MEDICAL IMAGE CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

2024
PhD. THESIS
COMPUTER ENGINEERING

Pshtiwan Qader RASHID

Thesis Advisor
Prof. Dr. İlker TÜRKER

MEDICAL IMAGE CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

Pshtiwan Qader RASHID

Thesis Advisor
Prof. Dr. İlker TÜRKER

T.C.
Karabuk University
Institute of Graduate Programs
Department of Computer Engineering
Prepared as
PhD. Thesis

KARABUK
July 2024

I certify that in my opinion the thesis submitted by Pshtiwan RASHID titled "MEDICAL IMAGE CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS" is fully adequate in scope and in quality as a thesis for the degree of PhD.

Prof. Dr. İlker TÜRKER .....
Thesis Advisor, Department of Computer Engineering

This thesis is accepted by the examining committee with a unanimous vote in the Department of Computer Engineering as a PhD thesis. 19/07/2024

Examining Committee Members (Institutions)		Signature
Chairman : Prof. Dr. Ergin YILMAZ	(BEÜN)	.....
Member : Prof. Dr. İlker TÜRKER	(KBÜ)	.....
Member : Assoc.Prof.Dr. Dr. Serhat Orkun TAN	(KBÜ)	.....
Member : Assist.Prof.Dr. Sait DEMİR	(KBÜ)	.....
Member : Assist.Prof.Dr. Veli BAYSAL	(BARÜ)	.....

The degree of PhD by the thesis submitted is approved by the Administrative Board of the Institute of Graduate Programs, Karabuk University.

Assoc.Prof.Dr. Zeynep ÖZCAN .....
Director of the Institute of Graduate Programs

"I declare that all the information within this thesis has been gathered and presented in accordance with academic regulations and ethical principles and I have according to the requirements of these regulations and principles cited all those which do not originate in this work as well."

Pshtiwan Qader RASHID

ABSTRACT

PhD. Thesis

MEDICAL IMAGE CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

Pshtiwan Qader RASHID

Karabuk University
Institute of Graduate Programs
Department of Computer Engineering

Thesis Advisor:
Prof. Dr. İlker TÜRKER
July 2024, 106 pages

The implementation of deep learning methods in medical image classification domain has yielded many progress in developing reliable systems for diagnosis of diseases. Researchers have conducted numerous studies to develop models with improved accuracy. Computed tomography (CT) scans served as an important imaging technique for quickly diagnosing lung diseases via deep learning techniques.

This doctoral dissertation presents a novel approach for diagnosing COVID-19 disease with enhanced accuracy, as well as an innovative method for detecting COVID-19 by integrating the U-Net model with a Graph Convolutional Network (GCN) to develop a feature-extracted GCN (FGCN). We use the U-Net model for image segmentation and feature extraction. We utilize the derived characteristics to construct an adjacency matrix that represents the underlying graph structure. We also feed the original image and the image graph with the greatest kernel to the GCN.

This technique involves employing graph convolutional networks (GCN) with different layer configurations and kernel sizes to extract important features from CT scan images. To generate integrated input graph data, we combine these graphs and feed them into a graph convolutional network (GCN), which also incorporates a dropout layer to minimize overfitting during the COVID-19 diagnosis.

Unlike previous studies that only took deep features from convolutional filters and pooling layers without considering the nodes' spatial connectivity, we use graph convolutional networks (GCNs) for categorization and prediction. The developed model stands out as it is the first to view CT scans of the lungs as a graph of characteristics, categorized by a graph neural network model. Furthermore, it outperforms the latest methods proposed for COVID-19 detection in the literature. This allows us to identify spatial connectivity patterns, resulting in a significant improvement in association. Our study shows that the suggested structure, called the feature-extracted graph convolutional network (FGCN), is better at finding lung diseases than other recently suggested deep learning structures that don't use graph visualizations. The suggested approach surpasses several transfer learning models frequently employed for medical image diagnosis, emphasizing the capacity of the graph representation to abstract information beyond what traditional methods can do. Furthermore, we contrast the proposed FGCN model with six widely used transfer learning models: DenseNet201, EfficientNetB0, InceptionV3, NasNet Mobile, ResNet50, and VGG16. We observe that the FGCN outperforms various transfer learning models. These results highlight the capacity of the graph-induced approach to represent abstract concepts, making it appropriate for similar medical diagnosis tasks.

Key Word : COVID-19, Graph Convolutional Networks, Graph Representative Learning , Medical diagnostic imaging, Deep Learning.

Science Code : 92431

ÖZET

Doktora Tezi

GRAFİK KONVOLUSİYONEL AĞLARLA TIBBİ GÖRÜNTÜ SINIFLANDIRMASI

Pshtiwan Qader RASHID

Karabük University
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı:
Prof. Dr. İlker TÜRKER
Temmuz 2024, 106 sayfa

Derin öğrenme yöntemlerinin tıbbi görüntü sınıflandırma alanında uygulanması, hastalıkların teşhisi için güvenilir sistemlerin geliştirilmesinde birçok ilerleme sağlamıştır. Araştırmacılar, yüksek doğruluğa sahip modeller geliştirmek için çok sayıda çalışma yürütmüştür. Bu çalışmalarda bilgisayarlı tomografi (BT) taramaları, derin öğrenme teknikleri aracılığıyla akciğer hastalıklarının hızlı bir şekilde teşhis edilmesi için önemli bir görüntüleme tekniği olarak öne çıkmıştır.

Bu doktora çalışmasında COVID-19 hastalığını yüksek hassasiyetle teşhis etmek için, U-Net modelini Graph Convolutional Network (GCN) ile entegre ederek, yeni bir yöntem sunulmuştur (FGCN). Resim segmentasyonu ve özellik çıkartımı için U-Net modeli kullanılmıştır. Çıkartılan özellikler, temel graf yapısını temsil eden bir matris oluşturmak için kullanılmıştır. Ayrıca GCN'ye orijinal görüntüyle birlikte en büyük çekirdeğe sahip graf temsili de eklenmiştir. Böylece önerilen teknik, farklı

katman yapılandırmaları ve çekirdek boyutları ile GCN kullanılarak BT tarama resimlerinden önemli özellikler çıkarılmasını sağlar. Entegre giriş graf verileri oluşturmak için, bu graflar birleştirilmiş ve bir graf konvolüsiyonlu ağ (GCN) içine aktarılmıştır. Bu ağ ayrıca, COVID-19 teşhisi sırasında aşırı öğrenmeyi en aza indirmek için bir çıkış katmanı da içermektedir.

Sadece evrişimli filtrelerden ve havuzlama katmanlarından elde deilen derin özelliklerin kullanıldığı ve düğümler arası uzaysal bağlantıların dikkate alınmadığı eski çalışmaların aksine bu çalışmada uzaysal bağlılığın da önemli olduğu GCN ağları sınıflandırma ve tahminleme görevi için kullanılmıştır. Geliştirilen model, akciğerlerin CT taramalarını, bir graf sinir ağı modeliyle sınıflandırılabilir bir karakteristik graf olarak gören ilk model olarak öne çıkmaktadır. Ayrıca, literatürde COVID-19 tespiti için önerilen en son yöntemlerin performansını da geride bırakmıştır. Bu yaklaşım, uzaysal bağlantı biçimlerini tanımlamamıza olanak sağlamakta, bu da temsilde önemli bir iyileşmeyi beraberinde getirmeketdir. Çalışmamız, önerilen yapının (FGCN) akciğer hastalıklarını bulmada, son zamanlarda graf temsilleri kullanmayan derin öğrenme yapılarına kıyasla daha iyi olduğunu göstermektedir. Önerilen yaklaşım, tıbbi görüntüleme teşhisi için sık kullanılan transfer öğrenme modellerinin performansının ötesine geçerek, geleneksel yöntemlerin soyutlama kapasitesinin grafik temsilleri ile aşılabileceğini vurgulamaktadır. Önerilen FGCN modelinin karşılaştırıldığı altı yaygın transfer öğrenme modeli sırasıya DenseNet201, EfficientNetB0, InceptionV3, NasNet Mobile, ResNet50, ve VGG16'dir. FGCN'nin bu transfer öğrenme modellerinden üstün sınıflandırma sonuçları verdiği gözlemlenmiştir. Bu sonuçlar, graf temsilli yaklaşımın soyutlama kapasitesini vurgularken, benzer tıbbi teşhis görevleri için de kullanılabilirliğini göstermektedir.

Anahtar Sözcükler : COVID-19, Graph convolutional networks, Graph representative learning, Medizinische diagnostische bildgebung, Deep learning.

Bilim Kodu **:** 92431

FOREWORD

I would like to express my deepest gratitude to my supervisor Prof. Dr. İlker TÜRKER, for their guidance, support, and encouragement throughout the course of this research. Your expertise, patience, and insights have been instrumental in shaping this work.

I am also grateful to Assoc. Prof. Dr. Serhat Orkun TAN, Assist.Prof.Dr. Sait DEMİR, Prof. Dr. Ergin YILMAZ, and Assist.Prof.Dr. Veli BAYSAL for their constructive feedback and suggestions that have greatly enhanced the quality of this thesis.

I would like to thank my department at Karabuk University for providing the resources and facilities required to complete this research. Their assistance has been critical to achieving this thesis's results.

My heartfelt thanks go to my family, especially my wife, my mother, and my children, for their unwavering support, understanding, and encouragement throughout this challenging process.

Finally, I would like to thank anyone who has helped me, whether direct or indirect, in completing this work.

Thank you all for your immense support and encouragement.

TABLE OF CONTENTS

Page

Page

Page

Page

INDEX OF FIGURES

Page

Page

INDEX OF TABLES

Page

INDEX OF SYMBOLS AND ABBREVIATIONS

ABBREVIATIONS

AI : Artificial Intelligence
DL : Deep Learning
AM : Attention Mechanism
RF : Random Forest
EM : Electromagnetic Energy
CT : Computed Tomography
ML : Machine Learning
GML : Graph Machine Learning
ANN : Artificial Neural Network
GRL : Graph Representation Learning
CNN : Convolutional Neural Networks
GNN : Graph Neural Network
GCN : Graph Convolutional Network
NLP : Natural Language Processing
CAD : Computer Aided Diagnostics
MRI : Magnetic Resonance Imaging
GAN : Generative Adversarial Networks
FCN : Fully Convolutional Network
RNN : Recurrent neural networks
SVM : Support Vector Machine
AUC : Area Under the Curve
SGD : Stochastic Gradient Descent
ECC : Edge-Conditioned Convolution
ReLU : Rectified Linear Unit
DCNN : Deep Convolutional Neural Network

LSTM : Long Short-Term Memory

ARDS : Acute Respiratory Distress Syndrome

FGCN : Feature extracted Graph Convolutional Network

GCNN : Graph Convolutional Neural Networks

GG-NNs : Gated Graph Neural Networks

CGCNN : Cell-Graph Convolutional Neural Network

……..

CHAPTER 1

INTRODUCTION

Over the past few decades, we have witnessed the arrival of the big data era. As the amount of image data and demand for intelligent processing on it grows exponentially, standard image classification techniques suffer from restrictions and underestimation of a sufficient level of detection capability [1, 2]. At the end of the 1980s, neural networks gained prominence in the fields of artificial intelligence (AI) and machine learning (ML), thanks to the development of numerous effective learning techniques and network designs [3]. Hinton et al. first proposed the expression "Deep Learning" (DL), based on the concept of an artificial neural network (ANN), in 2006.

DL techniques have emerged to be more progressive in image recognition and classification tasks. Additionally, DL served as a gold standard in computer vision, video analysis, and speech recognition, as well as an emerging subclass of artificial intelligence. DL representations have achieved great success in many areas, with the advantage of automatic feature generation on unstructured data. It has a wide range of accessibility, including signal processing, time-series and textual content prediction, computational bioinformatics, etc. [4]. DL provides a set of machine learning methods and algorithms that use structures composed of numerous nonlinear transformations to represent high-level abstractions in data. Artificial neural networks (ANNs) are the basis of DL technologies. The neurons are the building blocks of all neural networks that offer deep learning processes. Simple building units, specifically neurons, relate to deep and fully connected layers in a multi-layer context [5].

The AI community has witnessed numerous studies about DL applied to medical diagnosis, most of which have the potential to outperform doctors' estimations. A

generic example would be lung disease, which is one of the most common diseases in the world, affecting millions of people each year. Early detection of a patient's nodules can improve the survival rate of lung cancer. However, the detection of pulmonary nodules requires special attention from radiologists. In the precancerous stage, the size of the nodule is relatively small, and it is difficult to distinguish it from other benign tissues. Even for a professional radiologist, the visual noise resolution of computed tomography (CT) scans makes it difficult to make a diagnosis. Existing systems, such as the Lung Imaging Reporting and Data System (LungRADS), encounter many false-positive problems during the detection of lung nodules [6]. To overcome these problems, artificial intelligence (AI) models are essential to detecting nodules in every hospital [7].

Recently, there has been widespread use of Graph Representation Learning (GRL), a new technique that focuses on feeding deep learning architectures with higher-order representations of entities. [7]. Some of the DL models developed for graph-based representations include Graph Neural Networks (GNNs), Gated Graph Neural Networks (GG-NNs), and Graph Convolutional Networks (GCNs) [8]. GCNs are one of the most adaptable data structures, and they provide access to the exceptional expressive power of graph representations. Several areas have effectively used machine learning trends to extract and predict graph statistics, as well as solve complex factors and their relationships. GCNs specialize in processing non-Euclidean data, including graphs. When GCN performs the convolution on the input image, it uses the graph Laplacian operator instead of a fixed grid of one- or two-dimensional Euclidean structured data [9].
In physical and social networks domain, GCN is the primary choice for classification tasks.

In this research, our goal to investigate the efficiency of GCN-based classification in medical diagnosis. The current study's main procedure is the segmentation and classification of lung diseases using U-Net and GCNs, inspired by the common approach for bioinformatics datasets. We also aim to introduce efficient graph representations for these segmented images, serving as an input to GCNs. To prove the efficiency of the graph representations of medical images, we compare them with

ground-truth classification results from the most recent studies using unstructured medical images and more common deep learning architectures like CNNs .

1.1. GRAPHS

In the domain of graphs, we can characterize objects based on how they connect with each other. A graph is an intuitive method to depict a group of items and their interconnections. According to Zhang et al. [10], Diagrams offer a common framework for describing numerous intricate systems. Complex systems underlie everything around us such as wired communication networks connecting electronic devices, thousands of proteins and genes communicating to control life, and hundreds of millions of neurons in our brains concealing human ideas [11]. Researchers have structured these complex systems as graphs.

For more than ten years, researchers commonly use specialized neural network structures known as graph neural networks, or GNNs, that can operate on graph-structured information [12]. Many real-world applications, including computer networks, molecular structure analysis, natural language processing (NLP), and image processing, can use graph-structured data. In general, we describe images as matrices (e.g., 512x512x3 floats) and think of them as rectangular squares of image channels. One can also conceptualize images as regular graphs, where each pixel serves as a vertex, connected to neighboring pixels by edges. Every non-border pixel has exactly eight neighbors, and each vertex stores data in the form of a 3-dimensional vector that represents the pixel's RGB value, as shown in Figure 1.1, and the image graph.

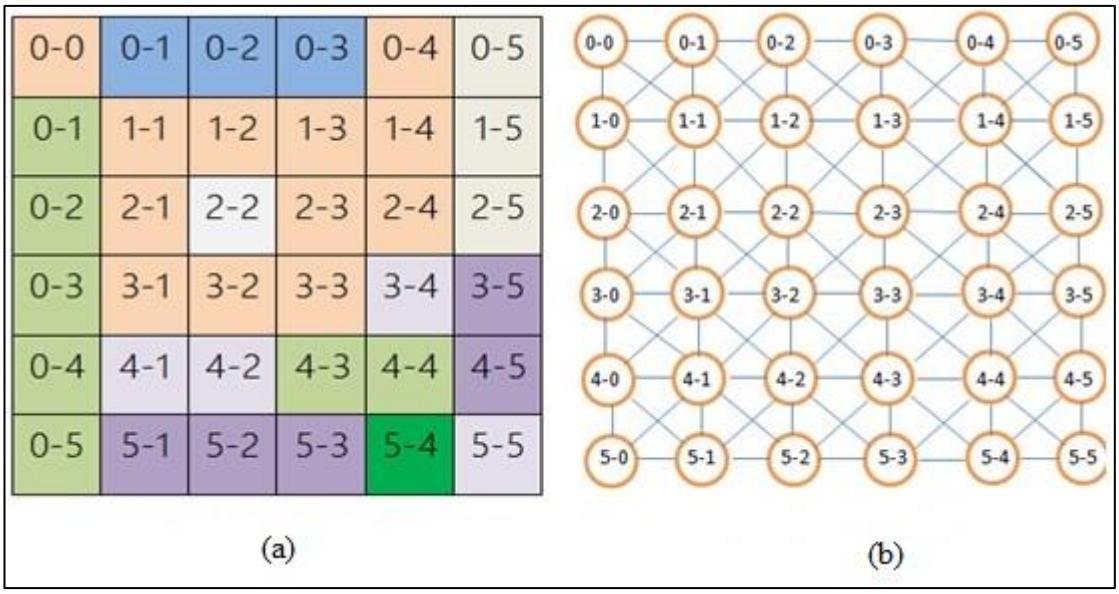

Figure 1. 1. a) Image pixels, b) Graph of the image.

1.1.1. Graph With Weighted Edges

A Graph with weighted edges is defined as the numerical weights assigned to each node in a graph. The result is a unique labeled graph, also known as a weighted graph, where the labels are numerical values commonly interpreted as positive. The weight indicates the degree of connection between the two points. For image segmentation, we use weighted graphs, in which the weight of their links represents the similarity between two pixels. A weighted graph assigns a numerical label, known as a weight, to each edge [13]. Assigning weights or other values to a graph's edges may be required. A tripling $G = (E, N, w)$ can be used to describe a "weighted" or labeled set of the edges in a graph, in which $w: E \rightarrow (N)$ node is a function that maps edges or directed edges to their values and node is a collection of possible items. For weighted graphs, the node can take any form, typically representing the real numbers. Several representations exist for a weighted graph. The initial representation directly reflects the function w. We can show this in Figure 1. 2 that the directed and undirected weighted graphs associate each edge with its corresponding value.

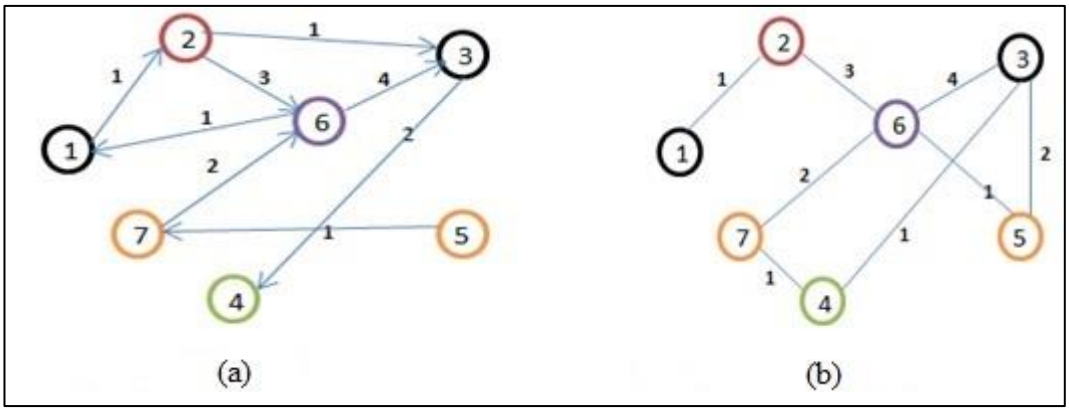

Figure 1. 2. Weighted a) Directed , and b) Undirected Graph representation.

1.1.2. Attributed Graph

Networks in the real world include nodes that have properties given in an attribute graph denoted by X. These properties, typically expressed as a vector X_v, which is describe the nodes' characteristics. Recent research on graph-structured data has mainly concentrated on modeling graphs that lack attributes [10]. There has been limited work on graph-structured data that incorporates both real-world structural features and associated attributes. Figure 1.3 illustrates an attributed graph containing five nodes and six edges, with each node assigned an attribute vector.

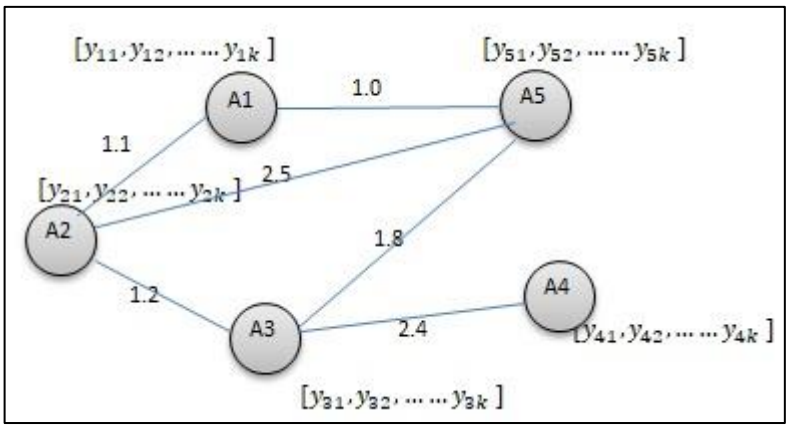

Figure 1. 3. Attributed graph.

1.2. GRAPH MACHINE LEARNING

Graph Machine Learning (GML) is a subfield of machine learning that develops algorithms to learn from data organized in a graph structure. Graphs consist of nodes (vertices) and edges (connections between nodes) and are used to represent various real-world data, including social, biological, and communication networks. GML is a specialized area within machine learning that focuses on the use of graph-structured data to perform various tasks. Here are some categories for GML:

1.2.1. Supervised Gml

Figure 1.4 displays three of the most popular GML tasks for supervised learning:

i. *Node Prediction:* This method determines whether a node attribute is continuous or discrete. As shown in Figure 1.4 (a), we can think of node predictive properties as predicting a description about an item, such as which class to assign an item to and whether to recognise an account on a financial services platform as fraudulent.

ii. *Link prediction*: It involves in speculating about the existence of a relationship between two nodes and possibly estimating certain aspects of the link. Link prediction is useful in applications such as bioinformatics, which predicts drug and protein interactions; recommendation systems, which determine what a user will want to buy or interact with next; and entity resolution, which determines whether two nodes reflect the same underlying entity, as shown in Figure 1.4 (b).

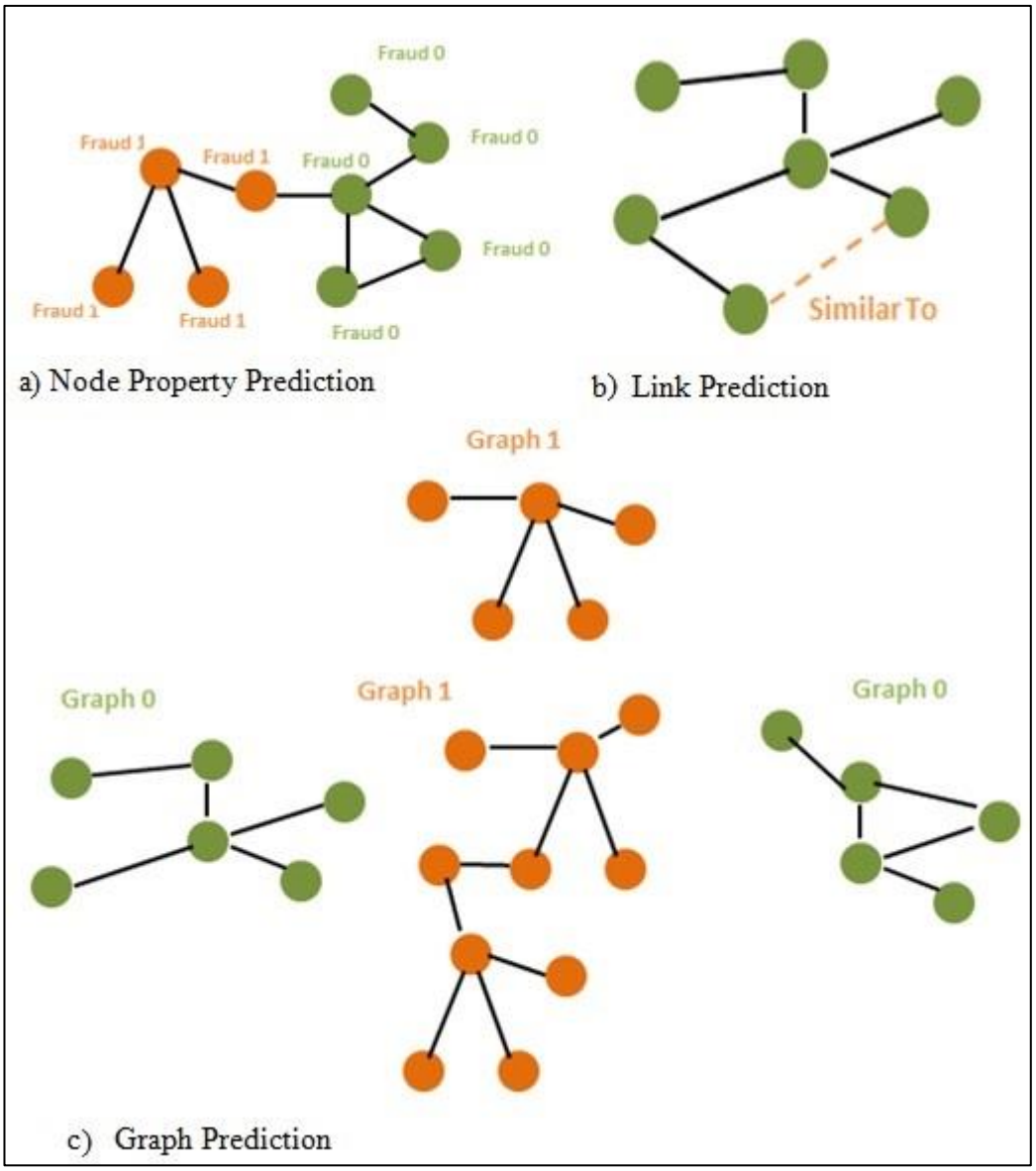

Figure 1. 4. a) Node, b) Link, and c) Graph prediction respectively in supervised ML.

Due to its usefulness in practical applications like e-commerce and friend recommendations, as well as networks for finding potential collaborators, link prediction (LP) is gaining a lot of attention. The LP problem's goal is to predict whether links will exist or not in the future. Despite LP exploration over the past two decades, Jon Kleinberg and David Liben-Nowell's work has significantly impacted this field and garnered significant attention recently [13]. The task entails predicting the continuous or discrete characteristics of a subgraph or graph.

iii. *Graph Prediction:* Predicting a graph or subgraph's discrete or continuous attribute, as shown in Figure 1.4.c. When every object is modeled as a separate graph for prediction rather than as nodes inside a larger graph that represents an entire dataset, graph property prediction comes in handy. Individual graphs may represent chemicals or proteins that you want to make predictions about in use cases such as drug development, material science, and bioinformatics.

1.2.2. Unsupervised GML

Unsupervised learning is a type of machine learning that involves processing data without human supervision. Unlike supervised models, unsupervised models train on unlabeled data, allowing them to independently identify patterns and insights without explicit guidance or support. The four of the most popular GML tasks for unsupervised learning are as follows:

i. **Representation Learning:** A key concept for GML implementations is dimensionality reduction without sacrificing significant signals, as shown in Figure 1.5(a). Graph representation learning specifically achieves this by creating low-dimensional features from graph topologies, typically for use in downstream exploratory data analysis (EDA) and machine learning.

ii. **A clustering method:** known as community detection locates closely related clusters of nodes in a graph. Refer to Figure 1.5 (b), for an example real-world applications of community detection include anomaly detection, fraud detection, investigation data analysis, social network analysis, and biology.

iii. **Similarity:** In GML, similarity refers to the process of identifying and evaluating similar node pairings within a network. Numerous use cases, such as fraud detection, anomaly detection, entity resolution, and recommendation, might benefit from similarity. Node similarity algorithms K-Nearest Neibor (KNN) and topological link prediction are examples of common similarity methods. Figure 1.5 (c) indicates similarity.

iv. **Centrality and Path Finding:** Many application cases, including transportation, logistics, problems with infrastructure, fraud and finding anomalies, and recommendation, use centrality in one way or another, as shown in Figure 1.5(d). You can use path finding to evaluate the quality and availability of paths, or to identify the paths with the lowest costs in a graph. Numerous application cases involving physical systems, including supply chains, infrastructure, and transportation, can benefit from path finding.

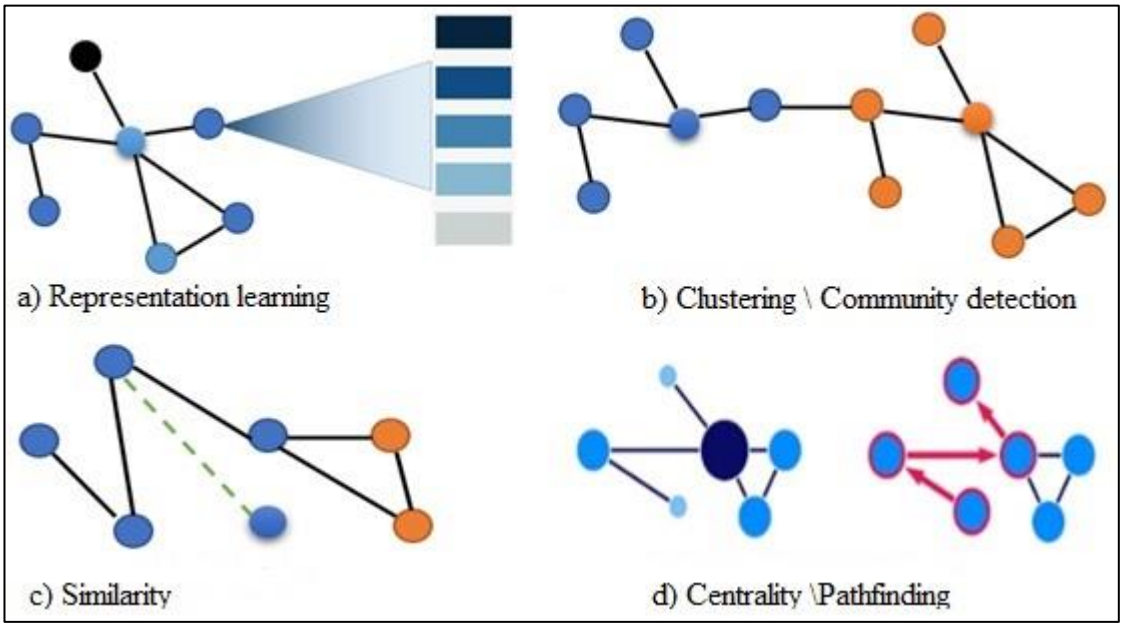

Figure 1. 5. a) , b), c), and d) Illustrates four types of unsupervised GML, respectively.

1.2.3. Semi-Supervised Learning:

Combines both labeled and unlabeled data to improve learning accuracy, particularly useful when labeled data is scarce.

1.3. TECHNIQUES FOR GRAPH ANALYSIS WITH MACHINE LEARNING

According to Langley in [14], machine learning is a scientific approach to employ algorithms and computers to generate analytical models that enable computers to gain knowledge from data without the need for specific instructions. Numerous fields

have benefited from machine learning, including the analysis of medical images, language synthesis, prediction, classification, and recommendation. Machine learning algorithms, without specific training, generate a mathematical model from sample data, or "training data," to predict "class labels". It is possible to categorize the machine learning task as either semi-supervised, supervised, or unsupervised [14]. By examining the input-label pairs from the training data, the computer attempts to develop a function that translates the input data to output labels in supervised deep learning applications. A successful supervised machine learning technique can effectively determine the class labels for new, unseen data .Using the training dataset without any already-present labels, the system tries to understand a formula that connects input data to output categories in tasks involving unsupervised deep learning. The training dataset for the hybrid semi-supervised machine learning job provides limited input-label data [14].

Turker et al. propose an intelligent detection method that converts sensory data from a CNC milling machine into a visibility graph representation. This process transforms the data, consisting of 44 machining-related attributes, into a multilayer visibility graph, resulting in a 44-layer, 128x128 adjacency matrix. Additionally, they developed a novel data augmentation technique for graph representation to increase the original dataset of 18 trials. This technique expands the dataset to 360, with each experiment represented as a multilayer graph [15].

This approach has significantly advanced various fields, such as medical image analysis, language synthesis, prediction, classification, and recommendation. Machine learning (ML) algorithms create mathematical representations based on sample data referred to as "training data" to forecast "class labels" without needing explicit guidance. As mentioned in section 1.2, ML tasks can be divided into three categories: supervised, unsupervised, and semi-supervised.

1.4. DEEP LEARNING ON GRAPHS

According to Groover and Leskovec [16], deep learning techniques are powerful machine learning techniques that can understand the complex hidden characteristics

of information from any domain without requiring specific attribute development. From image recognition [17] to natural language synthesis [18-21], the use of deep learning has revolutionized many difficult tasks, those are originating from the Euclidean domain and exhibiting a comprehensive grid-like structure.

Text data typically has 1D grid structure, while images typically have 2D structure. Because of the large volume of data coming from non-Euclidean areas [22], scholars have concentrated on deciphering complex structured data, such as data networks and 3D images. These intricate data structures are too complicated for the current techniques to manage. Because of this, when designing the architecture of neural networks based on graphs, researchers take cues from CNN [23], recurrent neural networks [24], and deep autoencoders [25]. This dissertation's main goal is to extend deep learning techniques, particularly GCNs, to weighted, attributed graphs. The standard term for this family of techniques is graph Graph convolutional networks (GCN).

DL community specifically proposed Graph convolutional networks (GCN) for detailed analysis of graph-structured data [26]. As the name suggests, a GCN's design is a useful variation on CNN's architecture, which is originally effective for assessing visual representations. GCN, which has a graph structure, generalizes the CNN convolution and pooling techniques that perform well on grid-structured data. Existing GCNs use two convolution operation methods: spatially based and spectral-based [27]. The convolution operation of spectral-based GCNs is distinguished by graph signal processing-inspired filters. These models can learn spectral filters specified via the Laplacian matrix's Eigen decomposition [26, 28, 29]. Spatial-based methods combine the features of nearby vertices to change how a vertex's attributes are shown in every layer of the neural network design [30]. Even if GCNs function at the vertex level, we can alternate graph pooling components with the GCN layer to coarsen graphs into high-level substructures. Node classification is a strong suit for the majority of graph neural network topologies currently in use. Research on graph categorization for attributed, weighted graphs has not been extensive.

1.5. MOTIVATION

The motivation for this study is driven by two primary points:

i. *Improved Feature Extraction and Focus:* The current work uses the U-Net architecture for intelligent segmentation and classification, taking advantage of its ability to extract features and adding an attention-based layer to find areas that are likely to be infectious. This approach is particularly crucial for identifying pulmonary conditions, including COVID-19 and other respiratory infections.

ii. *Innovative Algorithm Development:* The objective of this study is to create and demonstrate an algorithm that can identify lung diseases by utilizing graph convolutional networks (GCNs). The GCN approach employs feature extraction from medical images and incorporates them into the U-Net design, offering numerous benefits compared to conventional networks. These advantages include the use of high-resolution CT scan images to improve segmentation and performance, even with limited training datasets.

The study analyzes the COVID-CT scan image dataset, which consists of 2482 slices of chest CT scans. Additionally, the suggested approach has the capability to use Alternative CT scan datasets. We use deep learning algorithms because they can learn from both deep-parallel and wide-distributed models, making them cost-effective and efficient for advanced analytics.

The suggested technique effectively draws intricate information from vertices and edges by integrating convolutional neural networks (CNNs) with graphs. This will also enhance the accuracy and quality of the trained model. This makes it a reliable and effective option for detecting lung diseases.

1.6. PROBLEM STATEMENT

The classification of CT scan images presents a significant challenges, especially in the context of rapidly evolving healthcare demands such as those experienced during the COVID-19 pandemic. Traditional convolutional neural networks (CNNs), while

effective, suffer from complex computation times and often produce a fixed set of features that may not adapt well to varying medical image datasets.

Hospitals and healthcare facilities have faced increased demands for accurate and efficient CT scan analysis to detect diseases like COVID-19. However, many institutions lack the necessary expertise and resources, leading to a bottleneck in processing and diagnosing these images accurately. The shortage of medical experts exacerbates this problem, making it crucial to develop automated and efficient image classification methods.

Graph convolutional networks (GCNs) offer a promising solution to these challenges. By leveraging the relational information inherent in graph structures, GCNs can enhance feature extraction from CT scan images, leading to improved classification performance. When you combine GCNs with the U-Net model, you can make CT scan image classification even more accurate and efficient. This solves both the problem of how hard it is to compute and the need for high-quality, flexible feature extraction.

This study aims to develop a novel CT scan image classification framework that combines GCNs with the U-Net model to provide a more accurate, efficient, and robust solution for medical image recognition, particularly in the detection of COVID-19 and other lung diseases.

1.7. CONTRIBUTIONS

This thesis makes several key contributions towards the application of graph convolutional networks (GCNs) for classifying medical CT scan images:

i. *Literature Review on GCN in Medical Imaging:*

We conducted a comprehensive review of the latest articles to explore the application of the GCN algorithm for medical CT scan images. Chapter Four

presents this review, providing future researchers with a comparative analysis of various deep learning approaches used for CT scan image classification.

ii. *Hyperparameter Optimization for GCN:*

Investigated the importance of essential hyperparameters for effective GCN training. This investigation includes the impact of batch size, learning rates, and other critical parameters on GCN performance, helping to identify optimal configurations for medical image classification tasks.

iii. *Development of a Hybrid Deep Learning Architecture:*

We introduced a new classification method specifically for CT scan images by combining GCN with U-Net to create a novel GCN architecture. This hybrid deep learning model includes five layers with fully connected layers, aimed at improving the accuracy and efficiency of CT scan image classification. We compared the performance of different optimizers and activation functions to determine the most effective combination.

iv. *Evaluation of GCN Versus CNN:*

In Chapter Five, we investigated the impact of employing GCN compared to fully training CNN models. This comparison was based on experiments conducted on a large set of CT scan images, providing insights into the relative advantages of GCNs in terms of computation time and classification performance.

v. *Block-wise Fine-Tuning of GCN:*

We examined the effects of block-wise GCN fine-tuning on a CT scan dataset to determine the optimal depth for a GCN. This analysis intended to improve the performance and adaptability in GCN technique for medical image classification.

vi. Impact of Different Optimizers on GCN Performance:

The functionality of GCN was evaluated using various optimizers with different learning rates. This contribution highlights the importance of optimizer selection in training deep learning models and provides guidelines for choosing appropriate optimizers for GCN-based medical image classification.

These contributions collectively advance the field of medical image classification by providing a robust framework for integrating GCNs with other deep learning architectures, optimizing training parameters, and evaluating the comparative performance of different approaches.

1.8. THESIS ORGANIZATION

The structure of this thesis is organized as follows:

Chapter 1: Introduction : This chapter introduces the concept of graphs and their application in machine learning. It covers methods for analyzing graphs using machine learning and outlines the primary motivation and contributions of this research.

Chapter 2: Graph Convolutional Networks: In this chapter, we discuss graph convolutional networks (GCNs), including the concepts of graphs and graph signals. We delve into the spectral and spatial graph convolutional networks that underpin our thesis method and explain the integration of GCNs with convolutional neural networks (CNNs).

Chapter 3: Literature Review: This chapter provides a comprehensive review of medical image analysis and the various types of medical images used for classification. We examine deep learning applications for medical images, highlighting key studies and approaches relevant to our research.

*Chapter 4: Proposed Hybrid Method :*Here, we detail our proposed hybrid method, which combines GCNs with the U-Net model. We explain the architecture, implementation, and anticipated benefits of this approach.

Chapter 5: Experimental Results: This chapter presents the experiments conducted on deep learning algorithms using a bioinformatics dataset. We provide the results of these experiments, along with a comprehensive analysis and insights gained from the data.

Chapter 6: Discussion and Conclusions : The final chapter discusses the success of our method, summarizes the key findings, and draws conclusions based on the research outcomes. We also suggest potential directions for future research. This structured approach ensures a thorough exploration of the research topic, from foundational concepts to detailed experimental analysis and conclusive insights.

CHAPTER 2

GRAPH CONVOLUTIONAL NETWORK (GCN)

2.1. INTRODUCTION

Deep learning (DL) is a sub-field of machine learning that uses layered algorithmic design to interpret data. A convolutional neural network (CNN) is one of the most important networks in deep learning, and it is a type of feed-forward neural network that uses a convolutional architecture to extract features from the input. The convolutional neural network (CNN) has produced some outstanding results. Similar to artificial neural networks (ANN), CNN consists of neurones that self-optimize. Deep learning neural networks have grown to be one of the most representatives. CNN-based computer vision has enabled individuals to perform tasks previously thought impossible, such as facial recognition, driverless automobiles, self-service supermarkets, and intelligent medical treatments. The inclusion of the convolution layer distinguishes CNN from other neural networks. It is impossible to separate the development of CNNs from that of artificial neural networks (ANNs).

CNNs consist of three distinct layers. These layers include convolutional layers, pooling components, and fully connected layers. McCulloch and Pitts proposed the MP model as the first mathematical representation of neurons [31]. Rosenblatt proposed a single-layer perceptron model that incorporates learning capabilities into the MP model [32, 33]. A 1-D CNN, as suggested by Waibel et al., is a time-delay neural network for voice recognition [34]. Zhang et al. proposed the initial 2-D CNN, a shift-invariant ANN [35]. In addition, LeCun et al. developed a CNN for deciphering handwritten zip codes and coined the word "convolution," which is the precursor of LeNet [36]. According to Krizhevsky et al., ANNs with multiple hidden layers should be excellent at learning new

features. And "layerwise pretraining" as a way to get around the problems of deep neural network (DNN) training is what led to research in deep learning (DL).[23]. In the ImageNet Large Scale Visual Recognition Challenge (LSVRC), they used deep CNN to produce the top classification result at the time, which sparked significant research interest and contributed to the development of the current CNN [24]. Ajmal et al. mentioned CNN as a method for image segmentation[37]. Convolutional layers are vital because they reduce the number of connections needed, and Figure 2.1 represents how the convolution works. It additionally considers the spatial and temporal data included in images. The convolutional layer uses the window's kernel size to convolve the image and identify key features for categorization.

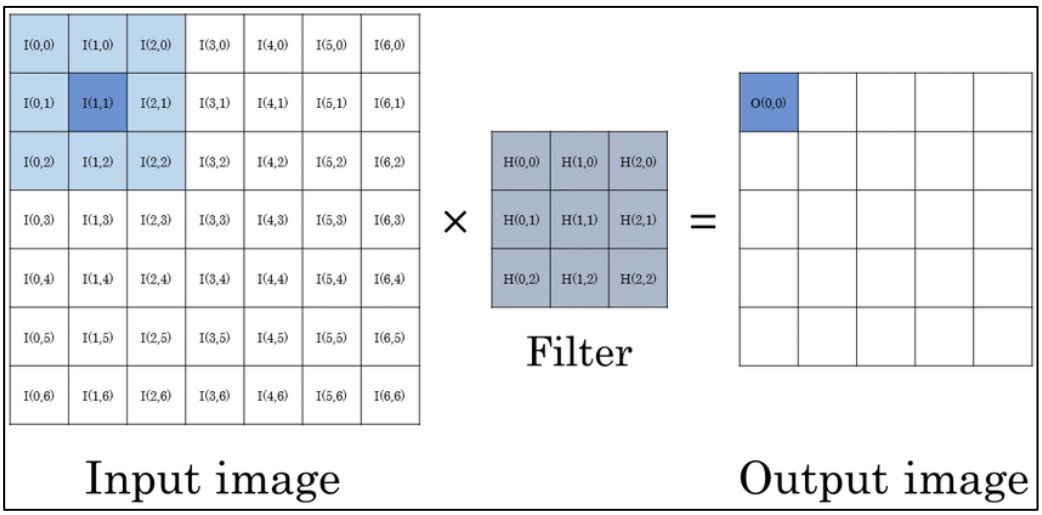

Figure 2. 1. A typical method of convolution on an image. The input image, filtering, and outputs [38]

Gori et al. hve initially introduced the idea of graphed neural networks. A Graph Convolutional Network (GCN) is a kind of neural networks that structures data, uses the structure of a graph, and convolutionally collects data from neighboring nodes. It can also be defined as a type of deep learning that employs methods to organise deep neural networks in non-Euclidean spaces, such as graphs [39]. Graph Convolutional Networks offer a high expressive capacity for learning Graph Representations and have excelled in a variety of tasks and applications.

In [40] , we are employed a graph convolutional network (GCN) model to diagnose cases of COVID-19. GCN is a deep learning architecture specifically designed for data with a graph structure. The investigation uses the SARS-COV-2 CT-Scan Dataset, which includes 1252 positively labelled CT scans and 1230 negatively labelled CT scans, for a total of 2482 CT scans. The results demonstrate that the GCN architecture outperforms the current research, which uses several models of deep learning, with an accuracy of 98.8% and an F1-score of 98.1%.

Recent years witnessed rapid development and proposal of various types of graph neural networks, including graph attention model [41, 42], graph generative models [22, 43], graph auto-encoders [44], and graph recurrent neural networks [45, 46]. With graph convolutional networks, Zhang et al. give a thorough evaluation that includes many other graphs known as graph neural networks, including graph attention networks, and gated graph neural networks [47].

2.1.1. Graphs And Graph Signals In GCN

A complex nonlinear data structure, known as a graph, represents a one-to-many connection in a non-Euclidean space. One might use an undirected graph to depict the relationships between the spectral signatures. The expression *G = (N , E)* denotes a graph without direction, with *N* and *E* representing the nodes and edge sets, respectively. In our context, the image's pixels make up the node set, while the connections between any two pixels, *Pi* and *Pj*, represent the pixels *i* and *j*.

2.1.2. Building The Adjacency Matrix

The adjacency matrix, abbreviated as *A*, describes the connections (or edges) between vertexes. You can generally calculate every element in the matrix using the radial basis function (RBF), as shown in Equ. 1:

$$A_{i,j} = \exp(- \frac{||xi - xj||^2}{\sigma^2} \quad (2.1)$$

The parameter σ regulates the RBF's width. Vectors x*i* and x*j* represent the spectral characteristics that the vertexes v*i* and v*j* connect to. Given *A*, we generate the equivalent graph, or Laplacian matrix *L*, as follows:

$$L = D - A \tag{2.2}$$

where *D* represents the degree of the diagonal matrix in the (*A*) adjacency matrix.as we see that $D_{i,i} = \sum_{j=1}^{N} A_{i,j}$.[48, 49] The normalized Laplacian matrix with symmetric can be applied as in equation to improve the graph's performance for generalization [50].

$$L_{sym} = D^{-1/2} \text{ L } D^{-1/2} = \text{I - } D^{-1/2} \text{ A } D^{-1/2} \tag{2.3}$$

where L_{sym} is a symmetric normalization Laplacian matrix, and I is the identity matrix.

2.2. SPECTRAL DOMAIN TO GRAPH CONVOLUTIONS

The convolution of two functions such as *f* and g can be represented as:

$$g(t) * f(t) \triangleq \int_{-\infty}^{\infty} f(\tau) g(t-\tau) d\tau \tag{2.4}$$

In this case, τ represents the shifting distance, and * represents the convolution process.

Based on Equation 2.4, we define the Fourier transformation of the convolution in two theorems.

The first theorem: The convolution of two functions *f* and *g* their Fourier transforms is the product of their equivalent Fourier transforms. This can be represented by equation 2.5 .

$$F[g(t) * f(t)] = F[g(t)] * F[f(t)] \quad (2.5)$$

Where * represents the multiplication operation, and *F* represents the Fourier transform.

The second theorem: The product of the inverse Fourier transforms of the two functions, f and g in convolution is equal to 2π, and the inverse Fourier transform (F^{-1}) of the convolution of this two functions can be represented as Equation 2.6.

$$F^{-1}\,[g(t) * f(t)] = 2\pi F^{-1}[g(t)].\,F^{-1}[f(t)] \quad (2.6)$$

by using the two popular theorems stated in Equation 2.5, and 2.6 [51]. The graph convolution can be generalized to the new equation as sate in Equation 2.7:

$$g(t) * f(t) = F^{-1}[F[g(t)]\,.\,F[f(t)]] \quad (2.7)$$

So, it is possible to define the Fourier transform (*F*) or to get a set of basic functions using the convolution operation on a graph.

Lemma 1: A set of eigenvectors of L can equivalently represent the basis functions of *F*.

Proof: The proof can be obtained by looking into [51]. For numerous functions lacking convergence within the domain, for instance $y(t) = t^2$, A real-valued exponential function is $e^{-\sigma t}$,and it is always accessible ,to create $y(t)e^{-\sigma t}$ converge, Therefore, the Dirichlet condition of *F* is satisfied, that is represent as Equation 2.8 .

$$\int_{-\infty}^{\infty} |y(t)e^{-\sigma t}|\,dt < \infty \quad (2.8)$$

Attaching $y(t)e^{-\sigma t}$ in to *F;*

$$\int_{-\infty}^{\infty} y(t)e^{-\sigma t}e^{-2\pi i x\xi}dt \tag{2.9}$$

And the Equation 2.9 can be rewritten as:

$$\int_{-\infty}^{\infty} y(t)e^{-st}\,dt \tag{2.10}$$

where $s = \sigma + 2\pi i\, x\xi$. Remember that a Laplace transform is expressed as Equation 2.10 Stated differently, the basic functions of F and L have the same eigenvectors. Given Lemma 1, we can conduct spectral decomposition on L based on Lemma 1. After that, we have:

$$L = U \wedge U^{-1} \tag{2.11}$$

Let $U = (u_1, u_2, \ldots., u_n)$, be the collection of eigenvectors of *L*, which serves as the foundation of *F*. Given that $L = U \wedge \mathrm{U}^{-1}$ and U is an orthogonal matrix, we have $U^{-1} = U^T$ [52]. Therefore, we can rewrite the eigenvalue decomposition of L in Equation 2.11 can also be expressed as :

$$L=U\wedge U^T \tag{2.12}$$

Equation. 2.12 states that we can represent the graph of F for f as is $\mathcal{G}F[f] = U^T f$, and we can express the reverse transformation as f = U$\mathcal{G}F[f]$. By drawing a comparison to Equation 2.7, we may express the convolution of functions *f* and g on a graph such as:

$$\mathcal{G}\,[f * g\,] = \mathrm{U}[[U^T f\,].[U^T g\,]] \tag{2.13}$$

The convolution on a graph is able to be expressed as Equation (2.12) if we write $U^T g$ as g_θ.

$$\mathcal{G}[f * g_\theta] = \mathrm{U} g_\theta U^T f \tag{2.14}$$

Where g_θ, also known as $g_\theta(\wedge)$, can be thought of as a function of L's eigenvalues (∧) with regard to the variable. Hammond et al. [53] used the Kth order truncated expansion of Chebyshev polynomials to roughly fit g_θ to reduce the computing complexity of Equation 2.14. This allows Equation 2.14 to be rewritten as:

$$\mathcal{G}[f * g_\theta] \approx \sum_{k=0}^{k} \theta_k^{-} \, T_k(\tilde{L}) f \tag{2.15}$$

Where $\mathrm{T_k}(\bullet)$, and θ_k^{-} are, the polynomials of Chebyshev respectively with regard to a variable • and the Chebyshev coefficients. $\tilde{L} = (2/\ \lambda_{max})\ L_{sym} - I$ represent the L normalization. One can further simplify Equation 2.15 by setting K = 1 and setting the greatest eigenvalue in λmax of $\tilde{\mathbf{L}}$ to 2 [28].

$$\mathcal{G}[f * g_\theta] \approx \theta\,(\mathrm{I} + D^{-\frac{1}{2}} \mathrm{A}\, D^{-\frac{1}{2}})f \tag{2.16}$$

2.2.1. Building The Graph Convolutional Networks (GCNs)

The propagation rule for graph convolutional networks (GCNs) is as follows by using Equation 2.16:

$$\mathrm{H}^{(l+1)} = h(\ \widetilde{D}^{-\frac{1}{2}} \widetilde{A}\, \widetilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)} + b^{(l)}\ ) \tag{2.17}$$

$\widetilde{\mathrm{A}}$ = A + I, and $\widetilde{D}i, i$=$\sum_j \widetilde{\mathrm{A}}\, i, j$ In order to improve stability during network training are defined as the renormalization in terms of A and D, respectively. Additionally, $\mathrm{H}^{(l)}$ is the result obtained in the l^{th} layer, and also h (•) is the activation function (in our instance, ReLU) with regard to the biases $[b^{(l)}]_{l=1}^{p}$ and the weights to-be-learned $[W^{(l)}]_{l=1}^{p}$ of all layers (l = 1, 2,..., p) [52].

2.3. OPTIMIZATION ALGORITHMS IN DEEP LEARNING

We use deep learning as a branch of machine learning, to perform complex tasks. If we deal with deep learning, it consists of an activation function, an input layer, an output layer, a hidden layer, and a loss function that make up the deep learning model. All deep learning algorithms aim to generalize the data that they use and generate predictions based on previously unseen information. We require both an optimization method and an algorithm that translates instances of inputs into samples of outputs. When translating inputs into outputs, an optimization method determines the value of the parameters that minimize the loss. The goal of optimization is to identify the model parameters that will result in the loss function L and the smallest error possible.

Equation 2.18 officially defines the function of loss L as follows: given a Dataset=$[(x_i , y_i)]_{i=1}^{N}$, where (x_i , y_i) represents a group of pairs, With a finite of cardinality N, x_i is the i^{th} example, and y_i is its label.

$$L(\theta) = argmin \frac{1}{N}\sum_{i=1}^{N} l_i\,(f(x_i;\,\theta), y_i) \tag{2.18}$$

However, for binary classifications, the binary cross-entropy, which is technically defined by Equation 2.19, and is used as the function of loss:

$$L_b(\hat{y}_i , y_i) = -\frac{1}{N}\sum_{i=1}^{n}[\, y_i \; log\hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)] \tag{2.19}$$

Where the $\hat{y}_i$ is expected label. We can present a general optimization in Equation 2.20.

$$\theta_{t+1} = \theta_t - \eta\nabla_\theta L(\theta) \tag{2.20}$$

2.3.1. Adam Optimizer In Deep Learning

Kingma et al. [54] developed the Adam optimizer to combine the advantages of the momentum approach and the RMSProp method. We use Equations 2.21 to 2.26 to modify the network parameters.

$$\theta^{i}_{t=\theta^{i}_{t-1}-\frac{\eta}{\sqrt{\hat{v}_t}+\in}\cdot\widehat{m_t}} \tag{2.21}$$

$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{2.22}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{2.23}$$

$$m_t = \beta_1 m_{t-1} + (1+\beta_1)\mathsf{G} \tag{2.24}$$

$$v_t = \beta_2 v_{t-1} + (1+\beta_2[G]^2) \tag{2.25}$$

$$G = \nabla_\theta(\theta) \tag{2.26}$$

Where (η, β_i) are the learning rate and hyperparameter, respectively, that determine how much data from the previous update is required where $\beta i \in [0, 1]$. The first moment is m_t, and the second moment is v_t.

2.3.2. RMSprop Optimizer In Deep Learning

In [55], Hinton et al. developed the RMSProp optimizer to deal with the Adagrad optimizer's systematically declining learning rate. The Equations 2.21 to 2.23 dictate the updating of the network parameters.

$$G = \nabla_\theta(\theta) \tag{2.27}$$

$$E[G^2]_t = [G^2]_{t-1} + (1-\lambda)\, G_t^2 \tag{2.28}$$

$$\theta_t^i = \theta_{t-1}^i - \frac{\eta}{\sqrt{[EG^2]_t + \in}} . \nabla_\theta L(\theta_t^i) \tag{2.29}$$

The amount of data from the previous update that is required is chosen using the λ symbol.We have avoided the gradually dropping gradients from the AdaGrad optimizer where $E[G^2]_t$ is part of Equation 2.28 by using the running average of the squared gradients. Additionally, the learning rate is η.

2.4. GRAPH FILTERING

Graph signals are capable of limited operations, such as graph filtering. In both the vertex and spectral domains, it is possible to localize a graph signal similarly to how traditional signal filtering is done in the time or frequency domains.

Frequency filtering: Using frequency filtering, the convolution of a signal with a filter in the time domain is typically represented as frequency filtering for a standard signal. Unfortunately, convolving a graph within the vertex domain is more complicated than standard signal convolution in the time domain because of the non-uniform nature of networks, where different nodes can have varying numbers of neighbors. In the case of classic signals, it's important to remember that the inverse Fourier transform of the multiplication of two signals' spectral representations equates to their convolution in the time domain. Therefore, the definition of spectral graph convolution can be expressed as follows:

$$(x * g\ y)(i) = \sum_{l=1}^{n} \hat{x}\ (\lambda 1)\hat{y}(\lambda 1)ul(i) \tag{2.30}$$

Keep in mind that the spectral domain filtering is indicated by $\hat{x}(\lambda 1)\hat{y}(\lambda 1)$. Consequently, the method of applying filtering of frequency to a signal x on the graph *G* using a filter of *y* can be described identically as demonstrated in Equation 2.30:

$$x_{out} = x * g\ y = \mathrm{U}\begin{bmatrix} \hat{y}(\lambda 1) & 0 \\ 0 & \hat{y}(\lambda n) \end{bmatrix} \tag{2.31}$$

Vertex filtering involves applying a linear approach of signal elements from neighboring nodes to process a signal within the vertex domain. For a node *i*, this process can be mathematically described as follows:

$$x_{out(i)} w_{i,i} x(i) + \sum_{j \in N(i,K)} w_{i.j}\, x(j), \tag{2.32}$$

The term N(i, K) indicates the K-hop neighbor of node *i* in a graph, while parameters representing the weights applied for the combination [56]. Using a *K*-polynomial filter [1], we can derive the interpretation of frequency filtering from the perspective of vertex filtering.

2.5. SPECTRAL GCN

GCN is subdivided into spectral, and spatial based techniques, respectively. The former is detailed in this subsection, while in the next sections we discussed. Those techniques that begin with building the frequency filter are referred as spectral-based techniques. Bruna et al. were the first to offer a noteworthy spectral-based GCN [26]. The idea for this deep model on graphs came from the original CNN. The framework includes various spectral convolutional layers take a vector representation *Xp* with dimensions *n* × *dp* as as the input for the pth layer, generating a feature map X^{p+1} with dimensions × d_{p+1}.

$$x^{p+1}(:, j) = \sigma\left(\sum_{i=1}^{d_p} \mathrm{V}\begin{bmatrix} (\theta_{i,j}^{p})(1) & 0 \\ 0 & (\theta_{i,j}^{p})(n) \end{bmatrix} V^T\, X^p(:, i)\right), \forall j = 1., dp + 1 \tag{2.33}$$

The *ith* and *jth* vectors in the input (or output) feature map are indicated as by X^{p}(:, *i*) (X^{p+1}(:, *j*)), respectively; $\theta_{i,j}^{p}$ represents a vector of accessible filter parameters at the p^{th}

layers. The activation function is denoted by σ(·), and the eigenvector of every column V is L. However, this convolutional architecture has a number of problems. First, the eigenvector matrix *V* has an O(n^3) time complexity, making it unfeasible for large-scale graphs because it necessitates the explicit computation of the graph Laplacian matrix's eigenvalue decomposition. Secondly, even with precomputed eigenvectors, Eq. (2.28)'s temporal complexity remains O(n^2). Third, we need to learn O(*n*) parameters for every layer.Additionally, these filters of non-parametric that are not limited to a specific vertex domain. The authors propose utilizing a rank-r estimation of eigenvalue decomposition to bypass these limitations. In particular, they used the initial *r* eigenvectors from *V,* which represent the most continuous graph geometry, resulting in an *O(1)* reduction in the number of parameters for every filter. Furthermore, this kind of rank-r factorization can restrict the filters if the graph exhibits a clustering structure. Henaff et al. add to [26] By proposing that the parameters for filtering in graph spectral convolutions ought to be an input smoothing kernel (such as splines) along with the corresponding interpolated weights [57]. It is possible to accomplish some degree of spatial localization in the vertex domain, as stated in [57]. However, the localization ability and computational expenses hinder the development of stronger graph representations.

As a way to fix these problems, Defferrard et al. proposed ChebNet, which implements K-polynomial filters within the convolutional layers to enhance localization [29].$\hat{y}(\lambda l) = \sum_{k=1}^{k} \theta_k \lambda_l^k$ Represents such a K-polynomial filter. When you use K-polynomial filters to combine the node properties in the K hop neighborhood [58], the number of trainable parameters drops to *O(K) = O(1)*. This lets you get an effective positioning within the vertex area. Moreover, the Chebyshev polynomial approach [53] calculates the spectral graph convolution to further minimize the computing cost.In mathematics, $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_0 = 1, T_1(x) = x$ can can be employed to build the Chebyshev polynomial Tk (x) of degree k through recursive methods. The filters are normalized by Defferrard et al. by $\widehat{\lambda l} = 2\frac{\lambda l}{\lambda\, max} - 1$ so that the scaled eigenvalues fall inside [−1, 1]. The convolutional layer is as a result:

$$x^{p+1}(:,j) = \sigma\left(\sum_{i=1}^{dp}\sum_{k=0}^{k-1} \mathrm{V}(\theta_{i,j}^{p})\ (k\ \ +1)Tk(\hat{L})X^{p}\ (:,i)\ \ \right), \forall j \\ = 1 \ldots\ldots, dp+1 \tag{2.34}$$

Where, at the −p[th] layer, ($\boldsymbol{\theta_{i,j}^{p}}$) is a K-dimensional parameter vector for the *i*th column of the input feature map and the *j*th column of the output feature map. Additionally, the authors create a successful max-pooling process using the multiple-level clustering algorithm Graclus [56] to reveal the graphs' hierarchical structure.

Kipf et al. introduced a particular type of graph convolutional network (GCN) aimed at performing semi-supervised node classification on graph structures [28]. In this framework, Chebyshev polynomial is limited to the first order (with K = 2 in Eq. (2.34)) by the authors, who also set (θ)*I* , *j*(1) = −(θ)*I* , *j*(2) = $\boldsymbol{\theta}$*i*,*j*. Furthermore, relaxing $\lambda_{max} = 2$ still ensures −1 ≤ $\hat{\lambda}l$ ≤1, ∀l = 1, · · ·, n, since the eigenvalues of $\hat{L}$ are within [0, 2]. As a result, the convolution layer is reduced to:

$$x^{p+1} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\,\tilde{D}^{-\frac{1}{2}}X^{p}\,\theta^{p}) \tag{2.35}$$

Matrix $\tilde{D}$ represents the diagonal degree matrice of $\tilde{A}\,\theta^{p}$, which is a parameter matrix of size dp+1×dp. Additionally, $\tilde{A} = I + A$ corresponds to the process of incorporating self-loops into the initial graph. Furthermore, a notable relationship exists between Equation 2.35 and the isomorphism testing behavior of the Weisfeiler-Lehman algorithm [59]. Furthermore, GCN has an easy explanation of vertex localization because Equation. 2.35 is virtually similar to gathering representations of nodes from their direct neighborhood. Therefore, it often serves as a bridge between spectral-based approaches and spatial-based techniques. For large-scale graphs, the training procedure could be memory intensive. Furthermore, learning illustrations for unseen nodes in the same network and nodes in various graphs is more challenging because GCN transduction interferes with generalization [28]. By facilitating effective minibatch training, FastGCN [60] enhances the initial GCN model in order to address the problems with GCN [28]. To guarantee that nodes *V* of graph *G* are drawn independently, and identically from nodes of a graph G′

(denoted as V′) based on a defined probability distribution *P*, it is assumed that the graph input *G* is a subgraph developed from a potentially infinite graph *G′*. In this method, Monte Carlo sampling can be used to approximate the initial convolution layer described by Equation. 2.35, indicate some i.i.d samples u_1^p,..., u_{tp}^p at *pth* layer. Graph convolution can be computed using the following method:

$$x^{p+1}(v,:) = \sigma\left(\frac{1}{tp}\sum_{i=1}^{tp}\tilde{A}\ \left(v,u_i^p\right)X^p\ (u_i^p,:)\ \ \Theta^p\right) \tag{2.36}$$

Using the graph convolution Monte Carlo estimator can lead to significant variation in estimates. To address this inconsistency, the researchers quantify the variability and evaluate the importance of the distribution by sampling *P* nodes. Additionally, Chen et al. employs control variate techniques to match the GCN framework [28], alongside rapid sampling stochastic algorithms for training [61]. Moreover, they present theoretical support for the convergence of their algorithm, regardless of the sample size during the training phase [61]. To enhance the training efficiency of GCN models, Huang et al. introduced an adaptive layer-wise sampling approach [62]. They suggest a top-down method for constructing the layers of a Graph Convolutional Networks and Suggest a method like layer-wise sampling to prevent the overexpansion of neighboring regions due to constant-size sampling. Additionally, they create a sampling strategy that explicitly addresses relevance to further minimize variation.

Along with the earlier models, which are based on Chebyshev polynomial methods, a new set of localized polynomial filters and matching graph convolutional networks has been put forward. For instance, Levie and colleagues propose approximating filters using a more sophisticated method known as the Cayley polynomial [63]. The CayleyNet model is based on the challenge of identifying narrow frequency bands, since the eigenvalues of the Chebyshev polynomials' Laplacian matrix are confined to the interval [−1, 1]. This characteristic of having narrow frequency bands is commonly seen in graphs with community structures, which can hinder the effectiveness and flexibility of ChebNet across various graph mining tasks. The Cayley filters of rank *K* display the following characteristics:

$$\tilde{y}_{c,h}\lambda_l = c_0 + 2Re\left\{\sum_{k=1}^{k} ck(h\lambda_l - i)^j ck(h\lambda 1 + i)^{-j}\right\} \quad (2.37)$$

Here, h > 0 represents a spectral expansion factor utilized to broaden the spectrum of a graph, enabling the Cayley filters to target specific frequency ranges. The parameters c = [c_0, · · ·, c_k] are the ones that need to be learned. By utilizing the Jacobi approximation further, it is possible to obtain both the localization property and the linear complexity[63]. Moreover, it is proposed that LanczosNet [64], can effectively capture the inherent multi-scale properties present in graphs, overcoming the computational restrictions encountered by many current models employing the graph Laplacian of exponentiated in their convolution of graph processes for multi-scale data analysis [29]. Specifically, the authors begin by employing the Lanczos algorithm to compute low-rank approximations of the matrix $\widetilde{A}$, resulting in the expression $\tilde{A}\approx \mathrm{VRV}^T$, where $\mathrm{BRB}^{\mathrm{T}}$ represents the eigenvalue analysis of a tridiagonal matrix *T*, and V=QB, with $Q \in R^{n\times k}$ encompassing the first K Lanczos vectors. Thus, $\tilde{A}^t \approx VR^t\, V^T$ can be used to easily estimate the t-th power of $\tilde{A}$. This leads to the formulation of the suggested spectral filter in LanczosNet as follows:

$$x^{p+1}(:,j) = x^p(:,i), V\hat{R}(1)V^T X^p(:,i) \ldots\ldots, V\hat{R}\,(k-1)V^T X^p(:,i)\Theta_{i,j} \quad (2.38)$$

When $\hat{R}(k) = \boldsymbol{f_k}([\boldsymbol{R^0}, \ldots, \boldsymbol{R^{K-1}}])$ is a matrix diagonal, and MLP $\boldsymbol{f_k}$ (multi-layer perceptron). To utilize data at multiple scales, we add short- and long-scale parameters to the previously mentioned spectral filter. In [64], they also introduce a variation for node representation learning. Xu et al. suggest using the spectral wavelet transform on graphs as an alternative to Fourier transform-based spectrum filters. This enables the resulting model to adjust the graph sizes for recording [65]. Furthermore, these fixed graphs might not be particularly effective at learning for certain tasks because most network topologies (like KNN graphs) are created manually according to the similarities between points of data. In their work, Li et al. [66], introduce a graph convolutional layer based on spectral methods that enables the simultaneous extraction of the Laplacian graph. This layer constructs an equation using the graph Laplacian, instead of just setting the filter coefficients, by integrating

the residual Laplacian concept. Nevertheless, this approach suffers from an inherent $O(n^2)$ complexity.

2.5.1. Graph Laplacian

The adjacency matrix *A* can be obtained from an undirected graph *G = [V, E]*, where *V* represents a set of vertices with *|V| = n* and *E* represents a set of edges with *|E| = m*. $A_{i,j}$ equals 1 if there is a link between vertex *i* and vertex *j*, and 0 otherwise. The equation $D_{(i,i)} = \sum_{j=1}^{n} A_{(i,j)}$ states that the degree of node *i* is equal to the sum of all the connections (edges) to that node. The matrix A's degree matrix is a diagonal matrix. The Laplacian matrix of *A* can be defined as *L = D - A*, where *D* is a diagonal matrix. Equation 2.39 represents the symmetrically normalized Laplacian matrix.

$$\tilde{L} = I - \mathrm{D}^{-\frac{1}{2}} \mathrm{A}\, \mathrm{D}^{-\frac{1}{2}} \tag{2.39}$$

The symbol *I* represents the identity matrix. Normalization's goal is to more evenly distribute the impact of nodes with a high degree across other nodes in the network.

2.5.2. Graph Fourier Transformation

The Fourier transform is a signal processing or image processing technique that breaks down an input signal from the original temporal or spatial domain into its component frequency domain basis functions. To determine the traditional Fourier transform of a one-dimensional signal *f*.

$$\acute{f}\,(\xi) = (f, e^{2\pi i \xi t}) \tag{2.40}$$

The Laplace operator's eigenfunction is represented by the complex exponential, while the frequency of $\acute{\boldsymbol{f}}$ in the spectral domain is represented by ξ.

The Laplacian matrix L represents the Laplacian operation on graphs. We can think of L's eigenvector and its corresponding eigenvalue at a given frequency as an analog

of the complex exponential. We can figure out the Fourier transform of graph signals by using the eigenvalue decomposition of $\acute{\boldsymbol{L}}$ = UΛU^{T}, where $\Lambda_{l,l}$ stands for the associated eigenvalue λ_l and u_l is the eigenvector in the lth column of U.

$$\grave{X}(\lambda) = (x, u1) = \sum_{i=1}^{n} x(i)u1^{*}(i) \tag{2.41}$$

The Equation 2.41 characterizes a signal of a graph within the vertex domain rather than the spectral domain. This allows the inverse graph Fourier transforms to be expressed as follows:

$$\boldsymbol{x}_{(i)} = \sum_{l=1}^{n} \hat{x}\,(\lambda_l)ul(i) \tag{2.42}$$

2.5.3. Graph Convolution Operator

The concept of graph Fourier transforms can establish several well-known theorems, including Parseval's identity and the generalized translation operator present in the classical Fourier transform. But we are interested in the graph convolution operator, which has some properties in common with the classical counterpart.

The convolution theorem, as i may remember, says that the point-wise multiplication of the pair of Fourier transforms of two signals is equal to the Fourier transform of a convolution among those signals. This characteristic allows us to define the convolution operation in terms of other domains. In this context, the convolution between two functions, *f* and *g*, is defined as:

$$f * g = LGF\big[GF[f] \odot \mathrm{GF[g]}\big] = U\,(U^{\mathrm{T}}g \odot U^{\mathrm{T}}f) = Ug_{\theta}(\Lambda)U^{\mathrm{T}}f = g_{\theta}(L)f \tag{2.43}$$

Where the Hadamard product is indicated by⊙ and

$$g_{\theta}(\Lambda) = diag(U^{\mathrm{T}}g) \tag{2.44}$$

Equation 2.44 is the diagonal matrix that the spectral filter coefficients correspond to. This spectral filter is especially important since different spectral-based graph convolutional networks (GCN) have different filter selections.

2.6. SPATIAL GCNS

Transferring spectral-based graph convolutional network models between graphs with varying eigenfunctions remains challenging due to their reliance on distinct eigenfunctions derived from the Laplacian matrix. As an alternative, Equation 2.32 indicates that filtering graphs in the vertex domain can extend the concept of graph convolution to the aggregation of graph signals among neighboring nodes. This section divides spatial graph convolutional networks into three primary categories: traditional models based on CNNs, models based on propagation, and several associated general frameworks.

2.6.1. Spatial Gcns Based On the CNN Architecture

It is common for CNN architectures to organize data in a grid-like structure, like images. These architectures have shown great success in a number of related tasks, including image classification[67, 68], object detection [69, 70], and semantic segmentation [23, 71]. Convolution designs leverage two fundamental characteristics of gridlike data: (1) each pixel has a constant number of neighboring pixels, and (2) the scanning sequence of images follows a regular spatial pattern, shifting from the left side to the right and from the top down, arbitrary graph data differs from images as it lacks a consistent count of neighboring elements and does not follow a specific spatial arrangement.

In order to tackle these problems, numerous studies have suggested constructing graph convolutional networks directly based on traditional convolutional neural networks (CNNs). Niepert et al. suggest resolving the mentioned difficulties by isolating locally related sections of graphs [72]. The PATCHY-SAN technique starts by arranging the nodes of graph using a labeling method that utilizes centrality

metrics like degree, PageRank, and betweenness. Following this, it selects a specific sequence of nodes to include in a designated arrangement. Furthermore, the model creates a predetermined and unchanging neighborhood for each individual node to address the issue of random neighborhood sizes. Graph labeling processes ultimately normalize the neighborhood graph to assign similar relative positions to nodes with similar structural roles. The next step involves representation learning using standard CNNs. Nevertheless, because of its reliance on graph structure for determining the spatial order of nodes, the PATCHY-SAN technique lacks the adaptability and universality required for a wider range of applications.

The LGCN model [73] aims to convert irregular graph data into a grid layout by integrating structural information with the input feature maps from the p-th layer. Unlike PATCHY-SAN, which organizes nodes using structural details [72]. For a vertex $u \in V$ in the graph G, the algorithm aggregates the input feature maps of u's adjacent nodes into a matrix $M \in \mathbf{R}^{|N(u)| \times dp}$. Here, $|N(u)|$ dipicts the count of nodes that are one hop away from u. The first r biggest values in each column of M are kept and create a new matrix called ˜M $\in \boldsymbol{R}^{\mathbf{r} \times \mathbf{dp}}$. The input feature map and the graph's structure data can be converted into a 1-D grid-like data set; the new matrix $\widetilde{M} \in \boldsymbol{R}^{\mathbf{r} \times \mathbf{dp}}$ is formed by preserving the first r biggest values for each column of M. In a straightforward manner, the input feature maps, over the network structure data, can be translated into one-dimensional grid-like data represented by $\widetilde{\boldsymbol{X}}$p $\in \boldsymbol{R}^{\mathbf{n} \times (\mathbf{r}+\mathbf{1}) \times \mathbf{dp}}$. Next, the traditional one-dimensional Convolutional Neural Network (CNN) on X_p can be used to acquire new node representations X^{p+1}. A sub graph-based training strategy is suggested to adapt the large-scale graphs of a model.

One approach to include graph data in standard CNNs involves introducing a structure-aware convolution operation, which can handle euclidean and non-euclidean data, to integrate graph data into conventional CNNs, a method that introduces a structure-sensitive convolution operation is used, capable of managing each non euclidean and euclidean data. Traditional CNN convolutions only handle data with consistent topological structures. Chang et al. establish a connection between traditional filters and single-variable functions, which they call filters functional. They then integrates graph structure into these filters functional to

improve their sensitivity to structural features [74]. To approximate this structure-aware convolution, they utilize the Chebyshev polynomial [41] because of the requirement to learn an infinite number of parameters. Another study [53] uses a different approximation due to the necessity of learning infinite parameters for this structure-aware convolution. A different study [75] transforms the traditional CNN by developing a series of adjustable fixed-size filters, with sizes spanning from 1 to K, demonstrating that these filters can conform to the graph's structure [4].

2.6.2. Spatial GCNs Employing A Propagation Based Architecture

This part emphasizes the spatial-based Graph Convolutional Network (GCN), which disseminates and combines node representations from adjacent nodes within the vertex domain. According to the significant study in reference [76], the convolution of a graph for node *u* at the p-th layer is formulated as follows:

$$\boldsymbol{x}^{\boldsymbol{p}}_{\boldsymbol{N(u)}} = \boldsymbol{x}^{\boldsymbol{p}}(\boldsymbol{u},:) + \sum_{\boldsymbol{v}\in\boldsymbol{N(u)}} \boldsymbol{x}^{\boldsymbol{p}}(\boldsymbol{v},:) \tag{2.45}$$

$$\boldsymbol{x}^{\boldsymbol{p+1}}(\boldsymbol{u},:) = \boldsymbol{\sigma}(\, \boldsymbol{x}^{\boldsymbol{p}}_{\boldsymbol{N(u)}}\, \theta^{\boldsymbol{p}}_{|\boldsymbol{N(u)}|}) \tag{2.46}$$

In the p^{th} layer, the weighted matrix for these nodes with the same degree as |N(u)| is represented as $\theta^{\boldsymbol{p}}_{|\boldsymbol{N(u)}|}$. However, for networks with a hugh number of nodes, the number of distinct values for node degree can be exceedingly high. Consequently, each layer will require the training of numerous weight matrices, potentially leading to the overfitting issue. In their paper, Atwood et al. describe DCNN, a graph convolutional network that uses graph diffusion processes to spread and combine node representations [77]. A k-step diffusion is performed by raising the matrix transition P to the power of k, where $P = D^{-1}A$. Subsequently, the diffusion-convolution process can be illustrated as follows:

$$Z(u, k, i) = \sigma\,(\theta(\mathrm{k,i})\, \textstyle\sum_{v=1}^{n} p^{k}(u, v)x(v, i)) \tag{2.47}$$

P^k combines the features where *Z(u, k, i)* signifies the i^{th} output attribute of node *u*. Here, *u, σ(•)* indicates the hyperbolic tangent function utilized as the nonlinear activation. For K-hop diffusion, the computational complexity of computing the K^{th} power of the transition matrix is $O(n^2K)$. This level of complexity is impractical for graphs with large-scale. Monti et al. propose a universal graph convolution network architecture called MoNet [78]. This framework features a patch operator that gathers signals from adjacent nodes. For a node *i* and its neighboring node $j \in N(i)$, a set of d-dimensional pseudo-coordinates *u(i, j)* is established. These coordinates are then fed into P learnable kernel functions $(w_1(u),...,w_p(u))$.

P^k combines the features, where *Z(u, k, i)* signifies the *i*-th output attribute of node *u*. Here, *u, σ(•)* indicates the hyperbolic tangent function utilized as the nonlinear activation. For K-hop diffusion, the computational complexity of computing K^{th} power of the matrix transition is $O(n^2K)$. This level of complexity is impractical for large-scale graphs. Monti et al. sugested a universal Graph Convolution Networks architecture called MoNet[78]. This framework features a patch operator that gathers signals from adjacent nodes. For a node *i* and its neighboring node $j \in N(i)$, a set of d-dimensional of pseudo coordinates *u(i, j)* is established. These coordinates are then fed into *P* learnable kernel functions $(w_1(u),...,w_p(u))$. The patch operator is expressed as $D_p(i) = \sum_{j \in N(i)} W_p\big(u(i,j)\big)x_j, p = 1, \ldots\ldots, p$, where p represents the index of the patch and *x(j)* represents the value of the signal at node j. We compute the graph convolution in the spatial domain using the patch operator.

$$(\mathbf{x} *_s \mathbf{y})(i) = \sum_{l=1}^{p} g(p) Dp(i) x \tag{2.48}$$

Many modern frameworks for graph convolutional networks, like those described in references [28, 77] can be interpreted as special instances of MoNet by appropriately choosing *u(i, j)* and the kernel of function $W_p(u)$. For example, SplineCNN [79] adopts the same structure [i.e. Equation. 2.48] but introduces a distinctive convolution kernel based on B-splines. When graphs possess information of edge attributes, the parameters of the filter weights frequently depend on the attributes of edges adjacent to a node [80] Inspired by the concept of a dynamic filter network, we introduce an edge-conditioned convolution (ECC) process to exploit edge attributes

[81]. This convolution operation is mathematically defined for the edge in between nodes *v* and *u* both are at the p-th ECC layer. This is done using the filter-generating network $\boldsymbol{F^p}$: $\boldsymbol{R^s} \rightarrow \boldsymbol{R^{dp+1 \times dp}}$, which creates the edge-specific weights matrix $\boldsymbol{\theta^p_{v,u}}$.

$$x^{p+1}(u,:) = \frac{1}{|N(u)|} \sum_{v \in N(u)} \theta^p_{v,u} \; X^p(v,:) + b^p \tag{2.49}$$

Multi-layer perceptrons execute the filtering-generating network F^p, making b^p a learnable bias. Furthermore, Hamilton et al. introduce a model for inductive representation learning called GraphSAGE, which is based on aggregation [30]. The batch algorithm in its entirety is straightforward: for any given node u, the GraphSAGE convolution layer operates as follows: (1) it utilizes a trainable aggregator to merge the representation vectors of all neighboring nodes at the current layer; (2) it incorporates node u's representation vector into this aggregated representation; and (3) it processes the resultant vector through a fully connected layer with a non-linear activation function σ (·), followed by normalization. The ppppth convolutional layer in GraphSAGE is composed of these elements:

$$x^p_{N(u)} = \; AGGREGATE_P(\{x^p(v,:), \forall_v \in N(u)\}) \tag{2.50}$$

$$x^{p+1}(u,:) = \sigma(\text{CONCAT}\big(X^P(u,:), X^P_{N(u)}\big)^{\theta^p}\big) \tag{2.51}$$

Alternative methods for aggregation include the mean, LSTM, and pooling aggregator. By employing mean aggregators, Equation (2.50) can be simplified:

$$x^{p+1}(u,:) = \sigma(\text{MEAN}\big((x^P(u,:)) \; \cup x^p \;\; (v,:), \forall_v \in N_{(u)}\big)^{\theta^p}\big) \tag{2.52}$$

This shows a close resemblance to the GCN model[28]. Moreover, we define the pooling aggregator as follows:

$$AGGREGATE_P^{POOL} = \max\big(\{\, \sigma(x^p(v,:)\theta^p + b^p), \forall_v \in \; N_{(u)}\}\big) \tag{2.53}$$

The authors have also introduced a variation where they evenly sample a set number of neighbouring nodes for each node to facilitate minibatch training [30].Nevertheless, the effectiveness of node representation learning tends to decline as the graph convolutional models increase in depth. Empirical evidence has demonstrated that a 2-layer graph convolution method consistently provides optimal performance in GCN [28] and GraphSAGE [30]. Reference [9] states that Laplacian smoothing [82] connects the convolution in Graph Convolutional Networks (GCN) [28]. Additionally, increasing the number of convolution layers leads to less discernible representations, even for nodes belonging to distinct clusters.

Xu et al. examine the expansion behaviours of two types of nodes: those in the core part of an expander-like structure ,and those in the tree section of graphs. They demonstrate that even with the identical number of propagation process, different outcomes can occur [83]. For example, the features of nodes in the core portion have a more rapid and extensive influence than nodes in the tree portion. As a result, the average spread of these features causes the node representations to become indistinguishable. To address this problem and enhance the complexity of graph convolutional models, Xu and colleagues developed the Jumping Knowledge Network, a skip connection framework inspired by the concept of residual networks used in computer vision [84] and [83].

The Jumping Knowledge Network (JKN) adeptly chooses diverse combinations from various layers of convolution. In the ultimate model layer, it autonomously merges the intermediate for every node representations. The aggregation methods for each layer feature a concatenation aggregator, a max-pooling, and an LSTM attention aggregator. Furthermore, the JKN can be incorporated with other prominent graph neural network methods, like GCN [28], GraphSAGE [30], and GAT[41].

2.7. RELATED GENERAL GRAPH NEURAL NETWORKS

Graph Convolutional Networks (GCNs), which apply convolutional aggregation techniques, belong to the broader category of graph neural networks (GNNs). Other GNN types use different aggregation methods, including gated graph neural networks (GGNNs) and graph attention networks [41] and graph attention networks

(GATs) [45] This overview highlights various GNN models, focusing on the specialized form of GCNs. One of the earliest GNN examples is described in [12], presenting the local parametric transition function f and the local output function of g. Here, $X^0(u,:)$ represents the input features of node u, while E_u denotes the edge attributes associated with node u.

$$\mathbf{H}(u,:) = f\ (\mathbf{X}^0(u,:),\ \mathbf{E}u\ ,\ \mathbf{H}(u,:),\mathbf{X}^0(N(u),:) \tag{2.54}$$

$$\mathbf{X}(u,:) = g(\ \mathbf{X}^0(u,:),\mathbf{H}(u,:)) \tag{2.55}$$

Equation 2.54 defines one common type of collection in graph neural networks, where H(u,:) and X(u,:) represent the undetected state and output representation of node u. To ensure convergence, Banach's fixed point proof [85] proposes the function *f* in [12] and restricts it to a contraction mapping. This method updates the hidden states using a traditional iterative technique. Achieving steady states through repeated iteration over the states is both inefficient and less effective. In contrast, SSE [86], employs a stochastic iterative method to learn the stable states of node illustrations. Specifically, SSE modifies the node display over *T* iterations to reach stability for a specific node u, beginning by sampling a set of nodes $\tilde{V}$ from *V*.

$$\mathbf{X}(u,:) \leftarrow (1-\alpha)\ \mathbf{X}(u,:) + \alpha T\ \theta\ [[\mathbf{X}(v,:), \forall v \in N(u)]] \tag{2.56}$$

Where Tθ is an aggregation function given by: and node $u \in \tilde{V}$.

$$T\ \theta\ [[\mathbf{X}(v,\ :),\quad \forall v\ \in\ N(u)]]\ =\sigma\ (\ [\ \mathbf{X}^0(u,\ :),\ \sum_{v\in N(u)}[\mathbf{X}(v,:),\boldsymbol{X^0}(\mathrm{v},: \tag{2.57}$$
$$)]],\theta_2)\ \theta_1$$

$X^0(u,:)$ indicates the input characteristics of node *u*. A study by Xu et al. [86] also looks at how expressive the neighbourhood aggregation-based graph neural networks that are currently available are. They explore the capabilities of graph neural networks and find that neighborhood aggregation-based models [28, 30] are only marginally more potent than the 1-D Weisfeiler Lehman implmentation for isomorphism has a drawback due to its close relationship with the process of graph

isomorphism in Weisfeiler-Lehman when applied to graph neural networks. Xu et al. [86] introduce a straightforward architecture called the Graph Isomorphism Network, which matches the expressive power of the Weisfeiler-Lehman test.

2.8. SUMMARY

In this chapter, we have presented an overview of graph convolutional networks. In deep learning context, we explained the role of graphs and graph signals in GCNs, the adjacency matrix construction process, graph convolutional networks (GCNs), and optimization algorithms. Afterwards, we outlined spectral and spatial GCNs using the graph Laplacian, the graph Fourier transformation, additionally; we discussed conventional CNN-oriented spatial graph convolutional networks. Finally, we introduced propagation-oriented spatial graph convolutional networks. The next chapter will provide a summary of the most recent methods employed in medical image analysis.

CHAPTER 3

MEDICAL IMAGE ANALYSIS REVIEW

Main ideas of deep learning (DL) and the architecture of graph convolutional networks (GCNs) are reviewed in the previous sections, which are important components of our study. In this section, we present relevant research in the field of medical image classification and analysis.

3.1. INTRODUCTION

Deep Learning architectures showed impressive performance for image categorization and other relevant computer vision tasks. In this chapter, we review image categorization techniques used to classify a variety of medical image types. The strategies discussed here mostly depend on convolutional neural networks, and we also examine their fundamental concepts and approaches while evaluating how well they perform when diagnosing medical images. We then focus on more recent context-aware models that can increase diagnostic accuracy by preserving contextual information across various parts of an input image. We also discuss image classification models that take ambiguity measures into consideration, providing us with a mechanism for measuring the level of ambiguity in a prediction. This knowledge is crucial in the field of medical imaging, where a wrong diagnosis could have serious effects. We also look at deep learning's benefits for analyzing medical image data. The chapter concludes with a discussion and review of the major discoveries and information, as well as an analysis of the research gaps in the field of medical image classification and possible solutions for discussing them.

3.2. MEDICAL IMAGE TYPES FOR CLASSIFICATION

Feature extraction, preprocessing, and classification are the three primary phases in the categorization of medical images [87]. Image classification poses a significant challenge for image analysis tasks, particularly in terms of selecting methods and strategies to utilize the output of image processing, pattern recognition, and classification methods, and then verifying these outputs with medical knowledge [87]. Medical image classification primarily aims to identify infected areas of the body while maintaining high accuracy [88]. Biomedical technologies utilize imaging methods such as computed tomography (CT), magnetic resonance imaging (MRI), and mammography to generate medical images [89]. Numerous healthcare imaging methods use radiation therapy, MRI, ultrasounds, and optical techniques as media modality. Each media modality has a unique quality and reacts differently to the organs and tissues of the human body [90]. There are four types of modality images available:

3.2.1. Projections Imaging (X-Ray)

It is electromagnetic energy (EM) that produces an X-ray, and it has a wavelength range from 0.1 to 10 nm. They transform into photons with energies ranging from 12 to 125 keV. The demand for laboratory testing as a diagnostic tool in medicine roughly coincided with the development of x-ray images. There are three basic steps in the image generation process: prereading, main reading, and image processing [89].

3.2.2. Ultrasound Imaging (Ultrasonography)

Sectional images of the human body are derived using high-frequency sounds that range from 1 to 20 MHz. The properties of the biological tissue that they travel through determine the strength of the echo-ultrasound return [91].

3.2.3. Magnetic Resonance Imaging (Mri)

Magnetic resonance imaging (MRI) uses a strong magnetic field, electromagnetic radiation, and a computer to create precise images of internal organs and structures in the human body that are often crisper, more accurate, and more likely to detect disease than images obtained using other imaging techniques. The body is tested for various illnesses, including cancer, liver, and heart diseases, and even monitoring the body of an unborn child is possible. MRI uses no ionizing radiation and is noninvasive [92].

3.2.4. Computed Tomography (CT) Scan Images

Applications for cancer screening, such as virtual colonoscopies and lung screenings frequently use CT-computed tomography. CT imaging comes in a few different forms, including dual-source and dual-energy CT, CT angiography, PET/CT, and CT perfusion. Small attenuation variations (less than 5%) can occasionally prevent the traditional x-ray imaging projection from producing satisfactory results. Below, and with less than 1% discrimination, CT enhances the subject contrast [89].

3.3. CONTEXT-AWARE METHODS FOR MEDICAL IMAGE CLASSIFICATION

The field of medical image analysis has recognized the significance of employing DL techniques to acquire contextual knowledge for image classifications. Contextual data helps to preserve the temporal relationships of a specific image region over a wide tissue region (i.e., a region's surroundings). According to several studies, contextual data is essential for decreasing abnormalities in various tissue architectures.

The context-aware method, introduced by Ruqayya et al.[93], is a two-stage process that involves two primary stages: They use a patch-based DL model based on ResNet-50. A different SVM classifier for image-based classification derives its features from an overlapping patch. They use their established model to gather contextual data within image patches. Bejnordi et al. [94] proposed a context-aware stacking CNN method to categorize breast WSIs. They built their model in two steps.

First, they trained a CNN to keep the cellular-level details from image patches. Next, they added a Fully Convolutional Network (FCN) to allow the fusion of global interdependencies between structures by encouraging predictions in nearby areas. Yan et al.'s [95] hybrid model for classifying breast cancer histology images combined convolutional and recurrent deep neural networks. It makes use of a bidirectional (LSTM) network to consider both the short- and long-term spatial correlations between image patches. They first extract feature representations from image patches of a geometric image and then insert these extracted features into the bidirectional LSTM to maintain the spatial correlations between the feature representations.

In order to capture the spatial connection between histopathology image patches, Huang et al. [96] presented a deep fusion network. They use a residual network to learn visual cues at the cellular level, up to massive tissue organization. They have constructed a deep fusion network to correct the residual network's predictions and simulate the uneven creation of distinguishing features across patches. In [97], researchers created and described a weakly supervised method for the classification of lung cancer WSIs. To provide a complete, all-around WSI explanation, they used patch-based FCN for discriminating block retrieval, as well as context-aware feature selection and aggregation.

Zhou et al. [98] presented a novel Cell-Graph Convolutional Neural Network (CGCNN) to evaluate colorectal cancer. Based on node similarity, the network transforms each large medical image into a graph by representing each element as a nucleus within the input image and displaying cellular relationships as edges within those nodes. To improve the accuracy of the model, the network makes use of the nuclei's local properties and the nodes' spatial dependencies. Xue et al. [99] proposed the use of Generative Adversarial Networks (GAN) to enhance the classification of medical images. Based on class categorization, it uses conditional GAN to produce realistic medical image patches. Rather than directly augmenting the training set with synthetic images, they developed a synthetic augmentation approach that carefully integrates GAN-generated synthetic image patches. By the aid of this structure, the quality of synthetic enhancement is upheld by selecting

synthetic images based on their label accuracy and the degree to which their features align with those of real labeled images. They showed that using selective augmentation with GAN-generated images greatly improves classification performance.

Pati et al. [100] have proposed a hierarchical cell-to-tissue graph (HACT) model to improve the structural description of histopathological tissue. Their approach uses two different types of graphs. Firstly, a low-level cell graph showcases the structure and connections among cells. Secondly, a high-level tissue graph illustrates the geographical distribution and morphological characteristics of various tissue sections. Graph representation learning methods have demonstrated comparable or even better classification results for a range of medical applications, spanning both signal and image analysis [101-103]. Furthermore, their technique captures hierarchies between cells and tissues, integrating the relative shape of cells with respect to tissue distribution. Additionally, they create a hierarchical graph neural network (HACT-Net) to transform HACT presentations into histological subtypes of breast cancer.

Li et al. first presented Hierarchical Conditional Random Field-Based Attention Mechanism (HCRF-AM) for the purpose of categorizing medical images [104]. Two parts make up the HCRF-AM model: an image classification module (IC) and an attention mechanism (AM). To identify attention regions, the AM component generates an HCRF model. The IC component trains a CNN based on the supplied attention regions. From the CNN's patch-level results, it then employs an ensemble learning strategy according to probability distribution to provide image-level outputs. Campanella et al. [105] presented a DL method based on MIL that uses the supplied diagnoses as labels for training, eliminating the need for costly and time-consuming manual pixel-by-pixel annotations. The developed framework trains DNNs using MIL, producing tile-level representations of features. Recurrent neural networks (RNNs) then provide the final classification results, using these representations to integrate the data over the entire slide.

Researchers found that the context-aware method improved the accuracy of DL-based algorithms in diagnosing medical images. Nowadays, nonetheless, it is

essential to increase model trust by adding a level of confidence to the generated models. This necessitates the use of models in clinical practice to assess a sample's uncertainty during prediction and enhance the reliability of automated diagnosis systems.

3.4. MEDICAL IMAGE ANALYSIS: MEASUREMENT OF UNCERTAINTY

The process involves measuring the uncertainty of predictions made by deep learning and machine learning methods. It is a crucial first step towards developing explainable segmentation and classification models [106]. Several newly proposed image classifications and segmentation techniques for medical image analysis incorporate measurement of uncertainty.

For instance, Simon et al. [107] identified areas of uncertainty in a model based on CNN by employing a variation map as a measure of ambiguity. Mobiny and Singh [108], suggest a Bayesian DenseNet-169 method that can generate an uncertainty measure for skin-lesion images by activating dropout layers during the testing stage. They looked into how the machine-physician collaboration may do better in skin-lesion classification with the use of Bayesian deep learning. Fraz et al. [109] presented a structure for micro-vessel segmentation for H&E-stained histology images. It includes a component for quantifying uncertainty. They have developed a calibration approach to enhance model calibration and maintain overall classification accuracy [110]. It offers an ECE, or expected calibration errors, which is a widely used statistic to measure miscalibration. Their method has proven its ability to reduce calibration error in a variety of neural network designs and datasets, making it a versatile solution for any classification problem. In a different study, Raczkowski et al. [111] presented an active, trustworthy, and accurate Bayesian network (ARA-CNN) image categorization structure for categorizing colorectal cancer medical images. Variational dropout and residual networks were the foundations for the network's construction.

This section of techniques does not have the flexibility of an ensemble of different DL models or classical ML models based on uncertainty measures. To enhance the

trustworthiness of an autonomous diagnosis system, a dynamic ensemble based on a confidence measure in image predictions is essential. This is due to its ability to only incorporate models with a certain level of confidence in the final image prediction, identify cases that the model cannot confidently classify for further study, and eliminate samples that are untrustworthy due to uncertain predictions [112].

3.5. DEEP LEARNING APPLICATIONS FOR MEDICAL IMAGE ANALYSIS

In image analysis, CNNs have been used for the processes of segmentation, detection, and classification for years. Research on machine learning distinguishes between detection, which involves drawing boundaries around several items, some of which may belong to different classes, and localization, which involves building boundaries around a single object in the image. Semantic segmentation targets objects by drawing borders around their edges and labeling them. Registration is the process of fitting one two or three-dimensional image onto another. According to the researchers, a practical machine learning system will combine a portion or all of the work into a single system, and task segmentation is not important. A single workflow would be excellent for detecting a lung tumor on a CT chest scan, localizing and segmenting it away from normal tissue, and prognosticating several treatment options, such as surgery or medication [113].

DL-based techniques have proven to be highly successful in introducing applications with exceptional performance for a variety of objectives. It is critical to draw attention to two categories of medical image analysis applications: classification and diagnoses. It is believed that biomedical image diagnostics serve as the fundamental level of differentiation across diverse class boundaries. For instance, we could categorize images of tumors as benign or malignant. DL-based diagnostic applications employ this method, first extracting attributes from input images and then using these attributes to distinguish between the structural features of benign and malignant samples. The MCUa diagnostic model [114], on the other hand, uses a classification task with multiple classes to tell the difference between four types of breast cancer samples: in situ carcinoma, invasive carcinoma, benign lesion, and

normal tissue. You can use MCUa to diagnose tasks that involve binary or multi-class classification. Furthermore, DL-based automated evaluation solutions primarily focus on discriminating between cancer grades, which is a difficult task.

3.5.1. Classification

Classification mainly stands for computer-aided diagnosis (CAD) in medical domain. As early as 1995, Lo et al. presented a CNN to identify lung nodules on chest X-rays [115]. They utilized a CNN with two hidden layers and 55 chest x-rays to determine whether or not a lung nodule is found in a specific region. The relative availability of chest x-ray images has likely driven the development of deep learning in this modality. One of the most prevalent and easily treatable health issues in the world is pneumonia, also known as a chest infection.

Rajpurkar et al. [116] used CheXNet, a better version of DenseNet [117] with 121 convolutional layers, to sort the 112,000 images from the ChestXray [115] into 14 sets that showed 14 different diseases. Shen et al. [118] used 1010 labeled CT scans of lung samples from the Lung Image Database Consortium (LIDC-IDRI) dataset to decide whether lung nodules were benign or malignant. They did this by using CNNs along with Support Vector Machine (SVM) and Random Forest (RF) classifiers. In order to extract attributes, they used three parallel CNNs, each with two convolution layers, and image patches at various scales. They created an output feature vector using the learned features and then used an RF classifier or SVM with a radial basis function (RBF) filter to identify it as benign or malignant. They discovered that their approach was resilient to various noise input levels and could accurately classify nodules with 86% accuracy.

3.5.2. Image Detection

Image detection, sometimes referred to as computer-aided detection (CADe), is an important task because failing to identify a lesion on a scan and can have serious circumstances for the patient and the physician. The goal of the 2017 Kaggle Data Science Bowl [119] was to use CT lung scans to identify malignant lung nodules.

The competition made available a total of 2000 CT scans, and Fangzhou [120] emerged victorious with a logarithmic loss score of 0.399. The Their method initially isolated local patches for nodule detection using a 3-D CNN, influenced by U-Net architecture [121]. They then sent this output into a second stage, consisting of two completely connected layers, to classify the possibility of malignancy.

In order to identify interstitial lung illness and thoraco-abdominal lymph nodes on CT scans, Shin et al. [122]examined five well-known CNN architectures. Recognizing lymph nodes is crucial since they may indicate cancer or an infection. Using advanced GoogLeNet, they were able to achieve a mediastinal lymph node detection average area under the curve (AUC) of 0.95 with a sensitivity of 85%. In addition, they provided evidence for the advantages of transfer learning, as well as the application of deep learning architectures with up to 22 layers, as opposed to the customary lower layer counts in medical image analysis. In the ILSVRC 2013 localization task, Overfeat, a CNN pre-trained on natural images, won the prize [123]. For their study, Ciompi et al [124] used Overfeat on 2-D slices of CT lung images in the coronal, axial, and sagittal planes to find nodules in and around lung fissures. They used a method that combined straightforward SVM and RF binary classifiers with a Bag of Frequencies[125], an original 3-dimensional descriptor they created.

3.5.3. Image Segmentation

MRI and computed tomography (CT) image segmentation studies focus on the brain, especially on tumor segmentation. In order to plan a surgical removal, it is MRI and computed tomography (CT) image segmentation studies focus on the brain, especially on tumor segmentation. In order to plan a surgical removal, it is particularly crucial to ascertain the precise limits of the tumor. Neurological conditions such as limb numbness, weakness, and cognitive impairment result when surgery loses too many expressive brain regions. The development of an automated solution for medical anatomy segmentation is commendable, given that doctors have traditionally carried out this laborious task by hand, meticulously sketching contours on a full MRI or CT volume stack, slice by slice.

Akkus et al. [126] conducted a comprehensive analysis of brain MRI segmentation, examining several CNN architectures and metrics employed in the process. He also described several challenges and associated datasets, including Ischemic Stroke Lesion Segmentation (ISLES), Brain Tumor Segmentation (BRATS), and Mild Traumatic Brain Injury Outcome Prediction (MTOP). Using the BRATS 2013 dataset, Havaei et al. [127] also examined gliomas and investigated different 2-dimensional CNN structures. Their algorithm executed in 3 minutes as opposed to one hundred minutes, outperforming the BRATS 2013 champion. Using a transmitted architecture, their input cascade CNN effectively funneled the output from one CNN into another. Chen et al. [128] suggested using fully connected conditioned random fields (CRFs), atrous spatial pyramid pooling, and up-sampled filters. Brosch et al.[129] also used multi-scale structure investigations to segment multiple sclerosis (MS) lesions from the brain on MRI. They used a unique strategy, but they also used a deconvolutional network that resembled a UNet design, and an encoder convolutional pathway made up of pre-trained RBMs.

3.6. DISCUSSION

This section represents the application of deep learning in medical image analysis, focusing on classification, detection, and segmentation tasks. The discussion highlights the significance of various medical image types and the effectiveness of deep learning models in improving diagnostic accuracy and efficiency. Deep learning models, particularly graph convolutional neural networks (CNNs), have revolutionized image classification in medical imaging. It can accurately classify diseases by learning intricate patterns from large datasets. For example, GCNs can classify chest CT scans to detect COVID-19, outperforming traditional methods. In medical imaging, object detection entails identifying and localizing abnormalities within images. DL models provide bounding boxes around regions of interest, facilitating precise diagnosis and treatment planning. Image segmentation is crucial for delineating anatomical structures and pathological regions. In MRI and CT scans, techniques such as U-Net and its variants have shown exceptional performance in segmenting organs and tumors. Segmentation aids in volumetric analysis, surgical planning, and radiation therapy.

Finally, we outlined that deep learning has significantly advanced medical image analysis, enhancing the accuracy and efficiency of classification, detection, and segmentation tasks. However, addressing existing challenges and continuing innovation will be critical for widespread clinical adoption of these technologies.

3.7. SUMMARY

Medical image analysis involves feature extraction, preprocessing, and classification, which are essential for identifying infected areas with high accuracy. This process discusses various imaging techniques, including computed tomography (CT), magnetic resonance imaging (MRI), and mammography. Different modalities, including X-ray, ultrasound, MRI, and CT scans, each have unique characteristics and interactions with body tissues. Context-aware methods in medical image classification leverage deep learning (DL) techniques to capture spatial relationships and improve diagnostic accuracy. For example, the GCN and ResNet-50 models enable looking at image patches in great detail, and hybrid models that combine convolutional and recurrent neural networks improve spatial correlation. The field also explores uncertainty measurement in predictions to improve model reliability and trustworthiness in clinical applications. Additionally, deep learning techniques like CNNs have significantly advanced image classification, detection, and segmentation tasks, providing high performance in diagnosing and differentiating medical conditions such as lung nodules and brain tumors. Despite these advancements, further innovation and addressing current challenges are necessary for broader clinical adoption of DL technologies in medical imaging.

CHAPTER 4

PROPOSED METHOD

4.1. OVERVIEW

Due to the significant progress in computer-assisted image acquisition in recent years, standard computer vision methods remain unsuccessful when it comes to processing large amounts of image data. Conventional machine vision techniques, for example, aren't particularly effective at distinguishing between high- and low-level characteristics in images [130]. In-depth image analysis requires first classifying and processing the image data. It also has enormous implications for the development of computer vision in the field of image processing applications. Currently, a variety of industries and fields, including aerospace, healthcare, the agricultural sector, and manufacturing, use computer-assisted image categorization research. The primary goal of image classification research is to accurately and efficiently categorize semantically related feature data within images, as well as large-scale image attributes. Once the image data is derived, the image data should be cleaned to better extract and filter its features and classify the image using a deep learning method. Researchers can utilize computer vision technology to quickly execute image analysis operations like noise reduction and feature extraction due to the growing applicability of machine learning in various fields [131].

Researchers have made significant progress in the study of images and classification, ranging from the use of artificial intelligence to the concurrent image processing technologies. One significant area of artificial intelligence is machine learning. Despite fifty years of advancements in machine learning, several issues remain unresolved,such as sophisticated image comprehension and identification, natural language interpretation, and recommendation systems [132].

Deep learning is one major field that has emerged in machine learning. In order to accomplish image classification or regression, it fully utilizes the hierarchical properties of artificial neural networks and biological neural systems for processing information. It learns low-level features and uses feature extraction techniques to acquire high-level features. Deep learning, as opposed to ordinary machine learning, uses multilayer neural networks to automatically recognize images and extract their most deeply ingrained features. Various feature learning techniques, as well as their combinations; can lead to distinct depth learning models. However, the current deep learning model's operating efficiency is competitive yet needs improvements.

Lung diseases are becoming an escalating global health issue, affecting millions and putting a substantial strain on healthcare systems around the world [132]. The SARS-CoV-2 virus, which causes COVID-19, has significantly affected human health, particularly the respiratory system, leading to issues such as respiratory distress, pneumonia, acute respiratory distress syndrome (ARDS), and other complications. [133]. Rapid and precise detection ,and diagnosis of lung illnesses are essential for optimal therapy and enhanced patient results. Conventional diagnostic ways frequently rely on the human analysis of medical pictures, which can be time-consuming and influenced by personal judgment. However, advancements in technology combined with medical science are paving the way for more accurate and efficient early detection of respiratory conditions [134].

This chapter demonstrates how to combine two deep learning architectures, GCN and UNet, to get the features of CT scan images, and then use their spatial connectivity patterns to classify into COVID and non-COVID groups by introducing the Feature Extracted Graph Convolutional Networks (FGCN), a revolutionary architecture that integrates the GCN with the UNet model. This architecture takes an arbitrary-sized medical image as input, uses an attributed graph as an output, and classifies it as COVID or non-COVID. FGCN takes out features and uses a pooling layer to filter out the graph's less important vertices while keeping its overall structure. After an effective convolution layer, the graph becomes coarser.

Figure 4.1 illustrates a sample graph that includes a set of nodes, edges, and the corresponding adjacency matrix. This matrix indicates the connections between each

pair of nodes, each entry in the matrix represents the strength of the connection between the i-th and j-th nodes, Located where the *i-th* row meets the *j-th* column. Different methods can be employed to create the graph, such as utilizing Pearson correlation, the algorithm of KNN, or distance-based techniques to determine the values within the matrix [59].

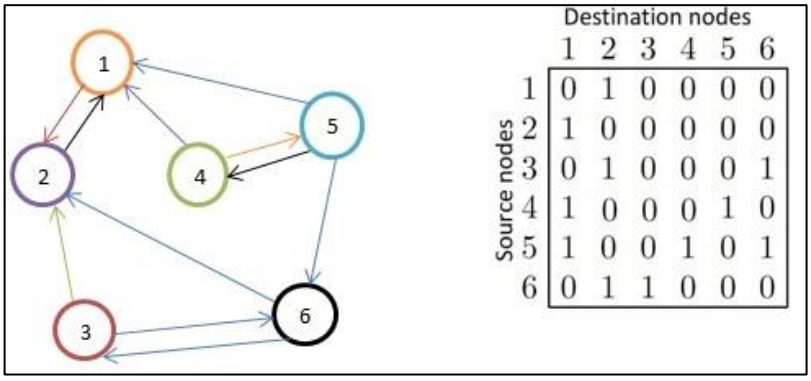

Figure 4. 1. Directed graph along with its corresponding adjacency matrix, adapted from an illustration. [135].

4.1.1. ChebNet

The convolution operation in spectral-based graph convolutional networks (GCNs) in the Fourier domain by the computation of the eigendecomposition of the graph Laplacian [26]. ChebNet inherits the principles of CNNs, with additional localized spectral filters for sophistication. Unlike traditional grid structures, these filters specialize in operating on graphs, enabling ChebNet to process complex data sets with sophisticated connectivity. ChebNet specifically uses Chebyshev polynomials, as shown in Figure 4.2, as the building blocks for its filters; those are approximating the behavior of graphs to construct localized filters. As a result, ChebNet provides higher accuracy while minimizing computational load. The Fourier transform constructs the convolution process for a spectral-based Graph Convolutional Networks (GCNs) involves calculating the eigen-decomposition of the Laplacian graph [26]. The normalization of the graph Laplacian is represented as $L = (I_N - D^{-1/2} A D^{-1/2} = U \wedge U^T)$, where D is the degree matrix, A is the adjacency matrix, U is the eigenvectors of a matrix, and Λ is the diagonal matrix containing the eigenvalues.This process can be described as the multiplication of a signal ($X \in$

R^N), by a scalar for each node, using a filter defined as $(g_\theta = diag(\theta))$, which is parameterized by ($\theta \in R^N$).

$$g_\theta * x = U g_\theta(\wedge) U^T x \tag{4.1}$$

ChebyNet, introduced by Defferrard et al. [29], circumvents the need to compute the Fourier basis is approximated using truncated Chebyshev polynomials to represent the spectrum filters.This approach utilizes a Chebyshev polynomial $T_m(x)$ of degree m [29].

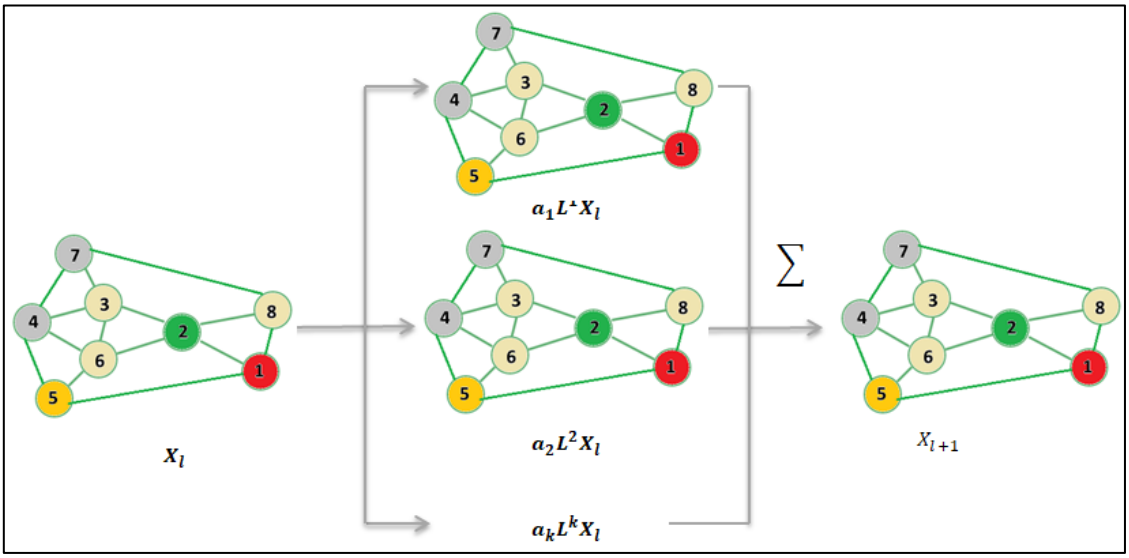

Figure 4. 2. ChebNet graph convolution [136].

$$g_\theta * x \approx \sum_{m=0}^{M-1} \theta_m T_m\left(\tilde{L}\right)_x \tag{4.2}$$

In Equation 4.2, $\tilde{L}$ is the diagonal matrix of a scaled eigenvalues with a formula $\tilde{\tilde{L}} = 2L / \lambda max - I_N$. The biggest eigenvalue of L is indicated by the symbol max. The Chebyshev polynomials are denoted by $T_m(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, where $T_0(x) = 1 \; and \; T_1(x) = x$.

ChebNet employs Chebyshev polynomials, allowing it to avoid calculating the eigenvectors of the Laplacian matrix, thus reducing computational costs. In Graph

Convolutional Networks (GCN), a graph pooling layer reduces the dimensions of the graph while expanding the area influenced by the graph filters. The layer before this in the graph convolutional network combines feature vectors into one, which is subsequently passed to a fully connected layer for producing classification outcomes.

4.1.2. Graph Convolutional Networks (GCN)

A Graph Convolutional Networks (GCN) is a form of graph neural network which applies mean pooling aggregation for spectral analysis. Introduced by Kipf and Welling [28] the GCN uses a limited first-order approximation of spectral convolutions on graphs. This approach features a simple layer-wise propagation method that converts the relationships among nodes in the graph into node features. Eliminating the problem of overfitting to the local neighborhood structure of graphs with a lot of node degrees is possible by making the convolution filter K=1. Additionally, we can simplify Equation 4.2 by approximating $\lambda \approx 2$, [28].

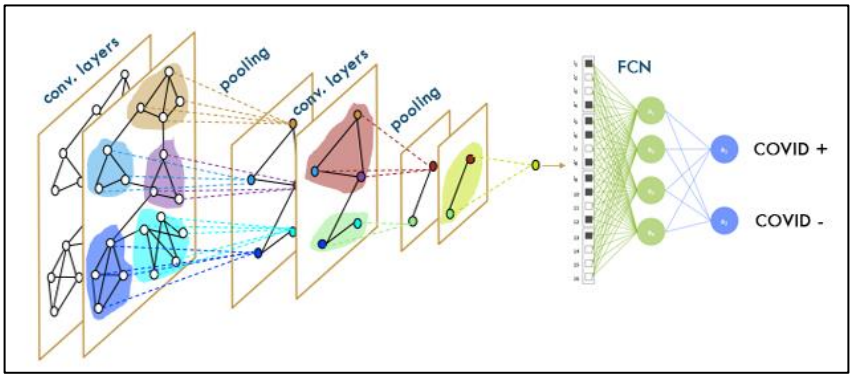

Figure 4. 3. Evaluation of graph-input data with a GCN [137].

$$g_\theta * x \approx \acute{\theta}_0 x + \acute{\theta}_1 x(L - I_N)_x = \acute{\theta}_0 x + \acute{\theta}_1 D^{-1/2} A D^{-1/2} x \tag{4.3}$$

The values $\acute{\theta}_0$ and $\acute{\theta}_1$ are not limited in Equation 4.3. GCN additionally assumes that $\vartheta = \acute{\theta}_0 = -\acute{\theta}_1$ to minimize the parameters and prevent excessive model complexity [135]. This leads to the development of a graph convolution defined as follows:

$$g_\theta * x \approx \theta\left(I_N + D^{-1/2}\ A\ D^{-1/2}\right)x \tag{4.4}$$

Layer stacking in this method can lead to numerical issues, where gradients may abruptly disappear or escalate uncontrollably. For defining the signal $x \in R^{F*C}$ with input channels of C ,and F refers to extraction filter for features, Kipf and Welling [28] elaborate on the concept as follows:

$$Z = \widetilde{D}^{-1/2}\tilde{A}\widetilde{D}^{-1/2}\ x\Theta \tag{4.5}$$

The matrix Θ, denoted as $\Theta \in R^{CXF}$, is generated by the filter bank parameters. Similarly, the matrix Z, denoted as $Z \in R^{N*F}$, is obtained by convolving the signals.

4.2. STEPS OF THE PROPOSED METHOD

This research utilizes a specific methodology: Initially, each image undergoes preprocessing steps such as normalization, filtering, and data augmentation to enhance the clarity of CT images. Then, we select images with nuclei ,and train a U-net model with this dataset. The U-net performs extraction features, resulting in a graph-based feature map for each patient. For diagnostic classification of the CT images, we use the graph-structured data as input to a modified GCN, named the "feature-extracted GCN" (FGCN). Figure 4.4 displays the block diagram of our proposed method.

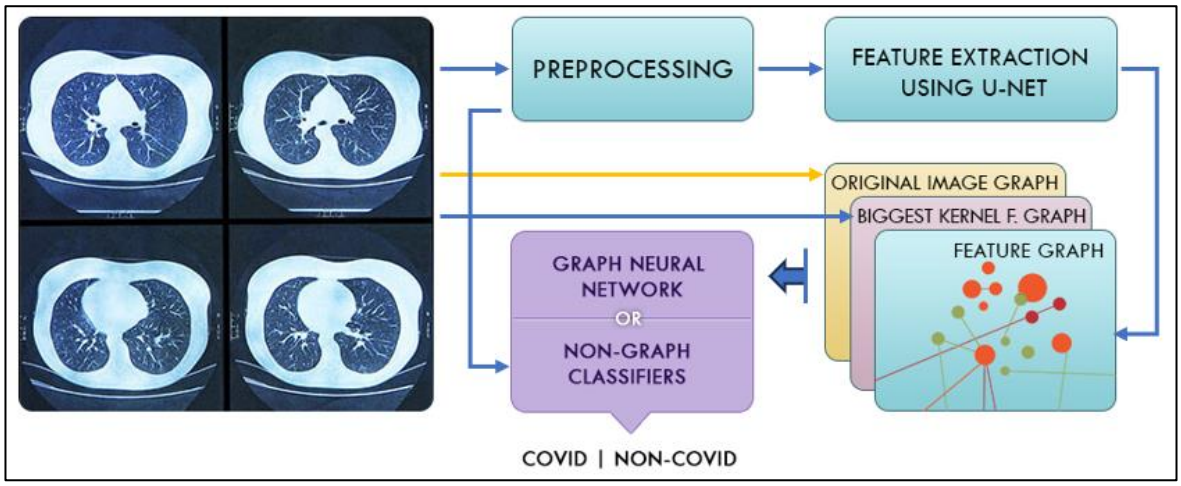

Figure 4. 4. Block diagram of the proposed work.

4.2.1. Preprocessing

The pre-processing step is designed to enhance the abstraction capabilities of 3D-CT images acquired from patients. This process consists of three phases:

i. **Filtering**: In the filtering stage, the goal is to improve the clarity of a 3D-CT image. This process involves applying a set of criteria to decide the value of each pixel in the final image, using the values from surrounding pixels.

ii. **Normalization**: This step changes the range of pixel intensity values, often known as "contrast stretching," to enhance the overall contrast of the image.

iii. **Data Augmentation**: This preprocessing strategy is commonly employed with the U-net model. It enhances the training set by introducing alterations like rotations, shifts, and symmetrical changes. This approach helps lower the model's vulnerability to noise and mitigates the chances of overfitting compared to using the unaltered data.

4.2.2. UNet Model

A standard convolutional network has a 3×3 convolution process done repeatedly. Following this convolution is a maximum pooling layer with a 2×2 size, and a ReLU activation function comes next. Every pooling procedure results in a doubling of the number of feature channels. The downsampling path aims to maintain the input image information for segmentation. The expansion path receives the data via links, as illustrated in Figure 4.5. In the upsampling process, the architecture reduces the count of feature channels by half after each phase of the expansion path. A four-block extension path is used. These blocks consist of the deconvolution layer, feature map integrated from the contracting path, 3×3 convolution layer, and activation function. A further 1×1 convolution operation is conducted in order to reduce the feature map to the necessary channel count and create the segmented image [121].

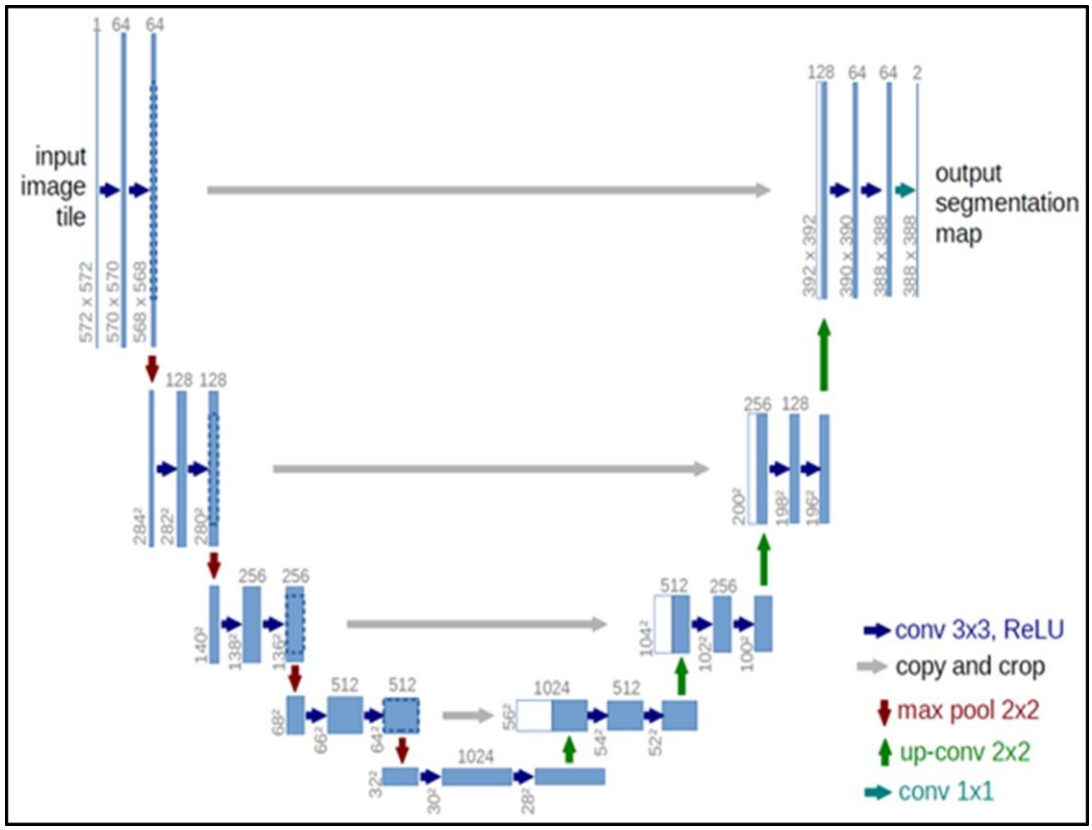

Figure 4. 5. U-net Architecture [121].

4.2.3. Classification With FGCN

4.2.3.1. Feature Extraction Using the U-Net Model

The strategies of effective extraction features are crucial for improving image classification and distribution. We use U-Net for features extraction from CT scan images. U-Net is preferred for its straightforward design, flexibility, and excellent performance in pixel-level segmentation, especially for biological images. Its well-balanced architecture is specifically crafted to support tasks in computer vision ,and image processing that are involve localization, extraction features, context modeling, and precise resolution recovery. The model utilizes both its encoder ,and the decoder components to perform an accurate and detailed output. The encoder section (or contracting path) is shown on the left side of Figure 4.5, while the decoder section (or expanding path) is shown on the right side. Blue boxes in Figure 4.5 represent multi-channel feature maps, and white boxes indicate replicated feature maps. Arrows in the figure illustrate different processes, such as max pooling, convolution, and copying and cropping operations [120].

4.2.3.2. Feature-Extracted Using Graph Convolutional Networks (FGCN)

We use an adapted form of GCN, referred to as feature-extracted GCN (FGCN), to carry out the classification task. The following outlines the procedure involved in FGCN.

Step 1: The U-Net model extracts pyramid features from an image using different layers and kernel sizes.

Step 2: We combine these extracted features to form a single combined adjacency matrix.

Step 3: The COVID-19 graph is constructed from a unified adjacency matrix that reflects the connections and identified characteristics.

Step 4: We combine the graph of the original image with the graph of the largest kernel.

Step 5: To mitigate overfitting, we merge these three graphs into one cohesive input block and process it through a Graph Convolutional Network (GCN) that includes an extra dropout layer.

Step 6: Ultimately, we conduct image classification to assess if the patient is infected with COVID-19.

4.2.4. Graph Convolution Layer

The convolutional layer serves as the fundamental component of a CNN model and is responsible for performing most of the calculations. This layer's parameter is a filter vector that can be learned, with dimensions represented as (w * h * d), where w is the filter's width in pixels, h is its height, and d refers to its depth. For example, a typical filter may have dimensions of 5x5x3, meaning it has a width and height of five pixels, with a depth of three pixels corresponding to each color channel. A CNN model generally utilizes multiple filters to capture different features of the input image. During forward propagation, each filter slides across the input volume's width and height, calculating the sum of the dot products between the filter elements and the input at every location. As the filter moves across the input volume, it produces a two-dimensional activation map. This activation map visually represents basic features of the input, such as edges or patches of color. Subsequent layers use this activation map to identify more complex characteristics of the input, like the wheels or headlights of a vehicle.

Following the convolutional layer is the rectified linear unit (ReLU), a non-linear activation function. An example of how a convolutional layer operates is shown in Figure 4.6. Here, a 3x3 filter is applied to a 5x5 input volume. The convolutions start from the top left corner of the input and proceed to the bottom right, covering the entire image. The pixel located at the top-left is referred to as the target pixel. Through convolution, new representations of this target pixel are generated. To compute the top-left pixel of the activation map, element-wise multiplication and summation are performed on the overlapping pixels from both the input and the filter.

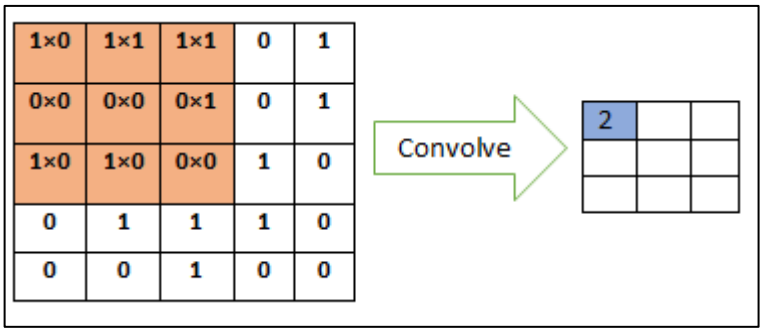

Figure 4. 6. The process of convolution creates an activation map.

Typically, a convolutional neural network (CNN) places the convolutional layer as its initial layer. Weight sharing is the primary advantage of a convolutional layer. Equation (4.6) provides a mathematical representation of this step.

$$x_m^L = f(\sum_m x_m^{L-1} * W_{mn}^L) + b_n^L \tag{4.6}$$

The symbols *m* and *n* represents the dimensions of the convolution filter. *L* represents the layer, x represents the features, *W* represents the weight, and *b* represents the basis. * denotes the convolutional operation, while *f* represents the activation function, which can be a sigmoid, rectified linear unit (ReLU), or hyperbolic tangent (tanh). You can adjust the activation methods based on the data's nature.

In this section, we introduce and construct the graph convolutional layer from the bottom up. In conventional networks, linear layers convert incoming data in a linear manner. Using a weighted matrix *W,* this process transforms the feature input x into hidden vectors h. Putting biases aside for the moment, we can consider the process as follows:

$$h = Wx \tag{4.7}$$

The relationship between vertices in graph data introduce another level of depth. Generally, networks tend to link similar vertices more frequently than different ones,

a phenomenon known as network compatibility, which makes the links significant. We can improve our node's representation by combining its attributes with those of its neighbors. We refer to this process as neighborhood gathering, or convolution. We use $\tilde{N}$ to stand for node i's surrounding area, which includes itself.

$$\boldsymbol{h_{i=\sum_{j\in \tilde{N}i} W x_j}} \tag{4.8}$$

Unlike the filters in convolutional neural networks (CNNs), each node shares and uniquely possesses the weight matrix *W*. However, unlike pixels, nodes don't have a set number of neighbors. This presents another problem. When a particular node has 400 neighbors and another has just one, how do we handle those situations? The resulting embedding h for the node with 400 neighbors would be significantly larger if we were to simply summarize the feature vectors. We can use the degree of nodes—the number of connections a node has—to normalize the outcome, provide a comparable range of values for each node, and establish comparability among them.

$$\boldsymbol{h_i = \frac{1}{deg_{(i)}} \sum_{j\in \tilde{N}i} W x_j} \tag{4.9}$$

The graph convolutional layer, which was first introduced by Kipf et al. [28], has one last enhancement. The authors noticed that characteristics from nodes with a large number of neighbors spread far more quickly than characteristics from nodes that are more isolated. Researchers proposed giving attributes from nodes with fewer neighbors higher weights in order to counteract this impact and balance the influence across all nodes. The notation for this operation is:

$$\boldsymbol{h_i = \sum_{j\in \tilde{N}i} \frac{1}{\sqrt{deg_{(i)}}\sqrt{deg_{(j)}}}} \tag{4.10}$$

4.2.5. Graph Pooling Layer

The pooling layer downsamples the input volume across the width and height dimensions. This makes the features in the activation map look like they are in a

lower-dimensional space. There are several pooling algorithms available, including max-pooling and average-pooling. Figure 4.7 illustrates the application of max-pooling to a convolutional neural network (CNN). Max-pooling involves creating an activation map by selecting the highest value within each pooling window. The pooling window is defined as an integer, a tuple, or a list of two integers that represent the window's height and breadth. The stride is an integer that determines the pooling process's step size. In Figure 4.7, max-pooling reduces the 4x4 input activation map to a smaller 2x2 size. The pooling process receives input parameters, including a window size of 2x2 and a stride size of 2. The method selects the highest value within a certain window size and then moves two pixels to the right to locate the subsequent value. In the initial pooling window, the maximum pixel value is 6. We exclude all other values and create an activation map consisting only of the highest pixel values from the image volume.

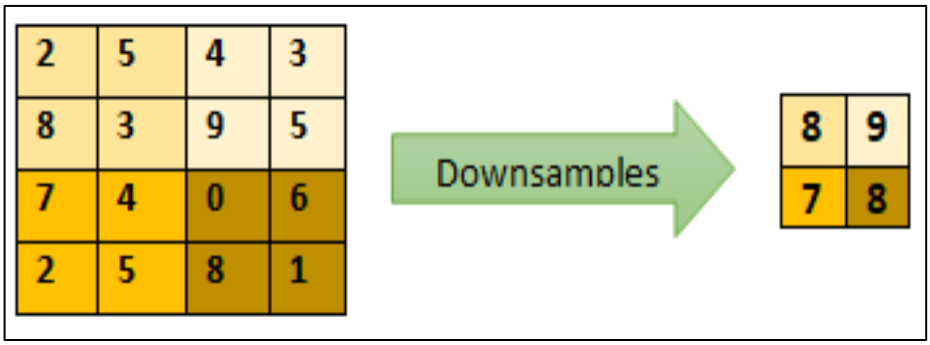

Figure 4. 7. An illustration of the max-pooling layer, which reduces the size of the activation map .

Bruna et al. [26] listed well-known graph clustering methods [56, 138]. One of the earliest applications for graph pooling in the GCN architecture can be found in [139, 140] . Defferrard et al. [29] specifically used the Graclus algorithm [56], which later found its way into other GNN-related publications [78, 79]. The majority of these approaches are based on voxelization techniques. The literature about learning points has introduced pooling strategies to generalize the standard pooling layers for grids [80, 141]. Numerous pooling operators have been developed based on graph spectral theory [80, 142] or other clustering, scarification, and decomposition methods [143-145], numerous pooling operators have been developed, as shown in Figure 4.8.

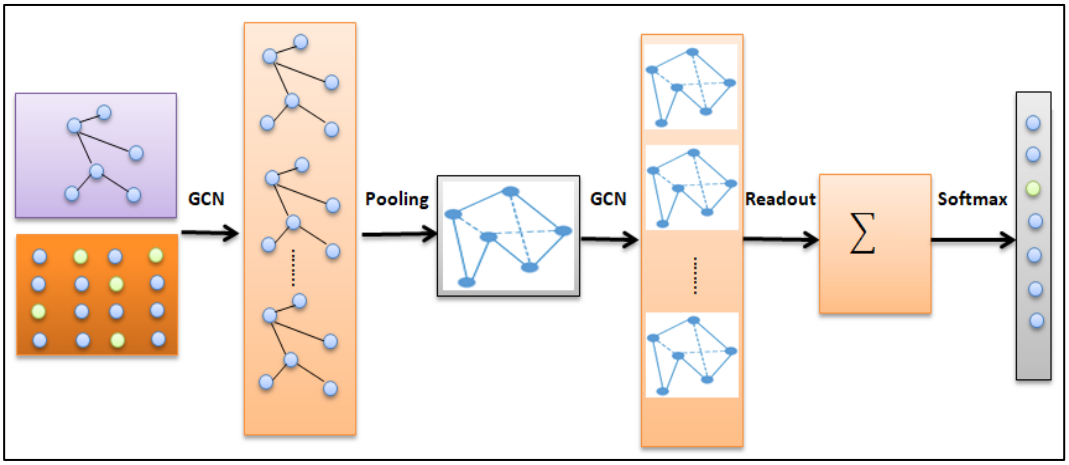

Figure 4. 8. Graph pooling layer.

4.2.6. Fully Connected Layer

The fully connected layer is the final block of the GCN architecture. The fully connected layer's primary function is to understand non-linear models for high-level data, as defined by the output from the final combination of convolutional pooling layers. To achieve this, we flatten the basic characteristics learned in the previous stages, which are appropriate for an ordinary neural network. A fully connected layer then connects the flattened characteristics, allowing concepts to be mapped between input and output. Once the GCN's building blocks are complete, the network can distinguish between various feature levels in an input image and use softmax classification to assign the image to a particular class.

CHAPTER 5

RESULTS AND COMPARATIVE ANALYSIS

This chapter introduces the results and discussion of the classification tasks using the proposed method we established, followed by a detailed examination of the dataset. It next assesses the suggested model's performance in terms of classification accuracy. We demonstrate the model's performance on a number of hyperparameters and compare our proposed method to multiple deep learning experiments on the datasets.

5.1. EXPERIMENTS AND RESULTS

This section provides an explanation of how we perform our proposed method to the dataset. It then introduces the six deep learning models used as baseline models, which we compare their performance with the proposed method.

5.1.1. Dataset

In this implementation, we used the SARS-COv2 dataset of CT scan images for our investigation. We utilized two distinct class categories and samples from the available dataset. We test the proposed methodology using a publicly available library of CT scan images of SARS-CoV-2 .This dataset is accessible via Kaggle and consists of 2482 slices in total, distributed as 1252 slices of SARS-CoV-2 (COVID-19)-infected patients and 1230 slices of healthy people [146]. Figure 5.1 displays some samples of CT scans for both COVID ,and non-COVID cases.

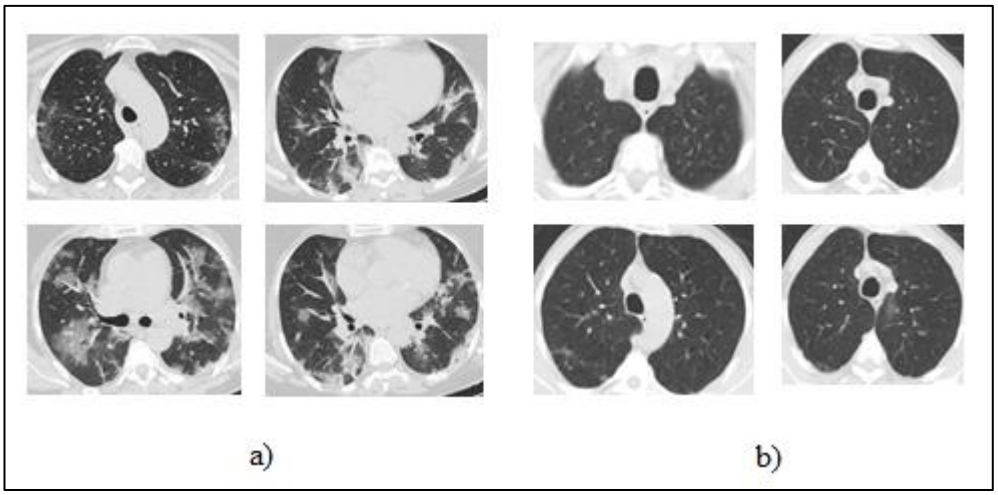

Figure 5. 1. a) SARS COVID ,and b) non-COVID samples.

5.1.2. Data Splitting

We divided the dataset into two sets: training ,and testing, which is an important step in developing machine learning models. This involves separating the dataset into two parts: training set, which is used to teach the model about the CT slices, and testing set, which is meant for assessing the model's performance on data ,and it hasn't encountered before. Typically, we allocate 80% of the dataset for training to provide sufficient learning opportunities for the model, while the remaining 20% is set aside for testing to measure the model's accuracy and ability to generalize. This split ensures that the performance metrics, like accuracy, accurately reflect the model's capability to predict outcomes on new, unseen data.

5.2. BASELINES

This chapter compares FGCN with six different baseline deep learning models: VGG16, DensNet201, efficientnetB0, Inception V3, Nasnet Mobile, and ResNet50. Being commonly used transfer learning models those are generally successful in medical classification tasks; researchers use these models as baseline architectures to compare newly proposed models.

5.2.1. VGG16

VGG16 is a deep learning model that introduced by Simonyan and Zisserman, comprises 16 layers [71]. It is widely recognized as a prominent model for transfer learning. It includes 13 convolutional layers and 3 fully connected layers. As illustrated in Figure 5.2, each convolutional layer utilizes 3x3 kernels, while each pooling layer uses 2x2 parameters. The architecture is organized into five blocks, each containing multiple convolutional layers followed by a pooling layer.

In Block 1, the process begins with two convolutional layers, each equipped with 16 filters, which captures the image features. Following this, a pooling layer decreases the image dimensions. This approach is maintained in the following blocks, with Blocks 1 ,and 2 each featuring two convolutional layers, while Blocks 3 through 5 include three convolutional layers. This arrangement gradually deepens the network and improves its precision. Finally, the network uses three fully connected layers to combine features into two distinct classes [147].

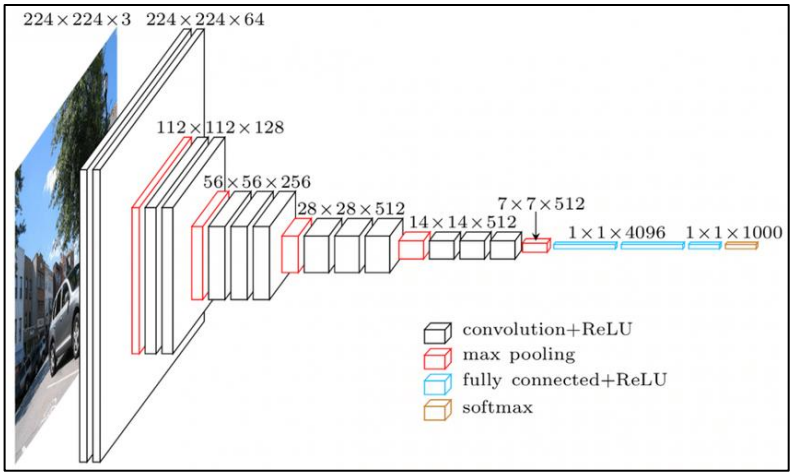

Figure 5. 2. VGG16 architecture [148].

5.2.2. DenseNet201

DenseNet201 is a specialized CNN model with 201 layers. We employed a pre-trained version of this model, which has been trained on more than a million of images from the ImageNet dataset. This pre-trained network has a feed-forward architecture that allows each layer to classify images into 1000 distinct categories, as

illustrated in Figure 5.3. This capability includes categorizing various objects and diagnosing diseases in medical imaging classification tasks. [117].

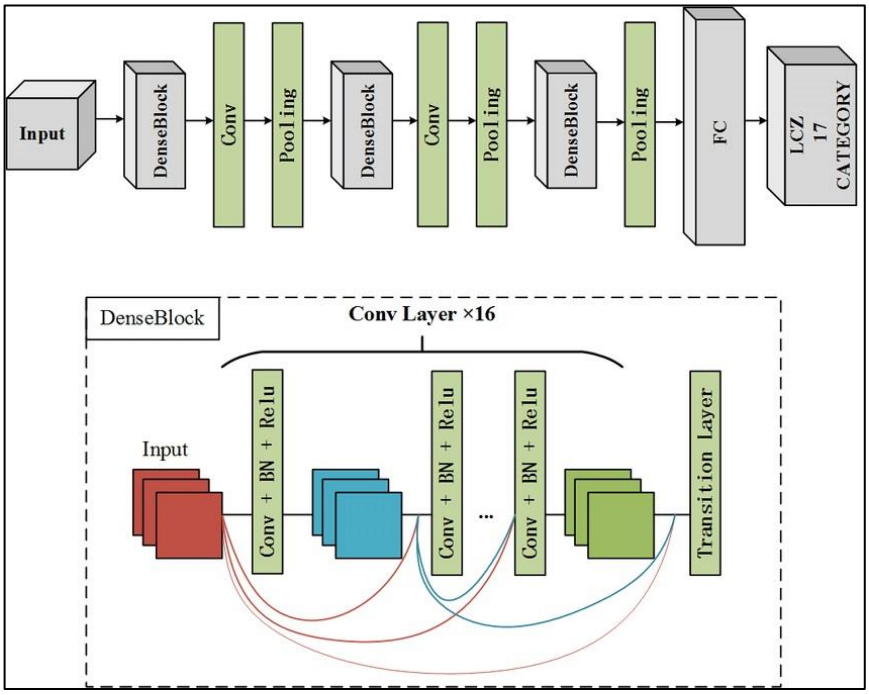

Figure 5. 3. DenseNet201 Architecture [149].

5.2.3. ResNet50

It is a convolutional neural network model that was designated to overcome the problems of the vanishing or expanding range. One of the most widely used CNN architectures recently is the ResNet structure. Microsoft Research first presented ResNet, an abbreviation for residual networks, in 2015. As shown in Figure 5.4, ResNet-50 introduces the concept of the residual neural network, which is a CNN with fifty layers. It learns residuals instead of features [150].

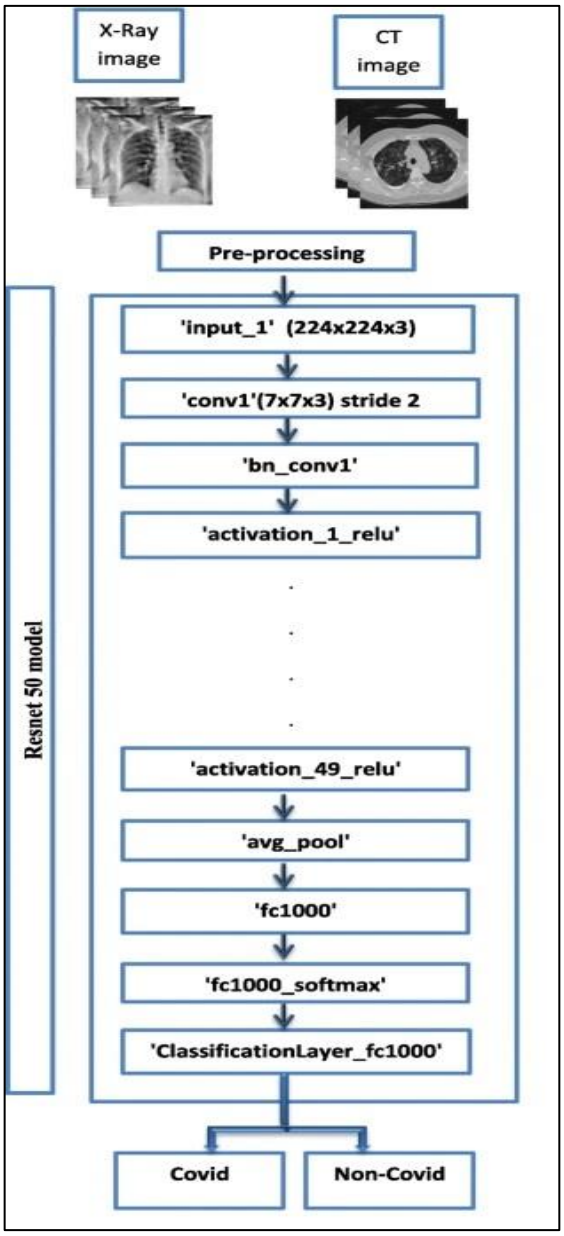

Figure 5. 4. ResNet50 Architecture for CT and X-ray COVID-19 classification [151].

5.2.4. NasNet Mobile

With its vast processing resources and brilliant engineering, Google proposed NasNet, which reframed the challenge of determining the ideal CNN structure as a reinforcement learning problem. In essence, the goal of this model is to find the optimal combination of each of the search space's parameters, as represented in Figure 5.5, which includes the sizes of filters, and the output is determined by the

number of channels, strides, and layers.. Every search process conducted in the context of reinforcement learning context rewarded the model's accuracy on the provided dataset. NasNet has achieved a state-of-the-art result in the ImageNet competition. However, a limited variety of classification tasks can employ the model due to its huge computation demand, which also optimizes the model architecture [152].

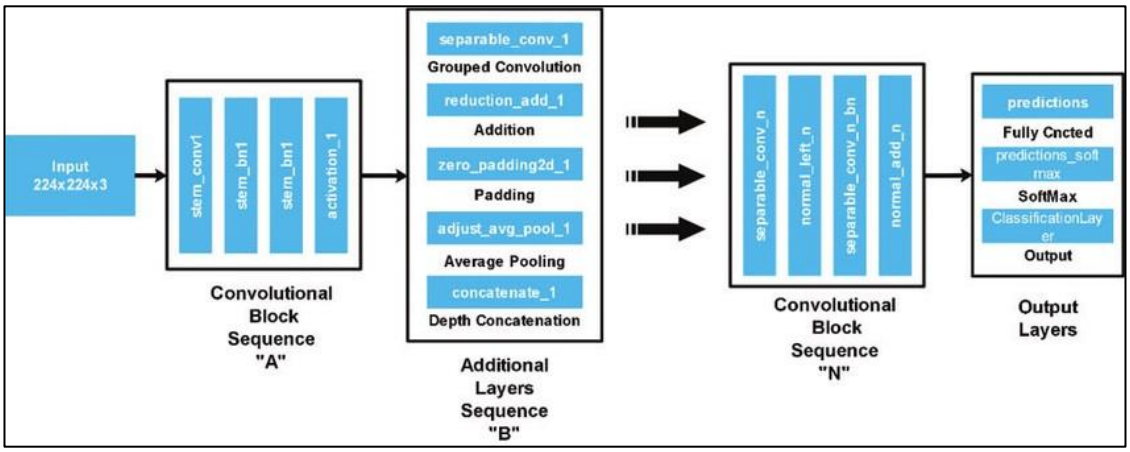

Figure 5. 5. NasNet Mobile architecture [153].

5.2.5. Inception V3

This model is a CNN model that featuring 48 of layers, specifically designed to recognize and analyze intricate patterns in medical images, as shown in Figure 5.6. Its key advantage lies in its capacity to manage large datasets and accommodate images of differing sizes and resolutions. Given the considerable variability in image size, quality, and resolution, this capability is essential in medical image processing. Typically, the model includes three convolutions of various types and incorporates a maximum of one pooling layer. [150].

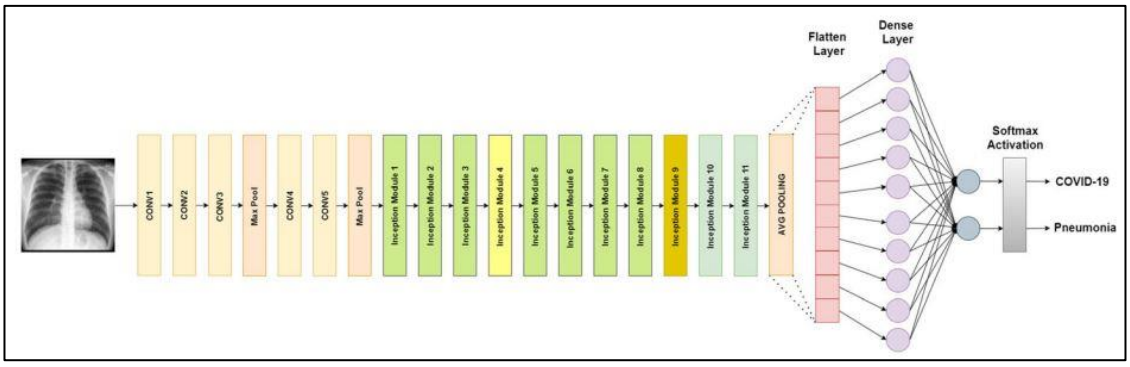

Figure 5. 6. Inception V3 model architecture for binary classifications [154].

5.2.6. EfficientNetb0

EfficientNet is a type of convolutional neural network (CNN) designed to optimize its parameters for depth, width, and resolution using a unified scaling method. Instead of adjusting these parameters randomly as in traditional approaches, EfficientNet applies a specific set of scaling factors to ensure balanced changes across all dimensions. This approach maintains consistency in scaling across depth, width, and resolution. The idea behind this mixed scaling is that larger input images need more channels to capture fine details and additional layers for expanding the network's receptive field. EfficientNetB0, one of the popular versions of this model, is depicted in Figure 5.7 [155].

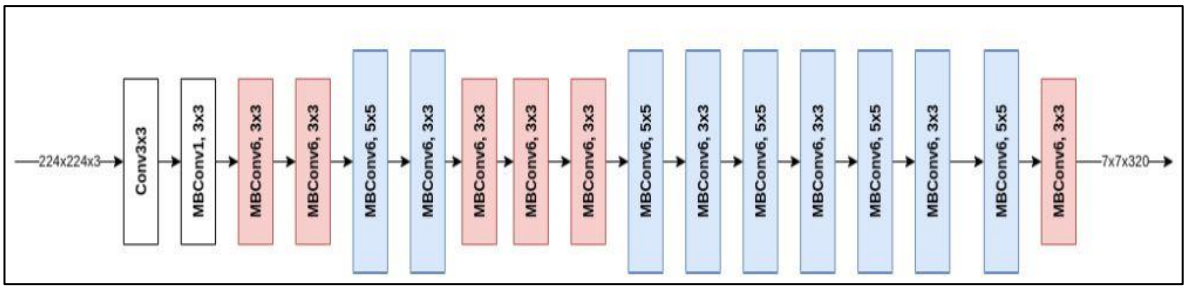

Figure 5. 7. Architecture of EfficientNetB0 [156].

5.3. CLASSIFICATION RESULTS

In this section, we provide the findings obtained from the conducted experiments performed on a variety of customized CNN models for medical image classification tasks, in combination with the proposed model, all tested on the same dataset detailed in the previous section.

We first used a split ratio of 80-20% to fix train and test data, as explained previously. Next, we randomly cropped the CT images from the original dataset to create fixed size images of size 512x512 pixels, that we input to the learning models. This could successfully minimize the complexity of computation while keeping the significance of the image data. To improve the dataset samples, we also applied some in-built image augmentation techniques including reversing the images in horizontal or vertical orientations, zooming in and out, rotating, skewing etc. The learning models were trained over 50 epochs with a learning rate of 0.0001 for each run.

We used Tensorflow and PyTorch libraries to implement the suggested FGCN model together with the transfer learning models, such as VGG16, Inception V3, Resnet50, efficientnetB0, Nasnet Mobile, and DensNet201. According to the results, the suggested model outperforms the other six DL models, as presented in Table 5.1.

A tabular method of displaying your prediction model's performance involves using a confusion matrix. Each value in a confusion matrix indicates how many predictions the model has made regarding whether the classes were correctly or incorrectly categorized. A binary classification problem clearly entails only two classification classes, ideally a positive (P) and a negative (N) class.

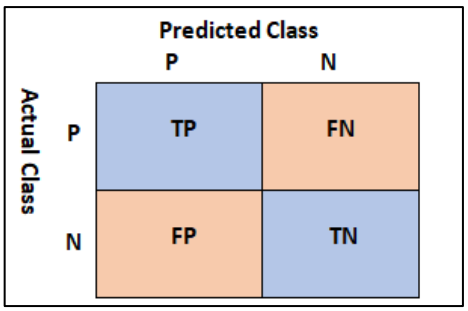

Figure 5. 8. Confusion matrix for binary class classification.

A confusion matrix given in Fig. 5.8, consists of TP, TN, FP and FN values, detailed as follows:

True Positive (TP): The number of occurrences accurately classified as positive, meaning that both the actual class and the model's prediction were positive.

True Negative (TN): This is the count of accurately predicted negative instances (i.e., the actual class was negative and the model predicted negative).

False Positive (FP): The quantity of false positive cases refers to situations where the model predicted a positive class while the actual class was negative. This phenomenon is commonly known as a "Type I error."

False Negative (FN): The model erroneously predicted instances as false negatives (FN) while their actual class was positive. It is also commonly known as a "Type II error."

The confusion matrix provides the necessary information to compute classification metrics like F1-score, precision, recall, and specificity for the predictions made on the test data. In multi-class or binary-class classification problems, we assume that TP, TN, FP, and FN are one-versus-all classes. Accordingly, the class of concern itself is the positive class, and all other classes are the negative class with regard to the specific category of concern. In this context, we define accuracy as a model's general predictive accuracy, determined by dividing the number of correctly predicted samples by the total number of predictions presented in the next Equation 5.1, [157].

Accuracy: is the ratio of correctly identified instances (including true positives and true negatives) to all instances.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

Precision: The proportion of true positive predictions out of all positive predictions made by the model.

$$\text{Precision } = \frac{TP}{TP + FP} \tag{5.2}$$

Recall, also known as sensitivity or true positive rate, refers to the proportion of real positive cases that were accurately predicted.

$$Recall = \frac{TP}{TP + FN} \tag{5.3}$$

F1-Score: The harmonic mean of precision and recall offers a balanced measure between the two

$$F1 - Score = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{5.4}$$

Specificity, often known as the true negative rate, refers to the accuracy of correctly predicting genuine negative outcomes.

$$Specificity = \frac{TN}{TN + FP} \tag{5.5}$$

In our tests through six deep learning models (VGG16, DensNet201, ResNet50, NasNet Mobile, InceptionV3, and EfficientnetB0) and the FGCN model proposed, we achieved confusion matrices as given in Figure 5.9 below.

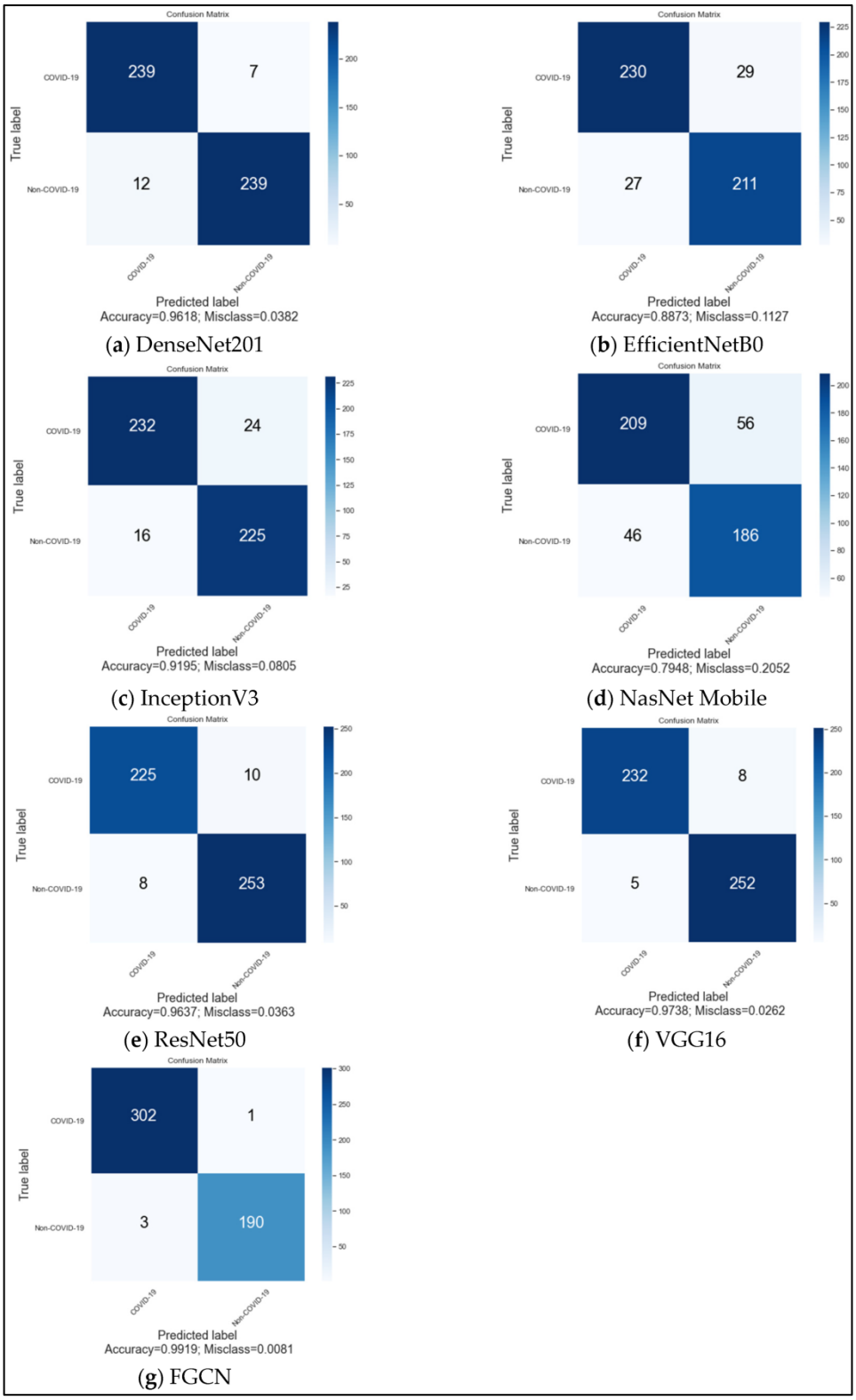

Figure 5. 9. Confusion matrix for six DL models with proposed method.

Table 5.1 displays the classification outcomes for all models. According to the data, the proposed model (FGCN) exhibits the accuracy of classification is 99.19%,

exceeding the performance of the other models evaluated. The models with the next highest accuracies are ResNet50, VGG16, and DenseNet201, with respective accuracies of 96.37%, 97.38%, and 96.18%. With accuracy values below 92%, Inception V3 and efficientnetB0 could not compete with the others. The model with the lowest capability is the NASNet Mobile, with scores of about 80%. The proposed model also boasts excellent recall, precision, specificity, and F-score metrics exceeding 99%.

We also compared the computational times devoted for training for all models together with the proposed method FGCN to determine the variations in running times. The running times are calculated for the optimal setup that would result the best results given in Table 5.1. First, the proposed model FGCN has a running time of 260.24 sec/epoch for the training phase. EfficientNet-B0 outperforms FGCN with 215.68 sec/epoch by the means of training speed, while NASNet Mobile is significantly faster than FGCN with the speed 132.78 sec/epoch, emerging as the fastest model in our experiments. InceptionV3 follows NASNet model with 145 sec/epoch, while ResNet50 approximates FGCN with a running time of 265 sec/epoch. DenseNet201 and VGG16 were the slowest models with 387.88 and 480.38 sec/epoch respectively. In summary, the comparison for computational time depends on the architectural design of the models. As a result, FGCN can compete with DenseNet201, VGG16 and ResNet in terms of computational complexity, while it also has comparable performance with EfficientNet-B0. NASNet Mobile and InceptionV3 show exceptional running times, while these outcomes in computational complexity do not reflect classification accuracy results as presented in Table 5.1.

Table 5. 1. Classification outcomes for the SARS-CoV-2 CT dataset were obtained using various pre-trained deep learning models.

DL models	Recall	Precision	Specificity	F-Score	Accuracy
DensNet201	0.9715	0.9522	0.9522	0.9618	0.9618
EfficientnetB0	0.8880	0.8949	0.8866	0.8914	0.8873
Inception V3	0.9062	0.9355	0.9336	0.9206	0.9195
Nasnet Mobile	0.7887	0.8196	0.8017	0.8039	0.7948
ResNet50	0.9574	0.9657	0.9693	0.9615	0.9637
VGG16	0.9667	0.9789	0.9805	0.9728	0.9738
FGCN	0.9967	0.9905	0.9845	0.9936	0.9919

Learning curves demonstrate how a model's accuracy improves through a given number of epochs, for both train and validation data. This type of visualization is valuable since it shows if an overfitting occurs during training, diagnosed by the divergence between the train and the validation curves. Fig. 5.10 depicts the learning curves for all models tested. These plots indicate that overfitting is evident for NasNet and VGG16 models, where the rest of the models have good consistency between train and validation curves. This behavior indicates that the generalization capacity of the models is in an adequate level.

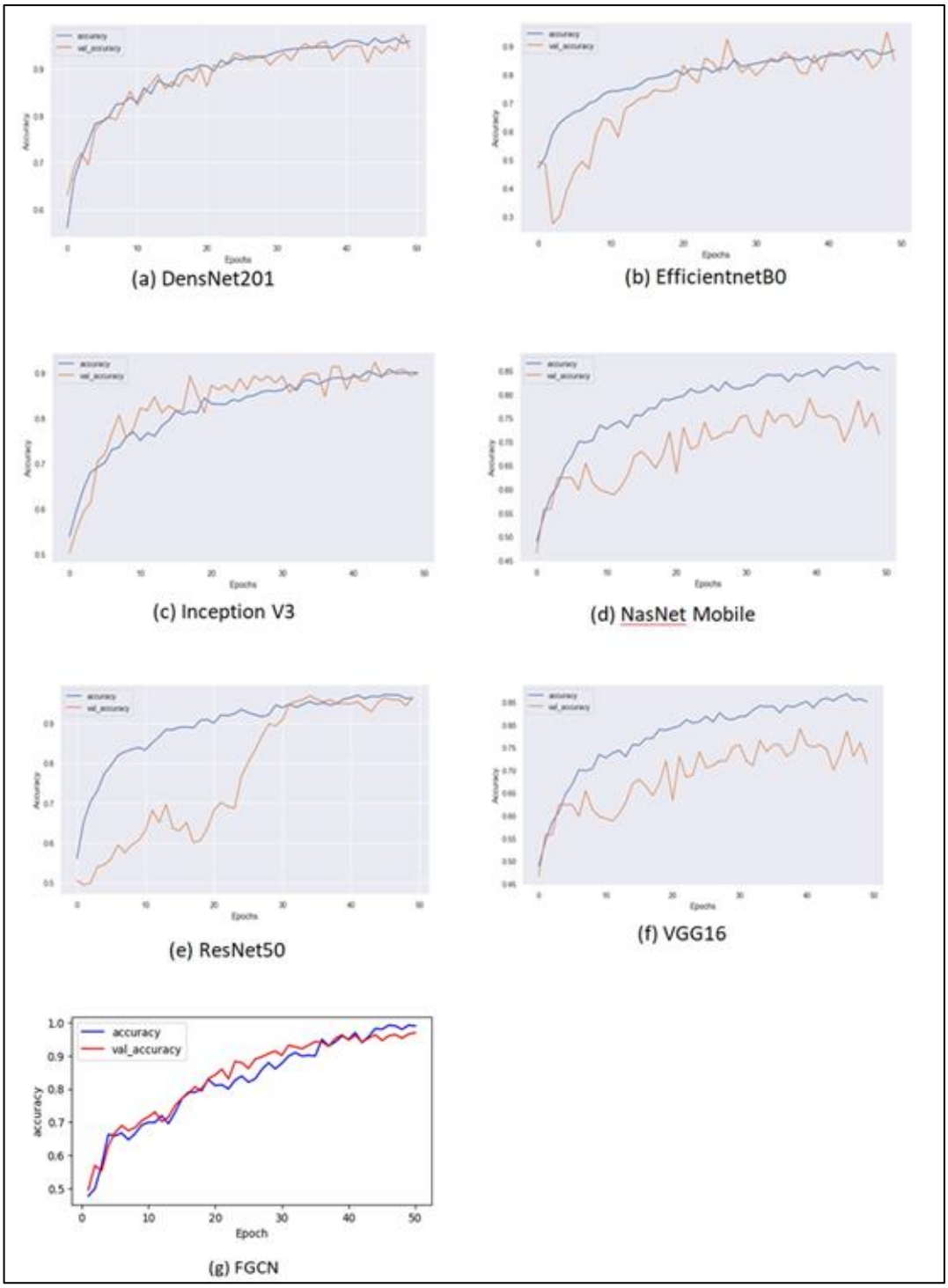

Figure 5. 10. Train and validation learning curves for FGCN with six DL models implemented for 50 epochs.

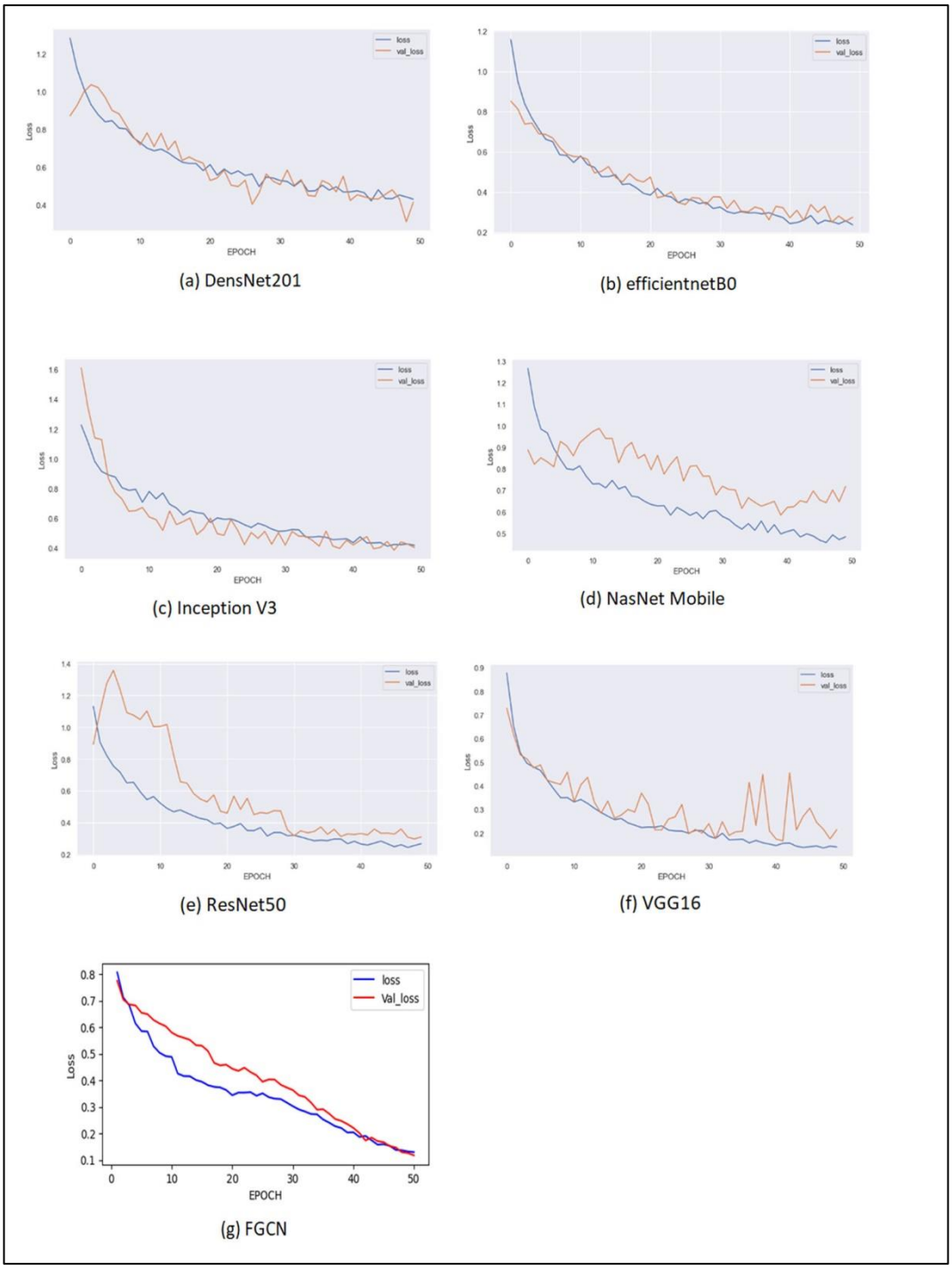

Figure 5. 11. Train and validation loss plot for FGCN implemented for 50 epochs.

Figure 5.11 demonstrates the loss curves across the tested models. Loss values reach very low values for all models, where the two models mentioned above show overfitting behavior due to divergence in their train and testing curves. The FGCN

model proposed has desirable learning and loss curves, indicating a good generalization capacity.

5.3.1. Implementation and Tools

This section covers the tools we've used to implement FGCN. The research by Ronnberger et al. [121] and Kapf & Welling [28] provided the code for the FGCN implementation. Kapf and Welling's code served as the foundation design for the GCN implementation. The majority of the novel contributions came from implementing the UNet model as a feature extractor and adjusting the hyperparameters.

5.3.2. Testing Environment

i. Operating system: Windows 10 Enterprise Edition, version 10, 64-bit
ii. CPU: Intel Core i7-12700HQ with 2.8 GHz
iii. RAM: 64 GB
iv. GPU: RTX 3080 8GB

5.3.3. Development Tools

i. Languages used: Python (3.7)
ii. Jupyter notebook
iii. Development tools: PyTorch (1.2), Tensorflow , Keras, NetworkX, Matplotlib, Numpy

5.4. COMPARISON WITH RECENT METHODS

Table 5.2 shows a comparison of our model's performance metrics against the latest state-of-the-art techniques. When contrasted with earlier studies, the FGCN model performs similarly, featuring a unique combination of the U-Net architecture with a GCN. Additionally, it highlights the success of using a graph-based method for identifying and diagnosing complex patterns related to lung diseases.

Table 5. 2. Comparative analysis of performance metrics between the proposed model (FGCN) and existing approaches on datasets of COVID-19 CT scan images.

Ref.no	Dataset	Model	Acuracy	Recall	Precision	Specificity	F1-Score	AUC
[158]	CT scan (4382 COVID, 9,369 non-COVID)	U-Net++	95.2	100	-	93.6	-	-
[159]	Total of 1918 CT scan	DCN	95.99	89.14	-	98.04	-	97.55
[160]	CT scan (449 COVID, and 595 non-COVID)	Two U-Nets	86	94	-	79	-	93
[161]	COVID-19 CT scan	U-Net+	82.9	97.4	-	92.2	-	-
[162]	CT scan (521 COVID, and 665 non-COVID)	EfficientNet-B4	87	89	-	87	-	90
[163]	CT scan (1262 COVID, 1230 non-COVID)	DenseNet-201	96.25	96.29	96.29	96.21	96.29	-
[164]	CT scan (108 COVID, 912 non-COVID)	Xception	99.02	96.29	96.29	96.21	96.29	-
[165]	CT scan (468 COVID, 2996 non-COVID)	ResNet-50	89.5	87	-	92	-	95
[163]	CT scan (723 COVID, 413 non-COVID)	U-Net and ResNet-50	94.8	97.4	-	92.2	-	-
[166]	CT scan (360 COVID, 34 non-COVID)	VGG16	91	94	100	-	97	-
[167]	CT scan (219 COVID, 399 non-COVID)	ResNet-50	86	96	79	-	-	95.96
[168]	Consists of 1065 CT	CNN	83	84	-	80.5	-	-
[169]	CT scan (1493 COVID, 4594 non-COVID)	Inception-ResNetV2	92.18	92.11	92.38	96.06	-	-
[170]	CT scan (260 COVID, 600 non-COVID)	VGG19	89.3	89	90	-	90	-
[171]	CT scans (1252 COVID-19 and 1230 non-COVID-19)	DarkNet19 with repeated holdout 10FCV	95.2	98.2	-	92.2	-	99.6
[172]	CT scan (313 COVID, 229 non-COVID)	U-Net and CNN	98.91	98.96	-	98.86	90	-
[173]	CT scan (192 COVID, 145 non-COVID)	nCOVnet	97.62	97.62	-	78.57	-	-

[174]	CT scan (347 COVID, 397 non-COVID)	ResNet-18	78.29	76.9	81	79.9	78.9	83.82
Current study	**CT scan (1252 COVID, and 1230 non-COVID)**	**FGCN**	**99.19**	**99.67**	**99.05**	**98.45**	**99.36**	**-**

Beyond the superior classification accuracy achieved across the recent studies, reviewing the fundamental differences of these studies would be beneficial to understand the distinction of the current study and the cutting-edge. The DL-based algorithm for high-resolution CT scan image-based detection of new coronavirus relies on U-Net ++ [158], while the Dual-Branch Combining Network (DCN) accurately segments and identifies COVID-19 tumors using CT scans [159]. The internal dataset achieved a classification accuracy of 95.99%, while the validated external dataset achieved a classification accuracy of 92.87% , the suggested DCN did better than other models. DCN was more sensitive and got the same results with fewer samples, especially when it came to finding small lesions. Compared to previously developed deep learning models, it offers best interpretability on infection origins because its categorization is based on significant semantic data.

Amyar et al. focus on the goal, which is to categorize and divide multitasking deep learning-based CT scans for COVID-19 pneumonia. The researchers assess and contrast their proposed model with alternative picture segmentation techniques using a dataset of 1,369 patients. This dataset includes 449 patients diagnosed with COVID-19, 425 patients with normal cases, 98 patients with lung cancer, and 397 patients with diverse other illnesses. The outcomes are extremely encouraging, with a dice coefficient for segmentation above 0.88 and an area under the ROC curve for classification exceeding 97% [160] .

Jin, S., et al. detailed their experience in developing and deploying an artificial intelligence system to autonomously analyze CT scans for identifying COVID-19 pneumonia characteristics. Unlike traditional medical AI projects, this effort was specifically to address an epidemic crisis. The team, consisting of over 30 experts in medicine and artificial intelligence from Beijing and Wuhan, collaborated to

overcome various challenges specific to this case. Remarkably, they completed the system within four weeks. With a dataset of 1,136 training cases, which consisted of 723 positive COVID-19 cases from five hospitals, the researchers achieved a sensitivity of 0.974 and a specificity of 0.922 on the test dataset. The test dataset contained various pulmonary ailments.The system also highlighted abnormal areas for quick examination. Until 2020, 16 hospitals have adopted this technology, performing over 1,300 screenings daily [161].

Radiologists use AI to improve their ability to differentiate between COVID-19 and pneumonia of different origins during chest CT scans. After using the EfficientNet-B4 model to divide the lungs into sections, they fed strange CT images into an EfficientNet-B4 DNN structure to distinguishing COVID-19 from other forms of pneumonia in each patient. They then pooled all the slices together using a two-layer, fully interconnected neural network. They used a 7:2:1 and proportional arrangement to divide the 1186 individuals (consisting of 132,583 CT slices) were divided into training, validation, and test sets. Two different hospitals conducted separate evaluations to assess the efficacy of the model. Six radiologists blindly reviewed studies, first without AI support and later with it [162].

Authors classified COVID-19-affected individuals using deep transfer learning, DenseNet201. They suggested a model that would use a convolutional neural network and its own discovered weights from the ImageNet dataset to extract attributes. Extensive testing assesses the effectiveness of the proposed DTL model on COVID-19 chest scan images. The suggested DTL-based COVID-19 classification framework performs better than the competing methods, based on comparisons [163].

In Ref. [164], the authors employed ten prominent convolutional neural networks to the models used to differentiate between COVID-19 infections and non-COVID-19 cases are AlexNet, VGG-19, VGG-16, GoogleNet, SqueezeNet, MobileNet-V2, ResNet-50, ResNet-18, ResNet-101, and Xception. Out of all the options, ResNet-101 and Xception demonstrated the most impressive performance. ResNet-101, in

particular, demonstrated a strong ability to differentiate between COVID-19 and non-COVID-19 instancesm[164].

In Ref. [165], authors developed a neural network model called COVNet to analyse dimensional chest CT scans and detect COVID-19 by extracting visual features. They evaluated the model's strength using CAP (community-acquired pneumonia) and other non-pneumonia CT examinations [165].

In Ref. [166], authors employ an algorithm that relies on DL, computed tomography (CT) and X-rays for the early identification of COVID-19 patients. We use CNN to look at the 360-image set of X-rays and CT scans and turn VGG-19, Inception_V2, and the decision tree method into two groups of pneumonia. The refined form VGG-19, Inception_V2, and the decision tree model are all very good at what they do. For example, the refined form VGG-19 has an average training and validation accuracy of 91%, while Inception_V2 has an average of 78% and the decision tree has an average of 60% [166].

The ground-glass opacity (GGO), in particular, is a visibly useful feature for physician-assisted diagnosis that the model could extract from the disease [167]. The study used a DL method to screen for coronavirus disease using CT scans, analyzing 1065 CT scans of COVID-19 samples that confirmed the presence of an infectious agent, additionally; this includes individuals who have previously been diagnosed with typical viral pneumonia. After creating the algorithm and making changes to the inception transfer-learning approach, they conducted internal and external validation. [168].

Utilizing X-ray images for deep learning enables rapid detection of COVID-19 illnesses. When evaluating several deep learning models, Inception_Resnet_V2 and Densnet201 outperform all other models examined in this study, with an accuracy of 92.18% and 88.09%, respectively [169].

To determine which pre-trained CNN model was best for recognizing COVID-19 cases, authors tested 15 of these models in Ref. [170]. In another study, authors

developed a deep learning system to identify pneumonia due to the coronavirus disease [175]. They use weak labels in DL for COVID-19 recognition from chest CT scan images. Without the need to annotate the lesions for training, the weakly supervised DL model can reliably predict the COVID-19 infectious probability in chest CT volumes. The rapid identification of COVID-19 patients made possible by the highly effective and easily trainable DL algorithm is helpful in containing the SARS-CoV-2 epidemic.

In Ref. [172], The authors created a software solution that utilizes weakly supervised deep learning to identify COVID-19 by analyzing 3D CT scans. The researchers utilized a pre-trained UNet model to accurately outline the lung area of each patient. The researchers then fed this information into a 3D deep neural network to evaluate the probability of COVID-19 infection. For training, they collected 499 CT volumes from December 13, 2019, to January 23, 2020, and used 131 CT volumes from January 24, 2020, to February 6, 2020, for testing. The deep learning method achieved a ROC AUC of 0.959 and a PR AUC of 0.976. The ROC curve demonstrated an operating point with a sensitivity of 0.907 and a specificity of 0.911 .The method attained an accuracy of 0.901, a positive predictive value of 0.840, and an extraordinarily high negative predictive value of 0.982 by using a probability threshold of 0.5 to distinguish between COVID-positive and COVID-negative cases. The technique efficiently analysed each patient's CT volume in a little 1.93 seconds using a dedicated GPU. Their unsupervised a deep learning model has the ability to accurately estimate the probability of COVID19 infection in chest CT data without requiring lesion annotation for training [172].

In Ref. [173], The authors introduce nCOVnet, a deep learning neural network designed for the rapid screening of COVID-19. This innovative approach analyzes patients' X-ray images to identify visual markers indicative of COVID19 in chest X-rays [173].

Multi-deep is an innovative CAD system that uses several CNNs to identify the coronavirus (COVID-19) from chest CT scans. The system combines and examines a

predetermined quantity of key elements, leveraging deep characteristics taken out of every CNN [174].

In conclusion, we present our suggested technique for identifying lung diseases, which utilizes a graph convolutional network (GCN) is used after the U-Net feature extractor .
The initial graph image and graph kernel, along with characteristics that have been retrieved are put into the GCN in the form of an adjacency matrix with a graph structure. After combining these three graphs into one block of graph input, the GCN processes them with an extra dropout layer to prevent overfitting. The output is then labeled as COVID-positive or negative.

Our proposed framework, termed Feature-Extracted Graph Convolutional Networks (FGCN), outperformed other non-graph-induced deep learning architectures in detecting and classifying lung diseases. As illustrated in Table 5.2, our method achieved the highest performance metrics compared to other approaches. However, we also highlight that GCNs face significant challenges in the classification of CT scan images in deep learning.

CHAPTER 6

DISCUSSION AND CONCLUSION

6.1. DISCUSSION

Graph Convolutional Networks (GCNs) have emerged as highly effective tools in diverse domains, owing to their adeptness in modeling and processing relational structures inherent in graph data. By extending convolutional operations to graphs, GCNs harness the intricate connectivity patterns present in actual datasets, such as social networks, molecular structures, and even pixel graphs derived from images. This capability empowers GCNs to excel in important tasks include graph categorisation, link prediction, and node classification, where understanding contextual relationships among data entities is pivotal.

In our research, we present FGCN, an innovative DL architecture specially designed for detecting and classifying CT scans with COVID-19. Integrating UNet with GCN enhances FGCN's ability to extract features crucial for both segmentation and classification tasks. We benchmarked FGCN against several leading models—VGG16, ResNet50, DenseNet201, Inception V3, Mobile NASNet, and EfficientNetB0—using the SARS COV2 CT scan dataset comprising 2,482 images. FGCN achieved outstanding results, notably a classification accuracy of 99.19%, surpassing all comparison models. ResNet50, VGG16, and DenseNet201 achieved accuracies 96.37%, of 97.38%, and 96.18%, respectively, while Inception V3 and EfficientNetB0 lagged behind, scoring below 92%. Mobile NASNet performed least effectively with scores around 80%. FGCN also demonstrated superior performance in recall, precision, specificity, and F-score metrics, all exceeding 99%.

CT imaging plays a crucial role in early COVID-19 detection and patient management. Leveraging deep learning techniques with CT scans provides physicians with robust tools for disease categorization, segmentation, and treatment expected in the context of this rapidly evolving pandemic.

6.2. CONCLUSION

This thesis investigates the effects of utilizing the UNet model and Graph Convolutional Network (GCN) technique to extract features from chest CT scan images, aiming to enhance image classification and provide significant advantages for healthcare professionals. Throughout the study, we explored various deep learning techniques and their implementations, focusing particularly on graph convolutional networks (GCNs) combined with UNet.

Key Findings and Contributions:

i. **Novel Deep Learning Models**: The primary goal was to develop innovative deep learning models for medical image processing, with an emphasis on CT scan images used for diagnosis and grading. Two main methods were devised:

   i. **Graph Convolutional Networks (GCNs)**: These were leveraged to enhance the performance of traditional CNNs by capturing spatial connectivity patterns in the images.

   ii. **UNet Technique**: This was used to categorize the SARS-CoV-2 CT scan dataset based on the model's predictions of the number of infections, enhancing confidence in the diagnostic process.

ii. **Performance Enhancement**: The proposed methods aimed to accelerate deep learning algorithms and improve the accuracy of image classification. Hyperparameters were optimized using both single- and multi-objective function optimization techniques to support these improvements.

iii. **Comparative Analysis**: The proposed Feature-Extracted Graph Convolutional Network (FGCN) was compared against six other deep learning models (VGG16, DenseNet201, ResNet50, EfficientNetB0, Inception V3, and NasNet Mobile). The assessment criteria included precision, accuracy, specificity, recall, and F1-score, based on confusion matrices.

iv. **Superior Results**: The FGCN demonstrated superior performance in classifying SARS-CoV-2 CT scan images into infected to COVID and non infected to COVID categories. This indicates the effectiveness of combining GCNs with UNet for diagnostic image classification tasks.

In conclusion, the suggested FGCN structure effectively enhances the classification accuracy of thoracic CT scan images to detect COVID-19. This approach not only outperforms several state-of-the-art deep learning models but also offers a robust methodology for handling the spatial connectivity inherent in medical images. The success of this model underscores the potential of integrating GCNs with traditional CNN techniques to advance medical image processing and diagnostic accuracy. Future research should continue to refine these methods, focusing on scalability, interpretability, and broader applications across different types of medical images and diseases.

REFERENCES

1. Yao, L., C. Mao, and Y. Luo. *Graph convolutional networks for text classification*. in *Proceedings of the AAAI conference on artificial intelligence*. 2019.

2. Qiu, J., et al., *A survey of machine learning for big data processing.* EURASIP Journal on Advances in Signal Processing, 2016. 2016: p. 1-16.

3. Karhunen, J., T. Raiko, and K. Cho, *Unsupervised deep learning: A short review*. Advances in independent component analysis and learning machines, 2015: p. 125-142.

4. Zhang, S., et al., *Graph convolutional networks: a comprehensive review.* Computational Social Networks, 2019. 6(1): p. 1-23.

5. Vieira, S., et al., *Using machine learning and structural neuroimaging to detect first episode psychosis: reconsidering the evidence.* Schizophrenia bulletin, 2020. 46(1): p. 17-26.

6. Dyer, S.C., B.J. Bartholmai, and C.W. Koo, *Implications of the updated Lung CT Screening Reporting and Data System (Lung-RADS version 1.1) for lung cancer screening.* Journal of Thoracic Disease, 2020. 12(11): p. 6966.

7. Kieu, S.T.H., et al., *COVID-19 detection using integration of deep learning classifiers and contrast-enhanced canny edge detected X-ray images.* It Professional, 2021. 23(4): p. 51-56.

8. Fu, S., X. Yang, and W. Liu. *The comparison of different graph convolutional neural networks for image recognition*. in *Proceedings of the 10th International Conference on Internet Multimedia Computing and Service*. 2018.

9. Li, Q., Z. Han, and X.-M. Wu. *Deeper insights into graph convolutional networks for semi-supervised learning*. in *Proceedings of the AAAI conference on artificial intelligence*. 2018.

10. Zhang, M., et al. *An end-to-end deep learning architecture for graph classification*. in *Proceedings of the AAAI conference on artificial intelligence*. 2018.

11. Gross, J.L. and J. Yellen, *Handbook of graph theory*. 2003: CRC press.

12. Scarselli, F., et al., *The graph neural network model.* IEEE transactions on neural networks, 2008. 20(1): p. 61-80.

13. Liben-Nowell, D. and J. Kleinberg. *The link prediction problem for social networks*. in *Proceedings of the twelfth international conference on Information and knowledge management*. 2003.

14. Langley, P., *Elements of machine learning*. 1996: Morgan Kaufmann.

15. Türker, İ., S.O. Tan, and G. Kutluana, *Tool wear prediction by deep learning from augmentable visibility graph representation of time series data.* Acta Technica Napocensis-Series: Applied Mathematics, Mechanics, and Engineering, 2023. 66(5).

16. Grover, A. and J. Leskovec. *node2vec: Scalable feature learning for networks*. in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016.

17. Hjelmås, E. and B.K. Low, *Face detection: A survey.* Computer vision and image understanding, 2001. 83(3): p. 236-274.

18. Dai, A.M., C. Olah, and Q.V. Le, *Document embedding with paragraph vectors.* arXiv preprint arXiv:1507.07998, 2015.

19. Le, Q. and T. Mikolov. *Distributed representations of sentences and documents*. in *International conference on machine learning*. 2014. PMLR.

20. Devlin, J., et al., *Bert: Pre-training of deep bidirectional transformers for language understanding.* arXiv preprint arXiv:1810.04805, 2018.

21. Jackson, P. and I. Moulinier, *Natural language processing for online applications.* 2007.

22. Wang, H., et al. *Graphgan: Graph representation learning with generative adversarial nets*. in *Proceedings of the AAAI conference on artificial intelligence*. 2018.

23. Krizhevsky, A., I. Sutskever, and G.E. Hinton, *Imagenet classification with deep convolutional neural networks.* Advances in neural information processing systems, 2012. 25.

24. Schuster, M. and K.K. Paliwal, *Bidirectional recurrent neural networks.* IEEE transactions on Signal Processing, 1997. 45(11): p. 2673-2681.

25. Vincent, P., et al., *Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion.* Journal of machine learning research, 2010. 11(12).

26. Bruna, J., et al., *Spectral networks and locally connected networks on graphs.* arXiv preprint arXiv:1312.6203, 2013.

27. Wu, Z., et al., *A comprehensive survey on graph neural networks.* IEEE transactions on neural networks and learning systems, 2020. 32(1): p. 4-24.

28. Kipf, T.N. and M. Welling, *Semi-supervised classification with graph convolutional networks.* arXiv preprint arXiv:1609.02907, 2016.

29. Defferrard, M., X. Bresson, and P. Vandergheynst, *Convolutional neural networks on graphs with fast localized spectral filtering.* Advances in neural information processing systems, 2016. 29.

30. Hamilton, W., Z. Ying, and J. Leskovec, *Inductive representation learning on large graphs.* Advances in neural information processing systems, 2017. 30.

31. McCulloch, W.S. and W. Pitts, *A logical calculus of the ideas immanent in nervous activity.* The bulletin of mathematical biophysics, 1943. 5: p. 115-133.

32. Rosenblatt, F., *The perceptron: a probabilistic model for information storage and organization in the brain.* Psychological review, 1958. 65(6): p. 386.

33. Rosenblatt, F., *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. 1961, Cornell Aeronautical Lab Inc Buffalo NY.

34. Waibel, A., et al., *Phoneme recognition using time-delay neural networks,* in *Backpropagation*. 2013, Psychology Press. p. 35-61.

35. Zhang, W., et al. *Shift-invariant pattern recognition neural network and its optical architecture*. in *Proceedings of annual conference of the Japan Society of Applied Physics*. 1988. Montreal, CA.

36. LeCun, Y., et al., *Backpropagation applied to handwritten zip code recognition.* Neural computation, 1989. 1(4): p. 541-551.

37. Ajmal, H., et al., *Convolutional neural network based image segmentation: a review.* Pattern Recognition and Tracking XXIX, 2018. 10649: p. 191-203.

38. Baskin, C., et al., *Streaming Architecture for Large-Scale Quantized Neural Networks on an FPGA-Based Dataflow Platform.* 2017.

39. Bronstein, M.M., et al., *Geometric deep learning: going beyond euclidean data.* IEEE Signal Processing Magazine, 2017. 34(4): p. 18-42.

40. Türker, İ. and P. Rashid, *SARS COV-2 CT-SCAN IMAGE CLASSIFICATION USING GRAPH CONVOLUTIONAL NETWORKS*. 2022.

41. Veličković, P., et al., *Graph attention networks.* arXiv preprint arXiv:1710.10903, 2017.

42. Lee, J.B., R. Rossi, and X. Kong. *Graph classification using structural attention*. in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2018.

43. You, J., et al. *Graphrnn: Generating realistic graphs with deep auto-regressive models*. in *International conference on machine learning*. 2018. PMLR.

44. Kipf, T.N. and M. Welling, *Variational graph auto-encoders.* arXiv preprint arXiv:1611.07308, 2016.

45. Li, Y., et al., *Gated graph sequence neural networks.* arXiv preprint arXiv:1511.05493, 2015.

46. Tai, K.S., R. Socher, and C.D. Manning, *Improved semantic representations from tree-structured long short-term memory networks.* arXiv preprint arXiv:1503.00075, 2015.

47. Zhou, J., et al., *Graph neural networks: A review of methods and applications.* AI open, 2020. 1: p. 57-81.

48. Hong, D., et al., *Learnable manifold alignment (LeMA): A semi-supervised cross-modality learning framework for land cover and land use classification.* ISPRS journal of photogrammetry and remote sensing, 2019. 147: p. 193-205.

49. Hong, D., et al., *CoSpace: Common subspace learning from hyperspectral-multispectral correspondences.* IEEE Transactions on Geoscience and Remote Sensing, 2019. 57(7): p. 4349-4359.

50. Chung, F.R., *Spectral graph theory*. Vol. 92. 1997: American Mathematical Soc.

51. McGillem, C.D. and G.R. Cooper, *Continuous and discrete signal and system analysis.* (No Title), 1991.

52. Hong, D., et al., *Graph convolutional networks for hyperspectral image classification.* IEEE Transactions on Geoscience and Remote Sensing, 2020. 59(7): p. 5966-5978.

53. Hammond, D.K., P. Vandergheynst, and R. Gribonval, *Wavelets on graphs via spectral graph theory.* Applied and Computational Harmonic Analysis, 2011. 30(2): p. 129-150.

54. Kingma, D.P. and J. Ba, *Adam: A method for stochastic optimization.* arXiv preprint arXiv:1412.6980, 2014.

55. Hinton, G., N. Srivastava, and K. Swersky, *Lecture 6a overview of mini–batch gradient descent.* Coursera Lecture slides https://class. coursera. org/neuralnets-2012-001/lecture,[Online, 2012.

56. Dhillon, I.S., Y. Guan, and B. Kulis, *Weighted graph cuts without eigenvectors a multilevel approach.* IEEE transactions on pattern analysis and machine intelligence, 2007. 29(11): p. 1944-1957.

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

58. Shuman, D.I., et al., *The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains.* IEEE signal processing magazine, 2013. 30(3): p. 83-98.

59. Shervashidze, N., et al., *Weisfeiler-lehman graph kernels.* Journal of Machine Learning Research, 2011. 12(9).

60. Chen, J., T. Ma, and C. Xiao, *Fastgcn: fast learning with graph convolutional networks via importance sampling.* arXiv preprint arXiv:1801.10247, 2018.

61. Chen, J., J. Zhu, and L. Song, *Stochastic training of graph convolutional networks with variance reduction.* arXiv preprint arXiv:1710.10568, 2017.

62. Huang, W., et al., *Adaptive sampling towards fast graph representation learning.* Advances in neural information processing systems, 2018. 31.

63. Levie, R., et al., *Cayleynets: Graph convolutional neural networks with complex rational spectral filters.* IEEE Transactions on Signal Processing, 2018. 67(1): p. 97-109.

64. Liao, R., et al., *Lanczosnet: Multi-scale deep graph convolutional networks.* arXiv preprint arXiv:1901.01484, 2019.

65. Xu, B., et al., *Graph wavelet neural network.* arXiv preprint arXiv:1904.07785, 2019.

66. Li, R., et al. *Adaptive graph convolutional neural networks*. in *Proceedings of the AAAI conference on artificial intelligence*. 2018.

67. Long, J., E. Shelhamer, and T. Darrell. *Fully convolutional networks for semantic segmentation*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

68. Chen, L.-C., et al., *Semantic image segmentation with deep convolutional nets and fully connected crfs.* arXiv preprint arXiv:1412.7062, 2014.

69. Girshick, R., et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014.

70. Girshick, R. *Fast r-cnn*. in *Proceedings of the IEEE international conference on computer vision*. 2015.

71. Simonyan, K. and A. Zisserman, *Very deep convolutional networks for large-scale image recognition.* arXiv preprint arXiv:1409.1556, 2014.

72. Niepert, M., M. Ahmed, and K. Kutzkov. *Learning convolutional neural networks for graphs*. in *International conference on machine learning*. 2016. PMLR.

73. Gao, H., Z. Wang, and S. Ji. *Large-scale learnable graph convolutional networks*. in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2018.

74. Chang, J., et al., *Structure-aware convolutional neural networks.* Advances in neural information processing systems, 2018. 31.

75. Du, J., et al., *Topology adaptive graph convolutional networks.* arXiv preprint arXiv:1710.10370, 2017.

76. Duvenaud, D.K., et al., *Convolutional networks on graphs for learning molecular fingerprints.* Advances in neural information processing systems, 2015. 28.

77. Atwood, J. and D. Towsley, *Diffusion-convolutional neural networks.* Advances in neural information processing systems, 2016. 29.

78. Monti, F., et al. *Geometric deep learning on graphs and manifolds using mixture model cnns*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

79. Fey, M., et al. *Splinecnn: Fast geometric deep learning with continuous b-spline kernels*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.

80. Simonovsky, M. and N. Komodakis. *Dynamic edge-conditioned filters in convolutional neural networks on graphs*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

81. Jia, X., et al., *Dynamic filter networks.* Advances in neural information processing systems, 2016. 29.

82. Taubin, G. *A signal processing approach to fair surface design*. in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. 1995.

83. Xu, K., et al. *Representation learning on graphs with jumping knowledge networks*. in *International conference on machine learning*. 2018. PMLR.

84. He, K., et al. *Deep residual learning for image recognition*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

85. Khamsi, M.A. and W.A. Kirk, *An introduction to metric spaces and fixed point theory*. 2011: John Wiley & Sons.

86. Dai, H., et al. *Learning steady-states of iterative algorithms over graphs*. in *International conference on machine learning*. 2018. PMLR.

87. Hosseini, M.S. and M. Zekri, *Review of medical image classification using the adaptive neuro-fuzzy inference system.* Journal of Medical Signals & Sensors, 2012. 2(1): p. 49-60.

88. Lashari, S.A. and R. Ibrahim, *A framework for medical images classification using soft set.* Procedia Technology, 2013. 11: p. 548-556.

89. Othman, M.F.B., N. Abdullah, and N.A.B.A. Rusli. *An overview of MRI brain classification using FPGA implementation*. in *2010 IEEE Symposium on Industrial Electronics and Applications (ISIEA)*. 2010. IEEE.

90. Aberle, D., et al., *A primer on imaging anatomy and physiology.* Medical imaging informatics, 2010: p. 15-90.

91. Hoskins, P.R., K. Martin, and A. Thrush, *Diagnostic ultrasound: physics and equipment*. 2019: CRC Press.

92. Chaban, Y.V., et al., *Environmental sustainability and MRI: challenges, opportunities, and a call for action.* Journal of Magnetic Resonance Imaging, 2024. 59(4): p. 1149-1167.

93. Awan, R., et al. *Context-aware learning using transferable features for classification of breast cancer histology images*. in *Image Analysis and Recognition: 15th International Conference, ICIAR 2018, Póvoa de Varzim, Portugal, June 27–29, 2018, Proceedings 15*. 2018. Springer.

94. Bejnordi, B.E., et al., *Context-aware stacked convolutional neural networks for classification of breast carcinomas in whole-slide histopathology images.* Journal of Medical Imaging, 2017. 4(4): p. 044504-044504.

95. Yan, R., et al., *Breast cancer histopathological image classification using a hybrid deep neural network.* Methods, 2020. 173: p. 52-60.

96. Huang, Y. and A.C.-s. Chung. *Improving high resolution histology image classification with deep spatial fusion network*. in *Computational Pathology and Ophthalmic Medical Image Analysis: First International Workshop, COMPAY 2018, and 5th International Workshop, OMIA 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16-20, 2018, Proceedings 5*. 2018. Springer.

97. Wang, X., et al., *Weakly supervised deep learning for whole slide lung cancer image analysis.* IEEE transactions on cybernetics, 2019. 50(9): p. 3950-3962.

98. Zhou, Y., et al. *Cgc-net: Cell graph convolutional network for grading of colorectal cancer histology images*. in *Proceedings of the IEEE/CVF international conference on computer vision workshops*. 2019.

99. Xue, Y., et al., *Selective synthetic augmentation with HistoGAN for improved histopathology image classification.* Medical image analysis, 2021. 67: p. 101816.

100. Pati, P., et al. *Hact-net: A hierarchical cell-to-tissue graph neural network for histopathological image classification*. in *Uncertainty for Safe Utilization of Machine Learning in Medical Imaging, and Graphs in Biomedical Image Analysis: Second International Workshop, UNSURE 2020, and Third International Workshop, GRAIL 2020, Held in Conjunction with MICCAI 2020, Lima, Peru, October 8, 2020, Proceedings 2*. 2020. Springer.

101. Türker, I. and S. Aksu, *Connectogram–A graph-based time dependent representation for sounds.* Applied Acoustics, 2022. 191: p. 108660.

102. Kutluana, G. and İ. Türker, *Classification of cardiac disorders using weighted visibility graph features from ECG signals.* Biomedical Signal Processing and Control, 2024. 87: p. 105420.

103. Turker, I. and S.O. Tan, *Scientific Impact of Graph-Based Approaches in Deep Learning Studies--A Bibliometric Comparison.* arXiv preprint arXiv:2210.07343, 2022.

104. Li, Y., et al., *A hierarchical conditional random field-based attention mechanism approach for gastric histopathology image classification.* Applied Intelligence, 2022: p. 1-22.

105. Campanella, G., et al., *Clinical-grade computational pathology using weakly supervised deep learning on whole slide images.* Nature medicine, 2019. 25(8): p. 1301-1309.

106. Abdar, M., et al., *A review of uncertainty quantification in deep learning: Techniques, applications and challenges.* Information fusion, 2021. 76: p. 243-297.

107. Graham, S., et al., *MILD-Net: Minimal information loss dilated network for gland instance segmentation in colon histology images.* Medical image analysis, 2019. 52: p. 199-211.

108. Mobiny, A., A. Singh, and H. Van Nguyen, *Risk-aware machine learning classifier for skin lesion diagnosis.* Journal of clinical medicine, 2019. 8(8): p. 1241.

109. Fraz, M.M., et al., *FABnet: feature attention-based network for simultaneous segmentation of microvessels and nerves in routine histology images of oral cancer.* Neural Computing and Applications, 2020. 32: p. 9915-9928.

110. Liang, G., et al., *Improved trainable calibration method for neural networks on medical imaging classification.* arXiv preprint arXiv:2009.04057, 2020.

111. Rączkowska, A., et al., *ARA: accurate, reliable and active histopathological image classification framework with Bayesian deep learning.* Scientific reports, 2019. 9(1): p. 14347.

112. Huang, L., et al., *A review of uncertainty quantification in medical image analysis: probabilistic and non-probabilistic methods.* Medical Image Analysis, 2024: p. 103223.

113. Chan, H.-P., et al., *Deep learning in medical image analysis.* Deep learning in medical image analysis: challenges and applications, 2020: p. 3-21.

114. Senousy, Z., et al., *MCUa: Multi-level context and uncertainty aware dynamic deep ensemble for breast cancer histology image classification.* IEEE Transactions on Biomedical Engineering, 2021. 69(2): p. 818-829.

115. Wang, X., et al. *Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

116. Rajpurkar, P., et al., *Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning.* arXiv preprint arXiv:1711.05225, 2017.

117. Huang, G., et al. *Densely connected convolutional networks*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

118. Shen, W., et al. *Multi-scale convolutional neural networks for lung nodule classification*. in *Information Processing in Medical Imaging: 24th International Conference, IPMI 2015, Sabhal Mor Ostaig, Isle of Skye, UK, June 28-July 3, 2015, Proceedings 24*. 2015. Springer.

119. HAMILTON, B.A. *Data Science Bowl 2017*. 2017 2022; Available from: https://www.kaggle.com/c/data-science-bowl-2017.

120. Liao, F., et al., *Evaluate the malignancy of pulmonary nodules using the 3-d deep leaky noisy-or network.* IEEE transactions on neural networks and learning systems, 2019. 30(11): p. 3484-3495.

121. Ronneberger, O., P. Fischer, and T. Brox. *U-net: Convolutional networks for biomedical image segmentation*. in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. 2015. Springer.

122. Shin, H.-C., et al., *Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning.* IEEE transactions on medical imaging, 2016. 35(5): p. 1285-1298.

123. Sermanet, P., et al., *Overfeat: Integrated recognition, localization and detection using convolutional networks.* arXiv preprint arXiv:1312.6229, 2013.

124. Ciompi, F., et al., *Automatic classification of pulmonary peri-fissural nodules in computed tomography using an ensemble of 2D views and a convolutional neural network out-of-the-box.* Medical image analysis, 2015. 26(1): p. 195-202.

125. Ciompi, F., et al., *Bag-of-frequencies: a descriptor of pulmonary nodules in computed tomography images.* IEEE transactions on medical imaging, 2014. 34(4): p. 962-973.

126. Akkus, Z., et al., *Deep learning for brain MRI segmentation: state of the art and future directions.* Journal of digital imaging, 2017. 30: p. 449-459.

127. Havaei, M., et al., *Brain tumor segmentation with deep neural networks.* Medical image analysis, 2017. 35: p. 18-31.

128. Chen, L.-C., et al., *Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs.* IEEE transactions on pattern analysis and machine intelligence, 2017. 40(4): p. 834-848.

129. Brosch, T., et al., *Deep 3D convolutional encoder networks with shortcuts for multiscale feature integration applied to multiple sclerosis lesion segmentation.* IEEE transactions on medical imaging, 2016. 35(5): p. 1229-1239.

130. Song, P., et al., *Feature extraction and target recognition of moving image sequences.* IEEE Access, 2020. 8: p. 147148-147161.

131. Zhang, W., X. Fu, and W. Li, *The intelligent vehicle target recognition algorithm based on target infrared features combined with lidar.* Computer Communications, 2020. 155: p. 158-165.

132. Dong-ming, L., et al., *Research on target recognition system for camouflage target based on dual modulation.* Spectroscopy and Spectral Analysis, 2017. 37(4): p. 1174-1178.

133. Karmouty-Quintana, H., et al., *Emerging mechanisms of pulmonary vasoconstriction in SARS-CoV-2-induced acute respiratory distress syndrome (ARDS) and potential therapeutic targets.* International journal of molecular sciences, 2020. 21(21): p. 8081.

134. Shorten, C., T.M. Khoshgoftaar, and B. Furht, *Deep Learning applications for COVID-19.* Journal of big Data, 2021. 8(1): p. 1-54.

135. Ahmedt-Aristizabal, D., et al., *Graph-based deep learning for medical diagnosis and analysis: past, present and future.* Sensors, 2021. 21(14): p. 4758.

136. Yang, T., X. Tang, and R. Liu, *Dual temporal gated multi-graph convolution network for taxi demand prediction.* Neural Computing and Applications, 2023. 35(18): p. 13119-13134.

137. Rashid, P.Q. and İ. Türker, *Lung Disease Detection Using U-Net Feature Extractor Cascaded by Graph Convolutional Network.* Diagnostics, 2024. 14(12): p. 1313.

138. Coates, A. and A. Ng, *Selecting receptive fields in deep networks.* Advances in neural information processing systems, 2011. 24.

139. Kushnir, D., M. Galun, and A. Brandt, *Fast multiscale clustering and manifold identification.* Pattern Recognition, 2006. 39(10): p. 1876-1891.

140. Von Luxburg, U., *A tutorial on spectral clustering.* Statistics and computing, 2007. 17: p. 395-416.

141. Lei, H., N. Akhtar, and A. Mian. *Octree guided cnn with spherical kernels for 3d point clouds*. in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019.

142. Bianchi, F.M., et al., *Hierarchical representation learning in graph neural networks with node decimation pooling.* IEEE Transactions on Neural Networks and Learning Systems, 2020. 33(5): p. 2195-2207.

143. Bacciu, D. and L. Di Sotto. *A non-negative factorization approach to node pooling in graph convolutional neural networks*. in *AI* IA 2019–Advances in Artificial Intelligence: XVIIIth International Conference of the Italian Association for Artificial Intelligence, Rende, Italy, November 19–22, 2019, Proceedings 18*. 2019. Springer.

144. Luzhnica, E., B. Day, and P. Lio, *Clique pooling for graph classification.* arXiv preprint arXiv:1904.00374, 2019.

145. Xie, Y., et al., *Graph convolutional networks with multi-level coarsening for graph classification.* Knowledge-Based Systems, 2020. 194: p. 105578.

146. PLAMENEDUARDO. *SARS-COV-2 Ct-Scan Dataset*. 2020; Available from: https://www.kaggle.com/datasets/plameneduardo/sarscov2-ctscan-dataset.

147. Jiang, Z.-P., et al., *An improved VGG16 model for pneumonia image classification.* Applied Sciences, 2021. 11(23): p. 11185.

148. Patel, A., et al., *Revealing the unknown: Real-time recognition of Galápagos snake species using deep learning.* Animals, 2020. 10(5): p. 806.

149. Feng, P., et al. *Embranchment cnn based local climate zone classification using sar and multispectral remote sensing data*. in *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*. 2019. IEEE.

150. Pan, Y., et al., *Fundus image classification using Inception V3 and ResNet-50 for the early diagnostics of fundus diseases.* Frontiers in Physiology, 2023. 14: p. 1126780.

151. Elpeltagy, M. and H. Sallam, *Automatic prediction of COVID− 19 from chest images using modified ResNet50.* Multimedia tools and applications, 2021. 80(17): p. 26451-26463.

152. Soni, A., et al., *Iris recognition using Hough transform and neural architecture search network.* 2021 Innovations in Power and Advanced Computing Technologies (i-PACT), 2021: p. 1-5.

153. Shah, F.A., et al., *A cascaded design of best features selection for fruit diseases recognition.* Comput. Mater. Contin, 2022. 70(1): p. 1491-1507.

154. Shazia, A., et al., *A comparative study of multiple neural network for detection of COVID-19 on chest X-ray*. EURASIP journal on advances in signal processing, 2021. 2021: p. 1-16.

155. Tan, M. and Q. Le. *Efficientnet: Rethinking model scaling for convolutional neural networks*. in *International conference on machine learning*. 2019. PMLR.

156. Xu, R., et al., *A forest fire detection system based on ensemble learning.* Forests, 2021. 12(2): p. 217.

157. Amin, H., et al., *End-to-end deep learning model for corn leaf disease classification.* IEEE Access, 2022. 10: p. 31103-31115.

158. Chen, J., et al., *Deep learning-based model for detecting 2019 novel coronavirus pneumonia on high-resolution computed tomography.* Scientific reports, 2020. 10(1): p. 19196.

159. Gao, K., et al., *Dual-branch combination network (DCN): Towards accurate diagnosis and lesion segmentation of COVID-19 using CT images.* Medical image analysis, 2021. 67: p. 101836.

160. Amyar, A., et al., *Multi-task deep learning based CT imaging analysis for COVID-19 pneumonia: Classification and segmentation.* Computers in biology and medicine, 2020. 126: p. 104037.

161. Jin, S., et al., *AI-assisted CT imaging analysis for COVID-19 screening: Building and deploying a medical AI system in four weeks.* MedRxiv, 2020: p. 2020.03. 19.20039354.

162. Bai, H.X., et al., *Artificial intelligence augmentation of radiologist performance in distinguishing COVID-19 from pneumonia of other origin at chest CT.* Radiology, 2020. 296(3): p. E156-E165.

163. Jaiswal, A., et al., *Classification of the COVID-19 infected patients using DenseNet201 based deep transfer learning.* Journal of Biomolecular Structure and Dynamics, 2021. 39(15): p. 5682-5689.

164. Ardakani, A.A., et al., *Application of deep learning technique to manage COVID-19 in routine clinical practice using CT images: Results of 10 convolutional neural networks.* Computers in biology and medicine, 2020. 121: p. 103795.

165. Li, L., et al., *Artificial intelligence distinguishes COVID-19 from community acquired pneumonia on chest CT.* Radiology, 2020.

166. Dansana, D., et al., *Early diagnosis of COVID-19-affected patients based on X-ray and computed tomography images using deep learning algorithm.* Soft computing, 2023: p. 1-9.

167. Song, Y., et al., *Deep learning enables accurate diagnosis of novel coronavirus (COVID-19) with CT images.* IEEE/ACM transactions on computational biology and bioinformatics, 2021. 18(6): p. 2775-2780.

168. Wang, S., et al., *A deep learning algorithm using CT images to screen for Corona Virus Disease (COVID-19).* European radiology, 2021. 31: p. 6096-6104.

169. El Asnaoui, K. and Y. Chawki, *Using X-ray images and deep learning for automated detection of coronavirus disease.* Journal of Biomolecular Structure and Dynamics, 2021. 39(10): p. 3615-3626.

170. Rahaman, M.M., et al., *Identification of COVID-19 samples from chest X-Ray images using deep learning: A comparison of transfer learning approaches.* Journal of X-ray Science and Technology, 2020. 28(5): p. 821-839.

171. Gupta, K. and V. Bajaj, *Deep learning models-based CT-scan image classification for automated screening of COVID-19.* Biomedical Signal Processing and Control, 2023. 80: p. 104268.

172. Zheng, C., et al., *Deep learning-based detection for COVID-19 from chest CT using weak label.* MedRxiv, 2020: p. 2020.03. 12.20027185.

173. Panwar, H., et al., *Application of deep learning for fast detection of COVID-19 in X-Rays using nCOVnet.* Chaos, Solitons & Fractals, 2020. 138: p. 109944.

174. Attallah, O., D.A. Ragab, and M. Sharkas, *MULTI-DEEP: A novel CAD system for coronavirus (COVID-19) diagnosis from CT images using multiple convolution neural networks.* PeerJ, 2020. 8: p. e10086.

175. Butt, C., et al., *Deep learning system to screen coronavirus disease 2019 pneumonia.* Appl Intell, 2020.

RESUME

Pshtiwan Qader RASHID graduated elementary school in Chamachamal town. He completed high school in Chamchamal town, Iraq. He got a bachelor's degree from Computer Sciences Department of the University of Sulaimanyah, in 2006. He completed M.Sc. Degree the Computer Science Department in EMU University, in the Northern Cyprus Country. He started Ph.D in the Department of Computer Engineering of Karabük University in 2019. He is currently working as a teacher in Ministry of Education in Chamchamal Industrial School, in Iraq.