2023

ANALYZING PULMONARY ABNORMALITY WITH SUPERPIXEL BASED GRAPH NEURAL NETWORKS IN CHEST X-RAY

Ronaj Pradhan

ANALYZING PULMONARY ABNORMALITY WITH SUPERPIXEL BASED GRAPH NEURAL NETWORKS IN CHEST X-RAY

By

Ronaj Pradhan

B.E., Tribhuvan University, 2018

A Thesis Submitted in Partial Fulfillment of
the Requirements for the Degree of Master of Science

Department of Computer Science

Master of Science Program
In the Graduate School
The University of South Dakota
December 2023

The members of the Committee appointed to examine
the Thesis of Ronaj Pradhan
find it satisfactory and recommend that it be accepted.

DocuSigned by:
KC Santosh
99944614C924467...
Chairperson

DocuSigned by:
Rodrigue Rizk
056A76355F814ED...

DocuSigned by:
Doug Goodman
3C0E646BD1404F3...

DocuSigned by:
William Chen
9671140C01924FB...

ABSTRACT

In recent years, the utilization of graph-based deep learning has gained prominence, yet its potential in the realm of medical diagnosis remains relatively unexplored. Convolutional Neural Network (CNN) has achieved state-of-the-art performance in areas such as computer vision, particularly for grid-like data such as images. However, they require a huge dataset to achieve top level of performance and challenge arises when learning from the inherent irregular/unordered nature of physiological data. In this thesis, the research primarily focuses on abnormality screening: classification of Chest X-Ray (CXR) as Tuberculosis positive or negative, using Graph Neural Networks (GNN) that uses Region Adjacency Graphs (RAGs), and each superpixel serves as a dedicated graph node. For graph classification, provided that the different classes are distinct enough GNN often classify graphs using just the graph structures. This study delves into the inquiry of whether the incorporation of node features, such as coordinate points and pixel intensity, along with structured data representing graph can enhance the learning process. By integration of residual and concatenation structures, this methodology adeptly captures essential features and relationships among superpixels, thereby contributing to advancements in tuberculosis identification. We achieved the best performance: accuracy of 0.80 and AUC of 0.79, through the union of state-of-the-art neural network architectures and innovative graph-based representations. This work introduces a new perspective to medical image analysis. The code is available in the 2AI-lab's GitHub repository: https://github.com/2ai-lab/Superpixels-in-graph-neural-network.

Thesis advisor: DocuSigned by: KC Santosh 1D1EB2065003 4B9...

KC Santosh, Ph.D.

ACKNOWLEDGEMENTS

I extend my deepest gratitude to my thesis advisor, Dr. KC Santosh, whose expert guidance, and unwavering support were instrumental in the completion of this thesis. Without his invaluable mentorship, this work would not have been realized. I also wish to acknowledge the contributions of the other esteemed members of my thesis committee, Dr. Doug Goodman, Dr. Rodrigue Rizk and Dr. William Chen. Their active involvement and assistance throughout this process have been highly appreciated. Furthermore, I would like to express my heartfelt thanks to the entire 2AI lab 2022 - 2023 team for their valuable assistance and support during the thesis writing journey. Your collective contributions have been indispensable, and I am truly thankful for your involvement in this academic endeavor.

DEDICATION

I humbly dedicate this thesis to my parents, whose unwavering support has been the cornerstone of my educational journey. Their continuous encouragement has not only provided me with a secure living environment but has also been an invaluable source of emotional sustenance. Their profound influence has been instrumental in shaping the person I am today.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

CHAPTER 1

1. Introduction

Summary: This chapter introduces the research on Tuberculosis, covering its historical context, motivation, and goals. It emphasizes the global impact of TB and the challenges in rapid detection, highlighting the importance of Chest X-Ray and exploring the potential of Graph Neural Networks. The chapter outlines the thesis's objectives, focusing on CXR image segmentation and GNN application methodology, setting the stage for the experimental exploration of GNNs in TB diagnosis.

Key topics: Motivation, goal, and methodology.

1.1. Background and Problem Statement

Tuberculosis (TB) has been intertwined with our species for millennia, potentially spanning several million years and has been an enduring human affliction [1]. Manifestation of TB in the lungs cause gradual degradation in health over a long period of time, with exhaustion, fatigue, airflow obstruction, chronic cough, chest pains, fever and even death in the later stages of sickness. One-third of the world's population harbors the bacterial infection Mycobacterium Tuberculosis (MTB), responsible for human affliction, with the annual tally of new TB cases exceeding 9 million [2]. Despite recent progress made in diagnostic and treatment methods, the worldwide impact of TB remains distressingly high, with 10 million individuals infected and 1.4 million lives lost to the disease in 2019 [3]. The correlation between poverty and TB is well recognized, high rates and frequent occurrence due to overcrowding, poor ventilation and poor hygiene increase the risk of transmission of the disease. Detecting the presence of TB is a crucial task, that requires expert medical practitioners and equipment. However, with a third of the world's population infected, rapid screening and identification of TB is a difficult and expensive deal. Regardless of the advanced screening options individuals lose their life as the result of late detection of TB. On the other hand, encouragingly, this bacterial disease can be effectively treated and cured through consequent administration of proper medication and screening methods. Between cure and identification of TB lies a bridge which can be shortened to avoid untimely deaths and instigate the process to accurately identify and diagnose TB.

Lung abnormality screening through Chest X-Ray (CXR) is a vital diagnostic tool, enabling early detection and intervention for various pulmonary conditions. This noninvasive imaging technique has enabled identifying clinical manifestations in the lungs which in turn has improved healthcare outcomes [4]. TB can be visually identified and characterized in CXR through various radiological features such as cavities, opacities, edges, colors, thickening, contours, and orientations which are required for decision making, in the case of expert practitioner as well as Computer Aided

Diagnosis (CAD) [5][6][7]. Automated radiological screening, algorithms and predictive models have been advancing to detect abnormalities in CXR [8]. Furthermore, the incorporation of Deep Learning (DL) approach such as Convolutional Neural Network (CNN), with CAD has made an effective breakthrough as a potent strategy for mass screening pulmonary TB via the analysis of CXRs[9][10]. Rendering radiologist and medical practitioners obsolete particularly in regions with few of them.

Even though we have rich state-of-the-art literature for pulmonary abnormality screening using CNN with different models and CXRs, they require a large dataset for such achievements. Given the scarcity of medical datasets, it raises the question of whether it is feasible to create models that demand a small amount of data while still maintaining a high level of performance. We explore an alternative perspective that has yet to be thoroughly investigated, the incorporation of GNN presents a promising avenue for addressing the existing problem. By representing images as graphs, this novel approach offers the potential to establish a more comprehensive data framework that capitalizes on the inherent graph structure within the data. This unique methodology holds the promise of advancing enhanced learning capabilities and delivering outcomes through a distinct and potentially more effective data structure.

1.2. Motivation

The motivation for exploring an alternative methodology and data-structure in classifying TB-positive and TB-negative cases stems from the substantial dataset requirements required in training existing state-of-the-art network models. There is an evident gap in the current diagnostic landscape, with the need of huge CXR images to train models for classifying the medical images[11]. As the diagnostic landscape undergoes an evolution, with the advent of new technologies, most graph-based problem-solving applications have been mostly concentrated on areas such as link prediction, bioinformatics, categorizing documents and recommender system. However, graph-structures are prominent in the real world and by leveraging this richer data structure, we want to experiment whether an image represented as a graph can represent more information within an image.

On a personal level, I am driven by a profound desire to venture into new avenues of research, on how pulmonary abnormalities in CXR images could be analyzed through superpixel-based graph neural networks. The prospect of harnessing graph-based learning and graph-structure to enhance the diagnosis of TB and play a pivotal role in the ongoing battle against disease serves as a source of motivation, driving me to commit my research endeavors to this cause.

1.3. Goal

The goal of this thesis is to experiment with new ideas centered around the segmentation of CXR images into groupings of pixels with shared color and other fundamental attributes, giving rise to

what are commonly referred to as ‘superpixels’ which are the basis of the construction of a graph structure. This approach is to facilitate image classification, by harnessing the potential of GNNs, and give the audience insights and outcomes that hold promise for enhancing the interpretation of medical images in a novel and impactful manner. As such, we hope to experimentally examine the effects of different graph structures on GNNs as well as find how graph structures could aggregate information through convolutional architectures and attention architectures, using GNNs.

1.4. Contribution

The accuracy of disease identification relies on data exploration and investigation using CXR scans, crucial for TB classification research. We utilize datasets sourced from publicly available databases[12][13], including CXR scans and lung boundaries outlined by expert radiologists. It is noteworthy that, out of the total dataset comprising of 811 CXR images, 359 are labeled TB-positive and 452 are deemed TB-negative.

For graph data creation, we employ the SLIC algorithm, dividing images into superpixels based on pixel similarity and proximity. A Region Adjacency Graph (RAG) is constructed for datasets containing 75, 200, 300, and 400 superpixels. These datasets serve as nodes for subsequent analysis via Graph Neural Network (GNN) models.

Two GNN models: Graph Convolutional Network (GCN) and Graph Attention Network (GAT) are employed to perform deep learning tasks on graph-structured data. Both models are designed for node classification, link prediction and graph classification but have distinct differences in their architecture and mechanisms.

Out of the two GNN models used, the GCN architecture comprised an initial embedding layer with two node features (pixel intensity and coordinates) or one (coordinates) along with edge connections between the nodes. This is followed by 4 layers of GCN layers stacked, each with 146 input and output channels. A readout layer, consisting of a Multi-Layer Perceptron (MLP) and a mean operation, is used for graph classification. In our case, we employ a mean readout layer consisting of three linear layers, ultimately classifying them into two categories. Similarly, the GAT architecture also starts with an embedding layer and is followed by 4 GAT layers, each with 152 input and output channels. Each GAT layer is accompanied by a LeakyReLU activation function, batch normalization, and a mean readout layer.

1.5. Thesis outline

The remainder of this thesis is structured as follows:

- Chapter 2: Description of related papers, research reports and articles about image segmentation, processing image data into graph domain, graph neural network, its types, and architectures.

- Chapter 3: Introduction to the dataset used in our experiment.
- Chapter 4: Explanation about the implementation of representing images as graph, superpixel generation, datasets with varying superpixel and network models.
- Chapter 5: Experimental setup, results and analysis are described in this chapter.
- Chapter 6: Talks about conclusion and future work.

CHAPTER 2

2. Literature Review

Summary: This chapter contains an overview of related works in the direction of GNN their background and brief description of convolutional architectures and attention architecture in GNN, brief literature review of image segmentation, graph representation and superpixel algorithm.

Key topics: Image Segmentation, Superpixel, Introductory works, convolutional architectures, attention architectures.

2.1. Background

In computing, an image is essentially a two-dimensional array of pixels, where each pixel corresponds to a specific location in the image and embodies a distinct color. The expression of these colors is articulated through numerical values, employing established color models like RGB (red, green, blue) and HSV (hue, saturation, value). For example, in the RGB color mode, a pixel is a 3-tuple of (R, G, B) where each value ranges from 0 to 255.

An essential and foundational undertaking in computer vision involves image classification, wherein the objective is to assign labels to a set of pixels, categorizing them into specific groups. Despite its simplicity for humans, this task has posed a persistent challenge for computers. CNNs have emerged as a highly effective method for image classification, demonstrating notable success in addressing this computational challenge. The CNN was first introduced with the LeNet Model, which demonstrated its capabilities in classifying handwritten characters [14]. However, CNN began solidifying its presence as a highly successful technique for image classification after ImageNet Large Scale Visual Recognition Challenge [15] in 2012 using a deep learning approach. There have been experiments with different iterations and variations of these models ever since.

GNNs constitute a segment within the broader domain known as geometric deep learning [16] . Geometric deep learning focuses on extending successful deep learning methodologies to non-Euclidean data structures, such as graphs and manifolds. The motive behind this stems from the accomplishments of established deep learning techniques, including CNNs and the prevalence of data formatted as graphs and manifolds. The existing shortfall of effective deep learning techniques directly applicable to such non-Euclidean data further motivates exploration in this domain. GNNs specifically pertain to the application of geometric deep learning principles to graph data.

2.2. Graph Neural Networks

2.2.1. Basics: GNN

The utilization of neural networks on graph data requires the models to effectively handle graph matrix representations, such as adjacency matrices. Moreover, these models must exhibit invariance to input permutation, ensuring that the network can process the graph nodes in any order. The primary objective of many GNN models is to acquire a node embedding for each node, a process that may involve multiple layers and diverse types of layers. Ultimately, these node embeddings are aggregated for graph classification, forming a comprehensive graph embedding. While much of the existing research on GNNs has concentrated on node classification, these architectures can readily be adapted for graph classification. This adaptation typically involves an additional step of aggregating node embeddings and subsequently classifying the graph embedding, often accomplished with an MLP.

The initial endeavors to extend neural networks to graph structures can be attributed to the work of *Scarselli et al.* [17] [18][19]. Those works are a series by the same authors, with the most mature models presented in [18], which is also referred to as vanilla GNN in literature. This model is specifically designed for undirected homogeneous graphs, accommodating both node and edge features. The model introduces the concept of learning node embeddings through a process called message passing, where each node communicates its information (messages) to its neighbors. The received information from neighbors is aggregated and utilized to update the node embedding. This iterative process is performed multiple times, expanding the radius of information exchange for each node. This is achieved by recursively applying the same set of weights until a stable state has been reached. These first generation of GNNs are also known as Recurrent Graph Neural Networks (RecGNN) [20].

The interest in non-Euclidean deep learning has recently surged in computer vision after the seminal work of *Bruna et al.* [21] which it explores the potential generalizations of CNNs to signals defined on more expansive domains without relying on the translational group's action. Two novel constructions are proposed: one based on hierarchical clustering of the domain which effectively captures hierarchical structures within the signal and the other being spectrum of graph Laplacian which enables the graph structure to train convolutional layers which is independent of the input size. A significant advancement in the field was achieved by *Kipf and Welling* [22], who introduced rudimentary filters designed to operate on 1-hop neighborhoods of a graph. These filters incorporated a message passing mechanism to construct node representations, leveraging the aggregation of local information from adjacent nodes.

GNNs have never been popular for image classification. *Di Massa et al.* [23] were among the early researchers to transform images into graphs to conduct image classification using a GNN. This undertaking was driven by an experimental motive to compare the efficacy of a recursive neural

network with that of a GNN when applied to graph-structured data. Their study utilized a subset of the Caltech benchmark dataset *Fergus et al.* [24] ,comprising four distinct image classes (bottles, camels, guitars, and houses). The primary objective was to observe whether the variance in input data representation between recursive and graph neural networks had an impact on performance in real-world applications. Expanding upon the groundwork laid by *Di Massa et al.* [23] *Quek et al.* [25] assessed the effectiveness of a GNN, specifically on the model proposed by *Scarselli et al.* [18], on diverse graphs derived from the same set of images. Their investigation involved the creation and comparison of four distinct graphs: a 4-connected uniformly sampled grid, an RAG, the minimum spanning tree (MST), and the Delaunay triangulation graph.

In the realm of image segmentation, *Shi and Malik* [26] undertook a pioneering endeavor by directly applying graph-based techniques to images. They conceptualized each pixel as a distinct graph node, intending to facilitate image segmentation. Despite the inherent potential of this approach, the realization of precise segmentation outcomes proved to be a persistently challenging task. An intermediary solution surfaced through the introduction of the concept of superpixels [27]. Superpixels involve the grouping of pixels that share similarities in color and other low-level properties, such as spatial proximity, into perceptually meaningful representations [28]. These over-segmented and simplified images, resulting from the superpixel process, find applicability across various tasks in computer vision, including but not limited to image classification [29].

Monti et al. [30] pioneered the application of GNNs to the domain of image classification. Their seminal contribution, the MoNET framework, was specifically designed to address geometric data, featuring the integration of a scale factor to account for geometric distance during neighborhood aggregation. In their initial experiments, the authors applied GNNs to the MNIST dataset [31], where the images were preprocessed by segmenting them into superpixels for the subsequent image classification task.

Avelar et al. [27] introduced RAG-GAT, a novel approach that involved segmenting input images into superpixels and constructing an RAG by establishing connections between each region and its neighbors. This RAG was then fed into a GAT. While RAG-GAT exhibited superior performance compared to other GNN models on grayscale images, it faced challenges when applied to three-channel RGB images. The diminished accuracy in the latter case was attributed to the forced connection between adjacent areas, leading to the aggregation of unnecessary information during the processing of RGB images.

Following this, *Matthias et al.* [32] introduced Spline CNN, which constitutes a graph convolutional operator reliant on the B-spline kernel. This specialized operator was designed to facilitate the extraction of graph features, thereby enabling the establishment of an end-to-end deep GNN. The incorporation of the B-spline kernel in the graph convolution process underscored the model's capacity to effectively capture and process intricate patterns within graph-structured data.

Ushasi et al. [33] employed a preprocessing strategy incorporating multi-scale superpixels for image data. In this approach, the images underwent a multi-scale superpixel segmentation as an initial step. Subsequently, the authors applied the GCN algorithm, as proposed by [22], within the context of a Siamese graph convolutional network framework. This integrated model was specifically crafted for image retrieval tasks, leveraging the benefits of multi-scale superpixel preprocessing and the expressive power of the GCN algorithm to enhance the retrieval performance in the realm of image analysis and retrieval.

Boris et al. [34] introduced a novel graph attention pooling technique and conducted a comparative analysis with existing methods, including the Graph Isomorphism Network (GIN) [35] and GCN on three datasets, two of which were self-generated to demonstrate attention and generalization in GNN which were color, triangles, and the Mnist-75 dataset. The 75 in the Mnist-75 dataset refers to the number of superpixels extracted from each image in the dataset. The study specifically delved into evaluating the anti-noise capabilities and robustness of various pooling methods. This assessment involved the introduction of random Gaussian noise to the Mnist-75 dataset, allowing the researchers to gauge the performance of the graph attention pooling technique in the presence of noise. Subsequently, in a related work, *Boris et al.* [36] proposed a Hierarchical Multigraph Network (HMN). This innovative network architecture was designed to process superpixels at different scales as input, aiming to amalgamate information from various granularities for comprehensive image classification. By leveraging hierarchical structures and incorporating information from superpixels of diverse scales, the HMN model demonstrated a holistic approach to image classification, contributing to the advancement of techniques in the field.

In a more recent study, *Dwivedi et al.* [37] conducted a comprehensive comparison of various contemporary GNNs including Graph Convolutional Networks (GCN) [22], Graph Attention Networks [38], MoNet [30], Gated Graph Convolutional Networks [39], Graph Isomorphism Networks [35] and GraphSage [40]. Their primary objective was to establish a framework for a fair and systematic comparison of GNNs, involving evaluations across diverse datasets.

As part of this comprehensive evaluation, [37] specifically focused on image classification. They transformed the CIFAR10 [41] and MNIST [31] datasets into graph representations, utilizing them as baseline performance metrics. The underlying assumption was that GNNs would exhibit proficiency in handling these datasets, rendering them suitable for use as a 'sanity-check' in the evaluation process. This meticulous assessment aimed to discern the relative strengths and weaknesses of different GNN architectures across varied datasets and contribute to a deeper understanding of their applicability and performance characteristics.

2.2.2. Convolutional Architectures

Convolutional Graph Neural Networks (ConvGNNs) have become increasingly popular in recent years due to their efficiency, compositional convenience, and notable alignment with CNNs [20]. These architectures are broadly classified into spectral-based or spatial-based models, depending on how graph convolution is defined, an inherently more intricate task for graphs than for images. Spectral approaches within ConvGNNs draw inspiration from graph signal processing, interpreting convolution to smooth the graph signal and effectively reduce noise [42]. These methods rely on the eigen decomposition of the Laplacian matrix and utilize the graph Fourier transform to manipulate the signal. This process involves convolving the signal in the spectral (frequency) domain using adaptable and learnable filters.

In contrast, Spatial Convolutional Graph Neural Networks (ConvGNNs) bear a closer resemblance to CNNs. They derive graph convolutions by considering a node's spatial relations, resembling direct information propagation within the vertex neighborhood [20]. While this spatial approach aims for computational efficiency (by avoiding eigen decomposition) and improved generalization (independence from a specific eigen basis), articulating convolution for neighborhoods of varying sizes while preserving local invariance presents significant challenges in this context.

One of the contributions in the field of ConvGNNs is detailed in [21], introducing the spectral network architecture and fundamental principles associated with the spectral analysis of graphs. This encompassed Laplacian eigen decomposition, interpreting convolution as a means of signal smoothing, and integrating learnable localized filters featuring multiple channels. The work also provided a spatial interpretation. Despite its pioneering nature, this initial endeavor faced limitations [43] [20] primarily linked to the exact computation of graph eigendecomposition. Any modification to the graph structure would influence the eigenbasis, leading to challenges in stability and generalization. Moreover, these filters lacked spatial localization and the eigen decomposition process itself posed a computational challenge with a complexity of $O(n^3)$.

Subsequent works on spectral ConvGNNs sought to simplify the ideas from [21], introducing approximations that not only reduced computational complexity but also enhanced regularization. For instance, the work in [44] proposed the concept of approximating convolutional filters through truncated expansion using a K-th order Chebyshev polynomial basis. ChebNet [45] implemented this idea for ConvGNNs, offering the advantage of K-localized convolution, as it represented a K-th order polynomial in the Laplacian. This allowed filters to extract local features independently of graph size and eliminated the need for computing the entire eigenvectors of the Laplacian, significantly reducing the computational burden. A more substantial simplification was presented in the influential work [22] as the GCN. This architecture constrained the layer-wise convolution to $K = 1$, addressing issues of overfitting on local neighborhood structures in graphs with wide degree distributions. This first-order approximation gained popularity in ConvGNNs due to its simplicity.

Spatial approaches define convolutions directly using the topological structure of the graph and draw inspiration from the convolution operations performed in an Euclidean space, like those employed in classical CNNs. In Euclidean convolution, the process can be conceptualized as a weighted aggregation of function values. For instance, when applying convolution to an image with a 3x3 kernel, the resulting value at a specific point in the image is derived from a weighted sum of pixel values at that point and its eight surrounding pixels. Spatial methods endeavor to emulate this convolutional process in the context of a graph. In the graph-based scenario, the value at a node post-convolution, represented as a feature vector, is determined by a weighted aggregation of the feature vectors belonging to its neighboring nodes, in conjunction with its own feature vector. This mechanism ensures that information is propagated and aggregated across the graph, capturing contextual dependencies like the spatial relationships considered in Euclidean convolution on images.

The Neural Network for Graphs (NN4G) [46], introduced coincidentally with the conventional GNN employs a compositional layered architecture to acquire and model graph dependencies. In this architecture, the convolution operation directly assimilates the informational content from a node's neighborhood. To enhance information flow and facilitate the integration of insights from distinct layers, residual and skip connections are strategically incorporated. These connections serve to harness information from various network depths, contributing to a more robust and expressive representation of graph-based structures.

The Diffusion Convolutional Neural Network (DCNN) [47] conceives graph convolution as an intricate diffusion process, here information flows between nodes guided by a defined transition probability, ultimately reaching an equilibrium state that fosters a well-balanced distribution of information. This process is designed to reach an equilibrium, facilitating a balanced distribution of information. Furthermore, this approach captures the evolving information dynamics across the graph during the diffusion process, providing a comprehensive representation. Diffusion Graph Convolution (DGC) [48], however similar in concept diverges in its aggregation strategy by summing up the outputs of each diffusion step instead of concatenating them. This is beneficial in scenarios where immediate neighbors exhibit a great importance but implies limited contribution from distant neighbors. Such limitation could be a drawback in applications characterized by extensive dependency chains, such as those encountered in biological contexts.

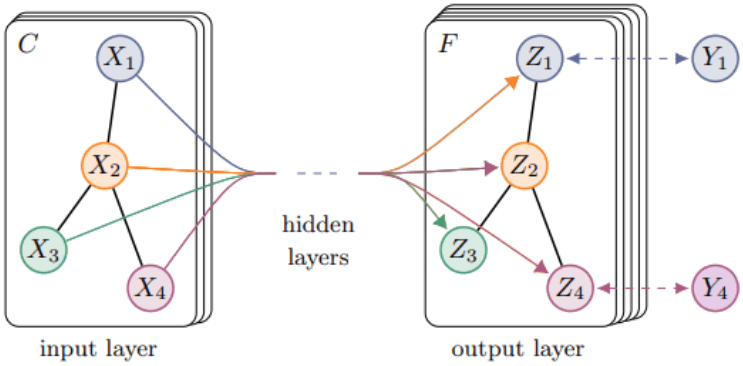

Figure 1 Convolutional Architecture [22].

2.2.3. Attention Architectures

The attention mechanism [49] [50] [51] serves as an adaptive weighting scheme within neural networks, assigning higher weights to more important input features. This stands in contrast to conventional convolutional architectures, where either identical contribution from neighbors to the central node are assumed (as in GraphSAGE [40]), or predetermined weights are applied (as in GCN [20]). A notable variant of attention is self-attention[50], frequently employed in GNNs. In self-attention, input features attend to themselves, discerning and highlighting the most crucial elements. Attentional architectures inherently possess a spatial nature, as they operate within the node neighborhood to assess the significance of neighboring elements. This adaptive weighting mechanism allows the model to dynamically focus on relevant information within the local context, enhancing its ability to capture and leverage important features in a more nuanced and context-aware manner compared to traditional convolutional approaches.

The GAT [38] introduces a self-attention mechanism, implemented as a single-layer feedforward neural network, to assess and weight node neighbors. To enhance the learning process [43] and augment model capacity [20], GAT incorporates multi-head attention [51], employing k independent attention mechanisms in parallel. The outputs of these mechanisms are either concatenated or averaged to produce the final node embedding.

This approach effectively addresses the challenge posed by varying node degrees, utilizing learned weights to mitigate issues associated with wide node distributions. However, a notable drawback lies in the potential for highly fluctuating predictions and elevated standard deviations due to the inherent weighting mechanism [52].

Recognizing the limitation of GAT in maintaining conditioned rankings of attended nodes, the authors [53] propose the GATv2 architecture. In GATv2, each node can attend to any other node, offering increased resilience to structural noise, such as minor changes in the graph structure. Interestingly, GATv2 achieves comparable or superior performance to the original GAT with

fewer attention heads. Notably, this structural enhancement incurs no additional computational cost, preserving the computational efficiency and advantages of the original GAT.

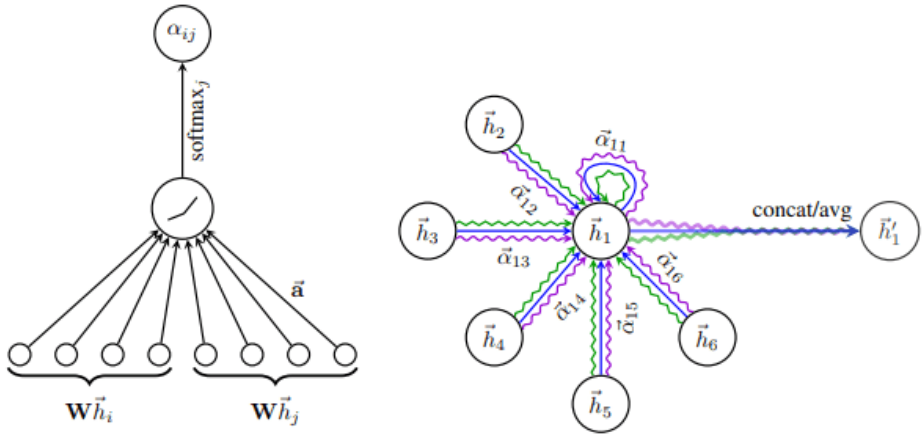

Figure 2 Attention Architecture [38].

2.2.4. Readout Layer

The readout layer performs a flat aggregation of node representations, condensing them into a singular graph embedding vector. This process is alternatively referred to as global pooling [54], global aggregation [35], or direct pooling [55]. In essence, the readout layer consolidates information from individual nodes to construct a comprehensive and concise representation of the entire graph. Fundamental and widely adopted readout techniques include mean, max, and sum operations [54] [55]. These operations are typically applied as the final layer preceding an MLP. Notably, according to findings in [35], the sum operation emerges as the most potent among the three. The superiority of the sum operation lies in its capacity to discern certain structural patterns in graphs that may elude detection by mean and max aggregators.

Alternatively, a weighted average can be employed, leveraging attention mechanisms to compute weights. This variant, referred to as global (soft) attention[54][56], allows for a more nuanced aggregation of node representations, with the attention weights serving as a mechanism to discern the relative importance of individual nodes in contributing to the overall graph embedding.

2.3. Image Segmentation

Conventionally, images are depicted through a 3-dimensional tensor, capturing the RGB color intensities of each pixel. This representation is integral to image classification using CNNs. However, GNNs work with data organized as graphs, so they require a different representation compared to the tensor used for images. Consequently, there arises the need to transform the tensor representation of an image into a suitable graph structure for effective utilization within GNNs.

A natural representation of an image in a graph format involves treating each pixel as a node and establishing edges to denote connections between pixels. These edges might be formed based on proximity or they could connect pixels sharing similar color intensities. Moreover, nodes have the flexibility to represent various aspects of an image; instead of individual pixels, they could encapsulate a region of pixels [57]. The graph can take on various characteristics, including being directed or undirected, and featuring edges with or without attributes. This diversity results in multiple potential graph representations for an image, and the impact of these distinct representations on the model remains uncertain. The choice of how to structure the graph introduces a level of ambiguity, as the selection of nodes, edges, and their attributes could influence the subsequent modeling outcomes.

2.3.1. Related Works

Di Massa et al. [23] conducted experimentation utilizing both undirected and directed graphs. Initially, undirected graphs, denoted as RAGs were generated. In these graphs, nodes corresponded to homogeneous image regions, and edges were defined based on the adjacency relationships between these regions. Nodes were endowed with attributes such as area and perimeter, while edge attributes encapsulated mutual orientation and average color differences between the respective regions. Subsequently, directed graphs, specifically directed acyclic graphs, were derived from the RAGs by identifying the node containing the central pixel of the image. The edges in these directed graphs were then directed based on the regions that were reachable from the central node. Notably, the RAG was tailored for integration with GNNs, while the recursive neural network was designed to operate on the directed acyclic graph. In terms of model performance, discerning the factors contributing to the disparate outcomes between the GNN and the recursive neural network remains an open question, necessitating further investigation to elucidate whether the observed differences are attributable to inherent model characteristics, or the specific nuances of the graph structures employed.

Quek et al. [25] specifically directed their attention towards comprehending the impact of diverse graph structures on GNNs. Their investigation involved a comparative analysis of the performance of a GNN, as proposed by *Scarselli et al.* [23] across four distinct graph configurations. These configurations encompassed a 4-connected uniformly sampled grid, a Region Adjacency Graph (RAG) proposed by *Quek et al*. [25], the Minimum Spanning Tree (MST) introduced by *Prim* [58] , and the Delaunay triangulation graph pioneered by *Delaunay et al*. [59]

The construction of these graphs involved the initial delineation of regions of interest within an image, each serving as a node. For the 4-connected grid, regions were defined as circles encompassing every 16th pixel in both row and column directions. In the RAG, regions were formulated utilizing the edge flow technique introduced by *Ma and Manjunath* [60]. In the case of both the MST and the Delaunay triangulation graph, the Hessian-Laplace scale-invariant detector, as proposed by *Mikolajczyk et al.*[61] , was employed to identify and create these regions.

After defining the regions, the establishment of edges followed a specific protocol for each graph structure. In the 4-connected grid, regions were linked to those situated above, below, to the left, and the right of a given region. In the RAG, edges were formed between adjacent regions. The MST connected all nodes in a way that minimized a specified cost, where the cost was determined by the Euclidean distance between the centers of regions [25]. Finally, the Delaunay triangulation created connections between the central points of each region. This involved the generation of triangles such that no point resided within the circumcircle connecting all corners of a triangle.

Across all graph structures, the node attributes encompassed the coordinates of the region's center, the scale of the region, and a label assigned by a 128-dimensional scale-invariant feature transform (SIFT) as introduced by *Lowe* [62] . Conversely, the edge attributes included the normalized Euclidean distance between nodes, the normalized scale difference, and the normalized Euclidean distance between the descriptors (SIFT).

Dwivedi et al. [37] initiated the graph creation process by segmenting an image into superpixels through the employment of the SLIC technique developed by *Achanta et al.*[57]. These superpixels, representing regions of homogeneous intensity, constituted the nodes of the graph. The edges of the graph were subsequently established by connecting each node to its k nearest neighbors, with k set to 8.

[23] [37] [25] employ diverse graph structures in their respective experiments, highlighting the absence of a universally standardized method for constructing a graph from an image. To be more specific, the impact of various graph structures on model performance remains uncertain, leaving the question of which structures are more effective or less effective unanswered. This lack of uniformity is anticipated, considering that the construction of a graph, modeling an image, allows for flexibility and choice in the approach taken.

2.3.2. Superpixel Algorithm

The term "superpixel" was introduced by *X. Ren and J. Malik et al*. [63]. The fundamental concept involves the grouping of similar pixels into larger entities referred to as superpixels, thereby establishing a more coherent representation of the image. Additionally, this process significantly reduces the number of elements that subsequent image processing algorithms need to handle. Superpixels represent a specific instance of image segmentation, a technique executed based on intrinsic image features like color [64], texture [57], or edges [65].

There is no definitive set of criteria that must be met for a segmentation algorithm to qualify as a superpixel algorithm, certain characteristics are commonly regarded as desirable. *D. Stutz et al.* [28], such as partition, connectivity, boundary adherence, compactness, and controllable number of superpixels. Spectral Linear Iterative Clustering (SLIC), introduced by *Achanta et al.* [57], stands out as one of the widely embraced superpixel algorithms. This algorithm employs a variation of k-means clustering to generate superpixels by emphasizing color similarities within

an image. In comparison to alternative algorithms existing at its introduction, SLIC demonstrates commendable performance in terms of both segmentation quality and speed. Due to the algorithm's inherent simplicity, numerous strategies have been proposed to enhance its performance. Instances of such improvements include approaches like S-SLIC [66] and gSLIC [67], both of which effectively facilitate real-time computation of SLIC superpixels. gSLIC demonstrates compelling outcomes by achieving a notable acceleration of the fundamental SLIC algorithm, reaching up to 20 times faster performance through the implementation of hardware acceleration. An alternative approach to superpixel segmentation is presented by *Van den Bergh et al.* in [68]. Their method, known as Superpixels Extracted via Energy-Driven Sampling (SEEDS), can achieve real-time performance by employing a coarse-to-fine strategy. The pixels undergo iterative assignment to superpixels, optimizing a designated energy function that considers both the color and shape of the superpixels. Notably, to maintain efficiency, the number of computations is minimized by confining superpixel updates to border pixels. Inspired by SEEDS in [69] *Yao et al.* incorporated an enhanced regularization term that leads to the creation of more uniformly shaped superpixels, all while preserving high processing speed.

CHAPTER 3

3. Superpixel-based GNN (Implementation)

Summary: This section provides a thorough exploration of datasets used in this thesis and the preprocessing steps required to transform Euclidean data, like images, into a graph structure. It covers methodologies and associated hyperparameters for generating these graphs, with a specific focus on the critical generation of superpixels at different levels. Additionally, the section delves into the embedding layer, GNN layers, and prediction layer, offering insights into their collaborative role in interpreting input data within the model. Overall, it serves as a comprehensive guide to representing Euclidean data in graph structures, detailing methods, hyperparameters, and crucial components in this transformative process.

Key Topics: Data Preprocessing**,** Superpixel generation, Graph representation, Embedding layer, GNN layer, Prediction layer.

3.1. Data Preprocessing

Radiological imaging, such as CXR scans, plays a pivotal role in early detection and exploration of diseases linked to the respiratory system. In this thesis, we explore a comprehensive dataset of TB CXR images, that serves as a valuable resource for researchers and machine learning practitioners. For our research, we curated our dataset from two publicly available and accessible databases. The selection of these databases was driven by the need for reliable sources of medical imaging data that align with the objectives of our study. By leveraging these databases, we aim to ensure the robustness and credibility of our research outcomes in the context of TB detection and diagnosis.

Table 1 Dataset Summary

Dataset	**Positive Set**	**Negative Set**	**Size in MB**
Montgomery Set	58	80	38.9
Shenzhen Set	287	279	221
Belarus Set	107	-	5.66
Total	452	359	38.05

1) NLM Dataset: We opted to utilize the reputable National Library of Medicine (NLM) [13] dataset which has made two datasets available for research in CAD of pulmonary disease especially with a focus on pulmonary TB. This database was established through a collaborative effort between the NLM and Shenzhen No.3 People's Hospital, comprising two distinct sets of CXR images: Montgomery County chest X-ray set (MC) and Shenzhen chest X-ray set. The Montgomery County CXR dataset comprises 138 CXR derived from Montgomery County's TB screening initiative. Within this dataset, there are 80 instances of normal CXR and 58 instances revealing manifestations of TB. In contrast, the Shenzhen CXR dataset presents a more extensive collection of 662 CXR, featuring 326 normal cases and 336 cases exhibiting TB manifestations. Both datasets categorize patients into one of two distinct medical conditions: (a) a state of normalcy or (b) the presence of TB-related manifestations. These datasets are enriched with meticulously crafted gold-standard segmentations of the CXR, that were painstakingly generated under the expert guidance of radiologists, ensuring a high level of precision, accuracy, and reliability. For our case study, we have chosen to frame the problem as a binary classification task, with a specific focus on distinguishing between two classes: (0) normal cases and (1) cases displaying pulmonary TB manifestations. This approach simplifies the problem into a clear-cut non-TB vs. TB classification, thereby facilitating a more focused and rigorous analysis and interpretation of the data.

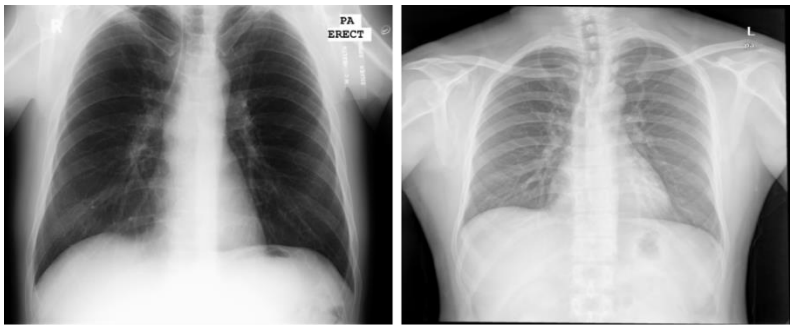

Figure 3 Chest X-ray from samples, one per dataset (left to right): Montgomery County (USA), Shenzhen (China).

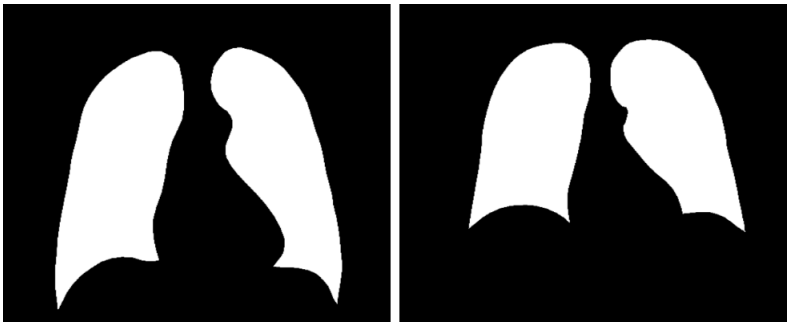

Figure 4 Chest X-ray mask from samples, one per dataset (left to right): Montgomery County (USA), Shenzhen (China).

2) Belarus Dataset: The Belarus dataset [12] was compiled as part of a pivotal drug resistance investigation, by the collaborative efforts of the National Institute of Allergy and Infectious Diseases and the Ministry of Health in the Republic of Belarus. Every image within this distinctive database was meticulously selected due to its affiliation with TB infection. The dataset encompasses a total of 306 CXR images, derived from a cohort of 169 patients. It is important to highlight that, despite the initial pool of images, a subset of 107 CXR images from the original pool was ultimately incorporated into the study. This careful selection process was guided by the availability of the dataset at that specific juncture.

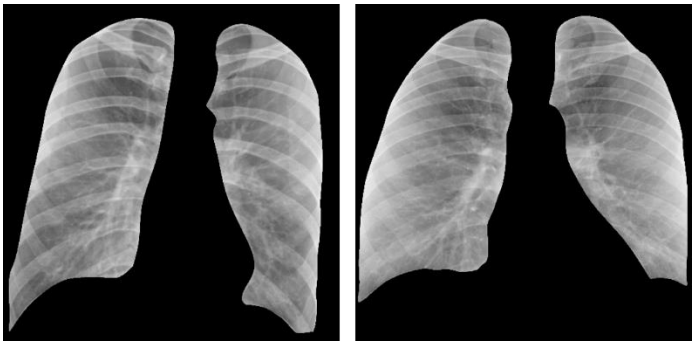

Figure 5 Chest X-ray segmented dataset obtained from Belarus Dataset

In total, there are 811 total CXRs available with 359 CXRs manifested with TB and 452 normal CXRs.

We have compiled a dataset comprising a total of 811 grayscale CXR images, each exhibiting varying dimensions. These images were sourced from three distinct datasets: the Shenzhen dataset, featuring dimensions of 3000 x 3000, the Montgomery dataset, characterized by dimensions of 4020 x 4892, and the Belarus dataset, which presented dimensions of 2248 x 2248. Each of these CXR images has undergone annotation to indicate the presence or absence of TB, with binary labels assigned as 0 for TB absence and 1 for TB presence. To enhance the dataset's utility, we undertook a preprocessing phase that involved overlaying lung boundary masks onto the CXR images, effectively outlining the region of interest (i.e., isolating the lung area within the CXR). Furthermore, we uniformly resized all CXR images to a resolution of 512 x 512 pixels and applied a superpixel algorithm to these images. This algorithm groups pixels into coherent regions, thereby replacing the rigid pixel grid structure. This strategic approach enables the capture of more meaningful image features, augmenting the overall image analysis process.

For the generation of superpixels within each image, we employed the SLIC algorithm [57]. These superpixels were subsequently organized into five distinct sets, each featuring a different count of superpixels, specifically 75, 200, 300, and 400. Each of these datasets has sizes of about 811 graphs obtained from the 811 CXR images. This diverse range of superpixel counts provides multiple options for subsequent analyses. In contrast to representing individual (512 x 512) pixels as nodes,

we have chosen to follow the approach outlined in [45] [38], where we experiment with the number of superpixels to represent nodes and the edges between them. This methodology offers a more adaptive and informative representation for the subsequent analysis of our dataset.

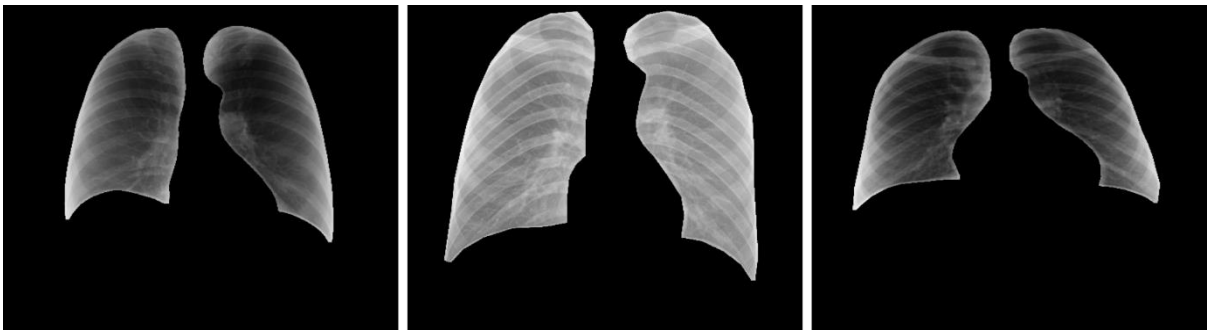

Figure 6 Chest X-ray Region of Interest (ROI) obtained after overlaying mask of the lungs onto the chest X-ray image, one per dataset (left to right): Montgomery County (USA), Shenzhen (China) and Belarus.

Table 2 Superpixel Data Summary

Dataset	Number of Nodes	Number of Edges	Total Samples	Size in MB
LUNGS_75	$58 \leq n \leq 75$	$118 \leq e \leq 150$	811	18.2
LUNGS_200	$169 \leq n \leq 200$	$389 \leq e \leq 400$	811	70.5
LUNGS_300	$262 \leq n \leq 300$	$522 \leq e \leq 600$	811	142
LUNGS_400	$358 \leq n \leq 400$	$714 \leq e \leq 800$	811	233

3.2. Superpixel Generation

There are many ways to generate graph structured data of images, in alignment with prior research in this field, we employ the SLIC algorithm [57] to generate the superpixels in the image and obtain good quality segmentations within a reasonable amount of time. The algorithm starts by sampling several points in the image, placed with distance:

$$S = \sqrt{N/k}, \tag{1}$$

where “N” and “k” represent the total number of pixels in the image and the desired quantity of superpixels, respectively. Each of these points i.e., “S” is termed "superpixel centers," corresponds to a designated superpixel in the resulting image. To establish these superpixel centers, an initial placement is determined. The placement strategy involves selecting positions with the smallest

gradient values within a local 3x3 neighborhood for each superpixel center. This step is crucial in preventing the initialization of superpixels along object edges. By relocating the superpixel centers to positions characterized by minimal gradient values within their immediate surroundings, the algorithm ensures that the superpixels are initiated away from high-gradient regions, typically corresponding to object boundaries. This approach helps in achieving a more accurate and visually coherent segmentation, as the superpixels are less likely to span across different objects or exhibit irregular shapes along prominent edges.

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 X 3 neighborhood.
Set label l(i) = -1 for each pixel i.
Set distance d(i) = ∞ for each pixel i.
repeat
    for each cluster center C_k do
        for each pixel i in a 2S X 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
    Compute new cluster centers.
    Compute residual error E.
until E ≤ threshold
```

Algorithm 1 summarizes the SLIC superpixels algorithm which uses a distance metric to determine the similarity between pixels and cluster centers. The distance measure (D) is crucial in assigning pixels to clusters. In SLIC, D is defined as a combination of color proximity (D_c) and spatial proximity (D_c), normalized by their respective maximum distances within a cluster (N_c and N_s). The color proximity (D_c) is calculated using the CIELAB color space, measuring the difference in color values between pixels. The spatial proximity (D_c) accounts for the position of pixels in the image plane, considering their x and y coordinates. To avoid inconsistencies in clustering behavior for different superpixel sizes, D_c and D_s are normalized by their maximum distances within a cluster. The final distance measure (D) is obtained, allowing for a balanced consideration of color and spatial proximity in the formation of SLIC superpixels. This approach ensures that SLIC superpixels adapt to both color and spatial characteristics in an image, leading to more meaningful and context-aware segmentation results.

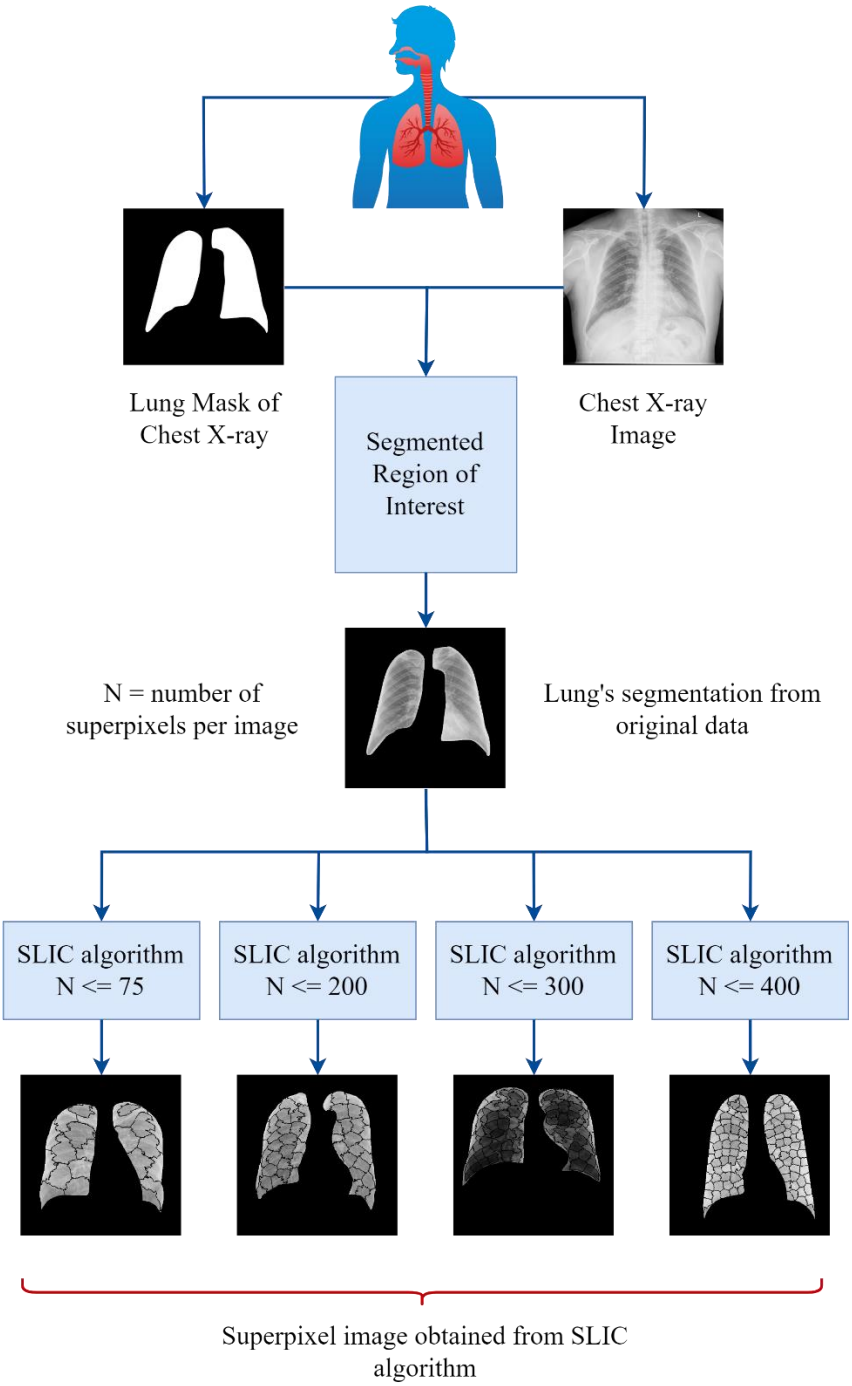

Figure 7 Segmented CXR images obtained through SLIC segmentation, varying the value of "k" to obtain superpixels with different range of node values.

3.3. Image as graph

GNNs are specifically designed to operate on graph-structured data, where information is represented as nodes and edges in a network. Generating a graph structure from other data types, such as images, is essential when employing GNNs for several reason:

Modeling Relationships: GNNs excel in capturing relationships and dependencies within data. By representing the data as a graph, one can explicitly model the connections and interactions between different elements. This is particularly valuable when dealing with data where relationships are not explicitly defined in a regular grid or sequence.

Non-Euclidean Data: GNNs are well-suited for handling non-Euclidean data, where the relationships between elements do not adhere to a traditional geometric structure. Examples of non-Euclidean data include social networks, molecular structures, or, as in our case, medical images where pixel relationships might not follow a regular grid.

Incorporating Local Context: GNNs leverage the local neighborhood information of each node to make predictions. By constructing a graph structure, you can capture the local context of each element, allowing the model to consider the information from nearby nodes and edges.

Flexibility in Representation: Graph structures offer flexibility in representing complex relationships and hierarchies in the data. This enables GNNs to capture intricate patterns that might be challenging for other architectures.

In the context of CXR images, sourced [13] and [12], transforming the pixel-based image data into a graph structure allows the GNN to exploit the inherent relationships between pixels, facilitating the detection of abnormalities or patterns relevant to medical diagnosis.

This thesis primarily aims to investigate the correlation between the quantity of superpixels in an image and the corresponding effectiveness of GNN models in accurately classifying CXR images as either TB positive or TB negative. To achieve this objective, we curated four distinct datasets, each characterized by a different number of superpixels (nodes). The datasets are denoted as follows: LUNGS_75, LUNGS_200, LUNGS_300, and LUNGS_400. The nomenclature signifies the varying number of superpixels employed in each dataset, achieved through the SLIC segmentation technique.

Through the systematic experimentation on these diverse datasets, our objective is to uncover complex patterns and trends in GNN performance, particularly in relation to the varying number of superpixels. This research endeavors to yield valuable insights into optimal configurations that enhance the GNN's efficiency in classifying CXR images as TB-positive or TB-negative. To achieve this goal, we introduce variability in the number of superpixels across all five datasets. This variation results in a spectrum of nodes for each dataset, where the maximum node value adheres to a predefined set number, hence, the graphs are topologically different from each other.

This deliberate variation enables a comprehensive exploration of the impact of superpixel quantity on GNN performance, providing insights into the relationship between image granularity and the GNN's diagnostic capabilities in medical imaging. From the hierarchy of 75, 200, 300 and 400 SLIC superpixels the resultant graphs of the train set, test set, and validation set are formed with each set having a varying number of nodes.

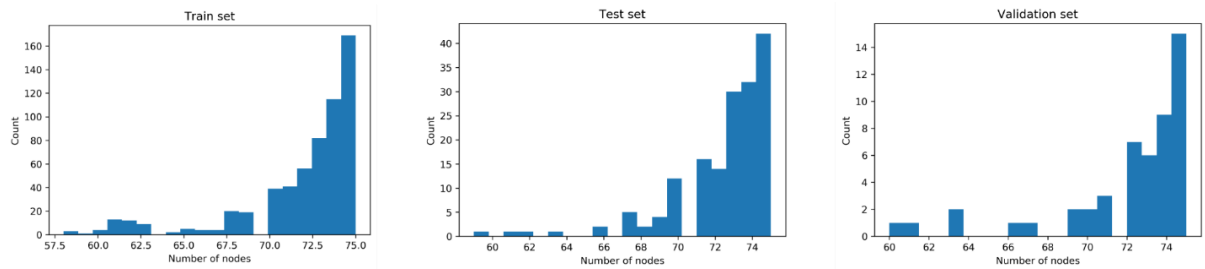

Figure 8 Histogram showing number of nodes and their count in Train set, Test set and Validation Set for LUNGS_75 dataset.

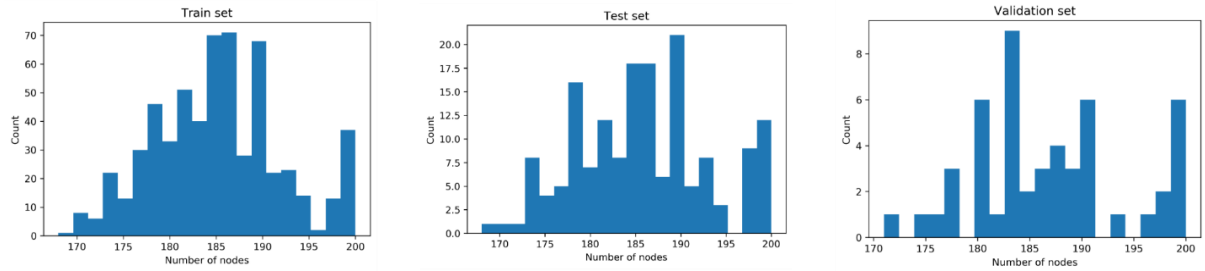

Figure 9 Histogram showing a number of nodes and their count in Train set, Test set and Validation Set for LUNGS_200 dataset.

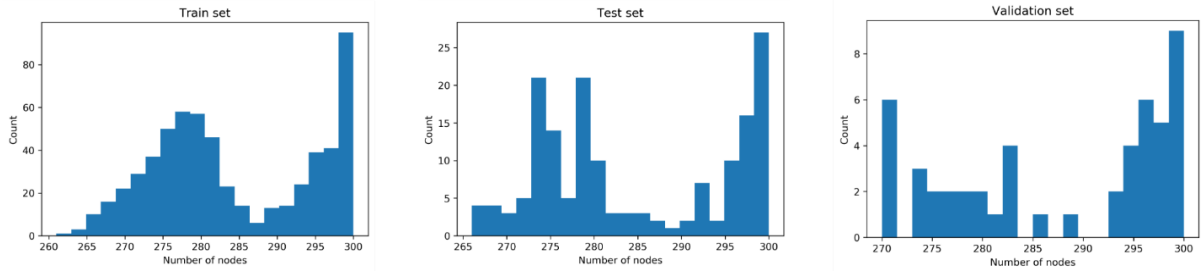

Figure 10 Histogram showing a number of nodes and their count in Train set, Test set and Validation Set for LUNGS_300 dataset.

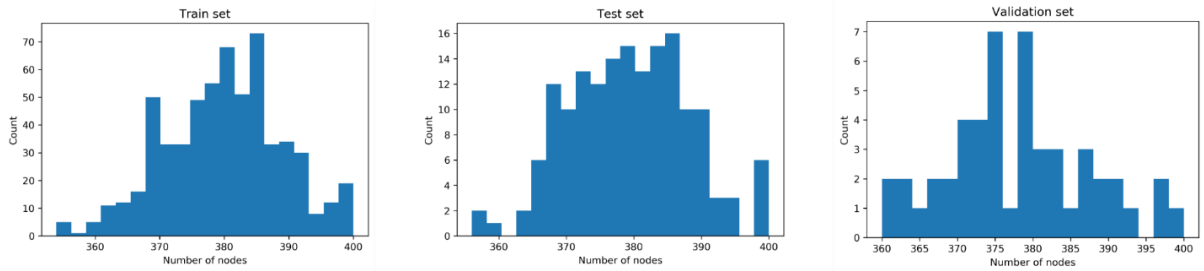

Figure 11 Histogram showing a number of nodes and their count in Train set, Test set and Validation Set for LUNGS_400 dataset.

3.4. Graph Representation

Following the superpixel segmentation process, we constructed a Region Adjacency Graph (RAG) where each superpixel was treated as a node representing segmented regions within the image. The relationships between nodes were established by considering their K-nearest neighbors, encompassing more than one neighbor level to capture a broader context. In this graph, every node has associated features that aggregate information derived from the inherent characteristics of the corresponding superpixel. To generate node features, various methods can be employed. One approach involves incorporating coordinate data, which includes the positional information of each superpixel within the image. Moreover, positional information of the nodes is necessary to create edges between nodes. This spatial information is valuable for understanding the arrangement and distribution of segmented regions. Statistical attributes such as the mean of each superpixels grayscale channels is included as a node feature. These statistical measures provide insights into the intensity distribution and variability within each superpixel. By leveraging these diverse features, the RAG becomes a rich representation of the image, encapsulating both spatial and intensity-related information. This comprehensive graph structure serves as a foundation for subsequent analysis, particularly in the context of GNN which can exploit these features to discern patterns and relationships essential for accurate classification tasks, such as distinguishing between TB-positive and TB-negative CXR images:

$$X = (x_1, x_2, \dots, x_n)^T \text{ and } x_i = \left[\frac{1}{N_i}\sum_{j=1}^{N_i}\left(I_j, a_j, b_j\right)^T\right], \tag{2}$$

where x_i is the node feature with superpixel label i, N_i is the number of pixels whose label is i, $I_j$is the mean pixel intensity of superpixel, a_j and b_j is the pixel's position (co-ordinate) in the image and X is the combination of all feature vectors that would be present in the node.

The calculation of edge features between nodes involves determining the Euclidean distance between the center of masses of the corresponding superpixels. This computation is facilitated by

employing a fixed-width Gaussian function. In the context of edge feature computation, this Gaussian function is likely used to assign weights to edges based on the Euclidean distance between the center of masses. The fixed width ensures a consistent influence of distance on the weight assigned to the edge:

$$G_{ij}^{K} = \exp\left(-\frac{\left|x_i - x_j\right|^2}{\sigma_x^2}\right), \tag{3}$$

where x_i, x_j are the 2-D coordinates of superpixels i, j, and σ_x is the scale parameter defined as the averaged distance x_k of the k nearest neighbors for each node and build adjacency matrix from each sample.

We make an initial assumption that each pair of nodes in the graph can have at most one edge connecting them. However, real-world data can be complex with nodes often having diverse relationships that encompass various semantic, physical, or abstract aspects. There may be additional relationships within the data that are not explicitly represented and could be captured by relaxing the restriction on the number of edges between nodes, allowing for the inclusion of multiple edges, beyond those predefined in the dataset. As depicted in Figure 4.2, once the original image undergoes the SLIC segmentation, we initially extract the relevant features from the resulting superpixel image. Provided that the features are not distinct when nodes rely solely on coordinate information for connectivity, we employ a combination of both coordinates and pixel intensity or just the coordinate point for establishing connections. This fusion serves as an effective initialization strategy, providing clearer representation of the target structure within the graph.

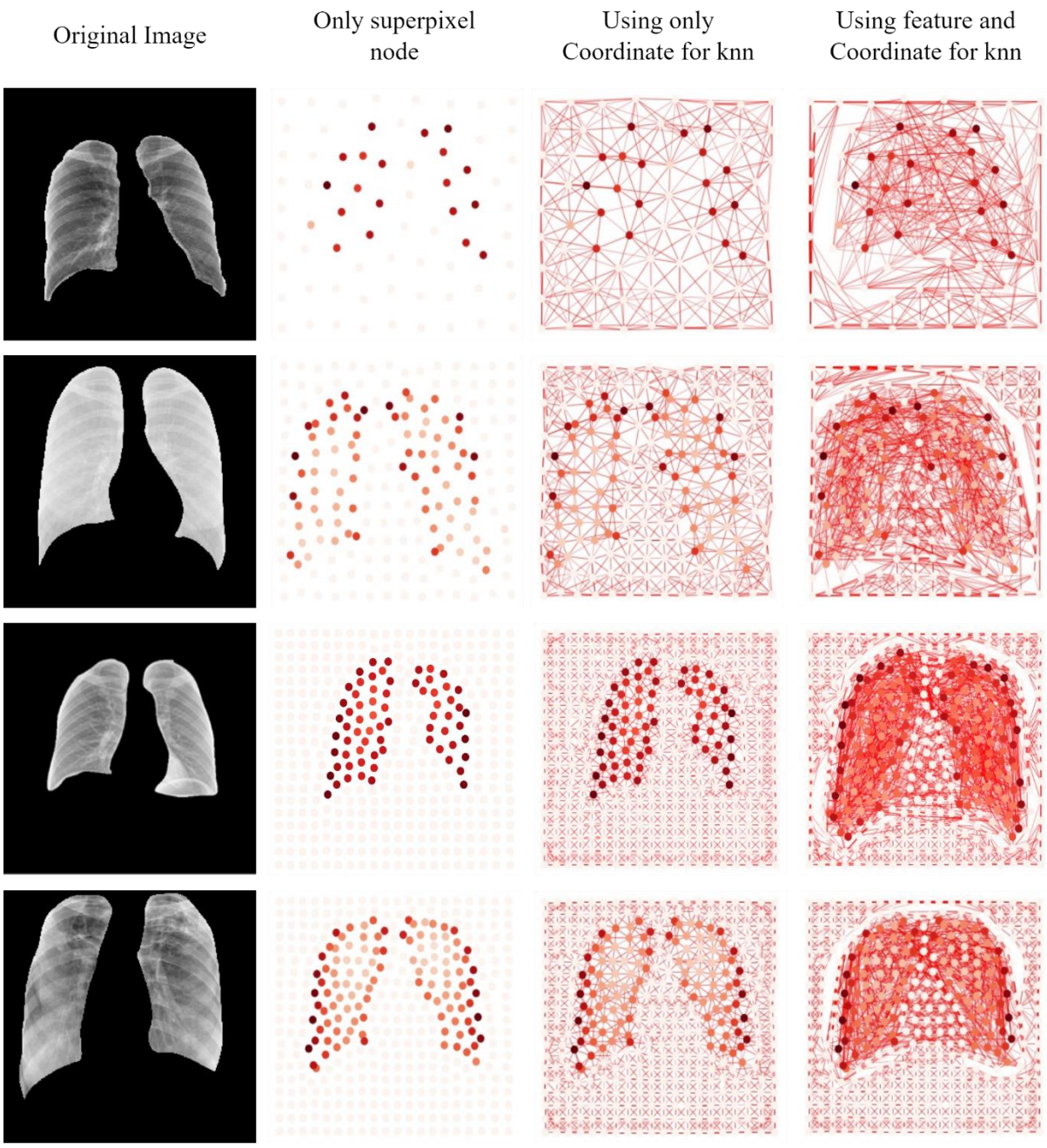

Figure 12 Visualizing data conversion (left to right): lung segmentation, superpixel nodes (centroids) with SLIC, connectivity (only coordinate) of graph representation with 75/200/300/400 nodes, and connectivity (feature and coordinate) of the graph represent with 75/200/300/400 nodes.

3.5. Graph as Dense Data Structures

Our hypothesis rests on the idea that a graph, as a representation, may offer a more robust and informative model for an image than a tensor. In essence, an image serves as a repository of

information, and while tensors serve as a conventional means of representing such information, they may not inherently capture all the intricate relationships within an image. On the other hand, a graph representation introduces the concept of nodes and edges, allowing for a more detailed depiction of an object and its connections to other elements in the image. In the context of an image, nodes can signify individual components like pixels, and edges describe the relationships between these components. This inclusion of relational information is a key differentiator from tensors, which typically lack an explicit representation of relationships. The belief underlying our hypothesis is that this explicit modeling of relationships in a graph provides a more comprehensive understanding of an image, potentially resulting in improved representation and, subsequently, enhanced performance in tasks such as image classification.

3.6. Superpixel-based GNN

There exists a variety of GNN architectures, given the accelerated growth in the area. However, it poses the problem of which models to consider in our experiments. We chose the GNN model based on the criteria of popularity and diversity which are frequently discussed and used in literature. The diversity of models is important to better understand and explain the subsequent results. This is the reason we selected GNNs from two different categories (spatial and spectral).

The models we use are Graph Convolutional Network (GCN) [22] and Graph Attention Network (GAT) [38] which are relatively popular. We want to evaluate how convolution-based GNN and attention-based GNN work on classifying CXR images. They are the different approaches for aggregating information from neighboring nodes in a graph.

Convolution-based GNN methods are spectral methods that perform localized convolutions mimicking CNNs, much like how CNNs use filters of different shapes to obtain the mean value of the central pixel by aggregating information from neighboring pixels. In the spectral domain graph structures are represented using eigenvectors of graph Laplacian matrix, which encodes the information about the neighboring nodes in the graph. By building polynomials on the Laplacian, we obtain the equivalent of filters in GNN that is applied to its own features and the features of its neighbors to compute the weighted sum of the features, where weights are determined by the graph structures (adjacency matrix). In most GCNs, the same weight matrix is shared across all the nodes during the convolution operation. This weight sharing simplifies the model and is powerful in capturing local patterns but hampers the adaptability to assign varying importance to neighbors based on their relevant central nodes.

Attention mechanisms can be considered an additional learned weighing scheme, such that instead of relying on fixed or predetermined weights for aggregating information from neighboring nodes, attention mechanism assigns learnable attention coefficients to each neighbor dynamically. These the attention coefficients are computed based on the compatibility or importance of each neighbor's features while downplaying the importance of less relevant ones. Most importantly, the attention mechanism allows for node-specific weights where each node in the graph has its own

set of attention coefficients, providing a level of customization and adaptability that is absent in plain convolutional mechanism. Attentional architectures are inherently spatial, as they operate on a node neighborhood to assess the importance of neighbors.

Both the GCN and GAT models rely on a propagation module to facilitate information exchange among nodes which is also known as message passing. As the information propagates through the layers, each node refines its representation by considering the features of its neighbors, capturing local and global patterns in the graph. The network model for GNN can be broken down into three steps:

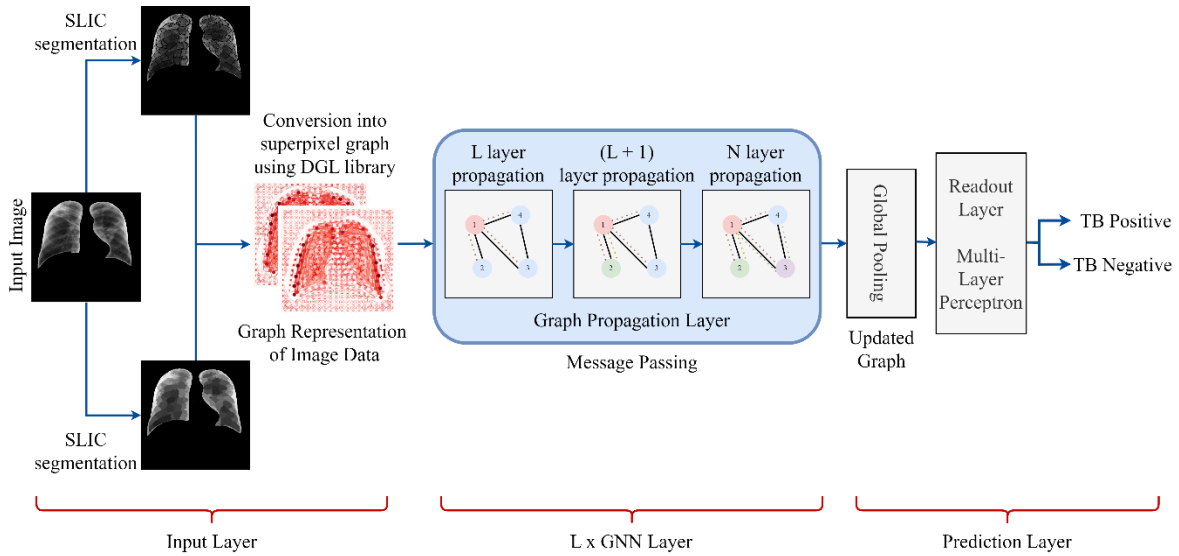

Figure 13 Workflow of graph neural network layer in the classification of TB positive and TB Negative [70].

3.6.1. Input Layer

In message-passing-based graph convolutional networks, the information encapsulated within a node, including pixel intensity, coordinate points, and edge connections, undergoes a process of exchange with neighboring nodes. This phenomenon, known as message passing, orchestrates the iterative update of nodes across different layers. This update process can be formally expressed as the transformation of node representations from one layer to the next as:

$$h_i^{l+1} = f\left[h_i^l, \left(h_j^l\right)_{j \epsilon N_i}\right], \tag{4}$$

where h_i^{l+1} is the updated node feature of layer h_i^l and h_j^l is node feature of the corresponding neighbors of h_i^l.

Each node in the graph is associated with node features, denoted as $\alpha_i \in R^a$, where a is the dimensionality of the node features. These attributes encapsulate diverse characteristics of the node, ranging from pixel intensity and spatial coordinates to other pertinent information. In addition to node features, there may be edges connection the nodes, and each edge can have associated edge features denoted as $\beta_{ij} \in R^b$, where b is the dimensionality of the edge features. These edge features could represent attributes associated with the connection between the nodes. Before both the node features and edge features are fed into the GNN, they undergo a linear transformation to project them into a lower-dimensional space. For node features and edge features, this transformation is represented by a weight matrix $W_h \in R^{dXa}$ and $W_e \in R^{dXb}$, where d is the desired dimensionality of the hidden features. These linear transformations can be represented as:

$$h_i^0 = W_h \alpha_i \; ; \; e_{ij}^0 = W_e \beta_{ij}. \tag{5}$$

3.6.2. GNN Layer

In every GNN layer, we iteratively compute d-dimensional representations for both nodes and edges in the graph. This process, known as neighborhood diffusion or message passing, involves nodes gathering information from their neighboring nodes, allowing them to encapsulate and represent the local structure of the graph. By stacking multiple GNN layers (denoted as L), the network systematically constructs node representations that span information from nodes within L hops away. Essentially, this stacking mechanism enables the network to progressively incorporate details from nodes that are increasingly distant from each target node. This hierarchical approach serves to capture both local and global graph structures, providing a comprehensive and nuanced understanding of the graph's organization.

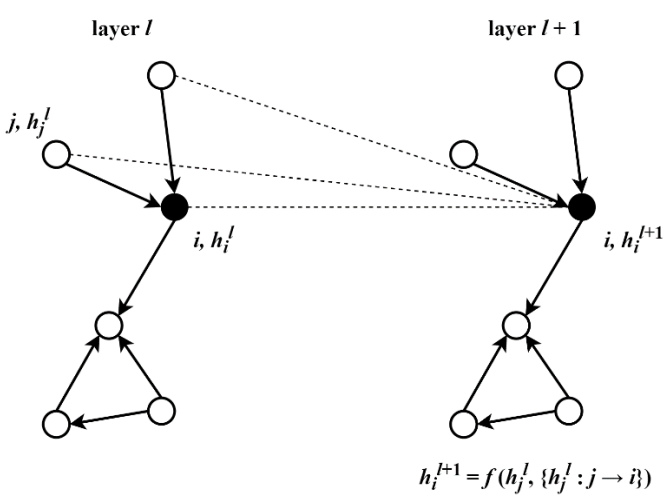

Figure 14 A generic graph neural network layer. Adapted from *Bresson and Laurent* [39].

As depicted in Figure 14, let h_l^i represent the feature vector at layer l associated with node i. The updated features h^i_{l+1} at the next layer $l + 1$ are derived by applying non-linear transformations to the central feature vector h_l^i and all the neighboring connected nodes j of node i. Equation 3 can be regarded as the generalized form of the feature vector h^i_{l+1} at vertex i, where j € N_i denotes the set of neighboring nodes, and f maps the input vector h_l^i along with an unordered set of vectors h_l^i. The specific choice of the mapping function f defines the class of GNNs withing the network architecture.

a. Graph Convolutional Network (GCN): In the simplest form of GNNs, referred to as Graph Convolutional Networks (Graph CovnNets). The initial formulation involves isotropic averaging, where each neighboring node's contribution is given equal weight. The equation is expressed as:

$$h_i^{l+1} = \mathrm{ReLU}\left(\mathrm{W}^l \mathrm{Mean}_{j \epsilon N_i} h_j^l\right), \tag{6}$$

where $\mathrm{W}^l \in \mathrm{R}^{dXd}$ is a matrix that updates node features through linear transform, $\mathrm{Mean}_{j \in Ni}\ h^l_j$ normalizes the averaging operation by the number of neighbors, it is also commonly known as convolution as it approximates a localized spectral convolution. The ReLU activation function is applied elementwise to the result of the linear transformation, introducing non-linearity to the model.

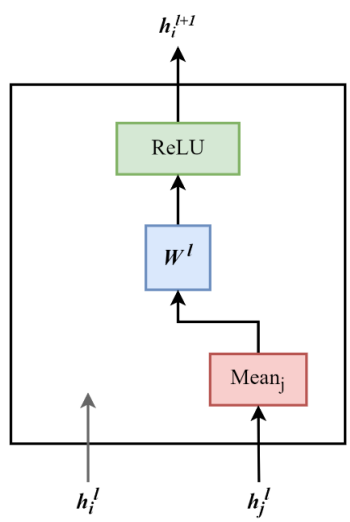

Figure 15 GCN Layer

The GCN model proposed by *Kipf and Welling* [22] uses symmetric normalization instead of isotropic averaging. Symmetric normalization involves dividing by the square root of the product of the in-degree of the central and neighboring nodes, which results in the following node update equation:

$$h_i^{l+1} = \mathrm{ReLU}\left(W^l \frac{1}{\sqrt{deg_i}\sqrt{\mathrm{deg_j}}} \sum_{j \epsilon N_i} h_j^l \right), \tag{7}$$

where $\mathrm{deg_i}$ is the in-degree of node *i*, and $\mathrm{deg_j}$ is the in-degree of the neighboring node *j*.

The node update equation involves linear transformation, isotropic averaging (or symmetric normalization), ReLU activation, and the potential inclusion of self-loops or residual connections. This formulation enables the model to capture information from the neighborhood of each node, making it suitable for graph-structured data. The symmetric normalization variant addresses some limitations of isotropic averaging, providing more stability in the learning process.

b. Graph Attention Network (GAT): In the GAT each pair of neighboring nodes engages in a reciprocal attention mechanism, determining the significance of one another during the information aggregation process. This intricate attention mechanism empowers nodes to selectively concentrate on neighbors deemed more pertinent and informative for the specific task at hand. Unlike conventional GCN with fixed weights serving as normalizing constants, GAT introduces a dynamic paradigm. In GAT, attention weights are learned iteratively during training rather than being predetermined, endowing the network with adaptability and the ability to capture nuanced relationships within the graph. Notably, each node attends to itself, simulating a self-loop. To strengthen expressive power, node features undergo a linear transformation into a higher-level space before attention score computation. Post-transformation, attention scores are normalized using the SoftMax function, creating a probability distribution for each node. This dynamic and learned attention mechanism fosters flexibility and expressiveness, allowing nodes to autonomously prioritize neighbors. This intrinsic adaptability contributes to the overall spatial nature characteristic of attentional GNNs.

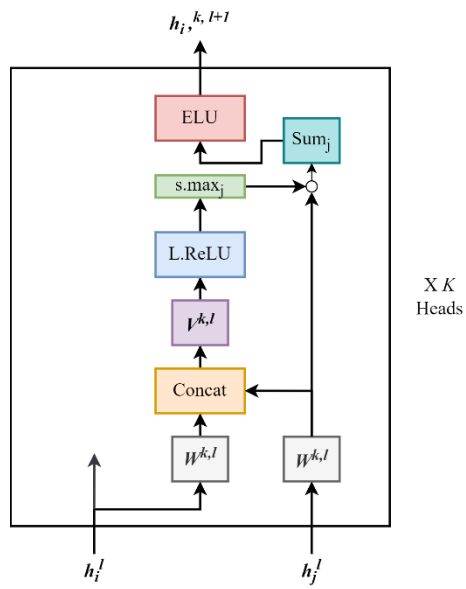

Figure 16 GAT layer

GAT uses Bahdanau attention [49] also known as the additive score function to introduce anisotropy in the neighborhood aggregation function. Anisotropy refers to the introduction of directionality or orientation in the neighboring aggregation, which makes the model sensitive to the directional relationship between the nodes. Nodes can selectively attend to neighbors with varying degrees of emphasis, capturing directional information in the graph. The node update equation is given by:

$$h_i^{l+1} = \text{Concat}_{k=1}^{K}\left(\text{ELU}\left(\sum_{j\epsilon N_i} e_{ij}^{k,l} W^{k,l}, h_j^l\right)\right), \tag{8}$$

where $W^{k,l} \in R^{d/K \text{ x } d}$ are the K linear projection heads, the use of multi-head attention is aimed at stabilizing the learning process and enhancing the model's ability to capture diverse and complex relationships in the graph. The idea of multi-head attention involves employing multiple attention mechanisms, in parallel. These heads operate independently and can either be concatenated or averaged to produce the final output. The attention coefficients for each head are defined as:

$$e_{i,j}^{k,l} = \frac{\exp\left(\hat{e}_{i,j}^{k,l}\right)}{\sum_{j\epsilon N_i} \exp\left(\hat{e}_{i,j'}^{k,l}\right)} \text{ and} \tag{9}$$

$$\hat{e}_{i,j}^{k,l} = \text{LeakyReLU}(V^{k,l})\text{Concat}\left(U^{k,l}h_i^l, U^{k,l}h_j^l\right), \tag{10}$$

where LeakyReLU is a smoothing approximation of ReLU and $V^{k,l} \in R^{2d/K}$, learns a mean over each node's neighborhood features sparsely weighted by the importance of each neighbor.

3.6.3. Prediction Layer

While local information exchange among neighboring nodes is important, achieving effective graph classification requires the integration of global information. This integration is facilitated through a Readout layer, which aggregates information from all node embeddings and consolidates them into a singular graph embedding vector. This operation, often referred to as pooling, ensures a comprehensive representation of the entire graph. By flattening the gathered node embeddings into a unified vector, the Readout layer encapsulates global structural characteristics, allowing for informed and accurate graph classification. In essence, this pooling operation harmonizes both local and global perspectives, enabling the model to make informed decisions based on the holistic features of the graph. Most often, simple and permutation invariant functions are used: mean, max and sum. For the case of both GCN and GAT we use the mean readout layer to aggregate the node embeddings, resulting in a single graph-level embedding.

The prediction layer serves the crucial role of producing task-specific outputs, which are subsequently input into a loss function for the end-to-end training of network parameters. Specifically, the input to the prediction layer is derived from the outputs of the final message passing the GCN layer for each node in the graph. For graph classification tasks, the process involves crafting a d-dimensional graph-level vector representation, denoted as *yG*. This representation is constructed by computing the average of all node features extracted from the last GCN layer. In essence, *yG* encapsulates the collective information from individual nodes, forming a comprehensive graph-level feature vector for downstream classification which is given by:

$$yG = \frac{1}{\mathrm{V}} \sum_{i=0}^{V} h_i^L , \tag{11}$$

where V is the total number of nodes in the graph and is used to compute the average of node features and $\sum_{i=0}^{V} h_i^L$ represents the summation of the node features across all the nodes in the graph.

The graph features are then passed to an MLP, which outputs un-normalized logits/scores $y_{pred} \in R^C$ for each class:

$$y_{\text{pred}} = P\text{ReLU}(QyG), \tag{12}$$

where $P \in R^{dXC}$, $Q \in R^{dXd}$, C is the number of classes. Finally, we minimize the cross-entropy loss between the logits and ground truth labels.

Chapter 4

4. Experiments

Summary: This chapter explores the experimental setup for evaluating GNNs in handling increasingly dense data as the number of nodes varies. The study employs standard dataset splits, Adam optimizer with dynamic learning rate decay, and diverse evaluation metrics. Two models, GCN and GAT, are evaluated on datasets with varying node sizes, emphasizing flexibility and attention mechanisms. Results reveal comparable performance between GCN and GAT, with nuanced findings on the impact of node quantity. The relationship between the number of nodes and model efficacy is identified as context-dependent, providing valuable insights for future research.

Key Topics: GCN Net Architecture, GAT Net Architecture, Evaluation metrics.

4.1. Experimental Setup

We hypothesized that by leveraging a graph structure, GNNs can extract more information from the denser data and learn more from the graph structure when increasing the node size. We use the standard splits of all four datasets LUNGS_75, LUNGS_200, LUNGS_300 and LUNGS_400. All the datasets have the same number of splits i.e., 513 train, 50 validation and 141 test graphs. The 50 graphs for the validation set are randomly sampled from the training set and the same split is used for both GNNs. To measure the performance, the following evaluation metrics are used:

- Accuracy (ACC) = $\frac{(t_p+t_n)}{(t_p+t_n+f_p+f_n)}$,
- Sensitivity (SEN) = $\frac{t_p}{(t_p+f_n)}$,
- Specificity (SPEC) = $\frac{t_n}{(t_n+f_p)}$ and
- Area under the ROC curve (AUC),

where t_p, f_p, t_n *and* f_n are the total number of true positive, false positive true negative and false negative, respectively.

We employ the Adam optimizer [71]with a consistent learning rate decay strategy across all models. Initially, we select an initial learning rate from the set (10^{-2}, 10^{-3}, 10^{-4}). If the validation loss fails to improve after a fixed number of epochs, the learning rate is halved. Notably, we do not set a predefined maximum number of epochs and instead, training continues until the learning

rate reaches the minimal value of 10^{-6}. This adaptive approach ensures that the training process is dynamically responsive, allowing for efficient convergence and termination based on the observed behavior of the optimization process.

Our experimental objective does not revolve around maximizing the absolute performance of the models; rather, we aim to discern trends in performance as the number of nodes increases and to understand the influence of node embeddings (coordinate points and pixel intensity) on models' performance. Each model is equipped with approximately 100,000 parameters, with slight variations depending on the hidden unit's configuration. To ensure robustness and account for variability, each experiment is conducted three times, each initialization employing a different seed. This meticulous approach allows us to derive insights into the nuanced relationship between model parameters, node quantity, and the impact of different node embeddings on overall model performance.

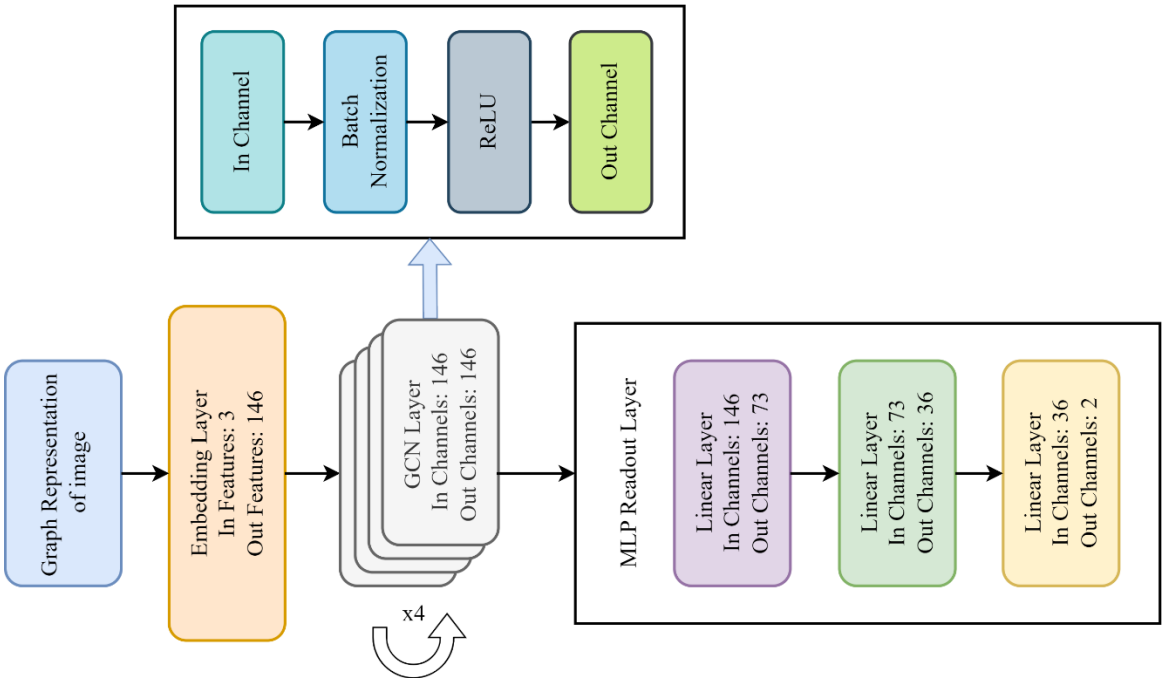

Figure 17 GCN Net Architecture

The GCN Net model is structured for efficient graph-based classification. It initializes with parameters defining input/output dimensions, hidden layers, batch normalization and activation function. Linear embedding adapts input features and undergoes a linear transformation through a weighted sum of their elements, resulting in output features. A stack of GCN layers processes the graph data, incorporating ReLU activation, batch normalization, and residual connections for stability. The model performs a mean readout operation to aggregate node features into a graph-level representation. An MLP readout layer then reduces the dimensionality for accurate

predictions. During training, the model calculates binary cross-entropy loss. This modular design emphasizes flexibility in graph-based tasks, offering customizable options for layer count, activation functions, and readout operations.

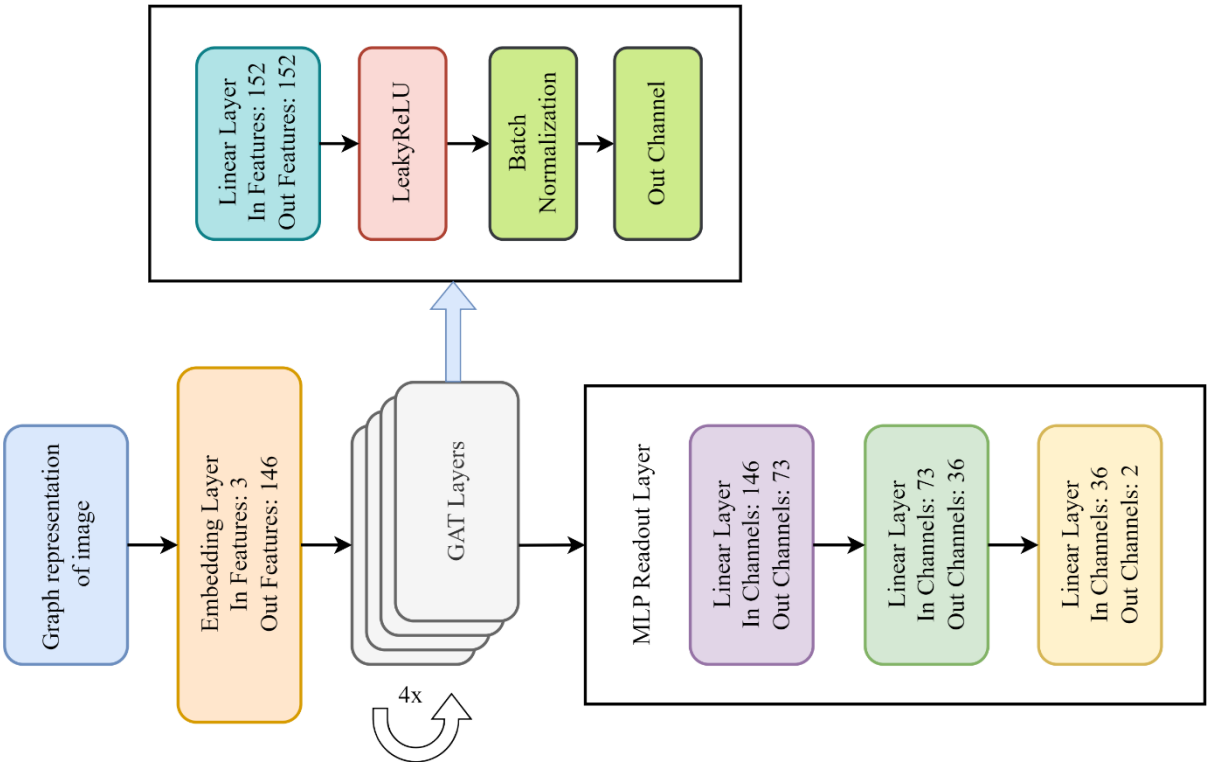

Figure 18 GAT Net Architecture

The GAT Net model follows a streamlined workflow, starting with parameter initialization for input/output dimensions, attention heads, and architecture details. Linear embedding adapts input features for multiple attention heads, allowing the model to learn diverse representations simultaneously. The stack of GAT layers performs attention operations, incorporating batch normalization and residual connections for stability. Leaky ReLU (Rectified Linear Unit) is an activation function commonly used to add nonlinearity to the model. It is like the traditional ReLU activation but allows a small, non-zero gradient when the input is negative. This adaptable design aligns with specified dimensions and attention head counts. The model aggregates node features using a mean readout operation, followed by an MLP readout layer for accurate predictions. The sequential processing of graph structure, node features, and edge features ensures flexibility in graph-based classification tasks, emphasizing attention mechanisms and modular architecture.

4.2. Results and Analysis

Our experimental evaluations involved two distinct models, GCN and GAT, applied to four different datasets: LUNGS_75, LUNGS_200, LUNGS_300, and LUNGS_400. To comprehensively assess the impact of node embeddings, we created two versions of each dataset—one with node embeddings containing only coordinate points and another with both coordinate points and pixel intensity. This design choice enabled a comparative analysis of how node embeddings influence the models' ability to capture relevant information.

In terms of overall performance, both GCN and GAT exhibited comparable results, displaying varying accuracy and AUC scores across the four datasets. When evaluating the models' ability to accurately classify graph structures, both GAT and GCN achieved the highest AUC scores of 0.8 and 0.79, respectively. This parity in AUC scores indicates that both models perform similarly in distinguishing between TB-positive and TB-negative cases. Notably, this achievement was observed on the LUNGS_300 dataset when GAT and GCN were incorporated both coordinate and pixel intensity as node embeddings.

Although we observe a gradual increase in the AUC scores and accuracy when the number of nodes is increased LUNGS_400 doesn’t perform quite well compared to the LUNGS_300 dataset. This could be because as the number of nodes increases the graph becomes more and more like grid structured data and because GCN works well on structured data we see GCN having a better accuracy and AUC score than the GAT model. We also observe that in most results when nodes with coordinates and nodes with coordinates plus pixel intensity are compared, they have some variations in them meaning that graphs are failing to capture enough relevant information.

Upon evaluating the GNN models across the four datasets, it becomes evident that the number of nodes in the graph influences overall performance. However, the impact is characterized by a mixed trend, with AUC scores fluctuating in response to variations in the number of nodes. This implies that while the number of nodes does exert an influence on model performance, it does not follow a straightforward pattern of improvement with an increase in nodes. The fluctuating trend suggests that the relationship between the number of nodes and overall model performance is intricate and cannot be assumed to follow a linear trajectory. Hence, the influence of node quantity on model efficacy appears to be context-dependent and requires careful consideration in the interpretation of GNN model outcomes. The number of edges in the graphs increases according to the increase in the number of nodes which would allow information to propagate faster and wider throughout the graph. However, given the comparable large number of nodes and their edges, noise in the graph would also propagate more. This could explain why the models perform in an alternating pattern while varying the number of nodes in the graph.

Table 3 Performance evaluation result of the four datasets LUNGS_75, LUNGS_200, LUNGS_300, and LUNGS_400 with respective models GCN and GAT with two different node embeddings.

GNN Models	Accuracy	AUC	Sensitivity	Specificity
LUNGS_75				
GCN (Coordinates only)	0.6073	0.6261	0.7562	0.4959
GAT (Coordinates only)	0.73	0.716	0.8063	0.5257
GCN (Coordinates and pixel intensity)	0.6871	0.6907	0.8688	0.5127
GAT (Coordinates and pixel intensity)	0.7055	0.742	0.8402	0.6438
LUNGS_200				
GCN (Coordinates only)	0.6993	0.7047	0.701	0.7083
GAT (Coordinates only)	0.7423	0.7063	0.7083	0.7042
GCN (Coordinates and pixel intensity)	0.6993	0.7088	0.7342	0.6833
GAT (Coordinates and pixel intensity)	0.773	0.7399	0.8007	0.6792
LUNGS_300				
GCN (Coordinates only)	0.8036	0.7673	0.8413	0.6934
GAT (Coordinates only)	0.7975	0.7497	0.6423	0.8539
GCN (Coordinates and pixel intensity)	0.7989	0.7911	0.851	0.736
GAT (Coordinates and pixel intensity)	0.8098	0.7935	0.881	0.706
LUNGS_400				
GCN (Coordinates only)	0.7055	0.7104	0.6414	0.7794
GAT (Coordinates only)	0.7484	0.7503	0.6924	0.8082
GCN (Coordinates and pixel intensity)	0.7668	0.7642	0.7368	0.7915
GAT (Coordinates and pixel intensity)	0.7361	0.747	0.7692	0.7505

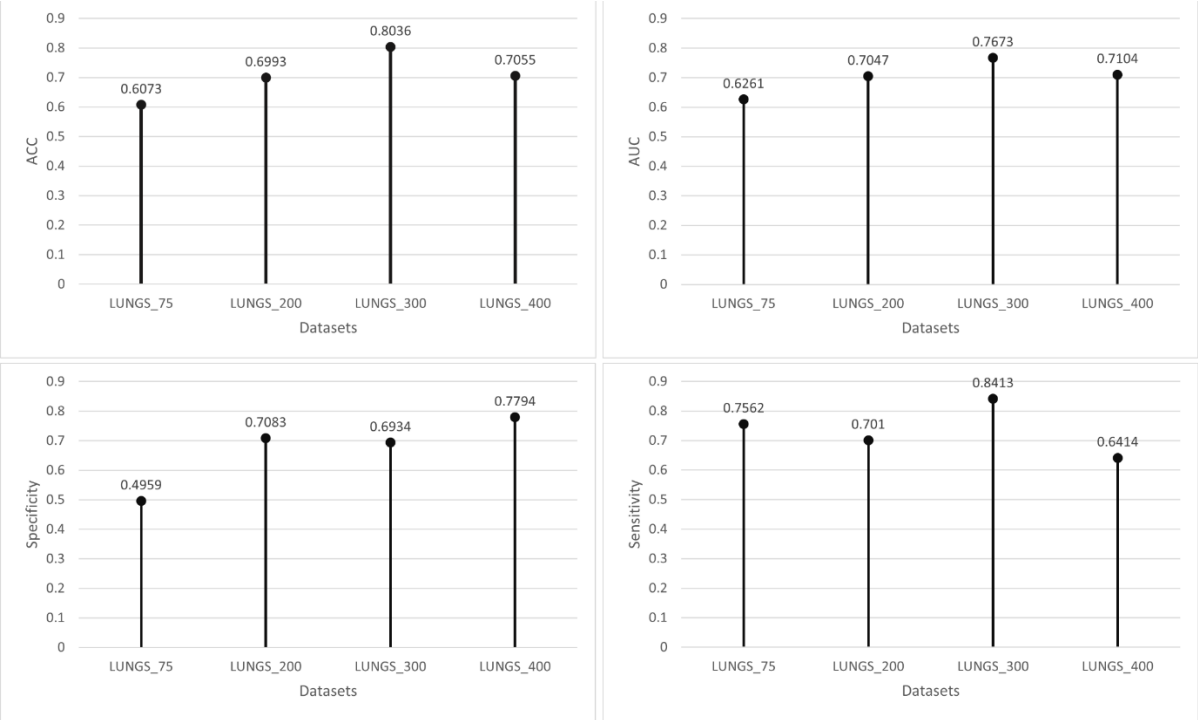

Figure 19 Graphical representation of GCN (Coordinate only): ACC, AUC, SPE, SEN

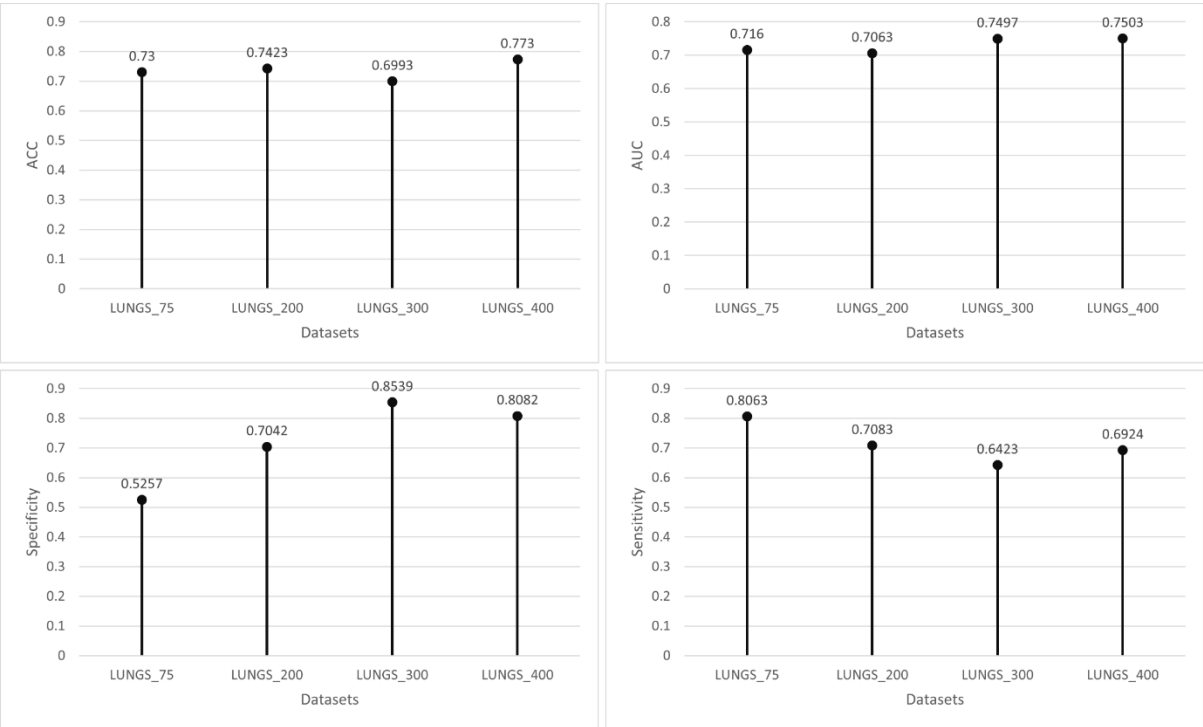

Figure 20 Graphical representation of GAT (Coordinate only): ACC, AUC, SPE, SEN

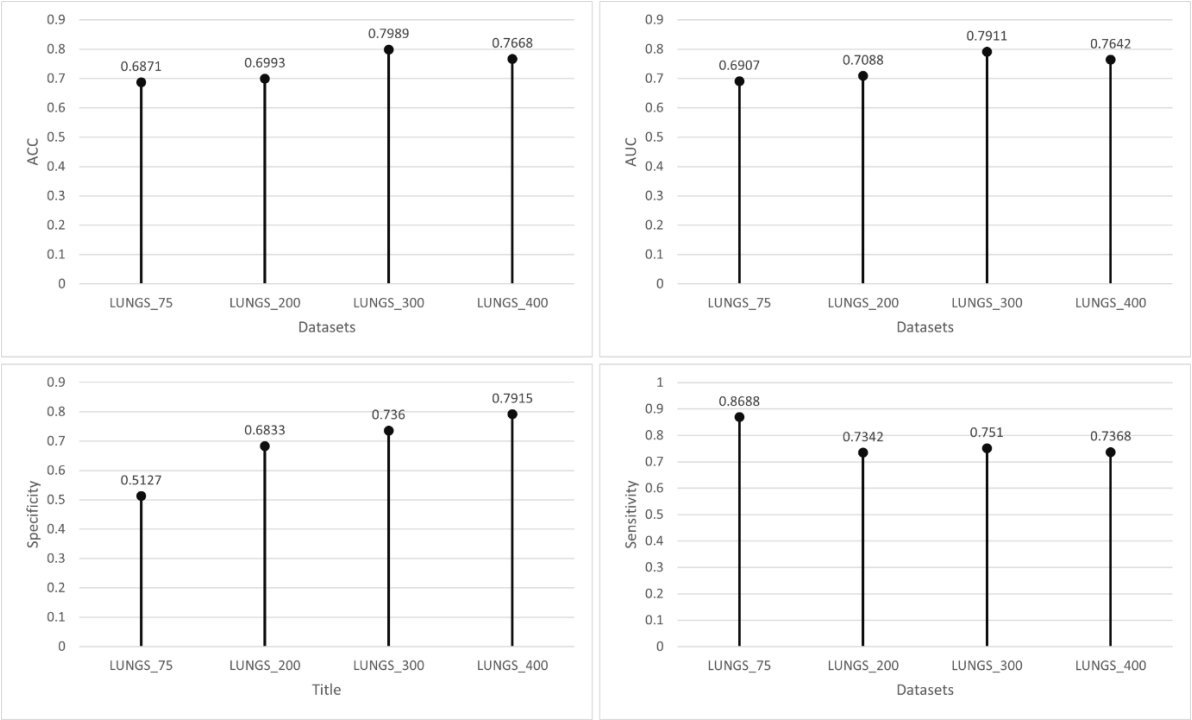

Figure 21 Graphical representation of GCN (Coordinate and pixel intensity): ACC, AUC, SPE, SEN

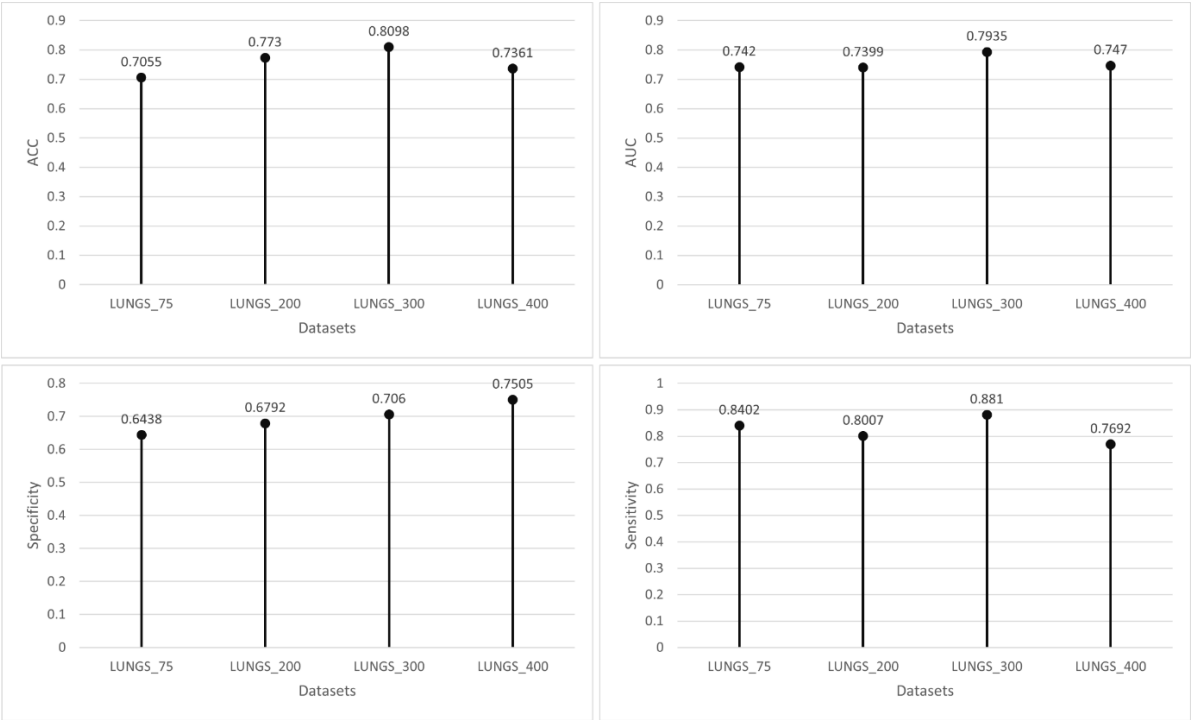

Figure 22 Graphical representation of GAT (Coordinate and pixel intensity): ACC, AUC, SPE, SEN

4.3. Comparision

In our study, we carried out experiments with popular GNN models: GCN and GAT for performance comparison. For a fair comparison, we have split the results according to the datasets so that we can compare which model outperforms the other model. Table 3 shows the comparative study among GNN models for LUNGS_75, here for both cases GAT outperforms GCN in terms of AUC and ACC, this trend is extended and is observed with similar results in Table 4 and table 5. However, for Table 6 this trend holds true for cases dealing with coordinate only, but GCN has a better result when the case was coordinate and pixel intensity.

Table 4 Comparison: AUC, ACC, SPEC, and SEN of GCN (Coordinate only), GCN (Coordinate + Pixel Intensity), GAT (Coordinate only), GAT (Coordinate + Pixel Intensity) in LUNGS_75 dataset.

Models	AUC	ACC	SPEC	SEN
GCN (Coordinate only)	0.6261	0.6073	0.4959	0.7562
GAT (Coordinate only)	0.716	0.73	0.5257	0.8063
GCN (Coordinate + Pixel intensity)	0.6907	0.6871	0.5127	0.8688
GAT (Coordinate + Pixel intensity)	0.742	0.7055	0.6438	0.8402

Table 5 Comparison: AUC, ACC, SPEC, and SEN of GCN (Coordinate only), GCN (Coordinate + Pixel Intensity), GAT (Coordinate only), GAT (Coordinate + Pixel Intensity) in LUNGS_200 dataset.

Models	AUC	ACC	SPEC	SEN
GCN (Coordinate only)	0.7047	0.6993	0.7083	0.701
GAT (Coordinate only)	0.7063	0.7423	0.7042	0.7083
GCN (Coordinate + Pixel intensity)	0.7088	0.6993	0.6833	0.7342
GAT (Coordinate + Pixel intensity)	0.7399	0.773	0.6792	0.8007

Table 6 Comparison: AUC, ACC, SPEC, and SEN of GCN (Coordinate only), GCN (Coordinate + Pixel Intensity), GAT (Coordinate only), GAT (Coordinate + Pixel Intensity) in LUNGS_300 dataset.

Models	**AUC**	**ACC**	**SPEC**	**SEN**
GCN (Coordinate only)	0.7673	0.8036	0.6934	0.8413
GAT (Coordinate only)	0.7497	0.7975	0.8539	0.6423
GCN (Coordinate + Pixel intensity)	0.7911	0.7989	0.736	0.851
GAT (Coordinate + Pixel intensity)	0.7935	0.8098	0.706	0.881

Table 7 Comparison: AUC, ACC, SPEC, and SEN of GCN (Coordinate only), GCN (Coordinate + Pixel Intensity), GAT (Coordinate only), GAT (Coordinate + Pixel Intensity) in LUNGS_400 dataset.

Models	**AUC**	**ACC**	**SPEC**	**SEN**
GCN (Coordinate only)	0.7104	0.7055	0.7794	0.6414
GAT (Coordinate only)	0.7503	0.7484	0.8082	0.6924
GCN (Coordinate + Pixel intensity)	0.7642	0.7668	0.7915	0.7368
GAT (Coordinate + Pixel intensity)	0.747	0.7361	0.7505	0.7692

Chapter 5

5. Conclusion and Future Work

We generated distinct datasets, namely LUNGS_75, LUNGS_200, LUNGS_300, and LUNGS_400, by constructing graphs from CXR images. The datasets varied in the number of superpixels (nodes) and featured unique node embeddings with either coordinates only or both coordinates and pixel intensity. Our evaluation is centered on two popular and diverse GNN architectures. The results highlighted the substantial impact of graph topology and node features on the performance of these architectures, underscoring the importance of factors such as the number of nodes, local neighborhood edge density, and positional information. However, it's noteworthy that increasing the graph structure by segmenting the image into a greater number of superpixels did not lead to improved performance. A potential factor contributing to lower results is the absence of a dataset with masked lung boundaries. Collecting large datasets for deep learning is challenging, and simply expanding the dataset size does not guarantee enhanced system robustness.

Additionally, our observations indicated that the positional variables (coordinates) have a more pronounced impact on the overall performance of GNNs compared to including pixel intensity of the superpixels. This discrepancy may arise from the inherent grid structure of image data, necessitating a comprehensive set of neighbors to effectively represent the entire CXR image. Furthermore, both GCN and GAT exhibited challenges in fully leveraging the provided graph structure, especially when compared to state-of-the-art CNN models. This observation aligns with concerns discussed in [72] and underscores the practical limitations of certain GNNs, emphasizing that they might not be as potent or effective for image data as established CNN models.

Future work for classifying medical images using GNNs can involve exploring and addressing various challenges. Here are some potential avenues for future research and development:

- Incorporate datasets with more classes to generalize GNN over multiple labels. This could help us benchmark the results obtained from this result in a better way.

- Explore other GNN models such as Gated Graph Convolutional Networks (GGCN), Relational Graph Convolutional Networks (RGCN), GraphSage and Graph Isomorphic Network (GIN).

- Investigate hybrid architectures where CNNs could help extract local features and GNNs for modeling global relationships.

- Embed node features with extra dimensions of relevant data and incorporating the concept of directed edges between nodes.

- Train a U-Net model to obtain lung mask of CXR image to increase dataset.

- Research on active learning frameworks for GNNs that can address label-sparse issues from medical datasets [73].

BIBLIOGRAPHY

[1] A. E. Hirsh, A. G. Tsolaki, K. DeRiemer, M. W. Feldman, and P. M. Small, "Stable association between strains of Mycobacterium tuberculosis and their human host populations," *Proc Natl Acad Sci U S A*, vol. 101, no. 14, 2004, doi: 10.1073/pnas.0305627101.

[2] S. Ravimohan, H. Kornfeld, D. Weissman, and G. P. Bisson, "Tuberculosis and lung damage: From epidemiology to pathophysiology," *European Respiratory Review*, vol. 27, no. 147. 2018. doi: 10.1183/16000617.0077-2017.

[3] M. K. Mahbub, M. Biswas, L. Gaur, F. Alenezi, and KC Santosh, "Deep features to detect pulmonary abnormalities in chest X-rays due to infectious diseaseX: Covid-19, pneumonia, and tuberculosis," *Inf Sci (N Y)*, vol. 592, 2022, doi: 10.1016/j.ins.2022.01.062.

[4] A. Makkar and KC Santosh, "SecureFed: federated learning empowered medical imaging technique to analyze lung abnormalities in chest X-rays," *International Journal of Machine Learning and Cybernetics*, vol. 14, no. 8, 2023, doi: 10.1007/s13042-023-01789-7.

[5] KC Santosh, S. Allu, S. Rajaraman, and S. Antani, "Advances in Deep Learning for Tuberculosis Screening using Chest X-rays: The Last 5 Years Review," *J Med Syst*, vol. 46, no. 11, 2022, doi: 10.1007/s10916-022-01870-8.

[6] Vajda S, Karargyris A, Jaeger S, Santosh KC, Candemir S, Xue Z, Antani S, Thoma G, "Feature Selection for Automatic Tuberculosis Screening in Frontal Chest Radiographs," *J Med Syst*, vol. 42, no. 8, 2018, doi: 10.1007/s10916-018-0991-9.

[7] KC Santosh and S. Antani, "Automated chest x-ray screening: Can lung region symmetry help detect pulmonary abnormalities?," *IEEE Trans Med Imaging*, vol. 37, no. 5, 2018, doi: 10.1109/TMI.2017.2775636.

[8] D. Das, KC Santosh, and U. Pal, "Cross-population train/test deep learning model: Abnormality screening in chest x-rays," in *Proceedings - IEEE Symposium on Computer-Based Medical Systems*, 2020. doi: 10.1109/CBMS49503.2020.00103.

[9] KC Santosh, D. GhoshRoy, and S. Nakarmi, "A Systematic Review on Deep Structured Learning for COVID-19 Screening Using Chest CT from 2020 to 2022," *Healthcare (Switzerland)*, vol. 11, no. 17. 2023. doi: 10.3390/healthcare11172388.

[10] KC Santosh, S. Vajda, S. Antani, and G. R. Thoma, "Edge map analysis in chest X-rays for automatic pulmonary abnormality screening," *Int J Comput Assist Radiol Surg*, vol. 11, no. 9, 2016, doi: 10.1007/s11548-016-1359-6.

[11] KC Santosh, "AI-Driven Tools for Coronavirus Outbreak: Need of Active Learning and Cross-Population Train/Test Models on Multitudinal/Multimodal Data," *J Med Syst*, vol. 44, no. 5, 2020, doi: 10.1007/s10916-020-01562-1.

[12] "Belarus Tuberculosis Portal," B. P. Health.

[13] S. Jaeger, S. Candemir, S. Antani, Y.-X. J. Wáng, P.-X. Lu, and G. Thoma, "Two public chest X-ray datasets for computer-aided screening of pulmonary diseases.," *Quant Imaging Med Surg*, vol. 4, no. 6, 2014, doi: 10.3978/j.issn.2223-4292.2014.11.20.

[14] Y. LeCun *et al.*, "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Comput*, vol. 1, no. 4, 1989, doi: 10.1162/neco.1989.1.4.541.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun ACM*, vol. 60, no. 6, 2017, doi: 10.1145/3065386.

[16] M. M. Bronstein, J. Bruna, Y. Lecun, A. Szlam, and P. Vandergheynst, "Geometric Deep Learning: Going beyond Euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4. 2017. doi: 10.1109/MSP.2017.2693418.

[17] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *EEE International Joint Conference on Neural Networks*, 2005.

[18] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans Neural Netw*, vol. 20, no. 1, 2009, doi: 10.1109/TNN.2008.2005605.

[19] F. Scarselli, A. C. Tsoi, M. Gori, and M. Hagenbuchner, "Graphical-based learning environments for pattern recognition," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3138, 2004, doi: 10.1007/978-3-540-27868-9_4.

[20] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," *IEEE Trans Neural Netw Learn Syst*, vol. 32, no. 1, 2021, doi: 10.1109/TNNLS.2020.2978386.

[21] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and deep locally connected networks on graphs," in *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, 2014.

[22] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017.

[23] V. Di Massa, G. Monfardini, L. Sarti, F. Scarselli, M. Maggini, and M. Gori, "A comparison between recursive neural networks and graph neural networks," in *IEEE International Conference on Neural Networks - Conference Proceedings*, 2006. doi: 10.1109/ijcnn.2006.246763.

[24] R. Fergus, A. Perona, and A. Zisserman, "A sparse object category model for efficient learning and exhaustive recognition," in *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, 2005. doi: 10.1109/CVPR.2005.47.

[25] A. Quek, Z. Wang, J. Zhang, and D. Feng, "Structural image classification with graph neural networks," in *Proceedings - 2011 International Conference on Digital Image Computing: Techniques and Applications, DICTA 2011*, 2011. doi: 10.1109/DICTA.2011.77.

[26] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans Pattern Anal Mach Intell*, vol. 22, no. 8, 2000, doi: 10.1109/34.868688.

[27] P. H. C. Avelar, A. R. Tavares, T. L. T. Da Silveira, C. R. Jung, and L. C. Lamb, "Superpixel Image Classification with Graph Attention Networks," in *Proceedings - 2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images, SIBGRAPI 2020*, 2020. doi: 10.1109/SIBGRAPI51738.2020.00035.

[28] D. Stutz, A. Hermans, and B. Leibe, "Superpixels: An evaluation of the state-of-the-art," *Computer Vision and Image Understanding*, vol. 166, 2018, doi: 10.1016/j.cviu.2017.03.007.

[29] J. Long, Z. Yan, and H. Chen, "A Graph Neural Network for superpixel image classification," in *Journal of Physics: Conference Series*, 2021. doi: 10.1088/1742-6596/1871/1/012071.

[30] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017. doi: 10.1109/CVPR.2017.576.

[31] Y. LeCun, C. Cortes, and C. J. C. Burges, "The MNIST database of handwritten digits, 1998," *URL http://yann. lecun. com/exdb/mnist*, vol. 10, no. 34, 1998.

[32] M. Fey, J. E. Lenssen, F. Weichert, and H. Muller, "SplineCNN: Fast Geometric Deep Learning with Continuous B-Spline Kernels," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018. doi: 10.1109/CVPR.2018.00097.

[33] U. Chaudhuri, B. Banerjee, and A. Bhattacharya, “Siamese graph convolutional network for content based remote sensing image retrieval,” *Computer Vision and Image Understanding*, vol. 184, 2019, doi: 10.1016/j.cviu.2019.04.004.

[34] B. Knyazev, G. W. Taylor, and M. R. Amer, “Understanding attention and generalization in graph neural networks,” in *Advances in Neural Information Processing Systems*, 2019.

[35] K. Xu, S. Jegelka, W. Hu, and J. Leskovec, “How powerful are graph neural networks?,” in *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[36] B. Knyazev, X. Lin, M. R. Amer, and G. W. Taylor, “Image classification with hierarchical multigraph networks,” in *30th British Machine Vision Conference 2019, BMVC 2019*, 2020.

[37] V. P. Dwivedi, C. K. Joshi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson, “Benchmarking Graph Neural Networks,” Mar. 2020.

[38] P. Veličković, A. Casanova, P. Liò, G. Cucurull, A. Romero, and Y. Bengio, “Graph attention networks,” in *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018. doi: 10.1007/978-3-031-01587-8_7.

[39] X. Bresson and T. Laurent, “Residual Gated Graph ConvNets,” Nov. 2017.

[40] W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in *Advances in Neural Information Processing Systems*, 2017.

[41] A. Krizhevsky, “Learning Multiple Layers of Features from Tiny Images,” *... Science Department, University of Toronto, Tech. ...*, 2009, doi: 10.1.1.222.9220.

[42] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, “The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains,” *IEEE Signal Process Mag*, vol. 30, no. 3, 2013, doi: 10.1109/MSP.2012.2235192.

[43] Z. Liu and J. Zhou, “Introduction to Graph Neural Networks,” *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 14, no. 2. 2020. doi: 10.2200/S00980ED1V01Y202001AIM045.

[44] D. K. Hammond, P. Vandergheynst, and R. Gribonval, “Wavelets on graphs via spectral graph theory,” *Appl Comput Harmon Anal*, vol. 30, no. 2, 2011, doi: 10.1016/j.acha.2010.04.005.

[45] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” in *Advances in Neural Information Processing Systems*, 2016.

[46] A. Micheli, “Neural network for graphs: A contextual constructive approach,” *IEEE Trans Neural Netw*, vol. 20, no. 3, 2009, doi: 10.1109/TNN.2008.2010350.

[47] J. Atwood and D. Towsley, “Diffusion-convolutional neural networks,” in *Advances in Neural Information Processing Systems*, 2016.

[48] Y. Li, R. Yu, C. Shahabi, and Y. Liu, “Diffusion convolutional recurrent neural network: Data-driven traffic forecasting,” in *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018.

[49] D. Bahdanau, K. H. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[50] J. Cheng, L. Dong, and M. Lapata, “Long short-term memory-networks for machine reading,” in *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, 2016. doi: 10.18653/v1/d16-1053.

[51] A. Vaswani *et al.*, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, 2017.

[52] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, “Pitfalls of Graph Neural Network Evaluation,” Nov. 2018.

[53] S. Brody, U. Alon, and E. Yahav, “HOW ATTENTIVE ARE GRAPH ATTENTION NETWORKS?,” in *ICLR 2022 - 10th International Conference on Learning Representations*, 2022.

[54] M. Fey and J. E. Lenssen, “Fast Graph Representation Learning with PyTorch Geometric,” Mar. 2019.

[55] J. Zhou *et al.*, “Graph neural networks: A review of methods and applications,” *AI Open*, vol. 1. 2020. doi: 10.1016/j.aiopen.2021.01.001.

[56] Y. Li, R. Zemel, M. Brockschmidt, and D. Tarlow, “Gated graph sequence neural networks,” in *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.

[57] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, “SLIC superpixels compared to state-of-the-art superpixel methods,” *IEEE Trans Pattern Anal Mach Intell*, vol. 34, no. 11, 2012, doi: 10.1109/TPAMI.2012.120.

[58] R. C. Prim, “Shortest Connection Networks And Some Generalizations,” *Bell System Technical Journal*, vol. 36, no. 6, 1957, doi: 10.1002/j.1538-7305.1957.tb01515.x.

[59] B. Delaunay, “Sur la sphere vide,” *Bulletin de l’Académie des Sciences de l’URSS*, vol. 6, 1934.

[60] W. Y. Ma and B. S. Manjunath, “EdgeFlow: a technique for boundary detection and image segmentation,” *IEEE Transactions on Image Processing*, vol. 9, no. 8, 2000, doi: 10.1109/83.855433.

[61] K. Mikolajczyk *et al.*, “A comparison of affine region detectors,” *Int J Comput Vis*, vol. 65, no. 1–2, 2005, doi: 10.1007/s11263-005-3848-x.

[62] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Int J Comput Vis*, vol. 60, no. 2, 2004, doi: 10.1023/B:VISI.0000029664.99615.94.

[63] X. Ren and J. Malik, “Learning a classification model for segmentation,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2003. doi: 10.1109/iccv.2003.1238308.

[64] C. Liu, R. Zhao, and M. Pang, “Lung segmentation based on random forest and multi-scale edge detection,” *IET Image Process*, vol. 13, no. 10, 2019, doi: 10.1049/iet-ipr.2019.0130.

[65] L. Li, J. Yao, J. Tu, X. Lu, K. Li, and Y. Liu, “Edge-based split-and-merge superpixel segmentation,” in *2015 IEEE International Conference on Information and Automation, ICIA 2015 - In conjunction with 2015 IEEE International Conference on Automation and Logistics*, 2015. doi: 10.1109/ICInfA.2015.7279427.

[66] I. Hong, J. Clemons, R. Venkatesan, I. Frosio, B. Khailany, and S. W. Keckler, “A real-time energy-efficient superpixel hardware accelerator for mobile computer vision applications,” in *Proceedings - Design Automation Conference*, 2016. doi: 10.1145/2897937.2897974.

[67] C. Y. Ren and I. Reid, “gSLIC: a real-time implementation of SLIC superpixel segmentation,” *University of Oxford, Department of Engineering Science*, 2011.

[68] M. Van den Bergh, X. Boix, G. Roig, and L. Van Gool, “SEEDS: Superpixels Extracted Via Energy-Driven Sampling,” *Int J Comput Vis*, vol. 111, no. 3, 2015, doi: 10.1007/s11263-014-0744-2.

[69] J. Yao, M. Boben, S. Fidler, and R. Urtasun, “Real-time coarse-to-fine topologically preserving segmentation,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015. doi: 10.1109/CVPR.2015.7298913.

[70] Pradhan Ronaj and KC Santosh, “Analyzing pulmonary abnormality with superpixel based graph neural network in chest x-ray,” *6th International Conference on Recent Trends in Image Processing and Pattern Recognition (RTIP2R)*, 2023.

[71] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[72] F. Errica, M. Podda, D. Bacciu, and A. Micheli, “A FAIR COMPARISON OF GRAPH NEURAL NETWORKS FOR GRAPH CLASSIFICATION,” in *8th International Conference on Learning Representations, ICLR 2020*, 2020.

[73] S. Nakarmi and KC Santosh, “Active Learning to Minimize the Risk from Future Epidemics,” in *Proceedings - 2023 IEEE Conference on Artificial Intelligence, CAI 2023*, 2023. doi: 10.1109/CAI54212.2023.00145.