Set Representation Learning: A Framework for Learning Gigapixel Images

by

Mohammed Adnan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Masters of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2021

Author's Declaration

Statement of Contributions

The thesis is based on the following publications:

[1] Shivam Kalra*, **Mohammed Adnan***, Graham Taylor, and Hamid R Tizhoosh. Learning Permutation Invariant Representations using Memory Networks. In European Conference on Computer Vision, pages 677–693. Springer, 2020.

[2] **Mohammed Adnan***, Shivam Kalra*, and Hamid R Tizhoosh. Representation Learning of Histopathology Images using Graph Neural Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pages 988–989, 2020.

[3] Shivam Kalra, **Mohammed Adnan**, Sobhan Hemati, Taher Dehkharghanian,Shahryar Rahnamayan, and Hamid Tizhoosh. Pay Attention with Focus: A novel learning scheme for classification of whole slide images. To appear in MICCAI 2021.

I have contributed to ideation, implementation, experimentation, and writing of all the above mentioned papers. I am corresponding author for Paper [1] with equal contributions as the first author. My contribution in Paper [3] is to lesser extent.

* denotes equal contribution

Abstract

In Machine Learning, we often encounter data as a *set* of instances such Point Clouds (set of x,y, and z coordinates), patches from gigapixel images (Digital Pathology, Satellite Imagery, Astronomical Images, etc.), Weakly Supervised Learning, Multiple Instance Learning, and so on. It is then convenient to have Machine Learning or AI algorithms that can learn set representation. However, most of the progress made in the last two decades has been limited to single instance-based algorithms and smaller image datasets such as MNIST, CIFAR10, and CIFAR100. In this work, I present novel algorithms for Set Representation Learning. The contribution of this work is two-fold:

1. This work introduces three novel methods for learning Set Representations; Memory based Exchangeable model (MEM), Graph Neural Network based Set Representation Learning method, and a Hierarchical Set Representation Learning method.
2. This work demonstrates that learning gigapixel images can be formulated as a set representation problem and provides a framework for efficiently learning gigapixel image representations.

Different themes are explored for Set Representation Learning. This work investigates Permutation Invariant Representations for Set Learning and introduces a new Permutation Invariant method - 'MEM'. Memory-based Exchangeable (MEM) model uses a Permutation Invariant architecture and memory networks to learn inter-dependencies/relation between different elements of the set. Subsequently, Graph Neural Networks (GNNs) are studied for Set Representation Learning, and a new GNN based Set Representation Learning method is proposed. Motivated by learning inter-dependencies among different elements in MEM, the proposed method learns an equivalent graphical representation to model interaction and interdependencies among different elements of the set. Lastly, this work introduces a new learning scheme for learning Hierarchical Set Representations.

To demonstrate the efficacy of the proposed algorithms, they are validated and benchmarked on a variety of synthetic and real-world datasets such as MNIST, Point Clouds, and Gaussian Distributions. Histopathology Images are used to demonstrate the application of Set Representation Learning for learning gigapixel images. State-of-the-art results on all datasets are achieved, thus demonstrating efficacy.

Acknowledgements

Writing this thesis has been fascinating and extremely rewarding. My journey at the University of Waterloo began in 2018 as a Research Intern during the final year of my undergrad studies in India. The last two years of graduate studies at the University of Waterloo amidst the pandemic taught me many life lessons. This thesis couldn't have been possible in the pandemic without the help and support of many people.

Looking back at the past three years, I'm immensely thankful to my supervisor Prof. Hamid Tizhoosh who molded me from an undergraduate student to a graduate student with an improving scientific acumen, work ethics, independent scientific thoughts and understanding. His encouragement and guidance throughout the MASc. program helped me in all times of research and during the writing of this thesis.

I would also like to express my gratitude to Prof. Graham Taylor for his unwavering support and valuable inputs. I would also like to gratefully acknowledge the computing resources provided by Prof. Taylor.

I owe special thanks to Prof. Oleg Michailovich and Prof. Apurva Narayan for taking out their time to review my thesis and provide valuable suggestions.

I'd like to thank my co-author Shivam Kalra for his help in writing papers and conducting the research. I appreciate the research discussions I had with my colleague Sobhan Hemati during the last two years.

I would like to acknowledge the Vector Scholarship in AI offered by Vector Institute and the Ontario Research Fund grant for providing much-needed financial support throughout my degree.

I'd like to thank my parents for raising me to value education. I sincerely appreciate their support and encouragement throughout my academic life.

At last, I would like to thanks my friends for being supportive; their encouragement and laughs kept me going. Finally, I pay my obeisance to god, the almighty, to have bestowed upon me good health, courage, inspiration, and the light.

Dedication

To family and friends.

Table of Contents

List of Figures **x**

List of Tables **xiii**

1 Introduction **1**

1.1 Background 1

1.2 Problem Definition 3

1.3 Contributions 4

I Background Literature **7**

2 Permutation Invariant Models **8**

2.1 Background 8

2.2 Exchangeablilty 10

2.3 Deep Sets: Universal Approximation of Sets 11

3 Graph Neural Networks **12**

3.1 Introduction 12

3.2 Spectral and Spatial GNNs 13

3.3 Graph Convolution Neural Networks (GCNNs) 14

3.4 Set Representation Learning using Graphs 15

4 Gigapixel Histopathology Images **17**
4.1 Background 17
4.2 Multiple Instance Learning for WSIs 19

II Proposed Methods **20**

5 Memory-Based Exchangeable Model **21**
5.1 Motivation 21
5.2 Proposed Model 23
5.3 Model Architecture 25
5.4 Analysis 26
5.5 Experiments 26
5.5.1 Toy Datasets 27
5.5.2 Real World Datasets 30
5.6 Summary 33

6 Set Representation Learning using Graph Neural Networks **34**
6.1 Proposed Method 35
6.2 Experiments 38
6.2.1 MUSK1 Dataset 38
6.2.2 LUAD vs LUSC Classification. 39
6.2.3 Inference 40
6.3 Summary 43

7 Pay Attention with Focus: Hierarchical Set Learning **44**
7.1 Proposed Method 44
7.2 Results 47
7.3 Summary 51

8 Summary and Conclusions **53**

8.1 Future Work 54

References **55**

List of Figures

1.1 Caption for LOF 4

4.1 Example of Histopathology Images: The patches extracted from two WSIs of patients with (a) LUAD and (b) LUSC. Each slide roughly contains 500 patches. 18

5.1 An exemplar application of learning permutation invariant representation for disease classification of Whole-Slide Images (WSIs). (a) A set of patches are extracted from each WSI of patients with lung cancer. (b) The sets of patches are fed to the proposed model for classification of the subtype of lung cancer—LUAD versus LUSC. The model classifies on a per set basis. This form of learning is known as Multi Instance Learning (MIL). 22

5.2 X is an input sequence containing n number of f-dimensional vectors. (a) The **memory block** is a sequence-to-sequence model that takes X and returns another sequence $\hat{X}$. The output $\hat{X}$ is a permutation-invariant representation of X. A bijective transformation model (an autoencoder) converts the input X to a permutation-equivariant sequence C. The weighted sum of C is computed over different probability distributions p_i from memory units. The hyper-parameters of a memory block are i) dimensions of the bijective transformation h, and ii) number of memory units m. (b) The **memory unit** has A_i, an embedding matrix (trainable parameters) that transforms elements of X to a d-dimensional space (memories). The output p_i is a probability distribution over the input X, also known as attention. The memory unit has a single hyper-parameter d, i.e. the dimension of the embedding space. (* represents learnable parameters.) 23

5.3 The overall architecture of the proposed Memory-based Exchangeable Model (MEM). The input to the model is a sequence, for e.g., a sequence of images or vectors. Each element of the input sequence X is passed through **(a)** feature extractor (CNN or MLP) to extract a sequence of feature vectors F, which is passed to **(c)** sequentially connected memory blocks. A memory block outputs another sequence which is a permutation-invariant representation of the input sequence. The output from the last memory block is vectorized and given to **(c)** MLP layers for classification/regression. 25

5.4 Comparison of MEM and feature pooling on a regression problem involving finding the sum of even digits within a set of MNIST images. Each point corresponds to the best configurations for the two models. 28

5.5 The patches extracted from two WSIs of patients with (a) LUAD and (b) LUSC. Each slide roughly contains 500 patches. 31

6.1 **Transforming a WSI to a fully-connected graph.** A WSI is represented as a graph with its nodes corresponding to distinct patches from the WSI. A node feature (a blue block beside each node) is extracted by feeding the associated patch through a deep network. A single context vector, summarizing the entire graph is computed by pooling all the node features. The context vector is concatenated with each node feature, subsequently fed into adjacent learning block. The adjacent learning block uses a series of dense layers and cross-correlation to calculate the adjacency matrix. The computed adjacency matrix is used to produce the final fully-connected graph. In the figure, the thickness of the edge connecting two nodes corresponds to the value in the adjacency matrix. 36

6.2 **Classification of a graph representing a WSI.** A fully connected graph representing a WSI is fed through a graph convolution layer to transform it into another fully-connected graph. After a series of transformations, the nodes of the final fully-connected graph are aggregated to a single condensed vector, which is fed to an MLP for classification purposes. 37

6.3 Six patches from two WSIs diagnosed with LUSC and LUAD, respectively. The six patches are selected, such that the first three (top row) are highly “attended” by the network, whereas the last three (bottom row) least attended. The first patch in the upper row is the most attended patch (more important) and the first patch in the lower row in the least attended patch (less important). 40

6.4 t-SNE visualization of feature vectors extracted after the Graph Pooling layer from different WSIs. The two distinct clusters for LUAD and LUSC demonstrate the efficacy of the proposed model for disease characterization in WSIs. The overlap of two clusters contain WSIs that are morphologically and visually similar. 41

6.5 The ROC curve of prediction. 42

7.1 **Training a Feature Extractor.** A feature extractor is trained with hierarchical target labels of a WSI. (a) A set of representative WSI patches (called mosaic) is extracted [43]. (b) The patches are used to fine-tune a deep network; each patch is assigned the parent WSI's labels, i.e., anatomic site and primary diagnosis. 45

7.2 **Classification of WSIs with FocAtt-MIL.** The two-stage method for the classification of WSI. (a) The mosaic of a WSI is converted to a bag X containing a set of feature vectors $\{x_1, \ldots, x_n\}$. (b) The feature vectors in a bag X are transformed to the primary diagnosis probability through FocAtt-MIL. The prediction probability p_i is computed for an individual feature vector x_i. A WSI context g_X is computed for the entire bag X using (7.1). The WSI context g_X is used to compute the attention value a_i and the focal factor γ. The final prediction is computed using (7.2). 46

7.3 **Attention Visualization.** The attention values are augmented on the two exemplars WSIs. **Left Image (LUAD):** Regions of high importance come from the cancerous regions while sparing normal lung tissue, fibrosis, and mucin deposition. Additionally, by inspecting important regions at a higher magnification, it is noticeable that the malignant glandular formations border with non-malignant areas. **Right Image (LUSC):** Regions that are considered to be important for classification are composed of malignant squamous cells. However, unlike LUAD, the attention model seems to be responsive to regions with solid malignant structures. 49

7.4 **FocAtt-MIL Training.** The loss and accuracy on validation dataset during the training of three different configurations of FocAtt-MIL, i.e FocAtt-MIL-DN, FocAtt-MIL-KimiaNet, and FocAtt-MIL-FDN. 50

List of Tables

5.1 Results on the toy datasets for different configurations of MEM and feature pooling. It must be noted that for Maximum of Set, the configuration FF + Max (DS) achieves the best accuracy but it may predict the output perfectly by learning the identity function therefore we highlighted second best configuration FF + Dotprod (DS) as well. 27

5.2 Test accuracy for the point cloud classification on different instance sizes using various methods. MEM with configuration FF + MEM + MB1 achieves 85.21% accuracy for the instance size of 100 which is best compared to others. 30

5.3 Accuracy for LUAD vs LUSC classification for various methods. For our experiments, we conducted comprehensive 5-fold cross validation accuracy whereas other methods have used non-standardized test set. 32

6.1 Evaluation on MUSK1. 39

6.2 Performance of various methods for LUAD/LUSC predictions using transfer learning. Our results report the average of 5-fold accuracy values. 40

6.3 Comparison of different network architecture and pooling method (attention, mean, max and sum pooling). **BN** stands for BatchNormalization [37], **Cheb** stands for Chebnet with corresponding filter size and **SAGE** stands for SAGE Convolution. The best performing configuration is Cheb-7 with mean pooling. 42

7.1 Performance comparison for LUAD/LUSC classification via transfer learning. 47

7.2 Pan-cancer vertical classification accuracy of FocAtt-MIL for features from regular DenseNet (FocAtt-MIL-DN), KimiaNet (FocAtt-MIL-KimiaNet), and DenseNet fine-tuned with hierarchical labels (FocAtt-MIL-FDN). 52

Chapter 1

Introduction

The performance of a Machine Intelligence model depends on the data representation. Before Deep Neural Networks (DNNs), hand-crafted kernels were most commonly used to extract features as a representation method. Although hand-crafted kernels can incorporate human and domain knowledge, they are labor-intensive and not scalable to high-dimensional datasets. Hand-crafted feature usage limits applications to only simple datasets such as MNIST [54] which contains only 784-dimensional feature vectors for a grayscale image. Most of the real-world images are much more complex than MNIST. It is thus crucial to make representation learning algorithms less dependent on hand-crafted features such that machines can learn efficient and better representations for real-world high dimensional data. The success of DNNs is due to their ability to learn efficient data representations without any expert/domain knowledge. This is achieved by using the backpropagation algorithm to minimize the loss. In this chapter, a brief background about Representation Learning is presented, which follows by problem definition.

1.1 Background

Before discussing the current state-of-the-art representation learning methods, it is important to understand what constitutes a good representation or feature. Bengio et al. [7] studied the problem of representation learning for various tasks such as vision and NLP. Bengio et al. in [7] postulated few characteristics important for a 'good 'representation:

1. **Smoothness**: For two inputs x, y such that $x \approx y$, smoothness requires that the corresponding representation of x and y are also similar i.e., $f(x) \approx f(y)$. Smoothness

is the most basic prior in most machine learning algorithms but is insufficient to get around the curse of dimensionality.

2. **Natural Clustering**: A good representation should be easily separable into clusters of different labels and classes using simple clustering algorithms such as K-means.

3. **Manifolds**: Probability mass concentrates near regions that have a much smaller dimensionality than the original space where the data lives. Any Machine Learning method should learn the manifold of the data for better generalization.

4. **Sparsity**: For any observation x, it is possible to represent it with only a fraction of total features or dimensions, i.e., most of the features are zero. Generally, sparse representations are associated with better generalization [65].

5. **Reconstruction**: For any input signal x and its representation $f(x)$, it should be possible to recover the input signal from the representation $f(x)$. This principle is used in Auto-encoders proposed in [31].

6. **Disentanglement**: Disentanglement refers to breaking down or disentangling each feature into narrowly defined variables and encoding them as separate dimensions in an unsupervised way. The objective is to mimic a human brain's function, which disentangles the underlying generative process from the data. For example, the human brain disentangles a visual image into shape, color, size, and orientation. Most of the time, explanatory factors of the data vary independently of each other in the input distribution, and only a few changes when one considers a sequence of consecutive real-world inputs.

With their success for various machine intelligence tasks, DNNs has become the de facto standard for extracting features from high dimensional data such as images. Many methods have been proposed to learn better data representation using DNNs. In contrast to hand-crafted kernels, DNNs use backpropagation [56] to minimize the loss functions. Various architectures are used for learning representation for different types of information/data. *Multiple Layer Perceptrons (MLPs)* are feed-forward networks where each neuron in a layer is connected to all neurons in the previous layer through weights and biases. MLPs can efficiently learn representation for simple vectorized data which don't contain spatial information. *Convolution Neural Networks (CNNs)* were proposed by LeCun et al. in [55] to learn representation from visual information (images). CNNs are similar to conventional feed-forward neural networks, but they make the implicit assumption that inputs are images and thus use filters to learn feature maps. Weights are incorporated in kernel/filters, which are learned during the training via backpropagation. CNNs, through application of many filters/kernels, can capture spatial information in images more efficiently and thus are used for learning image representation. Large scale CNNs such as AlexNet [53], ResNet [30], and DenseNet [33] have achieved state of the art results on many visuals tasks. *Recurrent Neural Networks (RNNs)* were proposed by Hochreiter et al. [32] to learn representation for ordered serial information such as texts and have become the standard framework for Natural Language Processing (NLP). RNNs consist of Long Short Term Memory (LSTM) units, composed of a cell, an input gate, an output gate, and a forget gate. The cell remembers values over arbitrary time intervals, and the three gates regulate the flow of information into and out of the cell. RNNs use past information along with current information to learn representation. This is important for NLP as words are often connected and context-dependent. Another type of architecture is autoencoder, often used for denoising and representation learning [5]. Autoencoders are based on the principle that a 'good' representation can recover the original input signal. It consists of two sub-networks; an encoder and a decoder. Encoder learns the latent representation $e(x)$ of the input signal x, and the decoder's objective is to reconstruct the original signal from the latent representation $e(x)$ by minimizing the Least Square error between the original signal x and recovered signal $\tilde{x}$. There has also been significant progress in generative modeling in the last decade. Generative Adversarial Networks (GAN) was proposed by Goodfellow et al. [22] uses adversarial loss to learn the manifold of the data for generating realistic-looking images. Another popular class of generative models are Variational Auto-Encoders (VAE) [48]. VAE uses Evidence Lower Bound Loss instead of Adversarial loss to learn the data distribution. Generative Models also find applications in unsupervised representation learning.

Despite significant progress, all the above-mentioned representation learning methods, except RNNs, take a single instance as an input. RNNs take a *ordered* set, e.g., language sentences, as an input. However, not much work has been done for learning representation for *unordered* or *exchangeable* sets.

1.2 Problem Definition

Recurrent Neural Networks (RNNs) are a popular approach to learn representations from sequentially ordered sets. However, the lack of permutation invariance renders RNNs ineffective for learning set representation. Sets can be mathematically described as exchangeable or unordered sequences, i.e., sequences in which order (permutation) of instances do not matter. For learning representations for such sequences, the model needs to be permu-

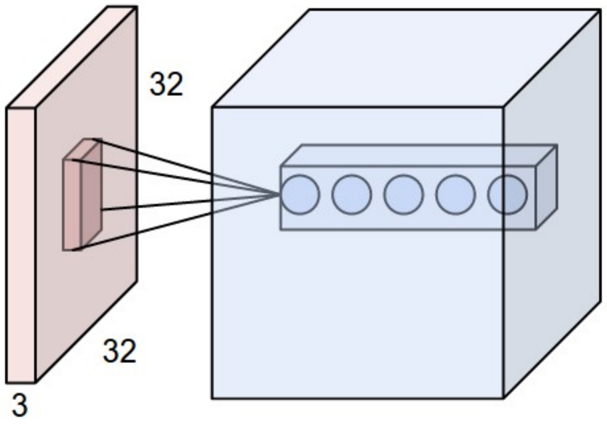


Figure 1.1: An example of a CNN. Each neuron in the convolutional layer is connected only to a local region in the input volume spatially, but to the full depth (i.e., all color channels). *Source*: Stanford CS231n

tation invariant. Set Representation Learning finds application in various scenarios, from gigapixel histopathology images to nano-scale quantum chemistry.

Any set representation learning algorithm must satisfy the following properties:

1. Permutation Invariance: This property implies that the order of elements in the set doesn't matter, i.e., permuting elements should not change the output.

2. Learning Inter-Dependencies: This refers to learning relations between different elements of the sets. It may be possible only a few elements are relevant for a specific task. For example, in digital pathology, non-cancerous patches don't provide any relevant information for diagnosis.

3. Universal Approximation: This implies that the algorithm should be capable of learning any set function or any mapping $\mathbb{R}^{N\text{x}d} \mapsto \mathbb{R}^m$, where N is the number of elements in the set.

1.3 Contributions

In this work, three novel frameworks are proposed for Set Representation Learning [2, 42, 41]. A new Permutation Invariant Model using Memory Networks was developed [42] called '*MEM*'. The thesis other main contribution is to explore the use of Graph Neural Networks for Set Representation Learning [2]. The thesis also introduces a hierarchical set representation learning framework in [41]. This is the first work that has combined both Hierarchical and Set Representation Learning to the best of our knowledge. The methods have been evaluated on various synthetic and real-world datasets achieving state-of-the-art accuracy in many tasks. Brief descriptions of proposed methods are given below.

Memory based Exchangeable Model (MEM): Many real-world tasks such as the classification of digital histopathology images and 3D object detection involve learning from a set of instances. In these cases, only a group of instances or a set collectively contains meaningful information, and therefore only the sets have labels, not individual data instances. A new permutation invariant neural network called *Memory-based Exchangeable Model (MEM)* is proposed for learning universal set functions. The MEM model consists of memory units that embed an input sequence to high-level features enabling it to learn inter-dependencies among instances through a self-attention mechanism. The learning ability of MEM is evaluated on various toy datasets, point cloud classification, and classification of whole slide images (WSIs) into two subtypes of lung cancer—Lung Adenocarcinoma, and Lung Squamous Cell Carcinoma. Patches were systematically extracted from WSIs of the lung, downloaded from The Cancer Genome Atlas (TCGA) dataset, the largest public repository of WSIs, achieving a competitive accuracy of 84.84% for the classification of two subtypes of lung cancer. The results on other datasets are promising as well and demonstrate the efficacy of the model [42].

Set Representation Learning using Graph Neural Networks : Representation learning for Whole Slide Images (WSIs) is pivotal in developing image-based systems to achieve higher precision in diagnostic pathology. The thesis proposes a two-stage framework for WSI representation learning. The algorithm first sample relevant patches using a color-based method and use graph neural networks to learn relations among sampled patches to aggregate the image information into a single vector representation. The thesis introduces attention via graph pooling to automatically infer patches with higher relevance. Proposed approach is validated for cancer subtype classification, Lung Adenocarcinoma (LUAD) Lung Squamous Cell Carcinoma (LUSC). 1,026 lung cancer WSIs with the $40\times$ magnification were collected from The Cancer Genome Atlas (TCGA) dataset, the largest public repository of histopathology images, and achieved state-of-the-art accuracy of 88.8% and AUC of 0.89 on lung cancer subtype classification by extracting features from a pre-trained DenseNet model [2].

Pay Attention with Focus: A Novel Hierarchical Set Learning Framework: A novel two-stage hierarchical framework for Set Representation learning of Histopathology Images is proposed. First, set of representative patches (called mosaic) are extracted from a WSI. Each patch of a mosaic is encoded to a feature vector using a deep network. The feature extractor model is fine-tuned using hierarchical target labels of WSIs, i.e., anatomic site and primary diagnosis. In the second stage, a set of encoded patch-level features from a WSI is used to compute the primary diagnosis probability through the proposed Pay Attention with Focus scheme, an attention-weighted averaging of predicted probabilities for all patches of a mosaic modulated by a trainable focal factor. Experimental results show that the proposed model is robust, and effective for the classification of WSIs [41].

The thesis is divided into multiple parts. The necessary background about Permutation Invariant Model is discussed in chapter 2, Graph Neural Networks in chapter chapter 3. chapter 5 discusses the proposed Memory based Exchangeable Model in detail. The Graph Neural Network framework is discussed in chapter 6 and hierarchical set learning framework in chapter 7. It is followed by chapter 8, summarizing key contributions and future works.

Part I

Background Literature

Chapter 2

Permutation Invariant Models

Deep artificial neural networks have achieved impressive performance for representation learning tasks. The majority of these deep architectures take a single instance as an input. Recurrent Neural Networks (RNNs) are a popular approach to learn representations from sequentially ordered instances. However, the lack of permutation invariance renders RNNs ineffective for *exchangeable* or *unordered sequences*. We often need to learn representations of unordered sequential data or exchangeable sequences in many practical scenarios such as Multiple Instance Learning (MIL), 3D Computer Vision. In these cases, only a *group* or a *set* of instances or a set, collectively, contains meaningful information, not individual data instances. Representations for such data can be learned using Permutation Invariant Models.

In this chapter, the mathematical definition for Permutation Invariant Models and Exchangeability, its application, and recent works will be discussed.

2.1 Background

In statistics, exchangeability has been long studied. de Finetti studied exchangeable random variables and showed that a sequence of infinite exchangeable random variables can be factorized to independent and identically distributed mixtures conditioned on some parameter θ. Bayesian sets [21] introduced a method to model exchangeable sequences of binary random variables by analytically computing the integrals in de Finetti's theorem. Orbanz et al. [67] used de Finetti's theorem for Bayesian modeling of graphs, matrices, and other data that can be modeled by random structures. Considerable work has also been done on partially exchangeable random variables [3].

Symmetry in neural networks was first proposed by Shawe et al. [72] under the name *Symmetry Network*. They proposed that invariance can be achieved by weight-preserving automorphisms of a neural network. Ravanbaksh et al. proposed a similar method for equivariance network through parameter sharing [69]. Bloem Reddy et al. [9] studied the concept of symmetry and exchangeability for neural networks in detail and established similarity between functional and probabilistic symmetry, and obtained generative functional representations of joint and conditional probability distributions that are invariant or equivariant under the action of a compact group. Zhou et. al. [99] proposed treating instances in a set as non-identical and independent samples for multi-instance problems.

Most of the work published in recent years has focused on ordered sets. Vinyals et al. [84] introduced Order Matter: Sequence to Sequence for Sets to learn a sequence to sequence mapping. Many related models and key contributions have been proposed that uses the idea of external memories like RNNSearch [4], Memory Networks [87, 83] and Neural Turing Machines [24]. Recent interest in exchangeable models was developed due to their application in MIL. Deep Symmetry Networks [20] used kernel-based interpolation to tractably tie parameters and pool over symmetry spaces of any dimension. Deep Sets [94] by Zaheer et al. proposed a permutation invariant model. They proved that any pooling operation (mean, sum, max or similar) on individual features is a universal approximator for any set function. They also showed that any permutation invariant model follows de Finetti's theorem. Work has also been done on learning point cloud classification, which is an example of MIL problem. Deep Learning with Sets and Point Cloud [68] used parameter sharing to get an equivariant layer. Another important paper on the exchangeable model is Set Transformer. Set Transformer [57] by Lee et al. used results from Zaheer et al. [94] and proposed a Transformer [83] inspired permutation invariant neural network. The Set Transformer uses attention mechanisms to attend to inputs in order to invoke activation. Instead of using averaging over instances like in Deep Sets, the Set Transformer uses a parametric aggregating function pool that can adapt to the problem at hand. Another way to handle exchangeable data is to modify RNNs to operate on exchangeable data. BRUNO [51] is a model for exchangeable data and makes use of deep features learned from observations so as to model complex data types such as images. To achieve this, they constructed a bijective mapping between random variables $x_i \in X$ in the observation space and features $z_i \in Z$, and explicitly define an exchangeable model for the sequences $z_1, z_2, z_3, \ldots, z_n$. Deep Amortized Clustering [58] proposed using Set Transformers to cluster sets of points with only a few forward passes. Deep Set Prediction Networks [98] introduced an interesting approach to predict sets from a feature vector which is in contrast to predicting an output using sets.

2.2 Exchangeablilty

This section explains the general concepts of exchangeability, its relation to de Finetti's theorem.

Exchangeable Sequence. A sequence of random variables $x_1, \ldots, x_n$ is exchangeable if the joint probability distribution does not change on permutation of the elements in a set. Mathematically, if $P(x_1, \ldots, x_n) = P(x_{\pi(1)}, \ldots, x_{\pi(n)})$ for a permutation function π, then the sequence $x_1, \ldots, x_n$ is exchangeable.

Exchangeable Models. A model is said to be Exchangeable or Permutation Invariant if the output of the model is invariant to the permutation of its inputs. Exchangeability implies that the information provided by each instance x_i is independent of the order in which they are presented. Exchangeable models can be of two types depending on the application: i) permutation invariant, and ii) permutation equivariant.

A model represented by a function $f : X \rightarrow Y$ where X is a set, is said to be permutation equivariant if permutation of input instances permutes the output labels with the same permutation π. Mathematically, a permutation-equivariant model is represented as,

$$f(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}) = [y_{\pi(1)}, y_{\pi(2)}, \ldots, y_{\pi(n)}]. \tag{2.1}$$

Similarly, a function is permutation invariant if permutation of input instances does not change the output of the model. Mathematically,

$$f(x_1, x_2, \ldots, x_n) = f(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}). \tag{2.2}$$

de Finetti's Theorem. In statistics, exchangeability has been long studied. de Finetti studied exchangeable random variables and showed that sequence of infinite exchangeable random variables can be factorized to independent and identically distributed mixtures conditioned on some parameter θ:

$$P(x_1, x_2 \ldots x_n) = \int p(\theta) \prod_{i=1}^{n} p(x_i \mid \theta) \mathrm{d}\theta, \tag{2.3}$$

where θ is some latent feature.

2.3 Deep Sets: Universal Approximation of Sets

Deep Sets [93] incorporate a permutation-invariant model to learn arbitrary set functions by pooling in a latent space. Any pooling operation such as averaging and max on individual instances of a set can be used as a universal approximator for any arbitrary set function. The authors proved the following two results about permutation invariant models.

Theorem 1. A function $f(x)$ operating on a set $X = \{x_1, \ldots, x_n\}$ having elements from a countable universe, is a valid set function, i.e., invariant to the permutation of instances in X, if it can be decomposed to $\rho\left(\sum \phi(x)\right)$, for any function ϕ and ρ.

Theorem 2. Assume the elements are from a compact set in $\mathbb{R}^d$, i.e., possibly uncountable, and the set size is fixed to M. Then any continuous function operating on a set X, i.e., $f : \mathbb{R}^{d\times M} \to \mathbb{R}$ which is permutation invariant to the elements in X can be approximated arbitrarily close in the form of $\rho \sum(\phi(x))$.

The Theorem 1 is linked to de Finetti's theorem, which states that a random infinitely exchangeable sequence can be factorized into mixture densities conditioned on some parameter θ which captures the underlying generative process in Equation 2.3.

Chapter 3

Graph Neural Networks

This chapter briefly discusses Graph Neural Networks, various types of Graph Convolution Neural Networks, Pooling Layers, and Graph Spectral Theory.

3.1 Introduction

Data in graphical representation occurs in many real-world applications such as social networks, knowledge graphs, protein-interaction networks, the World Wide Web, etc. Such datasets cannot be learned using conventional Deep Learning methods such as Convolution Neural Networks, Recurrent Neural Networks, etc. In recent years, much work has been done to generalize Deep Learning to such structured datasets. The central problem in machine learning on graphs is finding a way to incorporate graph structure into a machine learning model. For example, in the case of link prediction in a social network, one might want to encode pairwise properties between nodes, such as relationship strength or the number of common friends. Similarly, for node classification, one might want to include information about the global position of a node in the graph or the structure of the node's local graph neighborhood [28].

Graph Representation. A graph can be fully represented by its node list V and adjacency matrix $\mathbf{A}$. Graphs can model many types of relations and processes in physical, biological, social, and information systems. A connection between two nodes V_i and V_j is represented using an edge weighted by a_{ij}.

Graph Neural Networks (GNNs) is an effective framework for representation learning of graphs. Hamilton et al. in [28] discussed the emerging methods and applications of Graph Neural Networks. The objective is to learn a mapping from the graph or subgraphs to low dimensional vector space such that geometric relationships in the embedding space reflect the structure of the original graph. The learned vector transformations can then be used as inputs for machine learning tasks. The key difference between conventional approaches for learning graphs and GNN is that prior to GNN, hand-crafted statistics were used to extract features. GNN, in contrast to traditional approaches, treat this problem as a machine learning task itself, using a data-driven approach to learn embeddings that encode graph structure.

Depending on how the aggregation functions are learned, GNNs can be broadly classified as Spectral and Spatial based GNNs. Spectral and Spatial Graphs are briefly discussed in section 3.2.

3.2 Spectral and Spatial GNNs

GCNNs generalize the operation of convolution from grid data to graph data. A GCNN takes a graph as an input and transforms it into another graph as the output. Each feature node in the output graph is computed by aggregating features of the corresponding nodes and their neighboring nodes in the input graph. Like CNNs, GCNNs can stack multiple layers to extract high-level node representations. Depending upon the method for aggregating features, GCNNs can be divided into two categories, namely spectral-based and spatial-based.

Spectral GCNNs. Spectral-based approaches define graph convolutions by introducing filters from the perspective of graph signal processing. Spectral convolutions are defined as the multiplication of a node signal by a kernel. This is similar to the way convolutions operate on an image, where a pixel value is multiplied by a kernel value. Bruna et al. first combined the Spectral Analysis and Convolution Neural Networks giving rise to spectral graph convolutional networks that can be trained in a supervised way, for example, for the graph classification task. [11]. Spectral analysis of graph refers to the eigen-decomposition of the graph Laplacian matrix L. Normalized Laplacian matrix for a graph $G(V, E)$ can be mathematically defined as

$$L^{\text{sym}} := D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

where L is the normalized Laplacian, A is the adjacency matrix, and D is the degree matrix. The Laplacian allows a natural link between discrete representations, such as graphs and continuous representations, such as vector spaces and manifolds. Intuitively, the Laplacian of graph can be interpreted as 'Divergence of Gradient', which shows in what directions and how smoothly the energy will diffuse over a graph if we put some potential in node i. L can be factored as $L = U \Lambda U^T$ like the one below where U contains the eigenvectors of L, and Λ is a diagonal matrix containing the corresponding eigenvalues. These eigenvectors are also known as the graph Fourier modes, and the corresponded eigenvalues are the frequencies of the graph. Thus spectral graph convolution can be described as

$$x *_G g_\theta = U g_\theta U^T x$$

where filter g_θ is a diagonal matrix with learnable parameters θ.

One disadvantage of using spectral GCNNs is the cost of computing eigenvectors of the Laplacian matrix L and the non-localized spectral filters. Spectral filters are not localized and may not scale well to large graphs. Various approximations such as ChebNets (section 3.3) have been proposed to approximate spectral convolution.

Spatial GCNNs. Spatial-based approaches formulate graph convolutions as aggregating feature information from neighbors. Spatial graph convolution learns the aggregation function, which is permutation invariant to the ordering of the node. Spatial GCNNs learn different aggregation functions which combine information from each node's neighbors. Spatial GCNN can be viewed as message passing.

3.3 Graph Convolution Neural Networks (GCNNs)

Both spectral and spatial GCNNs are used to learn graph representation in chapter 6. ChebNet (Spectral GCNN) and GraphSAGE (Spatial GCNN) are implemented to compare the performance of our proposed algorithm. Ablation study for various hyper-parameters of GCNN is shown in Table 6.3.

ChebNet. It was introduced by Defferrard et al. [14]. Spectral convolutions on graphs are defined as the multiplication of a signal $x \in R^N$ (a scalar for every node) with a filter $g(\theta) = diag(\theta)$ parameterized by $\theta \in R^N$ in the Fourier domain, i.e.,

$$g_\theta \circledast x = U g_\theta U^T x,$$

where U is the matrix of eigenvectors of the normalized graph Laplacian $L = I_N - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$. This equation is computationally expensive to calculate as multiplication with the eigenvector matrix U is $O(N^2)$. Hammond et al. [29] suggested that g_θ can be well-approximated by a truncated expansion in terms of Chebyshev polynomials $T_k(x)$, i.e.,

$$g_{\theta'}(\Lambda) \approx \sum_{k=0}^{K} \theta' T_k(\Lambda).$$

The kernels used in ChebNet are made of Chebyshev polynomials of the diagonal matrix of Laplacian eigenvalues. ChebNet uses kernel made of Chebyshev polynomials. Chebyshev polynomials are a type of orthogonal polynomials with properties that make them very good at tasks like approximating functions.

GraphSAGE. Hamilton et al. in 2017 introduced a new graph learning framework called GraphSAGE [27]. GraphSAGE learns inductive node embedding based on matrix factorization and uses node features (e.g., text attributes, node profile information, node degrees) to learn an embedding function. Learned embedding function can generalize to unseen nodes by learning aggregation functions that can induce the embedding of a new node given its features and neighborhood. This method is known as inductive learning. GraphSAGE is a framework for inductive representation learning on large graphs that can generate low-dimensional vector representations for nodes and is especially useful for graphs with rich node attribute information. It is much faster to create embeddings for new nodes with GraphSAGE.

Graph Pooling Layers. Similar to CNNs, pooling layers in GNNs downsample node features by pooling operation. Evaluation is done with Global Attention Pooling, Mean Pooling, Max Pooling, and Sum Pooling. Global Attention Pooling [59] was introduced by Li et al. and uses a soft attention mechanism to decide which nodes are relevant to the current graph-level task and gives the pooled feature vector from all the nodes.

3.4 Set Representation Learning using Graphs

Set Representation Learning can be viewed as a graph learning problem where each element in the set can be considered a node in the graph. The main challenge of set representation learning is to learn inter-dependencies among elements of the set via adjacency matrix. Different methods have been proposed to learn the connectivity or the adjacency matrix. A new method is proposed which uses a fully connected graph with weighted edges in chapter 6. The weights of edges are learned in an end-to-end fashion as described in section 6.1. To the best of knowledge, this is an entirely new approach to learn gigapixel image representation.

Chapter 4

Gigapixel Histopathology Images

The success of deep learning has opened promising horizons for digital pathology. AI experts and pathologists are now working together to design novel image analysis algorithms. The last decade has witnessed the widespread adoption of digital pathology, leading to the emergence of machine learning (ML) models for analyzing Whole Slide Images (WSIs). Applications of ML in digital pathology include (i) reducing the workload on pathologists, and (ii) improving cancer treatment procedures [62]. The computational analysis of WSIs offers various challenges in terms of image size and complexity. These challenges necessitate the inquiry into more effective ways of analyzing WSIs. CNNs are at the forefront of computer vision, showcasing significant improvements over conventional methodologies for visual understanding [45]. However, CNNs can not be directly utilized for processing WSIs due to their large image dimensions. The majority of the recent work analyzes WSIs at the patch level that requires manual delineations from experts. These manual delineations reduce the feasibility of such approaches for real-world scenarios. Moreover, most of the time, labels are available for an entire WSI and not for individual patches [2]. Therefore, it is necessary to leverage the information present in all patches to learn a WSI representation. Representing WSI as a set of patches can be considered as a set representation learning problem. Hence, Set Representation Learning (MIL) is a promising venue for vision-related tasks for WSIs.

4.1 Background

CNN based methods for analyzing histopathological images is well represented in the literature [15, 13, 63, 19]. Deep learning methods generalize well across patients, disease

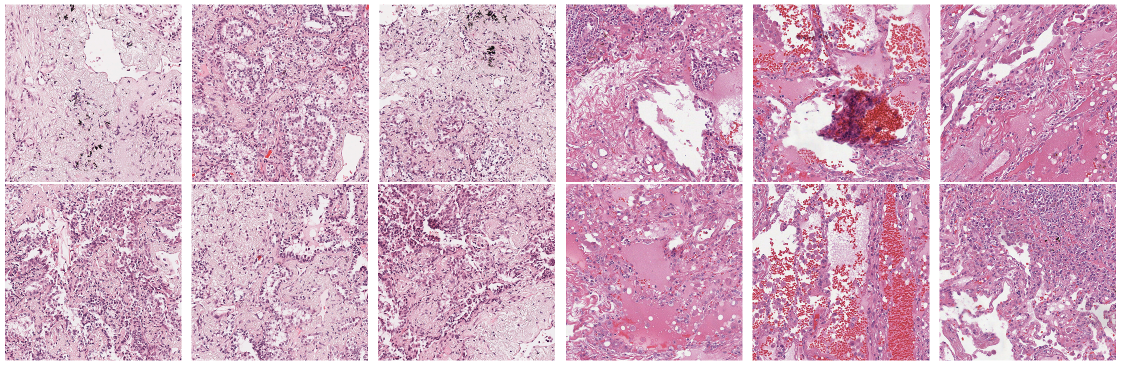

(a) Lung Adenocarcinoma (b) Lung Squamous Cell Carcinoma

Figure 4.1: Example of Histopathology Images: The patches extracted from two WSIs of patients with (a) LUAD and (b) LUSC. Each slide roughly contains 500 patches.

conditions, and are robust to the vendor or human-induced variations, especially when a large amount of training data is available [15]. A WSI usually contains at least two target labels, anatomic site, and primary diagnosis that are arranged in a hierarchy. The simplest way to deal with multi-label classification with k labels is to treat this as k independent binary classification. Although this approach may be helpful, it does not capture label dependencies. This limitation can degrade the performance in many applications where there is strong dependency among labels, for example, in WSI classification. To address this limitation, two different approaches, i.e., transformation and algorithm adaption methods, have been proposed [96]. In transformation-based methods, multi-label data is converted to new single label data to apply regular single-label classification. On the other hand, in the adaptation-based category, this is attempted to modify the basic single-label algorithm to handle multi-label data [77].

There are two main methods for characterizing WSIs [6]. The first method is called subsetting, which considers a small section of a large WSI as an essential region for analysis. On the other hand, the tiling method segments a WSI into smaller and controllable patches (i.e., tiles) [26]. The tiling or patch-based methods can benefit from Multiple Instance Learning (MIL). MIL is a weakly supervised algorithm where the label is available for a group or set of instances (patches for histopathology). MIL can be considered as a set representation learning problem and are well-suited for learning gigapixel image representation.

4.2 Multiple Instance Learning for WSIs

Multiple Instance Learning (MIL) algorithms assign a class label to a set of instances rather than to individual instances. The individual instance labels are not necessarily important, depending on the type of algorithm and its underlying assumptions [12]. Learning representation for histopathology images can be formulated as a MIL problem. Due to the intrinsic ambiguity and difficulty in obtaining human labeling, MIL approaches have their particular advantages in automatically exploiting the fine-grained information and reducing the efforts of human annotations. Exchangeable models or Permutation Invariant Models are useful for histopathological images analysis as ground-truth labeling is expensive, and labels are available at WSI instead of at the pixel level. A small pathology lab may process ≈10,000 WSIs/year, producing a vast amount of data, presenting a unique opportunity for MIL methods. Dismantling a WSI into smaller patches is a common practice; these patches can be used for MIL. The authors in [34] used attention-based pooling to infer important patches for cancer classification. A large amount of partially labeled data in histopathology can be used to discover hidden patterns of clinical importance [49]. Authors in [75] used MIL for breast cancer classification. A permutation invariant operator introduced by [80, 79] was applied to pathology images. Recently, graph CNNs have been successfully used for representation learning of WSIs [2]. These compact and robust representations of WSIs can be further used for various clinical applications such as image-based search to make well-informed diagnostic decisions [44, 43].

Part II

Proposed Methods

Chapter 5

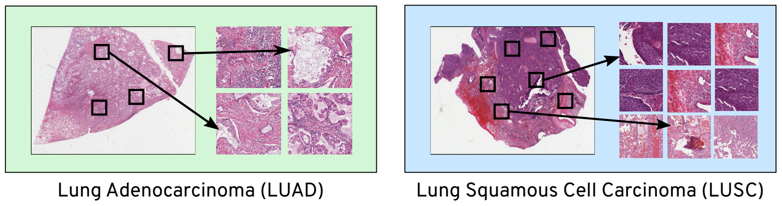
Memory-Based Exchangeable Model

In this chapter, a novel architecture for exchangeable sequences is proposed incorporating attention over the instances to learn inter-dependencies. Resultsfrom Deep Sets [94] are used to construct a permutation invariant model for learning set representations. The main contribution is a sequence-to-sequence permutation invariant layer called **Memory Block**. The proposed model uses a series of connected memory block layers to model complex dependencies within an input set using a self-attention mechanism. Model is validated using toy datasets and two real-world applications. The real-world applications include i) point cloud classification and ii) classification of WSI into two subtype of lung cancers—Lung Adenocarcinoma (LUAD)/ Lung Squamous Cell Carcinoma (LUSC) (see Figure 5.1).

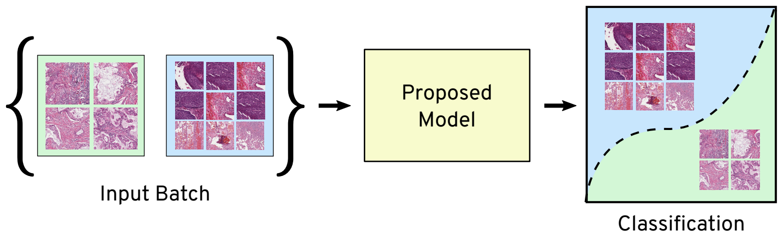
The chapter is structured as follows: section 5.1 discusses the motivation for developing an attention-based permutation invariant model. Approach and experimental results are explained in section 5.2 and section 5.5. Mathematical concepts for exchangeable models are covered in the previous chapter section 2.2.

5.1 Motivation

In order to learn an efficient representation for a set of instances, it is important to focus on instances that are "important" for a given task at hand, i.e., we need to attend to specific instances more than other instances. Therefore, memory network can be used to learn an attention mapping for each instance. Memory networks are conventionally used for NLP for mapping questions posted in natural language to an answer [87, 76]. The proposed method exploit the idea of having *memories* which can learn *key* features shared by one or


Lung Adenocarcinoma (LUAD)
Lung Squamous Cell Carcinoma (LUSC)

(a) Patch Extraction

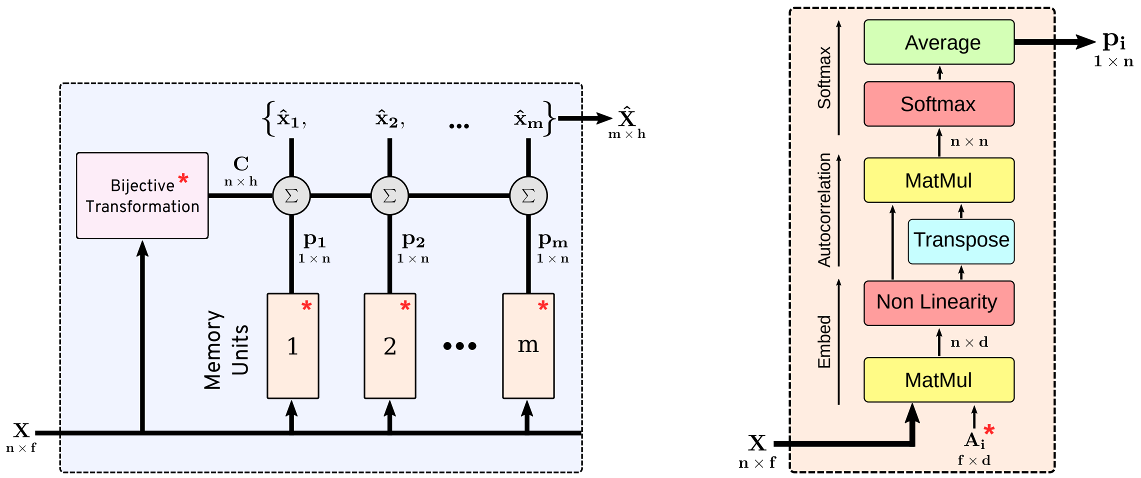


(b) Set Classification

Figure 5.1: An exemplar application of learning permutation invariant representation for disease classification of Whole-Slide Images (WSIs). (a) A set of patches are extracted from each WSI of patients with lung cancer. (b) The sets of patches are fed to the proposed model for classification of the subtype of lung cancer—LUAD versus LUSC. The model classifies on a per set basis. This form of learning is known as Multi Instance Learning (MIL).

more instances. Through these *key* features, the model can learn inter-dependencies using transformer style self-attention mechanism. As inter-dependencies are learned, a set can be condensed into a compact vector such that an MLP can be used for classification or regression learning.

Memory Networks. The idea of using an external memory for relational learning tasks was introduced by Weston et al. [87]. Later, an end-to-end trainable model was proposed by Sukhbaatar et al. [76]. Memory networks enable the learning of dependencies among instances of a set by providing an explicit memory representation for each instance in the sequence. The idea of self-attention is popularized by [83]; these models are known as *transformers*, widely used in NLP applications. The proposed MEM model uses the self-attention (similar to transformers) within memory vectors, aggregated using a pooling operation (weighted averaging) to form a permutation-invariant representation. The next section explains it in detail.



(a) memory block (b) memory unit

Figure 5.2: X is an input sequence containing n number of f-dimensional vectors. (a) The **memory block** is a sequence-to-sequence model that takes X and returns another sequence $\hat{X}$. The output $\hat{X}$ is a permutation-invariant representation of X. A bijective transformation model (an autoencoder) converts the input X to a permutation-equivariant sequence C. The weighted sum of C is computed over different probability distributions p_i from memory units. The hyper-parameters of a memory block are i) dimensions of the bijective transformation h, and ii) number of memory units m. (b) The **memory unit** has A_i, an embedding matrix (trainable parameters) that transforms elements of X to a d-dimensional space (memories). The output p_i is a probability distribution over the input X, also known as attention. The memory unit has a single hyper-parameter d, i.e. the dimension of the embedding space. (* represents learnable parameters.)

5.2 Proposed Model

MEM is composed of four sequentially connected units: i) a feature extraction model, ii) memory units, iii) memory blocks, and iv) fully connected layers to predict the output.

A *memory block* is the main component of MEM and learns a permutation invariant representation of a given input sequence. Multiple memory blocks can be stacked together for modeling complex relationships and dependencies in exchangeable data. The memory block is made of memory units and a bijective transformation unit shown in Figure 5.2

Memory Unit. A memory unit transforms a given input sequence to an attention vector.

The higher attention value represents higher “importance” of the corresponding element of the input sequence. Essentially, it captures the relationships among different elements of the input. Multiple memory units enable the memory block to capture many complex dependencies and relationships among the elements. Each memory unit consists of an embedding matrix $\mathbf{A_i}$ that transforms a f-dimensional input vector x_j to a d-dimensional memory vector u_{ij}, as follows:

$$u_{ij} = \rho(x_j \mathbf{A_i}),$$

where ρ is some non-linearity. The memory vectors are stacked to form a matrix $\mathbf{U_i} = [u_{i0}, \ldots, u_{in}]$ of the shape $(n \times d)$. The relative degree of correlations among the memory vectors are computed using cross-correlation followed by a column-wise softmax and then taking a row-wise average, as follows:

$$\begin{aligned} S_i &= \text{column-wise-softmax}(\mathbf{U_i U_i^T}), \\ p_i &= \text{row-wise-average}(S_i), \end{aligned} \tag{5.1}$$

The p_i is the final output vector $(1 \times n)$ from the i^{th} memory unit $\mathbf{U_i}$, as shown in Figure 5.2. The purpose of memory unit is to embed feature vectors into another space that could correspond to a distinct “attribute” or “characteristic” of instances. The cross correlation or the calculated attention vector represents the instances which are highly suggestive of those “attributes” or “characteristic”. The proposed method do not normalize memory vectors as magnitude of these vectors may play an important role during the cross correlation.

Memory Block. A memory block is a sequence-to-sequence model, i.e., it transforms a given input sequence $X = x_1, \ldots, x_n$ to another representative sequence $\hat{X} = \hat{x}_1, \ldots, \hat{x}_m$. The output sequence is invariant to the element-wise permutations of the input sequence. A memory block contains m number of memory units. In a memory block, each memory unit takes a sequential data as an input and generates an attention vector. These attention vectors are subsequently used to compute the final output sequence. The schematic diagram of a memory block is shown in Figure 5.2a.

The final output sequence $\hat{X}$ of a memory block is computed as a weighted sum of $\mathbf{C}$ with the probability distributions $p_1, \ldots, p_m$ from all the m memory units where $\mathbf{C}$ is a bijective transformation of X learned using an autoencoder. Each memory block has its own autoencoder model to learn the bijective mapping. The i^{th} element $\hat{x}_i$ of the output sequence $\hat{X}$ is computed as matrix multiplication of p_i and $\mathbf{C}$, as follows:

$$\hat{x}_i = p_i \mathbf{C},$$

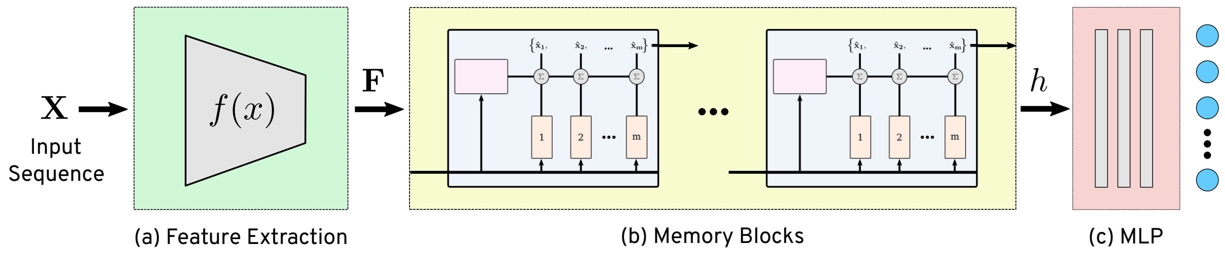


Figure 5.3: The overall architecture of the proposed Memory-based Exchangeable Model (MEM). The input to the model is a sequence, for e.g., a sequence of images or vectors. Each element of the input sequence X is passed through **(a)** feature extractor (CNN or MLP) to extract a sequence of feature vectors F, which is passed to **(c)** sequentially connected memory blocks. A memory block outputs another sequence which is a permutation-invariant representation of the input sequence. The output from the last memory block is vectorized and given to **(c)** MLP layers for classification/regression.

where, p_i is the output of i^{th} memory unit given by (5.1).

The bijective transformation from $X \mapsto C$ enables equivariant correspondence between the elements of the two sequences X & $\hat{X}$, and maps two different elements in the input sequence to different elements in the output sequence. It must be noted that bijective transformation is permutation equivariant, not invariant. The reconstruction maintains a one-to-one mapping between X and C. The final output sequence from a memory block is permutation invariant as it uses matrix multiplication between p_i (attention) and C.

5.3 Model Architecture

The overall architecture is shown in Figure 5.3.

1. Each element of a given input sequence $X = x_1, \ldots, x_n$ is passed through a feature extraction model to produce a sequence of feature vectors $F = f_1, \ldots, f_n$.

2. The feature sequence F is then passed through a memory block to obtain another sequence $\hat{X}$ which is a permutation-invariant representation of the input sequence. The number of elements in the sequence $\hat{X}$ depends on the number of memory units in the memory block layer.

3. Multiple memory blocks can be stacked in series. The output from the last memory block is either vectorized or pooled, which is subsequently passed to a MLP layer for classification or regression.

5.4 Analysis

This section discusses the mathematical properties of our model and uses theorems from Deep Sets [93] to prove that the model is a permutation invariant and universal approximator for arbitrary set functions.

Property 1. Memory units are permutation equivariant.

Consider an input sequence $X = x_1 \dots x_n$. Since, for each memory unit,

$$\mathbf{U_i} = [\rho(x_o\mathbf{A_i}), \rho(x_1\mathbf{A_i}), \dots, \rho(x_n\mathbf{A_i})]$$

By Equation (2.1), $\mathbf{U_i}$ is permutation equivariant and thus S_i in (5.1) is permutation equivariant. Finally, the attention vector p_i is calculated by averaging all rows, therefore the final output of memory unit p_i is permutation equivariant.

Property 2. Memory Blocks are permutation invariant.

A memory block layer consisting of m memory units generates a sequence $\hat{X} = \hat{x}_1, \dots, \hat{x}_m$ where $\hat{x_i}$ can be written as

$$\hat{x_i} = p_i\mathbf{C}$$

Since both $\mathbf{C}$ and p_i are permutation equivariant, therefore, $\hat{x_i}$, which is calculated by matrix multiplication of p_i and $\mathbf{C}$, is permutation invariant.

5.5 Experiments

Two series of experiments are performed comparing MEM against the simple pooling operations proposed by Deep Sets [94]. The first series of experiments establish the learning ability of the proposed model using toy datasets. For the second series, two real-world dataset are used, i) classification of subtypes of lung cancer against the largest public

1.5 Methods	Sum of Even Digits		Prime Sum	Counting Unique Images		Maximum of Set		Gaussian Clustering
	Accuracy	MAE	Accuracy	Accuracy	MAE	Accuracy	MAE	NLL
FF + MEM + MB1 (ours)	0.9367 ± 0.0016	0.2516 ± 0.0105	0.9438 ± 0.0043	0.7108 ± 0.0084	0.3931 ± 0.0080	0.9326 ± 0.0036	0.1449 ± 0.0068	**1.348**
FF + MEM + Mean (ours)	0.9355 ± 0.0015	0.2437 ± 0.0087	0.7208 ± 0.0217	0.4264 ± 0.0062	0.9525 ± 0.0109	0.9445 ± 0.0035	0.1073 ± 0.0067	1.523
FF + MEM + Max (ours)	**0.9431 ± 0.0020**	**0.2295 ± 0.0098**	0.9361 ± 0.0060	0.6888 ± 0.0066	0.4140 ± 0.0079	0.9498 ± 0.0022	0.1086 ± 0.0060	1.388
FF + MEM + Dotprod (ours)	0.8411 ± 0.0045	0.3932 ± 0.0065	**0.9450 ± 0.0086**	**0.7284 ± 0.0055**	**0.3664 ± 0.0037**	**0.9517 ± 0.0041**	**0.0999 ± 0.0097**	1.363
FF + MEM + Sum (ours)	0.9353 ± 0.0022	0.2739 ± 0.0081	0.6652 ± 0.0389	0.3138 ± 0.0094	1.3696 ± 0.0151	0.9430 ± 0.0031	0.1318 ± 0.0058	1.611
FF + Mean (DS)	0.9159 ± 0.0019	0.2958 ± 0.0049	0.5280 ± 0.0078	0.3140 ± 0.0071	1.2169 ± 0.0136	0.3223 ± 0.0075	1.0029 ± 0.0155	2.182
FF + Max (DS)	0.6291 ± 0.0047	1.3292 ± 0.0211	0.9257 ± 0.0033	0.7088 ± 0.0060	0.3933 ± 0.0059	**0.9585 ± 0.0012**	**0.0742 ± 0.0032**	1.608
FF + Dotprod (DS)	0.1503 ± 0.0015	1.8015 ± 0.0016	0.9224 ± 0.0028	**0.7254 ± 0.0063**	**0.3726 ± 0.0054**	**0.9548 ± 0.0017**	0.1355 ± 0.0027	8.538
FF + Sum (DS)	0.6333 ± 0.0043	0.5763 ± 0.0069	0.5264 ± 0.0050	0.2982 ± 0.0042	1.3415 ± 0.0169	0.3344 ± 0.0038	0.9645 ± 0.0111	12.05
1.5								

Table 5.1: Results on the toy datasets for different configurations of MEM and feature pooling. It must be noted that for Maximum of Set, the configuration FF + Max (DS) achieves the best accuracy but it may predict the output perfectly by learning the identity function therefore we highlighted second best configuration FF + Dotprod (DS) as well.

dataset of histopathology whole slide images (WSIs) [86], and ii) 3-D object classification using Point Cloud Dataset [89].

Model Comparison. The performance of MEM is compared against Deep Sets [94]. Same feature extractor is used for both Deep Sets and MEM, and experimented with different choices of pooling operations—max, mean, dot product, and sum. MEM also has a special pooling "$mb1$", which is a memory block with a single memory unit in the last hidden layer. Therefore, nine different models are tested for each experiment—five configurations of our model, and four configurations of Deep Sets. Hyper-parameters tuning is done to achieve the best performance for each configurations of both MEM and Deep Sets. MEM is found to have a higher learning capacity, therefore, higher number of parameters resulted in better accuracy for MEM but not necessarily for Deep Set. Common feature extractor is denoted as **FF** and Deep Sets as **DS** in the discussion below. The other approaches that are compared have been appropriately cited.

5.5.1 Toy Datasets

To demonstrate the advantage of MEM over simple pooling operations, the work considers four toy problems involving regression and classification over sets. The toy datasets are constructed using the MNIST dataset.

Sum of Even Digits. Sum of even digits is a regression problem over the set of images containing handwritten digits from MNIST. For a given set of images $X = \{x_1, \ldots, x_n\}$, the goal is to find the sum of all even digits. Loss used is Mean Absolute Error (MAE).

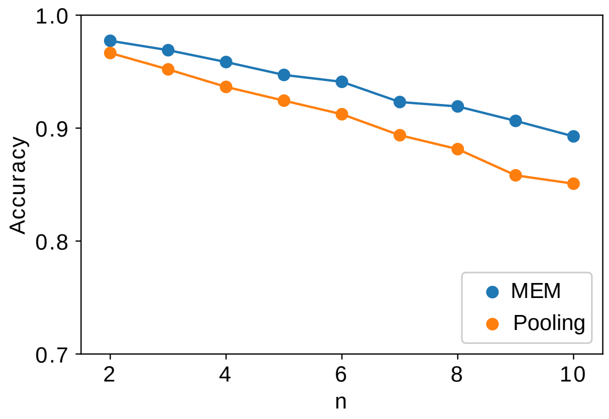


Figure 5.4: Comparison of MEM and feature pooling on a regression problem involving finding the sum of even digits within a set of MNIST images. Each point corresponds to the best configurations for the two models.

MNIST dataset is splitted into 70-30% training, and testing dataset, respectively. Sampling is done to get 100,000 sets of 2 to 10 images from the training data. For testing, sampled 10,000 sets of images containing m number of images per set where $m \in [2, 10]$. Figure 5.4 shows the performance of MEM against simple pooling operations with respect to the number of images in the set.

Prime Sum. Prime Sum is a classification problem over a set of MNIST images. A set is labeled positive if it contains any two digits such that their sum is a prime number. Dataset is constructed by randomly sampling five images from the MNIST dataset. Training data is constructed using 20,000 sets randomly sampled from the training data of MNIST. For testing, randomly sampled 5,000 sets from the testing data of MNIST are used. The results are reported in the second column of Table 5.1 that shows the robustness of the memory block.

Maximum of a Set. Maximum of a set is a regression problem to predict the highest digit present in a set of images from MNIST. A set of five images is constructed by randomly selecting samples from MNIST dataset. The label for each set is the largest number present in the set. For example, images of $\{2, 5, 3, 3, 6\}$ is labeled as 6. 20,000 training sets are used, and for testing, randomly sampled 5,000 sets are used. The detailed comparison of accuracy and MAE between different models is given in the second last column of Table 5.1. It is found that FF+Max learns the identity mapping and thus results in very high accuracy. In all the training sessions, experiments consistently obtained the training accuracy of 100% for the FF+Max configuration, whereas MEM generalizes better than the Deep Sets.

Counting Unique Images. Counting unique images is a regression problem over a set. This task involves counting unique objects in a set of images from fashion MNIST dataset [90]. We constructed the training data by selecting a set, as follows:

1. Let n be the number of total images and u be the number of unique images.
2. Randomly select an integer n between 2 and 10.
3. Randomly select another integer u between 1 and n.
4. Select u number of unique objects from fashion-MNIST training data.
5. Then add n-u number of randomly selected objects from the previous step.

The task is to count unique objects u in a given set. The results are shown in the third column of Table 5.1.

Amortized Gaussian Clustering. Amortized Gaussian clustering is a regression problem that involves estimating the parameters of a population of Mixture of Gaussian (MoG). Similar to Set Transformer [57], we test our model's ability to learn parameters of a Gaussian Mixture with k components such that the likelihood of the observed samples is maximum. This is in contrast to the EM algorithm, which updates parameters of the mixture recursively until the stopping criterion is satisfied. Instead, we use MEM to directly predict parameters of a MoG, i.e., $f(x;\theta) = \{\pi(x), (\mu(x), \sigma(x))_{j=1}^{k}\}$. For simplicity, we sample from MoG with only four components. The Generative process for each training dataset is as follows:

1. Mean of each Gaussian is selected from a uniform distribution i.e. $\mu_{j=1}^{k} \sim \text{Unif}(0, 8)$.
2. Select a cluster for each instance in the set, i.e.,

$$\pi \sim \text{Dir}([1, 1]^T); z_i \sim \text{Categorical}(\pi)$$

3. Generate data from an univariate Gaussian $\sim \mathcal{N}(\mu_{z_i}, 0.3)$.

We created a dataset of 20,000 sets, each consisting of 500 points sampled from different MoGs. Results in Table 5.1 show that MEM is significantly better than Deep Sets.

5.5.2 Real World Datasets

To show the robustness and scalability of the model for real-world problems, we have validated MEM on two larger datasets. Firstly, we tested our model on a point cloud dataset for predicting the object type from the set of 3D coordinates. Secondly, we used the largest public repository of histopathology images (TCGA) [86] to differentiate between two main subtypes of lung cancer. Without any significant effort in extracting histologically relevant features and fine-tuning, we achieved a remarkable accuracy of 84.84% on 5-fold validation.

Point Cloud Classification. We evaluated MEM on a more complex classification task using ModelNet40 [89] point cloud dataset. The dataset consists of 40 different objects or classes embedded in a three-dimensional space as points. We produce point-clouds with 100 points (x, y, z-coordinates) each from the mesh representation of objects using the point-cloud library's sampling routine [71][1]. We compare the performance against various other models reported in Table 5.2. We experimented with different configurations of our model and found that FF+MB1 works best for 100 points cloud classification. We achieve the classification accuracy of 85.21% using 100 points. Our model performs better than Deep Sets and Set Transformer for the same number of instances, showing the effectiveness of having attention from memories.

1.5 Configuration	Instance Size	Accuracy
3DShapeNet [89]	30^3	0.77
Deep set [94]	100	0.8200
VoxNet [64]	32^2	0.8310
3D GAN [88]	64^3	0.833
Set Transformer [57]	100	0.8454
Set Transformer [57]	1000	0.8915
Deep set [94]	5000	0.9
MVCNN [74]	$164 \times 164 \times 12$	0.901
Set Transformer [57]	5000	0.9040
VRN Ensemble [10]	32^3	0.9554
FF + MEM + MB1 (Ours)	**100**	**0.8521**

Table 5.2: Test accuracy for the point cloud classification on different instance sizes using various methods. MEM with configuration FF + MEM + MB1 achieves 85.21% accuracy for the instance size of 100 which is best compared to others.

[1]We obtained the training and test datasets from Zaheer et al. [94]

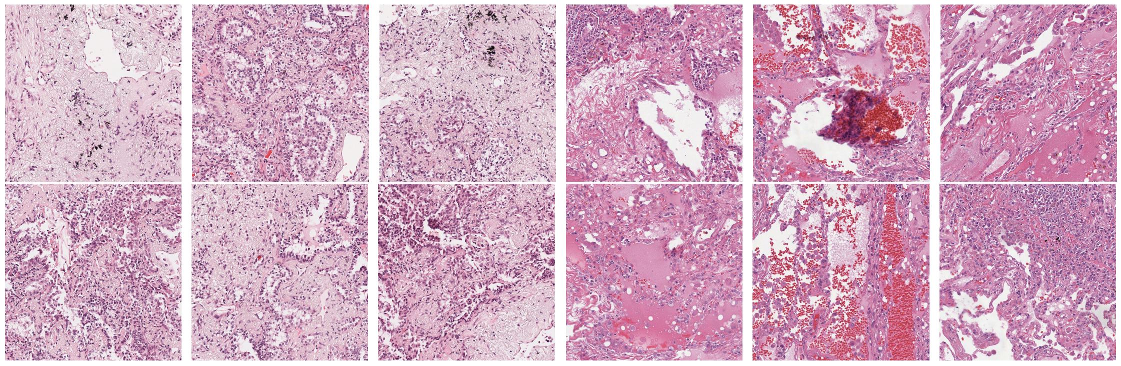


Figure 5.5: The patches extracted from two WSIs of patients with (a) LUAD and (b) LUSC. Each slide roughly contains 500 patches.

Lung Cancer Subtype Classification. Lung Adenocarcinoma (LUAD) and Lung Squamous Cell Carcinoma (LUSC) are two main types of non-small cell lung cancer (NSCLC) that account for 65-70% of all lung cancers [23]. Classifying patients accurately is important for prognosis and therapy decisions. Automated classification of these two main subtypes of NSCLC is a crucial step to build computerized decision support and triage systems. We present a two-staged method to differentiate LUAD and LUSC for whole slide images, short WSIs, that are very large images. Firstly, we implement a method to systematically sample patches/tiles from WSIs. Next, we extract image features from these patches using DenseNet [33]. We then use MEM to learn the representation of a set of patches for each WSI.

To the best of our knowledge, this is the first-ever study conducted on all the lung cancer slides in TCGA dataset (comprising of 2 TB of data consisting of 2.5 million patches of size 1000×1000 pixels). All research works in literature use a subset of the WSIs with their own test-train split instead of cross-validation, making it difficult to compare against them. However, we have achieved greater than or similar to all existing research works without utilizing any expert's opinions (pathologists) or domain-specific techniques. We used 2,580 WSIs from TCGA public repository [86] with 1,249, and 1,331 slides for LUAD and LUSC, respectively. We process each WSI as follows.

1. **Tissue Extraction.** Every WSI contains a bright background that generally contains irrelevant (non-tissue) pixel information. We removed non-tissue regions using color thresholds.

2. **Selecting Representative Patches.** Segmented tissue is then divided into patches. All the patches are then grouped into a pre-set number of categories (classes) via a clustering method. A 10% of all clustered patches are uniformly randomly selected distributed within each class to assemble *representative patches*. Six of these representative patches for each class (LUAD and LUSC) is shown in Figure 5.5.

3. **Feature Set.** A set of features for each WSI is created by converting its representative patches into image features. We use DenseNet [33] as the feature extraction model. There are a different number of feature vectors for each WSI.

1.5 Methods	Accuracy
Coudray et al. [13]	**0.85**
Jabber et al. [38]	0.8333
Khosravi et al. [46]	0.83
Yu et al. [92]	0.75
FF + MEM + Sum (ours)	**0.8484 ± 0.0210**
FF + MEM + Mean (ours)	0.8465 ± 0.0225
FF + MEM + MB1 (ours)	0.8457 ± 0.0219
FF + MEM + Dotprod (ours)	0.6345 ± 0.0739
FF + sum (DS)	0.5159 ± 0.0120
FF + mean (DS)	0.7777 ± 0.0273
FF + dotprod (DS)	0.4112 ± 0.0121
1.5	

Table 5.3: Accuracy for LUAD vs LUSC classification for various methods. For our experiments, we conducted comprehensive 5-fold cross validation accuracy whereas other methods have used non-standardized test set.

The results are shown in Table 5.3. We achieved the maximum accuracy of 84.84% with FF + MEM + Sum configuration. It is difficult to compare our approach against other approaches in literature due to non-standardization of the dataset. Coudray et al. [13] used the TCGA dataset with around 1,634 slides to classify LUAD and LUSC. They achieved AUC of 0.947 using patches at 20×. We achieved a similar AUC of 0.94 for one of the folds and average AUC of **0.91**. In fact, without any training, they achieved the similar accuracy as our model (around 85%). It is important to note that we did not do any fine-tuning or utilize any form of input from an expert/pathologist. Instead, we extracted diverse patches and let the model learn to differentiate between two subtypes by “attending” relevant ones. Another study by Jaber et al. [38] uses cell density maps, achieving an accuracy of 83.33% and AUC of 0.9068. However, they used much smaller portion of the TCGA, i.e., 338 TCGA diagnostic WSIs (164 LUAD and 174 LUSC) were used to train, and 150 (71 LUAD and 79 LUSC).

5.6 Summary

In this chapter, we introduced a Memory-based Exchangeable Model (MEM) for learning permutation invariant representations. The proposed method uses attention mechanisms over "memories" (higher order features) for modelling complicated interactions among elements of a set. Typically for MIL, instances are treated as independently and identically distributed. However, instances are rarely independent in real tasks, and we overcome this limitation using an "attention" mechanism in memory units, that exploits relations among instances. We also prove that the MEM is permutation invariant. We achieved good performance on all problems that require exploiting instance relationships. Our model scales well on real-world problems as well, achieving an accuracy score of 84.84% on classifying lung cancer subtypes on the largest public repository of histopathology images.

Chapter 6

Set Representation Learning using Graph Neural Networks

In this chapter, the thesis explore the application of graph neural networks for Set Representation Learning. A new framework is proposed that models a WSI as a fully connected graph to extract its representation. The proposed method processes the entire WSI at the highest magnification level; it requires a single label of the WSI without any patch-level annotations. Furthermore, modeling WSIs as fully connected graphs enhance the interpretability of the final representation. The proposed method treats each instance as a node of the graph to learn relations among nodes in an end-to-end fashion. Thus, the proposed method not only learns the representation for a given WSI but also models relationship among different patches. Method is evaluated for classifying of two subtypes of lung cancer, Lung Adenocarcinoma (LUAD) and Lung Squamous Cell Carcinoma (LUSC). LUAD and LUSC are the most prevalent subtypes of lung cancer, and their distinction requires visual inspection by an experienced pathologist. WSIs are obtained from the largest publicly available dataset, The Cancer Genome Atlas (TCGA) [86], to train the model for lung cancer subtype classification. The proposed method achieved an accuracy of 89% and 0.93 AUC.

The contributions of the paper are 3-folds, i) a graph-based method for representation learning of WSIs, and ii) a novel adjacency learning layer for learning connections within nodes in an end-to-end manner and iii) visualizing patches that are given higher importance by the network for the prediction.

The chapter is structured as follows: section 6.1 explains the approach, and experiments & results are reported in Section section 6.2.

6.1 Proposed Method

The proposed method for representing a WSI has two stages, i) sampling important patches and modeling them into a fully-connected graph, and ii) converting the fully-connected graph into a vector representation for classification or regression purposes. These two stages can be learned end-to-end in a single training loop. The major novelty of the method is the learning of the adjacency matrix that defines the connections within nodes. The overall proposed method is shown in Figure 6.1 and Figure 6.2. The method can be summarized as follows.

1. The important patches are sampled from a WSI using a color-based method described in [40]. A pre-trained CNN is used to extract features from all the sampled patches.
2. The given WSI is then modeled as a fully-connected graph. Each node is connected to every other node based on the adjacency matrix. The adjacency matrix is learned end-to-end using Adjacency Learning Layer.
3. The graph is then passed through a Graph Convolution Network followed by a graph pooling layer to produce the final vector representation for the given WSI.

The main advantage of the method is that it processes entire WSIs. The final vector representation of a WSI can be used for various tasks—classification (prediction cancer type), search (KNN search), or regression (tumor grading, survival prediction) and others.

Patch Selection and Feature Extraction. We used the method for patch selection proposed in [40]. Every WSI contains a bright background that generally contains irrelevant (non-tissue) pixel information. Non-tissue regions are removed using color thresholds. Segmented tissue is then divided into patches. All patches are grouped into a pre-set number of categories (classes) via a clustering method. A portion of all clustered patches (e.g., 10%) are randomly selected within each class. Each patch obtained after patch selection is fed into a pre-trained DenseNet [33] for feature extraction. These features are further feed to trainable fully connected layers and obtain final feature vectors each of dimension 1024 representing patches.

Graph Representation of WSI. A novel method is proposed for learning WSI representation using GCNNs. Each WSI is converted to a fully-connected graph, which has the following two components.

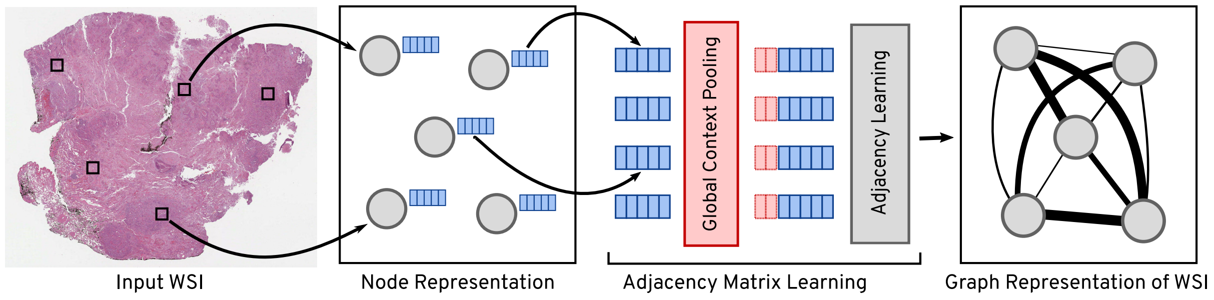


Figure 6.1: **Transforming a WSI to a fully-connected graph.** A WSI is represented as a graph with its nodes corresponding to distinct patches from the WSI. A node feature (a blue block beside each node) is extracted by feeding the associated patch through a deep network. A single context vector, summarizing the entire graph is computed by pooling all the node features. The context vector is concatenated with each node feature, subsequently fed into adjacent learning block. The adjacent learning block uses a series of dense layers and cross-correlation to calculate the adjacency matrix. The computed adjacency matrix is used to produce the final fully-connected graph. In the figure, the thickness of the edge connecting two nodes corresponds to the value in the adjacency matrix.

1. Nodes V: Each patch feature vector represents a node in the graph. The feature for each node is the same as the feature extracted for the corresponding patch.
2. Adjacency Matrix $\mathbf{A}$: Patch features are used to learn the $\mathbf{A}$ via adjacency learning layer.

Adjacency Learning Layer. Connections between nodes V are expressed in the form of the adjacency matrix $\mathbf{A}$. The proposed algorithm learns the adjacency matrix in an end-to-end fashion in contrast to the method proposed in [82] that thresholds the ℓ_2 distance on pre-computed features.The proposed method also uses global information about the patches while calculating the adjacency matrix. The idea behind using the global context is that the connection between two same nodes/patches can differ for different WSIs; therefore, elements in the adjacency matrix should depend not only on the relation between two patches but also on the global context of all the patches.

1. Let W be a WSI and $w_1, w_2, \ldots w_n$ be its patches. Each patch w_i is passed through a feature extraction layer to obtain corresponding feature representation x_i.

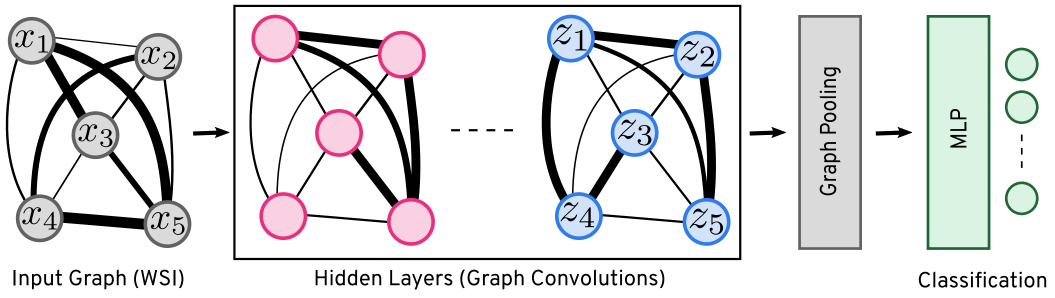


Figure 6.2: **Classification of a graph representing a WSI.** A fully connected graph representing a WSI is fed through a graph convolution layer to transform it into another fully-connected graph. After a series of transformations, the nodes of the final fully-connected graph are aggregated to a single condensed vector, which is fed to an MLP for classification purposes.

2. We use the theorem by Zaheer et al. [94] to obtain the global context from the features x_i. Feature vectors from all patches in the given WSI are pooled using a pooling function ϕ to get the global context vector c. Mathematically,

$$c = \phi(x_1, x_2, \ldots, x_n). \tag{6.1}$$

   Zaheer et al. showed that such a function can be used as an universal set approximator.

3. The global context vector c is then concatenated to each feature vector x_i to obtain concatenated feature vector x_i' which is passed through MLP layers to obtain new feature vector $x_i^* \cdot x_i^*$ are the new features that contain information about the patch as well as the global context.

4. Features x_i^* are stacked together to form a feature matrix $\mathbf{X}^*$ and passed through a cross-correlation layer to obtain adjacency matrix denoted by $\underset{n \times n}{\mathbf{A}}$ where each element a_{ij} in $\mathbf{A}$ shows the degree of correlation between the patches w_i and w_j. We use a_{ij} to represent the edge weights between different nodes in the fully connected graph representation of a given WSI.

Graph Convolution Layers. Once the graph representation of the WSI is implemented, two types of GCNN are experimented: ChebNets and GraphSAGE Convolution, which are spectral and spatial methods, respectively. Each hidden layer in GCNN models the interaction between nodes and transforms the feature into another feature space. Finally, a graph pooling layer transforms node features into a single vector representation. Thus, a WSI can now be represented by a condensed vector, which can be further used to do other tasks such as classification, image retrieval, etc.

General MIL Framework. The proposed method can be used in any MIL framework. The general algorithm for solving MIL problems is as follows:

1. Consider each instance as a node and its corresponding feature as the node features.
2. The global context of the bag of instances is learned to calculate the adjacency matrix $\mathbf{A}$.
3. A fully connected graph is constructed with each instance as a node and a_{ij} in A representing the edge weight between V_i and V_j.
4. Graph convolution network is used to learn the representation of the graph, which is passed through a graph pooling layer to get a single feature vector representing the bag of instances.
5. The single feature vector from the graph can be used for classification or other learning tasks.

6.2 Experiments

We evaluated the performance of our model on two datasets i) a popular benchmark dataset for MIL called MUSK1 [17], and ii) 1026 lung slides from TCGA dataset [26]. Our proposed method achieved a state-of-the-art accuracy of 92.6% on the MUSK1 dataset. We further used our model to discriminate between two subtypes of lung cancer—Lung Adenocarcinoma (LUAD) and Lung Squamous Cell Carcinoma (LUSC).

6.2.1 MUSK1 Dataset

It has 47 positive bags and 45 negative bags. Instances within a bag are different conformations of a molecule. The task is to predict whether new molecules will be musks or non-musks. We performed 10 fold cross-validation five times with different random seeds. We compared our approach with various other works in literature, as reported in Table 6.1. The miGraph [99] is based on kernel learning on graphs converted from the bag of instances. The latter two algorithms, MI-Net [85], and Attention-MIL [35], are based on DNN and use either pooling or attention mechanism to derive the bag embedding.

Algorithm	**Accuracy**
mi-Graph [99]	0.889
MI-Net [85]	0.887
MI-Net with DS [85]	0.894
Attention-MIL [35]	0.892
Attention-MIL with gating [35]	0.900
Ming Tu et al. [82]	0.917
Proposed Method	**0.926**

Table 6.1: Evaluation on MUSK1.

6.2.2 LUAD vs LUSC Classification.

Lung Adenocarcinoma (LUAD) and Lung Squamous Cell Carcinoma (LUSC) are two main subtypes of non-small cell lung cancer (NSCLC) that account for 65-70% of all lung cancers [23]. Automated classification of these two main subtypes of NSCLC is a crucial step to build computerized decision support and triage systems.

We obtained 1,026 hematoxylin and eosin (H&E) stained permanent diagnostic WSIs from TCGA repository [26] encompassing LUAD and LUSC. We selected relevant patches from each WSI using a color-based patch selection algorithm described in [40]. Furthermore, we extracted image features from these patches using DenseNet [33]. Now, each bag in this scenario is a set of features labeled as either LUAD or LUSC. We trained our model to classify bags as two cancer subtypes. The highest 5-fold classification AUC score achieved was 0.92, and the average AUC across all folds was 0.89. We performed cross-validation across different patients, i.e., training was performed using WSIs from a totally different set of patients than the testing. The results are reported in Table 6.2. We achieved state-of-the-art accuracy using the transfer learning scheme. In other words, we extracted patch features from an existing pre-trained network, and the feature extractor was not re-trained or fine-tuned during the training process. The Figure 6.5 shows the receiver operating curve (ROC) for one of the folds.

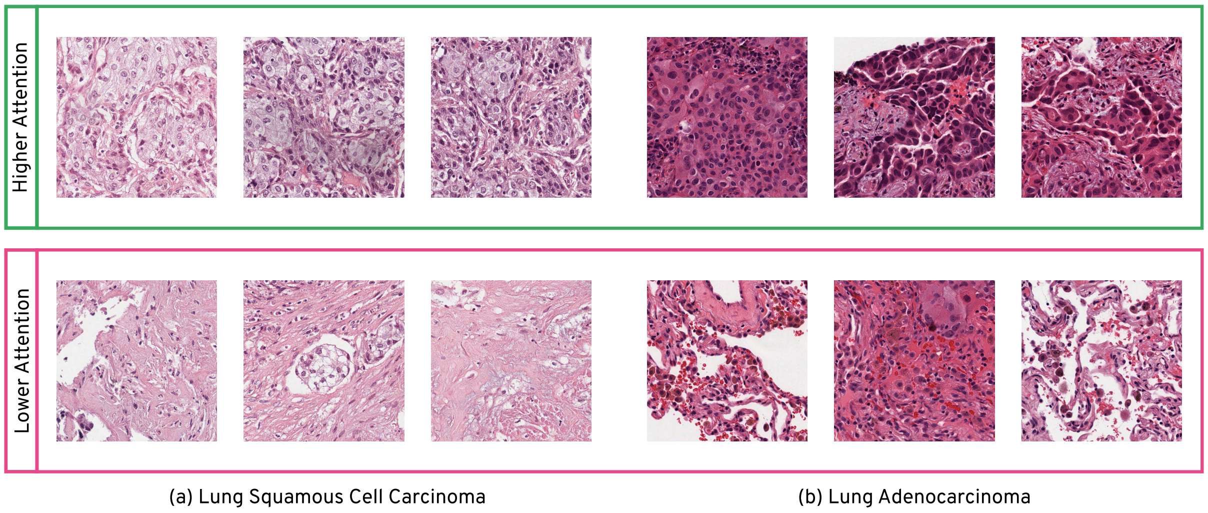


Figure 6.3: Six patches from two WSIs diagnosed with LUSC and LUAD, respectively. The six patches are selected, such that the first three (top row) are highly "attended" by the network, whereas the last three (bottom row) least attended. The first patch in the upper row is the most attended patch (more important) and the first patch in the lower row in the least attended patch (less important).

6.2.3 Inference

One of the primary obstacles for real-world application of deep learning models in computer-aided diagnosis is the black-box nature of the deep neural networks. Since the proposed architecture uses Global Attention Pooling [59], it can visualize the importance that the network gives to each patch for making the final prediction. Such visualization can provide more insight to pathologists regarding the model's internal decision making. The global attention pooling layer learns to map patches to "attention" values. The higher attention

Algorithm	**AUC**
Coudray et al. [13]	0.85
Khosravi et al. [46]	0.83
Yu et al. [92]	0.75
Proposed method	**0.89**

Table 6.2: Performance of various methods for LUAD/LUSC predictions using transfer learning. Our results report the average of 5-fold accuracy values.

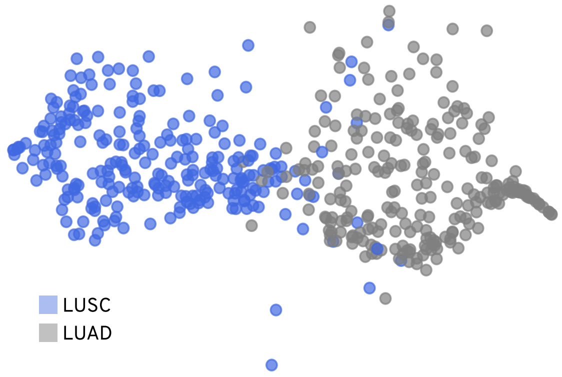


Figure 6.4: t-SNE visualization of feature vectors extracted after the Graph Pooling layer from different WSIs. The two distinct clusters for LUAD and LUSC demonstrate the efficacy of the proposed model for disease characterization in WSIs. The overlap of two clusters contain WSIs that are morphologically and visually similar.

values signify that the model focuses more on those patches. We visualize the patches with high and low attention values in Figure 6.3. One of the practical applications of our approach would be for triaging. As new cases are queued for an expert's analysis, the CAD system could highlight the regions of interest and sort the cases based on the diagnostic urgency. We observe that patches with higher attention values generally contain more nuclei. As morphological features of nuclei are vitals for making diagnostic decisions [66], it is interesting to note this property is learned on its own by the network. Figure 6.4 shows the t-SNE plot of features vectors for some of the WSIs. It shows the clear distinction between the two cancer subtypes, further favoring the robustness of our method.

Implementation Details. We used PyTorch Geometric library to implement graph convolution networks [18]. We used pre-trained DenseNet [33] to extract features from histopathology patches. We further feed DenseNet features through three dense layers with dropout ($p = 0.5$).

Ablation Study. We tested our method with various different configurations for the TCGA dataset. We used two layers in Graph Convolution Network—ChebNet and SAGE Convolution. We found that ChebNet outperforms SAGE Convolution and also results in better generalization. Furthermore, we experimented with different numbers of filters in ChebNet, and also different pooling layers—global attention, mean, max, and sum pooling. We feed the pooled representation to two fully connected Dense layers to get the

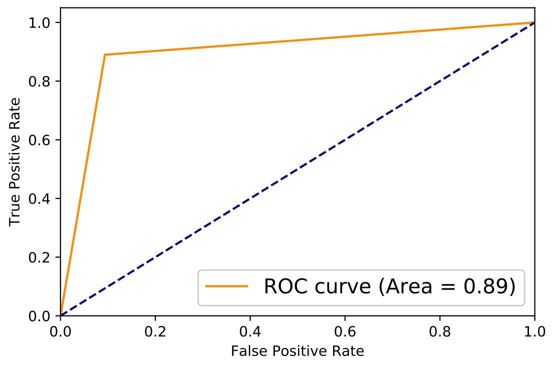


Figure 6.5: The ROC curve of prediction.

final classification between LUAD and LUSC. All the different permutations of various parameters result in 32 different configurations, the results for all these configurations are provided in Table 6.3. It should be noted that the results reported in the previous sections are based on Cheb-7 with mean pooling.

configuration	**mean**	attention	max	add
Cheb-7	**0.8889**	0.8853	0.7891	0.4929
Cheb 3_BN	0.8771	0.8635	0.8471	0.5018
Cheb 5	0.8762	0.8830	0.8750	0.5082
Cheb 3	0.8752	0.8735	0.8702	0.5090
Cheb 5_BN	0.8596	0.8542	0.7179	0.4707
Cheb 7_BN	0.7239	0.6306	0.5618	0.4930
SAGE CONV_BN	0.6866	0.5848	0.6281	0.5787
SAGE CONV	0.5784	0.6489	0.5389	0.5690

Table 6.3: Comparison of different network architecture and pooling method (attention, mean, max and sum pooling). **BN** stands for BatchNormalization [37], **Cheb** stands for Chebnet with corresponding filter size and **SAGE** stands for SAGE Convolution. The best performing configuration is Cheb-7 with mean pooling.

6.3 Summary

The accelerated adoption of digital pathology is coinciding with and probably partly attributed to recent progress in AI applications in the field of pathology. This disruption in the field of pathology offers a historic chance to find novel solutions for major challenges in diagnostic histopathology and adjacent fields, including biodiscovery. In this chapter, we proposed a technique for representing an entire WSI as a fully-connected graph. We used the graph convolution networks to extract the features for classifying the lung WSIs into LUAD or LUSC. The results show the good performance of the proposed approach. Furthermore, the proposed method is explainable and transparent as we can use attention values and the adjacency matrix to visualize relevant patches.

Chapter 7

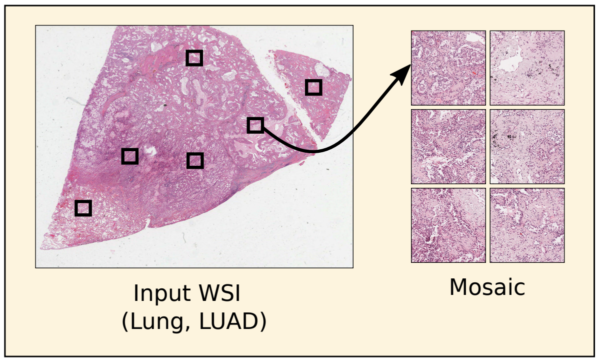
Pay Attention with Focus: Hierarchical Set Learning

Deep learning methods such as convolutional neural networks (CNNs) are difficult to directly utilize for analyzing whole slide images (WSIs) due to the large image dimensions. A new two-staged method is proposed to overcome this limitation. First, a set of representative patches (called mosaic) are extracted from a WSI. Each patch of a mosaic is encoded to a feature vector using a deep network. The feature extractor model is fine-tuned using hierarchical target labels of WSIs, i.e., anatomic site and primary diagnosis. In the second stage, a set of encoded patch-level features from a WSI is used to compute the primary diagnosis probability through the proposed *Pay Attention with Focus* scheme, an attention-weighted averaging of predicted probabilities for all patches of a mosaic modulated by a trainable focal factor. Experimental results show that the proposed model can be robust and effective for the classification of WSIs.

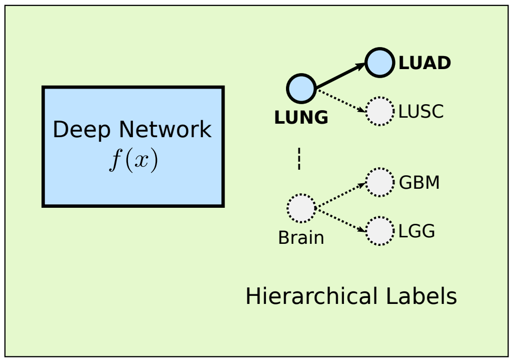
Contributions: The contribution is three-fold (i) proposed a novel attention-based MIL approach for the classification of WSIs, (ii) fine-tuned a feature extractor model using multiple and hierarchically arranged target labels of WSIs, and (iii) insights into the model's decision making by visualizing attention values. The method is tested on two large-scale datasets derived from The Cancer Genomic Atlas (TCGA) repository provided by NIH [81].

7.1 Proposed Method

There are two stages in the proposed method (i) bag preparation and (ii) multi-instance learning with FocAtt-MIL. In the first stage, representative patches (called mosaic) are



(a) Mosaic Preparation



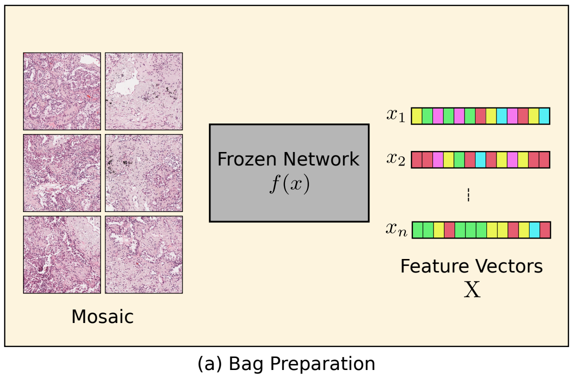
(b) Fine-tune using Heirarchical Labels

Figure 7.1: **Training a Feature Extractor.** A feature extractor is trained with hierarchical target labels of a WSI. (a) A set of representative WSI patches (called mosaic) is extracted [43]. (b) The patches are used to fine-tune a deep network; each patch is assigned the parent WSI's labels, i.e., anatomic site and primary diagnosis.

extracted from a WSI. The mosaic's patches are encoded to a set of feature vectors (called bag) using a deep network. The feature extraction model can be a pre-trained network or can be fine-tuned to increase its effectiveness, as shown in Figure 7.1. In the second stage, the proposed MIL technique (called FocAtt-MIL) is trained to predict the primary diagnosis for a given bag (a WSI). The schematic for the second stage is shown in Figure 7.2.

Bag Preparation. A patch selection method proposed by Kalra et al. [43] is used to extract the representative patches from a WSI. The method removes non-tissue regions using a color threshold. The remaining tissue-containing patches are grouped into a preset number of categories through a clustering algorithm. A portion of all clustered patches (e.g., 10%) are randomly selected within each cluster, yielding a *mosaic*. The mosaic is transformed into a bag $X = \{x_1, \ldots, x_n\}$, where x_i is the feature vector of i^{th} patch, obtained through a deep network (a feature extractor). The Figure 7.2 shows the bag preparation stage, the frozen network $f(x)$ represents a non-trainable deep network used as a feature extractor.

Fine-tune a Feature Extractor using Hierarchical Labels. In MIL, robust features enable weak learners to make better predictions, thus improving the final aggregated prediction. A WSI is generally associated with the following two labels—anatomic site and primary diagnosis. These two labels are arranged in a hierarchy as shown in Figure 7.1. Consider, y_{as} and y_{pd} represent anatomic site and primary diagnosis respectively. Then, instead of predicting these labels independently, the model predict $P(y_{as})$, and $P(y_{pd}|y_{as})$.



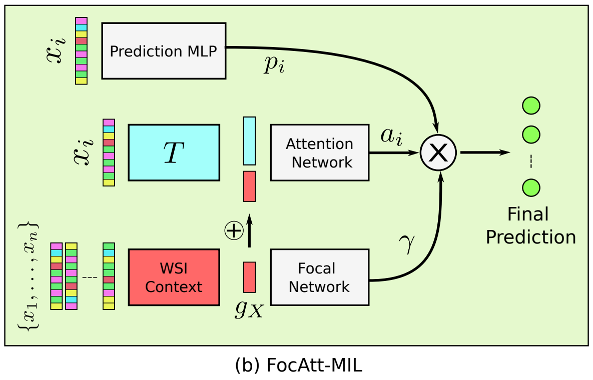


Figure 7.2: **Classification of WSIs with FocAtt-MIL.** The two-stage method for the classification of WSI. (a) The mosaic of a WSI is converted to a bag X containing a set of feature vectors $\{x_1, \ldots, x_n\}$. (b) The feature vectors in a bag X are transformed to the primary diagnosis probability through FocAtt-MIL. The prediction probability p_i is computed for an individual feature vector x_i. A WSI context g_X is computed for the entire bag X using (7.1). The WSI context g_X is used to compute the attention value a_i and the focal factor γ. The final prediction is computed using (7.2).

The conditional probability $P(y_{pd}|y_{as})$ helps in modelling the dependent relationship. Using Bayes theorem, we get, $P(y_{as}|y_{pd}) = P(y_{pd}|y_{as})P(y_{as})/P(y_{pd})$, where $P(y_{as}|y_{pd}) = 1$, because of the dependence. It can be simplified further, $P(y_{pd}) = P(y_{pd}|y_{as})P(y_{as})$, and compute cross entropy losses for the predictions of both y_{as} and y_{pd}. The model equally weight both the losses towards the final loss of the network.

WSI Context Learning. A single vector representation of a WSI (or a bag X) is computed as,

$$g_X = \phi(\theta(x_1), \ldots, \theta(x_n)), \tag{7.1}$$

where, θ is a neural network and ϕ is a pooling function, such as sum, mean, and max. It has been proven in [94] that (7.1) can approximate any set function. The vector g_X is used by the attention module and the focal network.

The FocAtt-MIL Approach. The FocAtt-MIL is a permutation-invariant model that learns to predict a target label (primary diagnosis) y_{pd} from a bag X (a WSI). The approach is composed of four major components (Figure 7.2):

1. *Prediction MLP.* A prediction p_i is computed for each item x_i in the bag X, using a trainable deep network called Prediction MLP.

2. *WSI Context.* It's a deep network that computes a single vector representing an entire bag X using (7.1).

3. *Attention Module.* The attention module is composed of two networks, a transformation network T, and the Attention Network. The attention module takes the i^{th} patch $x_i \in X$, and the WSI context g_X to compute an attention value $a_i \in [0, 1]$ for that patch.

4. *Focal Network.* Another deep network that uses WSI context g_X to compute a focal factor γ (a vector) that modulates the final prediction. The length of γ is same as the number of discrete values in the target label, thus allowing the per dimension modulation.

The Final Prediction. The final output from the FocAtt-MIL is computed by aggregating individual attention-weighted predictions modulated by the learned focal factor, as follows

$$y(j) = \sum_{i=1}^{n} \mathbf{p_i}(j)^{\gamma(j)} a_i. \tag{7.2}$$

The $\mathbf{p_i}$, and $\boldsymbol{\gamma}$ in (7.2) are both vectors. The y is converted to a probability distribution by dividing with $sum(y)$.

7.2 Results

The proposed approach is evaluated for two different WSI classification tasks. All experiments are conducted with 4 Nvidia V100 GPUs (32 GB vRAM each). The code has been written using the Tensorflow library [1].

Algorithm	**Accuracy**
Coudray et al. [13]	0.85
Kalra & Adnan et al. [42]	0.85
Khosravi et al. [46]	0.83
Yu et al. [92]	0.75
FocAtt-MIL (proposed method)	**0.88**

Table 7.1: Performance comparison for LUAD/LUSC classification via transfer learning.

LUAD vs LUSC Classification – Lung Adenocarcinoma (LUAD) and Lung Squamous Cell Carcinoma (LUSC) are two main subtypes of non-small cell lung cancer (NSCLC) that account for 65-70% of all lung cancers [95]. Automated classification of these two main subtypes of NSCLC is a crucial step to assist pathologists [23, 95]. For this task, we establish the efficacy of FocAtt-MIL to differentiate between LUAD and LUSC. We obtained 2,580 hematoxylin and eosin (H&E) stained WSIs of lung cancer from TCGA repository [81]. The data is split into 1,806 training, and 774 testing WSIs [42]. The dataset is approximately 2 TB. We obtained mosaic for each WSI using the approach in [43], and subsequently converted the mosaic to a bag X of features using a pre-trained DenseNet [33]. We did not fine-tune the feature extraction model for this task to have a fair comparison against other transfer-learning based approaches in the literature. We trained the FocAtt-MIL to classify bags between the two subtypes of lung cancer. We achieved an accuracy of 88% on test WSIs (AUC of 0.92). The accuracy has been reported in Table 7.1.

We conducted an **ablation study** to understand the effect of different model parameters. Removing the WSI context g_X from the attention module resulted in a 4% reduction of the accuracy. Excluding the focal factor γ and the global context g_X from the final prediction resulted in 6% reduction in the accuracy. The ablation suggests that the model's performance is the most optimal by (i) incorporating the WSI context g_X in the attention computation, and (ii) allowing the focal factor to modulate the final aggregated prediction. We used the attention module of the trained model to **visualize the attention heat-map** on the unseen WSIs (Figure 7.3). The visual inspection of these two WSIs reveals that the model decided based on regions containing malignant tissue and ignored non-cancerous regions. In the LUSC WSI (right), regions with squamous formations are deemed the most important ones. For the LUAD WSI (left), the salient regions solely come from the malignant area, implying that the model differentiates between normal lung alveolar tissue and LUAD. Therefore, one could say that attention heatmaps are histopathologically meaningful. For LUAD samples, regions, where cancerous tissue meets non-cancerous structures are deemed the most important. Such contrast makes cancerous glandular structures easier to recognize. However, this phenomenon cannot be seen in LUSC samples, as the model is responsive to regions that are completely composed of malignant squamous carcinoma.

Pan-cancer Analysis – In the second experiment series, we evaluated the approach against a large-scale pan-cancer classification of WSIs. The **dataset** used for this task has been proposed by Riasatian et al. [70]. It comprises more than 7 TB data, consisting of 7,097 training and 744 test WSIs, distributed across 24 different anatomic sites and 30 different primary diagnoses. All WSIs in the dataset are taken from a public repository of

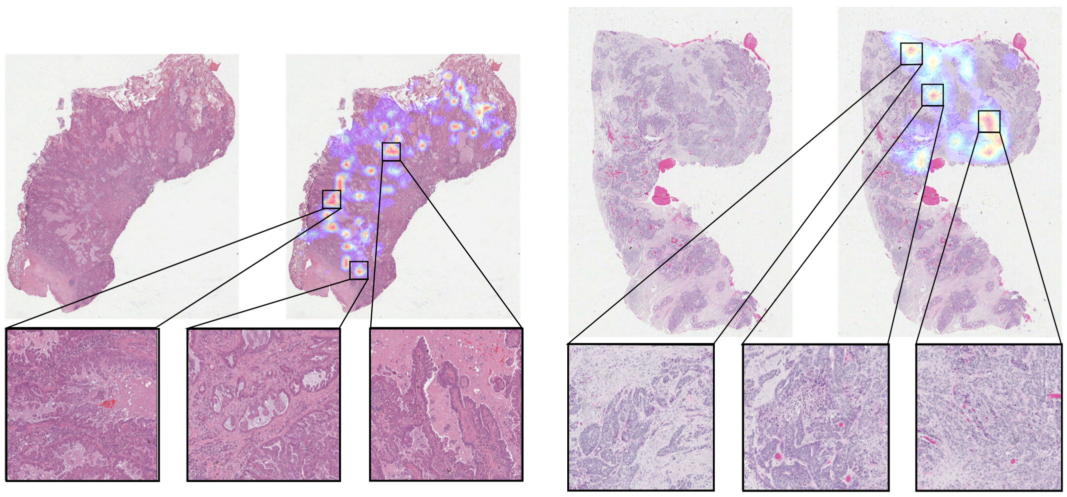


Figure 7.3: **Attention Visualization.** The attention values are augmented on the two exemplars WSIs. **Left Image (LUAD):** Regions of high importance come from the cancerous regions while sparing normal lung tissue, fibrosis, and mucin deposition. Additionally, by inspecting important regions at a higher magnification, it is noticeable that the malignant glandular formations border with non-malignant areas. **Right Image (LUSC):** Regions that are considered to be important for classification are composed of malignant squamous cells. However, unlike LUAD, the attention model seems to be responsive to regions with solid malignant structures.

WSIs, TCGA [81]. We obtained a mosaic for each WSI and then applied a cellularity filter [70] to further reduce the number of patches in each mosaic. Subsequently, we obtained 242,202 patches for training WSIs and 116,088 patches for testing WSIs. Each patch is of the size 1000×1000, but we resized them to 256×256 pixels.

We used three different **feature extractors** to validate the FocAtt-MIL. We prepared a separate "bag" for each feature extractor. These three feature extractors are: DenseNet (DN) [33], KimiaNet [70], and the fine-tuned DenseNet (FDN). We fine-tuned the DenseNet on training patches using weak labels obtained from their respective WSIs. The weakly labeled fine-tuning has shown to be effective [70]. In our case, the weak labels are anatomic site and primary diagnosis, arranged in a hierarchy. This hierarchical arrangement of labels is incorporated during the training using the approach outlined earlier in section 7.1.

We **trained the FocAtt-MIL** model with the same architecture for all the three different bags. We tested three different configurations of FocAtt-MIL, i.e., FocAtt-MIL-DN, FocAtt-MIL-KimiaNet, and FocAtt-MIL-FDN. For all three configurations, we used the

SGD optimizer with a learning rate of 0.01, weight decay of 10^{-6}, and momentum of 0.9. We applied *gradient clipping* of 0.01 and dropout between layers to prevent the exploding gradients. We trained models for 45 epochs. Figure 7.4 shows the validation loss and accuracy while training three different configurations. It is evident that FocAtt-MIL-FDN is outperforming from the very early epochs. Interestingly, both FocAtt-MIL-FDN and FocAtt-MIL-KimiaNet (feature extractors specialized for histopathology) seem to have converged to an optimal validation accuracy around 20-25 epochs. For the fine-tuning, we used Adam optimizer [47] and a learning rate of 10^{-5} were used for 20 epochs.

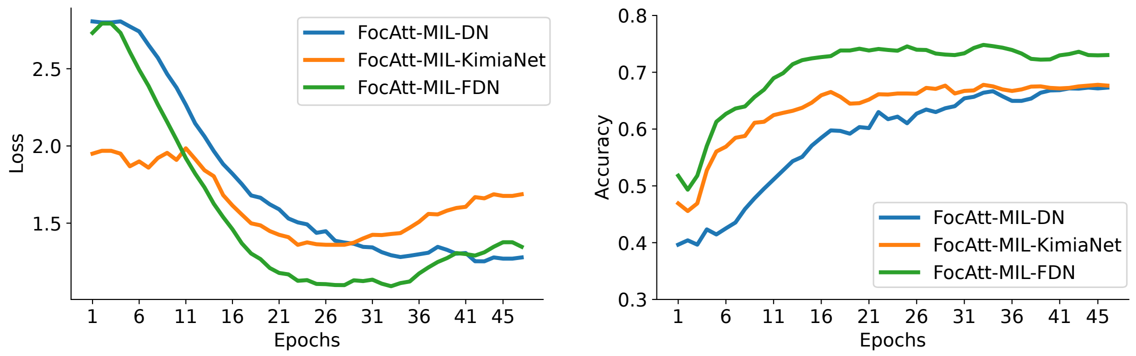


Figure 7.4: **FocAtt-MIL Training.** The loss and accuracy on validation dataset during the training of three different configurations of FocAtt-MIL, i.e FocAtt-MIL-DN, FocAtt-MIL-KimiaNet, and FocAtt-MIL-FDN.

The 30 unique primary diagnoses in the dataset can be further grouped into 13 tumor types. The type of tumor is generally known at the inference time, and the objective is to predict the cancer subtype. To **validate the efficacy of our model**, we computed the cancer subtype classification (i.e., primary diagnosis) accuracy for the given tumor type. This type of classification is called *vertical classification*. The vertical classification results are reported in Table 7.2. The results[1] show that FocAtt-MIL can elevate the accuracy of pre-trained features; DenseNet features have shown to underperform compared to KimiaNet features [70, 44]. However, within the proposed FocAtt-MIL scheme, DenseNet features become quite competitive. This applies to the fine-tuned DenseNet (FocAtt-MIL-FDN) as well, whose results are on par with the highly customized KimiaNet features when used within the FocAtt-MIL framework.

[1]For abbreviations GBM, LGG, ACC,..., see *wiki.cancerimagingarchive.net*

7.3 Summary

The accelerated adoption of digital pathology offers a historic opportunity to find novel solutions for major challenges in diagnostic histopathology. In this study, we proposed a novel attention-based MIL technique for the classification of WSIs. We introduced a focal factor, computed using a global representation of WSI for modulating the individual patch-level prediction, thus promoting a more accurate aggregated final prediction. We also proposed a novel fine-tuning approach to extract more robust features from WSI patches. We fine-tune a feature extraction model using patches and weak hierarchical labels from their respective WSIs. We validated the proposed framework on two large datasets derived from TCGA repository [81]. The results suggest competitive performance on both datasets. Furthermore, the proposed method is explainable and transparent as we utilized the attention values to visualize important regions.

Tumor Type	Primary Diagnosis	FocatAtt-MIL-DN	FocAtt-MIL-KimiaNet	FocAtt-MIL-FDN
Brain	GBM	**0.9714**	0.9429	0.8571
	LGG	0.6410	0.7692	**0.8205**
Endocrine	ACC	0.6667	0.6667	0.6667
	PCPG	**1.0000**	**1.0000**	**1.0000**
	THCA	0.9608	**1.0000**	**1.0000**
Gastrointestinal tract	COAD	**0.6875**	0.4375	0.5000
	ESCA	0.5000	**0.8571**	0.5714
	READ	0.0833	0.5000	**0.6667**
	STAD	**0.8333**	0.7333	**0.8333**
Gynaecological	CESC	0.8824	**0.9412**	0.7647
	OV	0.5000	0.8000	**1.0000**
	UCS	0.6667	**1.0000**	0.3333
Liver, pancreaticobiliary	CHOL	0.2500	0.0000	**0.5000**
	LIHC	0.8857	**0.9143**	0.8571
	PAAD	**1.0000**	0.7500	0.8333
Melanocytic malignancies	SKCM	**0.9167**	0.8750	**0.9167**
	UVM	**1.0000**	0.2500	**1.0000**
Prostate/testis	PRAD	**1.0000**	0.9500	**1.0000**
	TGCT	**1.0000**	**1.0000**	**1.0000**
Pulmonary	LUAD	0.5789	0.8158	**0.8947**
	LUSC	**0.9302**	0.6977	0.7442
	MESO	0.6000	**1.0000**	**1.0000**
Urinary tract	BLCA	0.9118	**1.0000**	0.8529
	KICH	0.5455	0.6364	**0.7273**
	KIRC	0.9200	0.9000	**0.9600**
	KIRP	0.5714	0.6786	**0.7143**

Table 7.2: Pan-cancer vertical classification accuracy of FocAtt-MIL for features from regular DenseNet (FocAtt-MIL-DN), KimiaNet (FocAtt-MIL-KimiaNet), and DenseNet fine-tuned with hierarchical labels (FocAtt-MIL-FDN).

Chapter 8

Summary and Conclusions

Gigapixel Images are involved in many interesting real-world applications such as Digital Pathology, Astronomy, Satellite Imagery, etc. and pose unique challenges for Artificial Intelligence. However, there has not been much work done on learning such gigantic images efficiently. Most of the advancements made in the computer vision domain are limited to smaller image datasets such as CIFAR100, CIFAR10 [52], MNIST [54], etc. This work studied the problem of learning gigapixel Images as a Set Representation Learning problem and subsequently, proposed three novel algorithms for learning gigapixel images. Such massive images can easily be converted into a set of patches, and thus learning gigapixel images can be viewed as a set representation learning problem. However, learning representation for sets is not trivial. Any set representation learning algorithm must satisfy the following properties:

1. Permutation Invariance: This property implies that the order of elements in the set doesn't matter, i.e., permuting elements should not change the output.

2. Learning Inter-Dependencies: This refers to learning relations between different elements of the sets. It may be possible only a few elements are relevant for a specific task. For example, in digital pathology, non-cancerous patches don't provide any information.

3. Universal Approximation: This implies that the algorithm should be capable of learning any set function or any mapping $\mathbb{R}^{N\text{x}d} \mapsto \mathbb{R}^m$, where N is the number of elements in the set.

Three new set representation learning algorithms are presented which satisfy above mentioned properties. A new Permutation Invariant Model called 'MEM' is introduced in chapter 5. Memory networks are used to learn the attention or inter-dependency among different elements of the set and showed mathematically that MEM satisfies all three properties listed above. The second significant contribution of this work is the application of Graph Neural Networks for set representation learning. In chapter 6, it is demonstrated that any set could be converted to an equivalent graph structure where each element of the set represents a node in the graph and corresponding edges can be learned in an end-to-end fashion. Lastly, this thesis proposes a hierarchical learning scheme for sets in chapter 7. Hierarchical labels provide more information than single labels; thus, helps in learning better representations in a progressive (hierarchical) manner.

8.1 Future Work

In subsequent work, it would be interesting to expand the idea of set representation learning to set generation. Generating a set or exchangeable sequences can find applications in Few-Shot Learning, anomaly detection, transfer learning, and multi-task situations. Another exciting line of research would be to explore the use of Transformer networks [16] to learn the relationship between different elements of the set. The ultimate goal would be to make highly optimized neural networks such that it's scalable to gigapixel images directly.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] Mohammed Adnan, Shivam Kalra, and Hamid R Tizhoosh. Representation Learning of Histopathology Images Using Graph Neural Networks. page 8, 2020.

[3] David J Aldous. Representations for partially exchangeable arrays of random variables. *Journal of Multivariate Analysis*, 11(4):581–598, 1981.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[5] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49. JMLR Workshop and Conference Proceedings, 2012.

[6] Jocelyn Barker, Assaf Hoogi, Adrien Depeursinge, and Daniel L Rubin. Automated classification of brain tumor type in whole-slide digital pathology images using local representative tiles. *Medical image analysis*, 30:60–71, 2016.

[7] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[8] José M Bernardo. The concept of exchangeability and its applications.

[9] Benjamin Bloem-Reddy and Yee Whye Teh. Probabilistic symmetry and invariant neural networks. *arXiv preprint arXiv:1901.06082*, 2019.

[10] Andrew Brock, Theodore Lim, J. M. Ritchie, and Nick Weston. Generative and Discriminative Voxel Modeling with Convolutional Neural Networks.

[11] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs, 2014.

[12] Marc-André Carbonneau, Veronika Cheplygina, Eric Granger, and Ghyslain Gagnon. Multiple instance learning: A survey of problem characteristics and applications. *Pattern Recognition*, 77:329–353, 2018.

[13] Nicolas Coudray, Paolo Santiago Ocampo, Theodore Sakellaropoulos, Navneet Narula, Matija Snuderl, David Fenyö, Andre L Moreira, Narges Razavian, and Aristotelis Tsirigos. Classification and mutation prediction from non–small cell lung cancer histopathology images using deep learning. *Nature medicine*, 24(10):1559, 2018.

[14] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

[15] Neofytos Dimitriou, Ognjen Arandjelović, and Peter D Caie. Deep learning for whole slide image analysis: An overview. *Frontiers in Medicine*, 6:264, 2019.

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[17] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[18] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[19] Fei Gao, Teresa Wu, Jing Li, Bin Zheng, Lingxiang Ruan, Desheng Shang, and Bhavika Patel. Sd-cnn: A shallow-deep cnn for improved breast cancer diagnosis. *Computerized Medical Imaging and Graphics*, 70:53–62, 2018.

[20] Robert Gens and Pedro M Domingos. Deep symmetry networks. In *Advances in neural information processing systems*, pages 2537–2545, 2014.

[21] Zoubin Ghahramani and Katherine A Heller. Bayesian sets. In *Advances in neural information processing systems*, pages 435–442, 2006.

[22] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[23] Simon Graham, Muhammad Shaban, Talha Qaiser, Navid Alemi Koohbanani, Syed Ali Khurram, and Nasir Rajpoot. Classification of lung cancer histology images using patch-level summary statistics. In *Medical Imaging 2018: Digital Pathology*, volume 10581, page 1058119. International Society for Optics and Photonics, 2018.

[24] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[25] Metin N. Gurcan, Laura Boucheron, Ali Can, Anant Madabhushi, Nasir Rajpoot, and Bulent Yener. Histopathological Image Analysis: A Review. *IEEE reviews in biomedical engineering*, 2:147–171, 2009.

[26] David A Gutman, Jake Cobb, Dhananjaya Somanna, Yuna Park, Fusheng Wang, Tahsin Kurc, Joel H Saltz, Daniel J Brat, Lee AD Cooper, and Jun Kong. Cancer digital slide archive: an informatics resource to support integrated in silico analysis of tcga pathology data. *Journal of the American Medical Informatics Association*, 20(6):1091–1098, 2013.

[27] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

[28] William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications, 2018.

[29] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[31] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In *International conference on artificial neural networks*, pages 44–51. Springer, 2011.

[32] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[33] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[34] Maximilian Ilse, Jakub M. Tomczak, and Max Welling. Chapter 22 - Deep multiple instance learning for digital histopathology. In S. Kevin Zhou, Daniel Rueckert, and Gabor Fichtinger, editors, *Handbook of Medical Image Computing and Computer Assisted Intervention*, pages 521–546. Academic Press.

[35] Maximilian Ilse, Jakub M Tomczak, and Max Welling. Attention-based deep multiple instance learning. *arXiv preprint arXiv:1802.04712*, pages 2127–2136, 2018.

[36] Maximilian Ilse, Jakub M Tomczak, and Max Welling. Deep multiple instance learning for digital histopathology. In *Handbook of Medical Image Computing and Computer Assisted Intervention*, pages 521–546. Elsevier, 2020.

[37] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[38] Mustafa I Jaber, Liudmila Beziaeva, Christopher W Szeto, John Elshimali, Shahrooz Rabizadeh, and Bing Song. Automated adeno/squamous-cell nsclc classification from diagnostic slide images: A deep-learning framework utilizing cell-density maps, 2019.

[39] Olav Kallenberg. *Probabilistic symmetries and invariance principles*. Springer Science & Business Media, 2006.

[40] S Kalra, C Choi, S Shah, L Pantanowitz, and HR Tizhoosh. Yottixel–an image search engine for large archives of histopathology whole slide images. *arXiv preprint arXiv:1911.08748*, 65:101757, 2019.

[41] Shivam Kalra, Mohammed Adnan, Sobhan Hemati, Taher Dehkharghanian, Shahryar Rahnamayan, and Hamid Tizhoosh. Pay attention with focus: A novel learning scheme for classification of whole slide images. *arXiv preprint arXiv:2106.06623*, 2021.

[42] Shivam Kalra, Mohammed Adnan, Graham Taylor, and Hamid R Tizhoosh. Learning permutation invariant representations using memory networks. In *European Conference on Computer Vision*, pages 677–693. Springer, 2020.

[43] Shivam Kalra, H. R. Tizhoosh, Charles Choi, Sultaan Shah, Phedias Diamandis, Clinton J. V. Campbell, and Liron Pantanowitz. Yottixel – An Image Search Engine for Large Archives of Histopathology Whole Slide Images. 65:101757.

[44] Shivam Kalra, H. R. Tizhoosh, Sultaan Shah, Charles Choi, Savvas Damaskinos, Amir Safarpoor, Sobhan Shafiei, Morteza Babaie, Phedias Diamandis, Clinton J. V. Campbell, and Liron Pantanowitz. Pan-cancer diagnostic consensus through searching archival histopathology images using artificial intelligence. *NPJ digital medicine*, 3(1):1–15, 2020.

[45] Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 53(8):5455–5516, 2020.

[46] Pegah Khosravi, Ehsan Kazemi, Marcin Imielinski, Olivier Elemento, and Iman Hajirasouliha. Deep convolutional neural networks enable discrimination of heterogeneous digital pathology images. *EBioMedicine*, 27:317–328, 2018.

[47] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[48] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[49] Daisuke Komura and Shumpei Ishikawa. Machine Learning Methods for Histopathological Image Analysis. 16:34–42.

[50] Daisuke Komura and Shumpei Ishikawa. Machine learning methods for histopathological image analysis. *Computational and Structural Biotechnology Journal*, 2018.

[51] Iryna Korshunova, Jonas Degrave, Ferenc Huszár, Yarin Gal, Arthur Gretton, and Joni Dambre. Bruno: A deep recurrent model for exchangeable data. In *Advances in Neural Information Processing Systems*, pages 7190–7198, 2018.

[52] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[53] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[54] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

[55] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer, 1999.

[56] Yann LeCun, D Touresky, G Hinton, and T Sejnowski. A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school*, volume 1, pages 21–28, 1988.

[57] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer. *CoRR*, abs/1810.00825, 2018.

[58] Juho Lee, Yoonho Lee, and Yee Whye Teh. Deep amortized clustering. *arXiv preprint arXiv:1909.13433*, 2019.

[59] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[60] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[61] Anant Madabhushi, Shannon Agner, Ajay Basavanhally, Scott Doyle, and George Lee. Computer-aided prognosis: Predicting patient and disease outcome via quantitative fusion of multi-scale, multi-modal data. 35(7):506–514, 2011.

[62] Anant Madabhushi and George Lee. Image analysis and machine learning in digital pathology: Challenges and opportunities. *Medical Image Analysis*, 33:170–175, oct 2016.

[63] Tahir Mahmood, Muhammad Arsalan, Muhammad Owais, Min Beom Lee, and Kang Ryoung Park. Artificial intelligence-based mitosis detection in breast cancer histopathology images using faster r-cnn and deep cnns. *Journal of clinical medicine*, 9(3):749, 2020.

[64] Daniel Maturana and Sebastian Scherer. VoxNet: A 3D Convolutional Neural Network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928.

[65] Andreas Maurer, Massimiliano Pontil, and Gabor Lugosi. Structured sparsity and generalization. *Journal of Machine Learning Research*, 13(3), 2012.

[66] Shivang Naik, Scott Doyle, Shannon Agner, Anant Madabhushi, Michael Feldman, and John Tomaszewski. Automated gland and nuclei segmentation for grading of prostate and breast cancer histopathology. In *2008 5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 284–287. IEEE, 2008.

[67] Peter Orbanz and Daniel M Roy. Bayesian models of graphs, arrays and other exchangeable random structures. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):437–461, 2014.

[68] Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Poczos. Deep learning with sets and point clouds. *arXiv preprint arXiv:1611.04500*, 2016.

[69] Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Poczos. Equivariance through parameter-sharing. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2892–2901. JMLR. org, 2017.

[70] Abtin Riasatian, Morteza Babaie, Danial Maleki, Shivam Kalra, Mojtaba Valipour, Sobhan Hemati, Manit Zaveri, Amir Safarpoor, Sobhan Shafiei, Mehdi Afshari, et al. Fine-tuning and training of densenet for histopathology image representation using tcga diagnostic slides. *arXiv preprint arXiv:2101.07903*, 2021.

[71] R.B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4, May 2011.

[72] John Shawe-Taylor. Symmetries and discriminability in feedforward network architectures. *IEEE Transactions on Neural Networks*, 4(5):816–826, 1993.

[73] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[74] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proc. ICCV*, 2015.

[75] P. J. Sudharshan, Caroline Petitjean, Fabio Spanhol, Luiz Eduardo Oliveira, Laurent Heutte, and Paul Honeine. Multiple instance learning for histopathological breast cancer image classification. *Expert Systems with Applications*, 117:103–111, 2019.

[76] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.

[77] Vaishali S Tidake and Shirish S Sane. Multi-label classification: a survey. *International Journal of Engineering and Technology*, 7(1045), 2018.

[78] Hamid Reza Tizhoosh and Liron Pantanowitz. Artificial intelligence and digital pathology: challenges and opportunities. *Journal of pathology informatics*, 9, 2018.

[79] Jakub M. Tomczak, Maximilian Ilse, and Max Welling. Deep Learning with Permutation-invariant Operator for Multi-instance Histopathology Classification. *arXiv preprint arXiv:1712.00310*, 2017.

[80] Jakub M. Tomczak, Maximilian Ilse, Max Welling, Marnix Jansen, Helen G. Coleman, Marit Lucas, Kikki de Laat, Martijn de Bruin, Henk A. Marquering, Myrtle J. van der Wel, Onno J. de Boer, C. Dilara Savci Heijink, and Sybren L. Meijer. Histopathological classification of precursor lesions of esophageal adenocarcinoma: A Deep Multiple Instance Learning Approach.

[81] Katarzyna Tomczak, Patrycja Czerwińska, and Maciej Wiznerowicz. The cancer genome atlas (tcga): an immeasurable source of knowledge. *Contemporary oncology*, 19(1A):A68, 2015.

[82] Ming Tu, Jing Huang, Xiaodong He, and Bowen Zhou. Multiple instance learning with graph neural networks. *arXiv preprint arXiv:1906.04881*, 2019.

[83] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[84] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets, 2016.

[85] Xinggang Wang, Yongluan Yan, Peng Tang, Xiang Bai, and Wenyu Liu. Revisiting multiple instance neural networks. *Pattern Recognition*, 74:15–24, 2018.

[86] John N Weinstein, Eric A Collisson, Gordon B Mills, Kenna R Mills Shaw, Brad A Ozenberger, Kyle Ellrott, Ilya Shmulevich, Chris Sander, Joshua M Stuart, Cancer Genome Atlas Research Network, et al. The cancer genome atlas pan-cancer analysis project. *Nature genetics*, 45(10):1113, 2013.

[87] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

[88] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T. Freeman, and Joshua B. Tenenbaum. Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling.

[89] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[90] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[91] Yang Yang, Meng Wang, and Bao Liu. Exploring and comparing of the gene expression and methylation differences between lung adenocarcinoma and squamous cell carcinoma. *Journal of cellular physiology*, 234(4):4454–4459, 2019.

[92] Kun-Hsing Yu, Ce Zhang, Gerald J Berry, Russ B Altman, Christopher Ré, Daniel L Rubin, and Michael Snyder. Predicting non-small cell lung cancer prognosis by fully automated microscopic pathology image features. *Nature communications*, 7:12474, 2016.

[93] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep sets, 2018.

[94] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017.

[95] Cecilia Zappa and Shaker A Mousa. Non-small cell lung cancer: current treatment and future advances. *Translational lung cancer research*, 5(3):288, 2016.

[96] Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837, 2013.

[97] Qi Zhang, Sally A Goldman, Wei Yu, and Jason E Fritts. Content-based image retrieval using multiple-instance learning. Citeseer.

[98] Yan Zhang, Jonathon Hare, and Adam Prügel-Bennett. Deep set prediction networks. *arXiv preprint arXiv:1906.06565*, 2019.

[99] Zhi-Hua Zhou, Yu-Yin Sun, and Yu-Feng Li. Multi-instance learning by treating instances as non-iid samples. In *Proceedings of the 26th annual international conference on machine learning*, pages 1249–1256. ACM, 2009.