis. The purpose of medical image segmentation is to divide an image into multiple non-overlapping regions based on some criterion or rules such as similar color, texture etc. Based on various traditional techniques, many researchers proposed a great number of automated segmentation approaches such as thresholding, edge detection, active contours and so on [49, 52].

Superpixel algorithms cluster a group of pixels together in order to increase computation efficiency. Achanta *et al.* [1] explain how superpixel algorithms group pixels into bigger regions, which can replace the stiff structure of the pixel grid (Figure 2.4). By capturing the image similarities in each region, superpixels increase computational efficiency. They are now being used in many applications including depth estimation [71], segmentation [32], etc.

There are different methods for creating superpixels which can be divided into two categories. Graph-based and Gradient-Ascent-based methods.

Graph-Based Algorithms

Graph-based algorithms set each pixel as a node of a graph. Then the similarity of adjacent pixels creates the edge weights between two nodes. Superpixels are generated by minimizing a cost function defined over this graph[1]. The following algorithms are some examples of this method.

NC05 The Normalized cuts algorithm [22] uses contours on images and repeatedly divides the image into graphs of all pixels. It produces very regular superpixels, but the boundary adherence of NC05 is poorer and slower than other methods. NC05 has a complexity of $O\left(N^{\frac{3}{2}}\right)$ [31], wherein N is the number of pixels.

GS04 Felzenszwalb and Huttenlocher [10] propose a hierarchical clustering of pixels as nodes on a graph and create superpixels in a way that each superpixel is the lowest spanning tree from the base pixels. GS04 holds well to the image's boundaries, but the superpixels it makes are irregular in size and shape. However, this method is fast, but we can not control the number of superpixels or the compactness. It has $O(N \log N)$ complexity. Also, **SL08** from Moore *et al.* [37], which strongly influence the quality and speed of the output, and **GCa10 and GCb10** Veksler *et al.* [59] are other example methods of this approach.

Gradient-Ascent-Based Algorithms

Gradient-ascent-based algorithms start from an initial clustering of pixels and reiterate through them until it meets the defined criteria for creating superpixels [1].

One of the examples of this approach is WS91 [62], watershed approach, performs a gradient ascent starting from local minima to produce watersheds, metaphoric to a geological watershed, which divides adjacent drainage basins. The superpixels made by this method are irregular in shape and do not adhere to the image boundaries. Although it is a fast algorithm (($O(N \log N)$ complexity), the compactness and the number of segments of the superpixels cannot be controlled.

SLIC superpixel

Achanta *et al.* [1] propose a new method of generating superpixels called SLIC (simple linear iterative clustering). The authors compared the methods mentioned above, their advantages and disadvantages, and provide useful information on how these models are unsatisfactory. The authors show how SLIC is improving performance in segmentation tasks. Further, they explain that each of these approaches may be better suited for certain applications. The authors also believe that although it would be difficult to find a perfect method that applies to all applications, if the generated superpixels adhere well to the images' boundaries, be fast to compute and provide control over its hyper-parameters such as compactness and number of segments, they will perform better and improve the results. Compactness is a parameter that controls the regularity in shape and size and the smoothness of the superpixels' boundaries [51].

On their paper, Achanta *et al.* [1] compared five state-of-the-art superpixel methods [58, 10, 22, 59, 31] on their speed, ability to adhere to image boundaries, and impact on segmentation performance. The authors state that none of these methods were satisfactory enough in all sense. Therefore, they proposed a new superpixel algorithm which is called simple linear iterative clustering (SLIC). This method uses k-means clustering to generate superpixels similar to [70]. The authors show that SLIC outperforms the mentioned state-of-the-art methods in terms of boundary adherence on the Berkeley benchmark [35] and also outperforms existing methods when used for segmentation on the PASCAL [9], and MSRC [53] data sets, which contain RGB images normally used for object detection and gesture recognition tasks. SLIC is also faster and more computationally efficient than those previous methods. Furthermore, SLIC offers control over the compactness and the number of segments of superpixels [1]. SLIC's algorithm is shown in Figure 2.5.

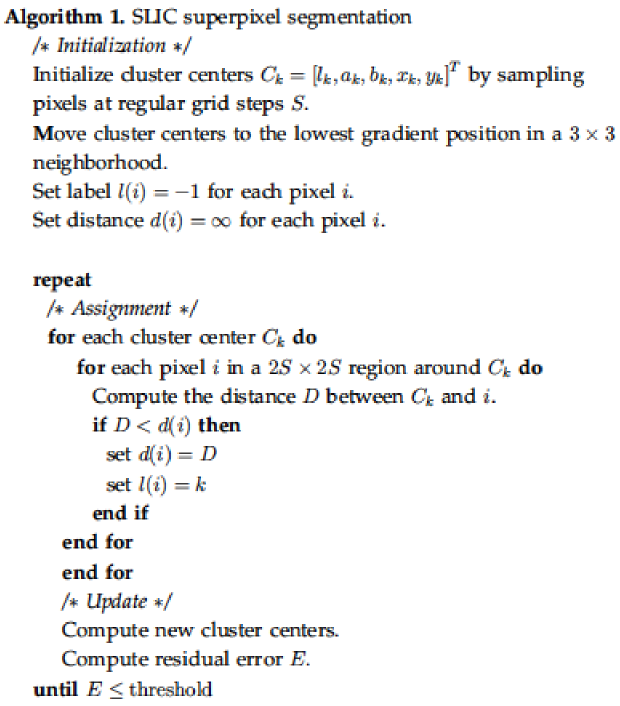
Our approach, in this thesis, is to use the SLIC superpixels, not for segmentation but to benefit from it to reduce the complexity of our classification task. Therefore, we did not review all the image segmentation methods as a part of the thesis background information.

2.3 Graphs

This section will talk about graph representation learning, machine learning on graphs, and, finally, graphs for image classification.

2.3.1 Graph Representation Learning

A graph G = (V, $\mathcal{E}$) is generally defined by a set of nodes V and a set of edges $\mathcal{E}$ between these nodes. We indicate an edge going from node $u \in V$ to node $v \in V$ as $(u, v) \in \mathcal{E}$. Usually, we deal with simple graphs with only one edge between each node, and there are no self loops on the nodes. $(u, v) \in \mathcal{E} \leftrightarrow (v, u) \in \mathcal{E}$. We usually represent graphs by an adjacency matrix $A \in \mathbb{R}^{|V| \times |V|}$.



```
Algorithm 1. SLIC superpixel segmentation
  /* Initialization */
  Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling
  pixels at regular grid steps $S$.
  Move cluster centers to the lowest gradient position in a $3 \times 3$
  neighborhood.
  Set label $l(i) = -1$ for each pixel $i$.
  Set distance $d(i) = \infty$ for each pixel $i$.

  repeat
    /* Assignment */
    for each cluster center $C_k$ do
      for each pixel $i$ in a $2S \times 2S$ region around $C_k$ do
        Compute the distance $D$ between $C_k$ and $i$.
        if $D < d(i)$ then
          set $d(i) = D$
          set $l(i) = k$
        end if
      end for
    end for
    /* Update */
    Compute new cluster centers.
    Compute residual error $E$.
  until $E \leq$ threshold
```

Figure 2.5: SLIC Segmentation Algorithm [1]

To represent a graph with an adjacency matrix, the nodes will be vertices on the column/row and elements of the matrix and then we indicate whether pairs of nodes are connected by an edge or not. The matrix will be: A[u, v] = 1 if $(u, v) \in \mathcal{E}$ and A[u, v] = 0, otherwise [15].

2.3.2 Machine Learning on Graphs

Many of the real-world data around us have graph structures in nature. From chemistry and drug architectures to social networks. However, these problems could not be solved with machine learning until years ago. Machine learning algorithms and deep learning models are only able to solve problems defined in a Euclidean space. The graph representation learning was introduced to solve the graph-structured problems and where data is in non-euclidean space. In machine learning, we seek to build models that can learn from data to solve particular tasks. Machine learning tasks are categorized into three types: Supervised, Unsupervised or Reinforcement Learning Tasks. Machine learning with graphs is the same, but the usual categories of supervised, unsupervised, and reinforcement learning are not necessarily the most informative for graphs [15].

Three main tasks implemented using machine learning on graph data are as follow:

1. Node classification

The goal is to predict the label y_u which can be a type, category, or attribute corresponded to all the nodes $u \in V$, when we are given the true labels on only a small training set of nodes $\mathcal{V}_{\text{train}} \subset \mathcal{V}$. Some of the instances of node classification are social networks and using hyperlinks or citation graphs for classifying the topic of documents [27], etc. Node classification may seem an easy task of supervised classification. However, one of the most important differences between this task and a classic supervised classification task is that nodes in a graph are not independent and identically distributed (i.i.d.). Usually, in classic supervised classification tasks, we assume that each datapoint is statistically independent of all the other data points and is identically distributed. In node classification, rather than modeling a set of i.i.d. data points, we are instead modeling set nodes interconnected in a graph structure [15].

2. Relation prediction

In this task, we want to predict the relations between the nodes when they are missing. This task is also called link prediction and graph completion. Some of the examples of this task include: predicting drug side-effects [72], recommending content to users in social platforms [69], etc. On this task, a set of nodes V and a partial set of edges between these nodes $\mathcal{E}_{\text{train}} \subset \mathcal{E}$ is presented. The goal is using this information to predict the missing edges $\mathcal{E} \backslash \mathcal{E}_{\text{train}}$ [15].

3. Clustering and community detection

Community detection is the graph analog of unsupervised clustering. We would expect this network to build a community structure, where nodes are more likely to create edges with nodes that belong to their same community. The challenge is doing this given only the input graph G = (V, $\mathcal{E}$) [15].

4. Graph classification (and clustering)

This task aims to learn over graph data and classify (or cluster) problems over entire graphs. For instance, predicting a molecule's toxicity or solubility when the structure of a molecule is given [11]. In these applications, we are given a dataset of multiple different graphs, and we seek to make independent predictions specific to each graph. Each graph is an i.i.d. data point associated with a label, and the goal is to predict the labels of unseen graphs [15].

The experiments in this thesis are graph classification tasks.

2.3.3 Graph Neural Networks and Graph Convolution Networks

This section discusses the graph neural network (GNN) framework, which defines deep neural networks on graph data. Representations of nodes are generated depending on the structure of the graph and any feature information there might be [15].

Neural Message Passing

GNNs use a form of neural message passing between the graph's nodes where vector messages interchanged between nodes in the graph and get updated through neural networks [15].

Overview of the Message Passing Framework :

GNNs perform iterative message passing among graph's nodes. During each message-passing iteration, the embedding $\mathbf{h}_u^{(k)}$ corresponding to each node $u \in V$ is updated according to information aggregated from $u's$ graph neighborhood $\mathcal{N}(u)$ (Figure 2.6). This message-passing update is:

$$\begin{aligned}\mathbf{h}_u^{(k+1)} &= \text{UPDATE}^{(t)}\left(\mathbf{h}_u^{(k)},\ \text{AGGREGATE}^{\ (t)}\left(\left\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\right\}\right)\right) \\ &= \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right)\end{aligned}$$

In this equation, UPDATE and AGGREGATE are arbitrary differentiable functions (i.e., neural networks) and $\mathbf{m}_{\mathcal{N}(u)}$ is the "message" that is aggregated from u's graph neighborhood $\mathcal{N}(u)$. Superscripts are used to distinguish the embeddings and functions at different message passing iterations (different "layers" of the GNN). At each iteration of the GNN, a set of embeddings of the nodes in u's graph neighborhood $\mathcal{N}(u)$ will be input to the AGGREGATE function and a message $\mathbf{m}_{\mathcal{N}(u)}$ will be generated based on this aggregated information. Then, the message $\mathbf{m}_{\mathcal{N}(u)}$ will be combined with the previous embedding $\mathbf{h}_u$ of node u by the update function UPDATE to generate the updated embedding. The embeddings at the starting point $k = 0$ are set to the input features for all the nodes, i.e., $\mathbf{h}_u^{(0)} = \mathbf{x}_u, \forall u \in \mathcal{V}$. After running K iterations, the output of the final layer will be used to define the embeddings for each node, i.e., $\mathbf{z}_u = \mathbf{h}_u^{(K)}, \forall u \in \mathcal{V}$. Also, since the AGGREGATE function takes a set as input, GNNs defined in this way are permutation equivariant by design [15].

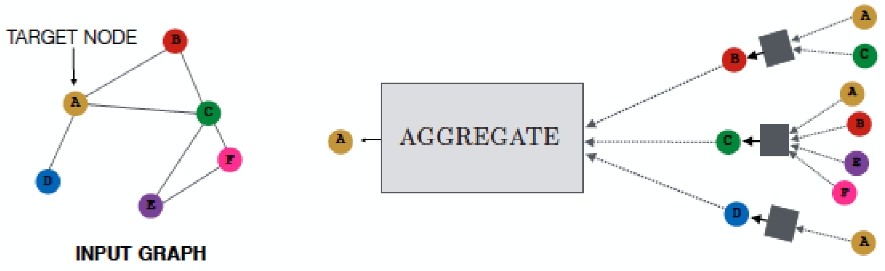


Figure 2.6: Message aggregation to the nodes from its adjacent neighbors. The model aggregates messages from A's neighbors (B, C, and D), and the messages coming from these neighbors are based on messages aggregated from their respective neighborhoods, and so on. This figure demonstrate a two-layer message-passing model.[15]

In summary, the GNN message-passing framework's intuition is that at each iteration, every node aggregates information from its adjacent nodes, and as iterations pass, each node embedding gains more information from further reaches of the graph. So, after the first iteration ($k = 1$), every node embedding gains the information from its 1-hop neighborhood, after the second iteration ($k = 2$), all node embeddings will have information from their 2-hop neighborhood and, after k iterations, they hold information about their k-hop neighborhood. This information can carry two main things. First is the structural information of the graph, such as degrees of all nodes throughout the iterations. The second form is that after t iterations, each node will also have information of all features of their t-hop neighbor nodes [15].

The Basic GNN

The following is the most basic form of the GNN framework. The basic GNN message passing can be expressed as:

$$\mathbf{h}_u^{(k)} = \sigma\left(\mathbf{W}_{\text{self}}^{(k)}\mathbf{h}_u^{(k-1)} + \mathbf{W}_{\text{neigh}}^{(k)}\sum_{v\in\mathcal{N}(u)}\mathbf{h}_v^{(k-1)} + \mathbf{b}^{(k)}\right)$$

where $\mathbf{W}_{\text{self}}^{(k)}, \mathbf{W}_{\text{neigh}}^{(k)} \in \mathbf{R}^{d^{(k)}\times d^{(k-1)}}$ are trainable parameter matrices and σ denotes an element-wise non-linearity (e.g., a tanh or ReLU). The bias term $\mathbf{b} \in \mathbf{R}^{d^{(k)}}$ is ommitted for notational simplicity but it is critical for achieving strong performance.

In the message passing in the basic GNN framework, the incoming messages from the neighbors are summed and then the neighborhood information will be combined with the nodes the previous embedding using a linear combination; and finally, an element-wise non-linearity will be applied.

UPDATE and AGGREGATE functions in the basic GNN are calculated through the:

$$\mathbf{m}_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \mathbf{h}_v$$
$$\text{UPDATE}\left(\mathbf{h}_u, \mathbf{m}_{\mathcal{N}(u)}\right) = \sigma\left(\mathbf{W}_{\text{self}}\, \mathbf{h}_u + \mathbf{W}_{\text{neigh}}\, \mathbf{m}_{\mathcal{N}(u)}\right)$$

The parameters $\mathbf{W}_{\text{self}}$, $\mathbf{W}_{\text{neigh}}$ and **b** can be either shared over the iterations or be trained separately [15].

2.3.4 Using GNN for image classification

There are many graph neural networks and graph convolutional networks that can be used for different applications. In a very recent publication, Dwivedi et al. [7] provided a benchmark for GNN models on different datasets for the first time. In this paper, they examined different GNN models on different datasets. One of their experiments was on MNIST and CIFAR10 Data. They used the SLIC superpixel method from [60]. They obtained their data from the same paper wherein they prepared the superpixeled files from the images. Then Dwivedi et al. [7] fed these files to their GNN models and reported the benchmarks on them. They have found that the SLIC superpixel is the best method among superpixel methods, they used the superpixeled dataset from the mentioned paper. Therefore, in this thesis, we will also examine the SLIC superpixel on our images, and we will also use the best-performed models that have been used in this paper. The following sections provide a brief description of each graph model we will be experimenting on and how they work:

GraphSage [14]

Low-dimensional embeddings of nodes in large graphs are useful in various prediction tasks. Hamilton *et al.* propose an inductive method that leverages node feature information (e.g., text attributes) to generate node embeddings for previously unseen data efficiently instead of needing all the nodes to be present in the training phase for embeddings to be generated. However, most of the current methods need all nodes in the graph to be present for training of the embeddings; these approaches are transductive and lack generalization to unseen data. This paper presents GraphSAGE,

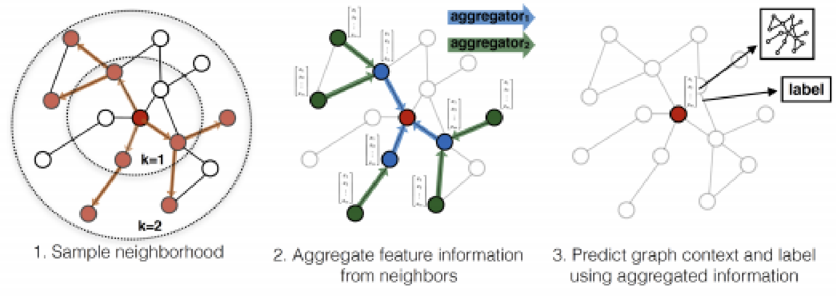

aggregator1
aggregator2
k=1
k=2
label
1. Sample neighborhood
2. Aggregate feature information from neighbors
3. Predict graph context and label using aggregated information

Figure 2.7: Visual illustration of the GraphSAGE sample and aggregate approach [14]

a general inductive framework that leverages node feature information (e.g., text attributes) to generate node embeddings for previously unseen data efficiently. Training individual embeddings for each node is computationally expensive especially for large graphs. Their alternative is to estimate a function that samples and aggregates features from a node's local neighborhood and generates embeddings (Figure 2.7). They propose a general framework, called GraphSAGE (SAmple and aggreGatE), for inductive node embedding. Adding node features helps the algorithm to learn the topological structure of each node's neighborhood as well as the distribution of node features. Their algorithm can also be applied to graphs without node features.

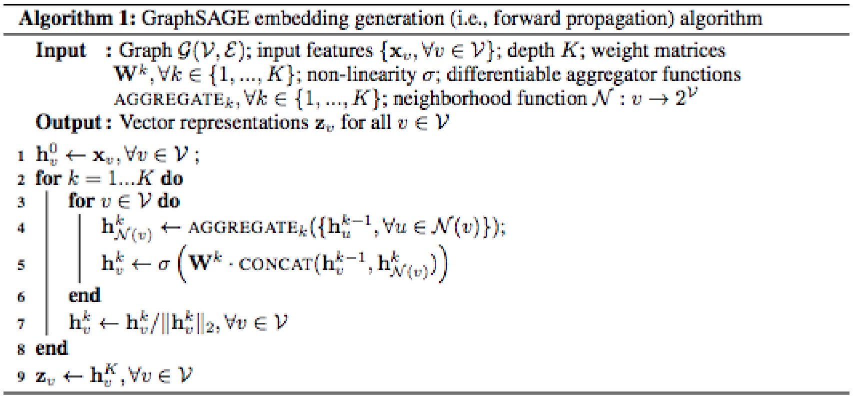
Each aggregator function aggregates information from different hops or search depths, away from a given node. They use their trained system to generate embeddings for entirely unseen nodes by applying the learned aggregation functions at the test or inference time. Algorithm 1 (Figure 2.8) is the algorithm of GraphSAGE generating embeddings.

The intuition behind Algorithm 1 is that at each iteration, or search depth, nodes aggregate information from their local neighbors. As this process iterates, nodes incrementally gain more and more information from further reaches of the graph. The GraphSage diagram is shown in Figure 2.9

GIN (Graph Isomorphism Network)[67]

Graph Isomorphism Networks (GIN)'s architecture is based the Weisfeiler-Lehman Isomorphism Test [66] to study the expressive power of GNNs. The node update equation is expressed as:

$$\hat{h}_i^{\ell+1} = (1+\epsilon)h_i^{\ell} + \textstyle\sum_{j\in\mathcal{N}_i} h_j^{\ell}$$
$$h_i^{\ell+1} = \mathrm{ReLU}\left(U^{\ell}\left(\mathrm{ReLU}\left(\mathrm{BN}\left(V^{\ell}\hat{h}_i^{\ell+1}\right)\right)\right)\right)$$



```
Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm
Input  : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth $K$; weight matrices
         $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity $\sigma$; differentiable aggregator functions
         $\textsc{aggregate}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$
Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$
$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$ ;
for $k = 1...K$ do
    for $v \in \mathcal{V}$ do
        $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \textsc{aggregate}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
        $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \textsc{concat}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
    end
    $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$
```

Figure 2.8: GraphSAGE Algorithm[14]

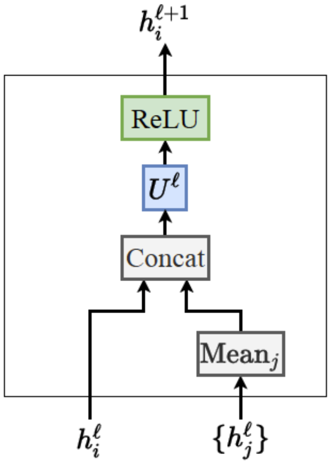


Figure 2.9: GraphSAGE Layer [7]

.

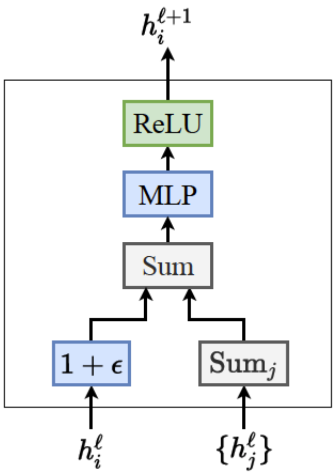


Figure 2.10: GIN Layer [7]

.

where ϵ is a learnable constant,$U^{\ell}, V^{\ell} \in \mathbb{R}^{d\times d}$ and BN denotes Batch Normalization [7]. You can find the GIN layer diagram in Figure 2.10.

GAT (Graph Attention Networks)

In [60], the Graph Attention Network (GAT) was developed, which consists of stacked graph attention building block layers. First, the authors explain the single graph attention layer. The input to this layer is a set of node features, $\mathbf{h} = \left\{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\right\}, \vec{h}_i \in \mathbb{R}^F$, where N is the number of nodes, and F is the number of features in each node. The layer creates a new set of node features (with possibly different degree of freedom F') $\mathbf{h}' = \left\{\vec{h}'_1, \vec{h}'_2, \ldots, \vec{h}'_N\right\}, \vec{h}'_i \in \mathbb{R}^{F'}$, as its output.

The network employs a multi-headed architecture to increase the learning capacity. The node update equation is given by:

$$h_i^{\ell+1} = \text{Concat}_{k=1}^{K}\left(\text{ELU}\left(\textstyle\sum_{i\in\mathcal{N}_1} e_{ij}^{k,\ell} U^{k,\ell} h_j^{\ell}\right)\right)$$

where $U^{k,\ell} \in \mathbb{R}^{\frac{d}{K}\times d}$ are the K linear projection heads, and $e_{ij}^{k,\ell}$ are the attention coefficients for each head defined as:

$$\hat{e}_{ij}^{k,\ell} = \text{LeakyReLU}\left(V^{k,\ell}\,\text{Concat}\left(U^{k,\ell}h_i^{\ell}, U^{k,\ell}h_j^{\ell}\right)\right)$$

$$e_{ij}^{k,\ell} = \frac{\exp\left(\hat{e}_{ij}^{k,\ell}\right)}{\sum_{j'\in\mathcal{N}_i}\exp\left(\hat{e}_{ij'}^{k,\ell}\right)}$$

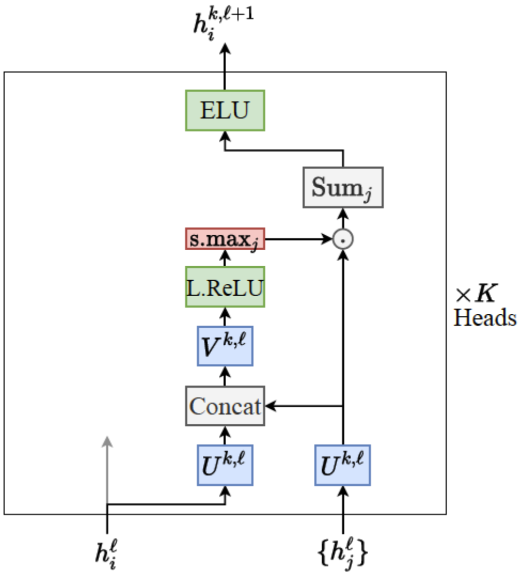


Figure 2.11: GAT Layer [7]

.

Model Name	CIFAR10	MNIST
GIN	55.25%	96.48%
GAT	64.22%	95.53%
GraphSAGE	65.76%	97.31%

Table 2.1: GraphSAGE, GAT and GIN accuracy on CIFAR10 and MNIST

where $V^{k,\ell} \in \mathbb{R}^{\frac{2d}{K}}$ see Figure 2.11. GAT learns a mean over each node's neighborhood features sparsely weighted by the importance of each neighbor [7].

Bechmark results on CIFAR10 and MNIST

In Table 2.1 you can find the accuracy of GraphSAGE, GAT and GIN networks on CIFAR10 and MNIST datasets. CFAR10 consists of 60000 32x32 color images in 10 classes, and MNIST is a dataset of 60,000 small square 28×28 pixel grayscale images of handwritten single digits between 0 and 9.

2.4 Transfer Learning

In machine learning, transfer learning is a technique where a model will be trained on one dataset for a specific task, and then this pre-trained model will be used as a starting point for the same task on a different dataset. Depending on the second task of interest, the model may require fine-tuned input, layers, and output.

2.5 Ensemble Models

In recent years ensemble learning has become the hot spot in the machine learning field for many researchers. The main idea of this method is to create multiple models and aggregate their outputs to improve the results. This approach brings in a unique advantage when dealing with small datasets, high dimensionality, and complexity in data structures. Its performance is generally better than individual models and improves the generalization of the solution [68].

Many researchers have investigated the improvement of skin lesion classification with ensemble approaches. They have shown, it is possible to improve performance significantly [16, 39].

Ensemble learning is also showing promising progress on graph representation learning. Most graph learning methods are trying to preserve certain characteristics of the original graph in the low dimensional space. Graphs in the real world are huge and have a combination of different factors, which makes it difficult to capture all of the important characteristics with a single method [12].

In the next chapter, we propose our two ensemble models consisting of a pre-trained ResNet18 model and a GAT network and the second one consisting of the GIN and GAT.

Chapter 3

Datasets, Methodology and Frameworks

In this chapter, we introduce our ensemble frameworks and the datasets used in our experiments.

3.1 Our Proposed Frameworks

Recent studies show that using superpixels instead of original pixel set of the image, increases a the performance of classification task on GNNs significantly [7]. Regions of interest in the colored image datasets as used in this thesis, were quite distinguishable from other parts which makes it a great fit to use superpixel. The dataset was converted to superpixel through SLIC algorithm and then superpixels were converted to graphs and fed into Graph neural networks. In the second part of our experiment, we are proposing two ensemble frameworks. At first, we combine a pre-trained ResNet18, and a GNN model (GAT Network), which we call it GATRes Network. Using the principles of transfer learning [38, 55], the first 10 layers on the ResNet18 have been frozen and the last 8 layers were fine-tuned on our dataset. In this approach, we simultaneously input the images into the ResNet18 network and the generated graphs on these images to the GAT network. Then we concatenate the features extracted by the two networks and pass them to a linear classifier for the final classification. The same process is applied to the second framework.

The second ensemble model is a combination of two Graph Neural Network models (GAT and GIN). We call this model GATGIN. The same method will be used as the final classification of each image. We implemented these frameworks on two real-world image datasets in the medical field,

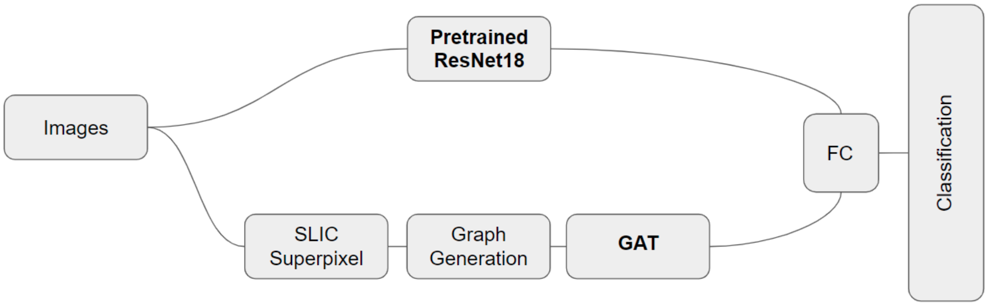


Figure 3.1: Proposed ensemble framework, GATRes

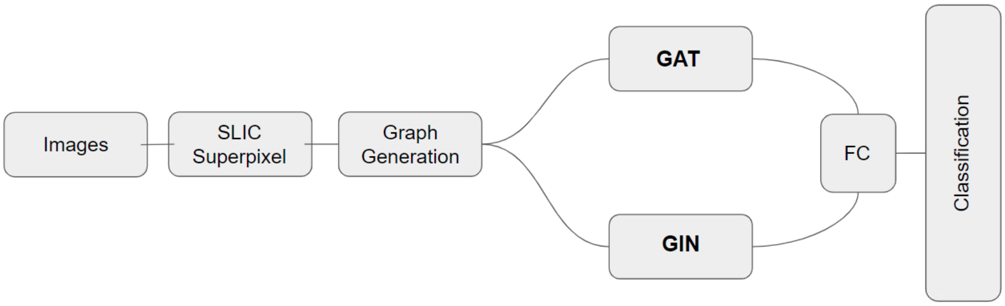


Figure 3.2: Proposed ensemble framework, GATGIN

and we report the results in Chapter 4. Figures 3.1 and 3.2 show the architecture of our framework.

Our frameworks can be used:

1- to convert any color images into superpixels, then into graphs and finally to be used as inputs of graph networks (implemented using a modification to the code from [28]).

2- as an ensemble method for colored image classifications using neural graph networks.

In the next sections, we explain the problem statements and datasets we implemented our ensemble frameworks on.

3.2 Problem Statements

In this section, two real-word medical problems have been discussed and the methodology of proposed models are explained. The first problem is the Cervical Positioning task to screen cervical cancer, and the second one is the skin cancer classification task (Melanoma).

3.2.1 Cervical Position Screening

Cervical cancer is easy to prevent if caught in its pre-cancerous stage. Today, women worldwide in low-resource settings benefit from programs where cancer is identified and treated in a single visit. However, due to lacking expertise in the field, one of the greatest challenges of these cervical cancer screens and treatment programs is determining the appropriate treatment method, which can vary depending on patients' physiological differences. Especially in rural areas, many women at high risk for cervical cancer are receiving treatment that will not work properly and it may even worsen the situation. This deficiency is mainly due to the high variability of the positioning of cervix in different patients. Health providers can identify high-risk patients but may not have the skills to discern which treatment will prevent cancer in these women reliably. Even worse, applying the wrong treatment has a high cost. A treatment that works effectively for one woman may obscure future cancerous growth in another woman, greatly increasing health risks. Healthcare providers' workflows would be greatly improved given the ability to make real-time determinations about patients' treatment eligibility based on cervix type [23].

The image dataset used in this thesis is coming from a Kaggle competition wherein Intel is partnering with MobileODT, which offers a Quality Assurance workflow to support remote supervision which helps healthcare providers make better treatment decisions in rural settings, to develop an algorithm that accurately identifies a woman's cervix type based on images. Doing so will prevent ineffective treatments and allow healthcare providers to properly refer to cases that require more advanced treatment.[1]

The objective here is to develop algorithms to classify cervix types based on cervical images

[1] A written permission for using this dataset is obtained.

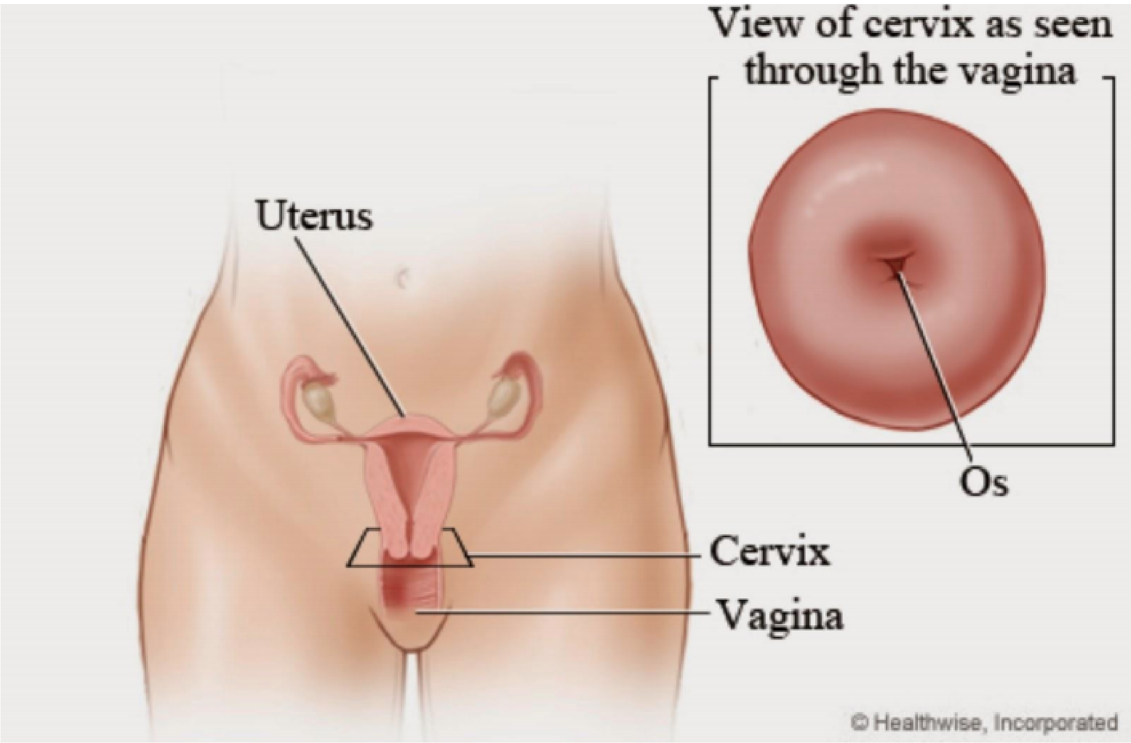

View of cervix as seen through the vagina
Uterus
Os
Cervix
Vagina
© Healthwise, Incorporated

Figure 3.3: Vaginal Canal and the Cervix position [23], Os : Orifice

correctly. There are three cervix types: type 1, 2 and 3. So, this problem is a multi-class classification, and it consists of three classes representing three cervix positions. We will explain what a cervix is and what we mean by cervical position in the following paragraphs.

In Figure 3.3 you can see the cervix anatomy and how it is positioned in the body. There is a zone called the transformation zone (TZ) on the entrance of the cervix (as indicated in Figure 3.4). Most cervical cancers begin in the cells in the transformation zone [23].

These TZs can be ectocervical or endocervical. The cervix is made of two parts and is covered with two different types of cells. The endocervix is the opening of the cervix that leads into the uterus. It is covered with glandular cells. The exocervix (or ectocervix) is the outer part of the cervix that can be seen by the doctor during a speculum exam[2] (please see Figure 3.5). In Type 1, the TZ is completely ectocervical, and it is fully visible. In Type 2, the TZ has an endocervical component, is fully visible, and may also have an ectocervical component. In Type 3, the TZ has endocervical component, however, it is not fully visible and it may also have ectocervical component (Figures

[2] https://www.cancer.org/cancer/cervical-cancer/about/what-is-cervical-cancer.html

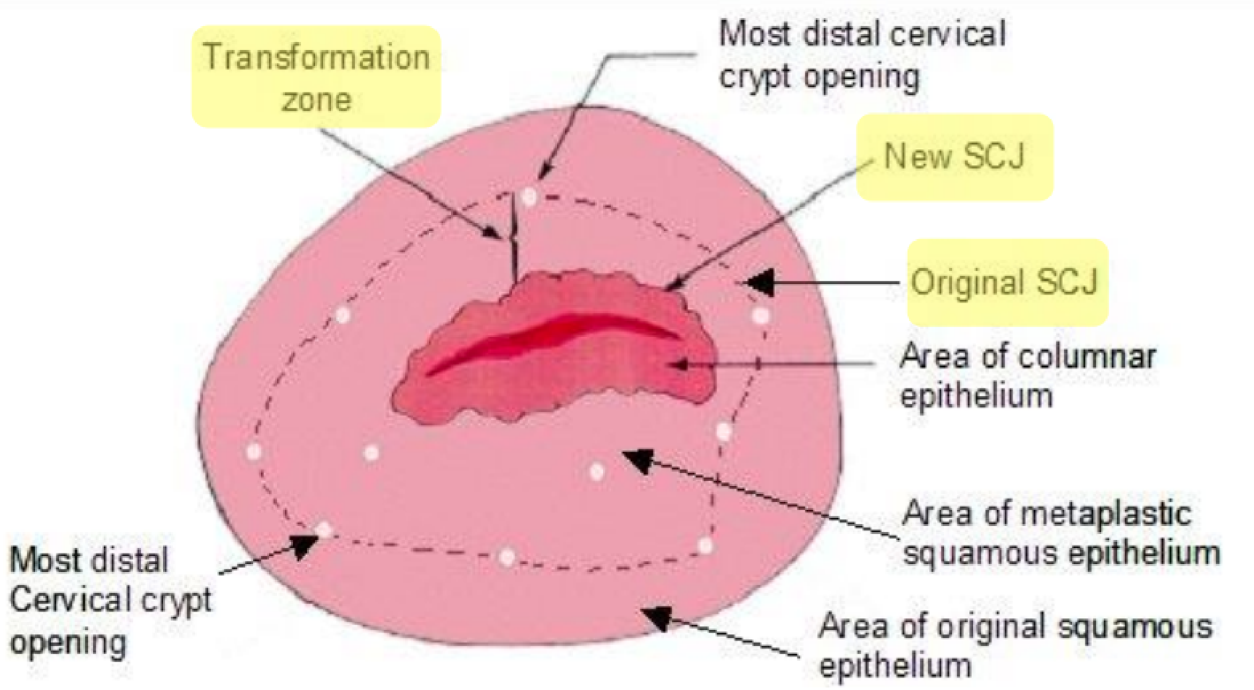

Transformation zone
Most distal cervical crypt opening
New SCJ
Original SCJ
Area of columnar epithelium
Area of metaplastic squamous epithelium
Most distal Cervical crypt opening
Area of original squamous epithelium

Figure 3.4: Transformation zone[23]

3.6a, 3.6b, and 3.6c).

These different types of the cervix in the data set are all considered normal (not cancerous). However, since the transformation zones are not always recognizable, some patients require further testing while some do not. This decision is critical for both the patient and the healthcare professional. Identifying the transformation zones is not an easy task for these heath care professionals. Therefore, an algorithm-aided decision will be extremely helpful and improve the efficiency of cervical cancer screening. Cervix Types 2 and 3 might include hidden lesions and require different treatment [23].

Previous work on the Cervix dataset

Since the Kaggle challenge in May 2017, a number of papers have been published using different machine learning methods on this dataset. The followings are a summary of these works (also shown in Table 3.1).

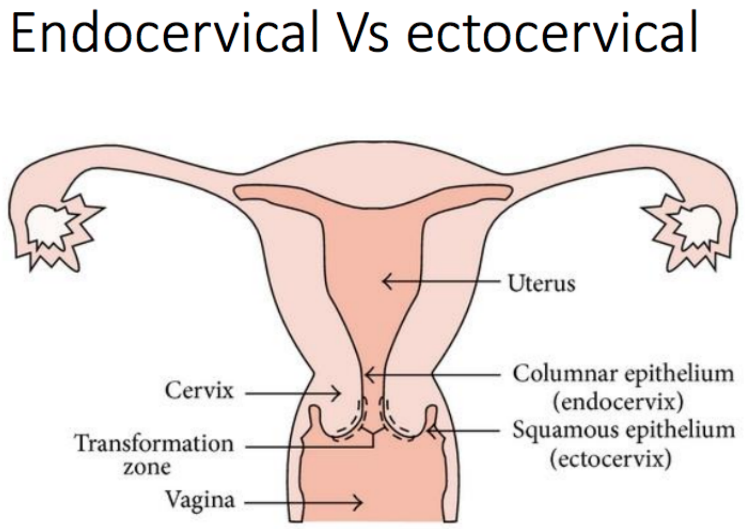
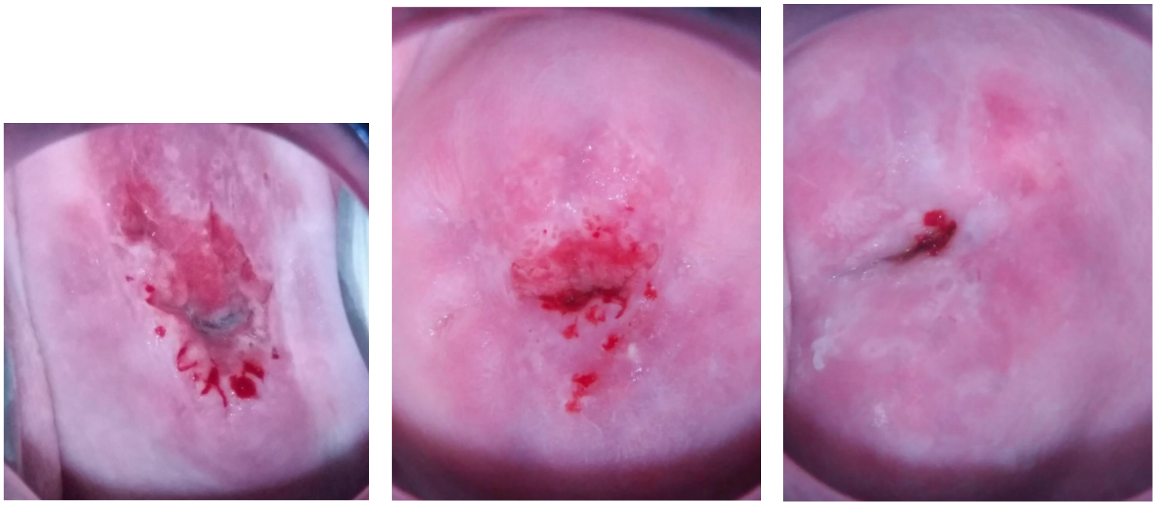

Endocervical Vs ectocervical
Uterus
Columnar epithelium (endocervix)
Cervix
Squamous epithelium (ectocervix)
Transformation zone
Vagina

Figure 3.5: Endocervical versus ectocervical [23]



(a) Type 1

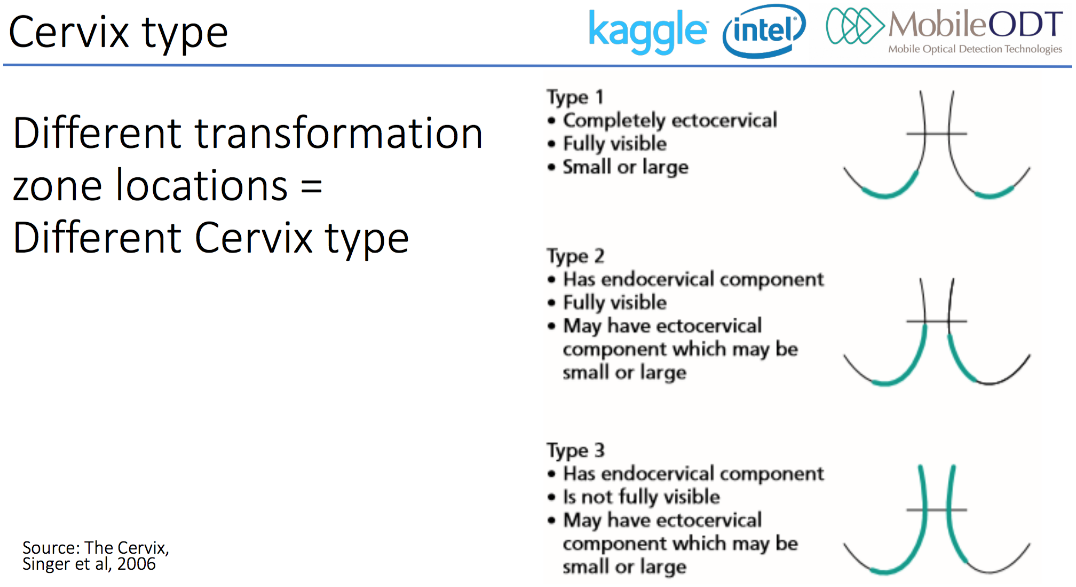
(b) Type2

(c) Type3

Figure 3.6: Cervix Types [23]

Dataset	ResNet32	VGGNet16	SqueezeNet	AlexNet
Cervix	58.8[18]	62.1[25]	63.3[2]	62.6[2]

Table 3.1: Comparison of previous models' accuracy on Cervix test set



Cervix type

kaggle intel MobileODT Mobile Optical Detection Technologies

Different transformation zone locations = Different Cervix type

Type 1
- Completely ectocervical
- Fully visible
- Small or large

Type 2
- Has endocervical component
- Fully visible
- May have ectocervical component which may be small or large

Type 3
- Has endocervical component
- Is not fully visible
- May have ectocervical component which may be small or large

Source: The Cervix, Singer et al, 2006

Figure 3.7: Transformation Zones [23]

In [41] the authors used ResNet32 and ResNet52 with various setups and the best results they achieved was with ResNet32 [18] with data augmentation which reached **58.8%** accuracy on their validation set. In [2] the authors performed AlexNet[29] and SqueezeNet and reached the accuracies of **62.6%** and **63.3%** respectively. They have reported that SqueezeNet [20] was achieving this accuracy very quickly. The other paper is [25] where in Kaur *et al.* used pretrained VGG16/19 [54] and achieved **62.1%** on VGG16.

3.2.2 Melanoma

Skin cancer is the most frequent type of cancer. Melanoma, specifically, is the cause for 75% of skin cancer deaths, despite being the least common skin cancer. At the moment, dermatologists evaluate every patient's moles to identify outlier lesions that are most likely to be melanoma [24].

Melanoma is a deadly disease, but most melanomas can be cured with minor surgeries if caught early. Better detection of melanoma will impact millions of people's lives positively [24]. In Fig. 3.8 you can see a melanoma lesion and Fig. 3.9 shows a benign lesion.

The dataset is generated by the International Skin Imaging Collaboration (ISIC), with images gathered from the following sources: Hospital Clinic de Barcelona, Medical University of Vienna,

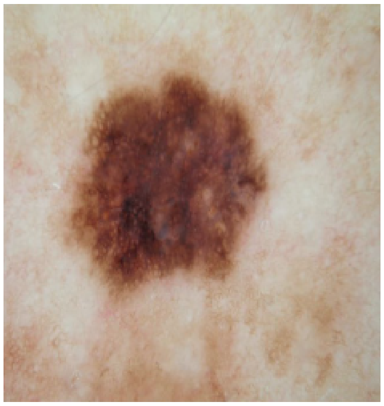


Figure 3.8: Melanoma lesion [4]

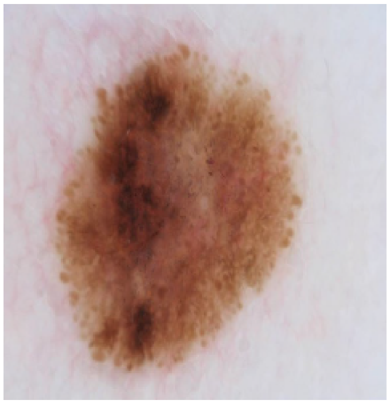


Figure 3.9: Benign lesion [4]

Memorial Sloan Kettering Cancer Center, University of Queensland, Melanoma Institute Australia, and the University of Athens Medical School [47]. The dataset used in this thesis is a balanced version of dermoscopic images from the International Skin Imaging Collaboration (ISIC) archive. The objective of our models is to correctly classify the images into "benign" and "malignant" classes.

Previous work on the Melanoma Dataset

The International Skin Imaging Collaboration (ISIC) publishes melanoma skin lesion images every year. Many papers have been written using these datasets or combining these data from multiple years using deep learning architectures for classification and segmentation purposes. For instance, Jaworek-Korjakowska *et al.* [21] conducted superpixel segmentation of pigmented skin lesions using the SLIC algorithm. Then, they constructed a region adjacency graph (RAG) image. Their graph nodes were the superpixels, and edges represent their neighborhood relationships. However, this study aimed to improve the segmentation method on this task and not a classification method. Deep learning architectures, especially convolutional neural networks (CNN), have been recently used in tasks of dermoscopic image analysis. Some examples of the previous work on Melanoma dataset and classification task of benign and malignant using ISIC dataset are as follows. Also, a summary of their results along with other models is shown in Table 3.2.

Majtner *et al.* [34] trained a support vector machine (SVM) on their extracted features on the ISIC 2016 dataset and achieved an accuracy of 82.6%. Gutman *et al.* [13] reviewed the ISIC 2016 challenge where all participants used deep learning methods. The best method achieved an AUC of 86%. Lopez *et al.* [33] conducted various VGG16 CNN architectures for melanoma detection and achieved an accuracy of 81.33% and a sensitivity of 78.66%. Esteva *et al.* [8] gathered 129,450 dermoscopic images consisting 2,032 different diseases. They trained a CNN on this data and its performance was tested against 21 board-certified dermatologists on biopsy-proven clinical images with two binary classification use cases. One of them was malignant melanoma versus benign lesions. Moreover, on this task, they achieved an accuracy of 72.10%. Brinker *et al.* [6] compared the performance of a CNN architecture against the performance of nine dermatologists on the classification of malignant melanoma and benign images. Oliveira *et al.* [39] applied ensemble classifiers with hand-crafted features and achieved 94.3% accuracy. And, finally, in a very recent publication

Author	Year published	Dataset	Classifier	Accuracy	Sensitivity
Gutman *et al.* [13]	2016	ISIC2016	Deep Networks	85.50%	50.70%
Majtner *et al.* [34]	2016	ISIC2016	SVM	82.60%	53.30%
Lopez *et al.* [33]	2017	ISIC2016	VGG16	81.33%	78.66%
Esteva *et al.* [8]	2017	ISIC, Edinburgh Dermofit Library and the Stanford Hospital	CNN	72.1%	-%
Oliveira*et al.* [39]	2017	ISIC2016	Ensemble of classifiers	94.30%	91.80%
Menegola *et al.* [36]	2017	ISIC2017	VGG16	72.10%	54.70%
Reboucas Filho *et al.* [42]	2018	ISIC2017 ISIC2016	SVM SVM	89.93% 94.50%	92.15% 95.23%
Brinker *et al.* [6]	2019	ISIC Dermoscopic Archive	CNN	-%	82.30%
Annaby *et al.* [3]	2021	ISIC2017 +250ISIC melanoma images	Random Forest	97.50%	100.00%

Table 3.2: Comparison of previous models' accuracy and sensitivity on Melanoma

(January 2021), Annaby *et al.* [3] applied SLIC superpixel algorithm on ISIC2017 images + 215 melanoma images and generated graphs on these obtained superpixels. They then used these graph superpixel features and trained a random forest classifier. They achieved 97.5% on accuracy and 100% on sensitivity. Although, they focus more on the feature extraction techniques in their paper.

The dataset we used in this thesis is none of the above-mentioned ones as they are highly imbalanced. We used a balanced version of the ISIC archive. We kept the malignant samples and removed the benign ones to reach 1800 images for both.

Chapter 4

Experiments, Setup and Results

In this chapter, we describe how we used the ensemble framework on two medical datasets. We firstly describe the datasets' setup and the amount of data each of them contains. Then, we describe how we prepared the data and all the pre-processing steps. The experiment setup details are explained next, and finally, we report the results of the experiments and compare the results of the different networks on the proposed method. Results show that the ensemble approach outperforms the accuracy of the individual GNNs in our classification tasks.

4.1 Datasets' splitting

This section will explain the amount of data in each dataset and the portions of the training, validation, and test sets.

4.1.1 Cervical Position Screening

This dataset consists of 1481 labeled training images. 250 images of type 1 , 781 of type 2 and 450 of type 3. There are also 6734 additional images ((1191 type 1, 3567 type 2 , 1976 type 3)) to help train the models. These images sometimes come from duplicated patients, so the images might look alike since they are taken in the same session; sometimes, they are not selected because of image quality. We went through all the images and selected the best quality images. We also removed the duplicate ones. This selection led us to the following proportions in the training set:

1638 type 1, 1580 in type 2 and 1556 in type 3. We used 1/5 of the images in each type as validation data. We were also provided with test images data with the following portions: Type 1: 84, Type 2: 260 and Type 3: 150 images.

4.1.2 Melanoma

ISIC publishes melanoma skin lesion datasets every year, and the dataset used in this thesis is the combination of the last 3 years. It is balanced through removing of benign images to match the number of malignant cases. This dataset contains 1440 benign and 1197 malignant images in the training set. In addition, it has 360 benign and 300 malignant in the test set. We used 1/5 of the training images in each type as validation data.

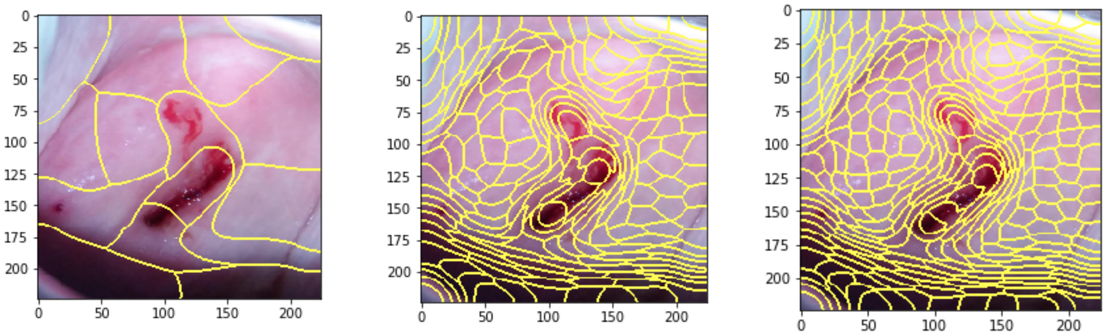
4.2 Data Preparation

This section describes the experiment environment, packages used for the experiment, and the data preparation steps. The programming language used is Python3.7 [46]. The models are implemented using DGL 0.4.2 [64] and Pytorch framework [40]. Other packages and libraries used for processing and segmenting the images are Numpy [17], Scipy [63], Skimage [57], PIL [56] and the plots are generated using Mathplotlib [19].

The pre-processing steps are as follows:

- **Resizing:** As the resolution of each image was different, all images were resized to 224 × 224 pixels.

- **Cropping:** On the Cervix dataset, the various views (angles and distances that each image has been taken from) and the existence of multiple objects on each image raised the need for cropping the images to show the same part of the cervix. We had to manually crop the images as the images were taken from different angles, lights, and patient positions. Other cropping methods, such as center-based crops or color-crops, were also explored. However, the results were not satisfying as the uterus' colors, and position also varied in each image.



(a) 15 segments

(b) 200 segments

(c) 300 segments

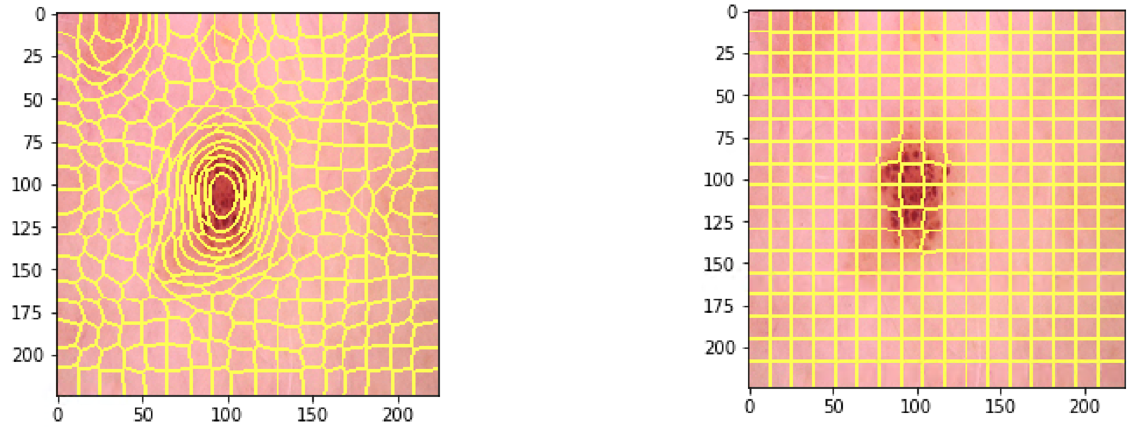
Figure 4.1: Superpixel segments

4.3 SLIC superpixel algorithm on the images and hyperparameter selection

After the pre-processing, the SLIC superpixel method was implemented on the pre-processed images. The hyperparameters of the SLIC method are superpixels' compactness, sigma and the number of segments. We tried various values for compactness and the number of segments. Four different numbers of segmentation (15,100, 200 and 300) were explored, as you can see in Figure 4.1 the higher number of segments, the more accurate the segmentation. We chose 300 segments as it leads to a better determination of the transformation zone, our region of interest. We also tried multiple values for compactness. In Figures 4.2 you can see the effect of different compactness (2 and 25) and how superpixels are shaped. Based on the results, we realize that the smaller compactness results in the better acquisition of the lesion's shape or the cervix. After multiple choices for this number, we heuristically chose 2 as a valid compactness value for our experiment.

4.4 Graph preparation

In order to be able to feed the images to the graph neural network models, we need to generate graphs over the images. The nodes, edges and adjacency matrix should be defined. We calculate the center coordinate of the mass for each superpixel, relative to the position of the superpixel in the image, and use them as our graph's nodes. We assign each node's features as the superpixel



(a) compactness = 2

(b) compactness = 25

Figure 4.2: SLIC superpixels with compactness of 2 and 25 on a benign skin lesion

coordinates and intensity. The edges will be the Euclidean distance between the nodes, and for each sample, a k-nearest neighbor adjacency matrix with

$$W_{ij}^{k-\mathrm{NN}} = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma_x^2}\right)$$

where x_i, x_j are the 2-D coordinates of super-pixels i, j and σ_x is the scale parameter defined as the averaged distance x_k of the k nearest neighbors for each node [7]. We use k = 8 for both of our datasets. Some examples of the graphs created based on the superpixels are shown in Figures 4.3.

4.5 Models and setup details

This section describes the models we used to train our data in this thesis and the information on the models' parameters and setup. We performed multiple experiments on our two medical datasets.

The first experiment was to implement GNN models on these datasets and investigate the results. As the GraphSage model had one of the best performances on CIFAR10 [7] and our datasets were also more similar to CIFAR10 in terms of colored images, we decided to start with the GraphSage model. Then we implemented GAT and GIN model and investigated the results on both datasets. (Results are shown in the next section)

Our second experiment was proposing two ensemble frameworks as described in 3.1. On the first framework, we ensembled a pre-trained ResNet18 with GAT and called this model GATRes.

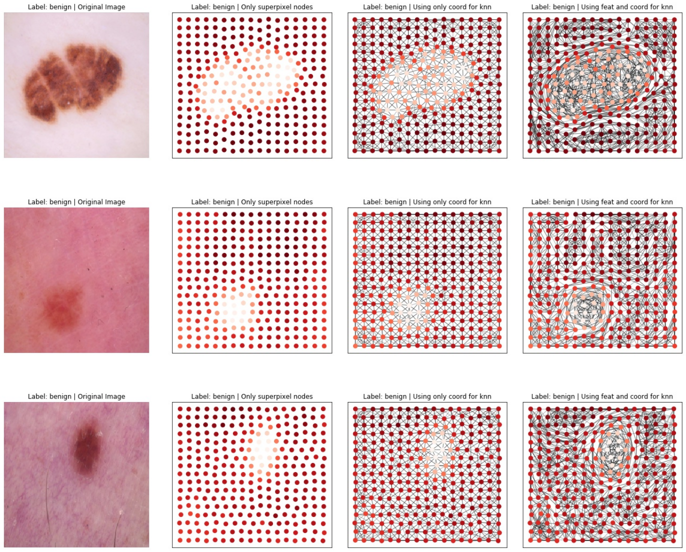

Label: benign | Original Image
Label: benign | Only superpixel nodes
Label: benign | Using only coord for knn
Label: benign | Using feat and coord for knn
Label: benign | Original Image
Label: benign | Only superpixel nodes
Label: benign | Using only coord for knn
Label: benign | Using feat and coord for knn
Label: benign | Original Image
Label: benign | Only superpixel nodes
Label: benign | Using only coord for knn
Label: benign | Using feat and coord for knn

Figure 4.3: Graphs created based on superpixels

On the ResNet18, which was pre-trained on the ImageNet dataset, firstly, we experimented with the feature extraction method (we froze all the layers except for the final layer). However, the results were not satisfactory. We tried multiple combination of fine tuned layer and the best results was achieved when the last 8 layers of this network was fine tuned on our data, and finally, we concatenated the results from this network with the output of the GAT network. As the second framework, we combined the GAT and GIN Networks and called it GATGIN. We performed these two frameworks on our generated graphs and compared the results with individual models.

The loss function used in our experiments is cross-entropy loss, and as for the optimizer, the Adam optimizer [26] was used.

The overall parameters are shown in Table 4.1. In all the experiments, the learning rate will start from an initial amount of $5e^{-5}$ and gradually decrease (by half) until it reaches 10^{-6}. From then onward, it will continue with the 10^{-6} learning rate.

The networks hyper parameters are explained in Tables 4.2 to 4.6. As shown in Table 4.6 due to the excessive number of parameters, our GATGIN model was prone to over-fitting on the cervix. Therefore, we modified the hyper parameters from the following values to the values indicated in the table. Number of heads: 8, hidden dimension: 20, number of layers: 4 and number of MLP :3.

Parameters	Values
Epochs	1000
Batch size	50
Initial learning rate	$5e^{-5}$

Table 4.1: Parameters

HyperParams	Cervix	Melanoma	**HyperParams**	Cervix	Melanoma
Input dim	300	300	# of heads	5	5
Hidden dim	10	10	# of Layers	4	4
Dropout	0.1	0.1	Readout	sum	

Table 4.2: GAT hyper-parameters

HyperParams	Cervix	Melanoma	**HyperParams**	Cervix	Melanoma
Input dim	300	300	# of MLP	2	2
Hidden dim	110	110	# of Layers	4	4
Dropout	0.1	0.1	Readout	sum	
Neighbor aggr	sum	sum	Learn Eps	True	True

Table 4.3: GIN hyper-parameters

HyperParams	Cervix	Melanoma	**HyperParams**	Cervix	Melanoma
Input dim	300	300	Sage Aggregator	meanpool	meanpool
Hidden dim	108	108	# of Layers	4	4
Dropout	0.1	0.1	Readout	sum	sum

Table 4.4: GraphSage hyper-parameters

HyperParams	Cervix	Melanoma	**HyperParams**	Cervix	Melanoma
Input dim	300	300	# of heads	8	8
Hidden dim	19	19	# of Layers	4	4
Dropout	0.1	0.1	Readout	mean	mean

Table 4.5: GATRes hyper-parameters

HyperParams	Cervix	Melanoma	**HyperParams**	Cervix	Melanoma
Input dim	300	300	# of heads	6	8
Hidden dim GIN	50	110	# of Layers	2	4
Hidden dim GAT	10	20	# of MLP	3	2
Dropout	0.1	0.1	Readout	sum	sum
Neighbor aggregation	sum	sum	Learn Eps	True	True

Table 4.6: GATGIN hyper-parameters

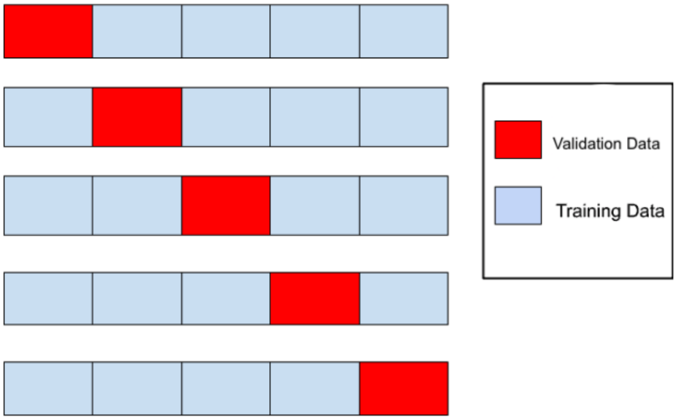


Figure 4.4: 5-Fold Cross Validation [1]

We performed 5-fold cross-validation (Figure 4.4) on both datasets. the data was divided into 5 roughly equal parts, and throughout the training, we iteratively trained 4/5th of the data and validated the model on the 1/5th. Finally, outcomes were averaged and reported results were achieved.

4.6 Results

We calculated three metrics for our experiments, accuracy, sensitivity and specificity. For both Cervix and Melanoma datasets, we report the best-achieved accuracy from implementing models, individually as well as the ensemble form in Table 4.7. We also can have a better insight and interpretation of our results by looking at their confusion matrices in Figures 4.5 and 4.6. These matrices show us how precise and reliable our networks are in predicting a certain class and help us calculate sensitivity and specificity. Sensitivity evaluates the model's ability to detect the disease. Sensitivity

	Cervical Screening			Melanoma		
Models	#Params	Accuracy	#Epochs	#Params	Accuracy	#Epochs
GraphSage	102711	55.4%	999	102683	78.9%	999
GAT	110431	49.4%	276	110392	80.4%	545
GIN	101769	55.8%	471	101214	79.2%	695
ResNet18	11178051	64.5%	103	11177538	86.1%	157
GATRes	11294749	**65.3%**	366	11294084	**88.0%**	198
GATGIN	28148	56.4%	370	230140	83.6%	595

Table 4.7: Results on Cervix and Melanoma test sets

is an important factor in medical studies as it can aid decision-makers in understanding the results better. Specificity evaluates the model's ability to identify the healthy class. In order to calculate these two factors, we need 4 metrics from the confusion matrices: True Positive samples (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). These elements' descriptions are as follows:

True Negative (TN) will be the outcome when the model correctly predicts the negative class (usually the healthy data).In the case of the Melanoma dataset, TN refers to benign images that have been correctly classified as benign.

False Positive (FP) is an outcome where the model incorrectly predicts the positive class (benign images that are classified as malignant).

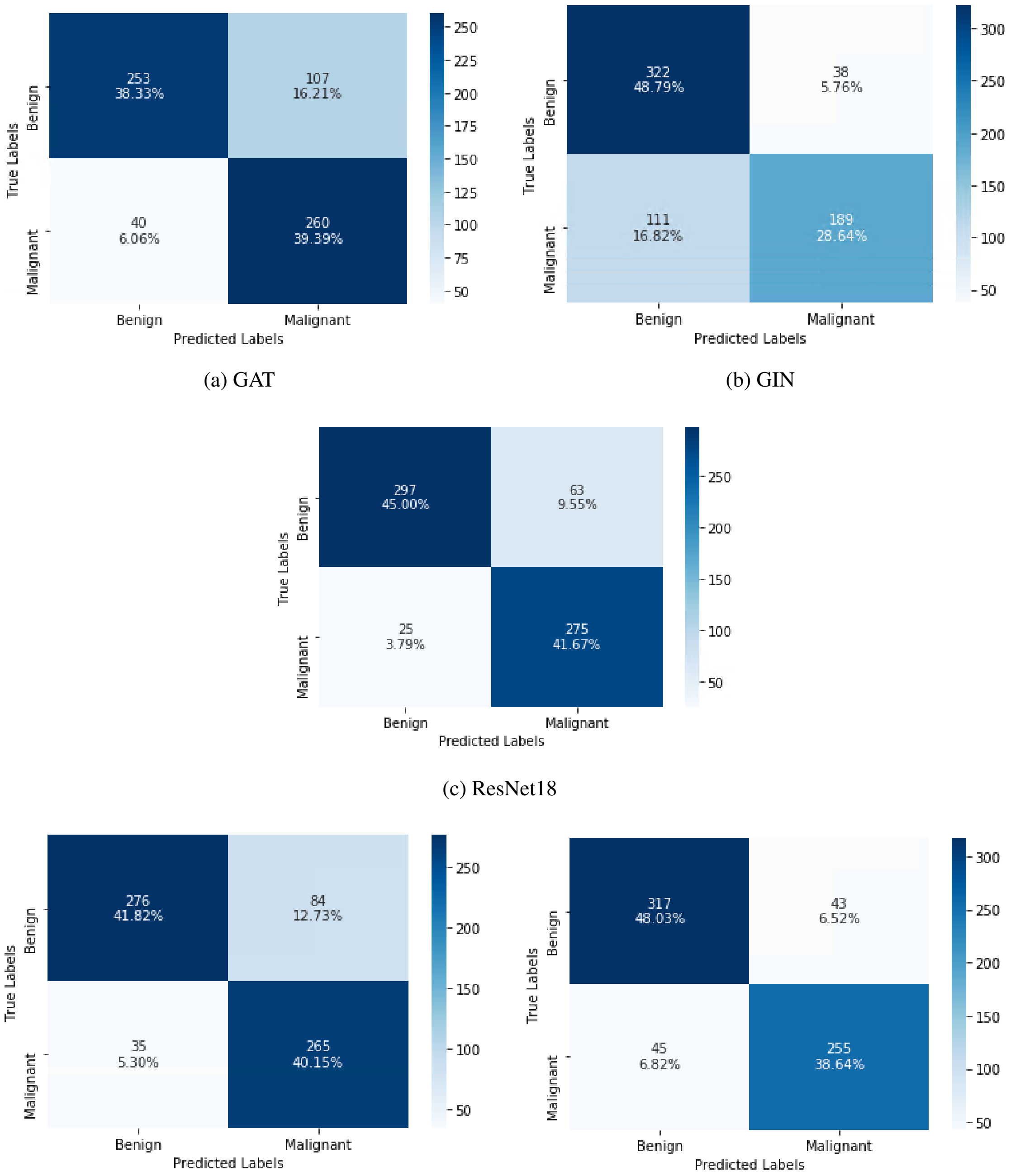
And **False Negative (FN)** is an outcome where the model incorrectly predicts the negative class (malignant lesions being classified as benign).

With these elements, we can now calculate the Sensitivity and Specificity with the following equations:

$$\text{Sensitivity} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}}$$

$$\text{Specificity} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}}$$

The Sensitivity and Specificity of our models are shown in Tables 4.8 and 4.9.



(a) GAT

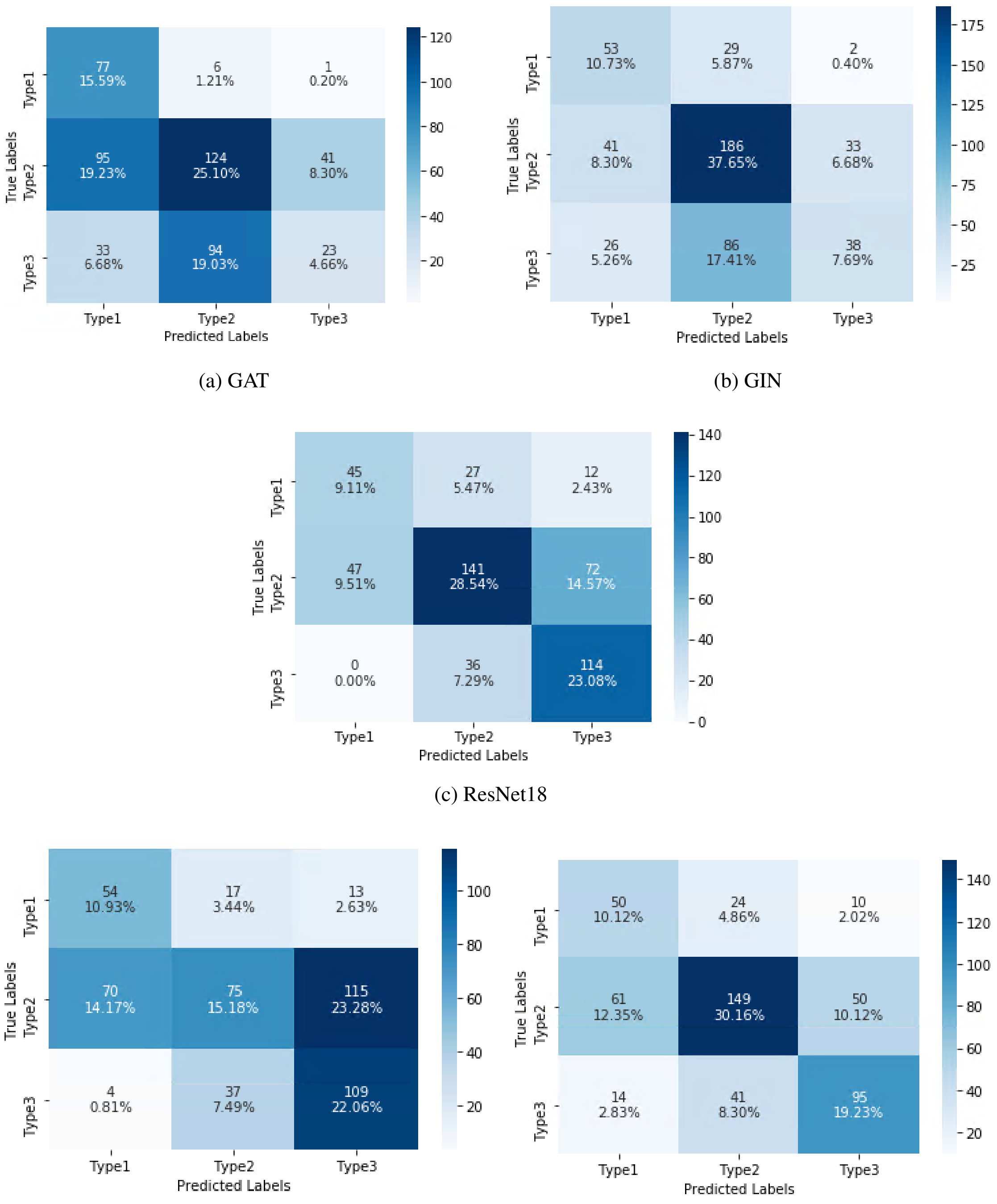
(b) GIN

(c) ResNet18

(d) GATGIN

(e) GATRes

Figure 4.5: Confusion matrices on Melanoma dataset



(a) GAT

(b) GIN

(c) ResNet18

(d) GATGIN

(e) GATRes

Figure 4.6: Confusion matrices on Cervix dataset

Models	Sensitivity	Specificity
GAT	86.6	70.2
GIN	63	**89.4**
ResNet18	**91.6**	82.5
GATRes	85	88
GATGIN	**91.3**	76.6

Table 4.8: Sensitivity and Specificity on Melanoma dataset

	Sensitivity			Specificity		
Models	Type 1	Type 2	Type 3	Type 1	Type 2	Type 3
GAT	**91**	47	15	68	57	**87**
GIN	63	**71**	25	83.6	50.8	**89.8**
ResNet18	53	54	**76**	**88**	73	75
GATRes	**59**	**59**	**63**	81.7	72.2	**82.5**
GATGIN	64	28.8	**72**	**81**	76	62.7

Table 4.9: Sensitivity and Specificity on Cervix datasets

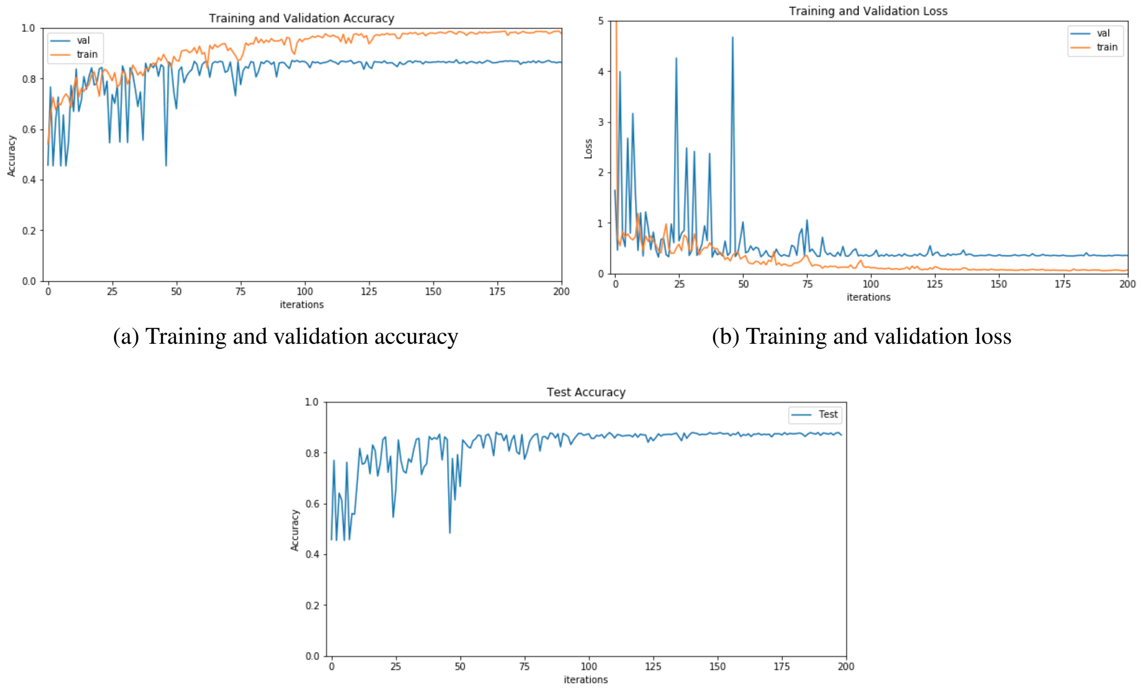
An example of convergence diagrams in the process of training GATRes on the Melanoma dataset is shown in Figure 4.7. The diagram ensures us that there has not been any over-fitting during the training.

The comparison of the results with the other models previously implemented on the Cervix dataset can be found in Table 4.10.

As we can see in the Results Table 4.7, our proposed GATRes model achieved the highest accuracy on both of our datasets. It even outperformed ResNet18, a state-of-the-art model. However, despite achieving the best accuracy, its sensitivity is lower than GATGIN which has achieved 91.3% in sensitivity 4.8. It's worth noting that for specificity, GIN showed higher performance on the melanoma dataset. For this problem sensitivity is the most important factor because if patients

Dataset	ResNet32[18]	VGG16[54]	Squeeze[20]	AlexNet[29]	GATRes	GATGIN
Cervix	58.8[18]	62.1[25]	63.3[2]	62.6[2]	**65.3**	56.4

Table 4.10: Comparison of different models' accuracy on the test set



(a) Training and validation accuracy

(b) Training and validation loss

(c) Test accuracy

Figure 4.7: Convergence Diagram of GATRes on Melanoma dataset

get diagnosed with a disease they do not have (False positive), some extra tests will be done. However, if a patient does not get diagnosed with their disease (False negative), their life will be at risk. Thus, for this problem, GATGIN would be the best model in our experiment.

For the Cervix dataset, accuracy is a better metric than sensitivity as we are dealing with 3 classes instead of two, and all of our classes are equally important for detection. Therefore, as we can see on the sensitivity table of the Cervix dataset 4.9 that GATRes performed better than other models as it has a better performance on all the three classes and it correctly classifies all the three datasets in 60% of the time.

As shown in the tables, our ensemble models outperform the accuracy of all the individual GNNs networks and the ResNet18 in this classification task.

As we can see, the GAT-ResNet network works better than ResNet18 on the Melanoma dataset; we can conclude that the graph neural network approach determines features that the CNN model does not learn. Therefore, the combination of these approaches can be very beneficial in many applications.

Chapter 5

Conclusions and Future Work

This chapter will give the conclusions and future work of this thesis.

5.1 Conclusions

This thesis conducted two experiments on two real-world medical applications: Cervical Cancer screening and Melanoma. Our first experiment applied three GNN models (GAT, GIN and Graph-Sage) on these datasets and investigated the results. Our second experiment was proposing two ensemble graph frameworks. The first framework consisted of a pre-trained ResNet18 and a GNN model (GAT), named it GATRes. The second framework included two graph neural network models (GAT and GIN) called GATGIN.

We transformed original images of the mentioned datasets into a graph of superpixels using the SLIC algorithm. Then we defined the center coordinates of these superpixels as nodes and their relative distances as edges. Then we fed these graphs to the three GNN models as well as our ensemble models. We demonstrated that our GATRes model had the highest accuracy among all the previous published papers on the Cervix dataset. It also outperforms the individual models in terms of accuracy on both datasets. Our GATGIN model also achieved a fairly high sensitivity on the Melanoma dataset. Our results showed:

- Our ensemble models outperform the current methods on the Cervical Cancer Screening dataset.

- Due to the high dimensionality of graphs, one GNN approach cannot capture all of the important characteristics of graphs' data. Thus, leveraging from multiple and ensemble approaches would increase the ability to capture more features and enhance the accuracy.

- Our framework can be used for transforming any color images into superpixels and then into graphs that can be fed to graph neural network models.

- Although our approach is not beating the results from deep neural networks on Melanoma such as ResNet, we can conclude that there is still a long way to explore for the implementation of graphs for image classification. However, the proposed ensemble approach has a better performance than any individual models on both datasets.

5.2 Future Work

- **SLIC Segmentation Hyper parameters** We picked the most optimized segmentation number and compactness for the SLIC segmentation manually and heuristically. However, an automated comparison among different segments and compactness is open to investigation to pick the most optimized ones (implementing a hyperparameter tuning system).

- **Automated Cropping** For this particular dataset, cropping showed very better results. This cropping can also become automated and part of the preparation. This automation experiment can be done using algorithms such as Yolo [43], etc.

- **Graph generation** The nodes for feeding the graph are now based on the center of the superpixels. Other ways can be investigated. Also, the defining edges based on different distance methods can be explored.

- **Ensemble method** In this thesis, the final classification was made with concatenation (the ensemble method). Other methods such as majority vote and Averaging can be explored as well.

- **Ensemble of other networks** On the cervix dataset, the ensemble of GAT and Squeeze net or GAT and Random Forest can be explored to investigate if any computation time can be saved while maintaining the accuracy. Also, based on the achieved results on sensitivity and accuracy of different models on the Cervix dataset, the ensemble of three models of GAT, ResNet18 and GIN can be a proper investigation to increase the overall accuracy.

Bibliography

[1] R. Achanta et al. “SLIC Superpixels Compared to State-of-the-Art Superpixel Methods”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.11 (2012), pp. 2274–2282. DOI: 10.1109/tpami.2012.120.

[2] O. E. Aina, Steve A. Adeshina, and A. Aibinu. “Classification of Cervix types using Convolution Neural Network (CNN)”. In: *2019 15th International Conference on Electronics, Computer and Computation (ICECCO)* (2019), pp. 1–4.

[3] Mahmoud H Annaby et al. “Melanoma Detection Using Spatial and Spectral Analysis on Superpixel Graphs”. In: *Journal of digital imaging* 34.1 (2021), pp. 162–181.

[4] Mahmoud H. Annaby et al. “Melanoma Detection Using Spatial and Spectral Analysis on Superpixel Graphs”. In: *Journal of Digital Imaging* 34.1 (2021), pp. 162–181. DOI: 10.1007/s10278-020-00401-6.

[5] C.M Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1995.

[6] Titus J Brinker et al. “Deep neural networks are superior to dermatologists in melanoma image classification”. In: *European Journal of Cancer* 119 (2019), pp. 11–17.

[7] Vijay Prakash Dwivedi et al. “Benchmarking Graph Neural Networks”. In: *arXiv preprint arXiv:2003.00982* (2020).

[8] Andre Esteva et al. “Dermatologist-level classification of skin cancer with deep neural networks”. In: *nature* 542.7639 (2017), pp. 115–118.

[9] Mark Everingham et al. “The pascal visual object classes (voc) challenge”. In: *International journal of computer vision* 88.2 (2010), pp. 303–338.

[10] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. “Efficient Graph-Based Image Segmentation”. In: *International Journal of Computer Vision* 59.2 (2004), pp. 167–181. DOI: 10.1023/b:visi.0000022288.19776.77.

[11] Justin Gilmer et al. “Neural Message Passing for Quantum Chemistry”. In: *CoRR* abs/1704.01212 (2017). arXiv: 1704.01212. URL: http://arxiv.org/abs/1704.01212.

[12] Palash Goyal et al. “Graph representation ensemble learning”. In: *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE. 2020, pp. 24–31.

[13] David Gutman et al. “Skin Lesion Analysis toward Melanoma Detection: A Challenge at the International Symposium on Biomedical Imaging (ISBI) 2016, hosted by the International Skin Imaging Collaboration (ISIC)”. In: *CoRR* abs/1605.01397 (2016). arXiv: 1605.01397. URL: http://arxiv.org/abs/1605.01397.

[14] William L. Hamilton, Rex Ying, and Jure Leskovec. “Inductive Representation Learning on Large Graphs”. In: *CoRR* abs/1706.02216 (2017). arXiv: 1706.02216. URL: http://arxiv.org/abs/1706.02216.

[15] WILLIAM L. HAMILTON. *GRAPH REPRESENTATION LEARNING*. MORGAN & CLAYPOOL PUBLISH, 2020.

[16] Balazs Harangi. “Skin lesion classification with ensembles of deep convolutional neural networks”. In: *Journal of biomedical informatics* 86 (2018), pp. 25–32.

[17] Charles R. Harris et al. “Array programming with NumPy”. In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[18] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[19] J. D. Hunter. “Matplotlib: A 2D graphics environment”. In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[20] Forrest N. Iandola et al. “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡1MB model size”. In: *CoRR* abs/1602.07360 (2016). arXiv: 1602.07360. URL: http://arxiv.org/abs/1602.07360.

[21] Joanna Jaworek-Korjakowska and Pawel Kleczek. “Region Adjacency Graph Approach for Acral Melanocytic Lesion Segmentation”. In: *Applied Sciences* 8.9 (2018). ISSN: 2076-3417. DOI: 10.3390/app8091430. URL: https://www.mdpi.com/2076-3417/8/9/1430.

[22] Jianbo Shi and J. Malik. “Normalized cuts and image segmentation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.8 (2000), pp. 888–905. DOI: 10.1109/34.868688.

[23] Kaggle Competition. *Intel & MobileODT Cervical Cancer Screening*. Accessed: 2020-06-15. URL: https://www.kaggle.com/c/intel-mobileodt-cervical-cancer-screening.

[24] Kaggle Competition. *SIIM-ISIC Melanoma Classification*. Accessed: 2021-02. URL: https://www.kaggle.com/c/siim-isic-melanoma-classification/overview.

[25] Navdeep Kaur, Nikson Panigrahi, and Ajay Mittal. “AUTOMATED CERVICAL CANCER SCREENING USING TRANSFER LEARNING”. In: 2017.

[26] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[27] Thomas N. Kipf and Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: *CoRR* abs/1609.02907 (2016). arXiv: 1609.02907. URL: http://arxiv.org/abs/1609.02907.

[28] Boris Knyazev, Graham W. Taylor, and Mohamed R. Amer. “Understanding attention in graph neural networks”. In: *CoRR* abs/1905.02850 (2019). arXiv: 1905.02850. URL: http://arxiv.org/abs/1905.02850.

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[30] Saima Anwar Lashari and Rosziati Ibrahim. "A Framework for Medical Images Classification Using Soft Set". In: *Procedia Technology* 11 (2013), pp. 548–556. DOI: 10.1016/j.protcy.2013.12.227.

[31] A. Levinshtein et al. "TurboPixels: Fast Superpixels Using Geometric Flows". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.12 (2009), pp. 2290–2297. DOI: 10.1109/TPAMI.2009.96.

[32] Yin Li et al. "Lazy snapping". In: *ACM Transactions on Graphics* 23.3 (2004), pp. 303–308. DOI: 10.1145/1015706.1015719.

[33] Adria Romero Lopez et al. "Skin lesion classification from dermoscopic images using deep learning techniques". In: *2017 13th IASTED International Conference on Biomedical Engineering (BioMed)* (2017), pp. 49–54.

[34] Tomas Majtner, Sule Yildirim-Yayilgan, and Jon Yngve Hardeberg. "Combining deep learning and hand-crafted features for skin lesion classification". In: *2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA)*. IEEE. 2016, pp. 1–6.

[35] David Martin et al. "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics". In: *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*. Vol. 2. IEEE. 2001, pp. 416–423.

[36] Afonso Menegola et al. "RECOD titans at ISIC challenge 2017". In: *arXiv preprint arXiv:1703.04819* (2017).

[37] Alastair P. Moore et al. "Superpixel lattices". In: *2008 IEEE Conference on Computer Vision and Pattern Recognition* (2008). DOI: 10.1109/cvpr.2008.4587471.

[38] Reza Vatani Nezafat, Olcay Sahin, and Mecit Cetin. “Transfer learning using deep neural networks for classification of truck body types based on side-fire lidar data”. In: *Journal of Big Data Analytics in Transportation* 1.1 (2019), pp. 71–82.

[39] Roberta B Oliveira, Aledir S Pereira, and João Manuel RS Tavares. “Skin lesion computational diagnosis of dermoscopic images: Ensemble models based on input feature manipulation”. In: *Computer methods and programs in biomedicine* 149 (2017), pp. 43–53.

[40] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[41] Jack Payette. “Intel and MobileODT Cervical Cancer Screening Kaggle Competition : Cervix Type Classification Using Deep Learning and Image Classification”. In: 2017.

[42] Pedro Pedrosa Rebouças Filho et al. “Automatic histologically-closer classification of skin lesions”. In: *Computerized Medical Imaging and Graphics* 68 (2018), pp. 40–54.

[43] Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: *CoRR* abs/1506.02640 (2015). arXiv: 1506.02640. URL: http://arxiv.org/abs/1506.02640.

[44] Rami Al-Rfou et al. “Theano: A Python framework for fast computation of mathematical expressions”. In: *CoRR* abs/1605.02688 (2016). arXiv: 1605.02688. URL: http://arxiv.org/abs/1605.02688.

[45] Brian D. Ripley. *Pattern recognition and neural networks*. Cambridge University Press, 1996.

[46] G. van Rossum. *Python tutorial*. Tech. rep. CS-R9526. Amsterdam: Centrum voor Wiskunde en Informatica (CWI), May 1995.

[47] Veronica Rotemberg et al. “A patient-centric dataset of images and metadata for identifying melanomas using clinical context”. In: *Scientific data* 8.1 (Jan. 2021), p. 34. ISSN: 2052-4463. DOI: 10.1038/s41597-021-00815-z. URL: https://europepmc.org/articles/PMC7843971.

[48] Sumit Saha. *A Comprehensive Guide to Convolutional Neural Networks-the ELI5 way*. Accessed: 2020-09. Dec. 2018. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[49] Bu¨lent Sankur and Mehmet Sezgin. “Survey over image thresholding techniques and quantitative performance evaluation”. In: *Journal of Electronic Imaging* 13.1 (2004), p. 146. DOI: 10.1117/1.1631315.

[50] D. R. Sarvamangala and Raghavendra V. Kulkarni. “Convolutional neural networks in medical image understanding: a survey”. In: *Evolutionary Intelligence* (2021). DOI: 10.1007/s12065-020-00540-3.

[51] Alexander Schick, Mika Fischer, and Rainer Stiefelhagen. “Measuring and evaluating the compactness of superpixels”. In: *Proceedings of the 21st international conference on pattern recognition (ICPR2012)*. IEEE. 2012, pp. 930–934.

[52] Neeraj Sharma et al. “Automated medical image segmentation techniques”. In: *Journal of Medical Physics* 35.1 (2010), p. 3. DOI: 10.4103/0971-6203.58777.

[53] Jamie Shotton et al. “TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context”. In: *International Journal of Computer Vision* 81 (Jan. 2009), pp. 2–23. DOI: 10.1007/s11263-007-0109-1.

[54] K. Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *CoRR* abs/1409.1556 (2015).

[55] Emilio Soria Olivas. *Handbook of research on machine learning applications and trends*. Information Science Reference, 2010.

[56] P Umesh. “Image Processing in Python”. In: *CSI Communications* 23 (2012).

[57] Stefan Van der Walt et al. “scikit-image: image processing in Python”. In: *PeerJ* 2 (2014), e453.

[58] Andrea Vedaldi and Stefano Soatto. “Quick Shift and Kernel Methods for Mode Seeking”. In: *Computer Vision – ECCV 2008*. Ed. by David Forsyth, Philip Torr, and Andrew Zisserman. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 705–718. ISBN: 978-3-540-88693-8.

[59] Olga Veksler, Yuri Boykov, and Paria Mehrani. “Superpixels and Supervoxels in an Energy Optimization Framework”. In: *Computer Vision – ECCV 2010 Lecture Notes in Computer Science* (2010), pp. 211–224. DOI: 10.1007/978-3-642-15555-0_16.

[60] Petar Veličković et al. *Graph Attention Networks*. Accessed: 2021-03-06. 2018. arXiv: 1710.10903 [stat.ML].

[61] W. N. Venables and B. D. Ripley. “Modern Applied Statistics with S-PLUS”. In: *Statistics and Computing* (1999). DOI: 10.1007/978-1-4757-3121-7.

[62] Luc Vincent and Pierre Soille. “Watersheds in digital spaces: an efficient algorithm based on immersion simulations”. In: *IEEE Computer Architecture Letters* 13.06 (1991), pp. 583–598.

[63] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[64] Minjie Wang et al. “Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks”. In: *arXiv preprint arXiv:1909.01315* (2019).

[65] Weibin Wang et al. “Medical Image Classification Using Deep Learning”. In: *Deep Learning in Healthcare: Paradigms and Applications*. Ed. by Yen-Wei Chen and Lakhmi C. Jain. Cham: Springer International Publishing, 2020, pp. 33–51. ISBN: 978-3-030-32606-7. DOI: 10.1007/978-3-030-32606-7_3. URL: https://doi.org/10.1007/978-3-030-32606-7_3.

[66] Boris Weisfeiler and Andrei Leman. “The reduction of a graph to canonical form and the algebra which appears therein”. In: *NTI, Series* 2.9 (1968), pp. 12–16.

[67] Keyulu Xu et al. *How Powerful are Graph Neural Networks?* 2019. arXiv: 1810.00826 [cs.LG].

[68] Pengyi Yang et al. “A review of ensemble methods in bioinformatics”. In: *Current Bioinformatics* 5.4 (2010), pp. 296–308.

[69] Rex Ying et al. “Graph Convolutional Neural Networks for Web-Scale Recommender Systems”. In: *CoRR* abs/1806.01973 (2018). arXiv: 1806.01973. URL: http://arxiv.org/abs/1806.01973.

[70] C Lawrence Zitnick and Sing Bing Kang. “Stereo for image-based rendering using image over-segmentation”. In: *International Journal of Computer Vision* 75.1 (2007), pp. 49–65.

[71] C. Lawrence Zitnick and Sing Bing Kang. “Stereo for Image-Based Rendering using Image Over-Segmentation”. In: *International Journal of Computer Vision* 75.1 (2007), pp. 49–65. DOI: 10.1007/s11263-006-0018-8.

[72] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. “Modeling polypharmacy side effects with graph convolutional networks”. In: *CoRR* abs/1802.00543 (2018). arXiv: 1802.00543. URL: http://arxiv.org/abs/1802.00543.