Irregular Domain Deep Learning

Sachin Shivdas Bahade

Submitted to Swansea University in fulfilment
of the requirements for the Degree of Doctor of Philosophy

Swansea University
Prifysgol Abertawe

Department of Computer Science
Swansea University

October 26, 2023

Declaration

This work has not been previously accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed ..... (candidate)

Date .....26 October 2023

Statement 1

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed ..... (candidate)

Date .....26 October 2023

Statement 2

I hereby give my consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed ..... (candidate)

Date .....26 October 2023

I would like to dedicate this work to my mother, my late father, and my idol,
Dr. B. R. Ambedkar,
who was a remarkable social reformer and the chief architect of the Indian Constitution.
We are because he was.

Abstract

In recent years, the use of machine learning and deep learning on graph data has increased significantly. Convolutional neural networks have achieved remarkable success with grid-like data such as images, but encounter enormous difficulties when learning from more general structures such as graphs. The inclusion of trainable local filters enables the automated extraction of high-level features in a regular domain; however, due to the irregular structure of graph data, regular domain deep learning operations are constrained. Thus, the development of deep learning models that can successfully learn from graph data is a promising research field with a great potential for impact. In this work, we are looking to generalise deep learning model for learning spatial feature representations on irregular domain topologies for the purpose of learning high-level model (structures, topology, parts) in medical image analysis problem. This thesis focuses specifically on graph neural networks as the primary machine learning model for effectively tacking graph data challenges.

This study investigates the use of irregular domains deep learning methods to enhance the high-level model, with applications in medical image segmentation and detection tasks. We first explore how graph construction affects the behaviour of graph convolutional operations. For this purpose, we use a parametrically specified graph to represent a localised sampling operation on an underlying domain, which we subsequently mine for features while analysing the effect of the graph's construction on the model's behaviour. After this, we learn features using advanced deep-learning, spectral, and spatial-based graph signal processing techniques for cell segmentation on immunostained images and present a comparative study. Third, we explore the problem of object recognition in an irregular environment. Conventional convolutional neural networks may process Euclidean data for object detection tasks in a number of ways, but the use of graphs to sample from image data requires special consideration, which has the possibility of generalising to non-Euclidean data. We describe a graph convolution-based region proposal method for the detection of non-Euclidean data objects. The extraction of a subgraph as a candidate for the prospective object regions is our primary focus. We dis-

covered improvements when comparing our technique to the region based convolutional neural networks method for the Euclidean domain. The last main chapter focuses on the problem of nuclei detection and finds that graph convolutional networks-based cascaded architecture outperforms convolutional neural networks-based techniques and is more stable. In conclusion, we demonstrate the stability of our irregular domain deep learning methods for graph construction, cell segmentation, and nucleus detection applications, as well as its improved performance in comparison to convolutional neural networks-based approaches.

Acknowledgements

I would like to express my gratitude to Dr. Mike Edwards and Dr. Xianghua Xie for their supervision during my studies. They have been a constant source of advice and support throughout the years, for which I will be eternally grateful. Without their tireless support and encouragement, this thesis would not have been possible. I would want to thank them for their assistance and discussions, which have helped me strengthen my professional skills. I am also thankful to Dr. Gary Tam for his support in beginning this path and his initial care.

I would also want to thank my colleagues in the SwanseaVision Lab and the department of computer science. Dr. Jingjing Deng, Dr. Ali Alqahtani, Dr. Majedaldein Almahasneh, Dr. Omnia Nagoor, and Elif Firat are specifically thanked for their help, discussion, and encouragement during my Doctorate. I would also want to acknowledge the direct and indirect support of Prof. Matt Jones, Dr. Deepak Sahoo, and Dr. Yogesh Meena.

Lastly, I would like to thank my family, particularly my mother and late father, my well-wisher Manishkumar Tayade, Csenge Berkin, for their patience, support, and encouragement despite the challenges, as well as my sister, brother, and other relatives who are proud to see me at this point.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Contents

1.1 Motivations

Deep learning methods [1] are becoming increasingly powerful in solving various difficult artificial intelligence tasks. An increase in computational power and an increasing rate of data capture have resulted in considerable advancements in deep learning methodologies [2]. In several computer vision image-related applications, including image classification [3], medical imaging [4], semantic segmentation [5], and object recognition [6–8], these deep learning techniques have shown promising performance and provide a wide range of approaches to extract features from observable data. The selection of data representations (or features) and the creation of the feature-extraction pipeline are key components in the development of the deep learning technique for representation learning [9]. The usage of local filters in convolutional layers that are trained, in contrast to conventional hand-crafted filters, led to strong performance, due to the network's ability to automatically determine what sort of features to extract by learning the weights in these trainable filters, hence avoiding hand-crafted feature extraction [10]. Deep learning approaches can learn hierarchical features without the need to manually construct them, having multiple layers of neurons that are interconnected, each layer learns progressively complicated features from the preceding layer's output. The network's initial layers learn simple features such as edges and corners, whereas the deeper layers learn more complex features such as shapes and objects. This provides deep learning models with the advantage of learning high-level features with less human intervention than manually constructed features.

One of the common characteristic behind these computer vision image tasks is that the data can be represented by grid-like structures. This permits the usage of kernel-based techniques like convolution and pooling in the form of the same local filters scanning every place on the input. Regular convolutions require that the number of neighbours for each node be constant and that these neighbours be ordered. These array-based data refer to regular domain data that exist on a regular grid or in Euclidean space. Yet, in many real-world applications, such as social, citation, and biological networks, the data can be naturally represented as graphs [11] and does not conform to this grid-like structure. However, applying these standard convolutional procedures on generic graphs confronts two fundamental obstacles. The numbers of nearby nodes generally differ for distinct nodes in a network, and there is no order information among a node's neighbours upon which we may organise them to guarantee the output is predictable. Due to these restrictions, conventional deep learning techniques have some limitations when used to analyse graph data directly. On the other hand, irregular domain refers to data that does

not exist on a regular grid or in Euclidean space, where the connectivity between data points can be represented as a graph. Many graph structures cannot be arbitrarily represented as an array, with irregular ordering and relations between nodes. Convolutional Neural Networks (CNNs) are the primary source of inspiration for Graph Neural Networks (GNNs) [1]. CNNs are well-known for their ability to extract local features using convolutional layers. This idea has been adapted by GNNs to perform localised feature extraction on graph-structured data. GNNs are a relationship-based model that uses 'message passing' between graph nodes to capture graph dependency; they are a deep neural network extension that can handle graphical information representations. Various operations are developed to process data on the graph similar to the CNNs operation, such as convolution, filtering, and Pooling [12]. Spectral convolution, spatial convolution, and graph pooling [13] are some of the examples. These operations are developed using spectral graph theory, an algebraic matrix, and a node's neighbourhood aggregation criteria. Developing such deep learning models to learn from graph data is an active research field with challenges in processing computer vision tasks.

The overarching motivation of this thesis is to explore the usage of graphs in sampling data and using that for segmentation and detection purposes, with studies in its application within medical image analysis. For these computer vision tasks, we employ graph neural network operators to build deep learning models. We also use graph-based techniques to learn localised features on applications in irregular domains and generalise the deep learning model to spatial feature representation learning for irregular domain topologies. We provide a generalising deep learning model for learning local features from imaging data, as well as a novel sampling strategy for feature learning, to examine the impact of parametrically specified graphs and their influence on the model's behaviour. We look at computer vision tasks like segmentation and detection in the context of irregular graph data and employ graph representation of image-based regular data to use irregular domain deep learning methods to address this. In this thesis, even though we are using graph representations of image-based regular data to use irregular domain methods such as GNNs, methods built may be capable of handling new irregular data as input data is in the form of a graph, which has a generic nature of irregular data structure. We provide application on a number of domains for the purpose of exploring graph spatial features, including MNIST classification, medical image segmentation, and nuclei identification, and compare them to CNN- and GNN-based approaches. These motivations can be broadly seen in three different parts, as described below.

1.1.1 Graph Construction

The recent growth in geometric and graph-based representation learning has highlighted the crucial role of graph construction in feature extraction methods. By representing the underlying domain structure, graphs depict the connectivity between nodes, enabling filters to capture relationships and patterns within the data. Consequently, any modifications to the graph, such as adding or removing edges, can impact the learned filters by altering the local context. The topology of the graph, encompassing its structure and connectivity, significantly influences information propagation. Different graph topologies affect the filters' receptive fields and their ability to capture local or global patterns. A graph filter, a commonly used filter in spectral graph convolution, relies on eigen decomposition to perform operations in the spectral domain. Changes in the graph can modify the graph's Laplacian matrix, which in turn alters the associated spectral characteristics and eigenvalues. These modifications affect the filters' frequency response and their ability to capture patterns of different scales and variations in the data. In certain application, such as the multi-label feature classification task discussed in Chapter 5, graph-based deep learning methods are utilised. This application involve different graphs with different topologies. In order to overcome the challenge of translation invariance, which impede filter learning, an effective index-based node ordering, is employed. Moreover, by closely examining the impact of graph construction on the learning of filters, we gain valuable insights into the performance of graph-based deep learning methods across various scenarios. This analysis allows us to better understand how different approaches to graph construction influence the learning process and subsequently affect the overall performance of the models. By considering this aspect, we can optimize graph-based deep learning techniques and enhance their effectiveness in different application domains.

The current state-of-the-art methods in both spatial and spectral domains have made significant progress in expanding the use of representation learning in various application areas. These areas often possess underlying structures that do not conform to classical grid-based layouts. Constructing a graph in such domains may involve interpreting the data and its inherent structure. This variability in graph construction can have implications for the performance of feature-mining methodologies and the features discovered. In this research study, we focus on investigating how graph construction affects the behaviour of graph convolutional operations. To explore this, a graph is parametrically constructed to represent a localised sampling operation on an underlying domain. Features are then extracted while assessing the effects of the graph's design on the model's behaviour.

1.1.2 Medical Image Cell Segmentation

Graph signal processing is a new subfield of deep learning that aims to handle a variety of non-Euclidean domain challenges. Due to the limitations of clinical procedures and image analysis, pathologists have trouble detecting diseases at an early stage. For more precise disease diagnosis and early detection, automated segmentation can play a crucial role [14]. However, the effectiveness and precision of the system are contingent on how the model was trained. Numerous techniques have been developed for the purpose of cell segmentation, and these can roughly be categorised into traditional (also known as non-deep learning) and deep learning methods [15]. Edward et al. [15] termed Otsu thresholding a traditional (i.e., non-deep learning) method. The objective of Otsu's thresholding method is to determine a suitable threshold that minimises the sum of foreground and background spreads [16]. In contrast to traditional method terms, Zhou et al. [17] included K-Means clustering-based methods and image thresholding-based methods in hand-crafted feature-based methods for white blood cell segmentation. Deep learning approaches have the advantage of a trained filter over traditional or hand-crafted feature-based methods, and they have proven to achieve higher accuracy in tasks such as image segmentation [18]. In addition, the recent development of deep-learning algorithms has yielded encouraging outcomes, particularly for medical imaging. Due to the complex underlying structure of the medical imaging data and the shortcomings of the training data, medical image analysis has moved to more complex models and feature mining processes such as U-Net [19].

By continuing the trend of improving deep learning models for medical image analysis with convolutional neural networks and graph neural networks, it provides interesting motivation. With this motivation, we want to address the research question of whether, by employing GNNs models, the performance of medical image cell segmentation can be improved or not. Also, the comparative performance of several hierarchical feature-learning deep learning architectures using CNNs and Graph Convolutional networks (GCNs) operators is an intriguing research topic, and overcoming the challenge of cell segmentation using a generalised model approach provides an interesting motivation. We proposed two graph-based convolution methods for cell segmentation to improve the analysis of immunostained slides. One method is inspired by the U-Net architecture and builds spectral-based GCNs, and another is a simple three-layer spatial-based GCNs.

1.1.3 Medical Image Cell and Nuclei Detection

Similar to segmentation, object detection is also one of the areas that needs exploration using GNNs. Localization and object detection play a critical role in the analysis of medical cell images. These tasks are essential for disease diagnosis, enabling the identification and localization of specific objects or regions of interest. Pathologists have challenges in the early detection of disease due to the limitations of clinical methods such as noisy imaging and ambiguity in the data. In this case, the detection of different objects in the cell image can help medical examiners automate their clinical work. Medical practitioners can assess illness severity, choose the best treatment options, and track the course of the disease by precisely identifying abnormal cells, tumours, or blood cell counts. There are many methods developed in computer vision for an object detection task in conventional convolutional neural networks such as RCNN, fast-RCNN, faster-RCNN, YOLO, etc., but the shortcoming of using graph neural networks requires attention in this regard. One of the great advantages of GNNs is that they are not limited to Euclidean domain data and have the power of generalisation. The primary representation of data to use in GNNs is graphs. Many computer vision tasks, including node-level classification and graph-level classification, can be carried out using graph structure and node information [20]. This gives motivation to work towards object detection using GNNs. To address this research gap, we investigate on current CNNs method and build new strategies that will perform object detection using GNNs with graph data as input. We use this strategy in two different applications. One is the detection of CD4 and CD8 cells in Hodgkin lymphoma microscopic immunostaining images by proposing a graph convolution-based region proposal mechanism for object detection. We created a graph-based technique to extract candidate region proposals for the cell detection task. Second, nuclei detection in histopathology images of cancerous tissue stained with standard hematoxylin and eosin stain by proposing simple GNN-based classification models in cascade arrangement improves nuclei detection. Nuclei detection in histopathology images of cancerous tissue stained with standard hematoxylin and eosin stain is also a challenging task due to the complexity and diversity of cell data.

1.2 Overview

Considering the motivations presented in Section 1.1, the purpose of this work is to investigate the use of graph-based domain deep learning methods in learning features on data sampled from an underlying domain. We use graphs as a means to represents the explicit structure of an

observation, and apply this to problems in the domain of medical image analysis; namely object detection, segmentation, and localization. This focus is driven developing on recent strategies of mining features in a data-driven manner seen in the CNN approaches, while generalising the strategy further by using GCN/GCNN.

To achieve this aim, several objectives have been identified. Firstly, we will explore the fundamentals of deep learning and graph neural networks. Secondly, we will develop a novel sampling strategy for feature learning using parametrically specified graphs. Thirdly, we will evaluate the performance of our proposed models against existing state-of-the-art methods for medical image analysis tasks such as segmentation, detection and classification. Finally, we will provide a generalizing deep learning model for learning local features from imaging data. In Chapter 2, we provide background information about the fundamentals of neural networks and graph neural networks. In Chapter 3, we will present the study of graph construction. Its importance, impact and observe the behaviour on graph convolutional operator. In Chapter 4, We explore the computer vision segmentation task and propose a method for cell segmentation using spatial- and spectral-based graph convolution and in Chapter 5, we address the challenge of object detection in irregular domain utilizes a graph convolution network for cell detection and in Chapter 6, nuclei detection problem by cascaded arrangement of graph convolution network based classification architecture.

1.3 Contributions

The main contributions of this work can be seen as follows:

- **Impact of Graph Construction on Localised Feature Mining**

  Exploring the impact of graph construction on the behaviour of the graph convolutional operations and analysing the impact of the graph's construction on the model's behaviour. Creating method to utilise a parametrically defined graph to represent a localised sampling operation on an underlying domain. In this study, we proposed weighted graph construction methods and subgraph sampling approach. We found subgraph sampling approach learn local feature more prominent with stability.

- **Graph Convolution Networks for Cell Segmentation**

  As motivation discussed in section 1.1.2, we proposed two graph-based convolution methods for cell segmentation to improve analysis of immunostained slides. First is

the spectral-based graph convolutional network, where the spectral-based graph convolution operator is combined with graph pooling in U-Net style. The proposed approach is able to learn features in an upsampling and downsampling manner, which is inspired by the CNNs-based approach with the added advantage of generalization on non-Euclidean domain data. The second spatially-based graph convolutional network method doesn't need graph pooling; with its simple architecture, it outperforms spectral-based method.

- **Graph Proposal Neural Networks for Cell Detection**

  In order to handle the difficulty of spatial feature extraction in the irregular domain and address the graph-based deep learning approach for cell recognition, we present a graph convolution-based region proposal mechanism in non-Euclidean domain data. The method of extracting positive object proposals is presented by introducing a graph proposal neural network algorithm and multi-label classification, which is analogous to the region proposal in region-based convolutional neural networks (R-CNN).

- **Cascaded Graph Convolution Approach for Nuclei Detection**

  Proposed a graph convolution-based classification model for nuclei classification and applied such models in a cascaded architecture for nuclei detection. The two-stage architecture focused on maintaining high precision and recall value. In the initial stage of soft negative elimination, a large number of background window patches are removed, while remaining positive samples are further eliminated by the second stage of hard negative elimination.

1.3.1 List of publications resulting from the research conducted for this thesis:

1. S. Bahade, M. Edwards, and X. Xie, Graph Convolutional Neural Network for segmentation of immunostained Hodgkin Lymphoma histology images, Medical Image Understanding and Analysis Conference (MIUA), 2019. [21]

2. S. Bahade, M. Edwards, and X. Xie, Graph convolution networks for cell segmentation, International Conference on Pattern Recognition Applications and Methods (ICPRAM), 2021. [22]

3. S. Bahade, M. Edwards, and X. Xie, Cascaded Graph Convolution Approach for Nuclei Detection in histopathology images, International Conference on Video and Image Processing, (ICVIP), 2022. [23]

4. S. Bahade, M. Edwards, and X. Xie, Graph Proposal Neural Networks for Cell Detection Immunostained Hodgkin Lymphoma Histology Images, IEEE 20th International Symposium on Biomedical Imaging, (ISBI) 2023. [24]

1.4 Outline

The remaining chapters of this work are outlined as follows:

Chapter 2 - Background:

The necessary background information surrounding neural networks and deep learning, as well as an overview of graph neural network theory required in understanding the formation of deep learning operations on graphs with the help of spectral graph theory and graph signal processing.

Chapter 3 - Graph Construction:

This chapter involves the study of the use of graph and their different methods of construction to understand the impact on graph convolution operator. We explore the construction of a special sampler function that facilitates convolution on a local graph. Additionally, we conduct a comparitive analysis of different deep learning methods, including both CNN-based and graph-based approaches while considering changes in node connectivity.

Chapter 4 - Segmentation:

In this chapter, we utilize spectral and spatial-based graph convolution operators to perform segmentation tasks on histology images. We define encoder-decoder architecture using a spectral-based graph convolution operator and simple architecture using a spatial-based graph convolution operator. Evaluation and results are given in comparison with state-of-the-art CNNs method.

Chapter 5 - Cell Detection:

Presents a method for cell detection in histology images using a strategy for object detection using graph data. There are three key modules in our system for detecting objects. The first creates a subgraph with objects and background graph proposals using a technique for graph proposals. The second module is a graph convolution network that assists in extracting the features from the graph proposal, and the third module is a classification network where graph features are classified into relevant labels.

Chapter 6 - Nucleai Detection:

In this chapter, we conducted a thorough investigation of existing methods for classification and regression approaches. Specifically, we focused on the task of nuclei detection in routine colon cancer histology images. Building upon the knowledge gained from the previous methods, we proposed a novel two-stage cascade architecture approach.

Chapter 7 - Conclusion and Future Work:

A conclusion of the works presented in previous chapters, noting opportunities and prospective directions for the future.

Chapter 2

Fundamentals of Deep Learning and GNNs

Contents

2.1 Introduction

There are many definitions for machine learning; in 1959, Samuel defined the field of study that gives computers the ability to learn without being explicitly programmed [25]. Another definition was presented by Tom Mitchell in 1998, who stated that a computer programme is said to learn from experience E with regard to some task T and some performance P, as measured by E [26]. For instance, classifying emails as spam or not spam. As the field progressed, the term 'machine learning' was created [27], and algorithms for making decisions or predictions based on data were developed [28, 29]. Machine learning algorithms can be broadly categorised into two fundamental types: supervised learning and unsupervised learning. Alongside these, there are other approaches, such as reinforcement learning and semi-supervised learning. Among these, supervised learning holds immense importance, as it involves building a model or function using labelled training data. Each training example consists of an input sample and a corresponding label output. The input sample comprises features, while the label output represents the desired or true output value, commonly referred to as the ground truth. The training process entails determining the function or model that best fits the data. Once the function or model is inferred, it can be used to predict or map additional samples to their respective targets. This process is commonly known as testing or prediction. In an ideal scenario, the algorithm accurately assigns class labels or regression values to unseen samples, demonstrating the effectiveness of the learned model.

There are several algorithms for supervised machine learning that are used in a variety of contexts. Linear regression and logistic regression are the two fundamental supervised learning methods. Linear regression is utilised to model the relationship between input variables and continuous output values. Consider a scenario for predicting house prices where the size of the house is used as the input feature and the corresponding price is the label. By training the model using this data, a straight-line approximation is generated, enabling the prediction of house prices for new inputs [30]. In the case of logistic regression, it is used to determine or forecast the likelihood of a binary (yes/no) event occurring, where the output variable can take only two possible values [31]. This technique finds application in various scenarios, such as identifying spam emails, detecting fraudulent transactions, or determining the malignancy of a tumor, etc. Moreover, logistic regression can be extended to tackle multi-class classification problems, where the output variable can take more than two possible values.

An artificial neural network can be seen as an extension of the logistic regression model, where more layers of feature interactions are added. These extra layers make it possible to

learn more complex, non-linear rules for making decisions. It forms the backbone of deep learning networks. When comparing the classification tasks performed by traditional machine learning and deep learning, it is clear that deep learning is much better because it can automate the learning of feature sets for multiple tasks, whereas conventional machine learning requires several sequential steps, such as pre-processing, feature extraction, smart feature selection, learning, and classification [32, 33]. Deep Learning enables learning and classification to be achieved in a single shot, as shown in Figure 2.1 [34].

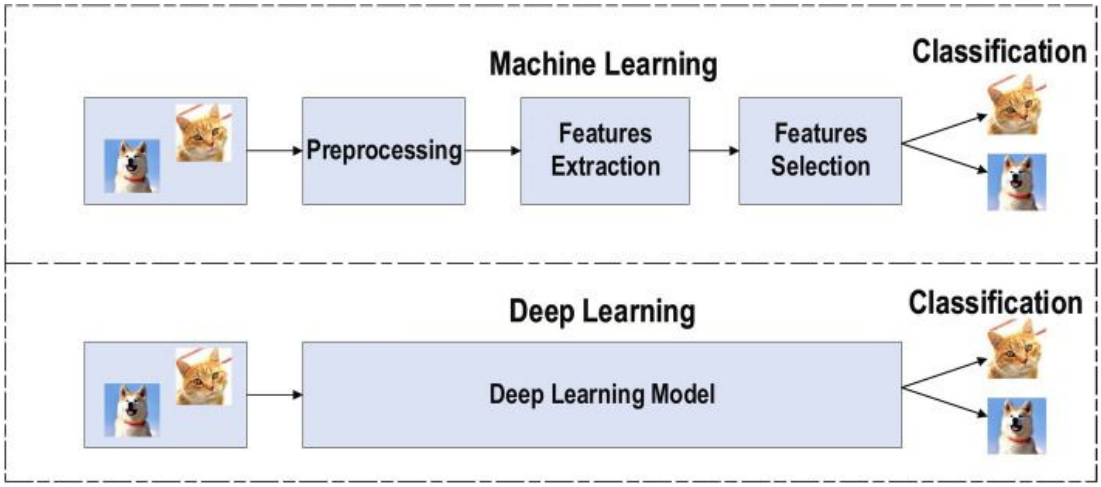


Figure 2.1: Difference between deep learning and machine learning. Deep learning automate the tasks such as preprocessing, feature extraction and feature selection.

This chapter gives the background information needed to understand this thesis. In Section 2.2, we first explain the basic idea behind the deep learning techniques used in the thesis, such as single unit biological neuron, neural networks, and convolutional neural networks. In Section 2.3, we look at the background of graph signal processing and the tools required for convolution operation in the irregular domain, for example, graph filters, graph Laplacian, graph convolutional networks, etc. In Section 2.4, we present some reviews of medical image analysis.

2.2 Fundamentals of Deep Learning

2.2.1 Neural Network

Neural networks (NNs) draw inspiration from the functioning of biological neurons, sparked by experiments on the cerebral cortex, a region of the brain capable of adaptive learning. In

biological neurons, the central cell body, or soma, is connected to branched structures called dendrites and axons. Electrical impulses, carrying information from other neurons, enter the dendrites through synapses, the connection points. The dendrites transmit this information to the soma for processing, and the resulting output signal, a sequence of impulses, travels to the synapses of other neurons via the axon. By connecting neurons together, a network is formed, allowing for complex information processing and communication. Figure 2.2 depicts the structure of a single biological neuron [35].

In the context of artificial neural networks, researchers such as Eluyode and Akomolafe [36] have undertaken studies that involve a comprehensive comparison between the behaviours of biological neural networks and their artificial counterparts. This cross-disciplinary exploration has successfully merged insights from biological systems with artificial learning, paving the way for significant advancements in machine learning and computational intelligence.

An artificial neuron receives information from another neuron, similar to a biological neuron. Weights have the ability to modify this data, which is then transmitted to the artificial neuron, activating it. To calculate the output of each neuron, the product of each input value and its corresponding weights is computed. This operation can be seen in Figure 2.3. Every neuron establishes connections with all other neurons, collectively forming a neural network. An artificial neural network with three distinct layers: the input layer, the hidden layer, and the output layer, is shown in Figure 2.4.

The perceptron, explored in depth by pioneers [37,38], was devised based on insights from the functioning of biological neural networks. From this foundational concept of the simplest neural network, further advancements led to the development of deep learning networks. A notable architectural model that has become popular is the feed-forward neural network.

2.2.1.1 Feedforward Neural Network

A feedforward neural network (FNN) is an artificial neural network with forward direction connections between nodes. It was the first and simplest artificial neural network to be created [39]. In this network, the information moves in only one direction from the input layer to output layer [40]. The perceptron, a feedforward neural network with no hidden units, is the simplest form of feedforward neural network. Hence, a perceptron consists of merely an input layer and an output layer; the output units are computed directly from the sum of their weights multiplied by their corresponding input units; a learned bias is added; and the result is sent to

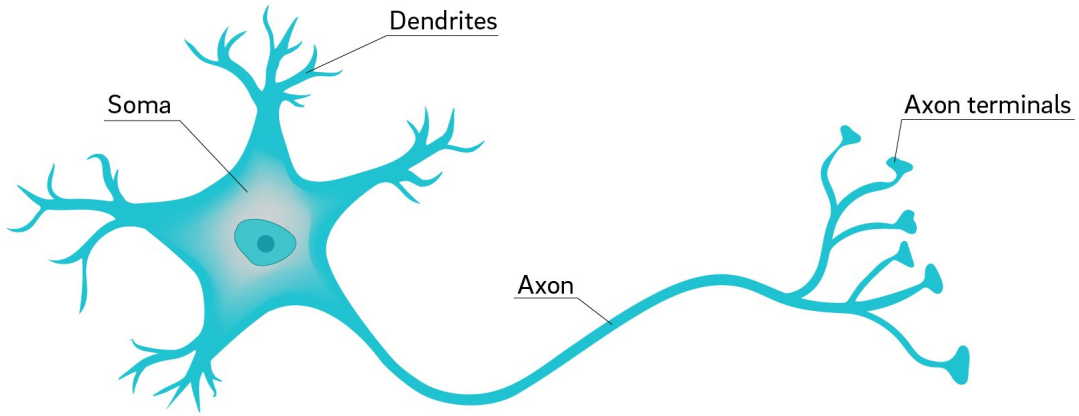


Figure 2.2: Biological neuron, consisting of soma, dendrites, and axons.

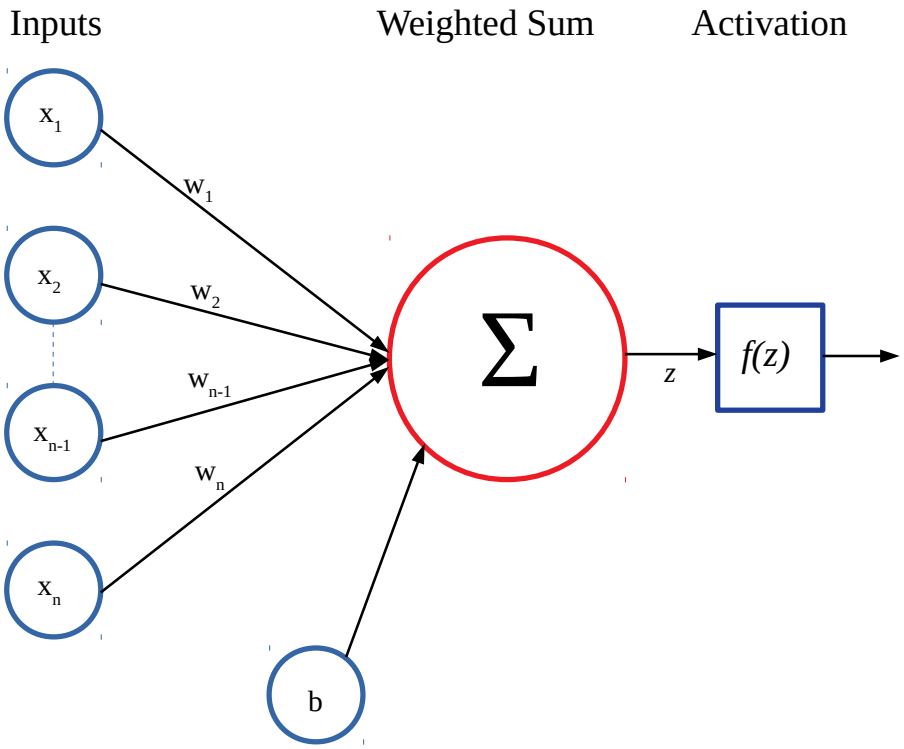


Figure 2.3: Artificial neuron / Perceptron operation.

an activation function to determine the output. [38, 41, 42]. This enables a single perceptron to transfer input characteristics to a new representation. All of these operations are illustrated in Figure 2.3 and mathematically it can be described as

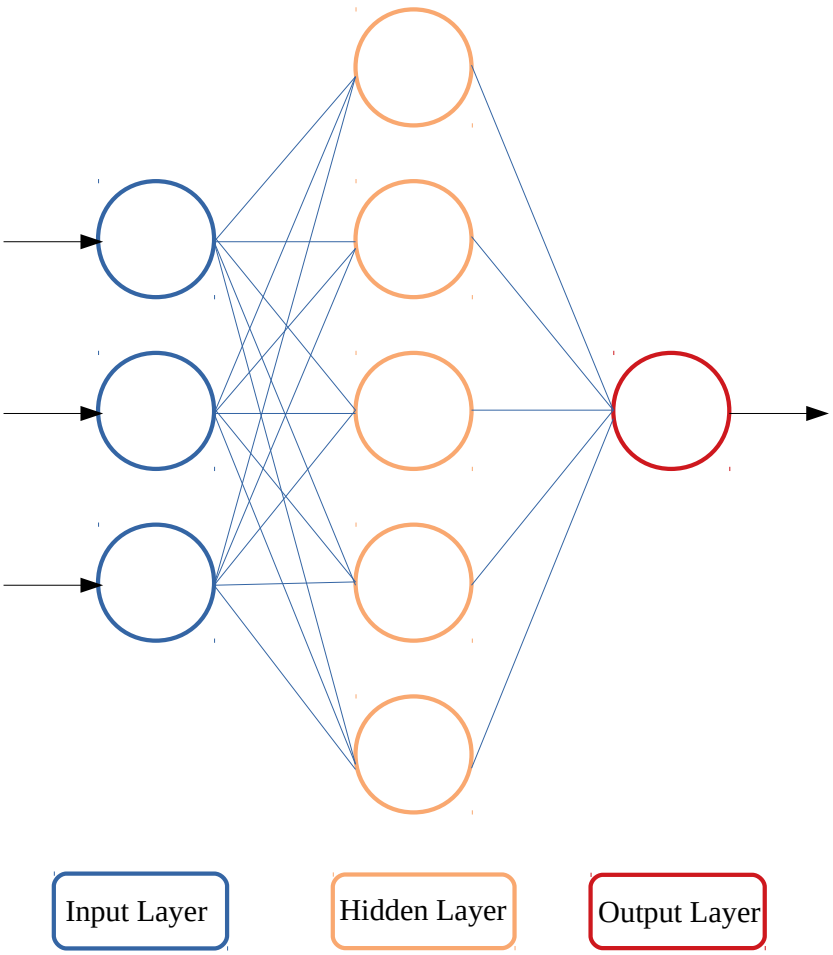


Figure 2.4: An overview of artificial neural network. Hidden layer consist of perceptrons from Figure 2.3, where each has its own trainable weights and bias as well as non-linear activation function.

$$z = b + \sum_{i=1}^{n} X_i W_i \tag{2.1}$$

where $X = (X_1, ..., X_n)$ is an given input vector, z is the linear weighted sum, b is the learned perceptron's bias, which is utilised to move the new feature space before activation, and W_i is the learnt weight for input X_i. For the perceptron's output, z can then be sent through an activation function $f(z)$. The training of a single layer perceptron involves adjusting the weights

of the connections between the input and output layers to minimize the error between the predicted output and the actual output. During supervised learning, a perceptron is presented with a set of input-output pairs, or training examples, and its weights are adjusted to minimise the error on the training data. With the new weights, this process is carried out again until convergence. To solve complex problems involving linear classification and regression, a group of perceptrons can be integrated into a "single-layer perceptron" for more complicated or higher dimensional data. An individual neuron can then be included into a Multilayer Perceptron (MLP) network, in which the output of one neuron serves as the input to the next neuron in the subsequent network layer. This concept generates a sequential network with numerous layers, each of which learns weightings based on the outputs of the previous layer, resulting in the capacity to model progressively complex function spaces. A network's depth is denoted by the number of layers it contains, and a layer's width is the number of neurons it contains. Using this principle, we can generalise Equation 2.1 as follows:

$$z_j{}^{l+1} = b_j{}^{l+1} + \sum_{i=1}^{n} X_i{}^{l} W^{l}{}_{ij} \tag{2.2}$$

where $b_j{}^{l+1}$ denoted the corresponding bias for the j^{th} neuron in the $l+1^{\text{th}}$ layer, $X_i{}^{l}$ denotes the input value of a perceptron i in the layer l. $W^{l}{}_{ij}$ is the connecting weight between i^{th} neuron from the previous layer l with j^{th} neuron in the $l+1^{\text{th}}$ layer. Each layer embeds the function space of the previous layer to a new representation, adding levels of abstraction and introducing non-linearity, allowing the MLP to obtain more accurate representations of the input data [32]. When dealing with MLPs, achieving the optimal ratio between the number of perceptrons in a layer and the number of layers becomes crucial. Larger perceptron counts enable the layer to more accurately capture the input characteristics but have little effect on generalisation as a whole due to the possibility of overfitting. While deeper networks can boost generalisation with varying amounts of feature representations, too many layers might cause input data overfitting [43, 44]. This delicate balance to achieve generalisation without overfitting makes architectural design (determining perceptron and layer counts, activation functions, etc.) a subject of several research articles [45–48].

2.2.1.2 Backpropagation

Backpropogation is employed to optimise weights through gradient computation. This backpropogation algorithm [49] is the foundation of neural network learning, where parameters are learned to obtain the minimal value of the cost function by utilising the back-propagation of

errors to improve the parameters. In supervised learning, inputs are forwarded to the network and output labels are compared to the expected result [50]. To calculate the loss and optimize the network weight, the output value must be passed backward through the network, and the difference between the two values must be determined [51]. If the rate of change of the data between perceptrons in different layers was monitored, then each weight could be changed based on that rate. The weight update would need to be minimal and incremental such that each update does not drastically alter the model. A cost function is built to update the weights by passing data through the network to produce projected outputs, which are then compared to the desired outputs. Binary cross-entropy is a prominent example of a loss function used in classification applications

$$L = -ylogp + (1-y)log(1-p) \tag{2.3}$$

where y is the anticipated outcome and p is the predicted output of the model given the input vector x. To update individual weights, the loss, L, can then be propagated backwards through the network using an optimization strategy such as gradient descent [52, 53]. We calculate the partial derivative with regard to the input and weights for each perceptron:

$$\Delta w_{ij} = -\alpha \frac{\partial L}{\partial w_{ij}} \tag{2.4}$$

where α is the learning rate, a term used to limit weight updates at each step, and w_{ij} is the weight between two perceptrons i and j in adjacent layers. Then each weight is updated by including the partial derivative, $w_{ij} = w_{ij} + \Delta w_{ij}$. The impact of specific neuron exerts on the network can also be determined by computing the partial derivative relative to the neuron bias, $\partial E/\partial b$. Depending on the task at hand, several objective loss functions can be utilised to optimise the network training scheme. There are numerous alternatives for these jobs in the MLP algorithm. The implementation of the task requires careful consideration of the appropriate optimiser and loss function [54]. With increasing network depth, back-propagation becomes less effective. As errors are back-propagated through each network layer, the derivatives for each neuron are obtained, and the gradients identified for use in stochastic gradient descent updates degrade rapidly [55]. This is because the activation function chosen determines the non-linear transformation performed to the inputs and resulting in the output of each neuron in the network. Different activation functions have different properties that can affect how the network learns and how well it can generalize to new data. One factor that can affect the performance of the network is the shape of the activation function. For example, the sigmoid

function is S-shaped and saturates at the extreme values, which can make it difficult to train deep networks because the gradients can become very small. In contrast, the ReLU (Rectified Linear Unit) function is piecewise linear and does not saturate, which can make it easier to train deep networks because the gradients remain large [56]. However, the ReLU function can also suffer from the "dying ReLU" problem [57], where some neurons become stuck at zero and stop learning. Leaky ReLUs allow a small, positive gradient when the unit is not active [58]. Some activation functions, such as the sigmoid and tanh functions, are more suited for binary classification problems, while others, such as the softmax function, are more suited for multi-class classification problems. Rectified Linear Unit (ReLU) activation has overtaken all other activation functions (such as sigmoid and Tanh) as the most popular activation function because it facilitates neural network training and provides a stable derivative for all positive values [59]. Along with activation, regularization, learning rate and different settings of optimization algorithms can lead to the overall generalization and performance of the model [9]. Regularization has been highly successful in enhancing the generalisation of deep network models [60, 61], such as batch normalisation [62] and dropout layers [63]; whereas optimization algorithms are equally important for faster convergence [64]. Some of the most used optimization algorithms for neural netwrok training are: stochastic gradient descent [65]. Mini-batch gradient descent [66], Adagrad [67], Adadelta [68], and Adam [69].

2.2.2 Convolution Neural Network

Convolutional Neural Networks (CNNs) are a specific kind of feedforward neural network. It became widely used for many computer vision applications, such as segmentation, object detection, and image classification. The invention of the LeNet architecture by Yann LeCun in the 1990s [1] was one of the early advances in the field of CNNs. This design employed convolutional layers to extract local features and pooling layers to minimise the dimension of feature maps. The input and output of each CNNs network stage are referred to as feature maps [70]. CNNs are presented as a type of neural network designed for processing grid-structured data, such as time-series (1-D grid) and image data (2-D grid of pixels). CNNs contain convolutional layers for spatially-related feature extraction, so instead of matrix multiplication, convolutional layers apply a mathematical operation called convolution that slides a locally connected filter with trainable weights through parts of the input to learn localised information from various regions of the input. This method effectively decreases the amount of trainable weights and enables CNNs to use multidimensional arrays of standard data (e.g. images) as input rather than

an arbitrary feature vector. The LeNet architecture was successfully applied to handwriting recognition tasks and contributed to the feasibility of CNNs in computer vision applications.

In recent years, a great deal of study has focused on improving the efficiency of CNNs. Due to the widespread development of deeper architectures, such as the VGG architecture [71] and the ResNet architecture [44], and a combination of convolutional layers and fully connected layers, it is now possible to construct extremely deep networks that can learn more complex image representations. Using pretraining and transfer learning enables huge datasets (such as ImageNet) to increase the network's performance on a new task and minimise the quantity of training data required. The structure of a typical CNNs includes convolutional layers, pooling layers [72], and several fully linked layers. It is one of the most studied deep learning models and is frequently employed in computer vision-related applications [3,43,71]. The convolution layer and the pooling layer, which are both often used in the fundamental development of the CNNs are shown in Figure 2.5.

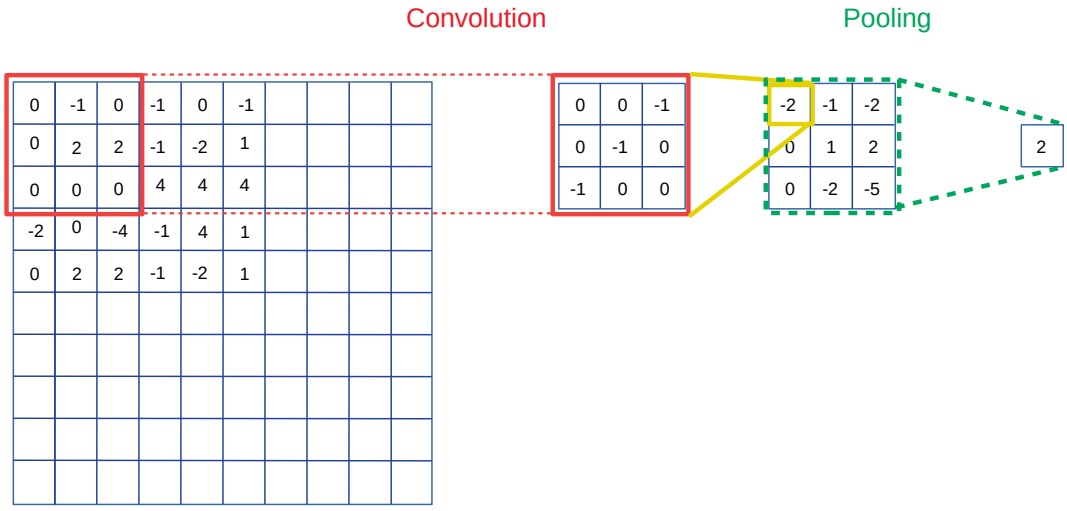


Figure 2.5: Convolution and Pooling operation: feature map value "-2" is obtain by convolving kernel over first 3×3 position of the image.

2.2.2.1 Convolutional Layer

The convolutional layer [1] is the most essential component of CNNs. It is comprised of convolutional filters known as kernels. These filters are convolved with the input image, which is represented as N-dimensional matrix, to form the output feature map. The definition of a kernel is a grid of discrete numbers or values. Each value is referred to as a kernel weight. At

the beginning of the CNNs training procedure, random numbers are assigned to serve as the kernel's weights and there are several techniques for initialising of that. As seen in Figure 2.5, each kernel executes a convolution operation across the input spatial domain, generating an output feature map that depicts the filter's responses at each spatial position of the input. To obtain a feature map, a kernel of size 3×3 is applied to an image striding in the height and width directions over an image of size 10×10. Similar to conventional neural networks, forward and back-propagation methods are used to train CNNs and estimate their parameters. A gradient-based optimization technique is used to minimise the loss function and update each filter weight parameter. At each training period, these weights are modified; hence, the kernel learns to extract significant features [34]. Filters in initial layers of a network learn low-level characteristics such as edges, curves, and corners, but filters in subsequent levels learn more complicated ideas such as parts and objects [73]. This hierarchical structure is common in image processing, which explains why CNNs function well for image processing without the requirement for preprocessing procedures such as hand-crafted features employed by conventional image processing techniques.

2.2.2.2 Pooling Layer

Pooling is performed to the feature map in order to minimise dimension and reduce the size of the input representation. Pooling is often performed after a series of convolutional layers, and consists of dividing the feature map into smaller parts and computing a summary statistic for each region. The max pooling process is depicted in Figures 2.5 and 2.6 [74]. Receptive fields are moved across input maps during the pooling procedure, which reduces the signal in each receptive field to a single pooled representation on the output feature map. The purpose of the pooling layer of a CNNs is to lower the number of trainable parameters, the network's total computation time, and to ensure translation invariance [75]. Over the years, several methods of pooling have been developed. Average and maximum pooling techniques are often used [76]. In addition to these common pooling procedures, there are other more specialised pooling operations that have been designed for specific application, that is spatial pyramid pooling (SPP) [77] which divides the feature map into many areas of varying sizes and produces a summary statistic for each region. SPP has been demonstrated to be successful for object detection tasks that require the network to locate objects at several scales. A global average pooling can be used as a replacement for CNN's fully connected layer, with the objective of producing one feature map for each associated classification problem category [78].

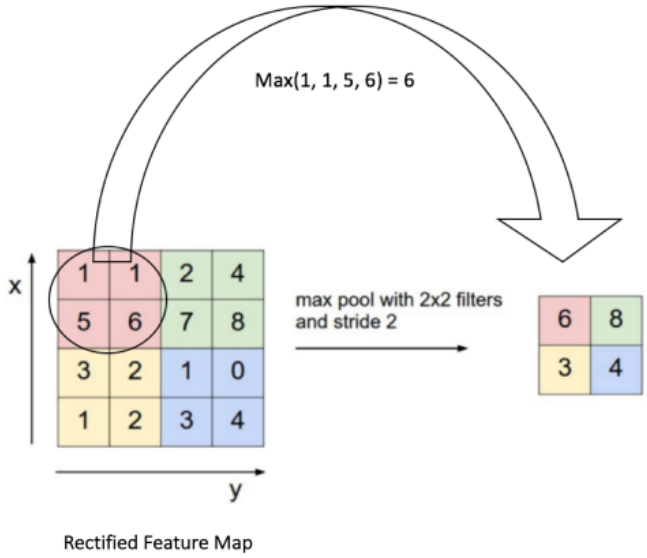


Figure 2.6: Pooling operation: Example of Max pool with 2×2 filters.

2.3 Deep Learning in Irregular Domain

An irregular domain is a domain that does not comply with the properties of regular domains and whose distance cannot be measured using Euclidean distance. Deep learning in irregular domains refers to the application of deep learning techniques to data that do not conform to the regular grid-like structures typically encountered in Euclidean space. In many real-world scenarios, data may be organised in irregular or non-Euclidean domains, such as graphs. These domains lack the regularity and fixed neighbourhood structure found in traditional grid-based data, making them more challenging to process using conventional deep learning approaches.

It is worth noting that the majority of deep learning theories and techniques have primarily focused on comprehending regular Euclidean data. The notion of striding a regular grid kernel across an input feature space does not work well in a domain where the spatial connection between points in an input feature map may not be regular. An attempt has been made to incorporate the irregular domain into a regular spatial grid. Such methods aim to use traditional CNN operators and architectures for representation learning [59, 79–83]. These strategies rely on the process of transformation between two different domains. While using such approaches, spatial information may be completely disregarded, and ordinary neural networks can be used to train non-linear mappings on input data with no intrinsic spatial relationship [84]. The alternative is to embed the input space into a conventional Cartesian grid, such as an image, and

utilise the standard CNN operators that are optimised for such a domain. The second method is gaining popularity due to performance improvements observed in the image processing community [79, 80, 85]. However, by ignoring spatial information available in the input domain, we may miss an underlying relationship between inputs, hence restricting the performance of feature representation in such environments. In contrast, by establishing a regular spatial topology onto which we project an input domain, we may make incorrect assumptions about the relationships between specific input characteristics.

In order to overcome these obstacles and gain more insight, it will be highly beneficial to perform filtering operations directly on the non-Euclidean domain. To handle such data, researchers have developed specialised deep learning methods that leverage graph-based and geometric representational learning techniques. Graph Neural Networks (GNNs) and other graph-based models have emerged as powerful tools for processing irregular data with complex relationships and dependencies between entities. GNNs extend the concept of convolutional neural networks to graph data, allowing the propagation of information through graph edges and capturing spatial dependencies among graph nodes. It attempts to provide localised filtering operations equal to those of CNNs by using the graph as an input feature space.

By embracing deep learning in irregular domains, researchers and practitioners can effectively tackle a wide range of problems in computer vision, natural language processing, social network analysis, and other domains where data naturally exists in non-Euclidean structures. This expansion of deep learning techniques to irregular domains opens up new opportunities for solving complex real-world challenges and advancing the capabilities of deep learning in diverse applications. Specifically, to tackle computer vision problems using irregular domain methods such as GNNs, there is a need to use graph representations of image-based regular data, as most computer vision tasks work on image data. To understand graph-based representational learning in this domain of irregular data, it is necessary to study spectral graph theory and graph signal processing, which we discussed in the next sections.

2.3.1 Foundation of Graph Signal Processing

Graph Signal Processing (GSP) is a framework for processing and evaluating data represented as signals on graphs or networks. In GSP, signals are associated with the vertices or edges of a graph, and the graph structure is utilised to define signal relationships. GSP attempts to extend classical signal processing methods, which are designed for data in Euclidean domains, to graph-based signals. Concepts from graph signal processing including spectral decomposition,

spectrum, and Fourier transform are closely related to the Jordan normal form of adjacency matrix. To create tools for graph signal processing, Sandryhaila et al. [86] combine computational analysis with spectral and algebraic graph theory. Some of these are discussed in the sections below.

2.3.1.1 Spectral Graph theory

The spectral graph theory [87] provides a mathematical framework for analysing the properties of graphs and their associated signals. The eigenvalues and eigenvectors of the graph Laplacian matrix may be utilised to examine the characteristics of a graph, which is one of the central concepts of spectral graph theory. The Laplacian matrix encapsulates the relationships between the vertices of a graph as a symmetric matrix, and its eigenvalues and eigenvectors can be used to define the graph's spectral decomposition. Some of the main characteristics that aid in the formation of graph signal processing are listed below.

2.3.1.2 Adjacency Matrix

Let G(V, E) be a graph containing V vertices and E edges. If there is an edge between two vertices u and v, the adjacency matrix of the graph may be expressed as 1. The adjacency can be expressed mathematically by

$$A(u,v) = \begin{cases} 1, & if \ \ (u,v) \in E, \\ 0, & otherwise, \end{cases}$$

2.3.1.3 Diagonal Matrix/Degree Matrix

It is a matrix where all diagonal elements are non-zero and all non-diagonal elements are zero. When considering the graph, it shows the degree of each node. Mathematically, it represents the following:

$$D_{i,j} = \begin{cases} deg(v_i), & if \ \ i = j, \\ 0, & otherwise, \end{cases}$$

2.3.1.4 Laplacian Matrix

The Laplacian matrix describes the structure of a graph. It is a square matrix with a size equal to the number of nodes in the graph. The diagonal elements are equal to the degree of each

node, which is the number of edges incident to the node. The off-diagonal elements are equal to -1 if there is an edge between two nodes and 0 otherwise. Mathematically it can be defines as below:

$$L(u,v) = \begin{cases} d_v, & if \;\; u = v, \\ -1, & if \;\; \text{u and v are adjacent}, \\ 0, & otherwise, \end{cases}$$

The Laplacian matrix is useful for understanding the structure of a graph because of a variety of significant properties. It is a positive semi-definite matrix [88], which indicates that all of its eigenvalues are positive. The smallest eigenvalue of the Laplacian matrix is always zero, and its corresponding eigenvector is composed entirely of ones [89]. The eigenvalues and eigenvectors of the Laplacian matrix can be used to build a spectral decomposition of the graph, which can be applied to clustering, community detection, and dimensionality reduction, among other applications. The definition of the unnormalized graph Laplacian matrix [90, 91] is

$$L = D - W$$

where D and W are the weight and diagonal matrix, respectively. Whereas normalised graph Laplacian matrix as

$$L = D^{-1/2} L D^{-1/2} \tag{2.5}$$

Furthermore, the Laplacian matrix is utilised to operate on graph-structured data, such as graph neural networks and graph convolutional networks. These methods utilize the Laplacian matrix to create graph filters, which are used to process graph signals (i.e., the data associated with each node) in a manner that takes the structure of the graph into account. The spectrum features of the Laplacian matrix are also employed to create a Fourier transform for graph signals [12, 92], which can be used to study the frequency content of graph signals.

2.3.1.5 Graph Signals

A graph signal is a signal defined on the vertices or edges of a graph. In other words, it is a function that maps each vertex or edge of a graph to a scalar value, vector, or matrix. The data on the graph can be represented as a collection of samples, and one sample is at each vertex. These samples are referred to as graph signals. It can represent various types of data, such as social networks, brain networks, traffic networks, sensor networks, and many more.

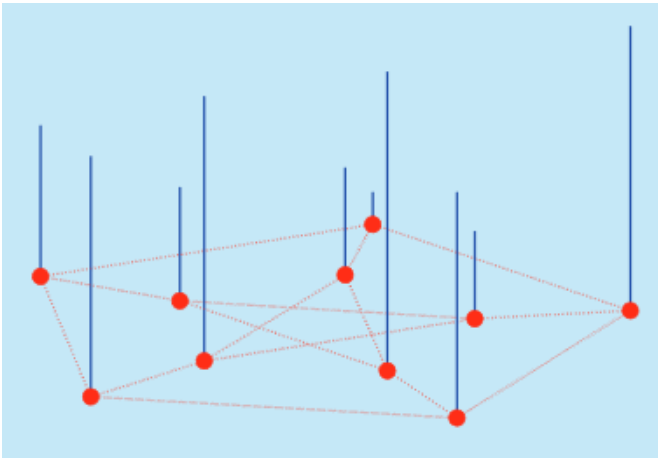


Figure 2.7: A random positive graph signal on the Petersen graph's vertices. Each blue bar's height reflects the signal value at the vertex color red from where the bar originates.

The visual representation of a graph signal is depicted in Figure 2.7 [12]. Each red dot represents a vertex, whereas each blue line above a vertex indicates the signal value at that vertex. It can be represented by a collection of vectors. While analysing real-world application data such as histology high-dimensional images, each pixel location is regarded as a node on a graph and each pixel's values are a vector. For example, grayscale images contain a single value, while RGB images contain three values of pixel that form the colour of that location.

2.3.1.6 Graph Signals Processing

Graph signal processing (GSP) serves as an extension of conventional discrete signal processing [93], allowing the application of signal processing techniques to graphs. With graphs, complex and irregular data, along with their interactions, can be effectively managed. This opens up possibilities for filtering, sampling, denoising, smoothing, modelling, and gaining insights into the structure of irregular data, inspired by principles from digital signal processing (DSP). A key concept linking DSP and GSP lies in the notion of shift, where in GSP, signals are processed at graph nodes, akin to how time signals are handled in DSP [12]. Here, applying a shift in GSP involves multiplying a signal vector by a filter matrix. Graph theory facilitates the transition from classical signal processing to GSP [87], while spectral graph theory comes into play to extract valuable information from signals and graphs. Such a framework finds diverse applications in data processing, including linear prediction, signal compression, data categorization, and consumer behaviour prediction [94]. Within the realm of GSP, several no-

table tools have been developed, further enhancing its capabilities and expanding its potential for various domains. Some of the notable tools are:

Graph Filters: A graph filter is a linear signal processing procedure applicable to graph signals. Sandryhaila et al. [86, 95] present the notion of graph filters and analyse its features, such as linearity, shift-invariance, and invertibility. Similar to DSP, graph filter processes graph signals and generates output signals. Graph shift is the core concept of graph filter. Every matrix for input that transfers to output corresponds to a graph filter that satisfies the demands of linearity and shift invariance when given in matrix-vector multiplication form. Given a graph $G = (V,A)$, each signal coefficient s_n on every node v_n can be represented by the degree of their neighbouring relations, which can be mathematically defined,

$$\tilde{s_n} = \sum_{m=0}^{N-1} A_{n,m} s_m \Longleftrightarrow \tilde{s_n} = A_s$$

where n is the number of nodes over m dimensions and s represents the signal. A graph filter is a linear operator that operates on a graph's adjacency matrix and a graph signal to generate a new output graph signal. The filter coefficients determine the filter's frequency response. In example, any matrix H $\in C^{NxN}$ that produces another graph signal $\tilde{s} = Hs$ given an input signal $s \in S$ corresponds to a graph filter [86]. There are a variety of graph filters, such as spectral filters, spatial filters, and wavelet filters. Based on the graph Fourier transform and operating in the spectral domain, spectral filters can be built to target specific frequencies or frequency ranges. Spatial filters are based on the local connectivity of the graph and operate in the vertex domain. Wavelet filters work in both the spectral and vertex domains and are based on wavelet transformations.

Graph Fourier Transform (GFT): is a tool for the transformation of graph signals from the vertex domain to the spectral domain. In classical signal processing, the Fourier transform translates signals from the time domain to the frequency domain. It provides a framework for discrete signal processing to handle and analyse complicated irregular data. Sandryhaila et. al. proposed discrete signal processing on the graph for the representation, processing, and analysis of graph-represented structured datasets [96]. The Fourier transform for graph signals is related to the generalized eigenvector of the graph's adjacency matrices. If a graph signal is sparsely represented in the spectral domain, that is, if its frequency content is dominated by a few frequencies, then it can be approximated effectively using a few spectrum coefficients [94].

The GFT is described in terms of the eigenvectors and eigenvalues of the graph Laplacian matrix, which represents the graph's local connectivity. The real and symmetric nature of the Laplacian matrix ensures that its eigenvectors form an orthonormal basis. The eigenvalues indicate the graph signals' frequencies, while the eigenvectors represent the spectral domain's basis functions. Mathematically graph Fourier transform can be defined as [96]

$$\tilde{f}(\lambda_1) =< f, \psi >= \sum_{n=1}^{N} f(n)\psi^*(n)$$

where $\psi^*(n)$ is the eigenvectors applied over function $f(n)$. Spectral decomposition of set of signals S expands each signal $s \in S$ in the basis given by the union of all generalized eigenvectors. This expansion can be written as,

$$s = V\hat{s},$$

It is called graph Fourier basis. Vector of coeffiecient is given by

$$\hat{s} = V^{-1}s$$

this is called the Graph fourier transform. In matrix form it represented as,

$$F = V^{-1}$$

and the coefficient $\hat{s}$ is called the spectrum of a signal s.

Spectral Representation of Graph signal: The representation of the signal is important for better performance, including compression, storage, and transmission. In some image compression applications, it transforms into some basis to improve quality. As demonstrated in [97], graph signals can be sparse in the frequency domain, which makes Fourier basis useful for better signal representation and compression. Graph Fourier transforms are orthogonal matrices, and their Fourier basis is orthogonal, which has the advantage of selecting the spectrum component with the largest magnitude to minimise the approximation error in the least-squares sense. The use of orthogonal graph Fourier basis is given in the compression algorithm, image compression, and incompression of sensor measurement.

In recent years, GSP has been combined with deep learning techniques to create graph neural networks that can learn representations of graph-structured data for tasks like node categorization and link prediction. The graph convolutional neural network (GCN), a form of

GNN that employs a spectral graph filter as its convolutional operation, is a key innovation that emerged from GSP.

2.3.2 Graph Neural Networks

Graph neural networks (GNNs) are deep learning-based methods that operate on graph domains [98]. With the growing accumulation of non-Euclidean data represented by graph structure data, scientists are beginning to focus on the processing of graph structure data, which may express complicated interactions between objects. Graph embedding methods, for instance, are used to translate graph structure data to simpler representations [99]. At the preprocessing phase, this technique may lose the topological information of the network structure, consequently impacting the final prediction outcome. Based on neural network research achievements, [100] introduced the idea of graph neural networks and created a model that can directly handle graph structure data. This model was further developed to demonstrate the better performance of GNNs than conventional techniques by iteratively exploiting the topological information of graphs [99]. On GNNs, various models and application studies have since been offered. GNNs are a neural network model that represents the dependence of graphs through message passing between nodes, taking into account simultaneously the size, heterogeneity, and deep topological information of the input data. Currently, GNNs demonstrate dependable performance in mining deep-level topological information, extracting the key features of data, and realising the rapid processing of massive data, such as predicting the properties of chemical molecules [101], extracting text relationships [102], reasoning the structure of graphics and images [103], link prediction and node clustering of social networks [104], network completion of missing information [105], and drug interaction prediction [106]. In recent years, several real-world applications for computer vision issues, such as image classification [103,107–110], region classification [111], semantic segmentation [112–115], and object detection [116, 117], have been created using GNN. Some of the systematic reviews of methods and applications in GNN [20, 98, 118] provide further information.

The GNN general design pipeline consists of four steps: (1) discover graph structure; (2) describe graph type and scale; (3) design loss functions; and (4) construct models utilising computational modules [98]. In this section, we provide broad design concepts and some background information. Additional information regarding the types of graphs and their construction is presented in Chapter 3.

The principle of a GNN is that a graph can be updated by message passing layers that

map it to a new version. Formally, they can be expressed as message passing neural networks (MPNNs). [119]. An MPNN layer can be expressed as follow:

$$h_u = \phi\left(X_u, \bigoplus_{v \in N_n} \psi(X_u, X_v, e_{u,v})\right)$$

where ϕ and ψ are differentiable functions (e.g., artificial neural networks), and $\bigoplus$ is a permutation invariant aggregation operator that can accept an arbitrary number of inputs (e.g., element-wise sum, mean, or max). In particular, ϕ and ψ are referred to as update and message functions, respectively. Each node can learn to gather information from its nearby nodes and then utilise this information to update its own features. This is accomplished by a process of message passing, in which each node sends and receives messages from and to its neighbours. Messages generally include the characteristics of the transmitting node and the edge connecting the nodes. Typically, the message-passing procedure is repeated many times, enabling each node to acquire information from its neighbours, change its own characteristics, and then transfer that information to its neighbours in the following round. Following a predetermined number of iterations, the final node characteristics are used for tasks such as node classification and graph classification. Researchers continue to explore and develop various architectures, making them an essential tool in the machine learning community for effectively handling graph-structured data and solving a wide range of real-world problems. Some of the further advances in GNN models for processing graph structure data that we discuss in this thesis are graph convolutional networks (GCNs).

2.3.2.1 Graph Convolution Networks

The foundation of a graph-based neural network classifier originated from the idea of spectral graph convolution [120]. As we explore existing GCN models, we can categorise them into two main groups: spectral-based and spatial-based GCNs. Spectral-based GCNs adopt the principles of graph signal processing, utilising the Laplacian and Fourier transforms to convert the irregular graph structure into a regular Euclidean space for the convolution process. On the other hand, spatial-based GCNs directly leverage the information dissemination mechanism on the graph to define the convolution operation, reflecting a propagation technique akin to the original GNN. In the subsequent discussion, we will delve into the intricacies of these two distinct models.

Spectral-Based GCNs Spectral-based graph convolutional networks are a type of neural network designed to handle graph-structured data. They are based on spectral graph theory, which is a branch of mathematics that studies the properties of graphs in terms of their eigenvalues and eigenvectors [87]. A graph Laplacian is the core operator for a spectral graph convolutional. An eigendecomposition of the graph Laplacian matrix, which gives Fourier modes and graph frequencies, forms the basis for spectral-based graph convolution [120]. From the defination given in equation 2.5, a normalized graph Laplacian can be decomposed into eigenbasis and eigenvectors as,

$$L = U\ \Lambda\ U^T \tag{2.6}$$

where $U = (u_0, u_1, ..., u_{n-1})$ is the eigenvector matrix and Λ is the eigenvalue diagonal matrix. On the graph, the normalised Laplacian matrix L and its eigenvector u create an orthogonal space, which corresponds to the Fourier transform ecosystem on the graph.

The applied graph Fourier-transfer on signal s gives,

$$F_G s = \hat{s} = U^T s \tag{2.7}$$

and inverse Graph Fourier transfer,

$$F_G^{-1}\hat{s} = U\hat{s} = UU^T s = s, \tag{2.8}$$

convolution on graph in Fourier domain from the last step is,

$$s *_G g = {F_G}^{-1}(F_G s \odot F_G g), \tag{2.9}$$

and it can be represented as,

$$s *_G g = \hat{g}(L)s, \tag{2.10}$$

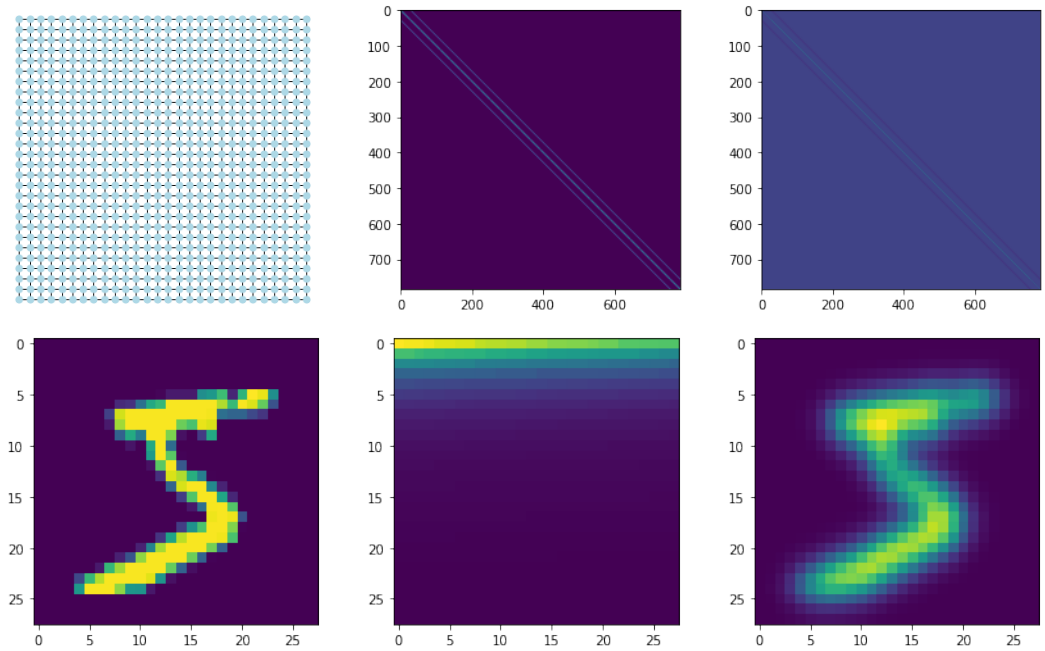


Figure 2.8: GCN Feature maps: Top Left- 2D regular grid graph 28×28, Top Middle- Adjacency matrix(A), Top Right- Laplacian matrix (L), Bottom Left- Input image as a signal on graph, Bottom Middle- Filtered coefficient ($e^{-\Lambda}$), Bottom Right- Filtered output signal.

The Laplacian matrix described in Section 2.3.1.4 can be implemented with the use of the degree matrix and adjacency matrix. Figures 2.8 depict the graphical representations of the 2D regular grid graph of size 28×28, the adjacency matrix, and the laplacian matrix, respectively. As shown in Equation 2.6, the graph Laplacian can be decomposed into eigenbasis and eigenvectors to obtain the fourier basis. The eigenvalues of the graph Laplacian is commonly used to define the filter coefficients in spectral graph convolution, similar to the filter for CNNs. A low-pass filter that attenuates high-frequency components of the spectral signal while preserving low-frequency components is commonly used. It effectively suppresses the higher eigenvalues and preserves the lower eigenvalues due to the exponential function's property of rapidly declining as its value increases. The diagonal form of this exponential filter is depicted in Figure 2.8, but it was reshaped to square, which is a bit unintuitive. This filter coefficient was applied to the input signal to produce the output signal represented in bottom row of the Figure 2.8. However, there are variants of spectral graph convolution that require the parameterization of eigenvalues or spectral filters. In the paper [121], for instance, the authors present a type of spectral graph convolution in which the filter coefficients are learned as a function of the input data and the graph structure. In particular, they learn a low-dimensional

linear transformation of the input features, which is then applied to the spectral coefficients of the graph Laplacian.

Nonetheless, spectral-based GCNs have certain drawbacks. One of these drawbacks is their sensitivity to the selection of the Laplacian matrix, as different selections may lead to distinct spectral representations of the graph. Moreover, the Laplacian matrix, although specifically set for a particular graph, may not be well-suited for managing graphs with variable sizes and topologies. Additionally, spectral-based GCNs pose another limitation: they are global frequency filters rather than translational filters. This means that they process the entire graph as a whole, lacking the ability to adapt to local variations or changes in specific regions of the graph.

Spatial-Based GCNs Spatial-based graph convolutional networks rely on the direct manipulation of the neighbourhood information of each graph node. This method includes convolving the graph signal at each node with a spatial filter determined by the graph's structure. Typically, the spatial filter is a linear function of the node features and the node features of its neighbours. The central concept of spatially-based GCNs is to aggregate the properties of surrounding nodes using a trainable weight matrix. This technique enables each node to incorporate information from its near-graph neighbours, allowing the network to discover local patterns and relationships.

Diffusion-Convolutional Neural Networks (DCNN) [122] claims that convolution is a process of diffusion between nodes and utilises the k-hop transition probability derived after random walking to establish the weight between nodes. The structure of layer m is as described below,

$$H^{(m+1)} = f(WP^kH^m), \tag{2.11}$$

where P^k is the probability of k-hop reachability between two nodes in a random walk, and W is a learnable model parameter. DCNN describes the high-order information between nodes, but its computational complexity makes it difficult to apply to a large graph. GraphSage [123] randomly samples the nearby nodes so that the neighbouring nodes of each node are lower than the specified number of samples in order to accommodate the application for large-scale networks. There are various studies that seek to define the general structure of GCNs. Among them, mixture model networks (MoNet) [124] emphasise translation invariance and convert the local structure of each node to a vector of the same size by creating a mapping function.

It learns the shared convolution kernel based on the mapping output. The message passing neural network (MPNN) [125] presents a framework by specifying a generic version of the aggregation function.

In mixture model networks, the interaction between nodes is expressed as a low-dimensional vector in the new coordinate system. Concurrently, a weight function is created on all nearby nodes centred on a node, and a vector representation of identical size is generated for each node.

$$D_j(x)f = \sum_{y \in N(x)} w_j \, u(x,y) \, f(y), \quad j = 1,...,J$$

where $N(x)$ represents the set of adjacent nodes of x, $f(y)$ represents the value of node y on the signal f, $u(x,y)$ refers to the low-dimensional vector representation of the node relationship in the coordinate system u, w_j represents the j-th weight function, and J represents the weight function number. This operation gives each node a J-dimensional representation, and the shared convolution kernel is defined on this.

MPNNs emphasise, unlike MoNet, that the essence of graph convolution is to define the aggregation function between nodes by using the aggregation function to obtain the local structure expression of each node and its neighbouring nodes and then applying the update function to itself and the local structure expression to obtain the new expression of the current node. The convolution procedure is described as follows:

$$h_v{}^{(k)} = U_k(h_v{}^{(k-1)}, \sum_{u \in N(v)} M_k(h_v{}^{k-1}, h_u{}^{k-1}, X_{uv}^e))$$

where U_k is the update function and M_k is the aggregate function. The aggregate function learned inside a spatial framework is more adaptable to the task and the particular graph structure. Figure 2.9 shows the MPNNs operation. Each graph node calculates a message for each of its neighbours. Messages are dependent on the node, its neighbour, and the connecting edge. Messages are transmitted, and each node aggregates the messages it gets using an aggregate function (such as sum, average, max, etc.). After receiving the messages, each node updates its attributes in accordance with its existing attributes and the aggregated messages.

Pooling on Graph In conventional CNNs, a pooling layer is used to reduce the resolution of the input feature map, but in the case of a spectral graph methods, there is no reduction in size due to the multiplication of the filter with the spectral signal [127]. To pool local feature

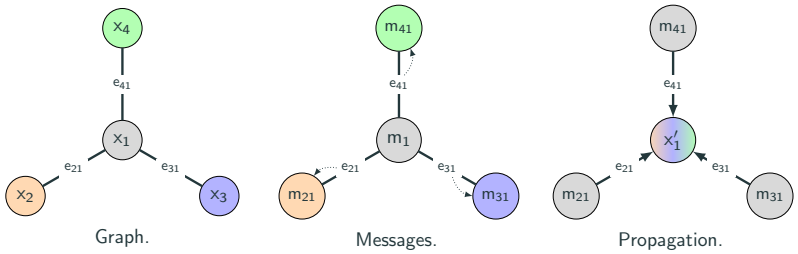


Figure 2.9: Message passing neural network: Convolution on graph operation [126]. Node $x_1{}'$ is the updated version of x_1 and aggregation message function of its neighbours.

output from the convolution layer, it is required to perform graph coarsening, which reduces the number of vertices and handles the edges between these vertices.

There are various methods for graph coarsening. One of the common methods for selecting vertices is to select a subset of the set of vertices or generate new nodes. Edwards et al. used Kron's reduction, which provides a reduction of nodes from the weight matrix by discarding vertices from the Laplacian matrix and selecting vertices used to construct a coarsened graph [128], and the Algebraic Multigrid (AMG) method for graph coarsening, which projects a signal to the coarse graph by greedy selection of vertices [127]. Whereas Chevalier et al. compare the two different coarsening schemes: strict and weighted aggregation (SAG and WAG) [129]. In SAG, the nodes are combined into a small subset. Two nodes are blocked together if their coupling is locally strong. It is shown in Figure 2.10 below.

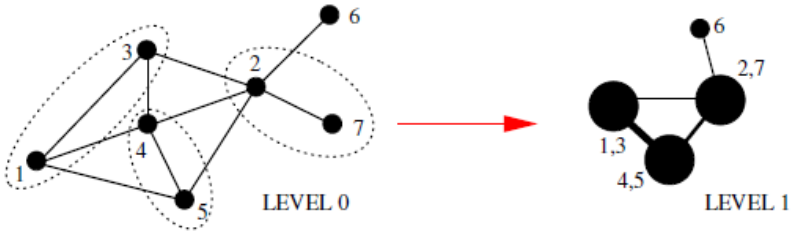


Figure 2.10: Strict Aggregation (SAG) is a method for coarsening graphs that combines disconnected pairs of nodes into a single node [129].

In the case of a WAG, each node can be divided into fractions. Different fractions belong to different aggregates. Nodes are divided into small subsets, and nodes that belong to more than one subset will be divided among the corresponding coarse aggregates as shown in the below Figure 2.11.

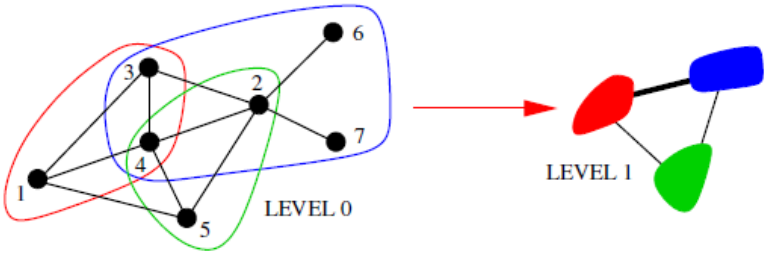


Figure 2.11: Weighted Aggregation (WAG) can divide a vertex from a finer representation among numerous coarser level nodes by using non-disjoint subsets of vertices [129].

2.4 Medical Image Analysis

Medical image analysis is the process of applying computational methods to extract clinically relevant information from medical images. Various tasks, including segmentation, feature extraction, and classification, are part of medical image analysis. Medical image segmentation is a computer task that involves dividing a medical image into various regions or segments, each of which corresponds to a different anatomical component or pathology of interest. This process plays a vital role in medical diagnosis, treatment planning, and research endeavours. Medical images, such as magnetic resonance imaging (MRI), computed tomography (CT), and immunostained histology, comprise a significant amount of visual data that can be difficult to interpret manually. Image segmentation can help extract meaningful information from these images and aid medical professionals in making precise diagnoses and treatment decisions.

There are numerous methods for segmenting medical images. One of the traditional approaches for image segmentation is developed using principal component analysis (PCA) and K-Means [130]. To assist doctors in identifying tuberculosis bacteria, [131] created a novel algorithm. They identified the issue of local minima in k-means and resolved it using the windowing method. The entire image is broken into small patches, and k-means clustering is used to segment each patch. For colour images, a region-, edge-, and pixel-based segmentation has a feature classification limitation. By transferring the RGB image into CIELAB (L*a*b) color space, [132] analyses the features of each pixel of an image, classifying the colours using K-Means and adopting support vector machine (SVM) classifiers to detect tumours by comparing the clustered image with labelled data. The two-colour components, a and b, from L*a*b colour space can also be used as a feature for K-Means clustering in the segmentation of white blood cells from microscopic images [133]. Another approach to obtaining segmen-

tation of white blood cells for acute leukaemia images is to use a linear-contrast segmentation technique using hue, saturation, and intensity (HSI) colour space [134]. In the medical field, lymphoma cancer cells have important elements that are a cluster of differentiation 4 (CD4) and a cluster of differentiation 8 (CD8), where CD8 helps to kill infected virus cells and CD4 works as a signal activator. In a clinical setting, the CD4/CD8 count ratio is used to judge the condition of the immune system. Automated segmentation of such immunostaining images helps to carry out a more accurate diagnosis of disease or even early detection [19]. A review of various automated methods for cell detection and segmentation is provided in the paper [135]. Clustering-based methods involve grouping similar pixels together based on their feature vectors using unsupervised learning algorithms like the K-Means clustering algorithm. The resulting clusters represent different regions within an image that can be further processed for object identification or classification. Whereas thresholding-based methods involve setting a threshold value that separates foreground from background pixels based on their intensity values. The Otsu method is a popular thresholding technique that automatically determines the optimal threshold value based on minimising intra-class variance between foreground and background pixels.

Recently, there has been a growth in the use of deep learning techniques for medical-image analysis, especially segmentation tasks for the early detection of disease. Notably, [136] employed deep learning techniques with the threshold-based Otsu method to identify tissue. Their approach involved dividing the image into patches and applying supervised classification to detect tumours. This pioneering work, along with other promising studies in deep learning for image segmentation [19, 134, 137–139], has garnered widespread attention from researchers. The state-of-the-art Fully Convolutional Network (FCN)-based segmentation [139] adopts an encoder-decoder architecture, fine-tunes a VGG-16 portion, and replaces the dense layer with 1×1 convolution for dense prediction. It uses skip connections to recover potential information loss during convolution. Another popular architecture, SegNet [137], eliminates the need for learning to upsample compared to FCN. On the other hand, the U-Net architecture, tailored for biomedical image segmentation with limited medical images available, is based on FCN [19]. The key difference lies in the upsample part, where pooling operations are replaced by up-convolution, increasing resolution and concatenating skip connections instead of adding, thus enhancing segmentation performance. The FCN model leverages convolutional neural networks (CNNs) to learn features from images at different scales, employing multiple layers with increasing receptive fields. Its output layer generates a pixel-wise classification

map that can be further refined using post-processing techniques like conditional random fields (CRFs).

2.5 Summary

This chapter serves as a foundation for the deep learning techniques employed in this thesis. It starts by introducing the concepts of biological neurons, neural networks, and convolutional neural networks. We discussed the area of deep learning and its benefits in feature mining. It covers neural networks, feedforward neural networks, backpropagation, convolutional neural networks covers convolutional layers and pooling layers. Moreover, we explore the significance of deep learning in irregular domains, particularly graph signal processing, and delve into graph deep learning methods, including spectral and spatial-based approaches. The general design pipeline for graph neural networks is presented, accompanied by essential design concepts. For a more comprehensive understanding of graph structures, Chapter 3 offers additional information on various graph types and their construction. Furthermore, we discuss the tools necessary for convolution operations in irregular domains, such as graph filters and graph Laplacian, thereby laying the groundwork for graph neural network design concepts. In addition, the chapter provides an overview of medical image segmentation techniques, setting the stage for our exploration of deep learning's applications in this domain.

Overall, this chapter provides a foundation for understanding the deep learning techniques used throughout the thesis. It explains how these techniques work and how they can be applied to irregular domains such as graphs. This chapter establishes a solid foundation upon which our subsequent analyses and investigations are built.

By the end of this chapter, readers should have a good understanding of the basic principles behind deep learning techniques like neural networks and convolutional neural networks. They should also understand how these techniques can be applied to irregular domains like graphs using tools like graph filters and graph Laplacian. This background information will be essential for understanding the rest of the thesis, which builds upon these concepts to develop new approaches for computer vision tasks using irregular domain deep learning.

Chapter 3

Graph construction as a study

Contents

3.1 Introduction

In the previous chapter, we looked at the background of the deep learning and irregular domain techniques. In this chapter, we investigates the specific problem of graph construction in GNNs, which is the process of creating a graph from data such as images, text, or molecular structures. Graphs serve as data structures to represent irregular data. Among the common methods for graph construction, manual annotation stands out, where a human expert manually labels the nodes and edges of a graph. Nodes are represented by features such as image pixels or text words, and edges are created based on the similarity or proximity of these features. This feature-based approach is time-consuming and subjective, and may not be feasible for large datasets. It can be automated, but may not capture important relationships between the data.

Another approach is data-driven graph construction, where the graph structure is learned directly from the data. This method can be more effective than manual annotation or feature-based graph construction, but it requires large amounts of labelled data. Recent developments in graph construction include the use of deep learning methods such as variational autoencoders (VAEs) [140] and generative models to create graphs. These methods can be trained on unlabeled data and can learn complex relationships between the data.

This chapter highlights the challenges and limitations of current graph construction methods, such as interpretability and generalization, and concludes by pointing out the importance of graph construction in GNNs and its potential impact on various applications.

3.1.1 Problem domain of Graph Construction

The specific problem of graph construction in Graph Neural Networks (GNNs) is the process of creating a graph from data such as images, text, or molecular structures in a way that can be used as input for GNNs. GNNs are a class of neural networks that are designed to operate on graph-structured data, where the data is represented as a set of nodes and edges. The graph structure captures the relationships and dependencies between the data points, and is crucial for the performance of GNNs. The graph construction process should be interpretable, meaning that the graph should reflect the underlying structure of the data in a meaningful way, be able to generalise to new unseen data, and work well on new problems. To construct a graph that captures the underlying structure of the data, it is important to choose the right type of graph and determine the right set of nodes and edges that represent the relationships between the data points. The graph construction approach can significantly impacts feature learning. It

determines the connectivity structure between nodes, influencing local and global information propagation. Additionally, it affects node representation, graph size, density, topology, and the ability to capture complex relationships. Some methods leverage unsupervised learning from unlabeled data, while domain-specific knowledge can guide construction. The choice of graph construction method is critical, as it directly influences feature learning and task performance. A poorly constructed graph may not capture the underlying structure of the data, leading to poor performance of the GNNs. Defferrard et al. shows the capability of generalising CNNs in the form of graph domains [120]. Here he uses the undirected connected graph, which is an 8 Nearest Neighbour (NNs) graph of a 2D grid with a weight matrix calculated based on the 2D coordinate of a pixel. The use of k-nearest neighbours (kNN) as a graph construction method is more suitable for supervised data classification [141]. Most of the computer vision based problems used primarily images, which are a 2D grid arrangement of the pixels. It is interesting to observe the effect of different node connections in the graphs on the performance of the model by considering the graph representation of the image data. To construct a pipeline in GNNs, there is a need to specify the graph type along with the vector signal. This type of graph is called a manual annotation or non-structural graph, where we have to first build the graph for the task, such as building a scene graph for an image [98]. In the supervised learning literature, only a few known graph construction techniques have been presented to directly handle vector-based data [142, 143]. However, a substantial amount of research has offered graph construction approaches for unsupervised, semi-supervised, and dimension reduction tasks (e.g., [144–149]). Thus, supervised works often adopt basic graph construction techniques, such as k-Nearest Neighbours (kNN) [150] or one of its versions, such as the combination of kNN and ε-radius criteria or the k-associated optimum graph [143, 151, 152].

By inspiration from [120], where a 2D regular grid can be represented in the form of a graph, using k-Nearest Neighbour, there is an opportunity to use a lattice graph from graph theory. More precisely, a square grid graph in which the vertices represent points on a plane with integer coordinates (x-coordinates in the range $1, \ldots, n$ and y-coordinates in the range $1, \ldots, m$) and in which two vertices are connected by an edge whenever the corresponding points are at a distance 1 from one another. A square grid graph can also refer to a 2D grid graph, which represents a 2D grid or lattice. The nodes in the graph represent the individual cells in the grid, while the edges represent the relationships that exist between the cells. The speciality of a 2D grid graph is that it can be embedded in Euclidean space, meaning that each node in the graph can be represented by a point in 2D or 3D space, and the edges between the nodes

can be represented by lines connecting those points. This embedding can be useful in tasks such as image or video processing, where the data is represented as a grid of pixels or voxels, because it allows the graph to be analysed and processed using the geometric properties of Euclidean space. It can also be embedded in other spaces, such as a torus, where the edges of the grid are wrapped around to create a closed surface, or in a non-Euclidean space, such as a hyperbolic space. We are exploring the interpretability and generalisation problems of 2D grid graph construction methods in the non-Euclidean domain by assigning different weight matrices and sub-graph localising approaches.

The creation of a graph can have a substantial effect on the understanding of the data it represents. A badly constructed graph can hide essential data patterns or relationships, whereas a well-crafted graph can reveal them. The weighting of a graph's edges refers to the assignment of numeric values to the connections between nodes. The weights can be used to indicate various connection properties, such as the intensity or frequency of interaction between nodes. Moreover, edge weighting may be employed to describe the distance or similarity between network nodes. Edge weighting can have a significant effect on the performance of community detection algorithms [153]. The degree weighting and strength weighting schemes can increase the quality of the community partition and the resolution limit, but they can also reduce noise resilience. Degrees of the nodes are also taken into account by MoNet [124] in their weighting method, and [154] presents the effects of a structural graph construction approach for brain networks.

In this study, to observe the impact of graph construction, we first use the baseline undirected 2D grid graph similar to that used by Defferrard et.al. [120] for MNIST image classification. After that, we apply different waiting schemes with a global and local approach, as discussed in further sections, and finally, we discuss the comparative results obtained by these approaches.

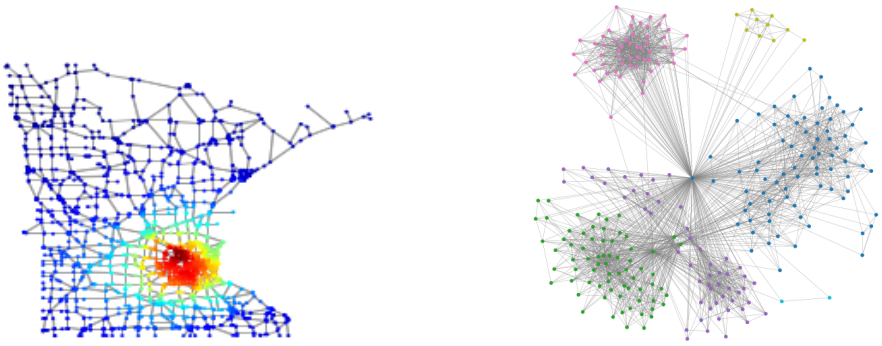
3.2 Graph construction

Graph construction plays a very important role in the process of convolution in the irregular domain. Extracting localised information from the spatial domain of such irregular data would be interesting but designing the filter and convolution operation is non-trivial when considering a domain in which the use of a spatially localised kernel and its translation across a grid are not regular. The idea of a localised neighbourhood in the spatial domain is simple, but creating a localised kernel-based filter for an irregular neighbourhood is not. The locally responsive con-

volutional layer filtering and the feature generalisation of pooling are still two operators that we would want to see implemented similarly to those introduced by CNNs. However, it can be difficult to define such an operation. It is feasible to try some kind of irregular domain embedding in a regular spatial grid. These methods attempt to use standard CNNs operators and architectures for representation learning through some reconfiguration of the spatial domain, typically through unrolling, projection, or resampling. These techniques rely on the transform operation between the original domain and the new Cartesian embedding to reconstruct the relationships between the input space's elements with sufficient accuracy. The learned CNNs filter kernels will be impacted by methods that introduce padding to assist the embedding since they will introduce spatial areas where there is no signal. The creation of CNN layers was an attempt to improve the architecture for the problem of image and volume recognition by utilising the regularity of the grid. CNNs layers are a means of adding a localised filtering constraint to fully connected neural networks. Without the presumption of utilising a grid, the field of deep learning in the irregular domain seeks to provide the same localised filtering limitations. Instead, these strategies aim to make use of the inherent spatial structure of the domain itself and reformulate the filtering procedure.

In the regular 2D grid, index i and j provide a relationship between pixels. The next pixel is situated in $i+1^{th}$, $j+1^{th}$, and the previous pixel's relationship can be described as $i-1^{th}$, $j-1^{th}$. In order to perform convolution operations, the weighted kernel must convolve over them, perform filtering operations and return an output feature map. In the domain where there is not regular cartisian space, it is difficult to produce such a regular kernel. As a real-life example in Figure 3.1 (a) and 3.1 (b), each city connected with its nearest neighbour city with variable distance, as in the case of the Facebook friend network. There might be a possibility that some friends are located at the same distance, but that does not hold true for each friend or group of friends. In this situation, the standard CNNs operator does not work, and some generalisation of convolution to such irregular domain data is required to get good performance in understanding the data and features from them.

As the focus of this thesis is to apply graph-based deep learning methods to image-based problems with a 2D grid arrangement, it can also be applied to other non-Euclidean domain problems. So we will discuss the representation of a 2D array and its different ways of construction. It also explores the behaviour of different graphs in the model.



(a) Sensory networks (b) Facebook network

Figure 3.1: Example of irregular spatial domain.

3.2.1 Representation of 2D arrays

It is vital to find a representation that allows us to perform filtering operations without making the assumption of a regular spatial kernel in order to adapt current convolutional style techniques to graph domains. The inclusion of sample relationships via graphs could provide fresh perspectives on analysis and improve data processing. In comparison to traditional data domains, graphs have the benefit that they naturally take into account a problem's irregular data relations as well as the accompanying data connectedness during analysis. Similar to traditional signal processing methods, graph signal processing tools are especially well suited to extracting meaning from data collected over graph data domains. We can construct a structure on top of which we can observe graph signals and carry out filtering operations by describing such a domain as a graph representation.

Representing the graphs as one vector requires fixing the order of the nodes. The same graph is labelled in two different ways, as shown in Figure 3.2. The graph label shows the node order from A to F. The optimal model should not depend on how it is presented. The model will produce results based on the order of the nodes. If the order changes, the results also change. Figure 3.2 depicts a graph with seven vertices and six edges. This quantity of vertices and edges permits the construction of ${}^{21}C_3$ distinct combinations of simple graphs.

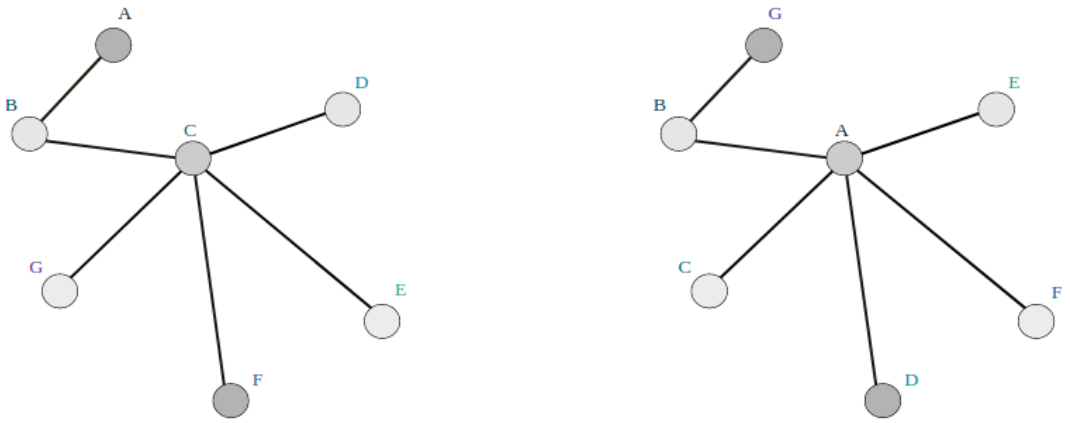


Figure 3.2: Graph labelling: The same graph can be labelled in two different ways [155].

A generalisation of graph G is that it is composed of vertices and edges, V and E, where the vertices of the graph represent the location of the input data over the entire input space and the edges show the spatial relationship between vertices. In an edge-weighted graph of N vertices, the adjacency matrix $A \in 0,1^{N\times N}$ is a binary matrix representation of the edge list where $a_{i,j} = 1$ indicates an edge between vertices v_i and v_j, given $i \neq j$. The weight matrix $W \in \mathbb{N} \times \mathbb{N}$ denotes the edge weights of the connected, undirected, non-self-looping edge vertices v_i and v_j. Such edge weighting metrics require defining a graph construction method on a given domain, with certain domains providing a natural definition for the relationship between two nodes, such as connectivity between two users or distance between cities on a map. In applications where such a definition is not readily available, one common approach is to use a Gaussian weighting function that utilises the Euclidean distances between nodes within a given locality,

$$W(i,j) = e^{\frac{i-\mu_i{}^2}{2*\sigma_i{}^2} + \frac{j-\mu_j{}^2}{2*\sigma_j{}^2}}, \tag{3.1}$$

where σ define the fall off provided by the Gaussian weighting scheme.

For array inputs like images, the Cartesian grid domain can be conceptualised as a graph describing vertex connectivity. In image processing, the relationship between pixels is presented by 4-neighbourhood and 8-neighbourhood. pixel q is in the 4-neighbourhood of pixel p(x,y) if they belong to $p(x-1,y), p(x+1,y), p(x,y-1), p(x,y+1)$, similarly for the 8-neighbourhood, which involves diagonal pixels as well as $p(x-1,y-1), p(x+1,y+1), p(x-1,y+1), p(x+1,y-1)$ in addition to 4-neighbourhood pixels. In the case of transformation into the graph domain, pixels are formulated as nodes on the graph, and neighbourhood connectivity described

as an adjacency. This phase in building the graph is crucial because it gives a foundational description of the spatial interactions among the components of the irregular domain problem. However, methods for creating such a representation for a particular domain are still an active area of research.

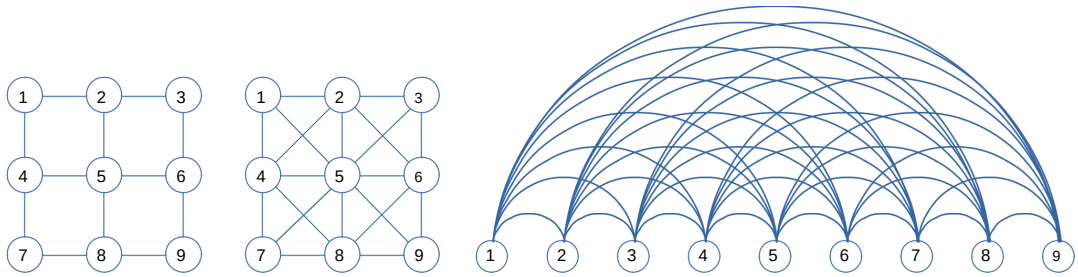


Figure 3.3: Linear ordering labelled examples of graph construction for a 3×3 regular spaced grid. Left to Right: 4-Neighbourhood, 8-Neighbourhood, fully connected.

Figure. 3.3 shows the methods of construction of a 2D grid graph where the connectivity varies from left to right. In a 4-neighbourhood, each node is connected to its four neighbouring nodes. The connection between two nodes in represented by 1 and 0 for no edge. Here, node "1" is connected with node 2 and node 4. Similarly node 5 is connected with nodes $2, 4, 6$, and 8. In the case of an 8-neighbourhood graph, each node is connected to its eight neighbouring nodes. In a fully connected graph, each node is connected to all the other nodes. Here the undirected, non-self-loop graph with 0/1 weighting is used. The connectedness between the vertices helps to undesrstand the spatial relationship between the elements. We define three graph construction method based on the distance matrix.

3.2.1.1 Binary Graph Construction

It creates a link from vertex v_i to vertex v_u if the distance (or similarity) between them is less than a pre-defined value ε. If the ε-radius neighbourhood value is 1, then it is a 4-neighbourhood grid graph where each node is connected to its nearest 4 neighbour, and the same is true for an 8-neighbourhood if the ε-radius neighborhood value is 2.

$$A_{iu} = \begin{cases} 1, \text{ if } dist(v_i, v_u) <= \varepsilon - radius \\ 0, \text{ otherwise.} \end{cases} \tag{3.2}$$

Here *dist* is a distance matrix calculated on 2D grid indices, and ε-radius is any neighbourhood from 4-N to fully connected, as shown in fig 3.3.

3.2.1.2 Euclidean Graph Construction

The adjacency matrix **A** of Euclidean graph is given by,

$$A_{iu} = \begin{cases} d, \text{ if } 1/dist(v_i, v_u) <= \varepsilon - radius \\ 0, \text{ otherwise.} \end{cases} \tag{3.3}$$

Here, inverse Euclidean matrix on 2D grid indices shows the neighbouring nodes having more distance than the far one.

3.2.1.3 Gaussian Graph Construction

An adjacency matrix can be created using a 2-D Gaussian equation by defining a weight for each edge based on the distance between the nodes. The distance between nodes can be calculated using their coordinates in the 2D grid. The weight of the edge between nodes i and j can be defined using the below equation,

$$A(x,y) = e^{\frac{x-\mu_x{}^2}{2*\sigma_x{}^2} + \frac{y-\mu_y{}^2}{2*\sigma_y{}^2}} \tag{3.4}$$

where (μ_x, μ_y) is the centre of the Gaussian, and σ_x and σ_y are the standard deviations along the x and y axes that control the spread of the Gaussian function, respectively.

To use this Gaussian function to create an adjacency matrix for a 2D grid graph, it needs to compute the value of the function for each pair of nodes in the graph. The resulting value would be the weight of the edge between those two nodes, which leads to the symmetric adjacency matrix.

3.3 Methods

An input layer, a group of convolutional and/or pooling layers, a fully connected neural network, and an output prediction layer make up the structure of a standard CNNs. The convolution of a filter over a spatial domain in the traditional CNNs operator is not trivial when taking into account an irregular domain with no regular structure. The complexity arises from the absence of ordering information and the lack of a constant distance between neighbours, prerequisites for the application of the traditional CNN operator. One solution is to do convolution in the spatial domain using multiplication in the spectral graph domain to produce the feature maps using graph signal processing methods. A graph-based convolution network, or GNNs,

learns spectral multiplier-based convolution in the spectral domain of the graph signal, much like a standard CNN network does. Training consists of making a feed-forward pass through the network to acquire outputs. This is done after the loss has been transmitted backwards through the network to update the weights that were randomly initialised.

A recent field known as "graph neural networks" processes signals using graphs. The signal x is carried by the graph G. Wavelet filtering, convolution, and Fourier Transform are just a few of the standard signal processing procedures that may be carried out on x via G since graphs have underlying knowledge about the spatial relationship between the vertices. It is possible to apply the signal processing operator to the observed data as a graph signal by describing the observed domain as a graph signal. Deep learning can be used in domains with irregular spacing that traditional CNNs are unable to convolve a regular kernel over by combining various graph signal processing approaches. As a result, the proposed method would enable deep learning to be applied to a larger range of machine learning and pattern recognition domains that have typical but spatially connected features.

3.3.1 Convolution on Graph

To perform convolution on the graph, spectral graph theory is used to define the analogue to the conventional convolution tool. A graph $G = \{V, W\}$ consists of a set of vertices, V, and a list of edge weights, W, where each edge is the non-directed, non-negative, non-selflooping value connecting two vertices, v_i and v_j. To carry out convolution on a graph, the Laplacian matrix plays an important role. The unnormalized graph Laplacian matrix L can be defined as $L = D - W$, where D is a diagonal matrix containing each diagonal entry $d_{i,j} = \sum_{i=1}^{N} w_i$ as a sum of all adjacent weights for a vertex. Given G, an observed data sample is a signal $x \in \mathbb{R}^{\mathbb{N}}$ that resides on graph G, where x_i corresponds to the signal amplitude at vertex v_i. The normalised Laplacian $\tilde{L} = I_n - D^{-1/2}LD^{-1/2}$ is an alternative to the non-normalised Laplacian, which enables normalisation of the edge weights in A.

Convolution is one of the most important processes in CNNs architecture, highlighting locally receptive features in the input image. A similar operator is presented in graph-based CNN; however, due to the potentially irregular domain, graph convolution makes use of the convolution theorem, where convolution in the spatial domain is approximated by element-wise multiplication in the frequency domain. To project the graph signal into the frequency domain, the Laplacian L is decomposed into a full matrix of orthonormal eigenvectors $U = u_{i=1...N}$ where u_i is a column of the matrix U, and the vector of associated eigenvalues $\lambda_{i=1...N}$. Using

the matrix U, the graph Fourier transform is defined as $\tilde{x} = U^x$, and the inverse as $f = U\tilde{x}$, where U^T is the transpose of the eigenvector matrix. A convolutional operator in the vertex domain can be formed as a multiplication in the Fourier space of the Laplacian operator for forward convolution. Given the spectral form of our graph signal $\tilde{x} \in \mathbb{R}$ and the spectral multiplier $k \in \mathbb{R}^{\mathbb{N}}$, the convolved output signal in the original spatial domain is the spectral multiplication, i.e., $y = U\tilde{x} \odot k$. This could potentially be expanded to support multiple input channels and output feature maps.

$$y_{s,o} = U \sum_{i=1}^{I} U^T x_{s,i} \odot k_{i,o}, \tag{3.5}$$

where I is the number of input channels for x, s is a given batch sample, and o indexes an output feature map from O output maps.

One issue with the above formulation of the filter k is the use of a spectral multiplier, an N-length vector that gives a filter with a separate parameter for each Laplacian eigenvector of the graph. This makes the convolution operator more expensive parametrically and reduces the chances of localising filters in the spatial domain. Another issue is that if the input graph is large, then creating a normalised Laplacian matrix and then the eigen decomposition of such a matrix would be beyond the limit of average computer resources. In convention CNNs, we have a kernel that keeps convolving over the kernel receptive field to get the localised region in the spatial domain. In a similar way, we can divide a larger graph into subgraphs and perform graph convolution and global pooling operations to get the unique value (sum or average) for that subgraph. After computing this for all subgraphs, we combine them to get the feature map for the larger graph. To construct a subgraph, consider a centre node and a local sampling region of size, for example, 'r', similar to the filter size in conventional CNNs. The neighbouring nodes that fall within the radius of 'r' create a graph connecting the centre node with other nodes. These subgraphs are weighted using the binary, Euclidean, and Gaussian graph construction strategies mentioned in the previous section on graph construction above. The convolution operation for such subgraphs is performed using the sampling function, as discussed in the next section 3.3.2.

3.3.2 Convolution on sub graph (Sampler function)

The GCNs filter is used for graph convolution operation in the spectral graph convolution operator [121]. There are some limitations to such a filter when it applies to large image graph

datasets, due to the spectral multiplication of the weight parameter and the GCNs filer in the form of a normalised Laplacian matrix.

One way to perform filtering operations on graph domain is to partition the large graph into small sub-graphs. In this sub-graph partitioning approach, we apply graph convolution to each sub-graph and get the aggregative result for each graph by means of a simple global pooling operation. With the help of this sub-partitioning approach, there is not only the possibility of processing large graphs but also more focus on the localised feature because of the small receptive field. This is a similar spatial filtering operation in CNNs but It is generalisable to graphs and can process large graphs in a computationally cost-effective way.

To perform convolution on a localise graph, a special sampler function is constructed. A graph $G = \{V, W\}$ consists of the number of vertices V and the list of edge weights W, where each edge is the non-directed, non-negative, non-selflooping value connecting two vertices v_i and v_j. The edge weight matrix W is constructed as per binary, Euclidean, or Gaussian weight formulas. It is also called an adjacency matrix, which describes the relationship between a node and its neighbour by some value assigned to it. This large graph adjacency matrix helps to extract the local signal x'. While creating a subgraph for each node, consider whether a main node with neighbouring nodes falls, and that can be calculated by edge weighting formula. This calculated subgraph would then transform into a modified Laplcian matrix for the graph convolution operation to be carried out. The normalised Laplacian is $\tilde{L} = I_n - D^{-1/2}LD^{-1/2}$. The unnormalized graph Laplacian matrix L can be defined as $L = D - W$, where D is a diagonal matrix containing each diagonal entry $d_{i,j} = \sum_{i=1}^{N} w_i$ as a sum of all adjacent weights for a vertex. Similar to equation 3.5, the local convolution process produces a feature map and then goes for the sum of all adjacent weights, which results in a single value for each node that corresponds to a large graph.

3.4 Case study/ experimentation on image domain

Let us define a graph $G = \{V, E, W\}$, where V & E are a set of vertices and an edge list, respectively. W is the weight matrix, which is a binary value. If there is a connection between edges, it is represented with a value of 1 and 0 elsewhere. While considering the MNIST RGB image of size 28×28 and wanting to transform on the graph domain, we need the size of the graph, i.e., the number of nodes, to be 28×28 which is 728 nodes. As the MNIST is a grey image, each node has a dimensional vector specifying its pixel values. Each given image can

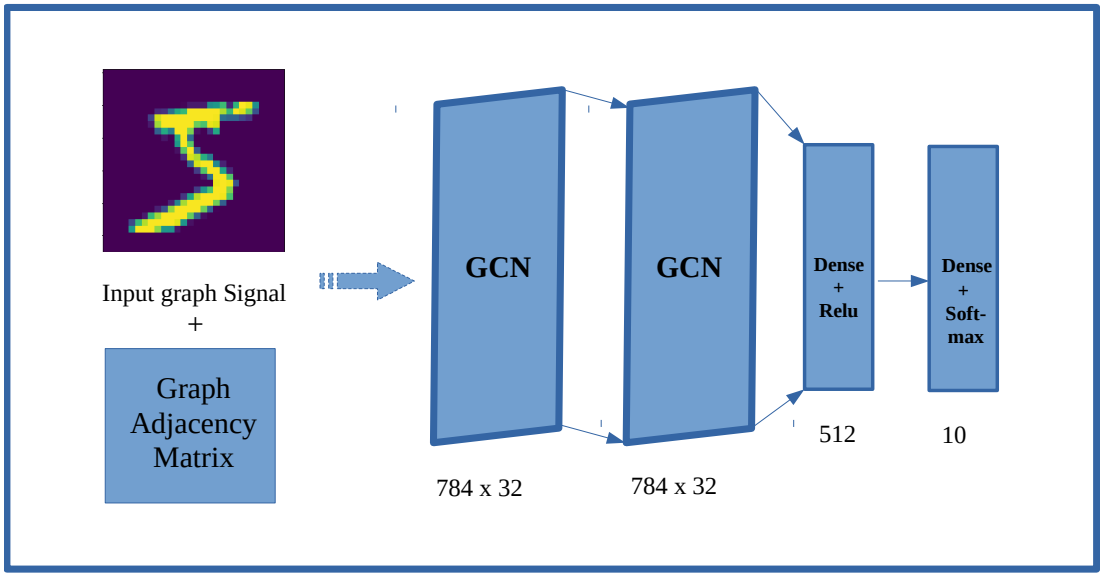
be transformed into a graph signal of size 728×1.

3.4.1 Architecture

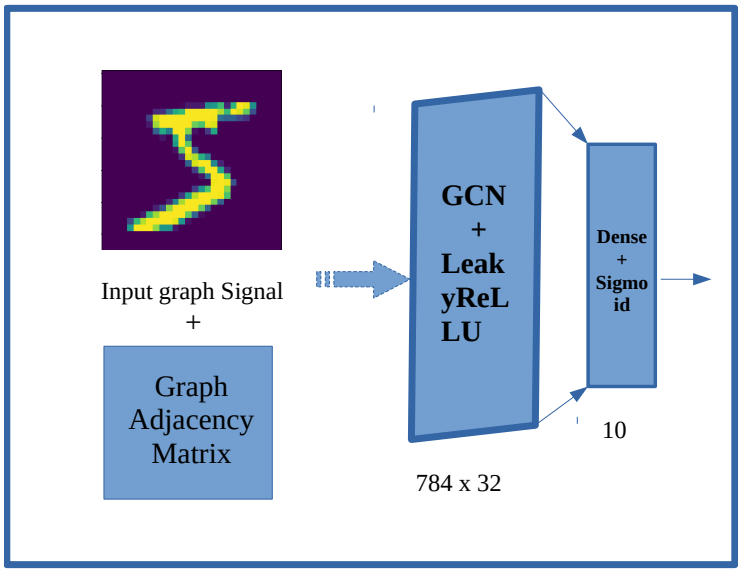
The architectures of the build network are shown in Figure 3.4 and 3.5. There are two architectures designed: one is for global feature learning and another focuses on the local subpartition graph patch. Each network uses all of the above graph construction methods under study. In global architecture, a network contains two graph convolution networks followed by two fully connected layers. The MNIST grey pixel values of each sample are passed through the graph convolution operator to extract the features. The extracted feature vector is classified into fully connected layers. The last layer is a dense classification layer with softmax non-linearity. The model training parameters are the same for all Global and local models. In local architecture, as shown in Figure 3.5, the graph convolution layer is replaced by the sampler function, which includes the customised graph convolution layer with golbal average pooling for learning localised features. The pictorial function of the sampler function is explained in Figure 3.6. Each subgraph is constructed from the edge weight formula, which is a binary, Euclidean, and Gaussian-weighted matrix. This weighted local subgraph, along with the associated signal value, is passed to the graph convolution, followed by average pooling, which gives a single node value for each local graph. For example, a MNIST image of size 28×28 has 784 nodes. For each node, a local subgraph of size 5×5 created. There is a 784 local subgraph passed for convolution operation as described in section 3.3.2 and it outputs a same size feature map with the same number of channels. This feature map passes to another sampler function layer, the dense layer, and finally achieves classification with the dense softmax layer.

3.4.2 Experiments and Results

For all experiments we have used the same parameters: batch size = 32, epochs = 50, regularisation rate = 5e-4, and learning rate = 1e-3. Our models are implemented in Tensorflow 2.0 and Keras and are trained and evaluated on a PC with an Intel i7 CPU and an NVIDIA GEFORCE GTX 1080Ti.



(A.) Two GCN: elu activation after each GCN



(B.) 1 GCN

Figure 3.4: Global Architecture: Single and Two layer GCN architecture for processing global graph construction methods for classification.

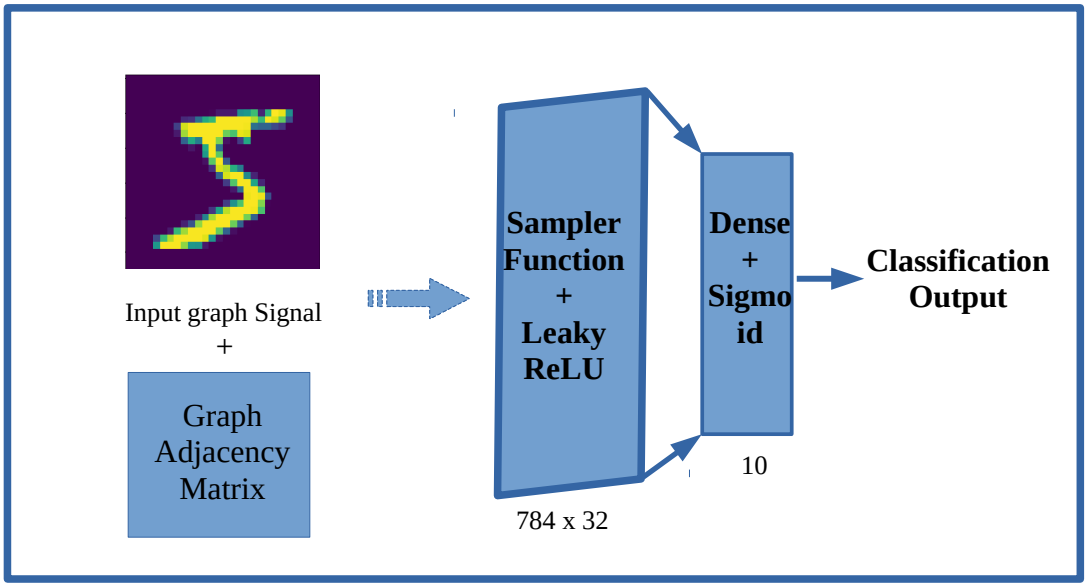


Figure 3.5: Local Architecture: Single sampler layer architecture for classification. Graph and image signals pass through the sampler function, where they are processed through a local subgraph sampling mechanism.

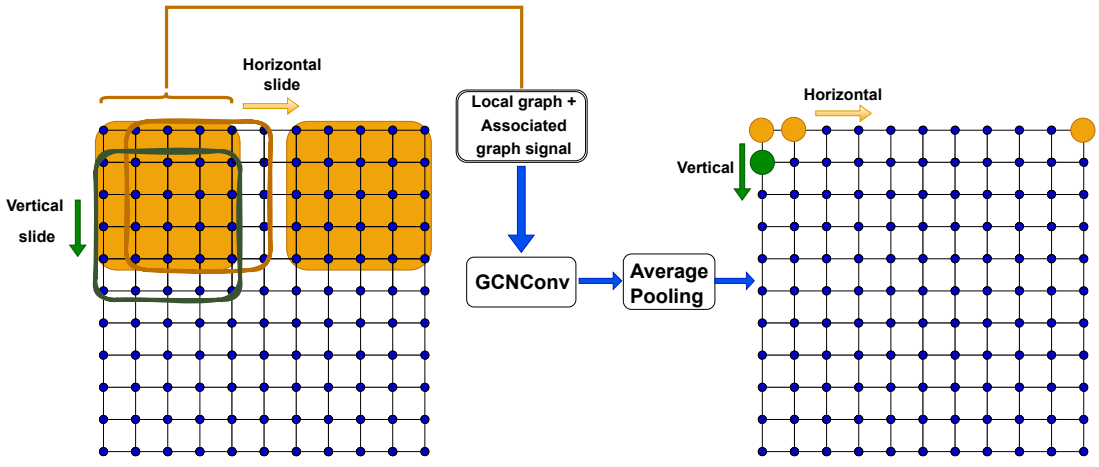


Figure 3.6: Sampler Function: Local grid graphs, sized 5×5, encapsulate local structures. As they shift horizontally and vertically, they convey distinct graph signals. Processed via GCNConv and average pooling, each local graph, paired with its graph signal, yields a single value specific to that graph.

3.4.2.1 Global Binary Adjacency matrix

In this experiment, we generated a binary adjacency graph with 1 and 0 edge connections. Global graph in the sense that the entire graph was used as the image size for the feature training phase. MNIST image size is 28×28. Hence, we create a 2D grid graph with a 28×28 grid size. By default, edges are connected in four distinct neighbourhood patterns, as depicted in Figure 3.3. It indicates that each node is connected to a maximum of four nearest neighbours with an edge distance of 1 or 0 if none are present. This is the binary representation of the nodes that are connected with a neighbourhood length of 1. Given that the length of the neighbourhood ranges from 1 to 27, the number of connected nodes between each node increases as the distance between them increases. For instance, the length of neighbourhood 1 produces a 4-neighbourhood, neighbourhood 2 gives an 8-neighbourhood, and so on up to a 27-neighbourhood. Figure 3.3 depicts a fully connected graph where each node is connected to all other nodes. The neighbourhood distance is computed using the Euclidean distance equation, where distances that fall within or are equal to the neighbourhood distance are noted with a 1, and the remainder are marked with a 0. Due to the fact that it is a binary adjacency matrix, all related nodes are represented by 1 and 0. As a GCNs Filter for the Learning feature over a whole 2D grid graph, we change the graph to a modified laplacian. For the classification task, the input graph signal and associated whole graph adjacency matrix are provided to the model as depicted in Figure 3.4.

3.4.2.2 Global Euclidean Adjacency matrix

Similar to the binary adjacency matrix, we have built the Euclidean adjacency graph, in which edge connections indicate Euclidean distance and 0 otherwise. It will start with the neighbourhood distance of 1 to 27. The length of neighbourhood distance 1 tells us which nodes lie within radius 1, the length of neighbourhood distance 2 tells us which nodes fall within radius 2, and so on up to a 27-neighbourhood distance. In the case of this adjacency matrix, the invested Euclidean distance has been considered. As a node moves away from the centre nodes, the Euclidean distance between them begins to decrease. In other words, the greater the Euclidean edge distance value, the closer the nodes are. The value 0 is assigned to all nodes whose Euclidean distance is greater than or equal to the neighbourhood distance. As a GCNs Filter for the Learning feature over a complete 2D grid graph, we have additionally transformed the graph to a modified Laplacian. For the classification task, the input graph signal and related graph adjacency matrix are provided to the model as depicted in Figure 3.4.

3.4.2.3 Global Gaussian Adjacency matrix

In this manual construction of a Gaussian adjacency matrix, the Gaussian value for each node is calculated by entering the parameters μ_x, μ_y, σ_x and σ_y into Equation 3.1. In each global adjacency matrix, assuming a single neighbourhood distance, produces a graph; similarly, for each of the 27 radiuses , present the corresponding number of graphs. In the case of Gaussian, the value of the standard deviation is treated as if it were a neighbourhood distance, which results in a single graph. To produce Gaussian values for all nodes with reference to the centred node and to shift the μ_x and μ_y values of each centre node to the next centre node. This produces a complete Gaussian adjacency matrix. We may generate different graphs by adjusting the standard deviation value. All self-nodes are set to 0, and the threshold for clipping the Gaussian curve, denoted by sigma x and sigma y, is equal to the neighbourhood radius.

3.4.2.4 Local Adjacency matrix

In this local adjacency matrix, the graph construction approach is similar to that of the global matrix, with the dimension being the sole difference. Instead of constructing a graph for the entire image, we divide the 2D grid graph of image size into subgraphs of lower sample size. Construct a sampling-size 2D grid graph, for instance, 5×5 in our case, using the same method described for global graphs for binary, inverse Euclidean, or Gaussian distance adjacency matrices. After creating the subgraph, the convolution on the subgraph using the sampler function will be performed, as described in Section 3.3.2. Figure 3.6 illustrates the straightforward architecture of the sampler function used to extract local subgraphs from a large graph and pass them through a graph convolution and pooling operation. Figure 3.5 demonstrates the classification of the MNIST digit image using two sampler functions with output channel 32 in conjunction with two desnse layers and a softmax classifier. For a fair comparison, we maintain the same architecture design for both global and local systems.

3.4.2.5 Discussion

To investigate the behaviour of the model, a range of hyperparameters are utilised throughout the training of both local and global models. In order to ensure a fair comparison, the FCNN, CNN, and GCN models maintain a simple architecture. It is understandable that increasing the hyperparameter could result in an improvement in the model's performance. As an illustration, we have considered the global graph model depicted in Figure 3.4A. with two GCN layers with output channel 32, followed by a Dense layer with ReLU activation and a softmax classifier.

Our straightforward architecture for all models consists of a single layer, then followed by a classifier. The performance is shown in Table 3.1 after the inclusion of these extra layers. Here are three distinct models constructed using three distinct techniques. This accuracy measure is evaluated at epoch 50 with a radius of 1. In comparison to the performance of the simple global architecture depicted in Figure 3.4B., Table 3.1 reveals that its accuracy is inferior. To understand the effect of the graph construction approaches on the working behaviour of the GCN operator, it is crucial to maintain a simple architecture throughout, consisting of a single layer between the input layer and output classifiers. Let's examine the global model results depicted by the changing graphs in Figure 3.7. Each curve is represented by distances ranging between 1 and 27. During epoch 50, accuracy and loss value are considered when plotting for each radius. Observing the behaviour of these simple global models with various graphs reveals the variable learning feature according to the radius and graph techniques. At radius 5, there is a major reduction in the ability to learn over a Gaussian graph, whereas binary and inverse Euclidean continue a gradual decline until radius 15, after which binary declines sharply. In contrast, the Gaussian graph joins the Euclidean graph trend with a smooth step. Conclusions drawn, we can conclude that inverse Euclidean and Gaussian graphs follow some stability in the consideration of distant features, whereas binary graphs do not. It is crucial that all graphs trained at epoch 50, with a radius of 1, exhibit improved performance. When selecting a radius of 1, let's train all global models across 500 epochs. As seen in Figure 3.10, the Gaussian graph has low performance when compared to the other two graphs. This study demonstrates that the Gaussian method of graph formation lacks the ability to establish efficient edge connections to be learned by the GCN operator and hence favours other methods of construction.

Let's discuss the local graph method shown in the architecture of Figure 3.5. As depicted in the sampler function Figure 3.6. The implementation of the sampler function in local architecture consists of dividing the graph into subgraphs, passing these local graphs with local signals to the GCNConv operator, followed by average pooling. Their operation is described in Section 3.3.2. There are various model combinations that are trained. It corresponds to the strategy of a single graph with many radii and a single radius with many graphs. The training curves are illustrated in Figures 3.11 and 3.12. When studying these curves, we discover that modifying the radius of a local, smaller subgraph has no appreciable effect on performance.

The performance of local and global graph models can be seen in Figure 3.8. In this instance, training was conducted until epoch 500 while maintaining the optimal radius of 1 for the global binary graph. Using the same parameters as our proposed local architecture, the

graph convolution operator acquires localised features more effectively than global features, which struggle to maintain performance. Its constancy across 500 epochs is remarkable to observe. Even when compared to CNN and FCNN models in Figure 3.9, the global GCN model's performance is superior. The outcomes of the training can be seen in Table 3.1, which reveals that all local models outperform global models at epoch 50, which is intriguing. Improved results can be noticed in addition to working on larger graphs.

	Layer	**Activation**	**Train/Test Acc@E50**
FCNN	Dense	ReLU	0.9973 / 0.9627
CNN	Conv2D	ReLU	0.9998 / 0.9843
GCN Global Binary	GCNConv	LeakyReLU	0.9508 / 0.9406
GCN Global InvEuclidean	GCNConv	LeakyReLU	0.9511 / 0.9422
GCN Global Gaussian	GCNConv	LeakyReLU	0.9421 / 0.9315
GCN Local Binary	Sampler function	LeakyReLU	0.9522 / 0.9514
GCN Local InvEuclidean	Sampler function	LeakyReLU	0.9527 / 0.9532
GCN Local Gaussian	Sampler function	LeakyReLU	0.9532 / 0.9497

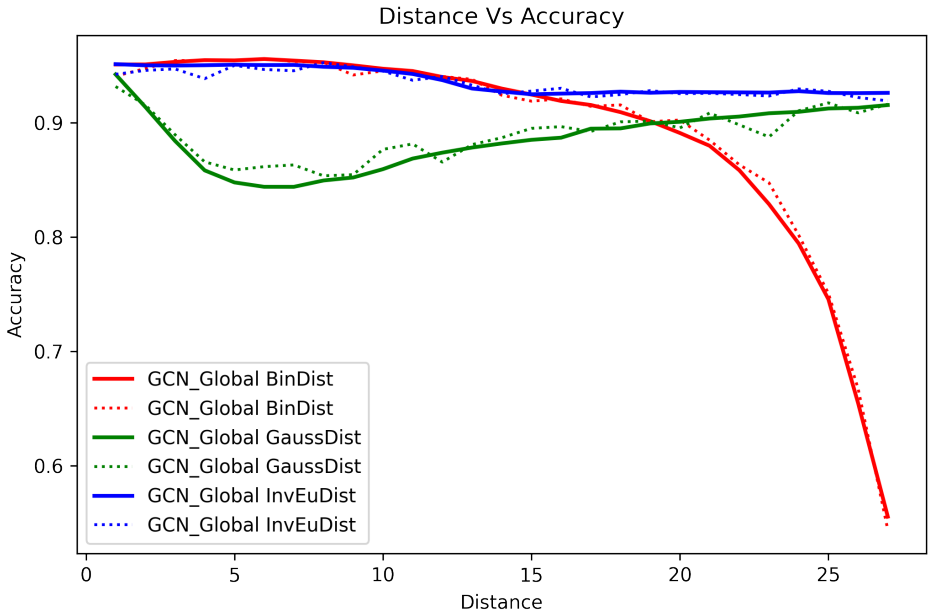
Table 3.1: Architecture parameter settings: FCNN, CNN, GCN Global and GCN Local. Sample function internaly used GCNConv and global average pooing. Here GCN Global trained at 500 epoch and observer at 50. CNN and FCNN trained at 100 epoch. Local Binary at 500, Gaussian and Inverse Euclidean at 1000.

	Training (%)	**Validation (%)**	**Testing (%)**
Binary Distance	0.9954	0.9810	0.9795
Inverse Euclidean Distance	0.9963	0.9780	0.9800
Gaussian Distance	0.9956	0.9778	0.9773

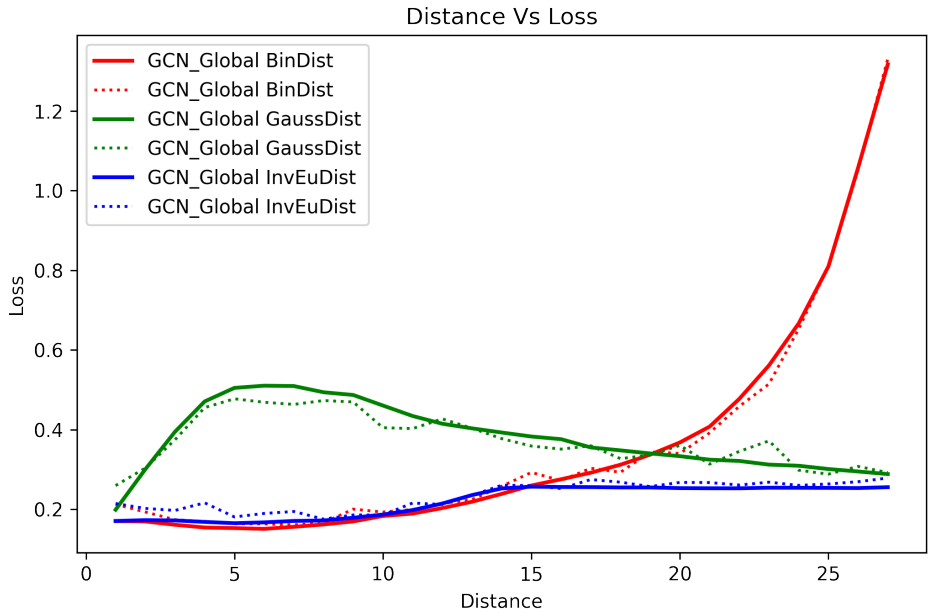
Table 3.2: Global graphs with architecture 3.4A.: Metrics accuracy value observed at epoch 50 and radius 1.

3.5 Summary

With the use of the local subgraph approach, this study suggests a unique technique for conducting convolutional operations on irregular graphs by breaking down the larger graph into smaller subgraphs. Convolutions are used to handle the non-trivial irregular kernel design while learning spatially localised features in the spectral domain of the graph Laplacian. Proposed weighted graph construction methods explore the behaviour of models in terms of stability. Results are shown for learning on the entire graph as a problem of irregular domain

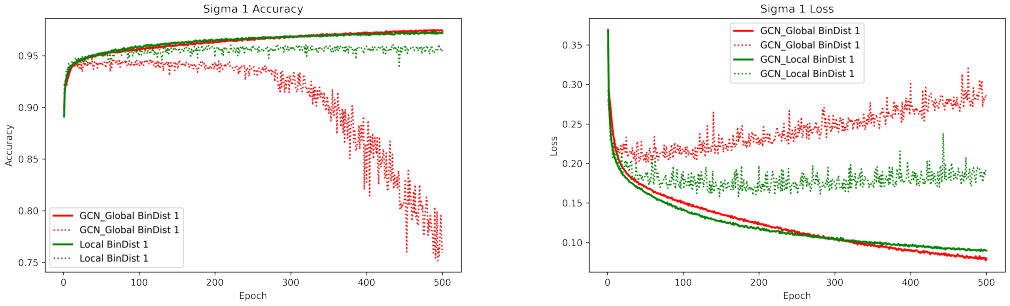


Global Accuracy



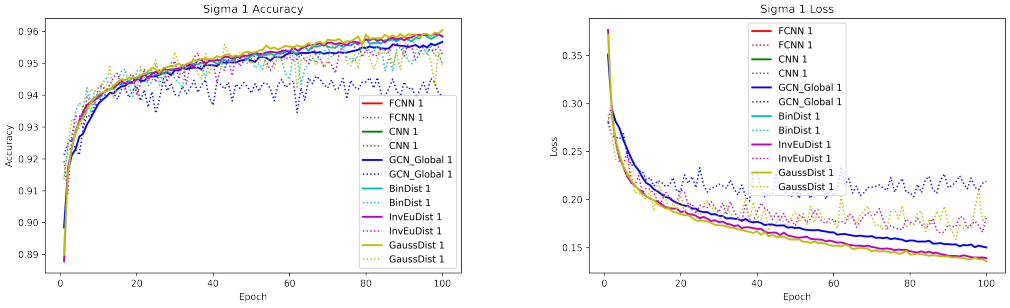
Global Loss

Figure 3.7: Metrics with different global models: Training and Testing curve plotted for an increasing radius from 1 to 27. Each metric value considered at epoch 50.



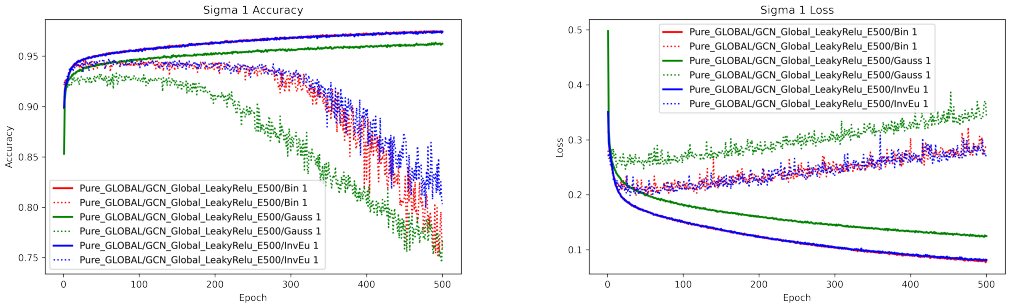
Global Vs Local Acc | Global Vs Local Loss

Figure 3.8: Global Vs Local: plotted metric vs epoch by keeping neighbourhood distance 1.



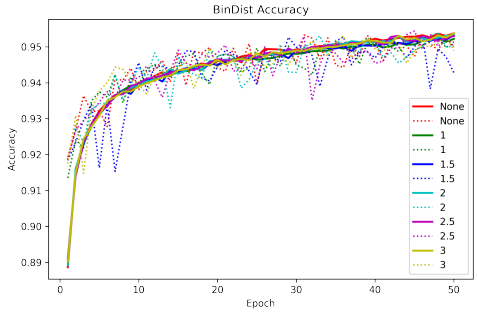
All Global and Local Single sigma Acc | All Global and Local Single sigma Loss

Figure 3.9: All Global and Local Single sigma: Comparision with CNN, FCNN and GNN based Global and Local models. Metric value has considered at epoch 100 with all model distance 1.

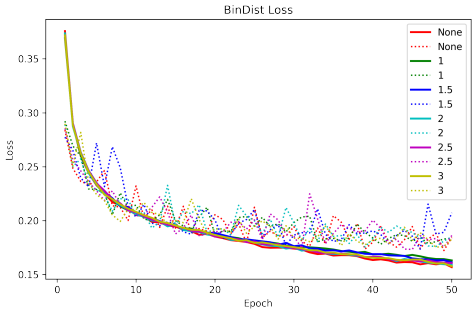


All Global different graphs Acc | All Global different graphs Loss

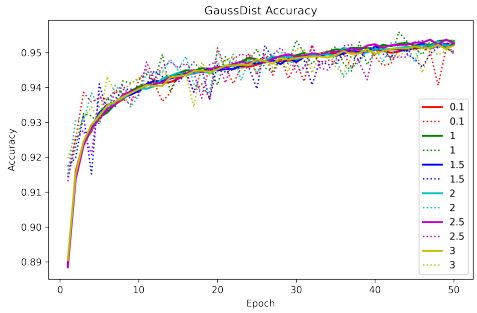
Figure 3.10: All Global different graphs: Training curve of global graph models with same locality distance of 1 for all.



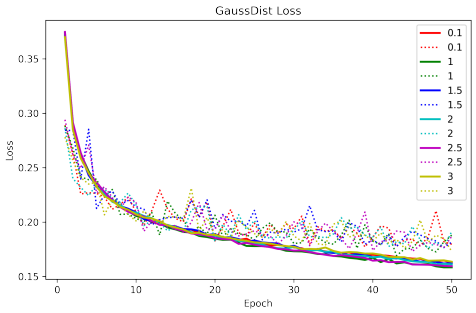
(A.) Local BinDist Training Accuracy



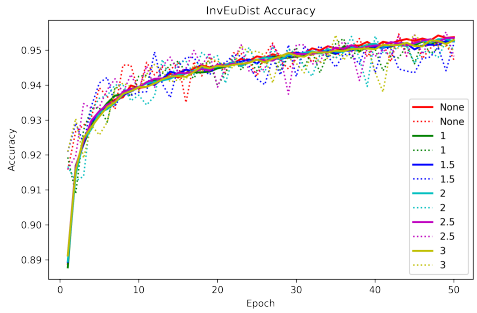
(B.) Local BinDist Training Loss



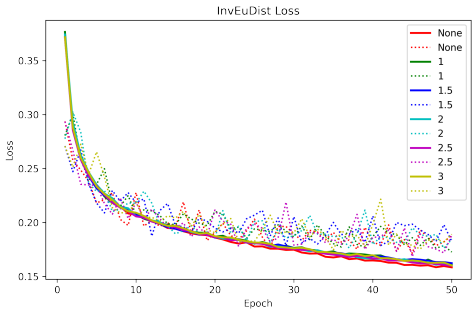
(C.) Local GaussDist Training Accuracy



(D.) Local GaussDist Training Loss

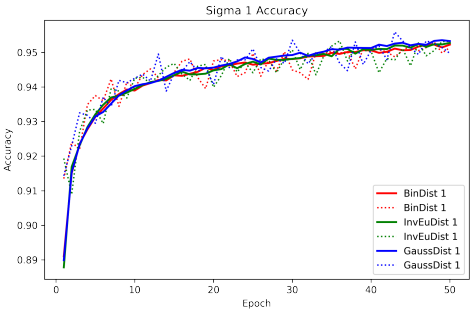


(E.) Local InvEuDist Training Accuracy:FiltSiz-5,R-variable

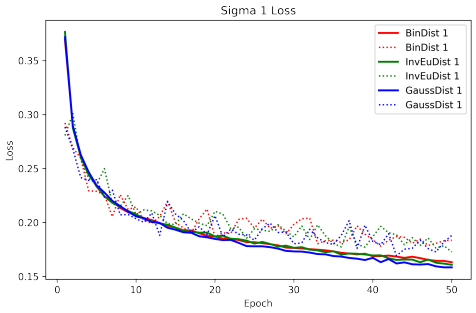


(F.) Local InvEuDist Training Loss

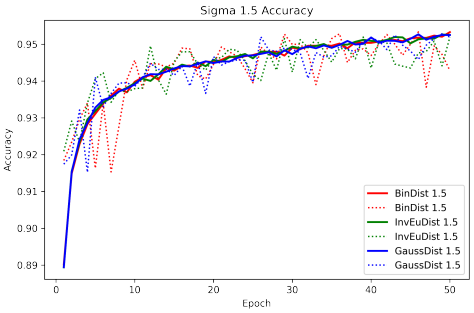
Figure 3.11: Traninig curves for Single model Vs Many sigmas: None is default graph. With local filter size 5, the value of sigmas is standard deviation in Gaussian, whereas in binary and inverse Euclidean graph, it is neighbourhood euclidean distance. Any distance greater than sigmas or neighbourhood distance is 0.



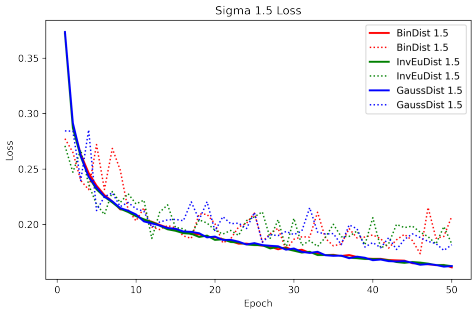
(A.) Local AllGraph Sigms-1 Training Accuracy



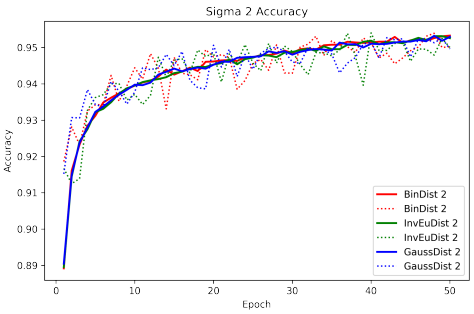
(B.) Local AllGraph Sigms-1 Training Loss



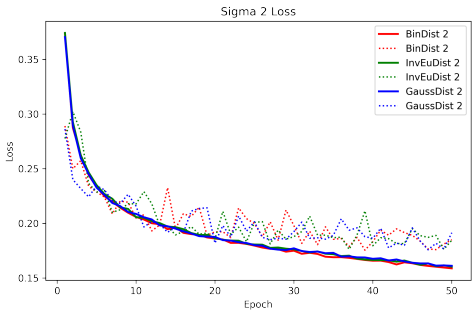
(C.) Local AllGraph Sigms-1.5 Training Accuracy



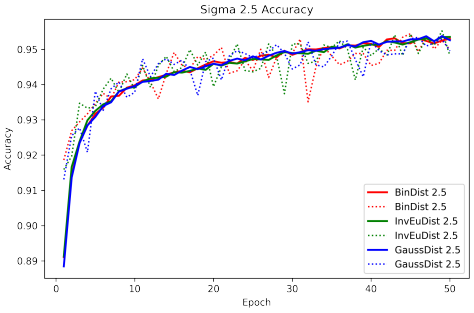
(D.) Local AllGraph Sigms-1.5 Training Loss



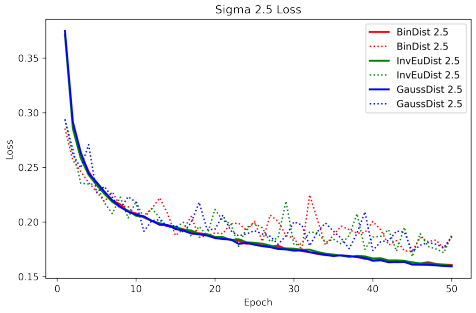
(E.) Local GaussDist Training Accuracy



(F.) Local AllGraph Sigms-2 Training Loss

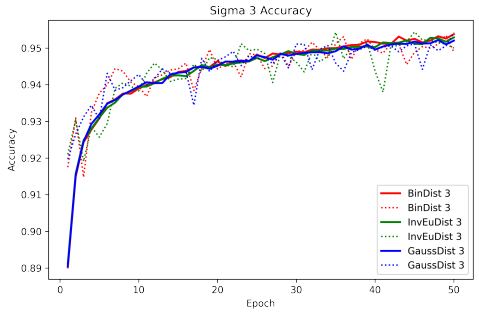


(G.) Local AllGraph Sigms 2.5 Training Accuracy

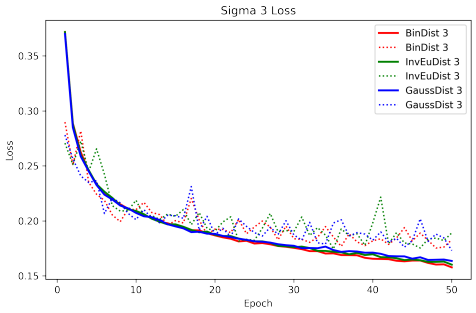


(H.) Local AllGraph Sigms 2.5 Training Loss

Figure 3.12: Training curve of single sigma Vs Many graph: Each curve is plotted to observe the effect of same distance with many graphs.



(I.) Local AllGraph Sigms 3 Training Accuracy



(J.) Local AllGraph Sigms 3 Training Loss

Figure 3.12: ALL Graph Diff Sigmas (continued from previous page)

categorization as well as learning on a localised extracted subgraph, and they demonstrate the capability of learning localised feature mappings throughout a network's numerous layers. The use of a graph global pooling approach that aggregates vertices in the subgraph spatial domain needs further investigation to improve the subgraph partition method. We found that local sampler functions learn localised features more prominently than global image arrangement. We also discovered that such a model is more stable. We further notice that the changing distance in local architecture does not appear to improve, although it does in global architecture, particularly in binary.

By taking into account the learning of localised features by subdividing the entire image data and processing each subdivided data, it may be possible to solve various computer vision-based medical image analysis tasks, such as cell segmentation and cell detection in immunostained histology images. Utilising the benefit of local graph processing, we will investigate the segmentation problem in the following chapters and cell detection in the chapter after that.

Chapter 4

Segmentation in Irregular Domain Data

Contents

4.1 Introduction

Medical image analysis is a critical component of modern healthcare, enabling the detection and diagnosis of diseases at an early stage. However, the accuracy and efficiency of medical image analysis depend on the quality of the segmentation process, which involves dividing an image into meaningful regions or objects. In medical imaging, segmentation is used to identify specific structures or tissues within an image, such as tumours or blood vessels. Due to the limitations of clinical procedures and image analysis, pathologists have difficulty understanding diseases at an early stage. Automated segmentation can play a vital role in improving the accuracy and efficiency of disease diagnosis and early detection. There are several segmentation methods, including conventional and deep learning. We are primarily concentrating on utilising graph-based convolution operations for medical image applications. As demonstrated in Chapter 3, we observed that sampler extraction of local features offers enhanced stability and performance when employing graph convolution techniques. This approach holds promise for improving the accuracy and efficiency of feature learning in biomedical image applications and contributing to the advancement of graph-based deep learning methods.

The research problem addressed in this study pertains to the accurate and efficient analysis of microscopic immunostaining images of Hodgkin lymphoma containing CD4, CD8, and FOXP3 cell staining. Hodgkin lymphoma is a complex malignancy characterised by the presence of specific immune cell markers, namely CD4, CD8, and FOXP3. Identifying and quantifying these cell populations in immunostaining images is crucial for understanding the disease's pathological characteristics and its progression. However, the manual analysis of such images is time-consuming and subject to inter-observer variability. In order to automate the segmentation and quantification of CD4, CD8, and FOXP3 cell staining in the microscopic images and enable more effective and accurate analysis for improved disease diagnosis and research outcomes, the research aims to develop advanced deep learning-based methods and computational techniques.

Graph-based convolution leverages the connectivity information within the graph to compute node-based features, allowing the analysis to be more localised and context-aware. This approach overcomes challenges posed by the irregular distribution of cells in Hodgkin lymphoma images and enables the extraction of relevant information at a finer scale. By integrating graph-based convolution as a sampler in the analysis pipeline, the research aims to enhance the accuracy and efficiency of cell segmentation and quantification, providing valuable insights into the distribution and density of CD4, CD8, and FOXP3 cell staining. Graphs are an effec-

tive representation of such irregular data, where nodes represent individual cells, and edges capture the spatial relationships between neighboring cells. By modeling the image as a graph, the graph-based convolution operation efficiently samples local features, improving the analysis process. With graph convolution, each node is updated based on the information from its neighboring nodes and processed for the convolution operation. This enables the analysis to capture important local patterns and characteristics of the cell distributions, leading to more robust and precise results in the study of Hodgkin lymphoma images containing CD4, CD8, and FOXP3 cell staining.

In this chapter, we propose two graph-based convolution methods for cell segmentation to improve the accuracy and efficiency of disease diagnosis and early detection. Our proposed methods use advanced deep-learning, spectral, and spatial-based graph signal processing approaches to learn features. The proposed methods are motivated by the convolutional neural network U-Net architecture that uses both spectral and spatial-based graph convolutions to learn features from irregular domain data such as immunostained slides. These FCN and U-Net architectures are primarily based upon convolutional neural network (CNN), where a FCN requires more training data and the U-Net is initially designed for biomedical images with few samples with no dense layer. Our contribution in this chapter is to apply a graph convolution operator on the non-Euclidean data and propose a method for segmentation task on biomedical cell image data such as immunostaining images of Hodgkin lymphoma.

Section 4.2 provides a proposed methods of cell segmentation followed by experimentation, results and summery in subsequent section 4.3, 4.4 and 4.5

4.2 Methods

We proposed spectral and spatial-based methods. Spectral-based convolution uses filters from the perspective of graph-signal processing while spatial-based convolution defines graph convolutions by information propagation. In a spectral-based approach, convolution operations are performed in the frequency domain. The matrix representation of the graph is convolved with the features matrix. The result multiply with the weights W^i on each nodes in the i^{th} layer and passed through the hidden layer non-linear function. In conventional CNN, the pooling layer is used to reduce the resolution of input feature map but in the case of a graph, there is no reduction of size due to the multiplication of the filter with spectral signal [127]. To pool local feature output from the convolution layer, it is required to perform graph coarsening which reduces the number of vertices, and handle the edges between these vertices based on

the similar properties. In graph convolution, there is no reduction of vertices, only changes in the output filter channel. But for the precise classification, pooling generalizes features in the spatial domain. Agglomerative pooling is a bottom-up approach to reduce vertices and project the features on a new graph. There are various methods to do graph coarsening such as graclus etc. One of the common methods for selecting vertices is to select a subset of the set of vertices or generate new nodes. Algebraic Multigrid (AMG) is a graph coarsening method which project a signal to a coarser graph by greedy selection of vertices. This method is used as a pooling operation on graph [127]. Spatial-based graph convolution follows the similar approach of convolutional operation of a conventional CNN on an image. Their operation is based on a node's spatial relations. Images are represented as a special form of 2D graph with each pixel representing a node and is directly connected to its nearby pixels. Similar to conventional convolution operation, spatial graph convolution perform convolution operation by considering its neighbours' representations of node and central node.

4.2.1 Proposed network architecture

Motivated by the convolutional neural network U-Net architecture, we propose a similar architecture using spectral based graph convolution and graph pooling. The network architecture is illustrated in Figure 4.1. The encoder part consists of its three blocks, each consisting of three graph convolutions and graph pooling layers. For graph-pooling operation, we utilize AMG coarsening to obtain the restriction and projection matrices. The decoder part starts with the graph up-pooling operation in sequence with the last pooling operation of an encoder. It consists of three layers of graph convolution, each initiated with a graph up-pooling operation. In the case of up-pooling operation, we used the projection matrix of previous coarsened graph to reconstruct original size graph dimension.

In the spatial approach based graph convolution architecture, we have used mixture model CNN (MoNet) framework for the graph convolution operation where each convolution operation is followed by activation. Here we have avoided the pooling operation as the spatial approach used aggregative methods of neighbour node used to learn efficiently large graphs.

4.2.2 Proposed method of utilizing Spectral based Graph-CNN

To perform convolution on the graph, spectral theory is used to define the analogue to convolution, and for the downsampling and upsampling operations, graph coarsening as a pooling layer is defined. While performing the graph pooling, we partition the graph into coarsened graphs

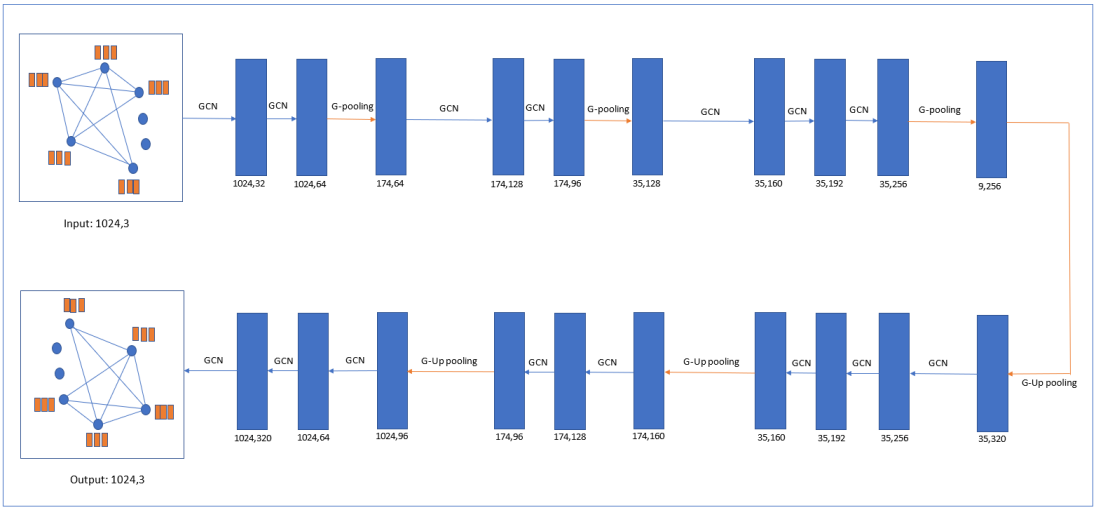


Figure 4.1: GCNN architecture: blue color box represents output result of the graph, processed by operations GCN/G-pooling. It also mentioned the size of the graph: number of nodes and output channels.

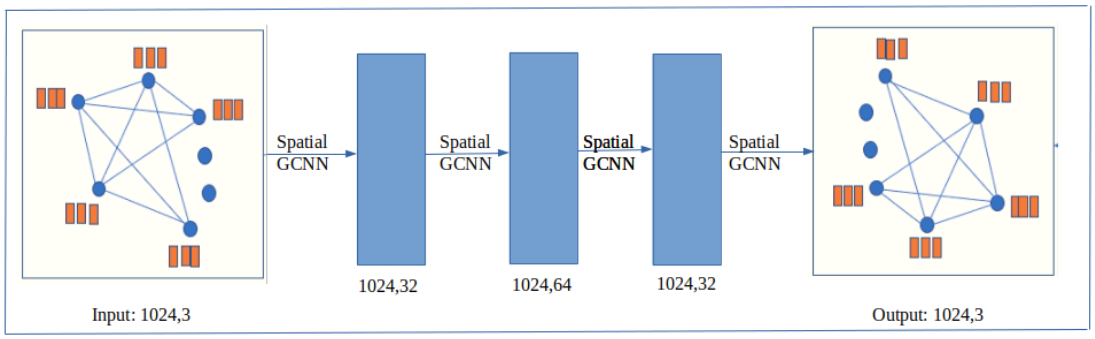


Figure 4.2: Spatial GCNN architecture: blue color box represents output result of the graph, processed by operations spatial GCNN. It also mentioned the size of the graph: number of nodes and output channels.

and use projection and restriction for graph pooling and graph up pooling operation [85, 156]. Training is fed forward through the network to obtain output, and loss propagates backward to update the weights. The graph holds spatial information about the connectivity of nodes and allows graph processing tools, such as convolution and pooling, to operate on signals [12].

To perform the graph convolution, an image is considered a 2D grid graph, having a set of nodes (V), set of weighted edges (E) and and adjacency matrix (A). The graph possesses the property that each node is connected with its neighbouring nodes, which form the basis of locality for the convolution operation. The graph structure of the image represents irregular data, and graph Laplacian is the core operator for the graph convolution layer. The graph

convolution operator used here is spectral-Based GCN, and their operations are described in section 2.3.2.1. However, a simplified expression of the graph convolution operation is explained here. Let N be the number of nodes holding the 3 RGB pixel values at each node, say, in the d dimension. This expression is nicely derived from spectral-base graph convolution [121]. X is a feature matrix of dimension $N \times F^0$, where N is the number of nodes and F^0 is the number of features on each node. The 2D grid graph is created to hold this signal, with the dimension N. The $N \times N$ binary matrix stores the connectivity of the node and is called adjacency matrix, represented by A. Like conventional CNN, the hidden layer in graph-CNN is represented by

$$H^i = f(H^{i-1}, A) \tag{4.1}$$

as H^i is i^{th} hidden layer. For optimisation and training, weight is assigned to edges between connecting nodes as W^i and also considers the self-node features that are added into the adjacency matrix. So, the new adjacency matrix is $\hat{A} = A + I$ where I is an identity matrix. Graph convolution is modified as follows: first compute the node feature representation of each node by aggregating the feature representation of its neighbours, and then transform it by multiplying by the weight matrix. To avoid the gradient exploding, normalise the feature representation by adding the degree matrix D^{-1}. The whole graph convolution is represented as

$$f(H^i, A) = \sigma(D^{0.5} A D^{0.5} H^i W^i) \tag{4.2}$$

where σ is a nonlinear activation function. The Laplacian matrix is decomposed into the orthonormal vector U. This eigendecomposition of the graph Laplacian gives the Fourier mode and graph frequencies. So, the generalised equation of graph convolution is

$$f(H^i, A) = \sigma(U * s) \tag{4.3}$$

where s is a signal on a graph.

For the graph pooling operation, we are using the AMG method to coarsen a graph and projecting signals on a new coarsened graph via a greedy selection of vertices [157]. The two-level coarsening is shown in Figure 4.5. Every AMG coarsened graph provides the restriction matrix R and the projection matrix P for the interpolation of the input signal s. Downsampling operation is performed by multiplication of signal and restriction matrix

$$s^j = s^i * R^i$$

and reverse pooling by multiplication with projection matrix

$$s^i = s^j * P^j$$

Where s^j is the output of the downsampling operation and s^i is the output of upsampling. R^i and P^j are the restriction and projection matrices, respectively.

4.2.3 Proposed method of utilizing Spatial based Graph-CNN

Similar to the conventional CNN on the two-dimensional grid image, spatial-based graph convolution defines the spatial relation between the node and its neighbouring nodes on the graph. Each node is represented as a vertex of the graph, and the value is the signal on that vertex. In the spatial graph convolution network, centre nodes are updated by averaging the neighbouring nodes, analogous to the conventional CNN.

In our case, we are using a spatial-based mixture model CNN for graphs (MoNet). MoNet's use of a Gaussian mixture model (GMM) to learn filters on graphs is one of its key innovations. This increases the interpretability of the filters, as each basis function corresponds to a local pattern in the graph. Each basis function is defined as a Gaussian centred on a certain graph node or pixel. During training, the weight of each basis function is learned, and the number of basis functions is a hyperparameter that can be tuned for optimal performance on the given task. In addition, the use of a GMM enables the model to learn a set of filters that capture various aspects of the graph structure, as opposed to relying on a single global filter. This can result in enhanced performance on tasks like graph classification. MoNet can utilise a predefined set of basis functions that correspond to various levels of image abstraction. One set of basis functions may capture low-level features like edges and corners, while another set may capture higher-level features like object shapes. In this approach, the pixel-neighbourhood relationship can be represented by a pseudo-coordinate. x represents a vertex on a graph, and $y \in N(x)$ are the vertices in the neighbourhood of x. Assign a d dimensional vector of pseudo-coordinate $u(x,y)$. The pseudo-coordinate calculates the degree of nodes by the equation

$$u(x,y) = (1/\sqrt{deg(x)}, 1/\sqrt{deg(y)})^T \tag{4.4}$$

In this coordinate space, parametric learnable Gaussian kernel function is defined as below:

$$W_j(u) = exp(-1/2(u-u_j)^T(\Sigma_j)^{-}1(u-u_j)) \tag{4.5}$$

where Σ_j and u_j are learnable $d \times d$ and $d \times 1$ covariance matrix and mean vector of Gaussian kernel. This Gaussian kernel acts as a basis function.

With the help of these kernel functions, a patch operator is used to perform the function of convolution. This operator applies the Gaussian kernel to each node pseudo-coordinate with all neighbourhoods and summons up the results [124].

$$D_j(x)f = \sum_{y \varepsilon N(x)} w_j(u(x,y))f(y), j = 1,...J, \tag{4.6}$$

The patch operator can be defined by the above equation, Where J represents the dimensionality of the extracted patch. The generalised graph convolution operation is written as

$$(f * g)(x) = \sum_{j=1}^{J} g_j D_j(x)f, \tag{4.7}$$

where g_j is the learnable weight matrix. The output of the convolution process is a set of feature vectors that encapsulate the structure of the graph at a particular level of abstraction. By utilising a set of basis functions, MoNet can capture a wide range of local graph structures, from low-level features such as edges and corners to higher-level features such as object shapes and structures. This enables MoNet to learn highly expressive feature representations, such as image classification.

Some of the advantages of MoNet over spectral-based networks are that it is highly expressive and capable of learning complicated filters that reflect the graph's underlying structure. It is computationally efficient since it permits the sharing of parameters among nodes and is scalable to huge graphs because only local information must be transmitted between surrounding nodes.

4.3 Experimentation

4.3.1 Generation of Hodgkin Lymphoma (Ground Truth) Segmentation

Microscopic immunostaining images of Hodgkin lymphoma are medical images obtained through a staining technique that allows specific immune cell markers to be visualized in tissue samples of Hodgkin lymphoma. Immunostaining is used to identify and highlight certain proteins or markers within the tissue, enabling researchers and pathologists to study the distribution and characteristics of specific immune cell populations in the lymphoma. CD4, CD8, and FOXP3 are markers for different types of immune cells. They play crucial roles in the body's immune response and are commonly used as indicators to identify and quantify specific immune cell populations in tissue samples. Here's a brief explanation of each marker:

1. CD4: CD4 is a protein found on the surface of helper T cells, which are a type of white blood cell. Helper T cells play a central role in coordinating the immune response, help-

ing other immune cells recognise and attack foreign invaders such as bacteria, viruses, and cancer cells.

2. CD8: CD8 is a protein present on cytotoxic T cells, also known as killer T cells. Cytotoxic T cells are responsible for directly attacking and destroying infected or cancerous cells in the body.

3. FOXP3: FOXP3 is a protein marker found on regulatory T cells, which are immune cells responsible for maintaining immune tolerance and preventing excessive immune responses that could damage healthy tissues.

In the context of medical imaging, immunostaining images of Hodgkin lymphoma containing CD4, CD8, and FOXP3 cell staining provide essential information about the presence, distribution, and density of these immune cell populations within the lymphoma tissue. Accurate identification and quantification of these cell populations are critical for understanding the disease's pathological characteristics, its progression, and potential treatment responses. Automated analysis and segmentation of these immunostaining images using advanced deep learning techniques, such as graph-based convolution operations, can significantly improve the accuracy and efficiency of cell identification and quantification. This can lead to more reliable and consistent results in disease diagnosis, prognosis, and research, contributing to improved patient care and medical advancements in Hodgkin lymphoma and potentially other diseases as well.

To create a ground truth generation, we have focused on CD4, CD8, and background (FOXP3). To create a ground truth label for segmentation, a manual labelling process was undertaken, involving the selection of contour points for each cell class. Subsequently, all the pixels within the contours were filled with the unique class ID corresponding to the specific cell type, whether CD4, CD8, or the background (FOXP3). This process results in a ground truth label that aligns with the original image size, where every pixel is assigned a class label, making it a dense labelling technique. The dense labeling approach ensures that each pixel is associated with the appropriate cell class, including the background (FOXP3), enabling accurate segmentation and analysis of CD4, CD8, and FOXP3 cell populations in the immunostaining images of Hodgkin lymphoma.

Further, the labelled data we have created is used for supervised machine learning to learn meaningful features from the data and for deep learning experimentation.

4.3.2 Segmentation using Clustering Method

Clustering is a crucial step in image segmentation, and among the various approaches available, the K-Means algorithm stands out for its popularity and effectiveness. K-Means is a simple yet powerful technique used to cluster pixels based on their colour information. The process involves randomly selecting k initial cluster centres and iteratively updating them to minimise the sum of squared distances between each pixel and its nearest cluster centre. Once the centres converge, pixels are assigned to the closest centre, resulting in distinct segments or regions within the image. Owing to the effectiveness of the K-Means clustering method [158], we have selected it for our data, as shown in Figure 4.3. In our specific application of microscopic immunostaining images of Hodgkin lymphoma, we aim to segment the images into three classes: CD4, CD8, and the background. The number of selected clusters, K, is set to three to accommodate these three classes. By employing K-Means, we can efficiently determine the nearest pixels assigned to random centroids and update these centroids based on the mean of the pixels in their respective clusters. The result of the K-Means clustering process is a segmented image that displays the distribution and spatial arrangement of CD4, CD8, and background elements within the immunostaining images. This segmentation is vital for gaining insights into the pathology of Hodgkin lymphoma, understanding the distribution of specific immune cell markers, and potentially aiding in disease diagnosis and prognosis. However, it is essential to critically consider the advantages and limitations of segmentation based on clustering. One significant benefit is its simplicity and efficiency, allowing for real-time processing of massive datasets. Moreover, clustering-based segmentation does not require prior knowledge or training data, making it valuable in situations where labelled data may be scarce or unavailable.

Despite these advantages, K-Means clustering has its limitations. One notable drawback is its sensitivity to the initial cluster centres or characteristics. Depending on the initial centroids' placement, the segmentation results may vary, and suboptimal placements can lead to poor segmentation. Additionally, clustering-based segmentation can sometimes result in over-segmentation or under-segmentation, where regions are separated into an excessive number of small segments or too few large segments, respectively. These challenges need to be carefully considered and addressed to ensure accurate and meaningful segmentation results.

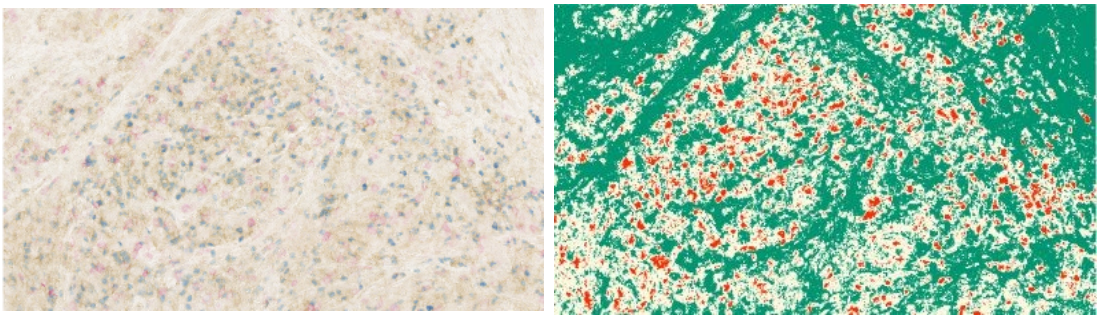


Figure 4.3: k-means clustering of Hodgkin lymphoma immunostaining image. Left: Original image-red colors represent a CD4, purple shows CD8 and rest are background FOXP3 protein., Right: Segmentation via k-means- red colors represent a CD4, yellow shows CD8 and rest are background FOXP3 protein.

4.3.3 Segmentation using Deep Learning

In using deep learning, we used a fully convolutional neural network as one of the state-of-the-art methods for segmentation. We prepared the data and created patches of size 224×224 from a 768×1366 size image. With the supervised deep learning approach, we need data images as well as respective class label masks for each patch.

In this architecture, the encoder contains several layers. Each layer is a combination of convolution followed by pooling operation. At each convolution, the 3×3 kernel convolve with the input image and produce output feature maps. The decoder is used to reconstruct the original image by upsampling and skip connections. In the experimentation, initially 224×224 size patches were fed into the network. The intersection over union (IOU) accuracy was observed with varying accuracy when it was tested on the unseen image. The resulting output obtained from this fine-tuning is shown in Figure 4.4. The choice of hyperparameters such as optimizer, learning rate, and loss function can greatly impact the segmentation model's performance. The optimizer is responsible for updating the model's weights to minimise the loss function during training. The Adam optimizer is a popular option for segmentation problems since it is an adaptive optimisation method that adjusts the learning rate for each parameter using a combination of momentum and RMSprop. It has been demonstrated to converge more quickly and effectively than other optimisation methods, making it a suitable option for segmentation problems using deep learning. The learning rate controls how frequently the optimizer updates the model's weights. A high learning rate can cause the optimizer to overshoot the minimum of the loss function, resulting in instability and slow convergence, whereas a low learning rate can cause the optimizer to become stalled at local minima. A decent starting point for the learning

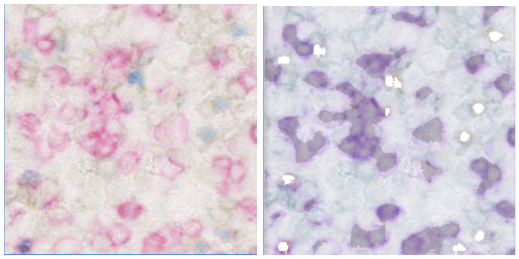


Figure 4.4: Left: Original Image, Right: FCN Fine-Tuning method. CD4 and CD8 can clearly distinguish in FCN results from background.

rate is often around 0.001, and that makes our models a good choice. The pixel-wise cross-entropy loss function is used by FCN to measure the error between the predicted segmentation map and the ground truth segmentation map at each pixel location. We have used a total of 414 patch samples for training and validation.

4.3.4 Segmentation using Spectral Graph-CNN

Due to the resource limitations associated with spectral-based graph convolution, we divided the original image into patches of size $32 \times 32 \times 3$. Random sampling was employed to select a limited number of 1400 patches per image. After assessing the patch sizes, we removed any inappropriate patches to ensure a standardized input size for the graph convolution process. To process this patch using the graph, first we need to construct the graph that holds the signal, which is $32 \times 32 \times 3$ size.

The input signal to the Graph-CNN is a 32×32 patch obtained from a divided histological image. Each node in the graph corresponds to a pixel within the patch, and the node signal is represented by a 1024×3 feature vector, capturing the spectral information of each pixel. The adjacency matrix A encodes the spatial relationships between nodes, defining the edges of the graph. The Graph-CNN learns a representation of this graph, combining both spatial and spectral features of the patch. By processing the graph representation, the model generates a pixel-wise segmentation map as output. This approach allows the model to effectively capture both local and global relationships between pixels, enabling accurate predictions for segmentation tasks. The Graph-CNN excels in handling data with complex structures and textures, making it particularly valuable for identifying objects and regions in an image while recognising their unique characteristics, thus enhancing the accuracy of segmentation predictions.

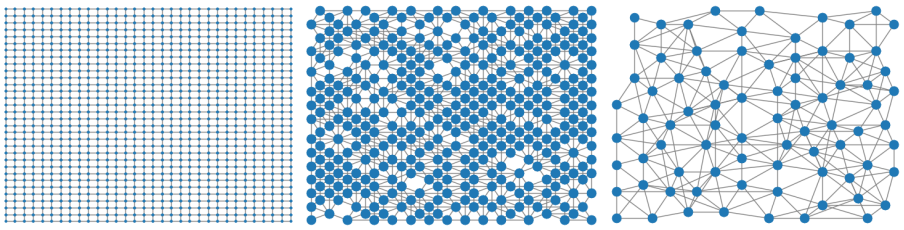


Figure 4.5: Graph representation: Left: Original graph of the 2D grid, Middle: first coarsening level with AMG pooling, Right: second coarsening level with AMG.

4.3.5 Segmentation using Spatial Graph-CNN

In the experimentation with spatial graph CNN, we are using a MoNet operator as described in the method section. We have set the neighbourhood of each vertex to 4 as changing distance in local architecture does not appear to improve as we found in Chapter 3, and based on the Euclidean distance, their respective four adjacent nodes have been collected in sorted order to build an adjacency matrix of k nearest neighbours graph. To perform a Gaussian kernel operation, the coordinate distance between the source and target nodes is used as a pseudo-coordinate. In the operation, apply the Gaussian kernel $w_j(u(x,y))$ over each pseudo-coordinate on the nodes and their neighbours $y \in N(x)$ the result is multiplied with the signal on the neighbour $f(y)$ and summed all the neighbour's results. We have used a Gaussian kernel of size 25 and summed up the result of each Gaussian output patch. All operations are defined by the patch operator. This patch operator is multiplied with the learnable weight matrix to perform a convolution operation, as shown in Equation 4.7.

The architecture used in the model contains four graph convolution blocks with output feature sizes of 32, 64, 32, and 3, giving the segmented output the same dimension as the input. Each layer is followed by the LeakyRelu activation function, except the last layer. In the architecture diagram 4.2, the blue box shows the output graph with the number of nodes and feature size, and the edges describe the convolution operation. For the training, ten thousand random patches from each image size of 1366×768 graph signal, have been collected for 21 such image patches. The data is trained with 5-fold cross-validation with the RMSprop optimizer and a learning rate of 1e-5. We have set the training epoch to 10 with batch size 1, which helps to enhance the functionality of the patch operator and learn the feature better as compared to spectral-based graph convolution, as shown in confusion matrix Figure 4.6.

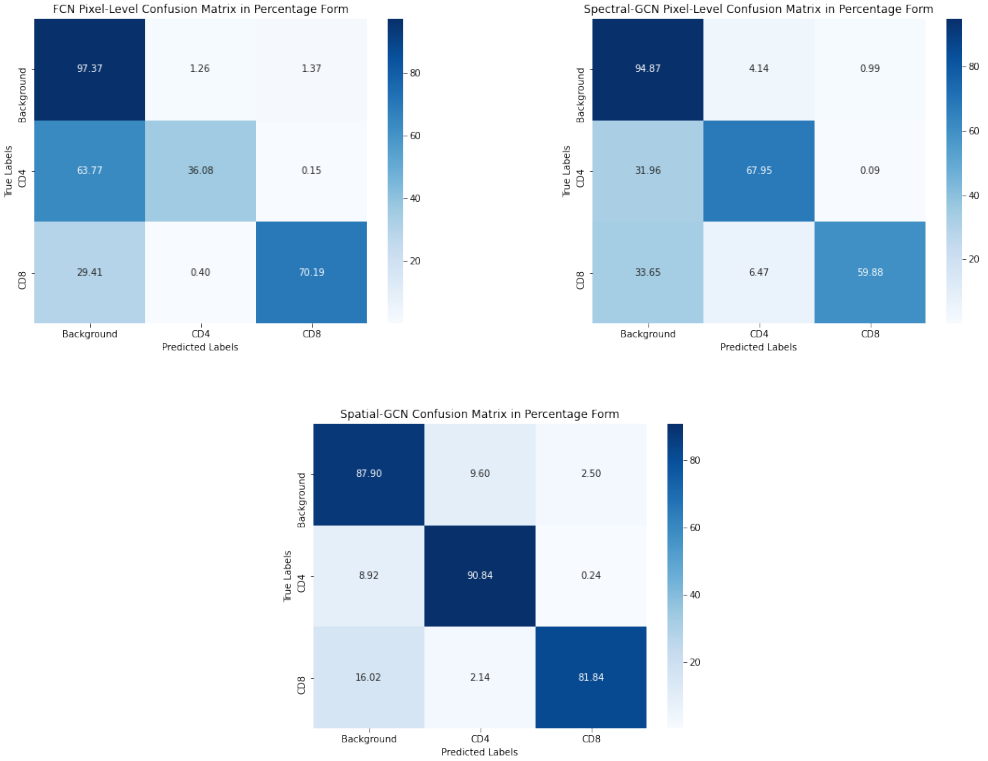


Figure 4.6: Confusion Matrices of segmentation methods. Top left: FCN, Top right: Spectral Graph-CNN, Bottom: Spatial Graph-CNN.

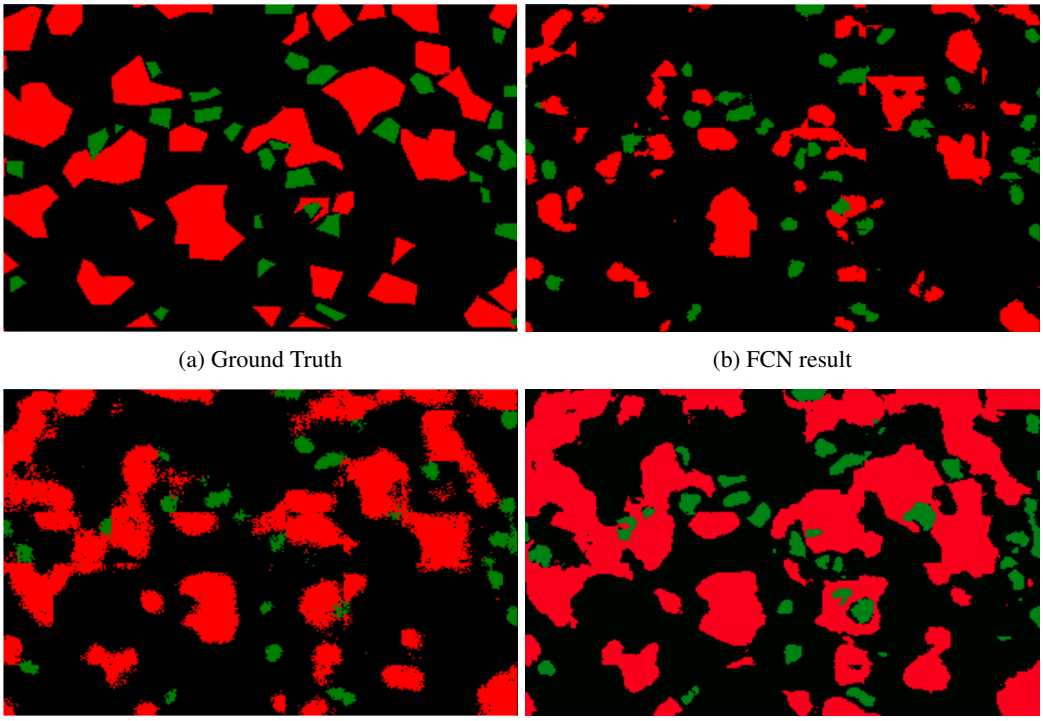
4.4 Results

For training and testing, we have used a total of 23 images of size 1366×768 cell segmentation dataset. For FCN approach training and validation, the data sample is divided into a ratio of 70:30 for the total of 414 patch samples. We report an average accuracy of 87.07% computed over a total of 18 patch samples of size 224×224. Regarding the graph-based approach: due to limitation of resources we used samples of size 32×32 and total number of sample for training and testing is 30800 with 70:30 ratio. The pixel accuracy is taken over 1008 unseen image samples for both spectral and spatial-based approaches, with an improvement of 2.2% and 3.94% compared with the FCN approach, respectively. Their class-based comparative analysis of quantitative measures can be seen in the confusion matrix of Figure 4.6 where both spectral and spatial-based methods show considerable improvement.

The comparative quantitative and qualitative results of the proposed method are shown in Table 4.1 and Figure 4.7. Along with the resultant image, we have also presented the cropped-

Method	Pixel Accuracy (%)
FCN	87.07
(Ours)Spectral Graph-CNN)	89.29
(Ours)Spatial Graph-CNN)	**91.03**

Table 4.1: Quantitative comparison of results.

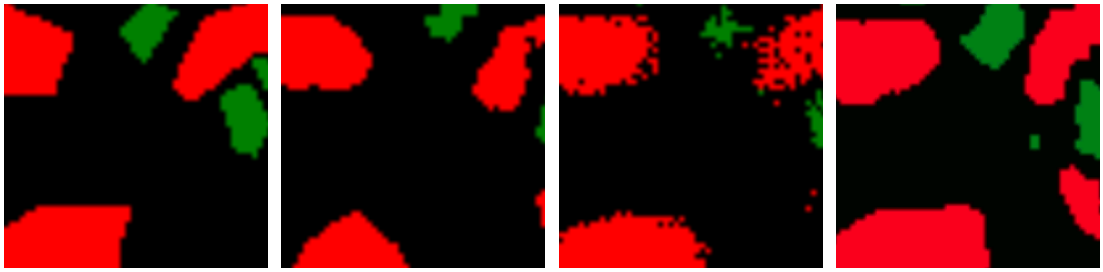


(a) Ground Truth (b) FCN result

(c) Spectral G-CNN result (d) Spatial G-CNN result

Figure 4.7: Qualitative Comparison of Results. (a): Ground Truth of original image of different samples, (b): Result obtained by FCN method, (c): Result obtained by G-CNN method, (d):Result obtained by Spatial G-CNN method. Red color represent the CD4 stain cells and Green color corresponds to CD8.

in region to show off the localization and segmentation in Figure 4.8. It is observed that the size feature of the cell is better represented by a graph-based approach with an improved result. It was also observed that spectral-based methods work best with small graphs, while spatial-based methods perform better with large graphs.



Ground Truth | FCN result | Spectral G-CNN | Spatial G-CNN

Figure 4.8: Cropped-in region to show off the localization and segmentation.

4.5 Summary

This study proposes a novel method of performing segmentation on cell images using spectral and spatial graph convolutional networks. It also allows patch-wise distribution of the original image for better feature learning. Convolutions are performed in the spectral domain of the graph Laplacian for learning of spatially localised features. Spatial-based graph convolution handles different graphs to learn locally at each node. Results are provided on both conventional CNN and graph-based CNN which shows graph-based CNN, has the ability to learn localised feature maps across multiple layers of a network. We have also experimented with these data with different traditional methods like clustering, thresholding, and deep learning methods such as FCN, spectral, and spatial-based graph convolution. In observation, we have found that graph convolution networks improve segmentation results.

Chapter 5

Cell Detection in Irregular Domain Data

Contents

5.1 Introduction

In this chapter, we present a novel approach for cell detection in non-Euclidean data using graph convolution networks (GCNs). GCNs have become increasingly popular in recent years due to their ability to handle complex data structures such as graphs and networks. Due to limitations of clinical methods such as noisy imaging and ambiguity in the data, the detection of different objects in the cell image helps medical examiners automate their clinical work. There are many methods developed to process Euclidean data for an object detection task in conventional convolutional neural networks (CNN), but non-Euclidean data requires attention in this regard.

In computer vision, object detection combines the tasks of classification and localisation. Classification predicts the class label of the object in the image, while localisation locates the object in the image. Various efficient techniques have been developed for object detection in the context of Euclidean data, leveraging the capabilities of convolutional neural networks. Noteworthy examples encompass RCNN, fast-RCNN, faster-RCNN, YOLO, etc [6, 159–161]. However, the exploration of spatial-based graph convolution networks was initiated as early as 2009 [162], and since then, a plethora of methods have emerged [122, 125, 163]. By harnessing the graph structure and node-specific information, these networks prove versatile in performing diverse computer vision tasks, including node-level and graph-level classification [20]. This compelling capability serves as motivation to delve into the arena of object detection within the non-Euclidean domain.

The prospect of employing a graph neural network for object detection holds immense potential, although its applicability has predominantly been explored within point cloud contexts [164]. When dealing with images represented as 2D grid graphs, the challenge lies in deciphering the spatial relationships of objects from the sparse data distribution. In addressing this challenge, we present a novel algorithm that retrieves object proposals in the form of subgraphs from the original 2D graph. Our approach involves a generalised feature extraction technique tailored for non-Euclidean data graphs. This study marks the pioneering instance of extracting object proposals as subgraphs from sparse data for the purpose of object detection. Our focus encompasses two key challenges: firstly, devising a mechanism for object localization through graph proposal extraction using a graph convolution network; and secondly, training a model in the presence of highly unbalanced and noisy detection data. While previous works by Szegedy et al. proposed localization as a regression problem, the sliding-window detector approach exhibited superior performance [159, 165]. Convolutional neural networks

have been effectively employing the sliding-window technique for over two decades, and we are contemplating its application within our system.

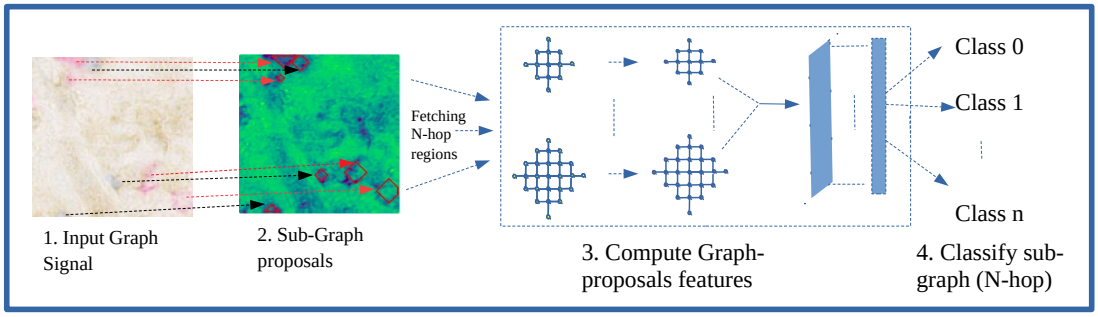
In this chapter, we propose a graph convolution-based region proposal mechanism for object detection in non-Euclidean data. We mainly focus on the extraction of a subgraph as a candidate for the prospective object region, the classification of the object candidate into corresponding class labels, and the detection mechanism. This chapter is divided into sections: Section 5.2 elaborates on the method, including the graph proposal algorithm, Section 5.3 provides details about the data we have used and explains the experimentation and results. In the last section, we conclude the findings of the chapter with a summary in Section 5.5.

5.2 Method

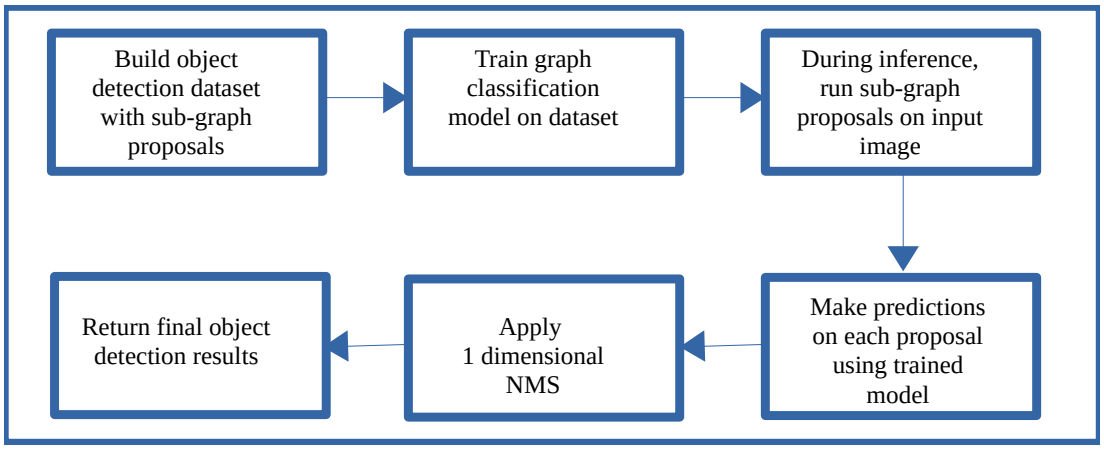
Our object detection pipeline consists of training and inference architectures. Figure 5.1 shows the system overview, where the training architecture is divided into four parts. In the first part, the input graph signal is a 1D vector representation of a 2D image. In the second, the subgraph proposal mechanism outputs sub-graph proposals corresponding to object and non-object regions. Next, the graph convolution network computes the features from the sub-graph proposals. Lastly, a softmax classifier is added to classify subgraphs/graph proposals into corresponding class labels.

In inference architecture, the terms "N-hop proposals" and "N-hop anchors" are used interchangeably. N-hop is the travel distance from the source node to the destination node. N-hop proposals are subgraphs fetched in a sliding window fashion from the 2D graph with the help of a breadth-first-search algorithm, and each fetched N-hop proposal has an arbitrary node size with the distance "N-hop". Next, all the N-hop proposals are passed to the trained classifier. The output N-hop proposals are referred to as bounding box proposals that qualify the threshold. Furthermore, non-maximum suppression is created to remove unwanted bounding box proposals. Finally, the object bounding box is produced.

Our object detection system consists of three main modules. The first is a graph proposal mechanism that produces the subgraph, containing objects and background graph proposals. The second module is a graph convolution network that helps extract the features from the graph proposal, and the third is a classification network where graph features are classified into related labels. We called our system 'Object Detection with Graph Proposal Neural Networks' (**G_p-NN**), because of the use of a graph proposal mechanism for an object detection task.



Training Architecture: From left to right, 1.Input graph signal on the 2d grid graph, 2.Fetch N-hop sub-graphs from input graph signal and apply graph proposal algorithm 1 & 2, 3.Extracted graph proposals passed through the graph convolution network(use GAT) for feature learning. 4. Softmax classifier to classify into object class



Inference and Detection Pipeline: From left to right: Building dataset for the training purpose using sub-graph/graph proposal mechanism, Used graph convolution networks for classification on this data, During inference time, extract graph proposals in sliding window fashion for classification. From right to left: Pass each proposals through trained model for prediction, Non-Maximum Suppression- Positive detected proposals passed through NMS to remove unwanted bbox, Desired object bounding box as final result

Figure 5.1: $\mathbf{G_p}$**-NN** Object detection pipeline

5.2.1 Graph proposal (G_p-NN)

There are many ways to find the proposals and use them for classification in conventional CNN as well as in the graph-based approach. In conventional CNN-based object detection, R-CNN and fast R-CNN use a selective search proposal method. There are other methods as well; for example, Cireşan et al. apply CNN on regularly-spaced square crops for the proposals [166]. In the case of a graph-based approach, graph matching and frequent subgraph mining are used to extract the subgraph as a feature for classification [167]. The general idea is to represent the regions of an image having similar properties by graph vertices and their relations between vertices by edges. Quad-trees represent the image in the form of tree-like structures [168], and Niusvel et al. used them for graph classification [167].

In our graph proposal approach, we consider the image as a 2D grid graph, with each vertex representing a part of structural and topological information. To extract the graph proposals, we used the Breadth-First-Search-based approach. In the graph domain, vertices are scattered and do not have any spatial relationship, such as in Euclidean data. Hence, the challenge of selecting the vertices that belong to a similar feature object can be solved with the help of Breadth-First-Search with a chosen N-hope distance. The Algorithms 1 and 2 explain the mechanism of graph proposal extraction. The input signal S, graph structure $G = (V, E)$ and N-Hop size are the essential elements of the algorithm. We created binary masks of each object within the image with a centre coordinate. To find the size of the object, we considered the varying N-hop distances from the centre coordinate node. The ground truth label is used in the same manner as in Section 4.3.1 for the segmentation tasks, and then binary masks are created for each object class from that ground truth label.

To generate a graph proposal, we have vectorised the 2D grid image. Each node is considered to have "k" number of N-HOP anchors as fetched in Algorithm 1, so the total number of N-HOP anchors will be the number of nodes times "k" ($N \times k$). Each anchor is the result of Breadth-First-Search and is a collection of neighbouring nodes that provide feature information. $N \times k$ anchors were compared with the ground truth binary masks. These binary mask nodes are the ground truth object representations of the corresponding image, and $N \times k$ anchors are all possible object proposals for the image. The possible object proposals that passed the threshold value were categorised into object graph proposals and background graph proposals, as mentioned in the Algorithm 2.

Algorithm 1: BFS (Breadth First Search for Graph)

Require: i: node to be search, $G = (V, E)$, N-HOP: max-nhop
  $N_{list} = []$
  while $j \leq len(N - HOP)$ **do**
    Apply BFS node searching with N-HOP length j and return all nodes n.
    Append nodes n into $N_{list}, N_{list} =\leftarrow n$
  end while
  return N_{list}

Algorithm 2: Graph Proposals

Require: S: signal set, $G = (V, E)$, V: vertices set, E: edges set, N-HOP: max-nhop
Ensure: $len(S) = len(V)$
  j = range [2, len(N-HOP)]: *N-hop anchor size*
  Create Label mask of Ground truth
  Create binary masks B_M of each object class from Ground truth
  for $i \leq len(V)$ **do**
    N_{list_i} = BFS(i, G, N-HOP)
    for $k \leq len(B_M)$ **do**
      if $IOU_{Overlap}((B_{M_k}, N_{list_i})) >= 0.70$ **then**
        Object Graph proposal = $S[N_{list_i{}^j}]$
      else
        Background Graph proposal
      end if
    end for
  end for

5.2.2 Multi-label graph classification network

Both the object graph proposals and background graph proposals will be referred to as graph proposals. These graph proposals have different node sizes carrying arbitrary topology. To remove translation invariance, we have transformed the coordinate system and considered the subgraph as a new graph with a different index-based coordinate system, as shown in figure 5.2. In our experimentation, we found this transformation helps Graph Attention Networks learn features better.

The feature extraction architecture used for the graph classification contains two Graph Attention Networks (GAT), followed by global attention pooling. The output of pooling was given to the last layer as soft max activation. The GAT used here operates on graphs. 2-D grid graphs have a Euclidean structure, but the attention mechanism learns weights based on local

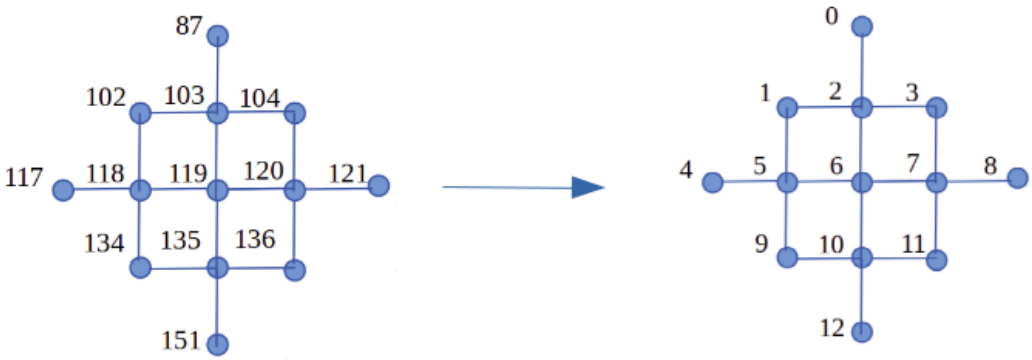


Figure 5.2: Left- Feature node graph of n-hop 2, Right- mapped graph of n-hop 2.

connectivity rather than a fixed distance matrix. Consequently, GAT favours representation regardless of Euclidean structure. In the GAT, graph structure inputs from the masked attention help to decide which neighbouring nodes take part in the feature learning and are to be decided from the feature itself, which makes this operator powerful in object detection [117,169]. A binary adjacency matrix, which represents the neighbourhood connections and graph structures, is created, helping to learn the attention coefficient from the features.

In global attention pooling [170], the varying sized graph features are made into unique sizes with the help of global aggregation of all the nodes in the graph, where each node is computed with a feature weight and an attention weight as shown below in the equation.

$$X' = \sum_{i=1}^{N} (\sigma(XW_1 + b_1) \odot (XW_2 + b_2))_i \tag{5.1}$$

where σ is the sigmoid activation function, X, W and b are the input signal, weight and bias values. This pooling layer helps to learn the graph features with varying sizes and shapes.

5.2.3 Classifier

The softmax classifier was used to assign a probability to each possible label for the graph. The output of the graph convolutional network was a vector, which was then passed through the softmax function to obtain a probability distribution over the possible labels. In this third module, we add the softmax activation layer to assign probabilities to each extracted feature from the graph convolution network, calculating the loss with the corresponding labels, and backward the propogation.

5.3 Experimentation

5.3.1 Dataset

We have used immunostained images of lymphoma cancer patients blood cells as a dataset for the experimentation to evaluate the proposed method.

In immunostained images, we focused on two proteins: cluster of differentiation 4 (CD4) and cluster of differentiation 8 (CD8). The dimensions of each image are $1366 \times 768 \times 3$ (width, height, and channel) pixels, which has a hardware resource limitation due to processing a huge number of node signals over the graph. We have divided the original size of the images into $128 \times 128 \times 3$ patches. We use the term images for $128 \times 128 \times 3$ patches. The images are manually annotated for the object detection task. Annotated label images contain object masks where all pixels belong to the object classes represented by unique labels. Furthermore, a binary mask for each object belonging to the same image is created. These label masks and images are utilised for possible object proposal creation in the graph proposal mechanism.

5.3.2 Training and Inference

We have used 100944 samples of arbitrary-sized graph signals from 40 Immunostained cell images and split them into an 80:20 ratio. For each node, there are 17 bbox anchors, and each anchor has a varying N-Hop size from 3 to 20. N-HOP anchors that belong to the proposed region anchors that contain the object. We have collected such region proposals that belong to the respective classes and non-region proposals of equal size. Trained the model one sample at a time due to the arbitrary nature of the graph size until early stopping at 75 epochs. The RGB pixel values of the signal were normalised in the range of [0-1], and focal loss was used for the training to take care of the imbalanced data. The Adam optimiser uses the default parameter and learning rate set to $5e^{-5}$.

Method	Prec@0.10	Prec@0.30	Prec@IOU.50	Prec@0.70	Average
R-CNN	0.60	0.42	0.34	0.078	0.3595
Our Methods(Gp-NN)	0.89	0.69	0.27	0.10	0.4875

Table 5.1: Precision Quantitative results of cell detection.

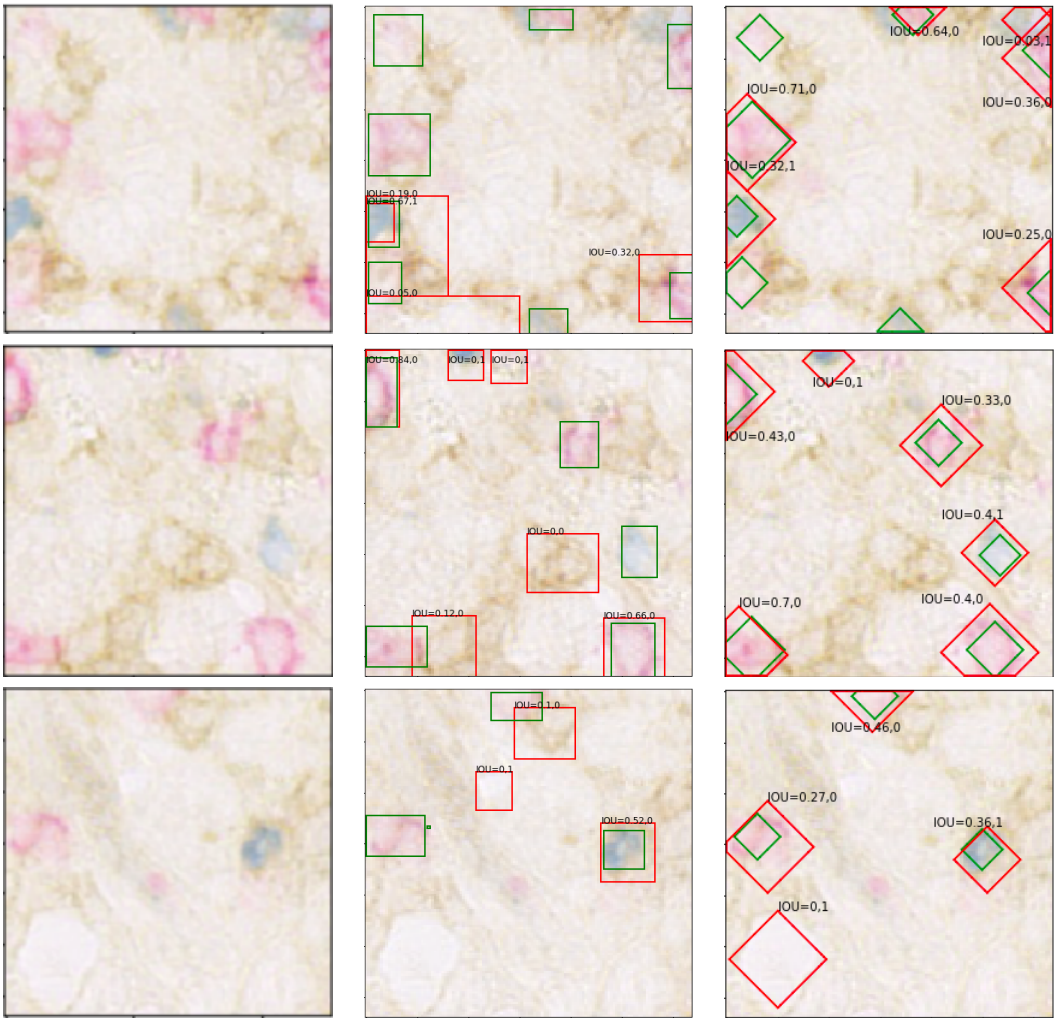


Figure 5.3: Qualitatively results: Left- Original images, Middle- detection results of R-CNN, Right- detection results our method (Gp-NN). Ground truth bbox represented by green colour where predicted bbox by red colour. Each predicted bbox shows an intersection over union score with class id seperated by class commas.

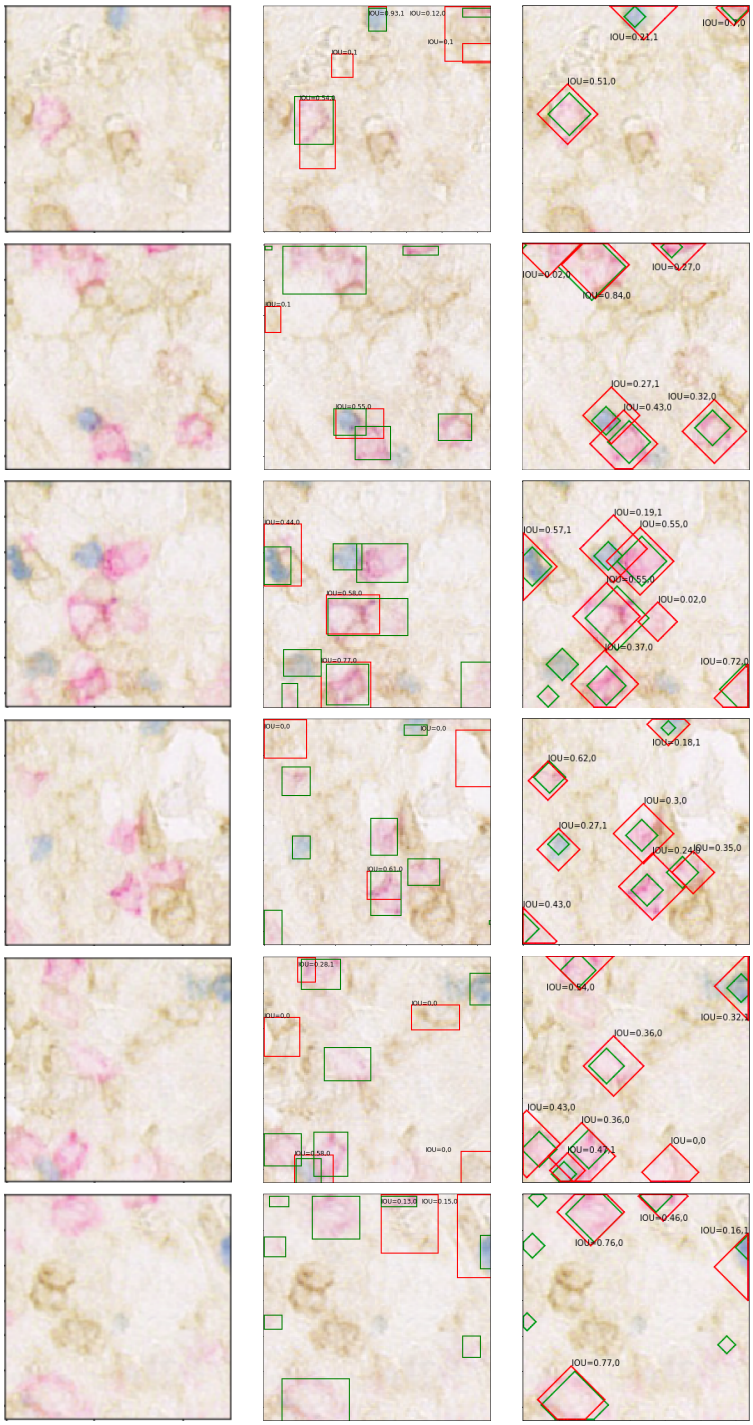


Figure 5.4: More examples: Ground truth and predicted bbox continue

Method	Recall@0.10	Recall@0.30	Recall@IOU.50	Recall@0.70	Average
R-CNN	0.37	0.29	0.25	0.0714	0.2453
Our Methods(Gp-NN)	0.79	0.74	0.53	0.31	0.5925

Table 5.2: Recall Quantitative results of cell detection.

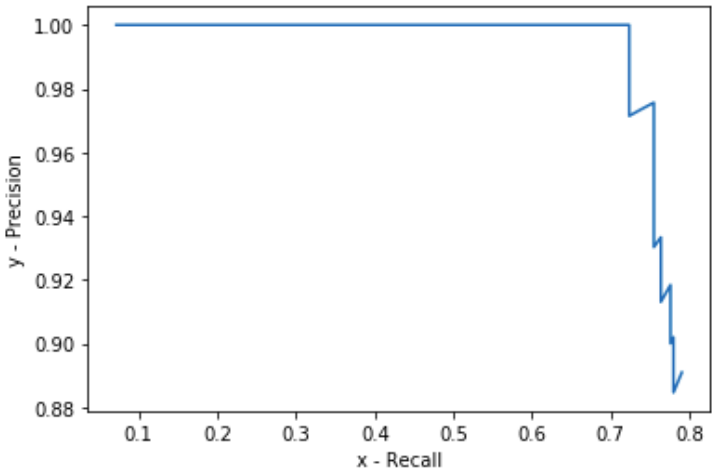


Figure 5.5: Precision recall curve for our method $(G_p - NN)$ with IOU threshold 0.10

In the inference time, the trained model was used for the object detection task. The learned model predicts the respective class of the graph. In the setup, we used the sliding window detector. For each node, we selected varying N-hop sizes, the respective feature signal vector, and the binary adjacency matrix. Pass this attribute to the feature extraction networks, and it outputs the classification prediction. If the objectness score is greater than the threshold, then the bounding box at the location of the node must be passed. Using non maximum suppression for the 1D graph signal, we have removed the unwanted bounding box.

5.4 Results

The visual representation of our results can be observed in Figure 5.3, with additional examples provided in Figure 5.4, giving insight into the effectiveness of our approach. Delving deeper into the quantitative evaluation, the average precision and recall scores, presented in Tables 5.1 and 5.2, shed light on the performance of our object detection method across various Intersection over Union (IOU) thresholds. Notably, our primary focus lies in precise object localization

rather than the tightness of bounding box tightness, as captured by the IOU score. To accomplish this, we strategically set the IOU threshold at 0.10, ensuring accurate true positives.

The corresponding precision-recall curve, illustrated in Figure 5.5, further illuminates the performance characteristics of our method. It's imperative to consider the complexity inherent in immunostained cell images. The challenge arises from the intricate task of annotating precise cell boundaries due to the scattered and potentially ambiguous nature of stained cell colours. Despite these complexities, our methodology underwent rigorous testing with varying IOUs to accurately calculate the Average Precision and Recall scores, ultimately highlighting the effectiveness of our approach.

In a comparative analysis against the conventional CNN-based Region Proposal Network (R-CNN) detector, our method showcases a substantial enhancement in both precision and recall scores by 12.80% and 34.72%, respectively. This notable enhancement underlines the competence of the graph-based proposal extraction mechanism within the contemporary object detection paradigm. Moreover, it hints at the potential to extend the applicability of this mechanism to new, irregular domain data.

5.5 Summary

This chapter proposes a graph proposal neural networks for cell detection in immunostained Hodgkin Lymphoma histology images. We demonstrate that our method of simple object detection for non-Euclidean data can be compared to CNNs-based object detection. We achieved an average precision score by proposing a graph proposal mechanism, multi-label graph classification, and training on scarce data. These results were achieved at a high computational cost due to the use of BFS algorithms. Firstly, the use of a graph traversal algorithm for each proposal and, secondly, online training of the arbitrary-size graphs. In our future work, we will focus on cost reduction and expand this method to a large number of classes.

Chapter 6

Nuclei Detection in Irregular Domain Data

Contents

In this chapter, we are addressing the nuclei detection problem in histology images of adenocarcinoma using irregular domain deep learning methods. Nuclei detection in histopathology images of cancerous tissue stained with standard hematoxylin and eosin stain is a challenging task due to the complexity and diversity of cell data. Recent advances in deep learning have exhibited significant promise in the domain of nuclei detection, predominantly emphasising classification and regression-based methodologies. Notably, contemporary research underscores the potency of regression-based approaches, which have demonstrated enhanced performance compared to their classification counterparts. Nonetheless, it is imperative to highlight a crucial gap within this context – the necessity to address classification within the framework of graph convolution. In light of the unique challenges posed by nuclei detection, especially in irregular and complex biological contexts, devising effective classification strategies within the graph convolution domain emerges as a critical focal point. In this chapter, we propose a graph convolution-based classification model to perform nuclei classification, and such models are used in cascaded architecture to perform nuclei detection. We have evaluated the CNN and GCN-based approaches on a large dataset of cancer histology images with almost 29,000 annotated nuclei. Our results show that graph convolutions show more stability than CNN and improved performance with cascading GCN architecture. We have compared our two-fold quantitative evaluation results with CNN-based models such as centre-of-pixel (CP-CNN) and spatial constrained convolutional neural network (SC-CNN). To observe the power of CNN and GCN operators, we used two different loss functions, such as binary cross-entropy and focal loss function, and explored their behaviour on CP-CNN and GCN models.

The chapter unfolds through various sections. Section 6.1 provides the introduction, and section 6.2 elaborates on the method, including the graph convolution architecture and cascaded pipeline, Section 6.3 provides details about the data we have used and explains the experimentation, followed by results in section 6.4. In the last section, we conclude the findings of the chapter with a summary in section 6.5.

6.1 Introduction

Nuclei detection in histology images is an important parameter for many biomedical image analysis tasks. Due to the varying sizes and shapes of the nuclei, detecting accurate nuclei becomes a challenging task. In the past, many methods have been proposed that can be categorised into classification and regression approaches. Due to the increased popularity of graph convolution networks, solving this challenging task in an irregular domain is important, which

gives the possibility and power of graph convolution over critical data.

The tumour is the result of highly uncontrollable cell growth and death. This heterogeneity in cells evokes an inflammatory response, angiogenesis, and tumour necrosis in tumour development [171, 172]. The location, sizes, arrangement, and placements of these heterogeneous cell types also show the different stages of the cancer level [173,174]. Therefore, the qualitative and quantitative analysis of nuclei helps to better understand the condition of the tumour and explore the different options for various cancer treatments. Pathologists use different colour markers and stains to understand the many insights and properties of the cancer tissues. However, it requires biological experts understanding of tumours to point out informative markers, and it is expensive to carry out work in the laboratory every time due to the availability of cell data [175]. Developing an automated system to do this task of nuclei detection would be a more efficient way to not only save laboratory time but also provide an effective analysis of the image to help biological experts understand the different conditions of the tissue cell.

There are many factors that affect the precise automation of the nuclei detection, mainly due to the noise and poor staining during the preparation process of the image slides. Disarrangement of nuclei, diversity of nuclear morphology, and complex tissue structure create a challenging task for computer vision researchers to automate and analyse. In addition, different types of cells often have irregular chromatin textures and seem to largely overlap each other without any visible boundary, which makes the detection of individual nuclei a challenging task. Some of the different types of nuclei appear to be the same type of nuclei, complicating the challenge of nuclei detection automation [176].

However, there are many methods for cell detection that have been developed based on classification, regression, and the traditional approach, for example, thresholding, region-growing, k-mean, etc. Sirinukunwattana et al. proposed a regression-based approach to train the convolutional neural network model to predict the centre coordinate of the nuclei used in the probability map carried out by post-processing [176]. In this method, the spatial constrained layer is used for regression-based nuclei centre coordinate prediction, and the parameter estimation layer is used to create a probability map. This work is influenced by the conventional method for object detection, centre-of-the-pixel CNN (CP-CNN), where each path gives the probability of being the centre of nuclei or not, and second, structural regression (SR-CNN) [177], where each patch is regressed instead of a single pixel. Veta et al. proposed techniques for nucleus detection in routine Hematoxylin and eosin (H&E) histology images that rely on morphological features such as symmetry and stability of the nuclear region to identify nuclei and

on the direction of the gradient to identify the centre of nuclei [178]. Cosattoet al. [179] use the difference of Gaussian (DoG) and Hough transforms to find symmetric shapes of cells for nuclei detection.

In a more specific context, various methodologies have been explored, particularly the classification approach for cell detection. One prominent avenue involves the utilisation of morphological features in H&E stained breast cancer images. Within this framework, the task entails classifying different cell types, such as cancer cells, lymphocytes, or stromal cells. This classification endeavour is intricately linked with the prerequisite segmentation of cell nuclei [180]. In a parallel vein, the work of Malon et al. [181] stands out. Their study employed the power of a CNN classifier to discern between mitotic and non-mitotic cells. This classification strategy entailed integrating colour, texture, and shape information. Similarly, Nguyen et al. [182] explored classification, with their approach grounded in the nuanced textures and appearances of nuclei. A noteworthy perspective shift is offered by Cruz et al. [183]. Their study resoundingly demonstrated the superior efficacy of deep learning approaches over traditional predefined feature sets and canonical representations. In contrast, Wang et al. [184] introduced a novel approach involving cascaded classification aimed at detecting mitotic cells. The motivation to use a graph sampling approach arose from the confluence of two key factors: the critical importance of addressing classification within the context of graph convolution and the benefits of a cascading architecture for filtering unprocessed samples. The utilization of a graph-based sampling technique has demonstrated stability and enhancement in the previous chapters, further prompting an exploration of its effectiveness in handling more intricate data and assessing its efficiency in sample classification architecture. Due to the graph sampling approach's emphasis on local regions, it excels at extracting positive samples. Notably, the employment of spectral-based graph convolution yields effectiveness with small sample sizes and focuses on structural aspects, thereby aiding in noise reduction during the learning process.

The problem of nuclei detection consists of finding a set of centroid coordinates of nuclei from a given input RGB image I. This problem is solved by a graph convolution-based deep learning supervised approach where the detector is trained on training samples with ground truth information about centroid coordinates. Each pixel is categorised into nuclei or non-nuclei classes. Our detector is a GCN-based pixel classifier. All the pixels in each training sample are assigned a class nuclei where the ground-truth pixel p is centred on the training sample and a non-nuclei class where the centred pixel does not fall into the proximity of ground-truth nuclei centred with euclidean distance d. The GCN network predicts the class

of the raw RGB image value that falls into the sample patch.

In the next section 6.2, we explain the Method of graph convolution working and building cascaded architecture using GCN for detection.

6.2 Methods

6.2.1 Detecting Nuclei with GCN

The architecture of the network is shown in Figure 6.1. It contains two graph convolution networks, followed by two fully connected layers. The RGB pixel values of each sample are passed through the graph convolution operator to extract the features. The extracted feature vector is classified by fully connected layers. The last layer is a single unit dense classification layer with sigmoid non-linearity. Dropout and ReLU non-linear activation functions are used before each dense layer to control the overfitting problem.

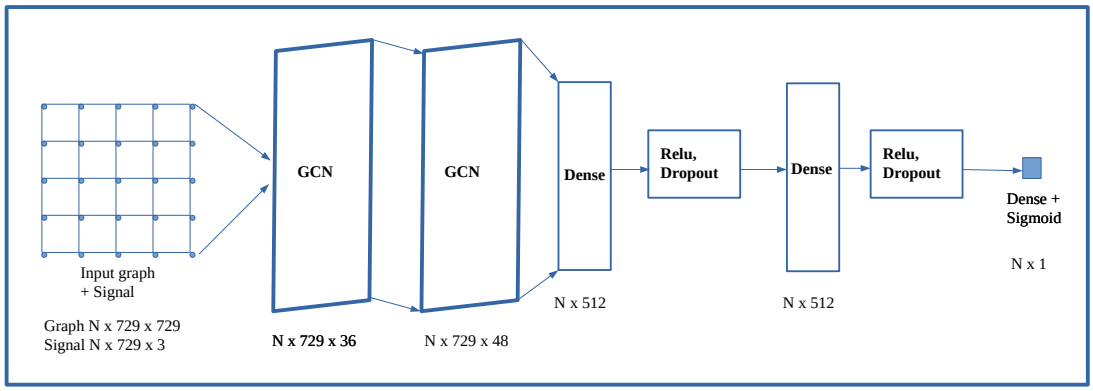


Figure 6.1: Graph convolution architecture for classification.

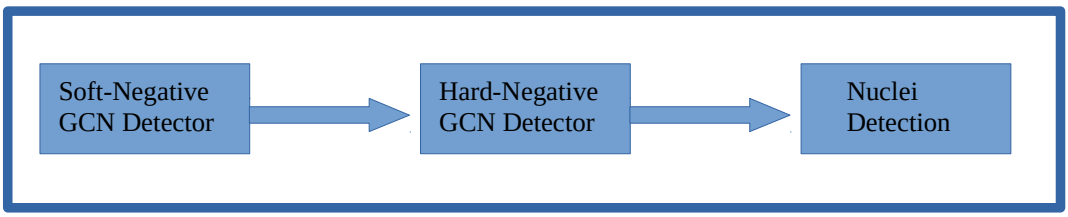


Figure 6.2: Cascade GCN pipeline for nuclei detection.

We formulate nuclei detection as a simple binary classification problem. The architecture of the method is shown in Figure 6.1. Given an input histology image of size 500 x 500,

extract patches of size 27 x 27 with centre coordinates as ground truth nuclei centres as positive samples. To extract negative samples, we have chosen random coordinates that do not fall into the proximity of the ground truth nuclei centre with distance d, as shown below.

$$Sample = \begin{cases} Positive, & \forall c, \text{ Such that c is ground truth coordinate.} \\ Negative, & \text{dist(Nc - c)} > \text{d, Where Nc is random center coord.} \end{cases}$$

To formulate the classification, we have labelled positive patches as 1 and negative patches as 0. We have randomly set the number of negative samples required to extract from each image. As we have used the spectral-based graph convolutional operator to learn features. To perform graph convolution, the patch of size 27×27 is considered a 2D grid graph having a set of nodes (V), a set of weighted edges (E), and an adjacency matrix (A). The graph $(N \times 729 \times 729)$ in Figure 6.1 is a binary adjacency matrix A and possesses the property that each node is connected with its neighbouring nodes, which form the basis of locality for the convolution operation. The operation of spectral-based GCN is described in Section 2.3.2.1. After the two graph convolution layers, there are two fully connected layers, with the last layer having one neuron activated by a sigmoid function to classify the feature vector extracted from the previous two graph convolutional layers. The inherent strength of the GCN architecture lies in its ability to adeptly extract local contextual information from these interconnected nodes. This capability empowers the GCN to learn complex representations that encapsulate the intricate features of the nuclei in a remarkably effective manner. To effectively improve nuclei detection and maintain the higher recall value, this graph convolutional layer-based classifier is used in cascaded architecture.

6.2.2 Cascaded GCN

In our pursuit of enhancing the detection process, we adopted the Cascade GCN method. This approach introduces a cascading architecture that progressively refines detection by systematically eliminating negative samples. However, this meticulous filtering process inevitably results in the removal of certain positive samples, thereby requiring a delicate balancing act between specificity and comprehensiveness. The cascade architecture unfolds in two distinct phases: soft negative elimination and hard negative elimination. Figure 6.1 illustrates the classification filtering architecture at each stage, while Figure 6.2 presents the complete flowchart of the Cascade GCN detector. This architectural choice is influenced by previous CNN-based nuclei detection architectures in CP-CNN and SP-CNN.

The initial stage involves the soft negative GCN detector, which primarily focuses on the elimination of the majority of negative background patches. The process of generating classification patches involves systematically scanning input images in a sliding window approach while ensuring minimal feature loss, with adjustments of one pixel for both height and width. It is worth noting that a considerable number of window patches were eliminated as background during the initial stage.

The second stage, referred to as the hard negative detector, is geared to undergo training with adjusted parameters. This enables the extraction of additional positive samples while simultaneously eliminating more complex false positives that may have persisted through the prior stage. Utilising ground-truth information, the positive samples are segregated into true positives and false positives. A positive sample is classified as a true positive if its predicted nucleus centre is in proximity to the ground truth nucleus centre with a pixel distance of less than d; otherwise, it is classified as a false positive.

$$Sample = \begin{cases} TP, & \forall c, \\ FP, & \text{dist(Nc - c)} > \text{d}, \end{cases}$$

Here, c represents the ground-truth coordinate, d is the proximity pixel distance, and N_c denotes the predicted centre coordinate.

The cascade architecture thus combines the soft and hard negative elimination phases, capitalizing on their collective efficacy to significantly enhance the detection process. The nuclei detection block from Figure 6.2 is designed to accurately locate nuclei while retaining effective positive patch samples at a higher resolution. This serves as the final phase of the cascading pipeline. The last two phases focus on binary classification as soft and hard elimination stages, filtering out the most negative samples while maintaining higher precision and recall. The remaining positive samples, containing nuclei centre coordinates, undergo Non-Maximum suppression (NMS) to eliminate redundancy, utilising an intersection over union (IoU) threshold of 0.30 and a maximum output bounding box size of 1200.

In essence, the Cascade GCN approach operates as a multi-layered, refined detection strategy. By meticulously balancing the removal of negative samples with the retention of positive samples, coupled with the finality of NMS, the cascade framework significantly elevates the accuracy and precision of nuclei detection.

6.3 Experimentation

6.3.1 Dataset and Implementation

The dataset employed consists of 100 H&E-stained histology images showcasing adenocarcinoma, with a 2-fold cross-validation experiment being conducted. This dataset is specifically used for problems centred around CNN-based nucleus detection. Each image boasts dimensions of 500×500 pixels. It's worth noting that these images do contain certain noises, such as overstaining and instances of autofocus failure. Notably, the annotation of nuclei is meticulously carried out by domain experts, yielding a total of 29,756 nuclei that serve as focal points for supervised learning tasks.

Our model is meticulously implemented using Tensorflow 2.0 and Keras. This proposed model undergoes comprehensive training and evaluation on a personal computer equipped with an Intel i7 CPU and an NVIDIA GEFORCE GTX 1080Ti. The learning rate is judiciously set at 0.001, with a batch size of 100. Additionally, a dropout rate of 0.2 is applied to both the CPCNN and the proposed model GCN, ensuring a fair and balanced comparison. To further explore the behaviours of both models within distinct loss environments, we assess the effectiveness of both binary cross-entropy and focal loss. For the implementation of SC-CNN, the learning rate is established at 0.01. The scheduling of learning adheres to the recommendations provided by the author [176].

Within this GCN architecture in Figure 6.1, a 2D grid graph of dimensions 27×27 serves as the input, accompanied by an adjacency matrix os size 729×729. The input graph signal, bearing dimensions $27 \times 27 \times 3$, is normalised between 0 and 1 through a normalisation function,

$$X = \left(\frac{X - \mu}{\sigma} \right) \tag{6.1}$$

where X signifies image data, μ denotes the mean, and σ represents the standard deviation. The architectural foundation features a 2D grid graph with a structure size of $N \times N$, where N signifies the number of nodes in the graph, corresponding to the 27×27 signal size. Each node in this configuration accommodates a 3-channel RGB signal. This 2D grid graph, depicted as a binary adjacency graph of $N \times N$, encapsulates the interconnections between nodes, marked as 1 or 0. The initial and second Graph Convolutional Networks (GCNs) employ 36 and 48 filters, respectively, harnessing their capacity to preserve vital features and eliminate low-frequency components. This strategic filter arrangement is pivotal for the enhancement of desirable characteristics. To ensure the retention of optimal patches for subsequent stages,

a higher recall is attained by training with the maximum precision-recall rate for a patience period of 10. Evaluation is facilitated by a sigmoid classification layer with a decision boundary set at 0.50.

For stage 1, where the objective is to sustain a high recall rate and eliminate challenging negative samples, an identical architecture as depicted in Figure 6.1 is employed. This model is trained with the objective of achieving the minimum validation loss with a patience factor of 10. The second stage involves leveraging this model with a training dataset comprising 38,643 false-positive and 100,000 true-positive samples. To create a balanced training dataset, negative samples are augmented through horizontal flipping and rotation by 90 degrees. This meticulous approach to data augmentation ensures that the training process is comprehensive and effective, facilitating the accurate and efficient functioning of subsequent stages.

6.3.2 Model Evaluation

Figure 6.4 shows the qualitative detection results on the unseen sample images. For quantitative analysis, we define the ground-truth areas as a circular region with 8 pixels in every annotated nuclei centre. A detected nuclei centroid is considered a true positive (TP) only if it lies within the ground-truth areas; otherwise, it is considered a false positive (FP). Each TP is matched with the nearest ground-truth annotated nuclei centre. The ground-truth nuclei centre that is not matched by any detected results is considered a false negative (FN). Based on the above definitions, we can compute the precision*(P)*, recall*(R)*, and F_1 score as $P = \frac{TP}{TP+FP}$, $R = \frac{TP}{TP+FN}$ and $F_1 = 2 * \frac{P*R}{P+R}$ respectively.

We evaluated the proposed model with *CNN-based Pixel-Wise Classification* (PWC) [177], or *Center of Patch Convolutional Neural Network* (CPCNN) [176] which share the same architecture as mentioned in the chapter [176] except that it utilises the sigmoid activation in the last layer.

Figure 6.3 shows the validation precision-recall curves with respect to the epochs of the GCN method and CPCNN. These curves are generated over the maximum precision-recall metric with patience 10. As we can see in Figure 6.3(a), that CPCNN fails to learn features over the focal loss, and their evaluation quantitative results in Table 6.1 also explain best. While using the binary cross-entropy loss for CPCNN, there is a large deviation between two folds as compared to the proposed model using GCN. The GCN-based model can learn features better in focal loss, but it also works for binary cross-entropy loss, which is not true in the case of CPCNN focal loss. Also, the deviation between the folds in GCN is smaller compared to

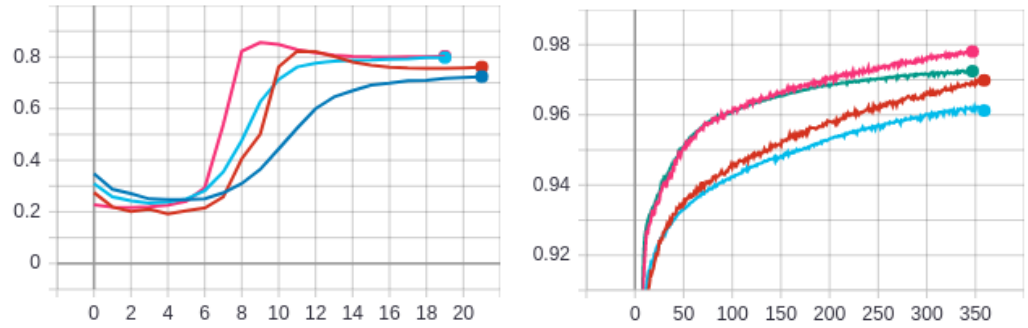
CPCNN, which shows the stability of the GCN-based proposed model.

Method	Precision	Recall	F1
SC-CNN (lr 0.01)	0.7183 ± 0.0063	0.6995 ± 0.0579	0.7078 ± 0.0328
SC-CNN (lr scheduled)	0.6355 ± 0.1163	0.7909 ± 0.0416	0.6951 ± 0.0555
CP-CNN (Bin cross)	0.7291 ± 0.0677	0.6930 ± 0.0227	0.7078 ± 0.0203
CP-CNN (Focal Loss)	-	-	-
GCN (Bin cross)	0.6832 ± 0.0019	0.6757 ± 0.0141	0.6793 ± 0.0061
GCN (Focal Loss)	**0.7336** ± 0.0055	0.6077 ± 0.0028	0.6647 ± **0.0005**
Cascade-GCN (Focal Loss)	0.7334 ± **0.0004**	**0.7927** ± **0.0005**	**0.7619** ± **0.0005**

Table 6.1: Quantitative results: Comparative precision, recall and F1 score results of nuclei detection various models.

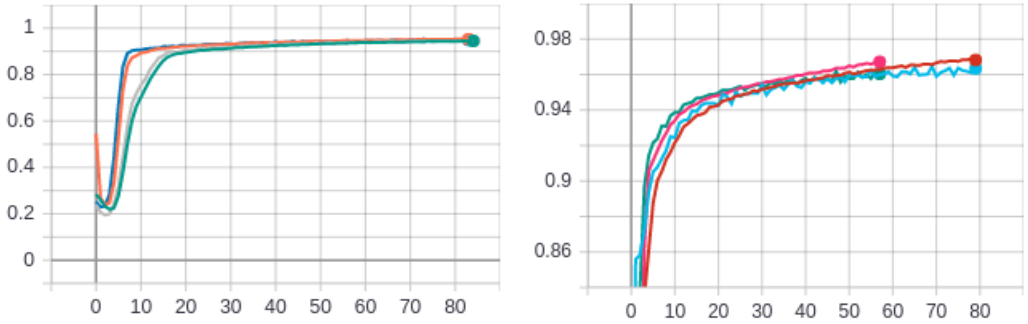
6.3.3 Comparison with other works

We also compared our model with the state-of-the-art proximity map-based nuclei detection regression model (SC-CNN) [176]. While comparing with SC-CNN, we have eliminated the preprocessing of augmentation and HSV space channel separation to keep training data the same for all methods for comparison. As demonstrated in Table 6.1, there is a substantial difference between the precision and recall values of SC-CNN with a stable learning rate and a scheduled learning rate. With careful observation of standard deviation values, we can see that the GCN-based proposed model performs better in precision, recall, and F1 score. While compared with the SC-CNN scheduled learning rate, the GCN precision value is increased by almost 10%. One of the advantages of our GCN model with focal loss is that it learns features better, while CP-CNN struggles to learn hardly any features, which proves the stability of the GCN-based model. When using the GCN in a cascade fashion, the F1 score increases almost 6% more than other models, as shown in Table 6.1. While observing the cascade qualitative results in Figure. 6.4, It shows a better recall value than the other but at the cost of a lower recall, whereas two-fold quantitative evaluation results on 50 images in Table 6.1, show that overall precision, recall, and F1 score are in a better position than the rest.



(a) CPCNN: Focal loss

(b) CPCNN:Binary loss



(c) GCN: Focal loss

(d) GCN:Binary loss

Figure 6.3: Qualitative Comparison of Results. The Figure shows the 2-fold training and validation curves of prc values along the y axis and epochs along the x axis. The colour name with a subscript shows the train or val curve of the fold number. (a): blue_{tr1}, red_{va1}, sky_{tr2} pink_{va2}, (b): red_{tr1}, sky_{va1}, pink_{tr2} $\text{green}_{\text{va2}}$, (c): $\text{green}_{\text{tr1}}$, gray_{va1}, $\text{Orange}_{\text{tr2}}$, blue_{va2}, (d): red_{tr1}, sky_{va1}, pink_{tr2}, $\text{green}_{\text{va2}}$

6.4 Results

The evaluation of the cascaded-GCN method unveils a significant advancement of 6% in the F1 score compared to alternative techniques, affirming its efficacy in nuclei detection. Notably, the two-fold cross-validation recall rate reaches 84.53%, showcasing the method's capacity to accurately identify a substantial portion of true positive samples with the configuration mentioned in Section 6.3.1. However, this achievement is accompanied by a trade-off, resulting in a slightly diminished precision rate of 65.45%. This balance between recall and precision, inherent in classification tasks, reflects the inherent tension of maximising both metrics simultaneously.

Examining the computational time for the two stages yields insightful observations. The first stage, characterised by soft negative elimination, consumes a considerable 4 hours, 27 minutes, and 43 seconds. Stage 2, focused on hard negative elimination, drastically reduces

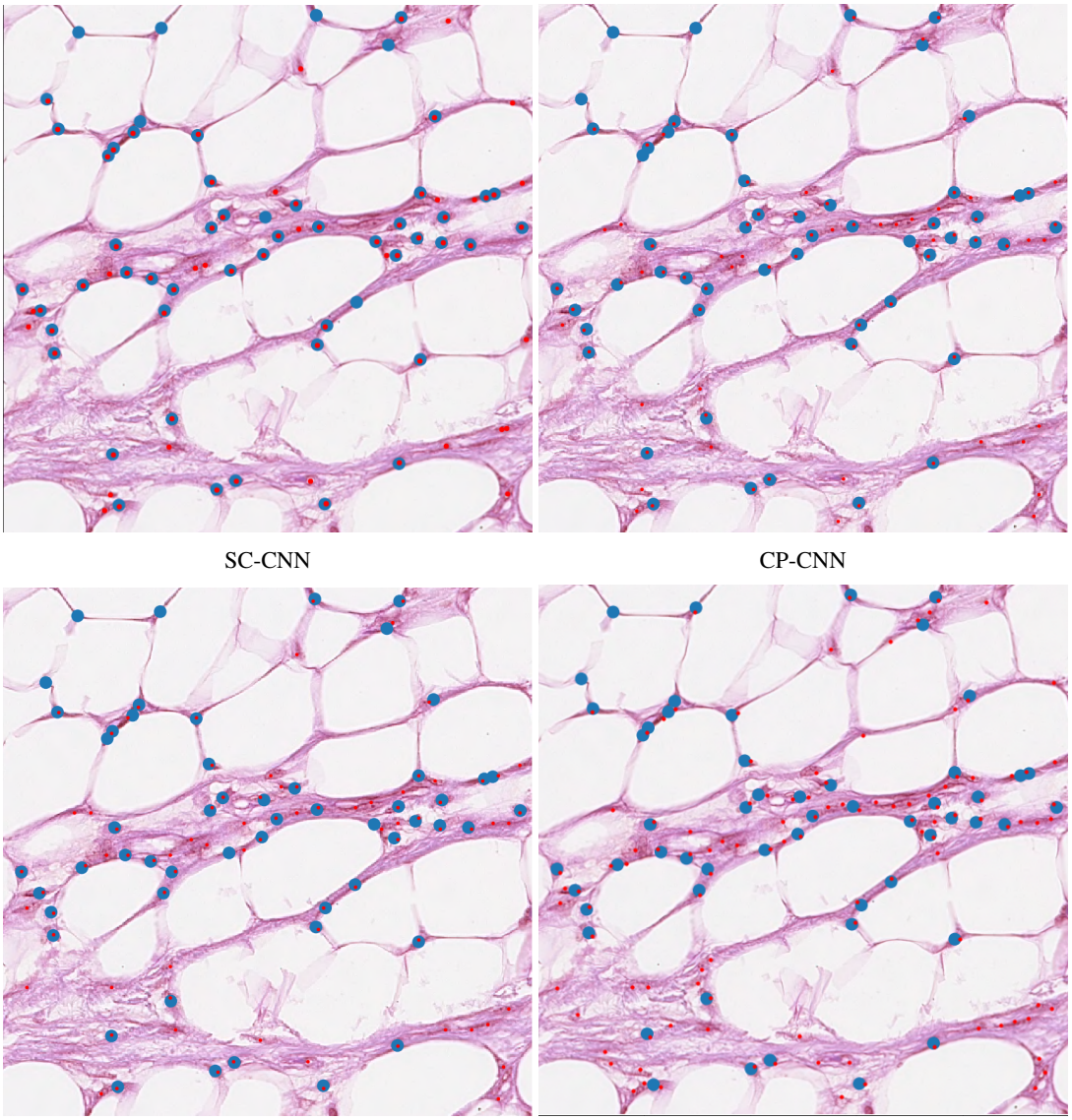


Figure 6.4: Nuclei detection results on the sample image. Red dots represent the detected nuclei centre. The ground-truth annotations are represented by blue circles for better illustrations.

the processing time to 17 minutes and 12 seconds due to the prior elimination of numerous negative samples in Stage 1. An interesting strategic choice is emphasised throughout the methodology, prioritising the retention of positive patches as the cascade progresses and aiming to augment the precision-recall rate. By emphasising recall, the approach ensures sensitivity to positive cases while striving to minimise false negatives. The quantitative results are described in Table 6.1 and the qualitative comparison in Figure 6.3.

The core idea of the proposed spectral GCN method lies in its capacity to harness the inherent graph structure of images, effectively capturing local dependencies between pixels. This architectural choice proves particularly advantageous in mitigating image-related challenges such as noise and poor staining. By treating the image as a graph, with pixels as nodes and edges as spatial relationships, spectral GCN facilitates the propagation of information between neighbouring pixels. This is especially valuable when dealing with missing or corrupted pixel values due to noise or staining issues. Furthermore, spectral graph convolution equips the network to learn domain-specific filters that identify patterns and features beyond the scope of spatial domain analysis. The network's ability to focus on the image's fundamental structural attributes rather than its noise or distortions enhances its ability to handle challenging scenarios like noise and poor staining.

A comprehensive assessment of the Cascaded GCN approach is achieved through ablation experiments. These experiments involve the systematic removal of elements to gauge their impact on performance. In this context, the F1 score is seen to decrease by 33.73%, with a small standard deviation of 0.01567, underscoring the methodology's sensitivity to its components. In conclusion, the cascaded-GCN approach demonstrates its potential to address the nuclei detection challenge effectively. By judiciously balancing recall and precision, leveraging the graph structure of images, and orchestrating a cascade of processing stages, the method showcases substantial improvements. Its performance gains over existing methods underscore its applicability, while its adaptability to complex domains and its adeptness at handling noise and staining issues make it a compelling tool for nuclei detection in histology images.

6.5 Summary

In this chapter, we propose a cascaded graph convolution approach for the nuclei detection task. The proposed GCN classifier differs from the traditional CNN classifier by introducing a graph convolutional operator to learn features in the training data. We have experimentally demonstrated the superior performance of the cascaded architecture of the GCN classifier in

terms of stability and precision, recall, and F1 score compared with the CNN-based state-of-the-art and also observed the validation behaviour of CPCNN and GCN with different loss functions. In future work, we aim to improve the performance with other graph convolutional operators and explore adding more classifiers.

Chapter 7

Conclusions and Future Work

Contents

7.1 Conclusion

In this thesis, we have investigated the feasibility of applying irregular domain deep learning to both medical image analysis and generic computer vision problems, more specifically, MNIST numerical digit classification, medical image segmentation, cell detection, and nuclei detection. Central to our approach has been the strategic utilization of graphs as a potent sampling mechanism, facilitating localized feature learning essential for robust and effective results.

In our graph construction study, we propose a unique technique for performing convolutional operations on graphs by using a local subgraph approach. The proposed weighted graph construction methods are also explored to understand their impact on the model's behaviour. Results show that the model can learn local features throughout the network with improved stability by introducing local subgraph features. We also observe that changing the size of the graph in local architecture does not appear to improve it, although it does in global architecture, particularly in binary. However, further investigation is needed to improve the subgraph partition method and the graph global pooling approach to aggregate vertices in the subgraph spatial domain. The study suggests that the proposed graph sampling approach could be useful in solving various computer vision image-based tasks, such as image classification, cell segmentation, and cell detection in immunostained histology images.

For medical image segmentation, we tried to fill the knowledge gap between computer scientists and experienced pathologists by introducing an automated segmentation scheme based on a graph sampling approach. The proposed method for performing cell image segmentation using spectral and spatial graph convolutional networks, which allow patch-wise distribution of the original image to improve feature learning, The comparative result of the proposed method of spectral, spatial-based graph convolution with FCN is presented. The observations from the experiments suggest that graph convolutional networks improve segmentation results.

The proposed graph proposal method demonstrates the ability to achieve results comparable to CNN-based object detection by using a simple object detection approach that has the potential to generalise to non-Euclidean data. The challenge of spatial relations in graphs is addressed by the BFS algorithm, which uses a graph sampling approach for classification. The proposed method achieved an average precision score. However, there were higher computational costs associated with this approach due to the use of a graph traversal algorithm for each proposal and the online training of arbitrary-size graphs.

The performance of nuclei detection can be improved by using a cascaded arrangement of graph-based convolution networks, and each stage in a cascaded architecture uses a clas-

sification network based on the graph sampling approach. In this thesis, we have shown that strategies and methods derived from irregular domain deep learning are efficient to address cell segmentation and nuclei detection problems. The main contributions of this work can be summarised as follows:

- **Impact of Graph Construction on Localised Feature Mining**

  Exploring the study of the impact of graph construction on the behaviour of the graph convolutional operations and analysing the impact of the graph's construction on the model's behaviour. Creating a method to utilise a parametrically defined graph to represent a localised sampling operation on an underlying domain. First proposed a three different weighted graph such as a binary-weighted, inverse Euclidean-weighted, and gaussian-weighted 2D grid graph. Second, we applied a graph convolution network for the MNIST digit classification tasks, and finally, we proposed a new approach to solve the limitation of constructing a larger graph by using a subgraph partition sampling approach.

- **Graph Convolution Networks for Cell Segmentation**

  Proposed two graph-based convolution methods for cell segmentation to improve the analysis of immunostained slides. First is spectral-based Graph-CNN, where spectral-based GCN operators along with graph pooling are arranged in U-Net style. The proposed approach is able to learn features in an upsampling and downsampling manner, which is inspired by the CNN-based approach with the added advantage of generalisation on Non-Euclidean domain data. The second spatial-based graph-CNN method does not need to use graph pooling with a simple architecture to outperform spectral.

- **Graph Proposal Neural Networks for Cell Detection**

  Propose a graph convolution-based region proposal mechanism for object detection in non-Euclidean data. Address the challenge of spatial feature extraction in irregular domains. Similar to the region proposal in R-CNN, the way of extracting positive object proposals is proposed by introducing the graph proposal neural network algorithm and multi-label classification.

- **Cascaded Approach for Nuclei Detection in histopathology images**

We proposed an irregular domain deep learning cascaded scheme that increases recall progressively with increasing stages. A graph convolution-based classification model to perform nuclei classification.

7.2 Future Work

There is a lot of potential for generalising deep learning approaches that use spatially connected feature descriptors. The formulation of deep learning techniques on the graph domain may benefit other application areas that employ convolutional operators on the Cartesian grid. The large improvements shown in self-learning feature descriptor techniques might be made available to new application areas by generalising the representation learning methodology. Social networking sites to decentralised weather monitoring stations are just a few of the many application domains. It will be fascinating to observe how these applications can leverage deep learning techniques that, up to now, have not been able to use the explicit spatial interactions between nodes on the input domain.

Deep learning techniques for irregular domains are still in the early stages of research. Study is necessary for understanding learned filters, including research into their optimisation and visualisation. Due to the specified spatial structure in some domains, it is feasible to see feature maps on the spatial domain; however, there is presently no efficient method for exploring learned descriptors, such as those found in CNN filter visualisation. ResNets [44], AutoEncoders [185–187], Inception modules [43], and Generative Adversarial Networks [188] are a few examples of deep learning techniques in the regular domain that make use of residual information. Although the usage of all of these techniques may theoretically be extended to the graph domain, they presently all employ the standard spatial domain input of images and volumes.

The choice of graph construction and pooling methods for a given problem domain is still a topic of active research, and opening up collaborations with the graph domain community will help us learn more about how to choose approaches for a given Graph-CNN architecture. The ability to employ a deep representation learning technique with a localised filtering behaviour is more widely accessible to a variety of application areas by generalising the convolution and pooling processes. A disadvantage of this relaxation is that implementing domain-specific restrictions might benefit applications inside the particular domain, much like how the implementation of the array-based input constraint resulted in major performance improvements for the image domain community. To aid in the widespread adoption of the techniques, efforts

to develop intuition regarding the application of graph-based deep learning are necessary, but they could take some time, especially given the ongoing development of novel techniques and understanding in the deep learning community as a whole. Utilising existing techniques for grouping different graph types and comparable vertices developed by the graph domain community [189] and [190] will make it easier to create Graph Convolutional Neural Network approach. There are several potential future work extending to current work in this thesis.

- There are a few alternative ways of constructing graphs, each of which has the potential to increase the functionality of the graph operator. There is also the opportunity to investigate suggested approaches for graph creation by applying them to a variety of datasets and performing additional computer vision tasks, such as segmentation and detection. The subgraph partitioning technique may be expanded to include pictures of a larger size, and this is an excellent chance to investigate comparative studies of the efficacy of images of varying sizes in relation to the various types of graphs.
- Finding spatial relationships in an irregular domain is an ongoing and challenging topic, particularly when it comes to the task of object recognition. The Breadth-First-Search strategy, which is used to solve such issues, may be expanded by using a variety of neighbourhood strategies as well as a variety of databases.
- Improvement to the already used method of segmentation by the use of a new spectral and spatial graph convolution operator, as well as alternative databases Additionally, the nuclei detection design might be enhanced by the addition of additional layers to the cascaded architecture.

Overall, this comprehensive exploration into medical image analysis has not only uncovered novel insights but has also underscored the significant potential of deep learning in irregular domains. The methodologies and findings presented herein not only advance the field of image analysis but also shed light on the vast array of challenges and opportunities that lie ahead. As the deep learning community continues to evolve, this work serves as a stepping stone towards wider applications, benefiting both the machine learning realm and the diverse domains that stand to harness the power of representational learning techniques. The culmination of this thesis is not an end but rather a new beginning, as the pursuit of knowledge and innovation remains boundless.

Bibliography

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2323, 1998.

[2] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "cuDNN: Efficient Primitives for Deep Learning," 2014.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the Association for Computing Machinery*, vol. 60, no. 6, pp. 84–90, 2017.

[4] S. K. Zhou, H. Greenspan, C. Davatzikos, J. S. Duncan, B. V. Ginneken, A. Madabhushi, J. L. Prince, D. Rueckert, and R. M. Summers, "A review of deep learning in medical imaging: Imaging traits, technology trends, case studies with progress highlights, and future promises," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 820–838, 2021.

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Conference on Neural Information Processing Systems - Volume 1*. Cambridge, MA, USA: MIT Press, 2015, p. 91–99.

[7] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2961–2969.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[9] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, "Dive into deep learning," *arXiv preprint arXiv:2106.11342*, 2021.

[10] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng, "End-to-end text recognition with convolutional neural networks," in *Proceedings of the 21st International Conference on Pattern Recognition*. IEEE, 2012, pp. 3304–3308.

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

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

[13] C. Liu, Y. Zhan, C. Li, B. Du, J. Wu, W. Hu, T. Liu, and D. Tao, "Graph pooling for graph neural networks: Progress, challenges, and opportunities," *arXiv preprint arXiv:2204.07321*, 2022.

[14] P. N. Srinivasu, S. Ahmed, A. Alhumam, A. B. Kumar, and M. F. Ijaz, "An AW-HARIS based automated segmentation of human liver using CT images," *Computers, Materials & Continua*, vol. 69, no. 3, pp. 3303–3319, 2021.

[15] E. Vendrow and Z. Ma, "Cell segmentation with traditional and deep learning methods on evican-a partially annotated dataset of grayscale images of 30 different cell lines from multiple microscopes," 2022.

[16] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[17] X. Zhou, T. Tong, Z. Zhong, H. Fan, and Z. Li, "Saliency-CCE: Exploiting colour contextual extractor and saliency-based biomedical image segmentation," *Computers in Biology and Medicine*, p. 106551, 2023.

[18] N. O'Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, L. Krpalkova, D. Riordan, and J. Walsh, "Deep learning vs. traditional computer vision," in *Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference, Volume 11*. Springer, 2020, pp. 128–144.

[19] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: convolutional networks for biomedical image segmentation," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9351, pp. 234–241, 2015.

[20] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2020.

[21] S. Bahade, M. Edwards, and X. Xie, "Graph convolutional neural network for segmentation of immunostained hodgkin lymphoma histology images," in *Medical Image Understanding and Analysis*, 2019.

[22] ——, "Graph convolution networks for cell segmentation." in *International Conference on Pattern Recognition Applications and Methods*, 2021, pp. 620–627.

[23] ——, "Cascaded graph convolution approach for nuclei detection in histopathology images," in *International Conference on Video and Image Processing*, 2022.

[24] ——, "Graph proposal neural networks for cell detection immunostained hodgkin lymphoma histology images," in *International Symposium on Biomedical Imaging*, 2023.

[25] A. L. Samuel, "Machine learning," *The Technology Review*, vol. 62, no. 1, pp. 42–45, 1959.

[26] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, 1998, pp. 92–100.

[27] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 44, no. 1.2, pp. 206–226, 2000.

[28] B. Cm, *Pattern Recognition and Machine Learning*. Springer, New York, 2010.

[29] D. George, "Feature driven learning techniques for 3D shape segmentation," 2019. [Online]. Available: http://csvision.swan.ac.uk/uploads/Site/Publication/thesis.dageo.pdf

[30] A. Ng, "Coursera machine learning linear regression." [Online]. Available: https://www.coursera.org/lecture/machine-learning/linear-regression-model-part-1-1ACA2

[31] J. Brownlee, "Logistic regression for machine learning," *Machine Learning Mastery*, vol. 1, 2016.

[32] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[33] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE Access*, vol. 7, pp. 53 040–53 065, 2019.

[34] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, pp. 1–74, 2021.

[35] M. Roos, "Deep learning neurons versus biological neurons," https://towardsdatascience.com/deep-learning-versus-biological-neurons-floating-point-numbers-spikes-and-neurotransmitters-6eebfa3390e9, online; accessed 6-May-2023.

[36] O. Eluyode and D. T. Akomolafe, "Comparative study of biological and artificial neural networks," *European Journal of Applied Engineering and Scientific Research*, vol. 2, no. 1, pp. 36–46, 2013.

[37] F. Rosenbaltt, "The perceptron: a perciving and recognizing automation," *Cornell Aeronautical Laboratory*, 1957.

[38] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological Review*, vol. 65, no. 6, p. 386, 1958.

[39] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[40] M. Sazli, "A brief review of feed-forward neural networks," *Communications Faculty Of Science University of Ankara*, vol. 50, pp. 11–17, 01 2006.

[41] F. Rosenblatt, "The perceptron: A perceiving and recognizing automaton," *Cornell University, Ithaca, NY, Project PARA, Cornell Aeronaunt Laboratory, Rep*, pp. 85–460, 1957.

[42] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[43] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[45] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[46] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci, "A hypercube-based encoding for evolving large-scale neural networks," *Artificial Life*, vol. 15, no. 2, pp. 185–212, 2009.

[47] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," *arXiv preprint arXiv:1611.06440*, 2016.

[48] M. Babaeizadeh, P. Smaragdis, and R. H. Campbell, "Noiseout: A simple way to prune neural networks," *arXiv preprint arXiv:1611.06211*, 2016.

[49] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[50] P. J. Werbos, "Applications of advances in nonlinear sensitivity analysis," in *System Modeling and Optimization*. Springer, 1982, pp. 762–770.

[51] D. Ruhmelhart, G. Hinton, and R. Wiliams, "Learning representations by back-propagation errors," *Nature*, vol. 323, no. 533-536, p. 10, 1986.

[52] J. Kiefer and J. Wolfowitz, "Stochastic estimation of the maximum of a regression function," *The Annals of Mathematical Statistics*, pp. 462–466, 1952.

[53] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, pp. 400–407, 1951.

[54] K. Janocha and W. M. Czarnecki, "On loss functions for deep neural networks in classification," *arXiv preprint arXiv:1702.05659*, 2017.

[55] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*. PMLR, 2013, pp. 1310–1318.

[56] A. F. Agarap, "Deep Learning using Rectified Linear Units (ReLU)," *arXiv preprint arXiv:1803.08375*, 2018.

[57] L. Lu, "Dying ReLU and initialization: Theory and numerical examples," *Communications in Computational Physics*, vol. 28, no. 5, pp. 1671–1706, Jun 2020.

[58] A. L. Maas, A. Y. Hannun, A. Y. Ng *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. in International Conference on Machine Learning*, vol. 30, no. 1, 2013, p. 3.

[59] M. Edwards, "Representation learning in irregular domains," Ph.D. dissertation, Swansea University, 2018.

[60] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[61] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

[62] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*. pmlr, 2015, pp. 448–456.

[63] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[64] S. Sun, Z. Cao, H. Zhu, and J. Zhao, "A survey of optimization methods from a machine learning perspective," *IEEE Transactions on Cybernetics*, vol. 50, no. 8, pp. 3668–3681, 2019.

[65] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," *Neural Networks: Tricks of the Trade: Second Edition*, pp. 437–478, 2012.

[66] M. Li, T. Zhang, Y. Chen, and A. J. Smola, "Efficient mini-batch training for stochastic optimization," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 661–670.

[67] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization." *Journal of Machine Learning Research*, vol. 12, no. 7, 2011.

[68] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

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

[70] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Time series classification using multi-channels deep convolutional neural networks," in *Web-Age Information Management: 15th International Conference*. Springer, 2014, pp. 298–310.

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

[72] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[73] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European Conference Computer Vision*. Springer, 2014, pp. 818–833.

[74] U. Karn, "An intuitive explanation of convolutional neural networks," *The Data Science*, 2016.

[75] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[76] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proceedings of the 27th International Conference on Machine Learning*, 2010, pp. 111–118.

[77] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[78] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[79] E. P. Ijjina and C. K. Mohan, "Human action recognition based on mocap information using convolution neural networks," in *13th International Conference on Machine Learning and Applications*, 2014, pp. 159–164.

[80] ——, "Human action recognition based on motion capture information using fuzzy convolution neural networks," in *Eighth International Conference on Advances in Pattern Recognition*, 2015, pp. 1–6.

[81] A. Johnson, "Spin-Images: A Representation for 3-D Surface Matching," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, August 1997.

[82] A. E. Johnson and M. Hebert, "Surface matching for object recognition in complex three-dimensional scenes," *Image and Vision Computing*, vol. 16, no. 9-10, pp. 635–651, 1998.

[83] D. George, X. Xie, and G. K. Tam, "3D mesh segmentation via multi-branch 1D convolutional neural networks," *Graphical Models*, vol. 96, pp. 1–10, 2018.

[84] W. Zhu, C. Lan, J. Xing, W. Zeng, Y. Li, L. Shen, and X. Xie, "Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[85] J. Masci, D. Boscaini, M. M. Bronstein, and P. Vandergheynst, "ShapeNet: convolutional neural networks on non-euclidean manifolds," *ArXiv Preprint*, vol. abs/1501.06297, 2015.

[86] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs: Graph filters," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, no. 412, 2013, pp. 6163–6166.

[87] F. R. Chung and F. C. Graham, *Spectral graph theory*. American Mathematical Soc., 1997, no. 92.

[88] H. LI, "Properties and applications of graph laplacians," http://math.uchicago.edu/~may/REU2022/REUPapers/Li,Hanchen.pdf, online; accessed 11-March-2023.

[89] A. M. Martınez, P. Mittrapiyanuruk, and A. C. Kak, "On combining graph-partitioning with non-parametric clustering for image segmentation," *Computer Vision and Image Understanding*, vol. 95, no. 1, pp. 72–85, 2004.

[90] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, pp. 395–416, 2007.

[91] R. Singh, A. Chakraborty, and B. Manoj, "Graph fourier transform based on directed laplacian," in *International Conference on Signal Processing and Communications*, 2016, pp. 1–5.

[92] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.

[93] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[94] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs," *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1644–1656, 2013.

[95] Y. Jin and D. I. Shuman, "An m-channel critically sampled filter bank for graph signals," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017, pp. 3909–3913.

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

[97] ——, "Discrete signal processing on graphs: Frequency analysis," *IEEE Transactions on Signal Processing*, vol. 62, no. 12, pp. 3042–3054, 2014.

[98] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI open*, vol. 1, pp. 57–81, 2020.

[99] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.

[100] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proceedings. IEEE International Joint Conference on Neural Networks*, vol. 2, 2005, pp. 729–734.

[101] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[102] N. Peng, H. Poon, C. Quirk, K. Toutanova, and W.-t. Yih, "Cross-sentence n-ary relation extraction with graph lstms," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 101–115, 2017.

[103] X. Wang, Y. Ye, and A. Gupta, "Zero-shot recognition via semantic embeddings and knowledge graphs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6857–6866.

[104] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[105] A. Bojchevski and S. Günnemann, "Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking," *arXiv preprint arXiv:1707.03815*, 2017.

[106] M. Zitnik, M. Agrawal, and J. Leskovec, "Modeling polypharmacy side effects with graph convolutional networks," *Bioinformatics*, vol. 34, no. 13, pp. i457–i466, 2018.

[107] V. Garcia and J. Bruna, "Few-shot learning with graph neural networks," *arXiv preprint arXiv:1711.04043*, 2017.

[108] C.-W. Lee, W. Fang, C.-K. Yeh, and Y.-C. F. Wang, "Multi-label zero-shot learning with structured knowledge graphs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1576–1585.

[109] M. Kampffmeyer, Y. Chen, X. Liang, H. Wang, Y. Zhang, and E. P. Xing, "Rethinking knowledge graph propagation for zero-shot learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 487–11 496.

[110] K. Marino, R. Salakhutdinov, and A. Gupta, "The more you know: Using knowledge graphs for image classification," *arXiv preprint arXiv:1612.04844*, 2016.

[111] X. Chen, L.-J. Li, L. Fei-Fei, and A. Gupta, "Iterative visual reasoning beyond convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7239–7248.

[112] X. Liang, X. Shen, J. Feng, L. Lin, and S. Yan, "Semantic object parsing with graph lstm," in *European Conference Computer Vision*. Springer, 2016, pp. 125–143.

[113] L. Landrieu and M. Simonovsky, "Large-scale point cloud semantic segmentation with superpoint graphs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4558–4567.

[114] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *Acm Transactions On Graphics*, vol. 38, no. 5, pp. 1–12, 2019.

[115] X. Qi, R. Liao, J. Jia, S. Fidler, and R. Urtasun, "3D graph neural networks for RGBD semantic segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5199–5208.

[116] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei, "Relation networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3588–3597.

[117] J. Gu, H. Hu, L. Wang, Y. Wei, and J. Dai, "Learning region features for object detection," in *European Conference Computer Vision*, 2018, pp. 381–395.

[118] X.-M. Zhang, L. Liang, L. Liu, and M.-J. Tang, "Graph neural networks and their current applications in bioinformatics," *Frontiers in Genetics*, vol. 12, p. 690049, 2021.

[119] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges," *arXiv preprint arXiv:2104.13478*, 2021.

[120] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, 2016, pp. 3844–3852.

[121] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *Computing Research Repository*, vol. abs/1609.02907, pp. 1–14, 2016.

[122] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[123] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[124] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Computer Vision and Pattern Recognition*, 2017, pp. 5115–5124.

[125] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International Conference on Machine Learning*, 2017, pp. 1263–1272.

[126] D. Grattarola, "Graph neural networks lecture slides," https://danielegrattarola.github.io/files/talks/2021-03-01-USI_GDL_GNNs.pdf, online; accessed 22-March-2023.

[127] M. Edwards and X. Xie, "Graph based convolutional neural network," in *British Machine Vision Conference*, 2016.

[128] ——, "Graph-based cnn for human action recognition from 3d pose," in *Deep Learning in Irregular Domains Workshop, British Machine Vision Conference*, 2017.

[129] C. Chevalier and I. Safro, "Comparison of coarsening schemes for multilevel graph partitioning," in *International Conference on Learning and Intelligent Optimization*, 2009, pp. 191–205.

[130] J. Katkar, T. Baraskar, and V. R. Mankar, "A novel approach for medical image segmentation using PCA and k-means clustering," in *International Conference on Applied and Theoretical Computing and Communication Technology*, 2016, pp. 430–435.

[131] R. Rulaningtyas, A. B. Suksmono, T. Mengko, and P. Saptawati, "Multi patch approach in k-means clustering method for color image segmentation in pulmonary tuberculosis identification," in *International Conference on Instrumentation, Communications, Information Technology, and Biomedical Engineering*, 2015, pp. 75–78.

[132] H. Yadav, P. Bansal, and R. Kumarsunkaria, "Color dependent k-means clustering for color image segmentation of colored medical images," in *Proceedings on International Conference on Next Generation Computing Technologies*, 2016, pp. 858–862.

[133] N. M. Salem, "Segmentation of white blood cells from microscopic images using k-means clustering," in *National Radio Science Conference*, 2014, pp. 371–376.

[134] A. S. Abdul Nasir, M. Y. Mashor, and H. Rosline, "Unsupervised colour segmentation of white blood cell for acute leukaemia images," in *IEEE International Conference on Imaging Systems and Techniques*, 2011, pp. 142–145.

[135] R. M. Thomas and J. John, "A review on cell detection and segmentation in microscopic images," in *International Conference on Circuits, Power and Computing Technologies*, 2017, pp. 1–5.

[136] D. Wang, A. Khosla, R. Gargeya, H. Irshad, and A. H. Beck, "Deep learning for identifying metastatic breast cancer," *Computing Research Repository*, vol. abs/1606.05718, 2016.

[137] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[138] Y. Arora and I. Patil, "Fully convolutional network for depth estimation and semantic segmentation." stanford. edu, 2017.

[139] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[140] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[141] M. G. Carneiro and L. Zhao, "Analysis of graph construction methods in supervised data classification," in *7th Brazilian Conference on Intelligent Systems*, 2018, pp. 390–395.

[142] T. C. Silva and L. Zhao, "Network-based high level data classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 6, pp. 954–970, 2012.

[143] J. R. Bertini Jr, L. Zhao, R. Motta, and A. de Andrade Lopes, "A nonparametric classification method based on k-associated graphs," *Information Sciences*, vol. 181, no. 24, pp. 5435–5456, 2011.

[144] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[145] M. G. Carneiro, T. H. Cupertino, R. Cheng, Y. Jin, and L. Zhao, "Nature-inspired graph optimization for dimensionality reduction," in *IEEE 29th International Conference on Tools with Artificial Intelligence*, 2017, pp. 1113–1119.

[146] B. Araújo and L. Zhao, "Data heterogeneity consideration in semi-supervised learning," *Expert Systems with Applications*, vol. 45, pp. 234–247, 2016.

[147] L. Berton and A. De Andrade Lopes, "Graph construction based on labeled instances for semi-supervised learning," in *2014 22nd International Conference on Pattern Recognition*, 2014, pp. 2477–2482.

[148] M. G. Carneiro, T. H. Cupertino, and L. Zhao, "K-associated optimal network for graph embedding dimensionality reduction," in *International Joint Conference on Neural Networks*, 2014, pp. 1660–1666.

[149] T. H. Cupertino, M. G. Carneiro, and L. Zhao, "Dimensionality reduction with the k-associated optimal graph applied to image classification," in *IEEE International Conference on Imaging Systems and Techniques*, 2013, pp. 366–371.

[150] T. H. Cupertino, L. Zhao, and M. G. Carneiro, "Network-based supervised data classification by using an heuristic of ease of access," *Neurocomputing*, vol. 149, pp. 86–92, 2015.

[151] M. G. Carneiro and L. Zhao, "High level classification totally based on complex networks," in *2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence*, 2013, pp. 507–514.

[152] M. G. Carneiro, L. Zhao, R. Cheng, and Y. Jin, "Network structural optimization based on swarm intelligence for highlevel classification," in *International Joint Conference on Neural Networks (IJCNN)*, 2016, pp. 3737–3744.

[153] A. Lancichinetti and S. Fortunato, "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities," *Physical Review E*, vol. 80, no. 1, p. 016118, 2009.

[154] S. I. Dimitriadis, E. Messaritaki, and D. K Jones, "The impact of graph construction scheme and community detection algorithm on the repeatability of community and hub identification in structural brain networks," *Human Brain Mapping*, vol. 42, no. 13, pp. 4261–4280, 2021.

[155] A. Daigavane, B. Ravindran, and G. Aggarwal, "Understanding convolutions on graphs," *Distill*, 2021, https://distill.pub/2021/understanding-gnns.

[156] P. Liu, X. Wang, and Y. Gu, "Graph signal coarsening: Dimensionality reduction in irregular domain," in *IEEE Global Conference on Signal and Information Processing*, 2014, pp. 798–802.

[157] C. Chevalier and I. Safro, "Comparison of coarsening schemes for multilevel graph partitioning," in *International Conference on Learning and Intelligent Optimization*, 2009, pp. 191–205.

[158] J. A. Katkar and T. Baraskar, "Medical image segmentation using PCA and k-mean clustering algorithm," in *Post Graduate Conference for Information Technology*, 2015, pp. 1–6.

[159] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[160] R. Girshick, "Fast R-CNN," in *International Conference on Computer Vision*, 2015, pp. 1440–1448.

[161] R. Huang, J. Pedoeem, and C. Chen, "YOLO-LITE: A real-time object detection algorithm optimized for non-gpu computers," in *IEEE International Conference on Big Data (Big Data)*, 2018, pp. 2503–2510.

[162] A. Micheli, "Neural network for graphs: A contextual constructive approach," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 498–511, 2009.

[163] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International Conference on Machine Learning*, 2016, pp. 2014–2023.

[164] W. Shi and R. Rajkumar, "Point-gnn: Graph neural network for 3d object detection in a point cloud," in *Computer Vision and Pattern Recognition*, June 2020.

[165] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," in *NIPS - Volume 2*, ser. NIPS'13. Red Hook, NY, USA: Curran Associates Inc., 2013, p. 2553–2561.

[166] D. C. Cireşan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Mitosis detection in breast cancer histology images with deep neural networks," in *Medical Image Computing and Computer Assisted Interventions*, 2013, pp. 411–418.

[167] N. Acosta-Mendoza, A. Gago-Alonso, and J. E. Medina-Pagola, "Frequent approximate subgraphs as features for graph-based image classification," *Knowledge-Based Systems*, vol. 27, pp. 381–392, 2012.

[168] R. A. Finkel and J. L. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta Informatica*, vol. 4, no. 1, pp. 1–9, 1974.

[169] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[170] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[171] P. Dalerba, T. Kalisky, D. Sahoo, P. S. Rajendran, M. E. Rothenberg, A. A. Leyrat, S. Sim, J. Okamoto, D. M. Johnston, D. Qian *et al.*, "Single-cell dissection of transcriptional heterogeneity in human colon tumors," *Nature biotechnology*, vol. 29, no. 12, pp. 1120–1127, 2011.

[172] C. A. O'Brien, A. Pollett, S. Gallinger, and J. E. Dick, "A human colon cancer cell capable of initiating tumour growth in immunodeficient mice," *Nature*, vol. 445, no. 7123, pp. 106–110, 2007.

[173] A. Basavanhally, M. Feldman, N. Shih, C. Mies, J. Tomaszewski, S. Ganesan, and A. Madabhushi, "Multi-field-of-view strategy for image-based outcome prediction of multi-parametric estrogen receptor-positive breast cancer histopathology: Comparison to oncotype dx," *Journal of pathology informatics*, vol. 2, 2011.

[174] J. S. Lewis Jr, S. Ali, J. Luo, W. L. Thorstad, and A. Madabhushi, "A quantitative histomorphometric classifier (quhbic) identifies aggressive versus indolent p16-positive oropharyngeal squamous cell carcinoma," *The American journal of surgical pathology*, vol. 38, no. 1, p. 128, 2014.

[175] G. N. van Muijen, D. J. Ruiter, W. W. Franke, T. Achtstätter, W. H. Haasnoot, M. Ponec, and S. O. Warnaar, "Cell type heterogeneity of cytokeratin expression in complex epithelia and carcinomas as demonstrated by monoclonal antibodies specific for cytokeratins nos. 4 and 13," *Experimental cell research*, vol. 162, no. 1, pp. 97–113, 1986.

[176] K. Sirinukunwattana, S. E. A. Raza, Y.-W. Tsang, D. R. Snead, I. A. Cree, and N. M. Rajpoot, "Locality sensitive deep learning for detection and classification of nuclei in routine colon cancer histology images," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1196–1206, 2016.

[177] Y. Xie, F. Xing, X. Kong, H. Su, and L. Yang, "Beyond classification: structured regression for robust cell detection using convolutional neural network," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 358–365.

[178] M. Veta, J. P. Pluim, P. J. Van Diest, and M. A. Viergever, "Breast cancer histopathology image analysis: A review," *IEEE Transactions on Biomedical Engineering*, vol. 61, no. 5, pp. 1400–1411, 2014.

[179] E. Cosatto, M. Miller, H. P. Graf, and J. S. Meyer, "Grading nuclear pleomorphism on histological micrographs," in *2008 19th International Conference on Pattern Recognition*. IEEE, 2008, pp. 1–4.

[180] Y. Yuan, H. Failmezger, O. M. Rueda, H. R. Ali, S. Gräf, S.-F. Chin, R. F. Schwarz, C. Curtis, M. J. Dunning, H. Bardwell *et al.*, "Quantitative image analysis of cellular heterogeneity in breast tumors complements genomic profiling," *Science translational medicine*, vol. 4, no. 157, pp. 157ra143–157ra143, 2012.

[181] C. D. Malon and E. Cosatto, "Classification of mitotic figures with convolutional neural networks and seeded blob features," *Journal of pathology informatics*, vol. 4, 2013.

[182] K. Nguyen, A. K. Jain, and B. Sabata, "Prostate cancer detection: Fusion of cytological and textural features," *Journal of pathology informatics*, vol. 2, 2011.

[183] A. A. Cruz-Roa, J. E. A. Ovalle, A. Madabhushi, and F. A. G. Osorio, "A deep learning architecture for image representation, visual interpretability and automated basal-cell carcinoma cancer detection," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2013, pp. 403–410.

[184] H. Wang, A. Cruz-Roa, A. Basavanhally, H. Gilmore, N. Shih, M. Feldman, J. Tomaszewski, F. Gonzalez, and A. Madabhushi, "Cascaded ensemble of convolutional neural networks and handcrafted features for mitosis detection," in *Medical Imaging: Digital Pathology*, vol. 9041, 2014, p. 90410B.

[185] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[186] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. JMLR Workshop and Conference Proceedings, 2012, pp. 37–49.

[187] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[188] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE signal processing magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[189] K. Riesen and H. Bunke, *Graph classification and clustering based on vector space embedding*. World Scientific, 2010, vol. 77.

[190] S. E. Schaeffer, "Graph clustering," *Computer science review*, vol. 1, no. 1, pp. 27–64, 2007.