Democratic and Popular Algerian Republic

Ministry of the Superior Teaching and the Scientific Research

Mohamed Khider University – Biskra

Faculty of Exact Sciences, Natural Sciences and Life

Department of Computer Science

Dissertation

Presented to obtain the academic master's degree in Computer Science

Option:IVA

By

KHELIL HATHEM

TITLE

Digital pathology image analysis with deep learning

Defended on Sept 2020, in front of the jury composed of:

Dr.		President	University M.K. Biskra
Dr.	**Dr.Zerari Abd El Moumene**	Advisor	University M.K. Biskra
Dr.		Examiner	University M.K. Biskra

2020

Acknowledgements

Firstly, thank you, God, for guiding me through this path, giving me such wonderful family, friends, and mentors to complete this dissertation.

This dissertation could not have been possible without the blessings and support of many people in my life. I sincerely appreciate and gratefully acknowledge their help and backing. I would like to express an appreciation particularity to the following.

I would like to express my sincere gratitude to my supervisor Dr.Abd El Mouméne Zerari for the continuous support of my dissertation, for his patience, motivation, and his continuous positive energy from the very first day. His guidance and observations always helped me throughout the research.My cordial thanks extend to my dissetation committee members who accepted to evaluate this modest work.

My deepest gratitude goes to my family members who always encouraged me to do my best. Their assistance, along with their continuous motivation kept me going. I am forever thankful to my parents for being such understanding and supportive figures in my life. I can never thank them enough for their endless support and encouragement especially in difficult times.

Hathem KHELIL

Contents

List of Figures

List of Tables

Abbreviations

ACC: Accuracy

ANN: Artificial Neural Network

AI: Artificial Intelligence

CAD:Computer-Aided Diagnosis

CLL: Chronic Lymphocytic Leukemia

CNN: Convolutional Neural Network

DP: Digital Pathology

DL: Deep Learning

F1: F1-score

FL: Follicular Lymphoma

H&E: Hematoxylin and Eosin

HL: Hodgkin Lymphomas

IDC: Invasive Ductal Carcinoma

MCL: Mantle Cell Lymphoma

MLP: Multilayer Perceptron's

NIA: National Intelligence Authority

NHL: Non-Hodgkin Lymphomas

NHL-CNN: Deep learning CNN for classifying the NHL subtypes

PR: PRecision

R-Net: Recurrent U-Net

ReLU: Rectified Linear Units

RE: REcall

SGD: Stochastics Gradient Descent

WHO: World Health Organization

WSI: Whole Slide Image

Introduction

Digital Pathology (DP) includes the process of digitizing histopathology slides using whole-slide scanners as well as the analysis of these digitized whole-slide images (WSI) using computational approaches. Such scanners were introduced two decades ago (around 1999), but the roots of digital pathology can be traced back to the 1960s, when Prewitt and Mendelsohn [15] devised a method to scan simple images from a microscopic field of a common blood smear. However, over the past two decades, technological advances have enabled efficient digitization of whole-slide images, subsequently helping to streamline pathology workflows across pathology labs worldwide [1].

The advantages of these developments are many: remote diagnostics, immediate availability of archival cases, easier consultations with expert pathologists, and perhaps the most significant benefit, the possibility for computerized or computer-aided diagnostics (CAD) [16]. These systems have the potential to aid pathologists in their clinical decisions associating image information with diseases features [17].

The application of CAD for histopathology can be classified into three main groups [3]: the first is the detection, segmentation and quantification of important tissue structures, the second is the classification of histopathology imagery based on grade or lesion type and the last is the disease diagnosis and prognosis.

In this master's project, we are interested in the design of a CAD system for the classification of Non-Hodgkin Lymphomas (NHL) subtypes on whole slide images.

Lymphoma is a clonal malignancy of lymphocytes, either T cells or B cells. However, 85% of cases are derived from B lymphocytes [18]. Lymphoid malignancies are diagnosed in 280,000 people annually worldwide and include at least 38 entities according to the World Health Organization (WHO) Classification of Lymphoid Malignancies [19]. There are

at least 38 subtypes of Lymphoma cancer [20] which are generally classified as Hodgkin Lymphomas (HL) and Non-Hodgkin Lymphomas (NHL).

NHL is one of the common malignant tumors with the fastest growth rate [21]. The Chronic Lymphocytic Leukemia (CLL), Follicular Lymphoma (FL) and Mantle Cell Lymphoma (MCL) are the major types of NHL [22].

To differentiate these diseases, a pathologist analyses tissue samples stained with Hematoxylin-Eosin (H&E) for recognizing important regions for diagnoses, such as nuclei and glandular structures, and their features. However, the distinction between them represents a major challenge for pathologists [23]; owing to the wide range of clinical presentations and histopathological features. Even the expert pathologist, sometimes, finds difficulties to differentiate these lymphomas subtypes of H&E. These challenges have made it necessary to adapt modern machine learning methods, such as Deep Learning (DL) [24] to solve clinically relevant tasks in diagnostic pathology.

Deep learning has various closely related definitions or high-level descriptions, one of which is :" a class of machine learning techniques, where many layers of information processing stages in hierarchical supervised architectures are exploited for unsupervised feature learning and for pattern analysis/classification" [25]. The essence of deep learning is to compute hierarchical features or representations of the observational data, where the higher-level features or factors are defined from lower-level ones [11].

Convolutional neural networks (CNNs) are a type of discriminative deep architecture consisting of a convolutional layer and pooling layer which subsamples the output of the convolutional layer and reduces the data rate of the layer below. These layers are often stacked layers [24].

The emergence of deep learning, in particular, CNN, has also energized the medical imaging field [9], [26], [27] and enabled development of diagnostic tools displaying remarkable accuracy [14]- [28]. This motivates the use of CNNs for classifying NHL subtypes on whole slide images.
In this setting, the aim of this work is to provide an overview of the DP tasks, and how DL methods have the potential for being the unifying approach for the many tasks in DP, by presenting a specific application for Non-Hodgkin Lymphomas diagnostic, which corresponds to 70% of Lymphomas cases and represents a major challenge for

pathologists due to its intrinsic complexity [29]. In addition, the distinction between its subtypes (CLL, FL and MCL) are essential for disease monitoring, identification of its stage and orientation towards appropriate treatments for the patient.

In this context, the dissertation is structured as follows:

The first chapter (1) offers a brief overview of DP domain and the variety of image analysis tasks in this domain, including detection and counting, segmentation, and tissue classification.

The second chapter (2) provides an understanding of the principles of machine learning, artificial neural networks and recent advances in convolutional neural networks, focusing on the applications of convolutional neural networks in the DP field.

The chapter (3) presents definitions and details about Lymphoma and its different subtypes considered in this study. A new deep neural network framework for classifying NHL subtypes on whole slide images is also described in this chapter.

The chapter (4) describes the detailed design and tool used in this project, and how to implement the deep learning system for classifying NHL subtypes. Empirical evaluation and results are also presented in this chapter.

In conclusion, we summarize and review our ideas and results by giving possible improvements and some perspectives.

Chapter 1

WHOLE SLIDE IMAGING IN DIGITAL PATHOLOGY

1.1 Introduction

In clinical medicine, histopathology refers to the microscopic examination of stained tissue biopsies in order to study disease. Histopathology is the gold standard for tissue diagnosis and thereby confirms clinical suspicions enabling treatment to proceed and provide data to inform treatment of neoplasic diseases and the patient's prognosis.

Digital pathology (DP) is the conversion of the light microscope image of a slide into a set of digitized files that allow the reproduction of the original slide on a computer workstation; it is also called virtual microscopy or whole slide imaging. Digitization also allows for the manipulation of the image and or its data to derive information of diagnostic, prognostic or therapeutic benefit, which would otherwise not normally be readily available. The field of digital pathology has exploded and is currently regarded as one of the most promising avenues of diagnostic medicine in order to achieve even better, faster and cheaper diagnosis, prognosis and prediction of cancer and other important diseases.

This chapter introduces the digital analysis of scanned tissue slides and outlines the current state of the technical and workflow aspects of digitizing glass slides as well as their characteristics. In addition, it provides examples of areas in which these tools are

currently utilized to generate data in preclinical and clinical workflows that go beyond the conventional histopathological data provided by manual pathology review.

1.2 Digital pathology

Digital Pathology (DP) is the process by which histology slides are digitized to produce high-resolution images. It is becoming increasingly common due to the growing availability of whole slide digital scanners [1]. It incorporates the acquisition, management, sharing and interpretation of pathology information, including slides and data, in a digital environment.

Digital pathology is the technique of analyzing high-resolution digitally scanned histology images, which may then take advantage of computational tools and algorithms [16]. In the first step, the whole histology glass slide is scanned with the help of a high-resolution image scanner. This image information is then shared with a distant pathologist using a high-speed Internet connection. Whole-slide imaging systems scan the slides at 20x, 40x, and even 100x with high precision. Whole-slide scanned images are remotely consulted for a second opinion, which saves time, cost, and physical transportation of slides and prevents slide damage.

The role of computer-based imaging is very important and helps pathologists in making a decision. Therefore, digital pathology is an intersection of pathology and computers that is capable of replacing the conventional microscope-based diagnosis in the near future [30].

1.3 The roots of digital pathology

The origin of analyzing images with objective tools is almost as old as microscopy itself [31]. It started with measuring and counting when Leeuwenhoek developed a system to measure microscopic objects in the 17th century.

Although there were many incremental advances throughout the centuries, image analysis remained largely unchanged until the advent of digital imaging and computerized

analysis in the second half of the last century. From the 1970s on, new tools were developed that allowed for easier measurement of cellular or tissue components.

In 1980, video cameras became widely available, leading to further improvement of available systems [2].

Around the year 2000, digital slide scanners became commercially available, and whole-slide imaging(WSI) started to become increasingly common [1]. The rapid progress of WSI technology, along with advances in software applications and high-speed networking, have made it possible to fully integrate digital pathology into pathology workflows.

Digital pathology enables pathologists to engage, evaluate, and collaborate rapidly and remotely, with transparency and consistency, thus improving efficiency and productivity. The future of digital pathology could eventually encompass enhanced translational research, computer aided diagnosis (CAD) and personalized medicine.

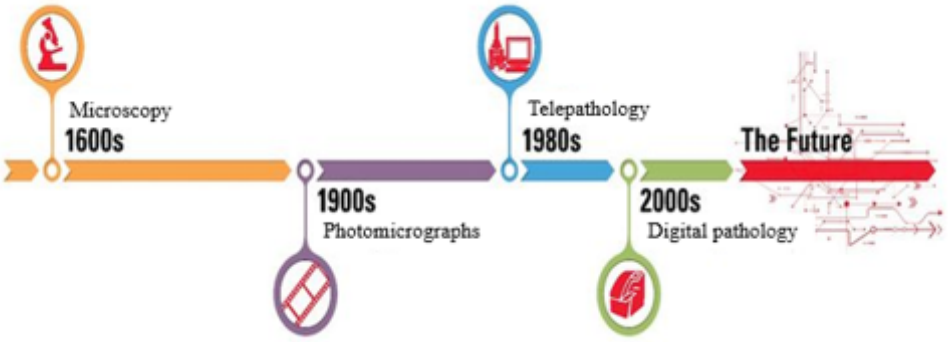


FIGURE 1.1: Evolution of Digital pathology over time [1]

1.4 Whole-slide imaging

In the last decade, digital imaging in pathology has been significantly impacted by the development and application of whole slide imaging (WSI) technology. The automated WSI scanner is a robotic microscope capable of digitizing an entire glass slide, using software to merge or stitch individually captured images into a composite digital image. The critical components of an automated WSI device (system) include the hardware (scanner composed of an optical microscope and digital camera connected to a computer), software (responsible for image creation and management, viewing of images, and image analysis where applicable), and network connectivity. Whole slide imaging

technology has evolved to the point where digital slide scanners are currently capable of automatically producing high-resolution digital images within a relatively short time [32].

The virtual image may represent an entire glass slide or a user-selected area of the glass slide, and is often referred to as a whole slide image or digitized slide. Upon retrieval of the digital file, the captured image of the slide can be viewed on a computer monitor without the use of an actual microscope. The software interface used to view digital slides simulates the operation of light microscopy (see Figure 2.13) . Several types of WSI scanners have been developed by vendors, all capable of producing automated, high-speed, high-resolution whole slide digital images [33].

FIGURE 1.2: Digital Pathology environment [2]

Whole slide digital images are giga-pixel images that are generally stored in a multiresolution pyramid structure [2]. These files contain multiple down sampled versions of the original image (see Figure 2.14). Each image in the pyramid is stored as a series of tiles, to facilitate rapid retrieval of subregions of the image.

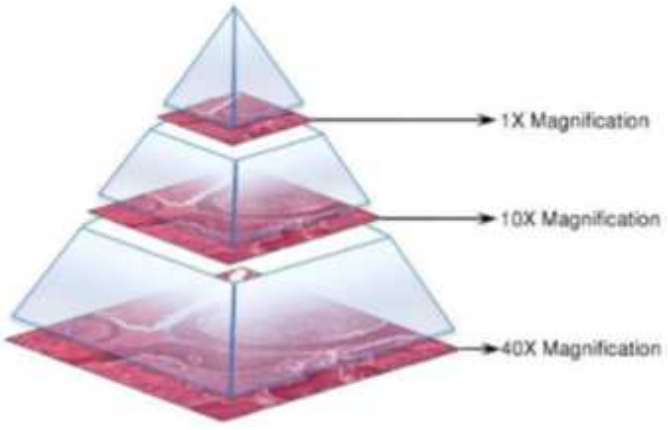


FIGURE 1.3: An illustration of how digital slides are stored in a pyramid structure [2]

Whole slide imaging technology has several advantages over conventional microscopy [16] such as:

1.4.1 Routine Histopathology

Pathology as a practice vitally relies on the accurate interpretation of microscopic images in order to correctly diagnose patients and guide therapeutic decision making. With the advent of whole slide imaging (WSI), pathologists have begun to transfer the act of viewing glass slides from the microscope to the computer monitor. The advantage of digital pathology is using digital scanners and computer software to process, analyze and share digital images of tissues electronically with doctors or entire teams of doctors around the world. Once the image of the tissue is captured, it can be used indefinite amount of times for diagnostics, sharing, and educational purposes. The goal of pathology digitalization is not to replace pathologists with computers, but to use computers for menial and time-consuming tasks. This way, pathologists will have more time to spend on making better diagnosis, and for a lab this means more time for improving planning and sample tracking.

1.4.2 Telepathology and Data Transmission

One of the earliest uses of digital pathology was the rapid transmission of pathologic data between pathologists and other pathologists, clinicians, or collaborators. Initially, this involved the transmission of static images between pathologists, but in recent years, digital pathology has been enhanced through the use of whole-slide imaging.

1.4.3 Education

Digital pathology is today widely used for educational purposes in telepathology and teleconsultation as well as in research projects. Digital pathology allows to share and annotate slides in a much easier way and to download annotated lecture sets generates new opportunities for e-learning and knowledge sharing in pathology.

1.4.4 Pathology Archiving

Manually filing glass slides requires physical space to store the slides with the risk that they can be misfiled, broken, or fade over time. A digital archive may provide a long-term storage solution, allowing for only tissue blocks to be physically stored. In addition, the digital archive can include associated metadata, allowing for retrieval of both the image file and metadata together and for the retrieval of images using metadata-based searches.

1.5 Histopathology

Histopathology refers to the examination of tissue in order to study the manifestations of disease [34]. The name Histopathology is derived from the Greek word for tissue = Histos, disease = Pathos and Logos = the study of.

In order to produce a translucent slice of tissue which can be observed under a light microscope, a series of preparation phases must be performed. Prior to tissue processing, a biopsy is obtained from the patient, which is transported to a pathology laboratory in sealed containers to preserve the tissue.

The procedure for preparation of tissue sections consists of the following steps [3]]:

- Collecting the tissue removed from the body for diagnosis.
- Applying a fixative to preserve the tissue.
- Embedding the tissue in paraffin.
- Cutting the tissue into thin (3-5 μm) sections and mounting it on glass slides,
- Treating the tissue with multiple contrasting stains to highlight different tissue structures and cellular features.

These steps are usually performed by a pathology lab technician who then sends the specimen to a pathologist for interpretation.

Staining is one of the most important steps in tissue preparation. This is because most cells are colorless and transparent. Therefore, histological sections have to be stained

to highlight important features of the tissue as well as to enhance the tissue contrast. Microscopical evaluation of such stained tissue sections provides invaluable information to the pathologists for diagnosing and characterizing various pathological conditions.

Hematoxylin and Eosin (H&E) staining is the most commonly used staining technique in histopathology. Hematoxylin stains cell nuclei blue, and the counter-stain eosin stains cytoplasmic and non-nuclear components in different shades of pink. H&E staining is non-specific, meaning that it stains most of the cells in much the same way. Specific staining techniques are those that selectively stain particular chemical groupings or molecules within cells or tissues. Immunohistochemistry (IHC), as an example, makes it possible to visualize the distribution and localization of specific cellular components within a cell or tissue. Figure 1.4 shows examples of H&E and IHC staining of a metastatic region in a lymph node tissue section. the tumor area can be well detected by the use of IHC [3]. Histopathological analysis of breast tissue specimens almost always starts with a careful examination of H&E stained specimens.

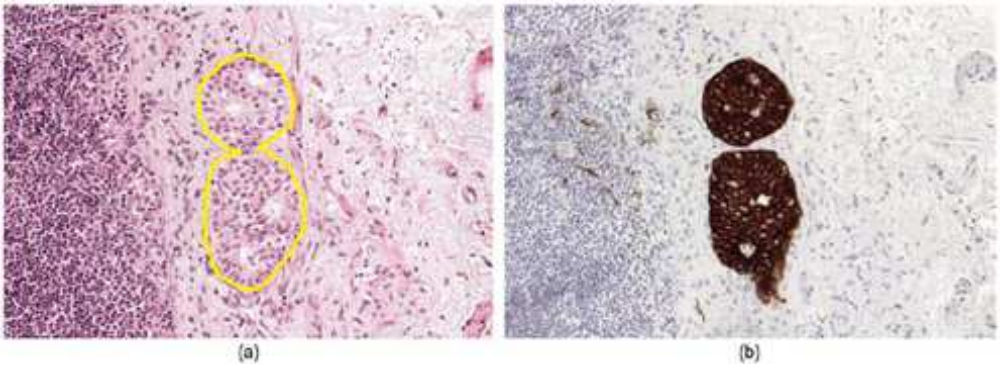


FIGURE 1.4: Metastatic region in a lymph node tissue section; (a) : Examples of H&E staining, (b): IHC staining [3]

1.6 Histopathological image analysis

DP is becoming increasingly common due to the growing availability of whole slide digital scanners.These digitized slides afford the possibility of applying image analysis techniques to DP for applications in detection, segmentation, and classification [4].

The figure 2.16 presents a list of the segmentation and detection tasks addressed. These tasks have been chosen as they represent the ensemble of critical components necessary for most of the pertinent pathology tasks (e.g., disease grading, mitotic counting) and thus span the current challenges in the DP image analysis space [1].

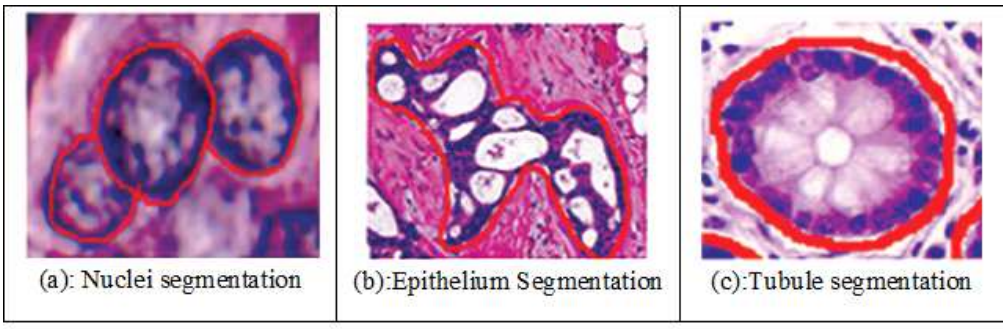


FIGURE 1.5: Segmentation and detection tasks addressed [4]

1.6.1 Segmentation and Detection Tasks

The term "segmentation" represents a process in which an image is spatially parsed into constituent parts that have some significance or utility. Segmentation can be applied to a histology image to delineate an accurate boundary for histologic primitives (i.e., nuclei, epithelium, tubules, and Invasive ductal carcinoma) so that precise morphological features can be extracted [27].

Detection tasks (i.e., lymphocyte and mitosis detection) are different from segmentation tasks in that the goal is typically to simply identify the center of the primitive of interest and not explicitly extract the primitive contour or boundary. Segmentation typically tends to be more challenging than detection, especially in the cases where the primitives of interest have multiple possible manifestations (e.g., mitotic cycles). Thus, a single monolithic classifier or model may not be able to capture the full range of diversity in presentation (i.e., the constellation of visual queues and features used to identify a particular histologic primitive) [35].

1.6.1.1 Nuclei segmentation

Nuclei segmentation is a very import problem in the field of digital pathology for several reasons (Figure 2.16.(a)) . First, nuclei morphology is a key component in most cancer grading shame, Second, and efficient nuclei segmentation techniques can significantly reduce the human effort for cell level analysis [Irshad, 2014]. However, segmentation of nuclei is a complicated task : tissue type, staining differences and cell type convey them different visual characteristics, which makes it very difficult to design traditional image segmentation algorithms that work satisfactorily for all of these different cases [27].

1.6.1.2 Epithelium Segmentation

Most of cases, the regions of cancer are manifested in the epithelium area, therefore epithelium and stoma regions are very important for identifying the cancer cancerous cells. It is very hard to predict the overall survival and outcome of breast cancer patients based on the histological pattern within Stroma region. Epithelium segmentation can help to identify the cancerous region where most of the cancer cells manifests (Figure 2.16.(b)) [36]

1.6.1.3 Tubule segmentation

The aggressiveness of cancer can be determine based on the morphology of tubule from the pathological images. The tubule region (Figure 2.16.(c)) become massively disorganized in the later stage of cancer [37].

1.6.1.4 Lymphocyte detection

Lymphocytes, a subtype of white blood cells, are an important part of the immune system. Lymphocytic infiltration is the process by which the density of lymphocytes greatly increases at sites of disease or foreign bodies, indicating an immune response.

A stronger immune response has been highly correlated to better outcomes in many forms of cancer, such as breast and ovarian. As a result, identifying and quantifying the density and location of lymphocytes has gained a lot of interest recently, particularly in the context of identifying which cancer patients to place on immunotherapy.

The main challenges and applications of lymphocyte detection (Figure 2.16.(d)): first, in general lymphocytes looks like a blue tint from the absorption of hematoxylin. Second, the appearance and other morphology is very similar in hue to nuclei. The applications of Lymphocyte detection are to identify cancer patient to place on immunotherapy and many more [4].

1.6.1.5 Mitosis detection

The cell growth rate can be determined with counting of mitotic events from the pathological images which is an important aspect to determine the aggressiveness of cancer diagnosis. Presently, the manually counting process is applied in pathological practice that is extremely difficult and time consuming. Therefore, automatic mitosis detection (Figure 2.16.(e))approach has efficient application in pathological practice [4].

1.6.1.6 Invasive ductal carcinoma detection

One of the very common type of breast cancer is Invasive ductal carcinoma (IDC) and most of the times pathologist focus on regions to identify of IDC cancer. A common

preprocessing step called automatic aggressiveness grading method is used to define the exact region of IDC from whole slides images (Figure 2.16.(f)) [28].

1.6.2 Tissue-Based Classification Task

Another set of use cases for digital pathology is tissue level classification, for example, Lymphoma subtype identification. Lymphoma is cancer that begins in infection-fighting cells of the immune system, called lymphocytes. Where, in some cases even the expert pathologist sometimes phases difficulties to differential Lymphoma subtypes. where its subtypes identification In the field of pathology.

As opposed to explicitly identifying individual tissue-based primitives (e.g., mitoses, nuclei) and trying to identify primitive specific features to make predictions regarding tissue class, an alternative strategy is to directly learn the set of features representative of the tissue class via machine learning methods. The machine learning classifier could thus be trained to self-discover the nuanced disease patterns within each class, however it is generally based on explicit primitive identification and a comprehension of the (potentially unknown) domain specific relationships of the primitives.

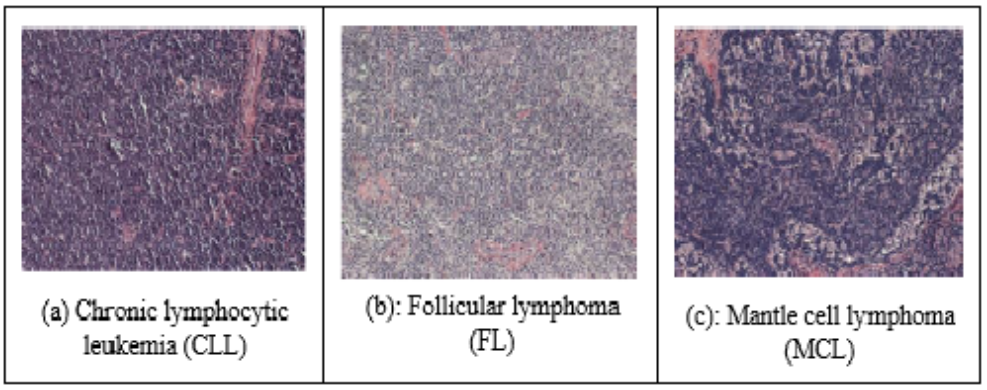


FIGURE 1.6: Lymphoma subtypes

1.7 Problems specific to histopathological image analysis

Histopathological image has some specific characteristics, which need computational methods to treat them. According to [5], they can be summarized as follow:

1.7.1 Enormous density of data

One of the principal challenges in analysis of digital histopathology data is the enormous density of data that the algorithms have to contend with, compared with radiological and other imaging modalities. For instance, the largest radiological datasets obtained on a routine basis are high-resolution chest CT scans comprising approximately 512 × 512 × 512 spatial elements or approximately 134 million voxels. A single core of prostate biopsy tissue digitized at 40× resolution is approximately 15,000 × 15,000 elements or approximately 225 million pixels. To put this in context, a single prostate biopsy procedure can comprise anywhere between 12 and 20 biopsy samples or approximately 2.5 - 4 billion pixels of data generated per patient study [38].

Images with large size often need to be resized into smaller size which is enough for sufficient distinction, as increase in the size of the input image results in the increase in the parameter to be estimated, the required computational power, and memory. However, resizing the entire WSI, which contains many cells, to a smaller size such as 256 × 256 would lead to the loss of information at cellular level, resulting in marked decrease of the identification accuracy. Therefore, the entire WSI is commonly divided into partial regions of about 256 × 256 pixels ("patches"), and each patch is analyzed independently, such as detection of regions of interest [5].

1.7.2 Different levels of information

Tissues are usually composed of cells, and different tissues show distinct cellular features. Information regarding cell shape is well captured in high-power field microscopic images, but structural information such as a glandular structure made of many cells are better captured in a lower-power field. The images are taken at multiple magnifications would each contain important information Figure 1.7. Pathologists diagnose diseases by acquiring different kinds of information from the cellular level to the tissue level by changing magnifications of a microscope. Even in image analysis, researches utilizing images at different magnifications exist [5]. However, it is difficult to handle the images at its original resolution directly, images are often resized to correspond to various magnifications and used as input for analysis. Regarding diagnosis, the most informative magnification is still controversial [39], but improvement in accuracy is sometimes

achieved by inputting both high and low magnification images simultaneously, probably depending on the types of diseases and tissues, and the used algorithm in analysis operation [40].

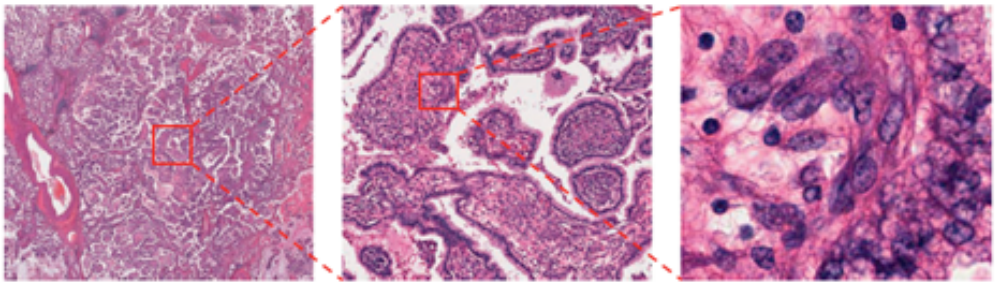


FIGURE 1.7: Multiple magnification levels of the same histopathological image. Right images show the magnified region indicated by red box on the left images. Left most image clearly shows papillary structure, and rightmost image clearly shows nucleus of each cell [5].

1.7.3 WSI as orderless texture-like image

Pathological images show repetitive pattern of minimum components (usually cells). Therefore, it is rather closer to texture than object. The analysis of these images need methods which utilize texture structure more intensively, such as gray level co-occurrence matrix [41].

1.7.4 Color variation and artifacts

WSIs are created through multiple processes as it is introduced in previous section; pathology specimens are sliced and placed on a slide glass, stained with Hematoxylin and Eosin, and then scanned. At each step undesirable effects, which are unrelated to the underlying biological factors, could be introduced. For example when tissue slices are being placed onto the slides, they may be bent and wrinkled; dust may contaminate the slides during scanning; blur attributable to different thickness of tissue sections may occur Figure 1.8; and sometimes tissue regions are marked by color markers [5].

Another serious artifact is color variation as shown in Figure 2.9. The sources of variation include different lots or manufacturers of reagents, thickness of tissue sections, staining conditions and scanner models.

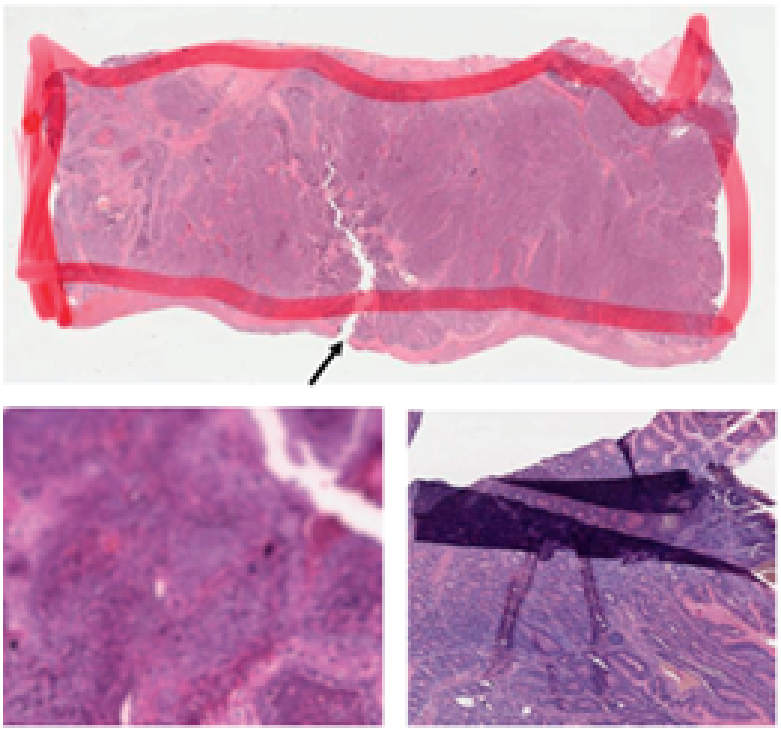


FIGURE 1.8: Artifacts in WSIs. Top: tumor region is outlined with red marker. The arrow indicates a tear possibly formed during the tissue preparation process. Left bottom: blurred image. Right bottom: folded tissue section [5].

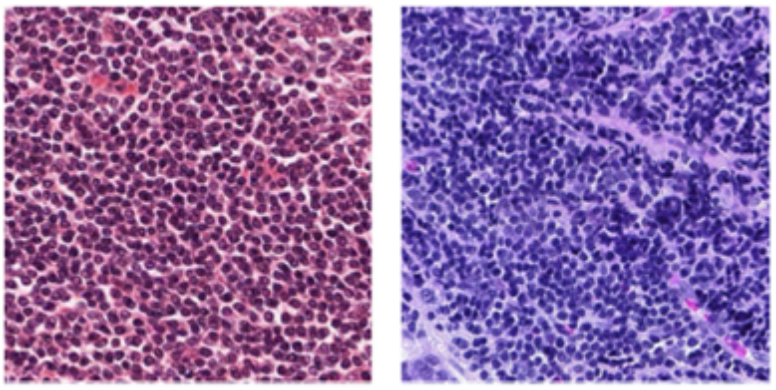


FIGURE 1.9: Color variation of histopathological images. Both of these two images show lymphocytes [5].

To address this issue, various methods have been proposed so far including conversion to gray scale, color augmentation , and color normalization [42]. Conversion to grayscale is the easiest way, but it ignores the important information regarding the color representation used routinely by pathologists. In contrast, color normalization tries to adjust the color values of an image on a pixel-by-pixel basis so that the color distribution of the source image matches to that of a reference image. However, as the components and composition ratios of cells or tissues in target and reference images differ in general, preprocessing such as nuclear detection using a dedicated algorithm to adjust the component is often required. For this reason, color normalization seems to be suitable when WSIs analyzed in the tasks contain, at least partially, similar compositions of cells or tissues.

1.8 Requirement of design Computer-aided diagnosis systems for histopathology

Computer-aided detection (CADe) and computer-aided diagnosis (CADx) refer to computerized procedures which assist doctors in the interpretation of medical images. CADe and CADx are similarly represented as CAD. Application of CAD for histopathology can be broadly categorized into three major groups [3]:

- Detection, segmentation, and quantification of important tissue structures, for examples: detection and segmentation of nuclei to analyze nuclear morphology as one of the hallmarks of cancerous conditions, segmentation of glandular structures and analysis of the morphology of the gland as the key criterion for cancer grading (e.g. colon cancer), and detection of cancer metastases in lymph node sections.

- Classification of histopathology imagery based on grade or lesion type, in this case, CAD may produce objective and accurate classification results to aid the decision of the pathologist. Examples applications are: Gleason scoring for prostate cancer, classification of breast proliferative lesions into benign, ductal carcinoma in-situ (DCIS), and invasive cancer.

- Disease diagnosis and prognosis based on the assessment of subtle sub-visual changes in the patterns of important structures in histopathological images that are invisible to or hard to recognize for human vision. This can potentially lead to early diagnosis and prognosis of a disease and can, in turn, facilitate the subsequent clinical management of patients.

The requirement to design CAD systems for histopathology that can be used in clinical practice necessitates the development of robust and high throughput algorithms that can operate at the WSI level. The underlying fundamental method that sits at the core of most CAD methodologies is machine learning. The ability of machine learning algorithms to discover and identify patterns and relationships between them from complex dataset has made them very popular for the development of computerized and computer-aided diagnostic systems.

1.9 Conclusion

Throughout this chapter, we focused on the definition of digital pathology field, whole slide imaging technology and the development of computer- aided or computerized diagnosis tools that can operate at the WSI level hence enabling the systems to be utilized in a practical clinical setting.

In the next chapter, we will introduce machine learning approaches as an alernative approaches overcoming the development challenges of the computerized and computer-aided diagnostic systems and that can provide state-of-the-art performance in different modalities in the field of Bio-medical imagining including Digital Pathology Image Analysis .

Chapter 2

MACHINE LEARNING

2.1 Introduction

Artificial Intelligence (AI) is part of the interdisciplinary information sciences area that develops and implements methods and systems that manifest cognitive behaviour [43], [44]. Since the 1950s, a small subset of Artificial Intelligence (AI), often called Machine Learning (ML), has revolutionized several fields in the last few decades. Deep Learning (DL)is a specific approach of ML, which deals with the training of deep neural networks. The relationship between AI, ML and DL is visualized in Figure 2.1.

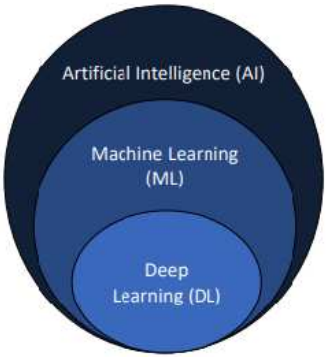


FIGURE 2.1: Relationship between AI, ML and DL [6]

In this chapter, we will present an understanding of the principles of machine learning, artificial neural networks and recent advances in convolutional neural networks, focusing on the applications of deep learning approaches in the DP tasks.

2.2 Machine learning concept

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

Machine learning models are provided experiences in the form of training data, and are tuned to produce accurate predictions for the training data by an optimization algorithm. The main goal of the models are to be able to generalize their learned expertise, and deliver correct predictions for new, unseen data. A model's generalization ability is typically estimated during training using a separate dataset, the validation set, and used as feedback for further tuning of the model. After several iterations of training and tuning, the final model is evaluated on a test set, used to simulate how the model will perform when faced with new, unseen data [45].

In recent years, machine learning has become more and more popular in research and has been incorporated in a large number of applications, including multimedia concept retrieval, medical image classification, video recommendation, social network analysis, text mining, and so forth [46]. For example, if a machine learning algorithm is applied for medical image classification tasks (Figure 2.1). Then it uses a set of input images (e.g., tumor images) and to some knowledge about these data (e.g., benign or malignant tumors), as training data, to identify the image features that, when used, will result in the correct classification of the image (that is,depicting benign or malignant tumor) as compared with the supplied labels for these input images (Figure 2.2.a).

Two types of image features can be extracted for image content representation; namely global features and local features, through information retrieval techniques. Global features (e.g., intensity, color and texture) describe an image as a whole. While, local features (e.g., Edges shapes) aim to detect key-points or interest regions in an image and describe them. Next, the desired features are selected and the noise is removed. These steps are used in every experiment of machine learning.

If the algorithm system optimizes its parameters such that its performance improves; that is, more test cases are diagnosed correctly, then it is considered to be learning that

task and can make a prediction. In this step, the learned model is applied to new images to assist radiologists in identifying the tumor type (Figure 2.2.b).

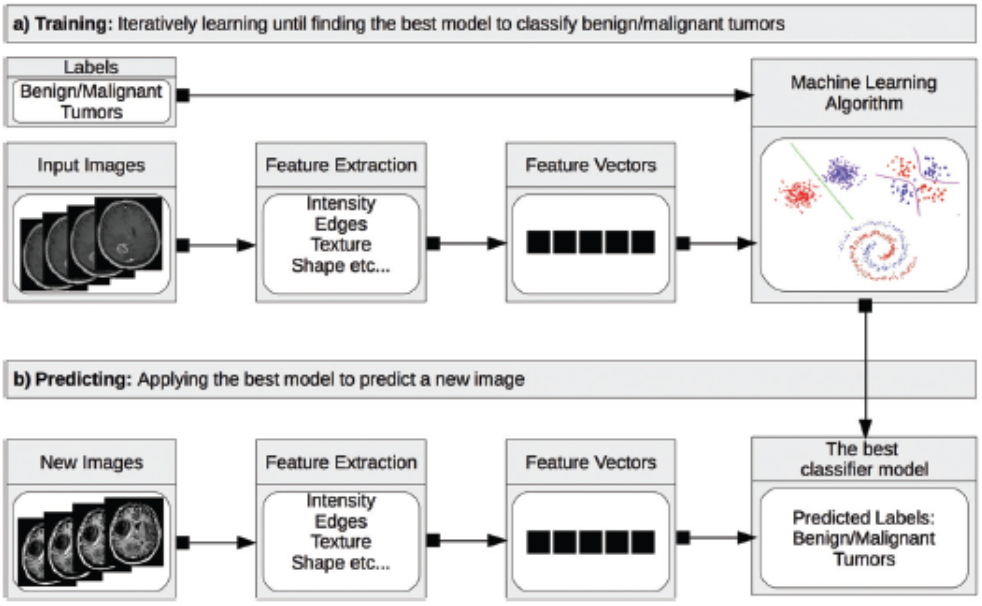


FIGURE 2.2: Machine learning model development and application model for medical image classification tasks [7]

The efficiency of machine-learning algorithms highly relied on the goodness of the representation of the input data. A bad data representation often leads to lower performance compared to a good data representation. Therefore, feature engineering has been an important research direction in machine learning for a long time, which focuses on building features from raw data and has led to lots of research studies. Furthermore, feature engineering is often very domain specific and requires significant human effort [47].

2.3 Types of machine learning algorithms

Depending on the nature of data and the objective of the study, machine learning algorithms can be broadly divided into: supervised learning, unsupervised learning, reinforcement learning and deep learning [26].

2.3.1 Supervised learning

Supervised learning is a learning technique that requires training with labeled data which has inputs and desired outputs. Given a database of training examples with

a specific target label (property) in the form $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ where x_i denotes the feature vector of the ith example and y_i is its label, the goal is to construct a model $g : X \rightarrow Y$ that can accurately predict the label Y for future instances of data X. When this target property is a continuous real value, the task is referred to as regression. Otherwise, when the target property is a finite set of discrete values, the task is referred to as classification.The common used techniques, in this category, are neural networks, k-nearest neighbors, support vector machines, decision trees and the naive Bayes algorithm [7].

2.3.2 Unsupervised learning

Unsupervised learning aims to discover implicit relationships in a given unlabeled training dataset. The most common unsupervised learning is clustering which automatically partitions the data into groups of similar items called clusters. K-means is one of the most used technique, in this category [48].

2.3.3 Reinforcement learning

Reinforcement learning enables learning from feedback received through interactions with an external environment. It is frequently used for robotics, gaming and navigation. For example, an agent interacts with its environment and learns its behavior based on the feedback it receives from its environment. Reward feedback known as reinforcement signal is required for the agent to learn its behavior. The agent must learn to act so as to maximize the total expected reward [45].

2.3.4 Deep learning

Deep learning is the advance phase of machine learning which mainly uses neural networks for learning and prediction of data. It is a group of different algorithms, which are used to design complex generalize system that can take any type of problems and give predictions. They use the deep graph with numerous processing layer, made up of many linear and nonlinear conversion [11].

The main common characteristic of deep learning methods is their focus on feature learning; automatically learning representations of data. This is the primary difference between deep learning approaches and more "classical" machine learning. Discovering features and performing a task is merged into one problem, and there fore both improved during the same training process [11] , [24] for general overviews field.

2.4 Artificial neural networks

Artificial neural networks (ANN) are brain inspired networks used in machine learning applications. They are composed of neurons. Each neuron is connected to other neurons through adaptive weights. The multi-layer perceptron (MLP) is a form of an ANN which the neurons are arranged in layers (see Figure 2.3). There are special layers for the input and output of the neural network. The input layer basically is a set of features connected by weights to the neurons in the first layer, and the output layer contains the output activations for the network, which gives the classification. The layers between the input and output layer are called hidden layers

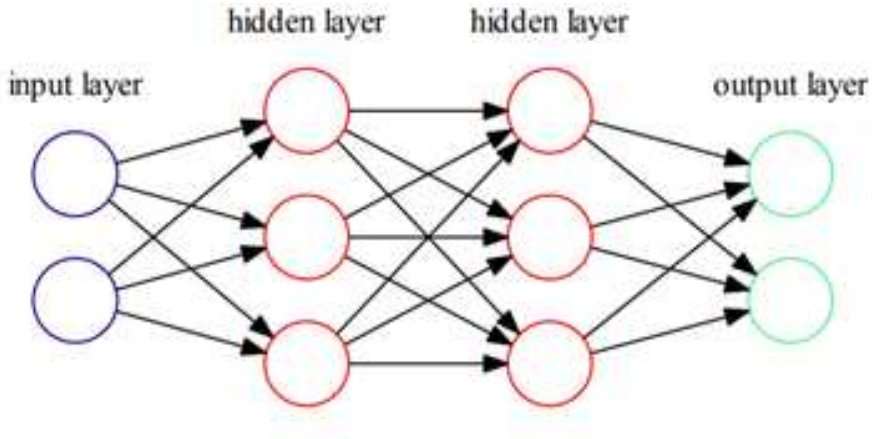


FIGURE 2.3: Schematic of a multi-layer perceptron

A neuron k (see Figure 2.4), in ANN, is a unit that computes the dot product of the inputs $X(x_1, x_2, ..., x_m)$ and their connection weights $W(\omega_1, \omega_2, \omega_3, ..., \omega_{km})$ and applies an activation function $\varphi(.)$ to generate the output y_k, in the form:

$$y_k = \varphi(v_k) + b_k \tag{2.1}$$

v_k is the linear combination of input signals, it is represented by:

$$v_k = \sum_{i=1}^{m} \omega_{ki} x_i \tag{2.2}$$

b_k is a bias which is added with a linear combiner of outputs v_k,

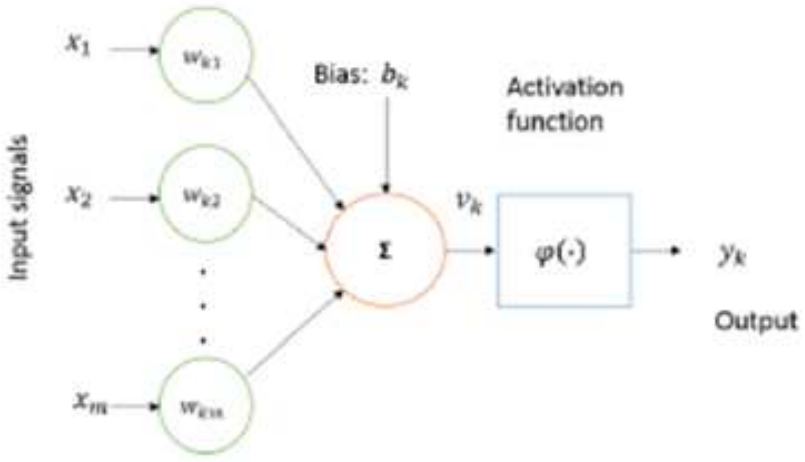


FIGURE 2.4: Basic model of a neuron

By layering the neurons and connecting them from an input layer to an output layer, the overall network represents a function $f : x \to y$ that maps the input signals that go into the input layer (layer 1) to an output signal that leaves the output layer (layer n). The goal of f is to approximate a target function f^* , e.g., a classifier $y = f^*(x)$ that maps an input x to a category y.

Learning is about finding the weights for the network that makes it solve a specific task in some optimal sense and is achieved by general-purpose learning procedures such as the backpropagation algorithm [49].

In the training process, the set of parameters (the weights, biases and thresholds) in all of the artificial neurons are adjusted in such a way that the output of f approximates the output of $f*$ with the best possible accuracy.

Training a neural network with backpropagation [49] is composed of two simple steps the feedforward and the backpropagation step. In the feedforward step a training case is classified using the current neural network. In the backpropagation step a classification

error is computed and propagated back through the neural network. The weights are updated based on the error, learning rate and gradient of the activation. Backpropagation is the method used to achieve gradient descent in neural networks.

2.5 Convolutional Neural Networks

The convolutional neural network (CNN) is a specialized type of neural network model designed for working with two-dimensional image data, although they can be used with one-dimensional and three-dimensional data. Central to the convolutional neural network is the convolutional layer that gives the network its name. This layer performs an operation called a "convolution".

In the context of a convolutional neural network, a convolution is a linear operation that involves the multiplication of a set of weights with the input, much like a traditional neural network.

The convolution operation can be visualized with the example in Figure 2.5. The kernel w slides across the input feature map I. At each location, the product between each element of the kernel and the input element it overlaps with is computed and the results are summed to obtain the output at the current location. The result is called the output feature map K.

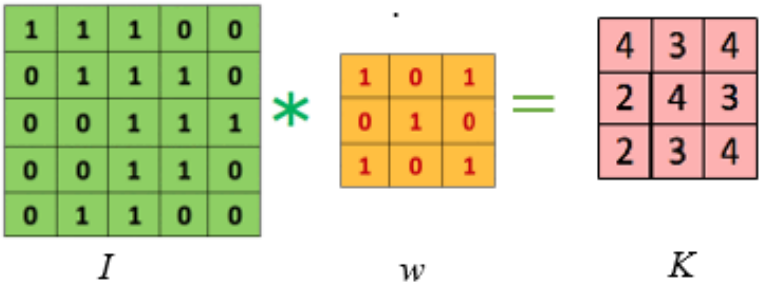


FIGURE 2.5: Convolution operation [8]

The following parameters can be adjusted in a convolutional layer [50]:

- the kernel size, also called filter size in neural networks;
- the step size, which is the distance between two consecutive positions of the kernel;

- zero padding, which is the number of zeros concatenated at the beginning and end of an axis.

The resulting effects of those parameter changes on the output are discussed in detail in [8]. There are numerous variants of CNN architectures in the literature. However, their basic components are very similar. The general structure of the deep convolutional neural network was introduced by LeCun [25]. His deep convolutional neural network architecture called LeNet is basically what is still being used today. It is an architecture composed of three types of layers, namely convolutional, pooling, and fully-connected layers Figure 2.6. The convolution layers interchanged with pooling layers [9].

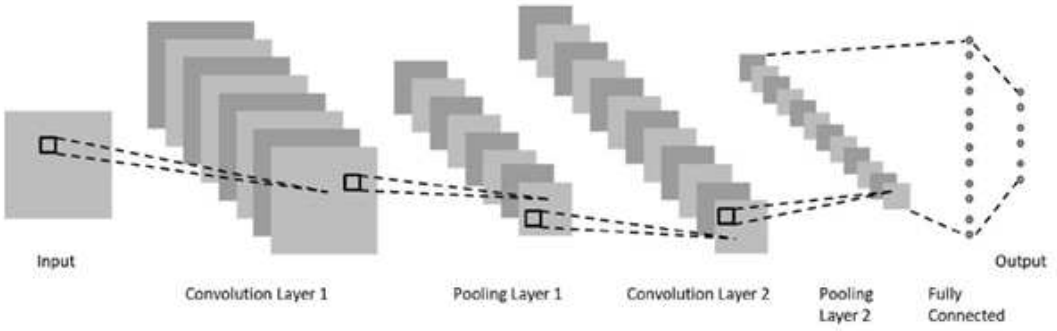


FIGURE 2.6: CNN with two convolution layers each followed by a pooling layer and one fully connected layer [9]

2.5.1 Convolutional layer

The convolutional layer aims to learn feature representations of the inputs.As shown in Figure 2.6, convolution layer is composed of several convolution kernels which are used to compute different feature maps. Specifically, each neuron of a feature map is connected to a region of neighbouring neurons in the previous layer. Such a neighbourhood is referred to as the neuron's receptive field in the previous layer. The new feature map can be obtained by first convolving the input with a learned kernel and then applying an element-wise nonlinear activation function on the convolved results.

Note that, to generate each feature map, the kernel is shared by all spatial locations of the input. The complete feature maps are obtained by using several different kernels. Mathematically, the feature value at location (i, j) in the k^{th} feature map of l^{th} layer, $z^l_{i,j,k}$, is calculated by:

$$z^l_{i,j,k} = \omega^{l^T}_k x^l_{i,j} + b^l_k \tag{2.3}$$

where $\omega^{l^T}_k$ and b^l_k are the weight vector and bias term of the k^{th} filter of the l^{th} layer respectively, and $x^l_{i,j}$ is the input patch centered at location (i, j) of the l^{th} layer. Note that the kernel $\omega^{l^T}_k$ that generates the feature map $z^l_{:,:,k}$ is shared. Such a weight sharing mechanism has several advantages such as it can reduce the model complexity and make the network easier to train.

The activation function introduces nonlinearities to CNN, which are desirable for multi-layer networks to detect nonlinear features. Let $a(\cdot)$ denote the nonlinear activation function. The activation value $a^l_{i,j,k}$ of convolutional feature $z^l_{i,j,k}$ can be computed as:

$$a^l_{i,j,k} = a(z^l_{i,j,k}) \tag{2.4}$$

Typical activation functions are sigmoid (sig), tanh and rectified linear unit ReLu [51] see Figure 2.7, where x is the feature value $z^l_{i,j,k}$ at location (i, j) in the k^{th} feature map of l^{th} layer.

$$sig(x) = \frac{1}{1 + e^{-x}} \tag{2.5}$$

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{2.6}$$

$$\mathrm{Re}\, Lu(x) = \max(0, x) \tag{2.7}$$

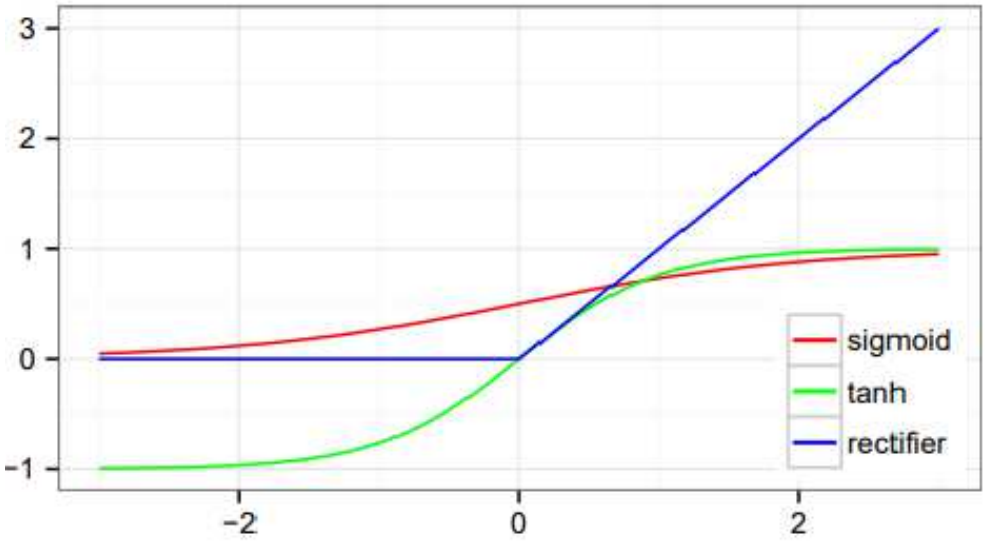


FIGURE 2.7: Activation function values

2.5.2 Pooling layer

The pooling layer aims to achieve shift-invariance by reducing the resolution of the feature maps. It is usually placed between two convolutional layers. Each feature map of a pooling layer is connected to its corresponding feature map of the preceding convolutional layer. Denoting the pooling function as $pool(\cdot)$, for each feature map $a^l_{:,:,k}$ we have:

$$y^l_{i,j,k} = pool(a^l_{m,n,k}) \quad \forall (m,n) \in R_{ij} \tag{2.8}$$

Where R_{ij} is a local neighbourhood around location (i, j). One of the most popular pooling operations is called max pooling [52], which returns the maximum input within a local neighbourhood R_{ij} . Figure 2.8 shows the operation of max pooling.

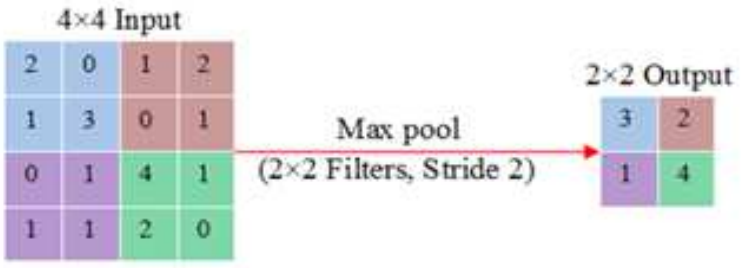


FIGURE 2.8: Max Pooling Operation

2.5.3 Fully-connected layers

After several convolutional and pooling layers, there may be one or more fully-connected layers which aim to perform high-level reasoning [53]. They take all neurons in the previous layer and connect them to every single neuron of current layer to generate global semantic information.

The last layer of CNNs is an output layer, or classification Layer, for classification tasks. This is the fully connected layer, which computes the score of each class from the extracted features from a convolutional layer in the preceding steps.

The final layer feature maps are represented as vectors with scalar values,which are passed to the fully connected layers that outputs a vector of K dimensions, where K is the number of classes that the network will be able to predict. This vector contains the probabilities for each class of any image patch being classified.

There are no strict rules on the number of layers which are incorporated in the network model [51]. However, in most cases, two to four layers have been observed in different architectures including LeNet [6], AlexNet [54], and VGG Net [55].

2.6 Network parameters and required memory for CNN

The number of computational parameters is an important metric to measure the complexity of a deep learning model [11]. The size of the output feature maps can be formulated as follows:

$$M = \frac{N - F}{S} + 1 \tag{2.9}$$

Where N refers to the dimensions of the input feature maps, F refers to the dimensions of the filters or the receptive field, M refers to the dimensions of output feature maps, and S stands for the stride length. Padding is typically applied during the convolution operations to ensure the input and output feature map have the same dimensions. The amount of padding depends on the size of the kernel. The number of rows and columns, for padding, is determined by :

$$P = \frac{F - 1}{2} \tag{2.10}$$

Here P is the amount of padding and F refers to the dimension of the kernels. Several criteria are considered for comparing the models. However, in most of the cases, the number of network parameters and the total amount of memory are considered. The number of parameters ($Parm_l$) of l^{th} layer is calculated based on the following equation:

$$Parm_l = (F \times F \times FM_{l-1}) \times FM_l \tag{2.11}$$

If bias is added with the weights, then the above equation can be written as follows:

$$Parm_l = (F \times (F + 1) \times FM_{l-1}) \times FM_l \tag{2.12}$$

Here the total number of parameters of l^{th} layer can be represented with P_l, FM_l is for the total number of output feature maps, and FM_{l-1} is the total number of input feature maps or channels. Thus, the amount of memory (Mem_l) needs for the operations of the l^{th} layer can be expressed as:

$$Mem_l = (N_l \times N_l \times FM_l) \tag{2.13}$$

2.7 Training a network

Training a network is a process of finding kernels in convolution layers and weights in fully connected layers, which minimize differences between output predictions and given ground truth labels on a training dataset.

Backpropagation algorithm is the method commonly used for training neural networks where loss function and gradient descent optimization algorithm play essential roles. A model performance under particular kernels and weights is calculated by a loss function through forward propagation on a training dataset, and learnable parameters, namely kernels and weights, are updated according to the loss value through an optimization algorithm called backpropagation and gradient descent, among others (Figure 2.9).

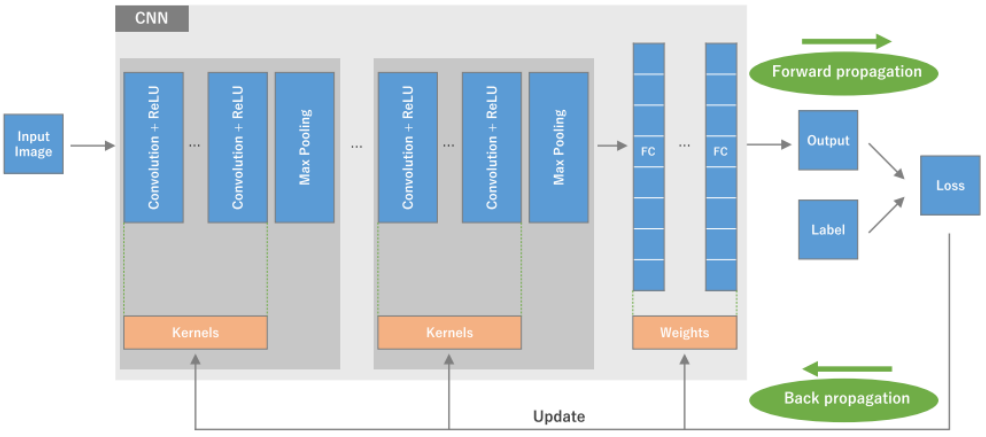


FIGURE 2.9: General architecture and training process of Convolutional neural network. A CNN is composed of a stacking of several building blocks: convolution layers, pooling layers (e.g., max pooling), and fully connected (FC) layers [10]

2.7.1 Loss function

A loss function, also referred to as a cost function, measures the compatibility between output predictions of the network through forward propagation and given ground truth labels.

Softmax loss is a commonly used loss function which is essentially a combination of multinomial logistic loss and softmax. Given a training set $\{(x^{(i)}, y^{(i)}); i \in 1, ..., N, y^{(i)} \in 1, ..., K\}$, where $x^{(i)}$ is the i^{th} input image patch, and $y^{(i)}$ is its target class label among the K classes. The prediction of j^{th} class for i^{th} input is transformed with the softmax function:

$$p_j^{(i)} = \frac{e^{z_j^{(i)}}}{\sum_{l=1}^{k} e^{z_l^{(i)}}} \tag{2.14}$$

where $z_j^{(i)}$ is usually the activations of a densely connected layer, so $z_j^{(i)}$ can be written as:

$$z_j^{(i)} = w_j^T + b_j \tag{2.15}$$

Softmax turns the predictions into non-negative values and normalizes them to get a probability distribution over classes [51]. Such probabilistic predictions are used to compute the multinomial logistic loss, i.e., the softmax loss, as follows:

$$\rho_{soft\max} = -\frac{1}{N}(\sum_{i=1}^{N} \sum_{j=1}^{k} 1\left\{y^i = j\right\} \log p_j^{(i)} \tag{2.16}$$

2.7.2 Gradient descent

Gradient descent is commonly used as an optimization algorithm that iteratively updates the learnable parameters, i.e., kernels and weights, of the network so as to minimize the loss.

The gradient of the loss function provides us the direction in which the function has the steepest rate of increase, and each learnable parameter is updated in the negative direction of the gradient with an arbitrary step size determined based on a hyperparameter called learning rate [10](Figure 2.10).

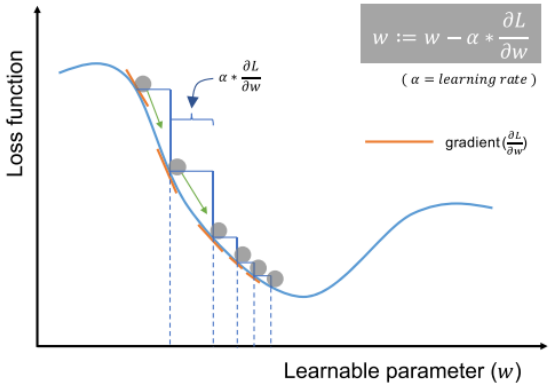


FIGURE 2.10: Gradient descent and iterative updating of learnable parameters to minimize loss, [10]

The gradient is, mathematically, a partial derivative of the loss with respect to each learnable parameter, and a single update of a parameter is formulated as follows:

$$w = w - \alpha \frac{\delta L}{\delta w} \tag{2.17}$$

Where w stands for each learnable parameter, α stands for a learning rate, and L stands for a loss function. It is of note that, in practice, alearning rate is one of the most important hyperparameters to be set before the training starts.

In practice, for reasons such as memory limitations, the gradients of the loss function with regard to the parameters are computed by using a subset of the training dataset called mini-batch, and applied to the parameter updates. This method is called mini-batch gradient descent, also frequently referred to as stochastic gradient descent (SGD), and a mini-batch size is also a hyperparameter. In addition, many improvements on the gradient descent algorithm have been proposed and widely used [56] , such as SGD with momentum, RMSprop, and Adam.

2.8 Advanced Training Techniques

The advanced training techniques or components which need to be considered carefully for efficient training of DL approach. There are different advanced techniques to apply for training a deep learning model better [11], such as regularization and batch normalization.

2.8.1 Regularization

Overfitting is an unneglectable problem in deep CNNs, it refers to a situation where a model learns statistical regularities specific to the training set, i.e., ends up memorizing the irrelevant noise instead of learning the signal, and, therefore, performs less well on a subsequent new dataset. Overfitting can be effectively reduced by regularization.

There are different regularization approaches that have been proposed in the past few years for deep CNN. The simplest but efficient approach called "dropout" was proposed by Hinton in 2012 [57]. In Dropout a randomly selected subset of activations is set to zero within a layer [11]. The dropout concept is shown in Figure 2.11.

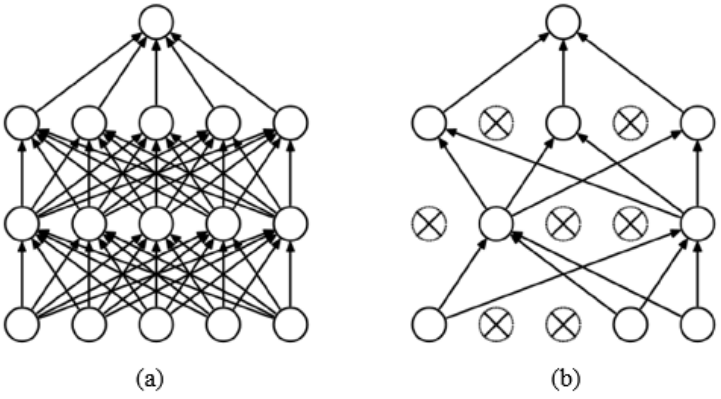


FIGURE 2.11: Dropout concept, (a): A densely connected neural net with two hidden layers, (b): An example of a smaller network produced by applying dropout (right); Crossed units have been deactivated by dropout [11]

2.8.2 Batch Normalization

Data normalization is usually the first step of data preprocessing. Global data normalization transforms all the data to have zero-mean and unit variance. However, as the data flows through a deep network, the distribution of input to internal layers will be changed, which will lose the learning capacity and accuracy of the network. Ioffe et al in [58] propose an efficient method called Batch Normalization (BN) to partially alleviate this phenomenon. It accomplishes the so-called covariate shift problem by a normalization step that fixes the means and variances of layer inputs where the estimations of mean and variance are computed after each mini-batch rather than the entire training set [51].

2.9 Popular CNN Architectures

In general, the architectures of most deep convolutional neural networks consist of stacks of several layers and max-pooling layers followed by a fully connected and SoftMax layers at the end. A detail description of popular [11].

Some examples of such models are LeNet [25], AlexNet [54], and VGG Net [55]. Other alternatives and more efficient advanced architectures have been proposed including GoogLeNet [59], Residual Networks [60], DenseNet [61] and FractalNet [12]. The basic building components (convolution and pooling) are almost the same across these architectures. However, some topological differences are observed in the modern deep learning architectures [11].

Of the many DCNN architectures, AlexNet [54], VGG Net [55], GoogLeNet [59], DenseNet [61] and FractalNet [12] are generally considered the most popular architectures because of their state-of-the-art performance on different benchmarks for object recognition tasks. Among all of these structures FractalNet [12].

2.9.1 LeNet

LeNet was introduced by Yan LeCun for digit recognition. The basic configuration of LeNet-5 is (see Figure2.12): 2 convolutions (conv) layers, 2 sub-sampling layers, 2 fully

connected layers, and an output layer with the Gaussian connection. The total number of weights and Multiply and Accumulates (MACs) are 431k and 2.3M respectively.

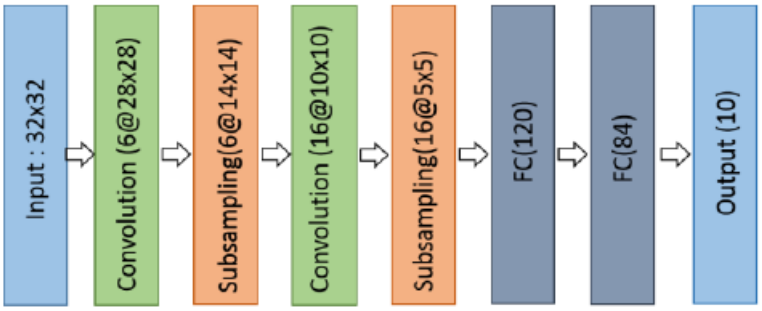


FIGURE 2.12: Architecture of LeNet model

2.9.2 AlexNet

The architecture of AlexNet [54] is shown in Figure 2.13. The first convolutional layer performs convolution (conv) max-pooling (MXP) with Local Response Normalization (LRN) where 96 different receptive filters are used that are 11×11 in size. The max pooling operations are performed with 3×3 filters with a stride size of 2. The same operations are performed in the second layer with 5×5 filters. 3×3 filters are used in the third, fourth, and fifth convolutional layers with 384, 384, and 296 feature maps respectively. Two fully connected (FC) layers are used with dropout followed by a Softmax layer at the end. The AlexNet was the first to show deep learning was effective in computer vision tasks.

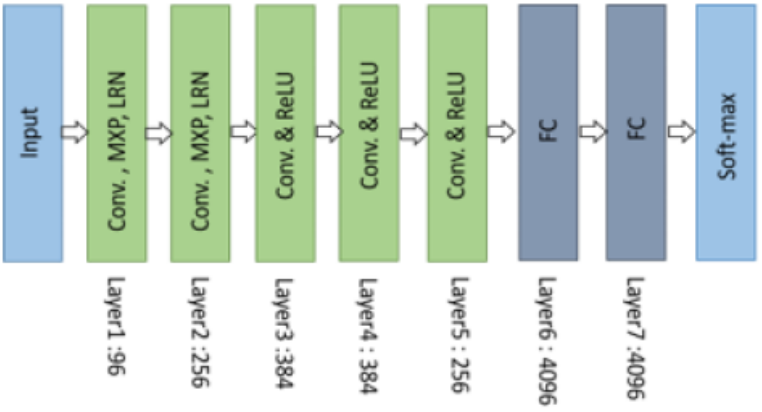


FIGURE 2.13: Architecture of AlexNet [11]

2.9.3 VGGNet

The VGG architecture [55] consists of two convolutional layers both of which use the ReLU activation function. Following the activation function is a single max pooling layer and several fully connected layers also using a ReLU activation function. The final layer of the model is a Softmax layer for classification.

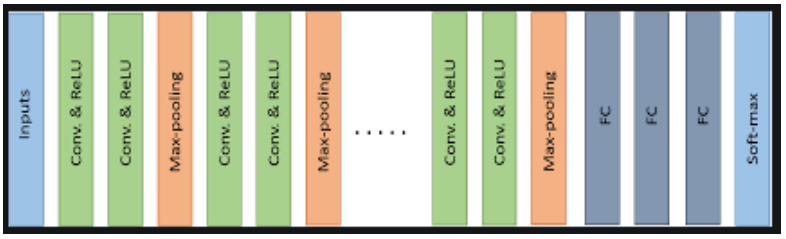


FIGURE 2.14: Basic building block of VGG network [11]

2.9.4 GoogLeNet

GoogLeNet improved the state of the art recognition accuracy using a stack of Inception layers (seen in Figure 2.15) that had variable receptive fields, which were created by different kernel sizes. These kernels allowed for dimensionality reduction before computationally expensive layers.

GoogLeNet consisted of 22 layers in total, which was far greater than any network before it. However, the number of network parameters GoogLeNet used was much lower than its predecessor AlexNet or VGG.

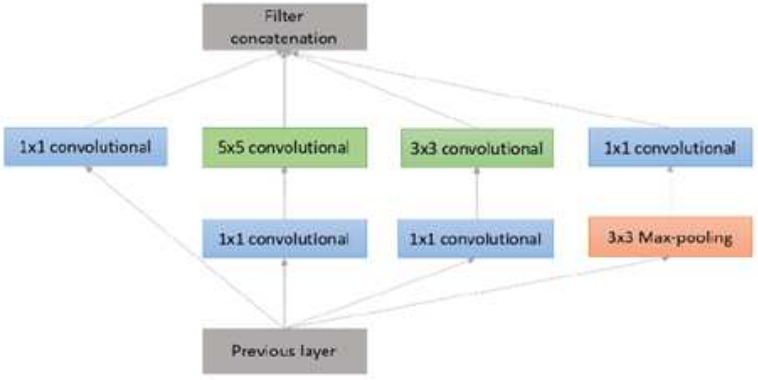


FIGURE 2.15: Inception layer with dimension reduction [11]

2.9.5 Residual Network (ResNet)

ResNet is developed with many different numbers of layers; 34, 50,101, 152,and even 1202. The popular ResNet50 contained 49 convolution layers and 1 fully connected layer at the end of the network. The basic block diagram of the ResNet architecture is shown in Figure 2.16. ResNet is a traditional feedforward network with a residual connection

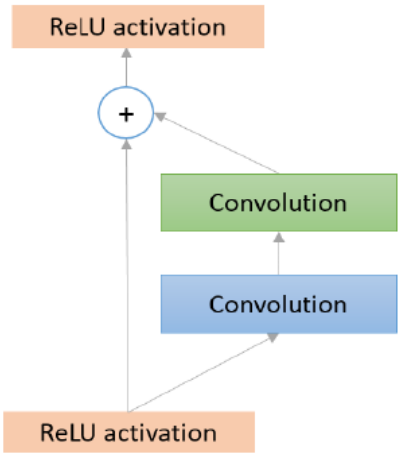


FIGURE 2.16: Basic diagram of Residual block [11]

2.9.6 Densely Connected Network (DenseNet)

DenseNet consists of densely connected CNN layers, the outputs of each layer are connected with all successor layers in a dense block. Therefore, it is formed with dense connectivity between the layers rewarding it the name "DenseNet". This concept is efficient for feature reuse, which dramatically reduces network parameters. DenseNet consists of several dense blocks and transition blocks, which are placed between two adjacent dense blocks. The conceptual diagram of a dense block is shown in Figure 2.17.

2.9.7 FractalNet

The architecture of FractalNet is an advanced and alternative architecture of ResNet model, which is efficient for designing large models with nominal depth, but shorter

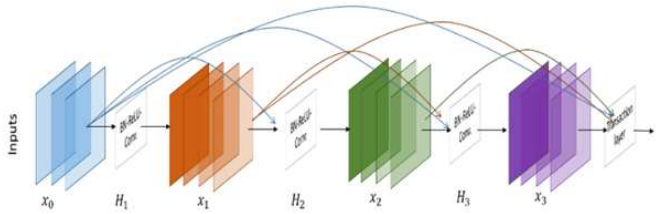


FIGURE 2.17: A 4-layer Dense block [11]

paths for the propagation of gradient during training. This concept is based on drop-path which is another regularization approach for making large networks. As a result, this concept helps to enforce speed versus accuracy tradeoffs. The basic block diagram of FractalNet is shown in Figure 2.18.

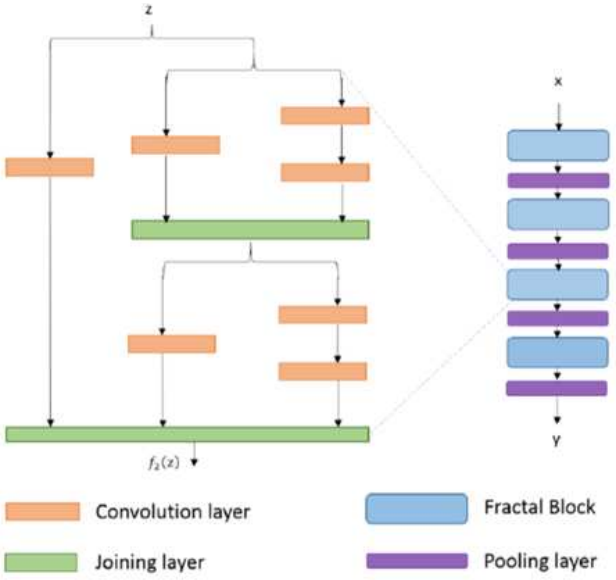


FIGURE 2.18: Detailed FractalNet module on the left and FractalNet on the right [12]

2.10 Deep learning for digital pathology image analysis

Deep learning is now one of the most active fields in machine learning and has been shown to improve performance in image and speech recognition [57], [46], natural language understanding [62], [63], and most recently, in computational biology [64], [27], [65], [66].

The potential of deep learning in high-throughput biology is clear [64]: in principle, it allows to better exploit the availability of increasingly large and high-dimensional data sets by training complex networks with multiple layers that capture their internal

structure. The learned networks discover high-level features, improve performance over traditional models, increase interpretability and provide additional understanding about the structure of the biological data.

Most methods in literature, for the analysis of histological images, are based on DL networks (e.g., AlexNet [54], ResNet [67], that automatically learn features that optimally represent the data for the problem at hand. In this context, we can cite some works which are summarized in Table 2.1.

In [4], a largest comprehensive study of DL approaches in DP, is presented, in this work AlexNet was adopted for solving different Digital Pathology Image Analysis (DPIA) problems that are evaluated on different publicly available benchmark datasets [1]. related to seven different tasks in digital pathology. The tasks include Lymphoma classification, Invasive Ductal Carcinoma (IDC) detection, Nuclei segmentation, Epithelium segmentation, Tubule segmentation, Lymphocyte detection, and Mitosis detection. Two performance metrics (accuracy, F-score) are used to evaluated the performances of each scenario. They have achieved 0.83, 0.84 and 0.83 testing F-score for nuclei, epithelium and tubule segmentation respectively, 0.90, 0.53 and 0.7648 testing F-score for lymphocyte, mitosis and invasive ductal carcinoma detection respectively, and 96.59 testing accuracy for Lymphoma classification.

The AlexNet model was also adopted in [68], in order to discriminate between benign and malignant breast cancer tumors. This approach outperformed the previously reported results obtained by other machine learning models trained with hand-crafted textural descriptors on the BreaKHis dataset [69].

Three different configurations of ResNet were used by the authors in [70] for the multi-classification of breast cancer obtaining a remarkable performance on the images provided for the ICIAR 2018 BACH Challenge [71].

In [72] a deep learning approach is proposed for two different use cases: the detection of invasive ductal carcinoma in breast histological images and the classification of lymphoma subtypes. Both use cases have been addressed by adopting a residual convolutional neural network. The performances of each scenario have been evaluated on public

[1] http://www.andrewjanowczyk.com/use-case-7-Lymphoma-sub-type-classification/

datasets[2]. They have obtained 97.67 testing accuracy for Lymphoma classification and 81.45 testing F-score for invasive ductal carcinoma detection.

In [65], the authors are interested in the distinction between follicular lymphoma (FL), the second most common subtype of lymphoma, and benign follicular hyperplasia (FH). They proposed a stochastic Bayesian neural network (BNN) to classify morphological changes occurring in FL or FH lymph nodes on H&E digital slides. In [66], a Semantic Segmentation model of Histopathological Slides called EU-Net is proposed for the classification of Cutaneous Lymphoma and Eczema.

References	DL networks	Addressed tasks
[4]	AlexNet	Lymphoma classification
		Invasive Ductal Carcinoma (IDC) detection
		Nuclei segmentation
		Epithelium segmentation
		Tubule segmentation
		Lymphocyte detection
		Mitosis detection
[68]	AlexNet	benign and malignant breast cancer tumors discrimination
[70]	ResNet	Detection of invasive ductal carcinoma in breast histological images
		Classification of lymphoma subtypes
[65]	ResNet	follicular lymphoma (FL), and benign follicular hyperplasia (FH) distinction
[66]	EU-Net	Classification of Cutaneous Lymphoma and Eczema

TABLE 2.1: Summary of works focused on Deep learning for digital pathology image analysis

[2]http://www.andrewjanowczyk.com/use-case-7-Lymphoma-sub-type-classification/

2.11 Conclusion

In this chapter, we have provided details of background study on deep learning convolutional neural networks and outlined the basic concepts of Convolutional Neural Networks and the advanced techniques to apply for training a deep learning model better. In addition, we have presented survey of the application of DL approaches in DP tasks.

From this study, we can conclude that:

- The choice of a DL network and the training strategy to apply for a given task, depends on the type of pathological analysis to be performed.
- For a specific task, the performance of the different approaches depends on the adopted network and the developed training strategy.
- Some work have adopted the same network architecture on the same dataset with different training strategies, for a specific task, but they got completely different results

In the next chapter, we will use the concepts of DL with a CNN algorithm to build non-Hodgkin lymphomas model that can accurately classify the subtypes of non-Hodgkin lymphomas, on whole-slide images, which correspond to 70% of lymphomas cases and represent a major challenge for pathologists due to its intrinsic complexity.

Chapter 3

DEEP NEURAL NETWORK FOR NON-HODGKIN LYMPHOMAS CLASSIFICATION

3.1 Introduction

A precise classification on pathological images is an important line of research, as for the subtypes of Non-Hodgkin's Lymphomas (NHL). In this chapter, we explore how DL can be used to accurately classify subtypes of NHL concerning chronic lymphocytic leukemia (CLL), follicular Lymphoma (FL) and mantle cell Lymphoma (MCL), on whole slide images, which is the subject of our master's project.

At the beginning of this chapter, we describe the problematic and the objectives of our master's project, and then we give an overview on Lymphoma's cancer, and the general form of construction of a DL architecture. Finally, we present the design of our classification system.

3.2 Motivation and problem statement

Lymphoma is a type of cancer affecting the lymphatic system classified into at least 38 subtypes [20]. These subtypes are divided into Hodgkin Lymphomas (HL) and Non-Hodgkin Lymphomas (NHL) in accordance with their morphological, immunophenotypical, genetical and clinical features. Global records of 2012 show that 199 thousand NHL-related deaths and 254 deaths resulting from HL occurred during that year [21].

Chronic Lymphocytic Leukemia (CLL), Follicular Lymphoma (FL) and Mantle Cell Lymphoma (MCL) are part of the NHL class. This class corresponds to 70% of Lymphomas cases and due to its wide range of clinical presentations and histopathological features, its classification and segmentation still represent a major challenge for pathologists [23].

In addition, the analyzing histopathological images, in particular, WSIs related of these NHL subtypes is, however, complicated by several factors. Firstly, histological stains show a large variability in their color and intensity. Such variations can potentially hamper the effectiveness of quantitative image analysis. Secondly, tissue preparation and digitization usually generates lots of artifacts which causes challenges for automated analysis. Finally, the large size of a WSI as well as the large number of heterogeneous structures that need to be analyzed inside a WSI make development of an accurate and reliable computer-aided detection for histopathology a challenge.

Deep learning-based diagnostic systems have recently provided automated methods for histopathology image analysis [73], [74], which may reduce inter- and intra-observer variability in cancer diagnosis through an objective analysis.

In our project, we want to use Deep Learning with a convolutional neural network (CNN) algorithm to build a NHL classification model that can accurately classify the subtypes of NHL (CLL, FL, MCL) on whole-slide images to help pathologists with this characterization and diagnosis.

In this context, our project is undertaken with the following objectives:

- Making decisions on selecting proper CNN for NHL datasets
- Accurately classify NHL subtypes, including chronic lymphocytic leukemia (CLL), follicular Lymphoma (FL) and mantle cell Lymphoma (MCL), on whole slide images,

- Development of an accurate and reliable computer-aided detection for histopathology.

3.3 Lymphoma

Lymphoma is one of the most common cancer whether in less developed or developing countries [22], which is a type of malignant cancer and begins in infection-fighting cells of the immune system, called lymphocytes. These cells are in the lymph nodes, spleen, thymus, bone marrow, and other parts of the body (see Figure 3.1).

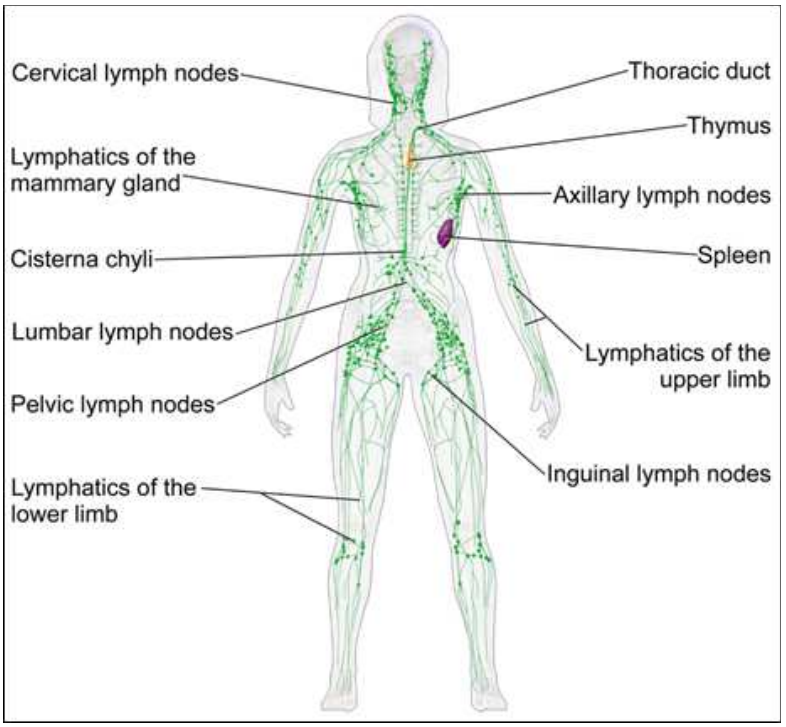


FIGURE 3.1: Diagram of vessels and organs in the lymphatic system

Originating from bone marrow, lymphocytes are subdivided into B lymphocytes, T lymphocytes and null cells. These type of cancer may develop in B cells as well as in T cells, however, 85% of cases are derived from B lymphocytes [21] . Development, activation and differentiation of normal B cells can be divided into three stages: pre-germinal center, germinal center and post-germinal center. Figure 3.2 represents these B cell development stages and a few Lymphomas related to each one of them [13].

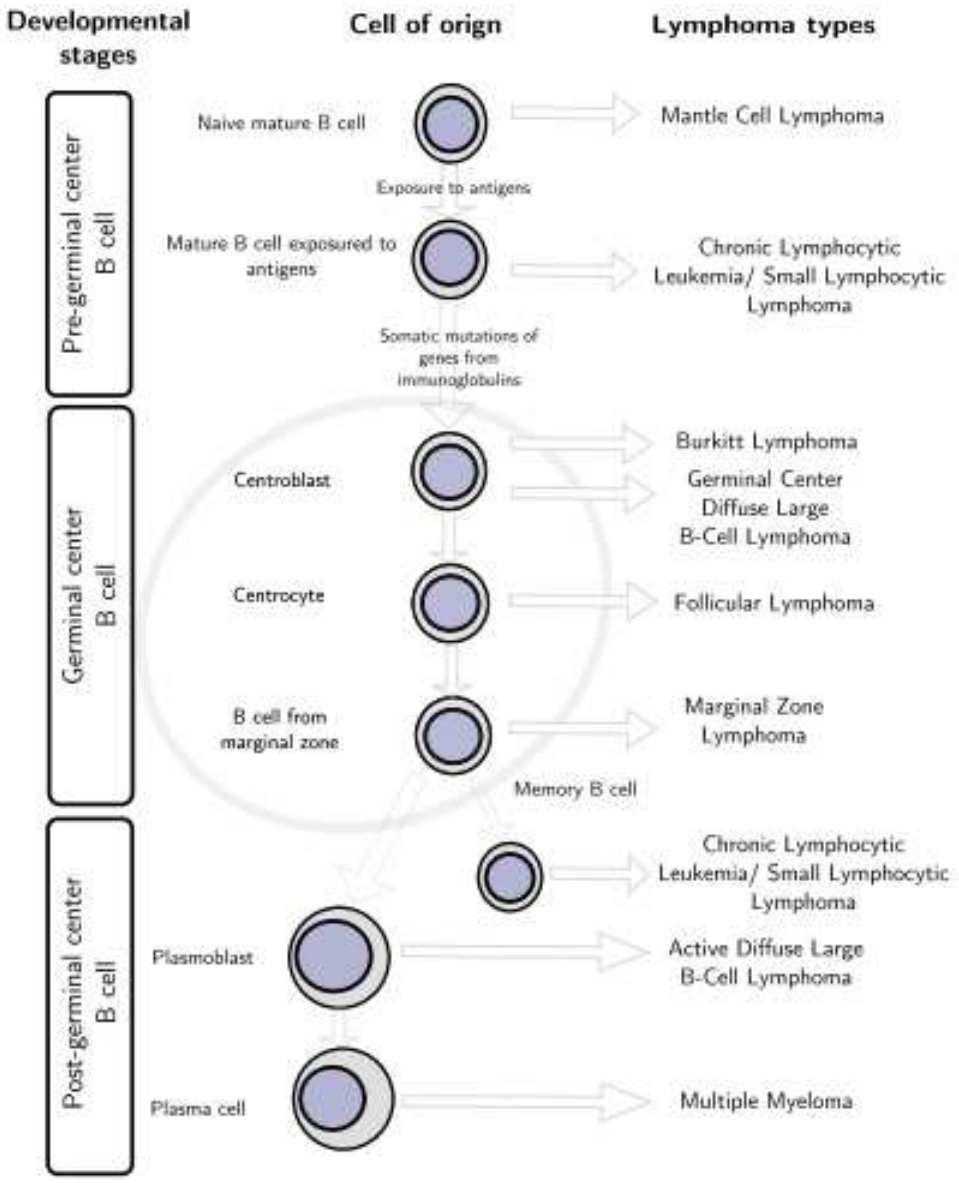


FIGURE 3.2: Stages of development, activation and differentiation of B cells related to different Lymphomas [13]

The two main types of Lymphoma are Hodgkin Lymphoma (HL) and Non-Hodgkin Lymphoma (NHL)[22]. The differences in these two types of Lymphoma are certain unique characteristics of the different Lymphoma cells. However, most people with Lymphoma have the Non-Hodgkin Lymphoma [13].

Non-Hodgkin Lymphoma is further classified into a variety of subtypes based on the cell of origin (B-cell or T-cell), and the cell chacteristics. The subtype of Non-Hodgkin Lymphoma predicts the necessity of early treatment, the response to treatment, the type of treatment required, and the prognosis [22].

A biopsy is essential in order to exclude or diagnose Lymphoma types. The biopsy is a minor operation (Figure 3.3). It takes a sample of tissue (made up of cells) from the body for examination in a laboratory. The tissue sample itself is sometimes also known as a 'biopsy' or a 'biopsy sample'. To diagnose NHL, a specialist analyses tissue samples stained with, for example, Hematoxylin&Eosin (H&E) to identify cancerous regions. These distinctions are essential for disease monitoring, identification of its stage and orientation towards appropriate treatments for the patient [20].

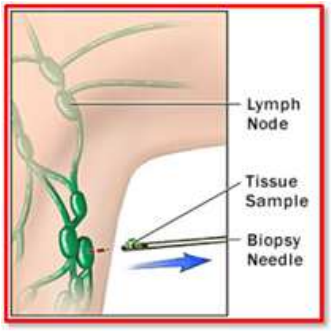


FIGURE 3.3: Biopsy operation

Chronic Lymphocytic Leukemia (CLL), Follicular Lymphoma (FL) and Mantle Cell Lymphoma (MCL) are subtypes of NHL [22]:, where differentiation between them in H&E , in some cases, is difficult by pathologists. Example of these types are extracted from curated dataset NIA [1] (see Figure 3.4)

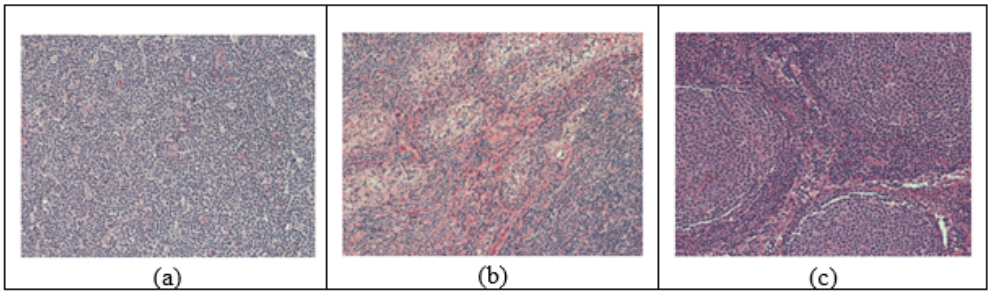


FIGURE 3.4: Histological images, of with 20 magnification and 1388 1040 pixels, for subtypes of NHL. (a): CLL, (b): MCL, and (c), extracted from curated dataset NIA

[1] http://www.andrewjanowczyk.com/use-case-7-Lymphoma-sub-type-classification/

3.3.1 Chronic Lymphocytic Leukemia

CLL is slow-growing, and starts in the blood and bone marrow. It is considered lymphoproliferative due to lymphoid cells proliferation. These cells are in their maturing stage but they are immunologically incompetent. In most cases,it is characterized by small lymphocytes with regular nuclei and condensed chromatin, without evident nucleoli and scant cytoplasm. Furthermore, the number of prolymphocytes, characterized by their medium size with abundant cytoplasm and evident nucleoli, is less than 10% [21]. CLL may present some similarities with MCL, however these cases can be distinguished by cellular features. Small lymphocytes with condensed chromatin are predominant, with variations in their nuclear morphologies [22].

The CLL diagnosis usually begins with lymphocytes counting, that must correspond to at least5 10^9 B lymphocytes/L of peripheral blood. The identification of CLL cells in blood is based on the search for small and mature lymphocytes, with a thin border of cytoplasm and a dense nucleus with undistinguished nucleoli.

3.3.2 Follicular Lymphoma

FL is a slow-growing form of B-cell Lymphoma. About 20% to 30% of all non-Hodgkin's Lymphomas are follicular Lymphoma (FL). This cancer usually starts in people who are over 65. It usually grows in the lymph nodes and bone marrow.

FL diagnosis is based on lymph node biopsies and must be obtained according to the World Health Organization classification considering the centroblasts absolute number in the neoplastic follicle [21]:

- Grade I: between 0 and 5 centroblasts;
- Grade II: between 6 and 15 centroblasts;
- Grade III: more than 15 centroblasts.

3.3.3 Mantle Cell Lymphoma

MCL is a rare and fast-growing cancer. About 6% of all non-Hodgkin's Lymphomas are this type. MCL starts in B cells of the "mantle zone" (an area on the outer edge of the

lymph nodes). This cancer often grows in the lymph nodes, bone marrow, and spleen. MCL is diagnosed through the identification of tumors with a morphological analysis of irregular cell nuclei with small/medium size. However, MCL lymphocytes can be presented by variant morphologies, such as small round cells and marginal zone-like lymphocytes. A minority of these cases are correctly identified, which makes advisable the review by a pathologist [22].

3.4 Construction of a DL architecture

When applying DL to a DP image analysis task four high-level modules need to be considered [4]:

3.4.1 Casting

To design an appropriate network such as input patch size, number of layers,and convolutional attributes we can attempt the popular and successful networks (for example AlexNet networks) with the popular open-source *DL* framework. However, finding the most successful network configuration for a given problem can be a difficult challenge given the total number of possible configurations one could avail of and also the concomitant amount of time for training and testing the network.

3.4.2 Patch generation

Once the network is defined, which involves locking down input sizes, image patches need to be generated to construct the training and validation sets. This stage requires modest domain knowledge in order to ensure a good representation of diversity in the training set.

3.4.3 Training model

The training procedure for all tasks is essentially the same and follows the well-established paradigm laid out in [24]. This strategy utilizes a stochastic gradient descent approach, with a fixed batch size, (a) a series of mean corrected image patches are introduced to

the network over a series of epochs, (b) an error derivative calculated, and (c) back-propagated through the network by updating the network weights. The learning rate is annealed over time so that a local minimum is reached. The resulting learned weights (i.e., the model) are stored to be used later at test time.

3.4.4 Testing model

By submitting image patches to the network, of the same size used during training, we obtain a class prediction from the learned model.

3.5 System design

In order to build a deep learning model for classifying NHL subtypes with CNN that can accurately classify the subtypes of NHL (CLL, FL, MCL), we propose a NHL-CNN framework presented, in Figure 3.5, which based on a tissue level classification strategy that is to directly learn the set of features representative of the tissue class via DL. In this case, the DL classifier could thus be trained to self-discover the nuanced disease patterns within each class.

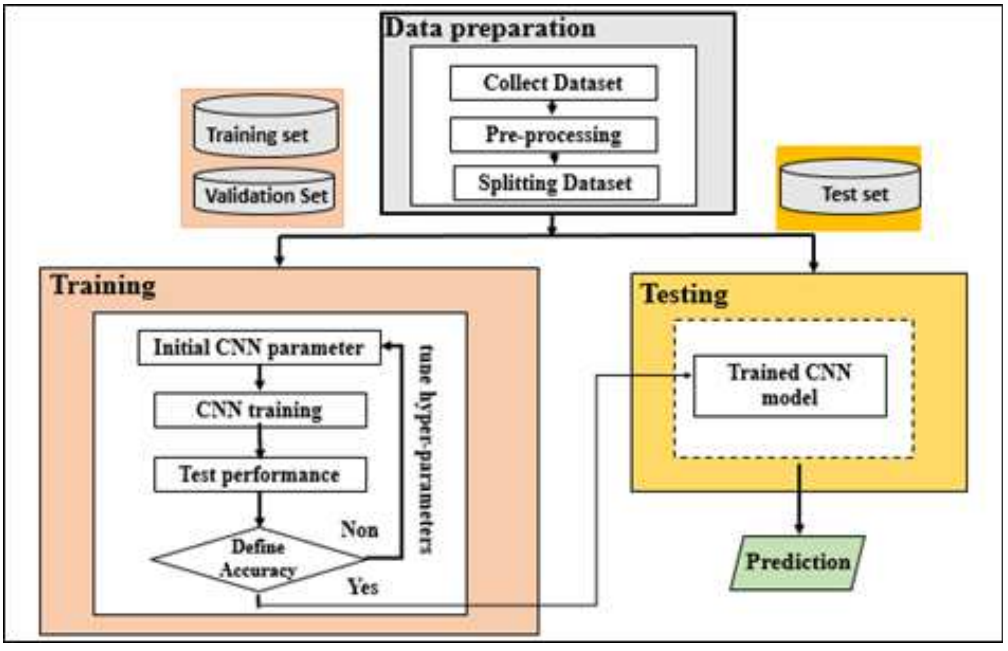


FIGURE 3.5: CNN framework for NHL classification model (NHL-CNN)

The NHL-CNN framework model contains three main components.

3.5.1 Data Preparation

Data Preparation is the first step has been applied before feeding the data to the CNN. It is divided to tree sub-components.

3.5.1.1 Collect data

As in numerous other fields which have adopted supervised deep learning techniques, collect data is the important step for provide training data for supervised multi-label learning of patch-level Histological Tissue Type (HTT) in a digitized whole slide image.

There exist many publicly available databases of labeled digital pathology images [75], which may be broadly classified into three types: government or academic repositories, educational sites, and grand challenges. Despite wider data publication in the recent years, much of the published work still employs proprietary WSI data set [4].

In our work, We obtained the Lymphoma sub-types WSI from the National Intelligence Authority (NIA) curated dataset [2]. This dataset was created to mirror real-world situations and samples prepared by different pathologists at different sites. Selected samples also contain a larger degree of staining variation than one would normally expect.

The dataset consist of 374 images of size 1388 x 1040. These are further broken down into 113 for the CLL class, 139 for the FL class and 122 for the MCL class. Unfortunately, there is no description with the data indicating if the prefix of the file name indicates a unique patient or a unique facility. They did indicate that the data has been curated from multiple sources to create a real-world type cohort, which contains typical stain and scanning variances. Figure 3.6 presents some examples of these images.

3.5.1.2 Preprocessing

Before providing the data to CNN, certain image processing techniques [11] can be applied in order to prepare the data and build a deep learning model.

In our work, the Preprocessing step consists en two operations: Normalization and Patch Extraction (see Figure 3.7).

[2] http://www.andrewjanowczyk.com/use-case-7-Lymphoma-sub-type-classification/

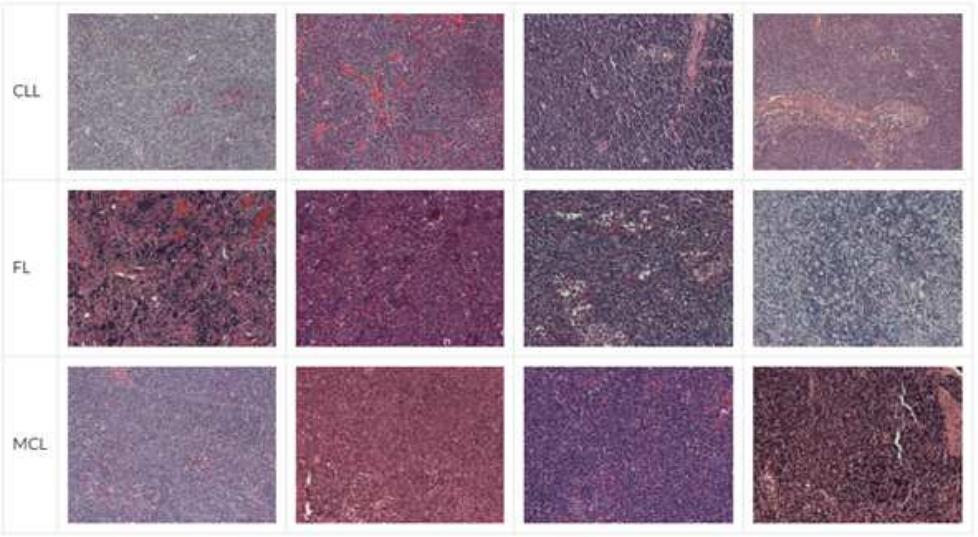


FIGURE 3.6: Examples of NHL images from NIA curated dataset

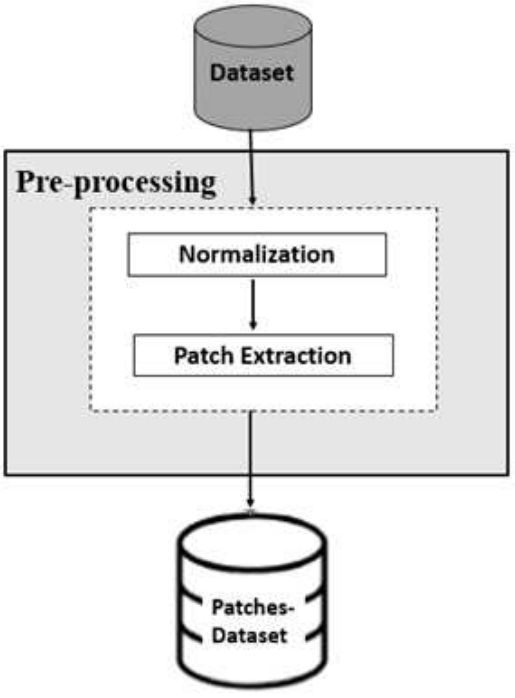


FIGURE 3.7: Preprocessing operations

a- Normalization operation is used in order to standardize the range of independent variables or features of data [11]. In addition, the best and easy method is Re-scaling (min-max normalization). It is also known as min-max scaling or min-max normalization, it consists in re-scaling the range of features to scale the range in [0, 1] or [-1, 1], according the following equation:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{3.1}$$

Where x is an original value, $x\prime$ is the normalized value. In our work, the Normalization consists in re-scaling the range of pixel values of each image in Dataset, to scale the range in $[0, 1]$.

b- Patch Extraction Most successful approaches to training deep learning models on WSIs, do not use the whole image as input, and instead extract and use only a small number of patches [73]. Image patches are usually square regions with dimensions ranging from 32×32 pixels up to 10000×10000 pixels with the majority of approaches using image patches of around 256×256 pixels [73]. In this operation, the extraction of the patches are done from all images, separated into the 3 subtypes, of normalized Dataset. We use overlapping patches to cover the full image. In this case, the original size of the image is 1338×1040 pixels; the images are down sampled to 1344×1024 to crop the overlapping and sequential patches of size: 64x64 overlapping by 4x4. Since each image is representative of the entire class, the extract patches, from an image, have the appropriate class of its root image and stored in Paches-Dataset. Table 3.1 shows the number of images and images patches for each class (CLL, FL, MCL) in Dataset and Paches-Dataset respectively. Some extract images patches, from an image of each class (CLL, FL, MCL) are presented in Figure 3.8.

	CLL	FL	MCL	Total samples
Dataset	113	139	122	374 images
Patches-Dataset	44183	54349	47702	146234 patches

TABLE 3.1: Total samples dataset and patches-dataset

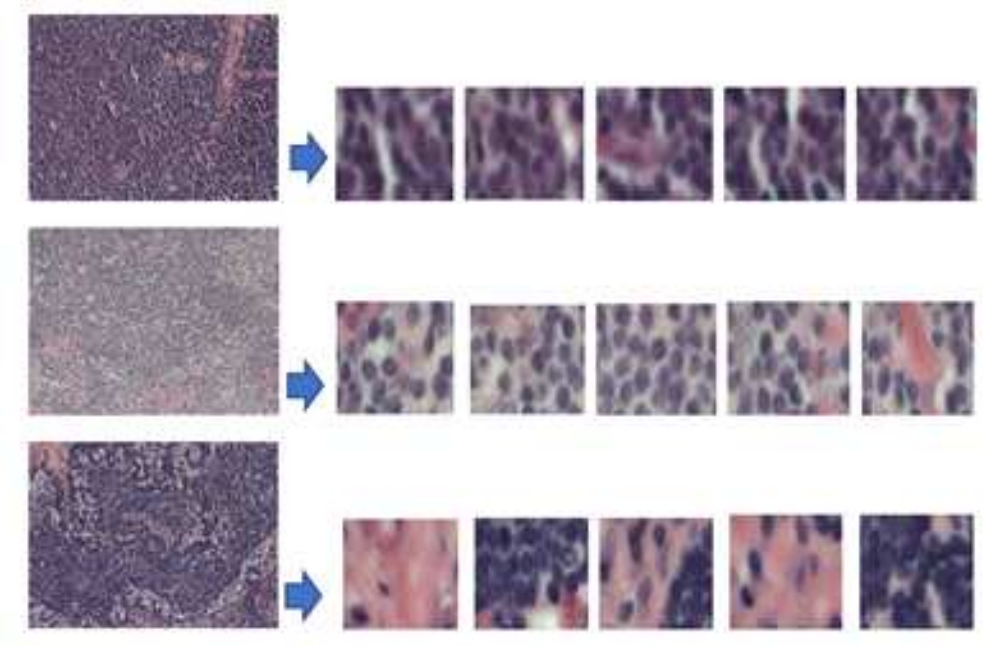


FIGURE 3.8: Some extract image patches, from an image of each class (CLL, FL, MCL)

3.5.1.3 Splitting Dataset

The most frequent splitting strategy of the Dataset, in deep learning, is to divide the dataset in training, validation, and test sets. The optimal ratio of samples distributed in each set varies for each problem [14]. However, as a rule of thumb, a split of 80% training, 10% validation, and 10% test division is commonly used.

For smaller datasets, the most commonly used sampling strategy is the k-fold cross-validation [76]. The dataset is divided equally in k folds. For each training, the algorithm is trained on almost all folds but tested on a single holdout fold of the data. The training is repeated k times using varying holdout folds. The final performance is the mean of the k measured performances (Figure 3.9).

Deep learning algorithms generally introduce two significant limitations to systematically use k-fold crossvalidation [14]. First, training deep learning algorithms on large datasets usually implies an intensive computational burden which prevents in practice a high number of training iterations with limited resources. Second, training of deep neural networks depends on many more hyperparameters than shallower machine learning algorithms.

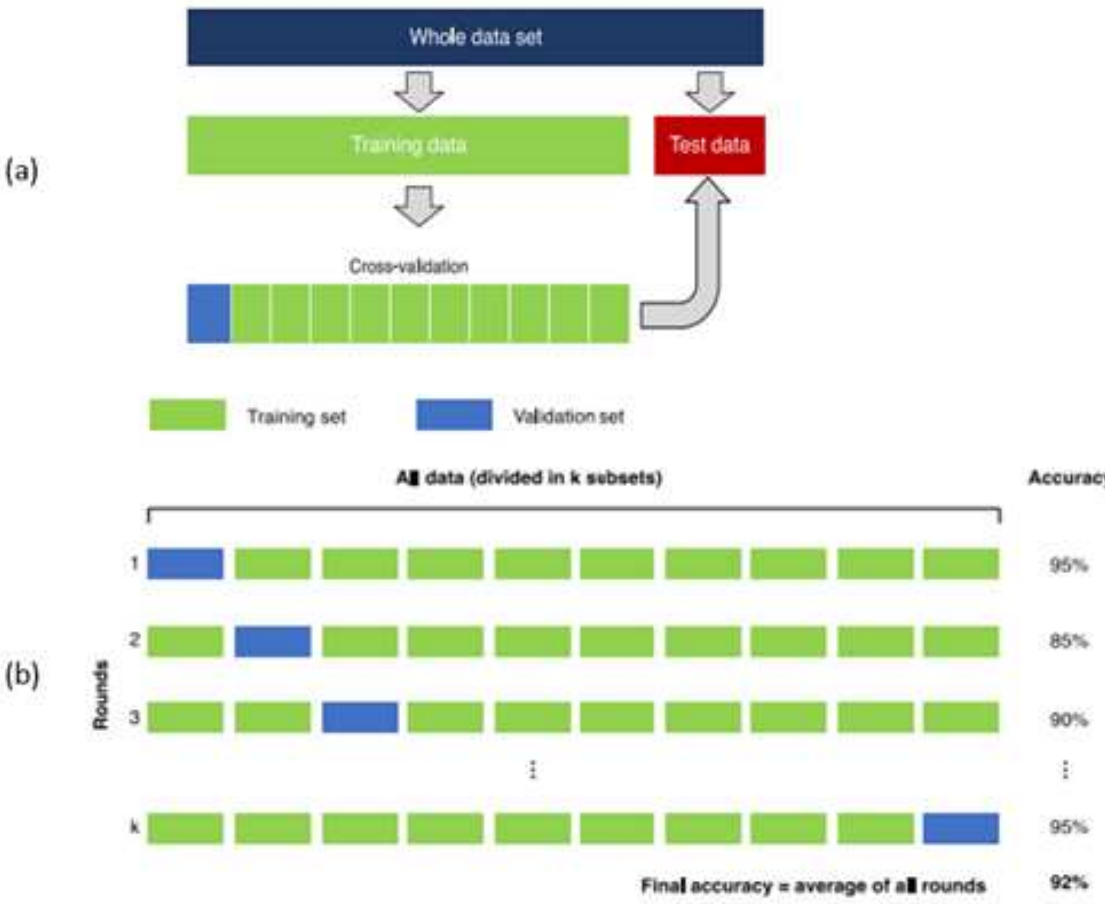


FIGURE 3.9: Data sampling strategies. (a): The whole dataset is split in two distinct subsets for training and testing purposes. Training dataset is subdivided to perform cross-validation, (b): k-fold cross-validation. The training dataset is divided in k subsets of equal size. Training is performed sequentially, considering at each iteration a specific subset as validation set [14]

In our work, we have a big Patches-Dataset; it contains 146234 samples (image patches). To construct the training, validation and test set, from this Patches-Dataset, we use a split of 70% training, 20% validation, and 10% test division, for the image patches of each class. (see Figure 3.10).

This division allows multiple trainings using the same training set to search for the optimal hyperparameters to maximize performance on the validation set. When the best performance is obtained on the validation set, the algorithm is ultimately used once on the test set to measure and confirm the final performance.

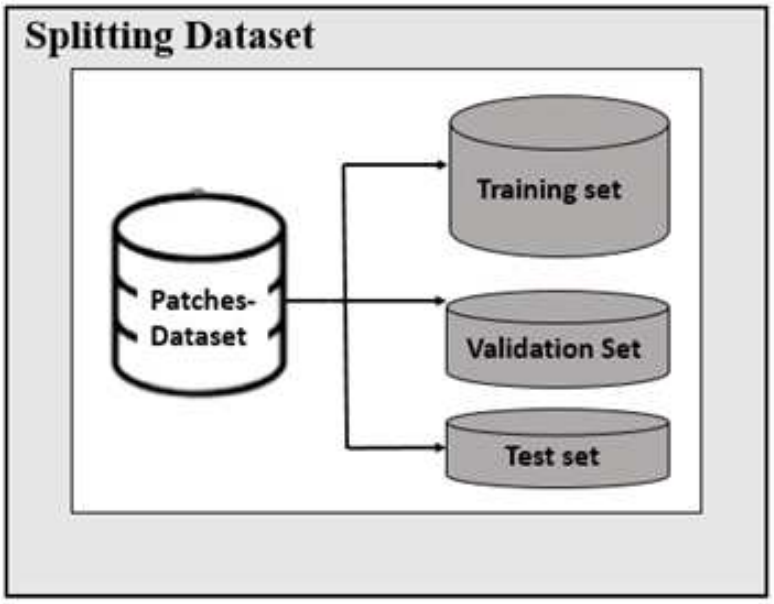


FIGURE 3.10: Splitting Patches-Dataset

3.5.2 Training

The goal of training component is to infer a function that can map the input patches to their appropriate labels (CLL, FL or MCL) well using Training set. The learning process is based on convolutional neural network which is a deep neural network of supervised learning, and its structure includes convolution layers, pooling layers, and fully connected layers, among which the convolution layer and pooling layer are core parts to realize feature extraction of CNN.

The structure is first arranged alternately by convolution layer and pooling layer to achieve the extraction and mapping of local features from the Training set, then successively arranged by several fully connected layers. The last layer of the network classifies those feature maps with a soft-max method and gives as output scores corresponding to the " probability" that each patch belongs to each of the learned classes.

The initialization of CNN parameters has a big impact on the overall prediction accuracy. We empirically initialized the hyper-parameters and fine-tuned them during the training process (Figure 3.5). The details of the hyperparameters used in this study are shown in Table 3.2.

	Parameters	Hyperparameters
Convolutional layer	Kernels	Kernels size, number of kernels, stride, padding, activation function
Pooling layer	None	Pooling methods, filter size, stride, padding
Fully connected layer	weights	Number of weights, activation function
Others		Learning rate, loss function, batch size, epochs, weights initialization, Dataset splitting

TABLE 3.2: The used parameters and hyperparameters in convolution neural network

The network is trained for Nb epochs with criteria to decrease learning rate if validation metrics, concerning the identification accuracy of the CNN model, did not improve after Nb iterations. Model parameters are saved as a "snapshot" after each reduction of validation error. The snapshot that performed best on the validation set was used on test set.

In this context, a training set is used to train a network, where loss values are calculated via forward propagation and learnable parameters are updated via backpropagation.

A validation set is used to test the intermediately trained CNN model (Figure 3.5). The accuracy or error of the CNN learning output from the validation set was used as a feedback parameter to tune the hyper-parameters,including the number of feature maps, kernel size of the convolutional layers, and filter size of the pooling layers in the network.

Throughout the training-validation cycles, we tuned the hyper-parameters and updated the training dataset until the accuracy of the CNN learning reached a satisfactory level.

3.5.3 Testing

A test set is ideally used only once at the very end of the project in order to evaluate the performance of the final model that is fine-tuned and selected on the training process with training and validation sets.

Ensemble predictions on test set were analyzed for true positives (TP), true negatives (TN), false positives (FP), and false negatives (FP). These parameters were used to calculate a variety of performance metrics, including including precision (PR), recall (RE), accuracy (ACC), F1-score ($F1$), according to the following equations:

$$PR = \frac{TP}{TP + FP} \tag{3.2}$$

$$RE = \frac{TP}{TP + FN} \tag{3.3}$$

$$ACC = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.4}$$

$$F1 = \frac{2TP}{2TP + FP + FN} \tag{3.5}$$

3.6 Conclusion

In this chapter, we have presented Lymphoma disease which is one of the most common cancer whether in less developed or developing countries and the classification of its subtypes (CLL, FL and MCL) on whole-slide images represent a major challenge for pathologists. Our proposed methodology to design a NHL classification model, by using Deep Learning with a convolutional neural network (NHL-CNN), is well detailed.

The environments and developing tools used in the realization of the proposed methodology will be described in the following chapter.

Chapter 4

DETAIL DESIGN AND DEVELOPMENT OF NHL CLASSIFICATION MODEL

4.1 Introduction

In this chapter, we present the development of detailed design that brings our methodology to design a deep learning system for classifying lymphoma subtypes with CNN (NHL-CNN), proposed in chapter 3, into a completed form of specifications. This stage includes:

- Defining how the information system should be built (environments and developing tools)
- Ensuring that the information system is operational and used,
- Ensuring that the information system meets quality standard (i.e., quality assurance).

4.2 Environments and developing tools

For the development of our system, different environments and tools are used,to build the front-end and back-end of system (see Table 4.1 and Table 4.2). The front-end refers

to the user interface; while "back-end" refers to parts of a computer application or a program's code that allow it to operate.

Tools Logo	Descriptions
	Brackets is a source code editor with a primary focus on web development. Created by Adobe Systems, it is free and open-source software licensed under the MIT License, and is currently maintained on GitHub by Adobe and other open-sourced developers. It is written in JavaScript, HTML and CSS.
	is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries.
	Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.
	Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.
	JavaScript often abbreviated as JS, is a high-level, interpreted programming language that conforms to the ECMAScript specification. JavaScript has curly-bracket syntax, dynamic typing,prototype-based object-orientation, and first-class functions.
	jQuery is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation,as well as event handling, CSS animation, and Ajax. It is free, open-source software using the permissive MIT License. As of May 2019, jQuery is used by 73% of the 10 millionmost popular websites.
	is a software application for accessing information on theWorld WideWeb. A web browser is a software application for accessing information on theWorld WideWeb. Each individual web page, image, and video is identified by a distinct Uniform Resource Locator, enabling browsers to retrieve these resources from a web server and display them on a user's device.

TABLE 4.1: Developing tools for front-end

Tools Logo	Descriptions
	Python is a programming language created by Guido van Rossum and first released in 1991. Python is successfully used in thousands of real-world business applications around the world e.g., Google and YouTube.The primarily rationale for adopting Python for ML and DL is because it is a general-purpose programming language for research, development and production, at small and large scales. Python features a dynamic type system and automatic memory management, with a large and comprehensive libraries for scientific computation and data analysis.
	PyCharm is an integrated development environment used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains.
	ANACONDA is a free and open-source distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. Package versions are managed by the package management system conda.
	TensorFlow is an open-source software library for numerical computation using data flow graphs. TensorFlow was created and is maintained by the Google Brain team within Google's Machine Intelligence research organization for ML and DL. It is currently released under the Apache 2.0 open-source license.
	Keras is Python wrapper library that provides bindings to other DL tools such as TensorFlow, CNTK, Theano, beta version with MXNet and announced Deeplearning4j. Keras runs on Python 2.7 to 3.6 and can seamlessly execute on GPUs and CPUs given the underlying frameworks.
	CUDA is a parallel computing platform and application programming interface model created by Nvidia. It allows software developers and software engineers to use a CUDA-enabled graphics processing unit for general purpose processing-an approach termed GPGPU.
	OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez. The library is cross-platform and free for use under the open-source BSD license.
	NumPy is the fundamental package for scientific computing with Python. Besides its obvious scientific uses, NumPy can also be used as an efficient multidimensional container of generic data.
	Pandas is a Python package providing fast, flexible, and expressive data structures designed to make it easier to work with relational or labelled data.
	matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+.

TABLE 4.2: Developing tools for Back-end

4.3 Developing the back-end

In order to implement our deep learning NHL-CNN model, we have used Python language with some well-known API's and libraries working on the deep learning (CNN). One of the most powerful and easy-to-use Python libraries for developing and evaluating deep learning models is Keras; It wraps the efficient numerical computation librarie TensorFlow.

The following steps present our implementation of deep learning NHL-CNN model with Python and Keras.

4.3.1 Data Preparation

After searching and downloading the NHL subtypes dataset, firstly, we have applied normalization and patch-extraction operations on each image of dataset to obtain Patches-dataset which is split in three distinct sets: training, validation and test, using a split of 70% training, 20% validation and 10% test. The ratio of samples distributed in each set is determined by experiments.

We have applied *split_folders.ratio*, as Python module, such as it is presented in Listing 4.1 and described in the Table 4.3.

```
1 import split_folders
2 split_folders.ratio(' C:/Users/HP/Desktop/patches_dataset ', output="C
    :/Users/HP/Desktop/final_dataset", seed=1337, ratio=(.7, .2, .1))
```

Listing 4.1: Split patches_dataset with a ratio.

Function	Arguments
Split_folder.ratio()	- input folder; path to patches_dataset - seed; set seed value for shuffling the items; defaults to 1337. - ratio: division percentage (.7,.2,.1) - Output folder; path to final_dataset

Table 4.3: Split with a ratio

split_folders.ratio splits, with a ratio, the input folder '*patches_dataset*' where we have stored all of image patch data (*Patches-dataset*)into: train '*train*', validation '*val*' and test '*test*' directories contained in output folder '*final_dataset*', under each of the dataset

directories, we have subdirectories, one for each class (CLL, FL, MCL) where the actual image files (patch files) will be placed, as shown in the Figure 4.1.

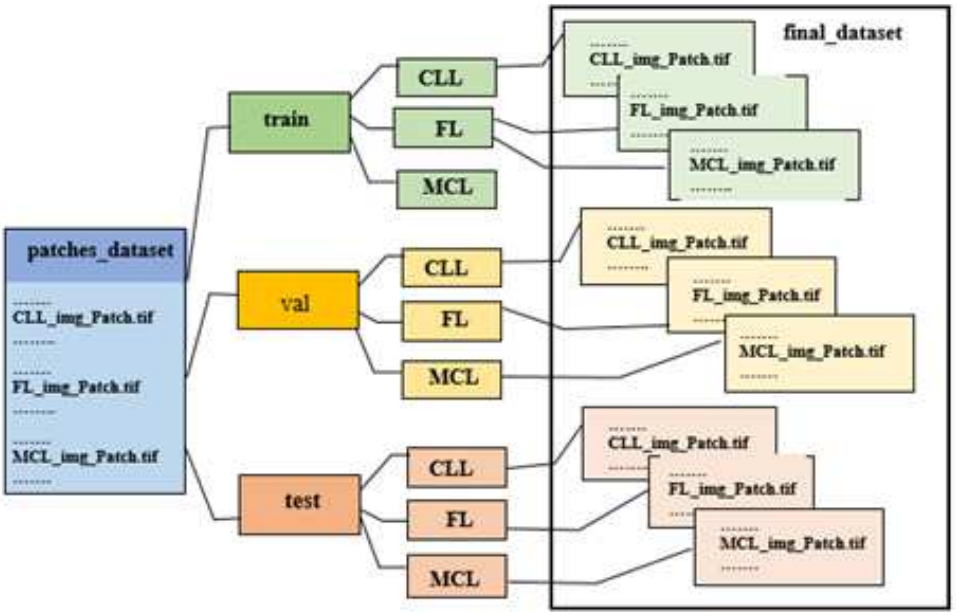


FIGURE 4.1: Data folder structures

4.3.2 Building NHL-CNN

4.3.2.1 Import libraries and modules

For building deep neural networks with Keras, we first import the various libraries and modules as they are presented in Listing 4.2 and described in the following Table 4.4.

```
1  from keras.preprocessing.image import ImageDataGenerator
2  from keras.models import Sequential
3  from keras.layers import Dropout, Flatten, Dense, Activation
4  from keras.layers.convolutional import Conv2D, MaxPooling2D,
     ZeroPadding2D
5  from keras.optimizers import Adam
6  from keras.layers.normalization import BatchNormalization
7  from keras.regularizers import l2
8  from keras.callbacks import TensorBoard
9  import time
10 from keras.callbacks import ModelCheckpoint
```

Listing 4.2: Import libraries code

Libraries/modules		Descriptions
Numpy		Is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.
ImageDataGenrator		Is used to load our train, validation datasets into memory and generate batches of tensor . image data with real-time data augmentation
Sequential		For creating deep learning models where an instance of the Sequential class is created and model layers are created and added to it. A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.
Keras core layers	Dense	Is used to instantiate a dense layer.
	Dropout	Applies dropout to the input. Dropout consist of randomly setting a fraction rate of input units to 0 at each update during training time, which helps prevent overfitting
	Activation	Is used to add an activation function . to the sequence of layers
	Flatten	Converts the pooled feature map to a single column that is passed to the fully connected layer.
Keras conventional layers	Conv2D	Creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs
	Maxpoling2D	Downsamples the input representation by taking the maximum value over the window defined by pool_size for each dimension along the features axis.
	ZeroPadding2D	Can add rows and columns of zeros at the top, bottom, left and right side of an image tensor
Adam		Optimizer that implements the Adam algorithm. Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments
BatchNormalization()		Normalize and scale inputs or activations.
kernel_regularizer		Create a regularizer that applies an L2 regularization penalty.
TensorBoard		Is a visualization tool provided with TensorFlow
ModelCheckpoint		Callback to save the Keras model or model weights at some frequency

TABLE 4.4: Description of the used libraries and modules

4.3.2.2 NHL-CNN parameter initialization

Before we can begin to train the network, we have to initialize its parameters. The values of the used parameters are determined basing on previously suggested values that

can be found in the literature and by experiments. The details of the hyper-parameters, used in this study, are described in Table 4.5 and shown in Listing 4.3.

Hyper-parameters	Descriptions	Values
img_width, img_heighy	Size of input patch (image)	64x64
batch size	is a hyperparameter that defines the number of samples to work through before updating the internal model parameters	64
Steps epoch	The number of batch iterations before a training epoch is considered finished	1600
Steps_validation	Similar to steps epoch but on the validation data set instead on the training data.	457
Epochs	The number of epochs is the number of complete passes. through the training dataset The number of epochs is a hyperparameter that defines the number times that the learning algorithm will work through the entire training dataset	40
Class_num	The number of classes; the simples are assigned one of more than three classes (CLL, FL or MCL).	3
L2_reg	The value of regularization parameter	0

TABLE 4.5: Hyper-parameter description

```
1 img_width, img_height = 64, 64
2 batch_size = 64
3 steps_epoch = 1600
4 steps_validation = 572
5 epochs = 40
6 classes_num = 3
7 l2_reg=0.
```

LISTING 4.3: Hyper-parameter initialization

4.3.2.3 Loading the datasets

We have used *ImageDataGenrator* class to load our train, validation datasets into memory and Generate batches of tensor image data with real-time data augmentation. The data is looped over (in batches). In this step, the *target_size*, *batch_size*, *normalization_data* and *class_mode* are defined. Next, an iterator is required to progressively load

images for a single dataset. This requires calling the *flow_from_directory()* function and specifying the dataset directory, such as the *train*, *test*, or *validation* directory. More details can be found in Table 4.6 and Listing 4.4.

```
train_data_path = 'C:/Users/HP/Desktop/final_dataset/train' #train_data
   path
validation_data_path = 'C:/Users/HP/Desktop/final_dataset/val' #
  val_data path

train_datagen = ImageDataGenerator(rescale=1. / 255) # rescale train
  data
test_datagen = ImageDataGenerator(rescale=1. / 255) # rescale valid
  data

train_generator = train_datagen.flow_from_directory(
train_data_path, #train_data path
target_size=(img_height, img_width), #target_size
batch_size=batch_size, #batch_size
class_mode='categorical') #class_mode

validation_generator = test_datagen.flow_from_directory(
validation_data_path, #val_data path
target_size=(img_height, img_width), #target_size
batch_size=batch_size, #batch_size
class_mode='categorical') #class_mode
```

LISTING 4.4: Loading datasets

4.3.2.4 Creating the model

After loading the datasets and initializing the hyper-parameters, we have use Tensorflow to design our architecture, according to Listing 4.5, for NHL-CNN, which was determined after trying many CNN configurations. The Table 4.7 describes the layers of the NHL-CNN model used.

```
model = Sequential()
```

Function	Arguments
ImageDataGenerator	- Rescale: we multiply the data by the value provided.
flow_from_directory	- directory: string, path to the target directory, must be set to the path where our train, validation folders are present. - target_size: the size of our input images, every image will be resized to this size (64,64) - batch_size: Size of the batches of data, in our case 64 No. of images to be yielded from the generator per batch (64) - class_mode: because we have three classes in our dataset this is why we set to"categorical".

TABLE 4.6: Function descriptions for loading datasets

```
# Layer 1
model.add(Conv2D(64, (11, 11), input_shape=(img_width, img_height,3),
  kernel_regularizer=l2(l2_reg)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(3, 3)))
# Layer 2
model.add(Conv2D(128, (5, 5)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
# Layer 3
model.add(ZeroPadding2D((1, 1)))
model.add(Conv2D(256, (3, 3)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
# Layer 4
model.add(Flatten())
model.add(Dense(1000))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))
# Layer 5
model.add(Dense(500))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))
# Layer 6
model.add(Dense(classes_num))
```

```
31  model.add(BatchNormalization())
32  model.add(Activation('softmax'))
```

Listing 4.5: Creating NHL-CNN model

Layers	Composed of
1	Conv2D(64,(11,11), kernel_regularizer, batchNormalization, activation(ReLU), MaxPooling2d(3,3)
2	Conv2D(128,(5,5)),batchNormalization, activation(ReLU), MaxPooling2d(2,2)
3	Zeropadding2D(1,1),Conv2D(256,(3,3)),batchNormalization, activation(ReLU),MaxPooling2d(3,3)
4	Flatten(),Dense(1000),batchNormalization, activation(ReLU),Dropout(0.25)
5	Dense(500),batchNormalization, activation(ReLU),Dropout(0.25)
6	Dense(3),batchNormalization, activation(softmax)

Table 4.7: Layer descriptions of NHL-CNN model

The model type that we have used is Sequential. Sequential is the easiest way to build a model in Keras. It allows building a model layer by layer. We have applied the '*add()*' function to add layers to our model (see Listing 4.5). We have *classes_num* nodes (*classes_num*=3) in our output layer, one for each possible outcome (0-2). The activation is '*softmax*'. Softmax makes the output sum up to 1 so the output can be interpreted as probabilities. The model can then make its prediction based on which option has the highest probability.

4.3.2.5 Compiling the model

Once the model is created, we can config the model with *model.compile()* method , which takes three arguments: *optimizer*, *loss* and *metrics* (see Table 4.8). The optimizer controls the learning rate. We have used:

- *'adam'* as our optimizer, witch adjusts the learning rate throughout training. The learning rate 'lr' determines how fast the optimal weights for the model are calculated.
- *'categorical_crossentropy'* for our loss function. It measures the performance of a classification model whose output is a probability value between 0 and 1.
- *'accuracy'* metric to see the accuracy score on the validation set when we train the model. It is a function that is used to judge the performance of your model.

Function	Arguments
Compile()	- Loss='categorical_crossentropy' - Optimizer='adam': The adam optimizer adjusts the learning rate throughout training. - learning_rate: our case we use lr=0.01 - Metrics=['accuracy']

TABLE 4.8: Function description for the model compiling

4.3.2.6 Training the model

To train our model, we have used the *'fit_generator()'* function. This function first accepts a batch of the dataset, then performs backpropagation on it, and then updates the weights in our model. This process is repeated until we have reached the desired number of epochs. *'ModelCheckpoint'* callback is used in conjunction with training using *'model.fit_generator'* to save a model or weights (in a checkpoint file) when there is an improvement in classification accuracy on the validation dataset (monitor='val_acc' and mode='max'). The argument descriptions of these functions are shown in Table 4.9.

Functions	Arguments
modelCheckPoints()	Inputs: - Filepath: the file path you want to save your model in string. - Monitor: quantity to monitor; we used val_acc - Verbose: verbosity mode, 0 or 1. - Mode: one of {auto, min, max}, we used max because the monitored value is val_acc
Fit_generator()	- Training and validation sets. - Epochs a number of epochs, we want to train our model. In our case is 40. - Batch_size; 64 per epochs - Callbacks; a list of callback functions applied during the training of our model. - Steps_per_epochs; specifies the total number of steps taken from the generator. We can calculate the value of steps_per_epoch as the total number of samples in your dataset divided by the batch size.

TABLE 4.9: Function descriptions for the model training

The complete code for compiling and training the model is described in following Listing 4.6.

```
filepath="./weights-{epoch:02d}-{val_acc:.3f}.hdf5"

checkpoint = ModelCheckpoint(filepath, monitor='val_acc', verbose=1,
  mode='max')

callbacks_list = [checkpoint]

model.compile(loss='categorical_crossentropy',optimizer=Adam(lr=0.01),
  metrics=['accuracy'])

model.fit_generator(
train_generator,
steps_per_epoch=steps_epoch,
epochs=epochs,
validation_data=validation_generator,
validation_steps=steps_validation,
callbacks=callbacks_list)
```

LISTING 4.6: Model compiling and training

4.3.2.7 Visualizing loss and accuracy during training

Keras provides the capability to register callbacks when training a deep learning model. TensorBoard is a visualization tool provided with TensorFlow. The callback logs events for TensorBoard, including: metrics summary plots,training graph visualization.

In this case, the path of the directory where to save the log files to be parsed by TensorBoard would be specified. The listing 4.7 shows the visualizing loss and accuracy during training.

```
name = "lymphoma-{}".format(int(time.time()))

tensorbord = TensorBoard(log_dir='logs/{}'.format(name))

model.fit_generator(
train_generator,
steps_per_epoch=steps_epoch,
epochs=epochs,
```

```
validation_data=validation_generator,
validation_steps=steps_validation,
callbacks=tensorbord)
```

LISTING 4.7: Visualizing training/validation accuracy and loss

4.3.2.8 Predictions on the trained model

Predictions on trained model aim firstly to estimate the generalization accuracy of a model on test data, and secondly to make a prediction on an entire image.

a- Estimate the generalization accuracy of a model on test data

The model evaluation on test data *'test'*, is done as follow:

1. As in the training, we use *ImageDataGenrator* and flow_from_directory() function to pass our test set 'set' easily by just pointing at a directory.
2. We use *load_model* to load trained model (the best saved model) and saved it in the hdf5 format.
3. we use the *predict_generator* to generate predictions 'prediction' from the input samples of the data generator
4. The *confusion_matrix* is used to classifier performance.

The details of used functions are presented in Table 4.10 and the code corresponding to the evaluation of the model on test data, is described in Listing 4.8

Function	Arguments
Load_model()	- Model path
Predict_generator()	- Test set - Steps=epochs
confusion_matrix	- Labels ; CLL: 001 , FL:010 , MCL:001 - prediction

TABLE 4.10: Function descriptions for evaluated model

```
import numpy as np
from keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import itertools
```

```
    import keras
    batch_size = 1000

    data_path = 'C:/Users/HP/Desktop/final_dataset/test'
    test_datagen = ImageDataGenerator(rescale=1. / 255)
    test_generator = test_datagen.flow_from_directory(
    data_path,
    target_size=(64, 64),
    batch_size=batch_size,
    class_mode='categorical')

    model = keras.models.load_model("weights-02-0.964.hdf5")
    prediction = model.predict_generator(test_generator, steps=4,
    verbose=0)
    print(prediction)

    imgs, labels = next(test_generator)
    prediction = np.argmax(prediction, axis=1)
    labels = np.argmax(labels, axis=1)
    cm = confusion_matrix(labels, prediction)
    classee = ['CLL', 'FL', 'MCL']
    plot_confusion_matrix(cm, classee, title='confusion_matrix')
```

LISTING 4.8: Predictions on the trained mode

b- Make a prediction on an entire image.

To make prediction on an entire image, we have loaded the best saved model and pre-process the image for dividing it into small patches (64X64) overlapping. Then the predictions are made for 100 patches, from this image, randomly selected. The predicted class value for the entire image is the most predicted of its selected patches. The following Listing (4.9) describes how to make prediction on an entire image. Predictions on trained model aim firstly to estimate the generalization accuracy of a model on test data, and secondly to make a prediction on an entire image.

```
from random import randrange,randint
import keras
```

```
import tensorflow as tf
import numpy as np
from keras.preprocessing.image import ImageDataGenerator

CATEGORIES = ['CLL', 'FL', 'MCL']

def prepare(filepath):
image = tf.keras.preprocessing.image.load_img(filepath)
input_arr = keras.preprocessing.image.img_to_array(image)
x = randrange(1300)
y = randrange(950)
image = input_arr[y:y+64,x:x+64]
input_ar = image/225
input_ar = np.array([input_ar])

return input_ar


model = keras.models.load_model("weights-02-0.964.hdf5")
i=0
j=0
s=0

for x in range(50):
pre = CATEGORIES[int(model.predict_classes([prepare('C:/Users/HP/
  Desktop/lymphoma/MCL/sj-04-3077-R2_006.tif')]))]
print(pre)
if pre == 'CLL':
i = i + 1
if pre == 'FL':
j = j + 1
if pre == 'MCL':
s = s + 1

print('CLL :',i)
print('FL :',j)
print('MCL :',s)
```

Listing 4.9: Prediction on an entire image

4.4 Developing the front-end

In the front-end part of the system, we create a simple design that can be easily used by any user (with different gender and ages), with some simple colors, and structure. The figure 4.2 shows the architecture of the front-end pages.

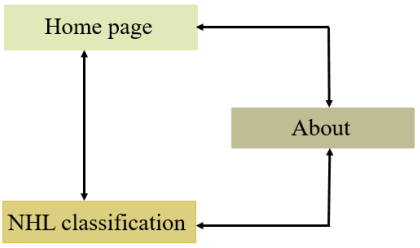


FIGURE 4.2: Architecture of the front-end pages

To create our web site, we have used html to build the structure of the page, CSS give us big advantage on designing and styling our page for better looking and well organizing. Moreover, we have used the JavaScript to make our pages moving and working giving some benefits on animations, button functions, pages function, uploading, printing, etc..., and finally, we used flask to relate our web pages with our code python. The figures below shows the content of some created web page.

- The figure 4.3 represents the home page (the main web page) which generally describes our project and facilitates navigation to other pages of the site.

FIGURE 4.3: Home page

- The page concerning the NHL classification, by selecting the entire image or the image patch, is shown in the figure 4.4.

FIGURE 4.4: NHL classification page

4.5 Empirical evaluation and results

All design and training of our deep learning CNN for classifying the NHL subtypes 'NHL-CNN' are performed on a 64-bit Windows machine with Intel Core i5 CPU (@2.20 GHz) and 12GB of RAM. For taking the benefit of the paralleling computing for fast training gaining time and resources, the local GPU on desktop machine, Nvidia GeForce820M, is used.

4.5.1 NHL-CNN architecture

When designing and configuring our deep learning model (NHL-CNN), we have attempted various configurations for the people's networks and using heuristics. The proposed architecture for NHL-CNN, is an adaptation of the AlexNet architecture. It summarized in Figure 4.5 and figure 4.6.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 54, 54, 64)	23296
batch_normalization_1 (Batch	(None, 54, 54, 64)	256
activation_1 (Activation)	(None, 54, 54, 64)	0
max_pooling2d_1 (MaxPooling2	(None, 18, 18, 64)	0
conv2d_2 (Conv2D)	(None, 14, 14, 128)	204928
batch_normalization_2 (Batch	(None, 14, 14, 128)	512
activation_2 (Activation)	(None, 14, 14, 128)	0
max_pooling2d_2 (MaxPooling2	(None, 7, 7, 128)	0
zero_padding2d_1 (ZeroPaddin	(None, 9, 9, 128)	0
conv2d_3 (Conv2D)	(None, 7, 7, 256)	295168
batch_normalization_3 (Batch	(None, 7, 7, 256)	1024
activation_3 (Activation)	(None, 7, 7, 256)	0
max_pooling2d_3 (MaxPooling2	(None, 3, 3, 256)	0
flatten_1 (Flatten)	(None, 2304)	0
dense_1 (Dense)	(None, 1000)	2305000
batch_normalization_4 (Batch	(None, 1000)	4000
activation_4 (Activation)	(None, 1000)	0
dropout_1 (Dropout)	(None, 1000)	0
dense_2 (Dense)	(None, 500)	500500
batch_normalization_5 (Batch	(None, 500)	2000
activation_5 (Activation)	(None, 500)	0
dropout_2 (Dropout)	(None, 500)	0
dense_3 (Dense)	(None, 3)	1503
batch_normalization_6 (Batch	(None, 3)	12
activation_6 (Activation)	(None, 3)	0

Total params: 3,338,199
Trainable params: 3,334,297
Non-trainable params: 3,902

FIGURE 4.5: NHL-CNN configuration

4.5.1.1 Tuning the batch size

In this first experiment, we have look at tuning the batch size on 20 epochs when training the deep learning NHL-CNN, to show its sensitivity for this parameter.

The batch size in iterative gradient descent is the number of patterns shown to the network before the weights are updated. It is also an optimization in the training of the network, defining how many patterns to read at a time and keep in memory. The number of epochs is the number of times that the entire training dataset is shown to the network during training.

To analyze the comportment of the deep learning NHL-CNN with the set of hyper-parameters for the CNN layers presented in table, concerning the batch size parameter,

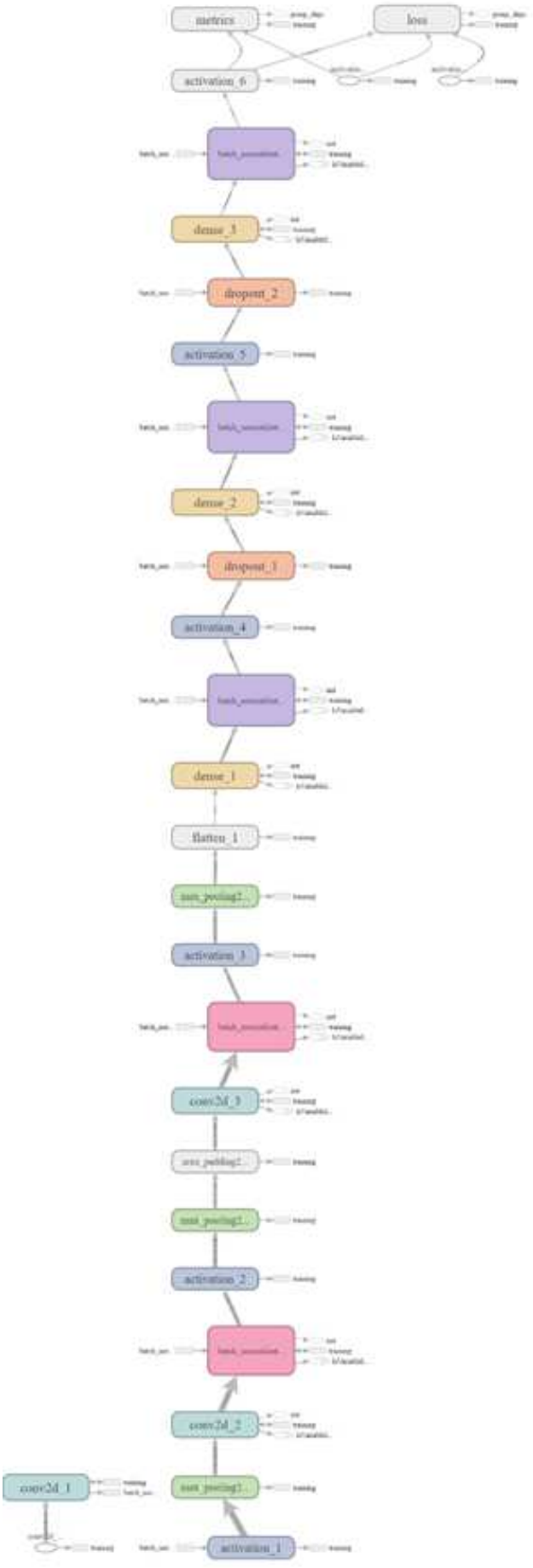


FIGURE 4.6: NHL-CNN architecture

we have evaluated its performance, using the training/validation accuracy (acc and Val_acc respectively) and the training/validation loss (loss and Val_loss respectively) at batch sizes : 1024, 512 and 64. The results for the performance are shown on plots: Figure 4.7, Figure 4.8, Figure 4.9 and Figure 4.10.

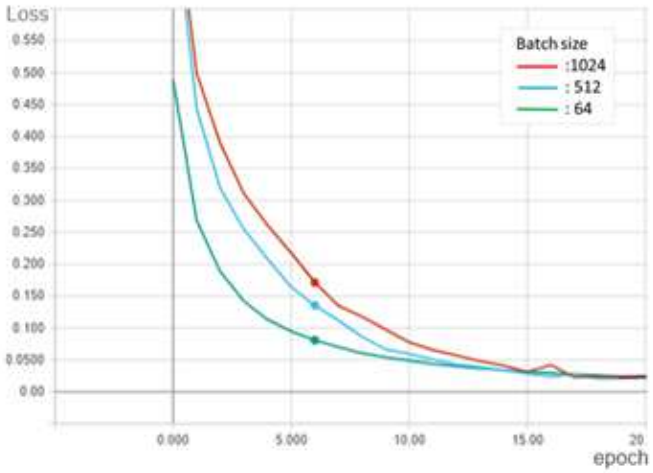


FIGURE 4.7: The loss on the training dataset over training epochs with batch sizes: 1024, 512 and 64

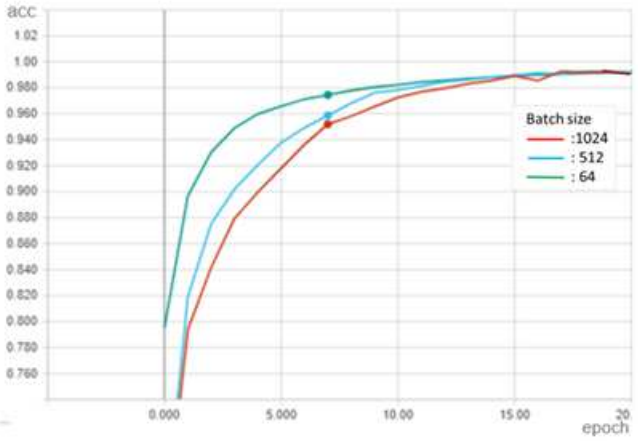


FIGURE 4.8: The accuracy on the training dataset over training epochs with batch sizes: 1024, 512 and 64

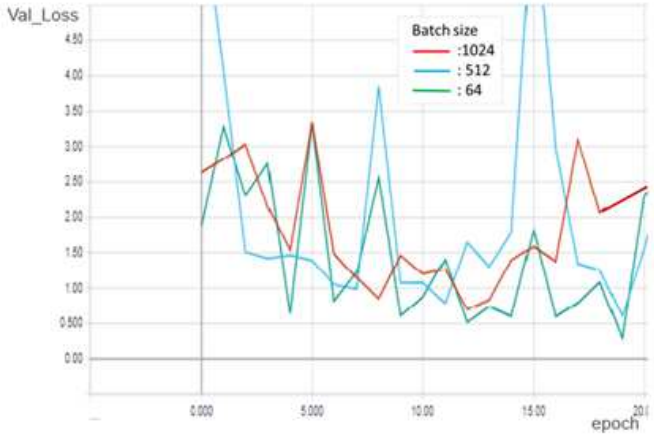


FIGURE 4.9: The loss on the validation dataset over training epochs with batch sizes: 1024, 512 and 64

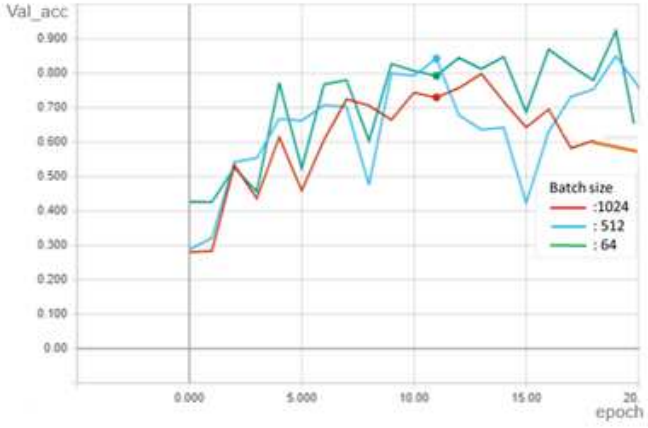


FIGURE 4.10: The accuracy on the validation dataset over training epochs with batch sizes: 1024, 512 and 64

From these plots, we can see that:

- Decreasing the training patch-size leads to better performance.
- The model could probably be trained a little more, using batch size = 64 (see green curve), as the trend for accuracy and loss on both training and validation sets.

4.5.2 Visualize the performance of NHL-CNN

To visualize the performance of our deep learning NHL-CNN over time during training, we have created:

- A plot of accuracy on the training dataset *'acc'* over training epochs.
- A plot of accuracy on the validation dataset *'Val_acc'* over training epochs.
- A plot of loss on the training dataset *'Val_loss'* over training epochs.
- A plot of loss on the validation dataset *'loss'* over training epochs.

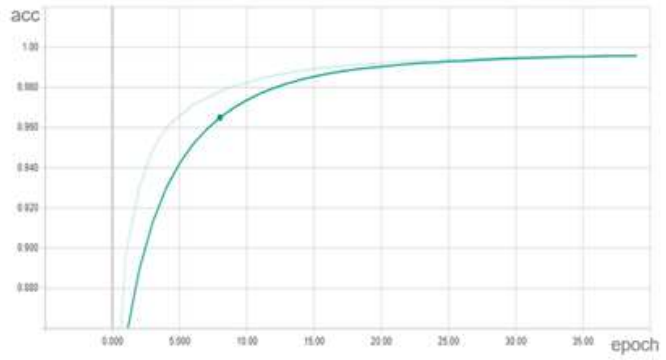


FIGURE 4.11: The accuracy on the training dataset over training epochs

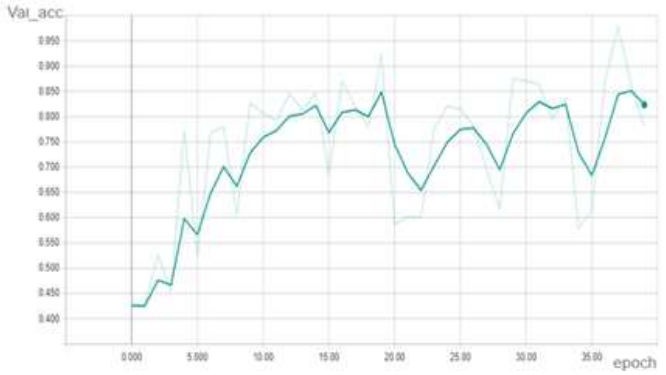


FIGURE 4.12: The accuracy on the validation dataset over training epochs

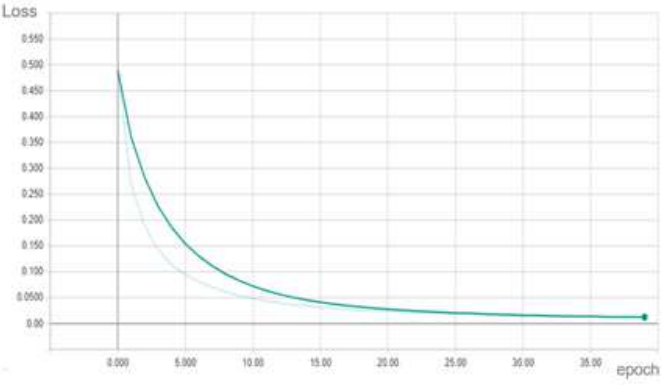


FIGURE 4.13: The loss on the training dataset over training epochs

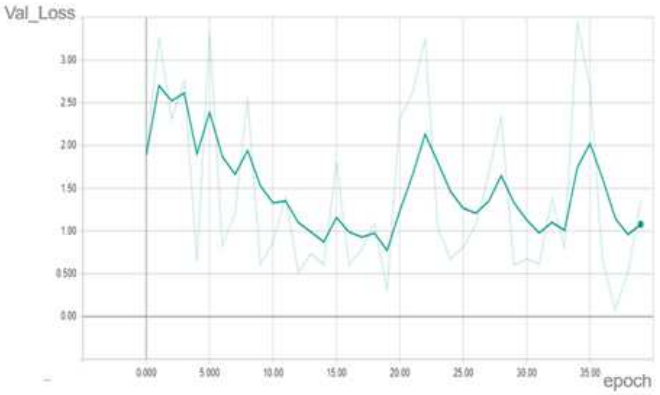


FIGURE 4.14: The loss on the validation dataset over training epochs

From these learning curves, we can see that:

- The plots of learning curves (Plot 4.11. and Plot 4.13) show a good fit because:
  - The plot of training accuracy increases to a point of stability.
  - The plot of training loss decreases to a point of stability.
- The plots of learning curves for validation accuracy and loss (Figure 4.12 and Figure 4.14) show noisy movements. This may occur if the validation dataset has too few examples as compared to the training dataset.
- In deep learning, the checkpoint is the stored weights of the model when there is an improvement in classification accuracy on the validation dataset, these weights can be used to make predictions as is, or used as the basis for ongoing training. In our case, the best value of training accuracy (accuracy) is in epoch number 37 (see plots Figure 4.7, Figure 4.8, Figure 4.9 and Figure 4.10). In this epoch, the values of:
  - Training loss 'loss' is 0.01%
  - Training accuracy 'accuracy' is 99.64%
  - Validation loss 'Val_loss' is 0.2%
  - Validation accuracy 'Val_accuracy' is 96.4 %

4.5.3 Evaluate model on test data

The evaluation of the NHL-CNN on test data '*test*', aims to estimate its generalization accuracy. A confusion matrix is a performance measurement technique used to assess the performance of the model, built in the training phase, on test set, by comparing the actual and predicted classes. It provides a more detailed breakdown of correct and incorrect classifications for each class. The confusion matrix for the accuracy prediction of 3000 image patches from image patch test set (from each class, 1000 image patches were randomly selected), is presented in table 4.11.

Confusion matrix of test set		Predicted class		
		CLL	FL	MCL
Actual class	CLL	975	10	15
	FL	7	988	5
	MCL	12	5	983

TABLE 4.11: Patch-based method evaluation

Each row of the confusion matrix represents the instances of an actual class and each column represents the instances of a predicted class. The diagonal elements represent the number of points for which the predicted label is equal to the true label, while anything off the diagonal was mislabeled by the system. Therefore, the higher the diagonal values of the confusion matrix the better, indicating many correct predictions. The table 11 shows our system CNN-NHL can predict perfectly: 975 image patches of CLL, 988 of FL and 983 of MCL. However, the highest rate of misclassification is 1.35%, which was between MCL and CLL. There are 15 image patches of CLL predicted as MCL and 12 image patches of MCL predicted as CCL. We postulate that the reason for these misclassifications is CLL may present some similarities with MCL as mentioned in [22].In addition, it can be seen that the confusion between FL class with each of two classes MCL and CLL is very weak; because there is a distinction between the slides of FL tissue samples and those of MCL and of CLL, as mentioned in [22]. We postulate that the reason for this weak confusion returns to H&E staining.

In Table 4.12, several examples of correctly and incorrectly classified image patches for different classes (CLL, FL, MCL) are presented. To simplify the explanation of this table, we take as example the image patches, (a), (b) and (c) which are CLL image patches. (a) is correctly classified whereas the (b) and (c) are incorrectly classified; (b) is classified as FL image patch but (c) is classified as MCL image patch.

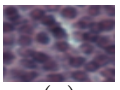
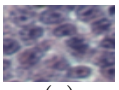
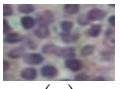
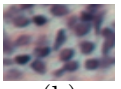
Actual class	*CLL*	 (a)	 (b)	 (c)
	FL	 (d)	 (e)	 (f)
	MCL	 (g)	 (h)	 (i)
		CLL	*FL*	*MCL*
	Predicted class			

TABLE 4.12: Several examples of correctly and incorrectly classified image patches

Ensemble predictions on image patch test set were analyzed for true positives (TP), true negatives (TN), false positives (FP), and false negatives (FP). These parameters were used to calculate a variety of performance metrics, including precision (PR), recall (RE), accuracy (ACC), F1-score (F1) (see equations 3.2, 3.3, 3.4 and 3.5) The testing performance and comparison againsted the existing methods are shown in Table 4.13 to existing deep learning-based approaches for NHL classification, on same dataset, in [4] and [72].

Methods	*PR*	*RE*	*F1*	*ACC*
Proposed approach	0.982	0.982	0.981	0.987
AlexNet [4]	-	-	-	0.965
ResNet [72]	-	-	-	0.976

TABLE 4.13: Patch-based method testing performance and comparison

4.5.3.1 Class prediction of an entire image

In this experiment, from each subtype (CLL, FL or MCL), 20 entire images were randomly selected and from each one, 100 patches were randomly extracted. A voting scheme per subtype (class) was used where votes were aggregated based on the CNN-NHL's output per patch. In a winner-take-all, the class with the highest number of votes became the designated class for the entire image. The following tables present the obtained results:

Votes	for CLL	entire	images
	CLL	FL	MCL
1	**98**	0	2
2	**98**	0	2
3	**94**	0	6
4	**94**	0	6
5	**78**	0	22
6	**90**	0	10
7	**90**	0	10
8	**98**	0	2
9	**90**	8	2
10	**100**	0	0
11	**88**	6	6
12	**96**	2	2
13	**54**	6	40
14	**100**	0	0
15	**100**	0	0
16	**100**	0	0
17	**90**	8	2
18	**64**	2	34
19	**86**	0	14
20	**94**	4	2
Average	**90.1**	1.8	8.1

Votes	for FL	entire	images
	CLL	**FL**	MCL
1	38	**62**	0
2	2	**98**	0
3	40	**60**	0
4	0	**84**	16
5	0	**94**	6
6	0	**100**	0
7	2	**98**	0
8	0	**98**	2
9	2	**90**	8
10	10	**84**	6
11	0	**98**	2
12	0	**98**	2
13	0	**100**	0
14	0	**100**	0
15	40	**60**	0
16	14	**82**	4
17	14	**74**	12
18	4	**96**	0
19	2	**98**	0
20	0	**100**	0
Average	8.4	**88.7**	2.9

Votes	for MCL	entire	images
	CLL	FL	**MCL**
1	2	0	**98**
2	34	0	**66**
3	42	0	**58**
4	33	0	**67**
5	11	0	**89**
6	0	0	**100**
7	0	0	**100**
8	0	2	**98**
9	2	0	**98**
10	0	0	**100**
11	22	4	**74**
12	0	2	**98**
13	18	0	**82**
14	0	0	**100**
15	2	0	**98**
16	2	0	**98**
17	0	0	**100**
18	0	0	**100**
19	2	2	**96**
20	0	2	**98**
Average	8.9	0.6	**90.5**

From this experiment, we have seen that the highest number of votes and their averages always corresponds to the actual class for the entire image. The following tables (Tables 4.14, 4.15 and 4.16) present three examples of entire images with their vote numbers, per subtype. These examples are randomly chosen. In this context, the total accuracy (100%) is achieved in entire image based method.

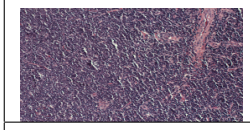
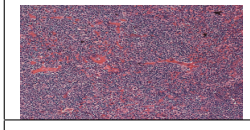
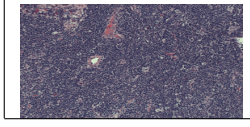
CLL's images	CLL	FL	MCL	Predicted class
	94	0	6	CLL
	100	0	0	CLL
	96	2	2	CLL

TABLE 4.14: Voting scheme and class predictions of entire CLL's images

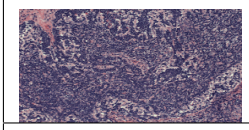
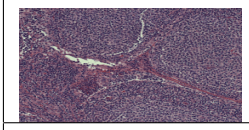
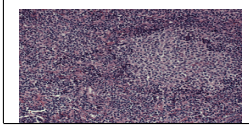
FL's images	CLL	FL	MCL	Predicted class
	14	82	4	FL
	0	100	0	FL
	2	90	8	FL

TABLE 4.15: Voting scheme and class predictions of entire FL's images

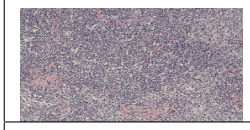
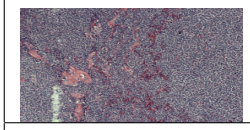
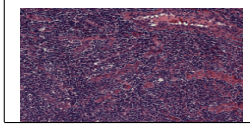
MCL's images	CLL	FL	MCL	Predicted class
	2	2	96	MCL
	0	0	100	MCL
	33	0	67	MCL

TABLE 4.16: Voting scheme and class predictions of entire MCL's images

4.6 Conclusion

In this chapter, we have described the detailed design for our NHL-CNN system with the used tools for developing front-end and back-end of this system. The proposed system, only needs the image patches which have been tagged with the class label (in NIA dataset) to learn the most discriminating representations for class separability, and self-discover the nuanced disease patterns within each class. The experimental results were evaluated considering different performance metrics including: precision, recall, accuracy and F1-score. The obtained results were very promising compared to existing deep learning-based approaches for NHL in [4] and [72].

Conclusion

With the latest advancements in digital pathology, enabling the rapid collection of digital slides, image analysis plays a key role in the day-to-day role of a pathologist. Accurate classification for different Non-Hodgkin Lymphomas (NHL)is one of the main challenges in clinical pathological diagnosis due to its intrinsic complexity.

In this work, we have explored the potential of DL for DP image analysis. We have investigated the concepts of DL with a CNN algorithm to build NHL model for three diagnostic categories: mantle cell lymphoma (MCL), follicular lymphoma (FL), and chronic lymphocytic leukemia (CLL), on whole-slide images.

The proposed deep learning CNN for classifying the NHL subtypes (NHL-CNN) model excels by learning relevant features directly from image patches of NIA curated dataset images.

In fact, in this setting, our NHL-CNN model is distinguished by:

- A tissue level classification strategy, which consists in directly learning all the representative features of the tissue class via DL.
- Obviates the need for explicit primitive identification and provides a more direct pathway to the final classification while not at the same time requiring a comprehension of the domain-specific (potentially unknown) relationships of primitives.
- Only needs the image patches which have been tagged with the class label (in NIA dataset) to learn the most discriminating representations for class separability, and self-discover the nuanced disease patterns within each class.

After successfully training the model, the testing accuracy is computed according to two methods: patch-based and entire image based method. In the first method, for

3000 image patches from image patch test data (from each class, 1000 image patches were randomly selected), we have achieved 98.2 % testing accuracy and the highest rate of misclassification between two classes is 1.35%, which was between MCL and CLL. However, the total accuracy (100%) is achieved in entire image based method, where the class prediction of an entire image is the most predicted of its 50 randomly selected patches. In addition, the achieved accuracy in our patch-based method was also very promising compared to existing deep learning-based approachs for NHL classification in [4] and [72].

From this evaluation, we can conclude that:

- As the number of samples increases, the performance of NHL-CNN model increases significantly.
- When the image patches were incorrectly classified, the quality of its root image may be poor.
- The image patches have significant artifacts, which likely caused its misclassification.
- The misclassification between MCL and CLL class can be back to the similarities beteween the H&E-stained MCL and CLL image, as it was mentioned in [22].

From this research, we conclude that:

- The deep learning based methods are able to solve clinically relevant tasks in diagnostic pathology at the required level for routine diagnostics.
- Our system could serve as a CAD to facilitate pathology diagnostics by discriminating between the three Non-Hodgkin Lymphomas subtypes MCL, FL, CLL.
- The use of our system can provide complementary information to the pathologist for more objective assessment of the lesion. Additionally, it can reduce costs and alleviate the lack of experienced pathologists.

However, this research could be improved in certain points which we consider as prospects for our work.

- Including other disease types of NHL.

- Integration of the pathology expert's diagnosis, into the evaluation of the obtained results, is very important.

- Using other NHL dataset for training, validation, and testing.

Bibliography

[1] F. Aeffner and et al. Introduction to digital image analysis in whole-slide imaging: A white paper from the digital pathology association. *Journal of Pathology Informatics*, 10(9):1–17, 2019.

[2] D. Ameisen and et al. Technologie des lames virtuelles - de la numerisation a la mise en ligne. *Med Sci (Paris)*, 28(11):977–982, 2012.

[3] B. E. Bejnordi. Histopathological diagnosis of breast cancer using machine learning, doctoral thesis in computer science. Radboud University Nijmegen, 2017.

[4] J. Andrew and A. Madabhushi. Deep learning for digital pathology image analysis: A comprehensive tutorial with selected use cases. *Journal of pathology informatics*, (7), 2016.

[5] D. Komura and S. Ishikawa. Machine learning methods for histopathological image analysis. *Computational and Structural Biotechnology Journal*, 8:34–42, 16 2018.

[6] R. Mayer and H. A. Jacobsen. Scalable deep learning on distributed infrastructures: Challenges,techniques and tools. *ACM Comput. Surv.*, 1, 2019.

[7] B. Erickson and al. Machine learning for medical imaging. *RadioGraphics*, pages 505–515, 2017.

[8] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning. arXiv: 1603.07285 [stat.ML], 2018.

[9] B. Sahiner and et al. Deep learning in medical imaging and radiation therapy. *Medical Physics*, pages 533–536, 2019.

[10] R. Yamashita and al. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9, 06 2018.

[11] Md.Z. Alom and al. A state-of-the-art survey on deep learning theory and architectures. *Electronics*, 8:292, 03 2019.

[12] G. Larsson, M. Maire, and G. Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. ICLR2017, 2017.

[13] E.J. Guerard and al. Bishop overview of non-hodgkin's lymphoma dis mon. volume 4, pages 208–218, 58 2012.

[14] E. Montagnon and al. Deep learning workflow in radiology: a primer. *Insights into Imaging*, 11:1–15, 2020.

[15] J. Prewitt and ML. Mendelsohn. The analysis of cell images. *Annals of the New York Academy of Sciences*, 1966.

[16] J.D. Webster and R.W. Dunstan. Whole-slide imaging and automated image analysis: Considerations and opportunities in the practice of pathology. *Veterinary Pathology*, 51(1):211–223, 2014.

[17] O. Sertel and al. Computer-aided detection of centroblasts for follicular lymphoma grading using adaptive likelihood-based cell segmentation. *IEEE transactions on bio-medical engineering*, 57:2613–6, 10 2010.

[18] Newitt. Whole slide imaging for primary diagnosis: 'now it is happening'. *CAP Today*, 2017.

[19] J. Swerdlow and al. Who classification of tumours of the haematopoietic and lymphoid tissues. *Lyon; IARC*, 2, 01 2008.

[20] N.V. Orlov and al. Automatic classification of lymphoma images with transform-based global features. *IEEE Transactions on Information Technology in Biomedicine*, 14:1003–1013, 2010.

[21] T. Tosta and al. Segmentation methods of h&e-stained histological images of lymphoma: A review. *Informatics in Medicine Unlocked*, 9, 2017.

[22] G.P. Canellos and al. The lymphomas. Saunders Elsevier, 2006.

[23] H. Jiang. An effective multi-classification method for nhl pathological images. IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2018.

[24] Y. LeCun and al. Deep learning. *Nature*, 521:436–444, 2015.

[25] Y. Lecun and al. Reading checks with multilayer graph transformer networks. volume 1, pages 151–154, 05 1997.

[26] J. Latif and al. Medical imaging using machine learning and deep learning algorithms: A review *. 03 2019.

[27] N. Peter and et al. Nuclei segmentation in histopathology images using deep neural networks, biomedical imaging,isbi 2017. In *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*, IEEE, pages 933–936, 2017.

[28] A. Cruz-Roa and et al. Automatic detection of invasive ductal carcinoma in whole slide images with convolutional neural networks. *SPIE Medical Imaging*, 9041(7): 1–14, 2014.

[29] R. Ferjaoui and al. Lymphoma lesions detection from whole body diffusion-weighted magnetic resonance images. In *5th International Conference on Control, Decision and Information Technologies*, pages 364–369, 2018.

[30] S. Dwivedi and et al. The advent of digital pathology: A depth review, journal of entomology and zoology studies. *JEZS*, 7(2):43–49, 2019.

[31] G.A. Meijeret and et al. Origins of image... analysis in clinical pathology. *J Clin Pathol*, 50(5):365–370, 1997.

[32] L. Pantanowitz and et al. Validating whole slide imaging for diagnostic purposes in pathology. *Arch Pathol Lab Med*, pages 1–13, 2013.

[33] N. Farahani and et al. Whole slide imaging in pathology: advantages, limitations, and emerging perspectives. *Pathol Lab Med Int*, 7:23–33, 2015.

[34] A. Karsnas. Image analysis methods and tools for digital histopathology applications relevant to breast cancer diagnosis. In *Dissertations from the Faculty of Science and Technology*, volume 4792, page 129. Acta Universitatis Upsaliensis, 2014.

[35] H. Irshad and et al. Methods for nuclei detection, segmentation, and classification in digital histopathology: A review-current status and future potential. *IEEE Rev Biomed Eng*, 7:97–114, 2014.

[36] A.H. Beck and et al. Systematic analysis of breast cancer morphology uncovers stromal features associated with survival. *Sci Transl Med*, (3):108–113, 2011.

[37] Md.Z. Alom and et al. Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation. *Computer Vision and Pattern Recognition (cs.CV)*, 2018.

[38] A. Madabhushi. Digital pathology image analysis: opportunities and challenges. *Imaging Med.*, 2018.

[39] V. Gupta and al. Breast cancer histopathological image classification: Is magnification important? Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2017.

[40] Y. Liu and al. Detecting cancer metastases on gigapixel pathology images. *ArXiv*, 2017.

[41] A. Saito and al. A novel method for morphological pleomorphism and heterogeneity quantitative measurement: named cell feature level co-occurrence matrix. *J Pathol Inform*, 2016.

[42] AM. Khan and al. A nonlinear mapping approach to stain normalization in digital histopathology images using image-specific color deconvolution. *IEEE Trans Biomed Eng*, 2014.

[43] N. Nilsson. The quest for artificial intelligence: A history of ideas and achievements. Cambridge University Press, 2010.

[44] J. Searle. Minds, brains, and programs. *Behavioral and Brain Sciences*, 3:417–424, 1980.

[45] A. Lundervold and Ar. Lundervold. An overview of deep learning in medical imaging focusing on mri. *Zeitschrift fur Medizinische Physik*, 29, 12 2018.

[46] L. Deng. A tutorial survey of architectures, algorithms, and applications for deep learning-erratum. *APSIPA Transactions on Signal and Information Processing*, 3, 12 2014.

[47] S. Pouyanfar and al. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys*, 51:1–36, 09 2018.

[48] C. Bishop. Pattern recognition and machine learning. Springer Science and Business Media, 2006.

[49] D. E. Rumelhart and et al. Learning representations by back-propagating errors. *Nature 323*, pages 533–536, 1986.

[50] A. Borovkov. Image classification with deep learning. Doctoral Thesis in Computer Science, University of Dundee, 2017.

[51] G. Jiuxiang and al. Recent advances in convolutional neural networks. *Pattern Recognition*, 12 2017.

[52] Y. Boureau and al. A theoretical analysis of feature pooling in visual recognition. Proceedings of the International Conference on Machine Learning (ICML), 2010.

[53] M.D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. volume 4792, pages 818–833. Proceedings of the European Conference on Computer Vision (ECCV), 2014.

[54] A. Krizhevsky and al. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.

[55] S. Karen and A. Zisserman. Very deep convolutional networks for large-scale image recognition. ICLR 2015, 2015.

[56] S. Ruder. An overview of gradient descent optimization algorithms. arXiv: 1609.04747, 2016.

[57] E. Hinton and al. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv*, pages 1–18, 2012.

[58] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 448–456. PMLR, 2015.

[59] I. Sergey and C. Szegedy. Batch normalization:accelerating deep network training by reducing internal covariate shift. arXiv: 1502.03167, 2015.

[60] H. Kaiming and al. Deep residual learning for image recognition. pages 770–778, 06 2016.

[61] G. Huang and al. Densely connected convolutional networks. arXiv:1608.06993 . DenseNet, 2016.

[62] M. Zhou and al. Progress in neural nlp: Modeling, learning, and reasoning, engineering. *Research Artificial Intelligence*, 6:275–290, 2020.

[63] C. Xiong and al. Dynamic memory networks for visual and textual question answering. *arXiv*, 2016.

[64] C. Angermueller and al. Deep learning for computational biology. *Molecular Systems Biology*, 2016.

[65] C. Syrykh and al. Accurate diagnosis of lymphoma on whole-slide histopathology images using deep learning. *npj Digit. Med.*, 63, 2020.

[66] J. Scheurer and al. Semantic segmentation of histopathological slides for the classification of cutaneous lymphoma and eczema. volume 1248, pages 26–42. Communications in Computer and Information Science, 2020.

[67] He. Kaiming and al. Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016.

[68] F. Spanhol and al. Breast cancer histopathological image classification using convolutional neural networks. 07 2016.

[69] F. Spanhol and al. A dataset for breast cancer histopathological image classification. volume 63, pages 1455–1462. IEEE Trans Biomed Eng., 07 2016.

[70] N. Brancati and al. Multi-classification of breast cancer histology images by using a fine-tuning strategy. volume 10882, pages 771–778. LNCS, Springer, 2018.

[71] G. Aresta and al. Bach: Grand challenge on breast cancer histology images. *Medical Image Analysis*, 56, 05 2019.

[72] N. Brancati and al. A deep learning approach for breast invasive ductal carcinoma detection and lymphoma multi-classification in histological images. *IEEE Access*, 7, 04 2019.

[73] N. Dimitriou and al. Deep learning for whole slide image analysis: An overview. *Frontiers in Medicine*, 6:264, 2019.

[74] J. Xu and al. Deep learning for histopathological image analysis: Towards computerized diagnosis on cancers. pages 73–95, 2017.

[75] M. Hosseini and al. Atlas of digital pathology: A generalized hierarchical histological tissue type-annotated database for deep learning. 07 2019.

[76] J. Rodriguez and al. Sensitivity analysis of k-fold cross validation in prediction error estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32:569–575, 2009.

Digital pathology image analysis with deep learning

Abstract:

The advent of whole-slide imaging in digital pathology (DP) has helped the advancement of computer-aided diagnostics of tissue, via digital image analysis.

In our master's project, we have explored the potential of deep learning (DL)for DP digital pathology image analysis.We have investigated the concepts of DL with a Convolutional Neural Network (CNN) algorithm to build a Non-Hodgkin Lymphomas (NHL) multi-classification model that can accurately classify subtypes of NHL concerning chronic lymphocytic leukemia (CLL), follicular lymphoma (FL) and mantle cell lymphoma (MCL), on whole slide images.

The proposed deep learning CNN for classifying the Non-Hodgkin Lymphomas subtypes (NHL-CNN) model excels by learning relevant features directly from image patches of NIA curated dataset images.

After successfully training the model, the testing accuracy is computed according to two methods: patch-based and entire image based method. In the first method, we have achieved 98.2 % testing accuracy. However, the total accuracy (100%) is achieved in entire image based method. The experimental results clearly demonstrate the robustness and efficiency of our proposed NHL-CNN as compared to existing DL approaches for NHL classification in literature.

Keywords: Digital pathology, Deep learning, Convolutional Neural Networks, Non-Hodgkin Lymphomas, Multi-classification

Analyse d'images de pathologie numérique avec apprentissage en profondeur

Résumé:

L'avènement de l'imagerie sur les lames virtuelles en pathologie numérique (DP : digital pathology) a permis de faire progresser le diagnostic assisté par ordinateur (CAD: computer-aided diagnostics) des tissus, via l'analyse d'images numériques.

Dans notre projet de Master, nous avons exploré le potentiel de l'apprentissage en profondeur (DL : deep learning) pour l'analyse d'images en pathologie numérique. Nous avons étudié les concepts DL avec un algorithme de réseau de neurones convolutionnel (CNN : Convolutional Neural Network) pour construire un modèle multi-classification des Lymphomes Non Hodgkiniens (NHL) qui peut classer avec précision les sous-types NHL concernant la Leucémie Lymphoïde Chronique (LLC), le Lymphome Folliculaire (FL) et Lymphome à Cellules du Manteau (MCL), sur des images de lames virtuelles.

Le modèle d'apprentissage en profondeur CNN proposé pour la classification des sous-types de lymphomes non hodgkiniens (NHL-CNN) excelle en apprenant les caractéristiques pertinentes directement à partir des pièces des images de jeux de données organisées par le NIA. Après avoir entraîné avec succès le modèle, la précision de test est calculée selon deux méthodes : méthode basée sur les pièces d'image et méthode basée sur l'image entière.

Dans la première méthode, nous avons atteint une précision de test de 98,2%. Cependant, la précision totale (100%) est obtenue dans la méthode basée sur l'image entière.

Les résultats expérimentaux démontrent clairement la robustesse et l'efficacité de notre proposition de NHL-CNN par rapport aux approches DL existantes pour la classification de NHL dans la littérature.

Mots clés: Pathologie numérique, apprentissage en profondeur,réseaux de neurones convolutionnels, lymphomes non hodgkiniens Le lymphome non hodgkinien (LNH), multi-classification, Lame virtuelle , L'apprentissage en profondeur, Les réseaux de neurones (CNN)

تحليل صور علم الأمراض الرقمي بالتعلم العميق

ملخص:

أدى ظهور التصوير على الشرائح الافتراضية في علم الأمراض الرقمي إلى إمكانية تطوير تشخيص الكمبيوتر للأنسجة، من خلال تحليل الصور الرقمية. لقد ارتأينا في مشروع بحثنا (الماستر)الخوض في تقنية التعلم العميق لتحليل صور علم الأمراض الرقمي، فقمنا بدراسة مفاهيم خوارزمية الشبكة العصبية التلافيفية لتصنيف بدقة الأنواع الفرعية للورم الليمفاوي اللاهودجكين والمتعلقة بسرطان الدم الليمفاوي المزمن (CLL)، ورم الغدد اللمفاوية (FL)وسرطان الغدد الليمفاوية لخلية الوشاح(MCL) ، على صور الشرائح الافتراضية. لقد اعتمدنا على مفاهيم التعلم العميق وعلى استخدام خوارزمية الشبكة العصبية التلافيفية(CNN) لطرح نموذج لتصنيف هذه الأنواع الفرعية للورم الليمفاوي اللاهودجكين. تفوق النموذج المقترح (NHL-CNN)للتعلم العميق لتصنيف هذه الأنواع، من خلال تعلم الميزات ذات الصلة المباشرةً لقطع صور مجموعة البيانات. بعد تدريب النموذج بنجاح، تم حساب دقة الاختبار وفق طريقتين: طريقة تعتمد على قطع الصور وطريقة تعتمد على الصورة بأكملها. في الطريقة الأولى، حققنا دقة اختبار 98.2٪ وتم تحقيق الدقة الكاملة (100٪) في الطريقة القائمة على الصورة بأكملها. النتائج التجريبية وضحت متانة وكفاءة المقترح، مقارنة بالدراسات المنجزة حاليا والمخصصة لتصنيف هذه الأنواع من الأورام. الشبكات العصبية الالتفافية

الكلمات المفتاحية:

علم الامراض الرقمي، التعلم العميق، الشبكة العصبية الالتفافية، الورم الليمفاوي اللاهودجكين، التصنيف المتعدد، صور الشرائح الافتراضية.