Deep Learning-Based Classification of Leukemic B-Lymphoblast Cells

Ludovico Gelsomino

Supervisor: Prof. Giacomo Zanella

Università Commerciale Luigi Bocconi

Bachelor of Science in Economics, Management and Computer Science

September 2022

Abstract

Leukemia is a cancerous hematological disorder, characterized by the uncontrolled growth of blood cells, usually white blood cells in the bone marrow. The malignant cells build up in the bone marrow, which eventually becomes unable to produce healthy blood cells. Leukemia is the most common cancer among children and teens, the American Cancer Society estimated that in 2022 at least 60,650 new cases of Leukemia will be diagnosed in the US and 24,000 people will die from the disease. As for most form of cancers, early diagnosis and treatment are crucial factors in survival rates. Thus, it is important to develop automated tools to support physicians throughout the diagnostic process. Automatic detection with computer-aided diagnosis could enable a faster, cheaper and more accurate diagnosis. This work is focused on the application of deep learning models for the classification of malignant B-Lymphoblasts from early benign B cell precursors, differentiating between the two is a vital component of the diagnosis of B-Lymphoblastic Leukemia. The final results show how a simple convolutional neural network, with three convolution layers and one fully connected layer, can classify malignant B-Lymphoblasts from early benign B cell precursors with 82.5% accuracy, 87% precision and 96% specificity.

Contents

List of Tables

List of Figures

1. Introduction

The blood system is constituted by more than 10 different blood cells type with a wide arrays of functions: Erythrocytes (Reed Blood Cells) deliver oxygen to the body tissues and return carbon diaxide to the lungs via the circulatory systems, Lymphocytes (White Blood Cells) are involved with innate and acquired immunity, and Megakaryocytes generate platelets to regulate hemostasis and thrombosis (Rieger and Schroeder, 2012; Holinstat, 2017).

The production of these cells in adults occurs mainly in the Bone Marrow by a process called Hematopoiesis. Throughout the process, a small number of multipotential cells called Hematopoietic Stem Cells (HSCs) differentiate and expand into as many as 10^{11} cells each day. HSCs are the building blocks of Hematopoiesis because they are at the apex of a hierarchy of numerous progenitor cells and can give rise to all blood cell lineages. During hematopoiesis, transcription factors that regulate gene expression and repression cause the HSCs to lose one or more developmental potentials, to commit to a single cell lineage, and eventually to mature into the corresponding blood cell type. According to the prevalent theories, the first stage of differentiation occurs when the HSC enters one of two pathways by becoming either a Common Lymphoid Progenitor (CLP) or a Common Myeloid Progenitor (CMP). The CLP has the potential to develop into a B-Lymphocyte, T-Lymphocyte, or a Natural Killer Cell (NK), whereas the CMP can generate Myeloid, Erythroid or Megakaryocytic lineages, as displayed in figure 1.1 (Kaushansky et al., 2015).

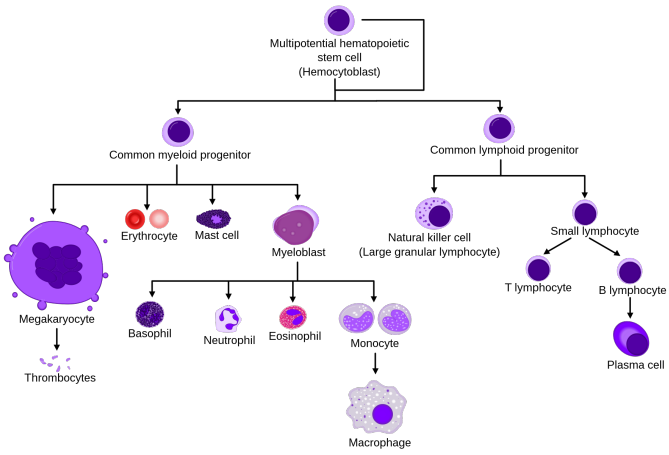


Figure 1.1: Simplified Human Hematopoiesis.
By Rad and Häggström (2009)

Genetic errors can halt the maturation of HSCs at different stages of hematopoiesis, causing the proliferation of leukemic cells. The build-up of leukemic cells in the bone marrow is called Leukemia and causes a disruption of the hematopoietic system. As a consequence, the individual affected by leukemia does not produce healthy blood cells and usually suffers from anemia, frequent infections and bleeding tendencies (Kaushansky et al., 2015). Leukemia is generally classified by the type of blood cells affected and by how fast the diseases progresses. In terms of the targeted blood cells, leukemia can be divided into Lymphocytic and Myeloid, whereas depending on the rate of progression leukemia is either Acute (rapidly progressing) or Chronic (slowly progressing). This classification differentiates four types of Leukemia:

- Chronic Lymphocytic Leukemia (CLL)
- Chronic Myeloid Leukemia (CML)
- Acute Myeloid Leukemia (AML)
- Acute Lymphocytic/Lymphoblastic Leukemia (ALL)

This neoplastic disease affects individual of all ages, however the age distribution of the different types is not alike. For instance, in the US the most prevalent forms of leukemia among adults (20 years of ages and older) are CLL (38%) and AML (31%), whereas among children and adolescents (ages 0 to 19 years) the most prevalent type is ALL (75%) (American Cancer Society, 2022).

In 2020, the Global Cancer Observatory identified leukemia as the 13th most commonly diagnosed cancer and the 10th cause of cancer mortality, with 474,519 cases and 311,594 deaths (J et al., 2020; Ferlay et al., 2021; Sung et al., 2021). The decline of the age-standardized incidence rate (ASIR) between 1990 and 2017 indicates a decrease in the incidence of the disease. Altough promising, the decline of the incidence rate in not accompanied by a reduction in the number of newly diagnosed cases. In fact, between 1990 and 2017, the number of newly diagnosed cases increased by 46% (Dong et al., 2020). Globally, the distribution of leukemia is patterned by country-level development, with leukemia being significantly more present in developed countries (Bispo et al., 2020).

To diagnose the disease physicians rely on complete blood counts, blood smear and bone marrow aspirates. The microscopic analysis of blood smear and bone marrow aspirates is used to identify the presence of leukemic blasts in the circulating blood and in the bone marrow. Unfortunately, the distinction between leukemic and normal cells can be subtle and physicians may be unable to pick on the small morphological differences, causing a delay in the diagnosis of the disorder and lowering the chances of survival. Therefore, it is important to develop tools and practices to help physicians throughout the diagnostic journey.

The following pages will be dedicated to outline the main characteristics of Acute Lymphoblastic Leukemia, and present a framework for a deep learning based computer-aided diagnosis systems that can be used by physicians to improve the detection and identification of leukemic blasts in the circulating blood for Precursor B-cell Acute Lymphoblastic Leukemia.

1.1. Acute Lymphoblastic Leukemia

Acute Lymphoblastic Leukemia (ALL) occurs when a single T- or B-Lymphoid Progenitor Cell undergoes a multistep somatic mutation during the developmental stage. These mutations slow down cell division, increase the time required to synthesize DNA and most importantly, they cause an abnormal response to growth and death signals. By disrupting normal cell fate decisions, these genetic alterations interfere with the development of the lymphoid progenitor cell and cause the build-up of these immature and anomalous progenitor cells, called Leukemic Lymphoblasts or Leukemic Blasts, in the marrow. Eventually, the malignant cells replace the normal hematopoietic cells, causing the suppression of normal blood cells development, and inducing neutropenia, anemia, and thrombocytopenia(Kaushansky et al., 2015; Rieger and Schroeder, 2012).

As mentioned, ALL can affect the maturation process of both T- and B-Lymphocytes. Immunologic, cytogenetic, and molecular genetic methods can be used to divide ALL in several sub classes depending on what lymphocyte has been affected and at what stage of the developmental process the lymphocyte has been compromised. Among all the different distinction, the only ones of therapeutic importance are those between: T-cell ALL, Mature

B-cell ALL and Precursor B-cell ALL (Kaushansky et al., 2015).

1.1.1. Incidence

Leukaemia is the most common childhood cancer and among all types of leukemia, ALL is the most prevalent among children and adolescents (ages 0 to 19 years), accounting for 75% of cases (American Cancer Society, 2022). The 2022 American Cancer Society estimates report that, in the US, 6,660 new cases of ALL will be diagnosed and 1,560 people will die from the disorder (Siegel et al., 2022; American Cancer Society, 2022).

ALL has a bimodal age distribution with pediatric incidence rates being extremely higher than older age groups (Bispo et al., 2020). The Surveillance, Epidemiology, and End Results (SEER) Program cancer statistics indicate that children between the age of 1 and 4 report the highest incidence rate, about 7.8 per 100,000 individuals. Throughout adulthood the incidence rate is fairly stable at 1 per 100,000 individuals, with a gradual increase for adults older than 60 (National Cancer Institute, 2022a). The incidence rate is slightly higher among males than females, respectively 1.9 and 1.3, which suggests that ALL is slightly predominant in males. The annual percentage change (APC) between 1975-2019 among all groups is 0.8 per 100,000 individuals which indicates that the incidence of ALL in the US is increasing over time (National Cancer Institute, 2022b). While other types of leukemia are more prevalent in developed countries, global incidence patterns indicate that the highest incidence of ALL is among South and Central American countries (Bispo et al., 2020).

1.1.2. Pathogenesis

Lymphoid cells are generated through a step-wise maturation process called Lymphoid Hematopoiesis (or Lymphopoiesis) in which HSCs developed into T- or B-Lymphocytes. Although the precise pathogenetic events that lead to ALL are unknown, it has been established that ALL begins when a sequence of mutations modify certain cellular functions, including an improved ability of self-renewal, a disruption of control of normal proliferation, a block in differentiation and an increased resistance to death signals. These mutations block the lymphoid differentiation and cause the proliferation and survival of these malignant cells in the bone marrow (Zuckerman and Rowe, 2014; Kaushansky et al., 2015). The abnormal accumu-

lation of the malignant cells in the bone marrow suppresses normal blood cell development, causing: low levels of healthy red blood cells or hemoglobin (Anemia), low concentration of neutrophils (Neutropenia) and low count of platelets (Thrombocytopenia)(Kaushansky et al., 2015). Altough the disease begins in the bone marrow, it is not uncommon for the leukemic blasts to spill out of the marrow and end up in several extramedullary sites such as the meninges, gonads, thymus, liver, spleen and lymph nodes (Kaushansky et al., 2015).

1.1.3. Etiology

To date, the precise causes of ALL are still unknown, however several risk factors have been identified. These include: Genetic Syndromes, Radiation Exposure, Chemical Exposure and Viral Infections. ALL doesn't have a strong inherited component and only 5% of cases can be associated with predisposing and inherited generic syndromes. Among all genetic syndromes, the Down syndrome is associated with the highest risk of developing ALL, with people affected by the syndrome having 30 times the risk of developing the disease with respect to non-affected individuals (Kaushansky et al., 2015).

1.1.4. Laboratory Features and Diagnosis

Patients with newly diagnosed ALL usually present anemia, neutropenia and thrombocytopenia. The severity of these is positively correlated with the number of Leukemic Lymphoblasts in the bone marrow. At diagnosis, the vast majority of patients present malignant cells in the circulating blood and other organs.

To diagnose ALL, physicians rely on blood and bone marrow tests. The blood test usually comprehend a Complete blood count (CBC) and a peripheral blood smear. The CBC is executed to count the number of red blood cells, white blood cells and platelets, whereas the microscopic analysis of the blood smear is performed to inspect the morphology of the blood cells and detect the presence of Leukemic Lymphoblasts in the circulating blood. The presence of circulating lymphoblasts is usually sufficient to diagnose the disease.
Given that approximately 10% of patients lack circulating blasts, the absence of Leukemic cells in the circulatory system cannot exclude ALL and renders necessary a microscopic examination of a bone marrow aspirate. If the patient presents leukemic lymphoblasts in the

bone marrow, she/he has developed acute lymphoblastic leukemia (Kaushansky et al., 2015).

As discussed, the diagnosis of ALL relies almost entirely on the morphological differences between lymphoblasts and healthy lymphoid progenitors. Fortunately, lymphoblasts present a series of characteristics that can be used to distinguish them from healthy lymphocytes or progenitor cells. They are relatively small (one to two times the size of normal lymphocytes) with a light-blue cytoplasm, a round or slightly indented nucleus and poorly noticeable nucleoli. In some cases, lymphoblasts may be large, with relatively large nucleoli and a moderate amount of cytoplasm (Kaushansky et al., 2015) .

1.1.5. Therapy and Prognosis

For therapeutic purposes it is important to distinguish between Mature B-cell ALL, T-cell ALL and Precursor B-cell ALL. Mature B-cell ALL is treated with a combination of drugs given over a short period of time, whereas the treatment of T-cell and Precursor B-cell ALL is differentiated in three stages: remission induction, intensification and prolonged continuation. During the first phase chemotherapy is used to induce a complete remission and restore normal hematopoiesis. Thanks to the advances in modern medicine, 98% of children and 85-90% of adults complete these phase successfully. After remission has been achieved, the patient begins an intensification therapy with the purpose of killing any cancer cell that may be left in the body. A successful intensification therapy significantly improves the odds of long term success. After all the leukemic cells have been killed, the patient begins the continuation therapy, during which drugs are administered to prevent re-laps and improve overall survival (Kaushansky et al., 2015).

The current 5-year survival rate in the US for people under age 20 is 89% and it drops to 20% for people of age 20 and older (American Cancer Society, 2022). The lower survival rates in adults is partly related to a high frequency of cases with unfavourable genetic abnormalities and poor tolerance for intensive treatment (Kaushansky et al., 2015).

1.2. Computer-Aided Diagnosis Systems

Computer-aided detection/diagnosis (CAD) has been been a major field of research and in the latest years it is experiencing a change thanks to advancement of artificial neural networks. In the current framework, CAD systems use image-based information, alone or in combination with other data, to help clinicians with the detection and diagnosis of diseases. When properly trained, the predictions generated by the CAD system may be used by a clinician as a second opinion or support information during the decision making process.

Basic research on computerized analysis of medical imaging began in the 1960s with the idea of building fully automated computer diagnosis software. However, the lack of computational performance and powerful image-processing techniques caused little to no progress in the field. The absence of significant results forced the researchers to pursue a different path: instead of building fully automated systems, they decided to develop systems that could be utilized by radiologist to enhance the diagnostic outcome. Thus, in the early 1980s systematic research on computer-aided diagnosis began in the Kurt Rossmann Laboratory at the University of Chicago, with the objective of building a system to aid radiologists during image interpretation and decision making (Doi, 2007). Several years later, in 1998, the FDA approved the first commercial CAD system for the detection of breast cancer in screening mammography. Since then, mammography CAD has been widely adopted and in 2016 approximately 92% of mammography labs in the US used CAD systems (Keen et al., 2018). Altough very popular, the performance and reliability of such systems has been somehow controversial. Early studies on the adoption of mammography CAD showed that single reading with CAD improved cancer detection and recall rate (Gilbert et al., 2006; Henriksen et al., 2019), whereas large community studies found little to no improvements in the use of CAD Lehman et al. (2015).

The lack of performance of traditional CAD systems in the clinical setting can be attributed to several technical limitations. The robustness and reliability of the systems are severely weakened by the limited computational resources, the small datasets and the poor image quality on which the systems were trained. The reference standard used for training

relies too much on human expertise, which has been proved to be prone to mistakes (Gao et al., 2019; Waite et al., 2017). Among all, the single factor that contributed the most to the clinical failings of traditional CAD systems was the lack of capability for independent improvements, due to the absence of tools for continuous feedback and learning.

1.2.1. AI based computer-aided diagnostic

Traditional CAD systems require the human designer to devise and create feature that should be recognized in the image, and then hard-code these features in the system. New CAD platforms based on AI technology offer new perspectives because they do not require manual feature design, reduce human involvement and provide capabilities for continuous feedback and learning so that these systems can actively learn from their mistakes. Deep Learning algorithms are particularly suitable to CAD systems because they can be designed to learn features independently and improve their performance over time. This technology is a better fit for the task than simpler Machine Learning (ML) models, such as decision trees, because medical images rarely produce the well defined and highly predictive feature required to successfully train ML models.

Currently, the task of recognizing features in an image is best carried out by convolutional neural networks (CNNs). CNNs were first applied to medical image analysis in 1993 for the detection of lung cancer in chest radiographs (Lo et al., 1993). In the following years CNNs were applied to other image analysis, but the limited computational power and data available limited the learning capacity of the networks. However, the recent advancement of machine learning techniques, the availability of low-cost graphical processing units (GPUs) and the effective network training strategies available for deep learning architectures enabled the training of networks containing millions of weights which sparked the AI revolution. These large networks were able to reach and sometimes surpass human-level performance on several tasks and have been successfully adapted to various applications such as self-driving vehicles, chess, Go games and most recently the control of nuclear fusion plasma (Degrave et al., 2022).

The overwhelming success of deep learning in image and pattern recognition triggered an interest in the application of this technology to computer-aided diagnosis. DL-based CAD

systems are still under development, however they have already shown encouraging results that often surpass those of tradition CAD systems for digital mammography, digital breast tomosynthesis and contrast enhanced mammography (Gao et al., 2019; Kooi et al., 2017; Kim et al., 2018; Al-Masni et al., 2018).

Although the development of DL-based CAD systems looks promising, it brings several technical challenges. First of all, DL requires large databases which can be costly to compile and need to comply with privacy laws. Medico-legal and regulatory considerations must be taken into account once the technology approaches the clinical environment. Agencies, such as the U.S. FDA, need to outline a regulatory framework for the use of the technologies. Also, a standardized framework for the evaluation of these systems has to be created and it should be aligned with the recommendations made by several bodies, such as those designed by the Computer Aided Image Analysis Subcommittee (CADSC) (Petrick et al., 2013; Huo et al., 2013).

1.2.2. CAD for ALL

Computer-aided detection and diagnosis of Acute Lymphoblastic Leukemia is currently under development. The current frameworks of CAD systems for ALL consist of four stages: pre-processing, segmentation, feature extraction and selection, and finally classification (Shafique and Tehsin, 2018; El Houby, 2018). See figure 1.2a.

During the preprocessing stage, the image is enhanced and image quality is improved so that it can be segmented and classified more accurately. Several factors, both human and non human-related, can affect the quality of the images: salt or pepper noise, low contrast, false background, mishandling of camera and poor lightning condition. To solve these technical issues several image processing techniques are applied. Usually, histogram equalization and normalization are used to adjust the image contrast while order statistical filters and Gaussian filters are applied to remove salt and pepper noise, and unnecessary image details (Shafique and Tehsin, 2018).

The segmentation stage is necessary to segment the image into different regions, whose

analysis can yield better results with respect to the whole image. In the context of ALL image segmentation is particularly important because segmenting the suspicious malignant cells out of the image allows for a more accurate classification of the leukemic blasts from the normal cells. A wide arrays of algorithms lend themselves to segmentation: K-Means clustering, Fuzzy C-Means Clustering, Watershed Segmentation, Thresholding and Region Growing and Mathematical Morphology have all shown good results (Shafique and Tehsin, 2018). Some research shows that applying a two-level segmentation in which, the sub-images containing cells are extracted out of the whole image and then the nucleus and cytoplasm are segmented out of the sub-images, offers very promising result in terms of feature extraction and accuracy performance (El Houby, 2018).

Feature extraction and selection is a vital part of the process. Extracting and selecting features for the classifiers can significantly decrease the computational cost while still ensuring high accuracy. The most common extracted features for acute lymphoblastic leukemia are both of morphological and texture/intensity nature. The morphological features usually include: Shape, Roundness, Bending Energy and Chain code. The Texture/Intensity features include: Gray Level Concurrence Matrix, Histogram, Gabor Texture Extraction, Color Intensity, Fractal and Hausdorff Dimensions, Entropy and Local Binary patterns (Shafique and Tehsin, 2018). Although feature extraction and selection is necessary for traditional ML algorithms, the more advanced CNNs are designed to automatically extract these features out of the input image. Thus, this part of the process may be unnecessary in a CNN-based CAD for Acute Lymphoblastic Leukemia.

The last stage is classification. Classification consists of two steps: model construction and model usage. During model construction, the model is trained using the dataset, whereas during model usage the model is utilized for the actual classification. The classification accuracy is measured by the percentage of correctly classified test cases. To date, different classifiers have been used to diagnose ALL; these include: Support Vector Machines, KNN, Random Forest, Naive Bayes, Multilayer Perceptron and Probabilistic Neural Networks (Shafique and Tehsin, 2018).

1.2.3. Deep Learning Based CAD for ALL

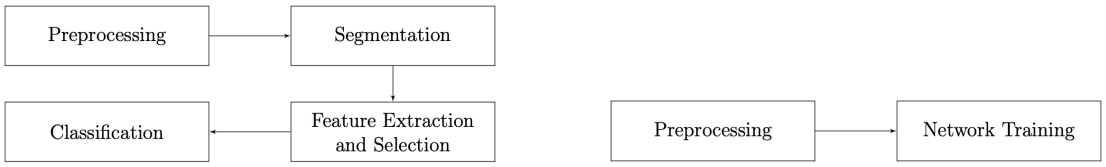
As discussed, the basic framework for building an effective computer-aided diagnosis systems for Acute Lymphoblastic Leukemia requires preprocessing, segmentation, feature selection and extraction, and finally classification. In this matter, the use of deep neural networks could decrease both the number of operations and the human involvement required to build an effective CAD system. For instance, deep convolutional neural networks (DCNNs) can be deployed to perform segmentation, feature extraction and classification.

To date, Convolutional Neural Networks (CNNss) are the most adopted algorithms for computer vision tasks. CNNs is a deep learning model utilized to process data with a grid pattern, such as images, and they are specifically designed to automatically learn spacial hierarchies of features. These networks are usually composed of three types of layers: convolution, pooling and fully connected layers. The convolution and pooling layers perform feature extraction, whereas the fully connected layers compiles the features extracted by the other layers and maps them to the final output (Yamashita et al., 2018). As one layer feeds its output to the next, extracted feature can become more complex and feature selection, if necessary to reduce overfitting or training time, can be carried out by adding a feature selection layer to the network (Figueroa Barraza et al., 2021).

Medical research has already proved the suitability of these architectures to medical image analysis. U-nets, a particular CNN architecture, have been successfully applied to medical image segmentation (Wu et al., 2022) and CNNs have been able to reach physician level performance in several detection and classification tasks, such as skin cancer and lymph node metastases detection (Esteva et al., 2017; Ehteshami Bejnordi et al., 2017).

The unique capability of these networks to perform segmentation, feature extraction and classification with very high accuracy and little to no human intervention, make them extremely suitable for the development of CAD systems. Moreover, the use of convolutional neural network can significantly decrease the complexity and the number of steps necessary to develop an effective CAD system for Acute Lymphoblastic Leukemia. A CNN-based CAD

system for ALL would only require the preprocessing of the images before they are fed to the network. The development process would be reduced from preprocessing, segmentation, feature extraction, feature selection and classification to preprocessing and network training. See figures 1.2a and 1.2b.



(a) Traditional CAD for ALL (b) Deep Learning-based CAD for ALL

Figure 1.2: Figure (a) depicts the pipeline of a traditional CAD system. Figure (b) depicts the pipeline of a deep learning-based CAD system

2. Deep Learning

Deep Learning is a sub-field of machine learning, based on family of computational models called Artificial Neural Networks (ANNs), which are loosely inspired by the structure of the human brain.

The functioning of a biological neuron can be explained by the interaction between three elements: the axon, the cell body, and the dendrites. The dendrites provide incoming stimuli to the cell body, and if the stimuli are strong enough the neuron transmits an electrical pulse along its axons to the other neurons that are connected to it. In this simple scheme, dendrites act as input units, the cell body acts as processing unit, and the axons behave like output units.

The structure and the operation performed by ANNs are loosely inspired by the biological neuron. These computational models are based on a collection of units, called nodes, that are connected to each other through edges. Edges act as input and output units, whereas nodes act as processing units. Each node receives one or multiple signals, computes their weighted summation, and passes it through an activation function. This function maps the numerical inputs of the node to an output value called activation value. The activation value of one neuron is typically passed as input value to all the neurons that are connected to it. Activation values flow through the network until they reach the last node, also called output node, which produces the output value that we are interested in (Kelleher, 2019).

This chapter will be dedicated to the exploration of the fundamental concepts behind the design and the astonishing success of deep learning. Firstly, I will briefly describe the perceptron and its role as the building block of deep neural networks. Secondly, I will present the basic architecture and training techniques of feed forward neural networks. Thirdly, I will focus on Convolutional Neural Networks, which have a unique architecture that is optimized for image processing.

2.1. The perceptron

The power of neural networks to model complex relationships is not a product of complex mathematical models, but rather comes from the dynamic interactions between a large set of simple neurons, called perceptrons.

The perceptron was proposed in 1958 by Rosenblatt as a computational model containing a single input layer and an output node (Rosenblatt, 1958). The input layer contains n nodes, and each one of these transmits a feature to the output node. The model can be defined by a vector of features $\bar{\mathbf{X}} = [x_1 ... x_n]$ and a vector of weights $\bar{\mathbf{W}} = [w_1 ... w_n]$. In the graphical representation, each weight is associated with an edge that connects the input node to the output node. Two functions in the output node process the weight and the feature vectors. The first function computes the weighted sum of the inputs to the neuron $s = w_0 + \sum_{k=1}^{n} x_k \times w_k$, which rewritten in terms of vectors and dot products is equal to $s = w_0 + \bar{\mathbf{X}}^\top \cdot \bar{\mathbf{W}}$. The second function, usually called activation function, maps the weighted sum to the neuron's final output value $y = \varphi(w_0 + \bar{\mathbf{X}}^\top \cdot \bar{\mathbf{W}})$. In many setting, the perceptron includes a bias term w_0 which allows the input of the activation function to be shifted to the left or right by a constant amount w_0. Figure 2.1 shows the basic architecture of the perceptron.

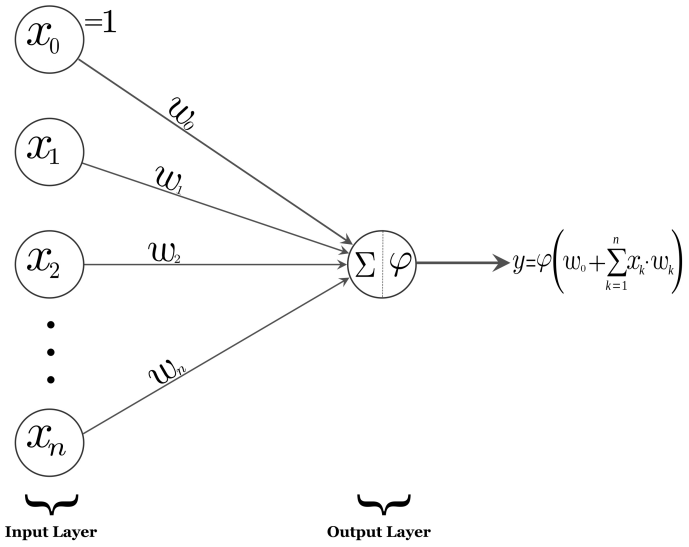


Figure 2.1: Perceptron

Typically, the activation function used within the perceptron is non-linear (e.g. sigmoid function). Different choices of activation functions can be used to simulate different machine learning models. The selection of the function should also be tailored to the particular task at hand. For instance, a sign function can be used when a binary class labels needs to be predicted (classification), an identity function can be implemented to predict real valued targets and the sigmoid function is recommended when predicting the probability of a binary class (Aggarwal, 2018).

Initially, the perceptron was developed to perform linear classification tasks. As a linear classifier, the perceptron was designed to classify data points into a discrete class by performing a linear combination of the input features. For example, consider a task where the training instance is of the form $(\bar{\mathbf{X}}, \boldsymbol{y})$, where each variable contained in $\bar{\mathbf{X}} = [x_1...x_n]$ is a feature, and $\boldsymbol{y} = [y_1...y_n]$ contains the observed value of the binary class variable that we want to predict. Then, the goal of a linear classifier would be to find the linear combination of $\bar{\mathbf{X}}$ that can be used to predicts the class variable of new, unlabeled observations with the best possible accuracy. In the context of classification, the perceptron learning algorithm proposed by Rosenblatt always converges to a solution when the dataset is linearly separable. Arguably, the perceptron was the first algorithm with such formal guarantee of convergence. Unfortunately, the perceptron presents severe limitations that do not allow the convergence of the algorithm when the dataset is not separable by a line or hyperplane. Altough the learning algorithm may be modified to allow the use of stochastic gradient descent, other methods are preferred to classify non-linearly separable datasets.

As discussed, the perceptron algorithm is very good at classifying linearly-separable data sets, but it performs poorly on data sets that do not have this property. To overcome these inherent modelling limitations, researchers developed more complex architectures by stacking together multiple perceptrons. These composite architecture are known as Neural Networks, and they are the fundamental structures of deep learning technologies. Figure 2.2 shows has a deep neural network is created by sequentially stacking perceptrons.

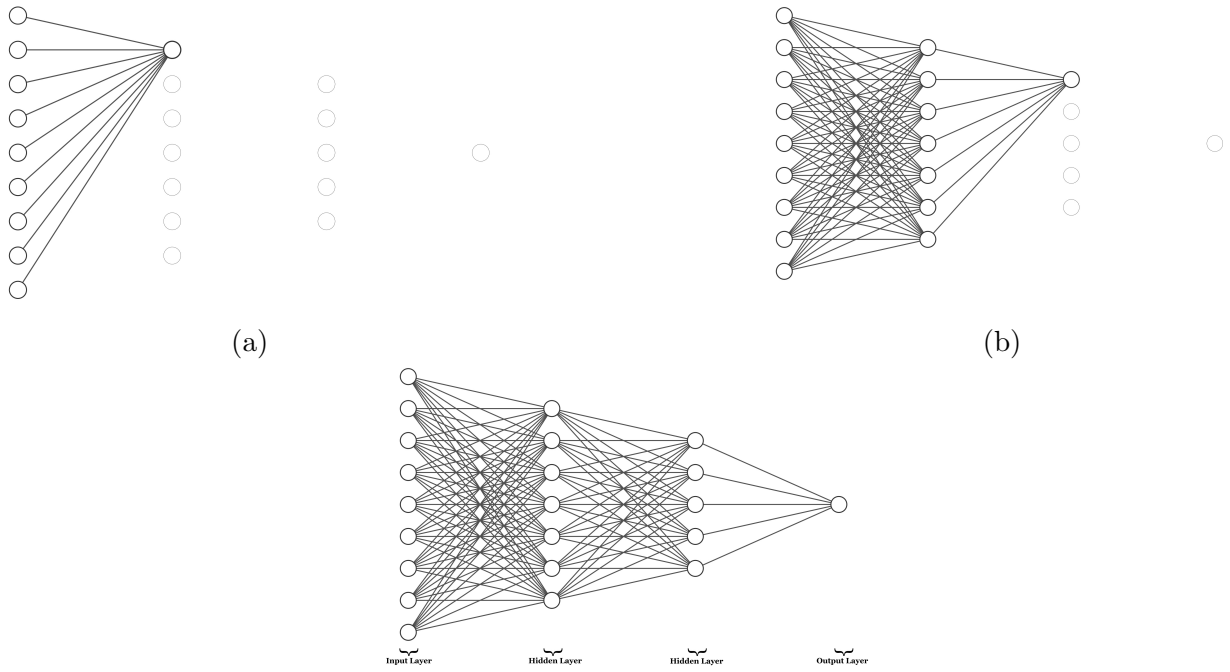


Figure 2.2: From the perceptron to a deep neural network

2.2. Deep Feedforward Networks

Deep Feedforward Neural Networks (DFNNs), also called multilayer perceptrons (MLPs), are the most popular algorithms in deep learning. These networks are called feedforward because the information flows from the input to the output layer, and there are no feedback connections in which the outputs of the model are fed back to itself. FNNs that include feedback connections are called Recurrent Neural Networks (RNNs). RNNs will not be discussed as they are rarely adopted as classification models. To simplify the terminology, from now on, the term neural networks will only refer to Feedforward Networks.

The goal of FNNs networks is to approximate a function f^*. For instance, a classifier can be represented as a function $y = f^*(\mathbf{x})$ that maps an input $\mathbf{x}$ to a category y. A feedforward network would approximate this unknown function f^* by defining a mapping $\mathbf{y} = f(\mathbf{x}; \boldsymbol{\vartheta})$ and learning the value of the parameter $\boldsymbol{\vartheta}$ that results in the best function approximation. Basically the goal is to find $\bar{\boldsymbol{\vartheta}} \in \Theta$, where Θ indicates the parameter space for $\boldsymbol{\vartheta}$, such that $f(\mathbf{x}; \bar{\boldsymbol{\vartheta}}) \approx f^*(\mathbf{x})$. During network training, we drive $f(\mathbf{x}; \boldsymbol{\vartheta})$ to mach $f^*(\mathbf{x})$ by changing the

value of $\boldsymbol{\vartheta}$ across its parameter space (Goodfellow et al., 2016).

The training examples $\mathbf{y}$ specify that at each point $\mathbf{x}$, the output layer must generate a value that is as close as possible $\mathbf{y}$. The behaviour of the last layer is somehow constrained by the training examples, whereas the behavior of the other layers is more freely specified by the learning algorithm. This algorithm orchestrates the behavior of all the other layers, called hidden layers, so that they generate the desired output, that is, a value $\hat{\boldsymbol{y}} \approx \boldsymbol{y}$. The number of hidden layers of a neural network is known as the depth of the model, while the dimensionality the hidden layers is known as the width of the model.

As mentioned at the end of section 2.1, feedforward networks were developed to overcome the inherent limitations of linear models. Altough linear model may be fit efficiently and reliably, their model capacity is limited to linear functions and they cannot recognize non-linear relationships between the input and the output variables. Linear models can be extended to represent non linear functions by fitting them to a non-linear transformation of the input variables, that is, by applying the linear model to a transformation ϕ that maps the linear input x to a non linear output $\phi(x)$. Before the advent of deep learning, the mapping ϕ was either chosen from a set of general models, or manually engineered. The drawbacks of these methodologies was that they produces a mapping that was either too generic or too domain specific. Deep learning solved these issues by introducing a relatively simple framework to automatically learn ϕ. The concept of learning the features rather then hard-coding them is called representation learning.

To showcase the idea behind representation learning, we can rewrite our model $y = f(\mathbf{x}; \boldsymbol{\vartheta})$ as $y = f(\mathbf{x}; \boldsymbol{\vartheta}, \boldsymbol{w})$, or equivalently $y = \phi(\mathbf{x}; \boldsymbol{\vartheta})^\top \boldsymbol{w}$. The parameter $\boldsymbol{\theta}$ can be used to learn ϕ from a broad class of functions, and the parameter $\mathbf{w}$ can map $\phi(\mathbf{x})$ to the output. Basically, the representation is parametrized as $\phi(\mathbf{x}; \boldsymbol{\vartheta})$ and the learning algorithm finds the $\boldsymbol{\theta}$ that corresponds to a good representation. The main advantages of representation learning are the replacement of manual feature engineering, and the possibility to use the extracted feature to perform a task. (Goodfellow et al., 2016)

Altough deep learning does not require the same level of feature engineering necessary to train traditional ML algorithms, it still shares many of the same design decisions involving the optimizer, the cost function, and the form of the output unit. Other choice, distinctive of deep learning models, involve the number of hidden units, their activation functions and the connection patterns. Many decision also have to be made with respect to the training procedure. Altough, network training relies mostly on gradient descent and back-propagation, there are many optimizers and efficient algorithms to choose from.

2.2.1. Architecture

Neural Network architecture refers to the overall structure of the computational model, that is, the number of units of a network and the connection pattern. The tradition architecture of a neural network is organized into group of units (neurons) called layers, which are usually arranged in a chain-like structure. Each layer is often defined as a function of the layer that preceded it. Starting from this representation, we can express the first layer as a function of the input layer

$$\mathbf{h}^{(1)} = \varphi^{(1)}(\mathbf{W^{(1)}}^{\top} \cdot \mathbf{x} + \mathbf{b}^{(1)})$$

the second layer is given by a function of the first layer

$$\mathbf{h}^{(2)} = \varphi^{(2)}(\mathbf{W^{(2)}}^{\top} \cdot \mathbf{h}^{(1)} + \mathbf{b}^{(2)})$$

and so on. Where $\mathbf{h}^{(i)}$ represent the activation value (output) of layer i, $\mathbf{W^{(i)}}$ represent the weight of the edges that connect layer $i-1$ to layer i, $\mathbf{x}$ represent the values of the input layer, $\mathbf{b}^{(i)}$ are the bias terms and $\varphi^{(i)}$ is the activation function of layer i (Goodfellow et al., 2016).

The main architectural considerations when building these chain-based architectures regard the width of the each layer, the number of layers and the connection pattern. Altough prior knowledge may guide the choice of these parameters, the ideal architecture must be found via experimentation.

Computational Science has discovered many interesting results regarding network struc-

ture and approximation capabilities. Specifically, the universal approximation theorem states that a FNNs that satisfies 3 conditions: (1) linearity of the output layer, (2) a number of hidden layer greater or equal to one, and (3) "squashing" activation functions, is capable of approximating any Borel measurable function from one finite-dimensional space to another with any desired non-zero amount of error, provided that the network is given enough hidden units (Hornik et al., 1989; Cybenko, 1989). An example of Borel measurable functions is given by any function on a closed and bounded set of $\mathbb{R}^n$. It is noteworthy to mention that the theorem has been proved to hold for a wide class of activation functions, including the now popular ReLU function. The approximation ability of FNNs is not restricted to Borel measurable function, but it can be extended to derivatives (Hornik et al., 1990) and to any function mapping from any finite dimensional discrete space to another (Leshno et al., 1993; Goodfellow et al., 2016).

In summary, the universal approximation theorem implies that a large enough FNNs will be able to represent any function arbitrarily well. However it is important to notice that the theorem does not specify how large this networks needs to be. Thus, we are not guaranteed that the training algorithm will be able to converge to a set parameters that leads to a good approximation. In practice, a FNNs may fail in the approximation task for two main reasons: (1) the optimization algorithm used for training may not converge to a set of parameters that corresponds to the desired function, and (2) the algorithm may choose the wrong function due to overfitting. These drawbacks enforce the importance of finding the optimal architecture through experimentation guided by monitoring the validation set error.

2.2.2. The Output Layer

One of the most important step of the design of a deep neural network involves the choice of the cost function. The cost function, also called loss function, is an important parameter that measure how well a model is performing at a given task. At a very basic level, it calculates the difference between the expected output value $\boldsymbol{y}$, and the actual value predicted by the network $\hat{\boldsymbol{y}}$. Generally speaking, a higher cost is associated with a lower model performance. During network training, the parameters $\boldsymbol{\theta}$ are updated to reduce the value of this function. Most parametric models in deep learning define a distribution $p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$ and use maximum

likelihood to define the parameter $\boldsymbol{\theta}$. In practice, the cost function used for neural network training often combines a primary cost function and a regularization term. As we will see, the choice of the primary cost function is tightly coupled with the choice of the output unit. The reason being that most of the time the loss function is simply the cross-entropy between the data distribution and the model distribution, thus, the form of the cross-entropy function is determined by the function used to represent the output.

Most NNs are trained with maximum likelihood which entails that the cost function is simply the negative log-likelihood. The negative log-likelihood can be equivalently defined as the cross-entropy between the training data and the model distribution. Thus, our cost function can be expressed as

$$J(\boldsymbol{\theta}) = -\mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim\hat{p}_{data}} \log p_{model}(\boldsymbol{y}|\boldsymbol{x})$$

The $\log p_{model}(\boldsymbol{y}|\boldsymbol{x})$ represents the model dependence of the cost function. The most beneficial aspect of this approach is that it removes the need to design the cost function for each model. In fact, by specifying a model $p(\boldsymbol{y}|\boldsymbol{x})$ we can automatically determine the cost function $\log p_{model}(\boldsymbol{y}|\boldsymbol{x})$. Another significant advantage of the maximum likelihood approach is that it prevents the gradient from becoming too small when the activation function saturate. Saturation usually occurs when the activation functions involve an exponential function that becomes very flat (saturates) whenever its argument is very negative. In this case, the log function in the negative log-likelihood undoes the exponential of some units, maintaining the gradient large and predictable enough for learning (Goodfellow et al., 2016).

In some context, we may be more interested in learning just one conditional statistic of $(\boldsymbol{y}|\boldsymbol{x})$ rather then a full probability distribution $p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$. In such cases alternative cost function may be used. The mean squared error cost function is usually implemented whenever we wish to predict the mean of $\boldsymbol{y}$ for each value of $\mathbf{x}$, whereas the mean absolute error is typically used to predict the median value of $\boldsymbol{y}$ for each value of $\boldsymbol{x}$. The MSE is usually denoted by

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim p_{data}} \|\boldsymbol{y} - f(\boldsymbol{x})\|^2$$

and the MAE by

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim p_{data}} \|\boldsymbol{y} - f(\boldsymbol{x})\|$$

Unfortunately, MSE and MAE often produce poor results when used with gradient-based optimization because their cost functions tend to produce very small gradients when saturation occurs (Goodfellow et al., 2016).

As we have discussed, the choice of the output unit determines the form of the cross-entropy function. Classification problems with two classes require the prediction of a binary variable y. Consider the problem of classifying a cell as a cancerous cell (1) or a healthy cell (2). The maximum likelihood approach imposes to define a Bernoulli distribution over y conditioned on $\boldsymbol{x}$. As the Bernoulli distribution is defined by a single parameter p, the task of the neural net is to predict $P(y = 1|\boldsymbol{x})$. Notice that to be considered as a valid probability, the predicted number must lie in the interval [0,1]. For optimization purposes, we need to ensure that the gradient of the cost function is always strong whenever the model outputs the wrong class. The most common cost function that ensures a strong gradient and a valid predicted probability, is based on a combination of the maximum likelihood and a sigmoid output unit. The resulting output unit is defined by

$$y_{pred} = \sigma(\mathbf{w}^\top \boldsymbol{h} + b)$$

where $\boldsymbol{h}$ is the set of hidden features provided by the network and defined by $\boldsymbol{h} = f(\boldsymbol{x}; \boldsymbol{\theta})$ and σ is the logistic sigmoid function given by

$$\sigma(x) = \frac{1}{1 + \exp\{-x\}}$$

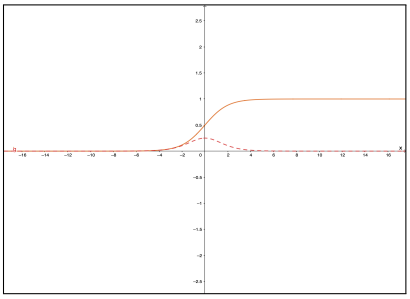
. Altough the logistic sigmoid function saturates (becomes very flat and insensitive to small changes in its input) for a very positive or a very negative x, the maximum likelihood approach overcomes the saturation by undoing the exponential. See Figure 2.3a for a graph of the sigmoid function and its derivative.

Some useful proprieties of the sigmoid function can be used to rewrite the cost function

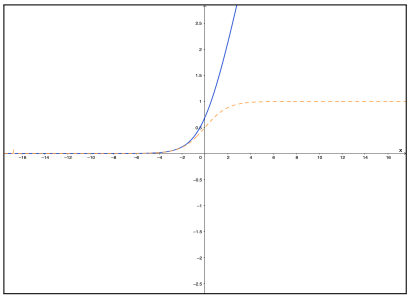
for maximum likelihood learning of a Bernoulli parametrized by a sigmoid as:

$$\begin{aligned} J(\boldsymbol{\theta}) &= -\log P(y|\mathbf{x}) \\ &= -\log \sigma((2y-1)z) \\ &= \zeta((1-2y)z) \end{aligned}$$

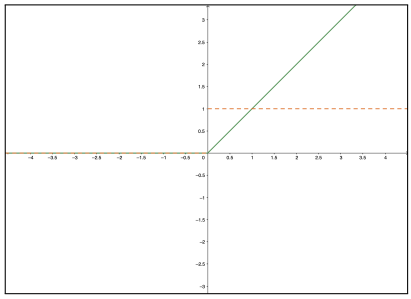
where ζ represents the softplus function and $z = \mathbf{w}^\top \boldsymbol{h} + b$. By rewriting the loss function in terms of the softplus function, it is possible to show that it saturates only when the model already has the right answer. The softplus function $\zeta(s)$ saturates only when s is small. Thus, in our case the function $\zeta((1-2y)z)$ saturates only when $((1-2y)z)$ is very negative. This can happen for two different reasons: (1) $y = 1$ and z is very positive, or (2) $y = 0$ and z is very negative. Case (1) implies that the gradient approaches 0 as $z \to \infty$. In this case $P(y = 1|x) = \sigma(z) \approx 1$ means that, by the time $z \to \infty$, the network already has the right answer. Contrarily, case (2) implies that the gradient approaches 0 as $z \to -\infty$. In this case $P(y = 0|x) = 1 - P(y = 1|x) = 1 - \sigma(z) \approx 1$. It is also noteworthy to mention that when the model makes incorrect prediction ($y = 1$ and z is negative or vice versa) the gradient does not shrink, in fact it tends to 1. This property enables the gradient-based learning to act quickly and correct a mistaken z (Goodfellow et al., 2016). See Figure 2.3b for a graph of the softplus function and its derivative.



(a) Logistic Sigmoid Function (orange) and its derivative (red)



(b) Softplus Function (blue) and its derivative (orange)



(c) Rectifier Linear Unit (green) and its derivative (orange)

Figure 2.3: The logistic sigmoid function (a) saturates for input values that are either very large or very small. For these input values the derivative becomes flat and the gradient vanishes. The softplus function (b) only saturates for very small input values. For these input values the derivative becomes flat and the gradient vanishes. The Rectifier Linear Unit (c) is the most used activation function for the hidden layers.

2.2.3. The Hidden Layers

Another important decision, that perhaps is unique to feedforward neural networks, involves the hidden units. Most of the time, the hidden units perform two operations: they compute a linear function of the input vector $\boldsymbol{x}$, defined by $\mathbf{z} = \mathbf{W}^\top \mathbf{x} + \mathbf{z}$, and then they pass $\boldsymbol{z}$ through an element-wise non-linear activation function $\varphi(\mathbf{z})$. Most hidden units differ from one to other by the type of non-linear activation function that they implement.

Among all, the Rectified Linear Unit (ReLU) is by far the most adopted. The ReLU activation function is defined by $\varphi(z) = \max\{0, z\}$. This simple function makes the computation of the hidden layer output $\boldsymbol{h}$ straightforward: $\boldsymbol{h} = \varphi(\mathbf{W}^\top \boldsymbol{x} + \boldsymbol{b}) = \max\{\mathbf{0}, \mathbf{W}^\top \boldsymbol{x} + \boldsymbol{z}\}$.
The derivative of $\varphi(z)$ is given by

$$\varphi'(z) = \begin{cases} 0, & \text{if } z < 0 \\ 1, & \text{if } z > 0 \\ \text{undefined}, & \text{if } z = 0 \end{cases}$$

.

The optimization of rectified linear units is fairly easy because they are very similar to linear units, with the only difference being that this unit outputs zero across half its domain. This function is particularly suitable for gradient-based learning because its derivative is 1 everywhere the unit is active ($z > 0$). We do not need to worry about the non-differentiability of the function at 0 because (1) it is generally acceptable to for the minima of the loss function to correspond to points with undefined gradient as we do not expect the gradient to reach 0 during training, and (2) software implementations usually ignore the non-differentiability and report one of the one-sided derivatives (Goodfellow et al., 2016). Figure 2.3c for a graph of the ReLU function and its derivative.

2.2.4. Training

Training a neural network is a straightforward process. The network accepts an input $\boldsymbol{x}$ and produces an output $\hat{y}$. Any discrepancy between the expected output y and the output predicted by the network $\hat{y}$ is used to compute a cost function $J(\boldsymbol{\theta})$. The information provided

by the cost function are used by a learning algorithm to modify the internal parameters of the network $\boldsymbol{\theta}$, also known as weights. The goal of these adjustments is to force the network into generating an output that is closer to the true output.

At a very basic level, the learning algorithm computes a particular vector, called the gradient vector, that for each weight indicates by what amount the cost function $J(\boldsymbol{\theta})$ would decrease or increase if the weight were increased by a small amount. The gradient vector of the loss function is indicated by $\nabla_\theta J(\boldsymbol{\theta})$. In the next step, the learning algorithm adjusts the weights in the opposite direction to the gradient vector. This procedure is known as gradient-based learning (LeCun et al., 2015).

2.2.4.1. Gradient-Based Optimization

Most deep learning algorithms involve the optimization of a cost function $J(\boldsymbol{\theta})$. In this case, the optimization problem can be written as $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ and refers to the task of minimizing $J(\boldsymbol{\theta})$ by changing $\boldsymbol{\theta}$.

Univariate functions can be optimized thanks to the information provided by their derivative. The derivative of a function f at a point x is really useful because it provides information on how to change x in order to increase or decrease the value of $f(x)$. When the derivative has a positive value, we can decrease the value of the function by decreasing x and when the derivative is negative we can decrease the value of the function by increasing x. The procedure by which we reduce $f(x)$ by moving x in smalls steps with opposite sign of the derivative is called gradient descent. Formally the update rule for single input functions is given by $x_{t+1} = x_t - \alpha \times f'(x)$. Where α is the step size. The gradient descent update rule stops when the derivative provides no information about which direction to move, that is, when the function reaches a stationary point (a point where $f'(x) = 0$). This point may be a global or local minimum/maximum, or a saddle point.

For multivariate function optimization we exploit the data provided by the partial derivatives. The partial derivative $\frac{\partial}{\partial x_i} f(\mathbf{x})$ measures how the function f changes with respect to an increase of the variable x_i at point $\mathbf{x}$. The gradient of f, denoted by $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$, is just a vector

containing all partial derivatives. Just as for the one-dimensional case, we can decrease the value of the function f by moving in the direction of the negative gradient. The update rule for multivariate gradient descent is given by $\boldsymbol{x_{t+1}} = \boldsymbol{x_t} - \alpha \times \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$. As for the univariate case, the update rule converges when every element of the gradient is zero.

In deep learning applications, the cost function $J(\boldsymbol{\theta})$ that we want to minimize usually contain many sub-optimal local minima and many saddle points surrounded by flat regions. This setting make optimization very difficult. As a consequence, we usually settle for a value of $J(\boldsymbol{\theta})$ that is very low, but not necessarily minimal Goodfellow et al. (2016).

2.2.4.2. Backward Propagation

As we have discussed, the gradient-based learning algorithm of a deep neural network performs two main steps. In the first step it computes the gradient of the cost function and then updates the parameter of the network in the opposite direction to the gradient vector. Unfortunately, the computation of the gradients can be very expensive. Thus, neural network rely on a simple and relatively inexpensive procedure called Backward Propagation. The back-propagation algorithm allows the information from the cost function $J(\boldsymbol{\theta})$ to flow backwards through the network. The gradient of the network is computed from top to bottom by using the chain rule of derivative, the value of the cost function and of the weight parameters (Kelleher, 2019).

2.2.4.3. Stochastic Gradient Descent

In practice, computing the gradient for the whole data set may be computationally expensive. Thus, most practitioners rely on an alternative procedure called stochastic gradient descent (SGD). This procedure differs from gradient descent because it randomly selects a small sample of the input vector and uses it to compute the outputs, the errors and the average gradient. The weights are then adjusted according to the gradient and the procedure is repeated until the average of the cost function stops decreasing. These procedure is can find a good set of weights, with a fraction of the computation required by more elaborate optimization techniques (Bottou and Bousquet, 2007).

2.2.5. Regularization

Deep learning algorithms are very powerful and they have a tendency to overfit. To prevent overfitting, a set of strategy, known as regularization, are implemented with the objective of reducing the test error, possibly at the expense of increasing the training error. In the context of deep learning, most regularization strategies are based on the bias-variance trade-off, that is, we accept to increase the bias of the model in exchange for a decrease in the variance. The main regularization strategies used in deep learning are: parameter norm penalties, early stopping, parameter sharing, bagging and dropout.

2.3. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of neural networks specifically designed to process data with a grid-like topology. CNNs differ from classical neural networks because they use a distinct kind of linear operation, called convolution, in place of general matrix multiplication in at least one of their layers. CNNs are a brilliant example of how tailoring the structure of a neural network to the specific characteristics of the data can improve the training time and the accuracy of the network.

Convolutional neural networks are the perhaps the greatest example of biologically inspired artificial intelligence, as they can be seen as representatives of Multi-Stage Hubel-Wiesel Architectures. The concept of these architectures is rooted in Hubel and Wiesel work on cat's primary visual cortex (V1), which is the first area of the brain that performs significantly advanced processing of visual inputs. The research discovered orientation-selective simple cells with local receptive fields and complex cells. Simple cells respond to oriented edges and gratings, while complex cells respond to features that are similar to those detected by simple cells but they invariant to small shifts in the position of the feature. CNNs effectively emulate the detection proprieties of the simple cells in the detector unit and the translation invariance feature of the complex cell in the pooling layers (Goodfellow et al., 2016; LeCun et al., 2010).

The first Multi-Stage Hubel-Wiesel inspired model to be simulated on a computer was Fukushima's Neocognitron for handwritten digit recognition (Fukushima, 1980). Later on, the architecture proposed by Fukushima was simplified and the back-propagation algorithm was used to train the entire system in a supervised manner (LeCun et al., 1989a,b). The approach was so successful, that in the 1990s a research group at AT&T developed a convolutional network for reading checks (Lecun et al., 1998) and by the end of the decade, the system was reading over 10% of all checks in the US. Since then, CNNs have been widely adopted in many fields such as biometric authentication, self-driving vehicles, object detection, and the detection of faces, text, pedestrians and human bodies in natural images. Despite their success, CNNs were not taken into consideration by the mainstream computer-vision and machine-learning communities. However, in 2012 deep convolutional networks were applied to the ImageNet data set, containing about a million images from the web that had to be classifies in 1,000 different classes. CNNs achieved tremendous results, with an error rate that was almost half of those of the best competing approaches (Krizhevsky et al., 2012). From that point on, CNNs have become the dominant approach for almost all detection and recognition tasks.

The most fundamental design objective of CNNs was to create a networks where the early layers would extract the local visual features, and the deeper layers would combine them to generate higher-order features. For instance, when applied to face recognition, the neurons in the early layers learn to activate in response to simple local features, such as segment of curves or lines at particular angles, neurons deeper in the network combine these simple features in higher-order features that represent body parts, such as mouth or nose, and the neurons in the final layers are able to identify faces in an image by combining the body part activations (Kelleher, 2019).

As discussed, a CNN needs to perform convolution, detection and pooling in order to detect simple features, combine them into more complex features and ensure translation invariance. These operations are performed within a special layer, called the convolution layer which consists of three stages. See figure 2.4. In the first stage (Convolution Stage), several convolutions are performed in parallel to produce a set of linear activations. In the

second stage (Detector Stage), the linear activation are passed through a non linear activation function. In the third stage, the pooling function is applied to further modify the output of the layer. A typical CNN architecture is composed of one, two or three convolutional layers, followed by a classification module.

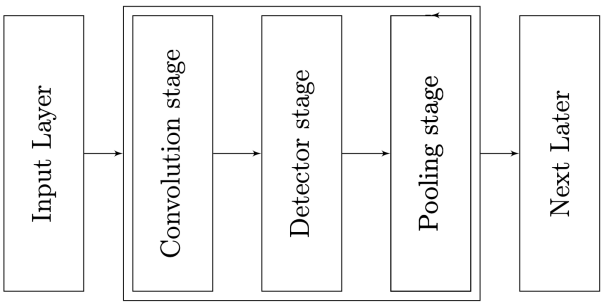


Figure 2.4: Stages of a typical Convolution Layer

2.3.1. Convolution Stage

The convolution stage is a fundamental components of the CNN architecture. It usually consists of a set of linear operations that perform feature extraction over the inputs.

The name convolution refers to a mathematical operation defined on to functions f and g that produces a third function $f * g$ which is formally defined by

$$(f * g)(t) = \int_0^{\tau} f(\tau)g(t - \tau)\, d\tau$$

. In neural network applications, the first function f is usually called input, the second function g is referred as kernel and the output function $(f * g)(t)$ as the feature map. In the context of machine learning, the input is usually a multidimensional array of data and the kernel is a multidimensional array of parameters (weights). The learning algorithm changes the parameter of the kernel. The input and the kernel arrays are often referred to as tensors. The convolution function can be adapted to discrete inputs (such as multidimensional arrays) to yield

$$(f * g)(t) = \sum_{\tau=-\infty}^{\infty} f(\tau)g(t - \tau)$$

. In deep learning context, the convolution is often computed over more than one axis at

the time. For instance, if we use a 2D image as input then we should also use a 2D kernel. Let $x[m, n]$ be a 2D image and $h[m, n]$ be a 2D kernel. The convolution of the two would produce an output y given by

$$y = (x * h)(m, n) = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x(i, j)h(m - i, n - j)$$

. The convolution function is commutative and can be rewritten as:

$$(x * h)(m, n) = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x(m - i, n - j)h(i, j)$$

. Notice that to apply the commutative property the kernel was flipped with respect to the input. It is noteworthy to mention that the vast majority of neural network libraries (e.g. PyTorch) implement a different function, called cross-correlation, which is the same as convolution but without kernel-flipping. The cross-correlation function is given by

$$(x * h)(m, n) = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x(m + i, n + j)h(i, j)$$

(Goodfellow et al., 2016).

In machine learning context, the kernel is represented by a matrix of weights, and convolution can be seen as sliding the kernel across the input image and computing, within each local region, the weighted sum of the inputs. The output of these processes is called feature map, and it can be seen as a map of all the locations in the image where the visual feature occurred. The act of convolving a kernel across an image is analogous to selecting a visual feature detector (kernel), sequentially sliding it across the image (convolution operation) and recording all the positions in the image where the visual feature was detected (feature map).

This procedure is repeated by using multiple kernels to generate an arbitrary number of feature maps. Each feature map generally represents a different characteristic of the input tensor, thus different kernels can be considered as different feature extractors. In this sense, the process of training CNNs with respect to the convolution stage consists in updating the

parameter of the kernel so that they can extract the most amount of features from the input.

2.3.2. Detector Stage

The convolution operation produces a linear output because it only performs a weighted summation of the inputs. Thus, the second stage of the convolution layer, the detector stage, is implemented to apply a non-linear function to the output of the convolution stage. Usually, the ReLU function described in section 2.2.3 is applied to each entry of the feature map resulting from the convolution. The ReLU activation function in this case simply changes all the negative values to zero.

2.3.3. Pooling Stage

The pooling stage performs a downsampling operation that reduces the dimensionality of the feature map by replacing its value at a certain location with a summary statistic of the nearby outputs. The most common form of pooling operation is max pooling. Max pooling simply reports the maximum output within a rectangular region (Kelleher, 2019).

The pooling layer introduces translation invariance to small shifts and distortions of the input. Translation invariance implies that the values of the majority of the pooled layers do not change if we translate the input by a small amount. Invariance to translation is one of the most useful properties of CNNs because we usually care more about the presence of the feature rather than its location. For instance, when determining whether an image contains a bird, we don't need to know the position of the wing with extremely high accuracy, we just need to know that there is a wing.

2.3.4. Fully Connected Layer

A CNNs is usually composed of many convolutional layer followed by a fully connected layer. This last layer maps the feature extracted by the convolution layer to the final outputs of the model. Usually, the output feature maps of the last convolution layer are firstly transformed into a vector, and then they are connected to one or more fully connected layers (as the one shown is Figure 2.2c).

2.3.5. Training

Training a CNN involves finding good kernels in the convolution layer and weights in the fully connected layers so that the loss function is minimized. The optimization of these parameters is performed with the same procedure described in section 2.2.4.

2.3.6. The Advantages of CNNs

The convolution layer leverages sparse interaction, parameter sharing and equivariant representations to reduce the number of computation required to train the network, and produce translation invariance representations.

Whenever every output unit of a network does not interact with every input unit, the network is said to have sparse interaction. In CNNs sparse interaction is a direct consequence of the use of a kernel that is smaller than the input. Storing a kernel that is smaller than the input requires less memory, thus CNNs are more memory efficient and require fewer computations. For instance, if a model has m inputs and n outputs, the matrix multiplication algorithm has $O(m \times n)$ runtime, whereas if we limit the number of connection for output to k the runtime decreases to $O(k \times n)$.

Parameter sharing, also knows as tied weights, occurs when a set of weights share on purpose the same value. In a typical CNN a single kernel is used over multiple input positions, thus the kernel is shared over many operations, and only needs to be stored and updated once. Thus, parameter sharing with a kernel of size k reduces the memory required to store the model to k parameters, improving memory and statistical efficiency.

The distinctive form of parameter sharing adopted in CNN ensures that the model is invariant to translations. Invariance to translation implies that if we move the object in an input image, its representation (the feature map created by the convolution layer) will shift by the same amount. Translation invariance is perhaps the most desirable property for a CNNs, because it ensures that the network will be able to detect and extract a feature regardless of where the feature is located in the input (Goodfellow et al., 2016).

3. Classification of Leukemic B-Lymphoblast Cells

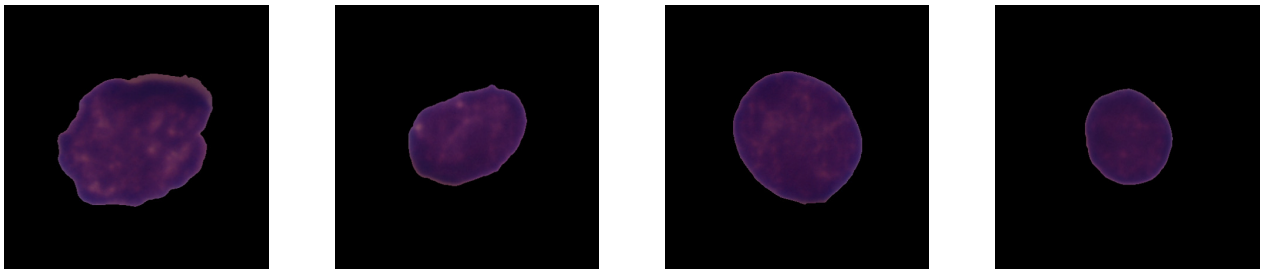
This chapter will be dedicated to the development of a convolutional neural network for the classification of Leukemic B-Lymphoblast cells from normal B-Lymphoid Precursors cells. The classification represents the last stage of the CAD system pipeline. The other steps of the pipeline, preprocessing and segmentation, will not be executed as state of the art research has already provided excellent results (Gehlot et al., 2020).

3.1. The Dataset

The networks will be trained and tested on the C_NMC_2019 Dataset. This dataset contains 15,135 cell images that have been segmented out of stain normalized blood smear images (Gupta and Gupta, 2019; Goswami et al., 2020; Clark et al., 2013). At the moment of access, the dataset was already divided into a training, a validation and a test set. As the test set came without labels, it was not used for this work. According to the dataset description, the images have been labeled by expert oncologists as either normal (HEM) or malignant (ALL) cells. The distribution of the classes is displayed in table 3.1. The task of differentiating between HEM and ALL cells is very challenging because, as shown in figure 3.1, the two cells can be very similar and subject level variability plays an important role in cellular structure.

.	Train Set	Validation Set	Total
Leukemic Cells (ALL)	7.273	1.223	8.496
Healthy Cells (HEM)	3.390	650	4.040

Table 3.1: Distribution of classes



(a) Malignant Cell (b) Healthy Cell (c) Malignant Cell (d) Healthy Cell

Figure 3.1: Leukemic B-Lymphoblast (Malignant cells) and B-Lymphoid Precursors (Healthy cells) can be morphologically very similar

3.2. Methods

To classify the images, two different approaches will be considered. For the first approach, transfer learning will be used to adapt two pretrained models to the classification task. For the second approach, a convolutional neural network will be developed and trained from scratch.

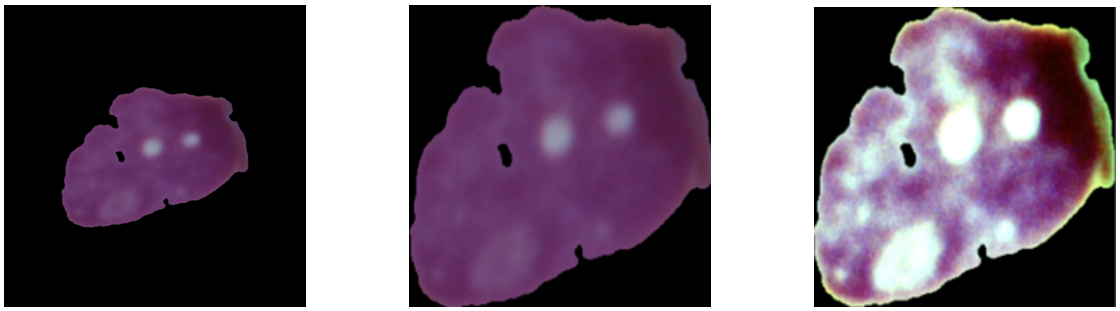
All the networks will have a single output layer. The cost function will calculate the binary cross entropy between the outputs and the targets, and it will be optimized with the Adam algorithm, which is a method for stochastic optimization (Kingma and Ba, 2014). Besides being very efficient, the Adam optimizer can perform L2 regularization by including weight decay in the algorithm. The program will be coded in python, and the deep learning models will be implemented using the PyTorch library (Paszke et al., 2019). The training and testing results will be visualized through the TensorBoard dashboard (Abadi et al., 2015). The source code has been anonymized and it is available at https://github.com/LemonHike/Thesis [Last Accessed: 10-Sept-2022].

3.2.1. Preprocessing

As mentioned in section 3.1, the dataset was already preprocessed by the authors. Despite the fact, the following transformations were performed to ensure a greater generalizing capability of the models. The validation and training data were combined and split 70-30. Successively, the images were cropped around the cell edges and normalized. The final image has a size of 224x224, and a pixel mean and standard deviation of 0 and 1 respectively. The result of the preprocessing pipeline are displayed in figure 3.2, whereas the distribution of the split is displayed in table 3.2.

.	Train Set	Validation Set
Leukemic Cells (ALL)	5.998	2.495
Healthy Cells (HEM)	2.729	1.310

Table 3.2: Distribution of classes after the 70-30 split



(a) Before Cropping (b) After Cropping (c) After Normalization

Figure 3.2: Cropping and Normalization

3.2.2. Transfer Learning

Transfer learning builds up on the idea that the knowledge learnt by a model for a given task can be adapted to another task. The term transfer learning usually refers to two distinct techniques, namely, "Fine-Tuning" and "ConvNet as feature extractor". Fine-Tuning consists in loading a network that has already been trained for a different task (ex. ImageNet classification), keep the parameters of the convolutional layers, and only train the parameters of the fully connected layers. ConvNet as feature extractor deviates from Fine-Tuning, as this technique also involves the training of the last convolution layer. For the modelling of the dataset, fine-tuning will be applied to two common architecture: ResNet and DenseNet. The two architectures have been selected because they differ quite a lot from one another.

Residual Networks (ResNet) were introduced in 2015 (He et al., 2015) with the purpose of solving the vanishing gradient problem (Hochreiter, 1998). ResNet leverages a technique called Sparse Connection to connect the activation of one layer to further layers by skipping some layers in between. Sparse connection between the layers generates Residual Blocks. The main advantage of sparse connection and residual blocks is that, if there is any layer that hurts the performance of the network, then it will be skipped by regularization. The number of layers in the ResNet model can vary largely, going from 18 up to 152. The analysis of the dataset was carried out using a ResNet architecture with 18 layers.

Just as ResNet, Densely Connected Convolutional Networks (DenseNet) were designed to solve the vanishing gradient problem, and they are characterized by a particular connectivity

patterns called Dense Connectivity. The idea behind Dense Connectivity is to set the input of every layer as the concatenation of all the previous layer outputs, that is, the input of each layer is the concatenation of the feature maps of the preceding layers. The resulting configuration strengthens feature propagation, reduces the number of parameters and alleviates the vanishing gradient problem (Huang et al., 2016). As for ResNet, various DenseNet architectures exist. The analysis of the dataset was carried out using a DenseNet with 121 layers.

3.2.2.1. Training and Results

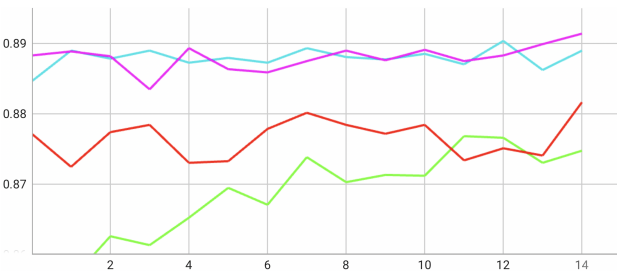
The ResNet and the DenseNet architectures were modified to host a single output unit. The fully connected layers of the architecture were retrained by optimizing a Binary Cross Entropy function with the Adam optimizer. The selection of the parameters for the optimizer (learning rate and weight decay) was carried out by grid search. The learning rate schedule was set to {0.01, 0.001}, whereas the weight decay schedule was set to {0.001, 0.0001}. The learning schedule for ResNet and DenseNet were run over 15 and 10 epochs respectively. The results for ResNet are displayed in in figure 3.3 and summarized in table 3.3. The results for DenseNet are summarized in table 3.4 and displayed in figure 3.4. The best models are distinguished by the bold text.

Epochs	Learning Rate	Weight Decay	Train Accuracy	Test Accuracy	Color Code
15	0.01	0.001	0.877	0.749	Lime
15	0.01	0.0001	0.882	0.754	Red
15	**0.001**	**0.001**	**0.890**	**0.758**	**Cyan**
15	0.001	0.0001	0.891	0.757	Magenta

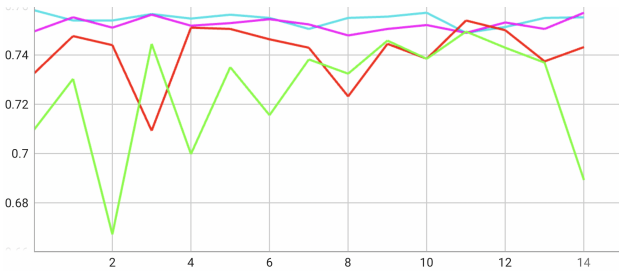
Table 3.3: ResNet grid search. The model with the highest test accuracy is in bold text.

Epochs	Learning Rate	Weight Decay	Train Accuracy	Test Accuracy	Color Code
10	0.01	0.001	0.901	0.764	Blue
10	**0.01**	**0.0001**	**0.908**	**0.774**	Green
10	0.001	0.001	0.914	0.772	Orange
10	0.001	0.0001	0.914	0.773	Black

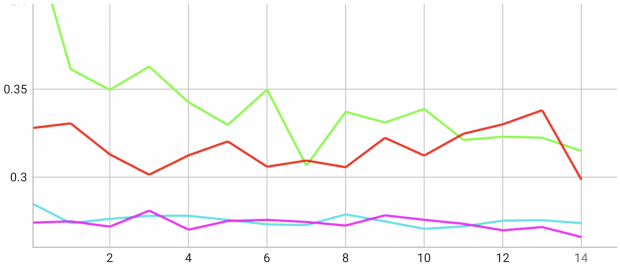
Table 3.4: DenseNet grid search. The model with the highest test accuracy is in bold text.



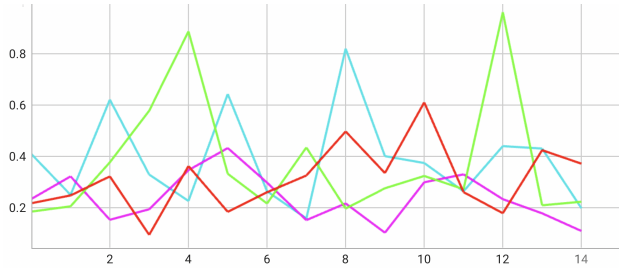
(a) Train Accuracy of ResNet



(b) Test Accuracy of ResNet

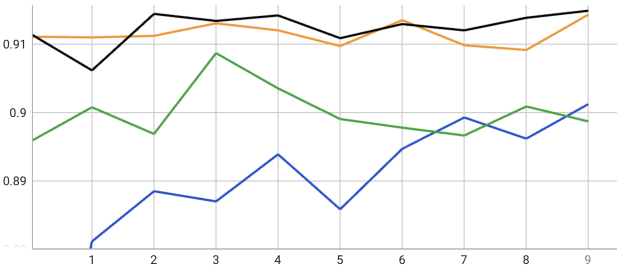


(c) Train Loss of ResNet

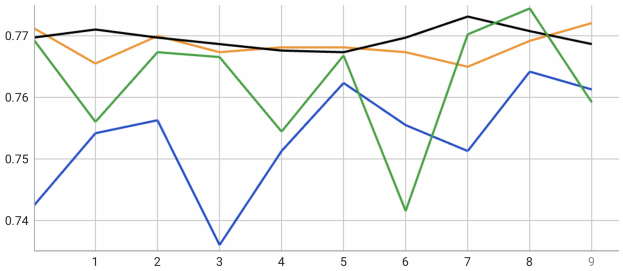


(d) Test Loss of the ResNet

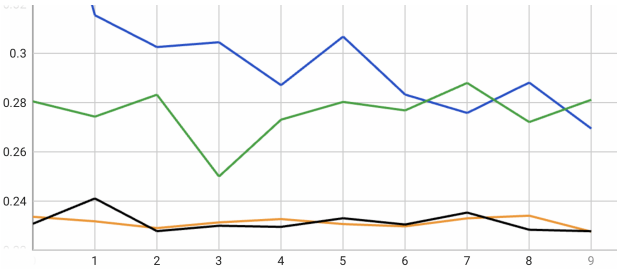
Figure 3.3: The learning metrics of the various ResNet models trained over 15 epochs.



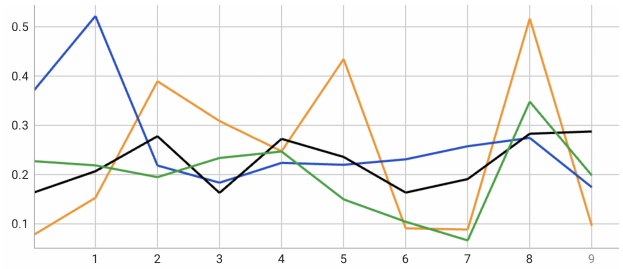
(a) Train Accuracy of DenseNet



(b) Test Accuracy of DenseNet



(c) Train Loss of DenseNet



(d) Test Loss of DenseNet

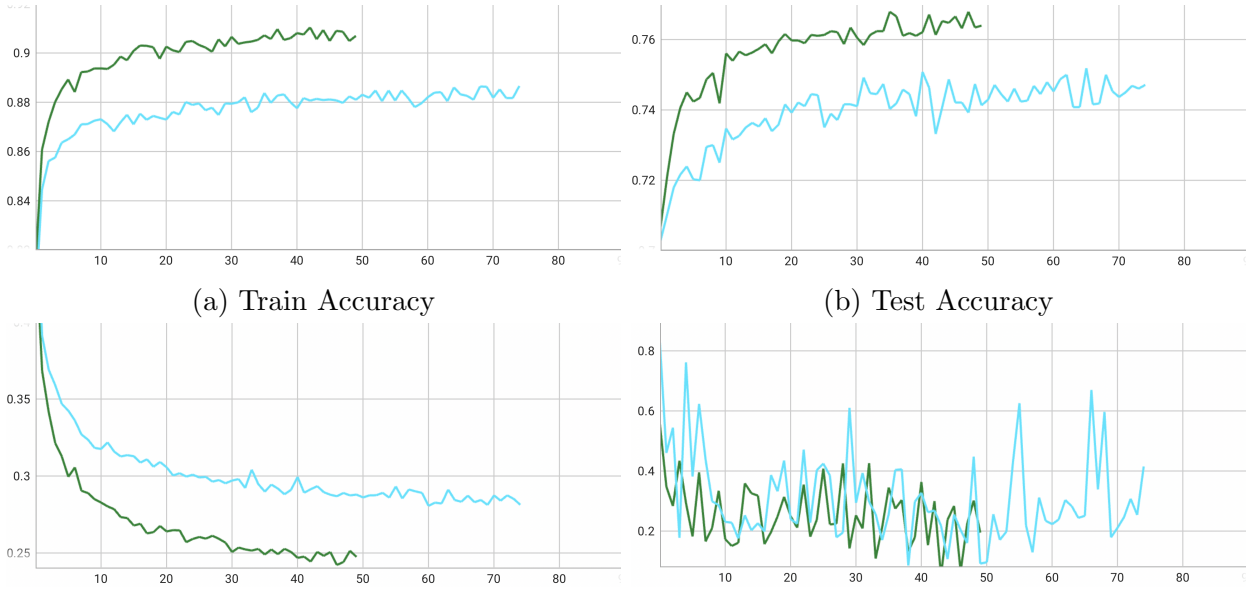
Figure 3.4: The learning metrics of the various DenseNet models trained over 10 epochs.

Out of all these models, the ones that reached the highest accuracy during grid search (indicated by the bold text) were retrained over a higher number of epochs. The final model for ResNet was trained with learning rate and weight decay of 0.001 over 75 epochs, whereas the final model for DenseNet was retrained with learning rate of 0.01 and weight decay of 0.001 over 50 epochs. The results are summarized in 3.5 and are displayed fully in figure 3.5. The best model is distinguished by the bold text. An interactive chart of the results is available at https://tensorboard.dev/experiment/92BvbOxiQ6esozy6oR397w/#scalars [Last accessed: 10-Sept-2020].

Epochs	Model	Train Accuracy	Test Accuracy	Color Code
75	ResNet	0.886	0.752	Cyan
50	**DenseNet**	**0.909**	**0.767**	**Green**

Table 3.5: DenseNet vs ResNet. The model with the highest test accuracy is in bold text.

The final results indicate that the fine-tuned DenseNet is more accurate then its counterpart, with an overall accuracy of 76.6% and a lower test loss. See figure 3.5c. Despite the partial success of DenseNet, both models failed to transfer their knowledge to the new domain as their accuracy is too low for real world applications.



(a) Train Accuracy

(b) Test Accuracy

(c) Train Loss

(d) Test Loss

Figure 3.5: The learning metrics of the best ResNet model (Green) trained over 75 epochs, and of the best DenseNet model (Cyan) trained over 50 epochs.

3.2.3. Developing an Architecture

Altough transfer learning provides a very useful tool for knowledge transmission across different domains, the success of the procedure is not always guaranteed. In fact, the knowledge embedded in the convolution layers may not be adaptable to a specific field. Therefore, feature extraction may fail whenever the difference between the knowledge built in the network and the new domain of application is large. Given that the majority of the these architectures are trained on set of images that is very different from medical microscopic images, transfer learning may fail to bring optimal results. Thus, it is necessary to build an alternative architecture whose entire parameter set can be trained without too much computational burden. For this reason, a simple architecture with three convolution layers and one fully connected layer was developed and trained on the dataset.
The convolution layers are configured as follows:

- Convolution layer 1: Convolution + Batch Normalization + ReLU + MaxPool
- Convolution layer 2: Convolution + ReLU
- Convolution layer 3: Convolution + Batch Normalization + ReLU

The input of the first convolution layer is a 224x224x3 image (3 stands for the 3 RGB channels). The first convolution layers uses a 3x3 kernel to map the input image into 12 distinct feature maps. Then, the feature maps are normalized with batch normalization, passed through a ReLU activation function, and finally pooled with MaxPool. The output of the first convolution layer consists of a set of 12 feature maps of size 112x112x3. These feature maps are the input of the second convolution layer, which outputs 20 feature maps of size 112x112x3 and feeds them into the activation function. Subsequently, the 20 feature maps reach the last convolution layer, where they are mapped into 32 feature maps, normalized and passed through the activation function. The third and last convolution layer outputs 32 feature maps of size 112x112x3. Finally, the maps are passed through a dropout layer and end up in a fully connected layer. This last layer has 112x112x3 inputs and a single output node that predicts the probability of a cell being malignant. The complete architecture is displayed in figure 3.6.

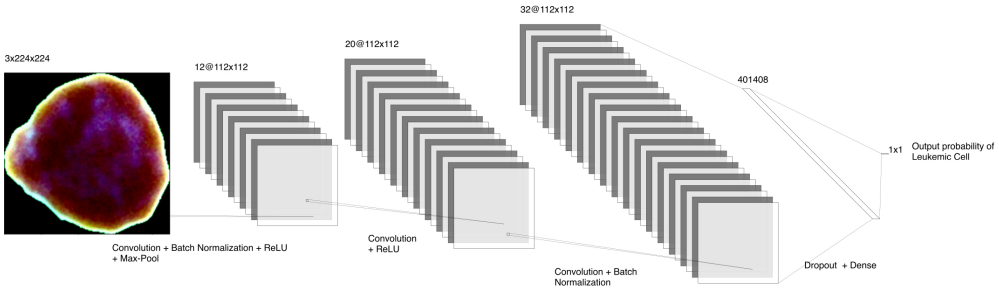


Figure 3.6: Network Architecture

3.2.3.1. Training and Results

Given the small size of the training set and the absence of implicit regularization, the network was prone to overfitting. To test the model for overfitting, a dropout layer was added before the fully connected layer, and the network was trained with dropout schedule of {0,0.5,0.75,0.95}. In this case, training with dropout and weight decay equal to zero is equivalent to training without regularization. The result of this test are summarized in figure 3.6, and show how the network, if not regularized, overfits after just 10 epochs. This particular experiment shows that overfitting is also strong when the dropout rates are 0.5 and 0.75. Thus, in order to prevent the network from memorizing the dataset, the dropout rate was set to 0.95.

Epochs	Learning Rate	Weight Decay	Dropout	Train Accuracy	Test Accuracy
10	0.001	0	0	0.995	0.745
10	0.001	0	0.5	0.981	0.753
10	0.001	0	0.75	0.972	0.759
10	**0.001**	**0**	**0.95**	**0.859**	**0.734**

Table 3.6: Testing the network for overfitting. The model with the lowest difference between train and test accuracy is in bold text.

Given the strong overfitting tendencies, L2 regularization was considered on top of dropout. A grid search was implemented to determine the optimal amount of L2 regularization and learning rate. The learning rate schedule was set to {0.001,0.001,0.01}, whereas the weight decay schedule was set to {0,0.1,0.2,0.3}. The results of the grid search are displayed in table 3.7.

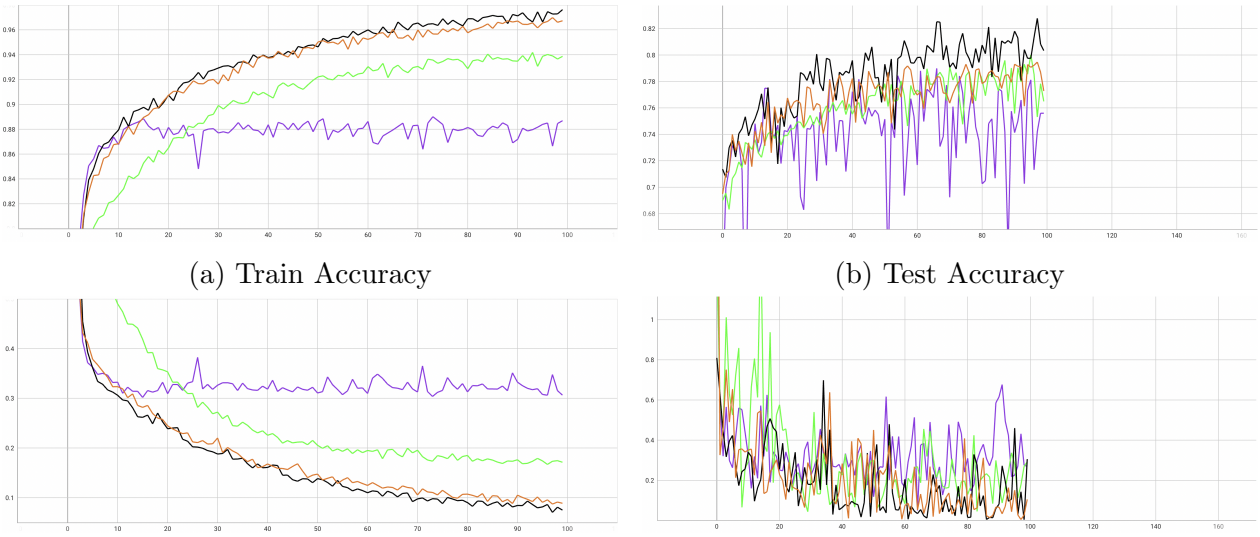
Epochs	Learning Rate	Weight Decay	Dropout	Train Accuracy	Test Accuracy
10	**0.001**	**0**	**0.95**	**0.859**	**0.734**
10	**0.001**	**0.1**	**0.95**	**0.827**	**0.736**
10	**0.001**	**0.2**	**0.95**	**0.864**	**0.747**
10	0.001	0.3	0.95	0.757	0.675
10	0.001	0.4	0.95	0.850	0.720
10	0.01	0	0.95	0.790	0.715
10	0.01	0.1	0.95	0.778	0.697
10	0.01	0.2	0.95	0.756	0.675
10	0.01	0.3	0.95	0.757	0.675
10	0.01	0.4	0.95	0.687	0.656

Table 3.7: Hyperparameters Grid Search. The three models that achieved the highest accuracies are in bold text.

The best models (bold text) were selected and trained over 100 epochs. Also, a model with weight decay equal to 0.0001 was trained to quantify the effect of very small L2 regularization over a relatively large number of iterations. The results are summarized in table 3.8 and displayed in figure 3.7. An interactive chart is available at https://tensorboard.dev/experiment/Ai7jn9TbRkKzhm6baur69A/#scalars [Last accessed: 10-Sept-2022]. Surprisingly, the model that yielded the highest performance was regularized solely by dropout. It is noteworthy to mention that adding a small L2 regularization term to the optimizer did not improve the performance of the model.

Epochs	L. Rate	W Decay	Dropout	Train Accuracy	Test Accuracy	Color Code
100	**0.001**	**0**	**0.95**	**0.967**	**0.825**	**Black**
100	0.001	0.0001	0.95	0.970	0.794	Orange
100	0.001	0.1	0.95	0.94	0.8	Lime
100	0.001	0.2	0.95	0.890	0.790	Violet

Table 3.8: Summary of the best results. The model that achieved the highest accuracy is in bold text.



(a) Train Accuracy

(b) Test Accuracy

(c) Train Loss

(d) Test Loss

Figure 3.7: Results of the Network Trained from Scratch

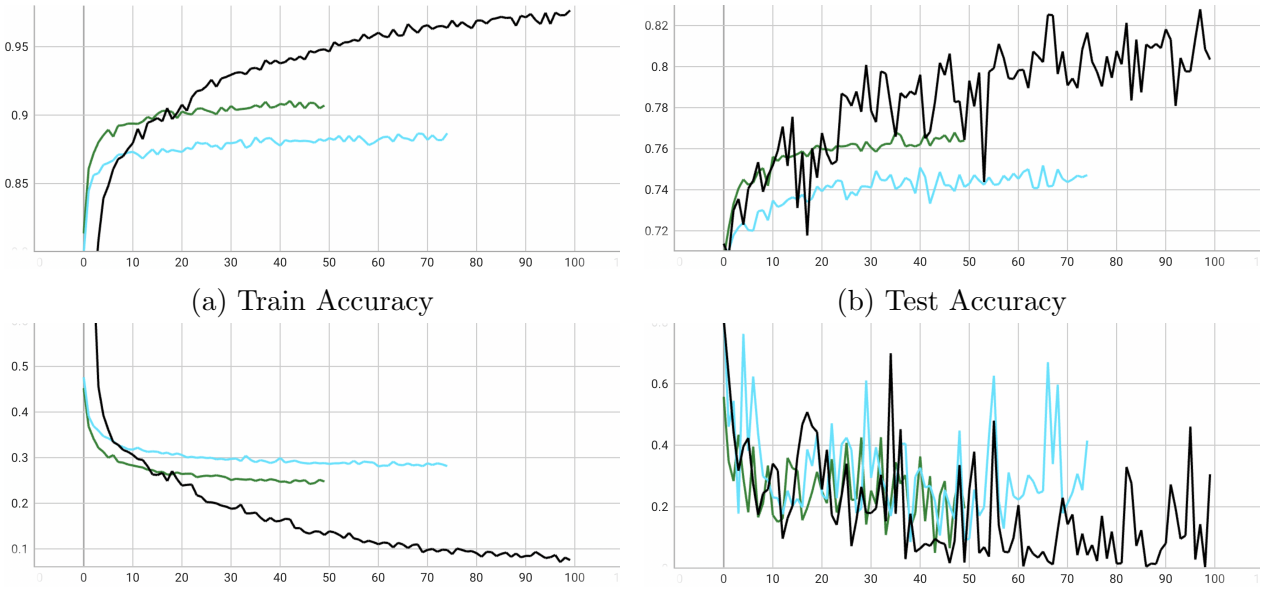
3.2.4. Cross Comparison

When comparing the results obtained by fine-tuning ResNet and DenseNet, with those obtained with the full network training, it is quite apparent that the network trained from scratch yielded better results in terms of accuracy, precision, specificity and sensitivity. See table 3.9 and figure 3.8. An interactive chart is also available at https://tensorboard.dev/experiment/pydqIVVLRPa61YBikhPkxA/#scalars [Last accessed: 10-Sept-2022].

Model	Test Accuracy	Precision	Specificity	Sesitivity	Color Code
Scratch Model	**0.825**	**87%**	**96%**	**50%**	**Black**
ResNet	0.752	69%	89%	46%	Cyan
DenseNet	0.767	79%	94%	42%	Green

Table 3.9: Cross Comparison

As mentioned in previous discussions, the reason behind the disparity in performance in most likely due to the heterogeneity between the network knowledge and the domain of application. In fact, both ResNet and DenseNet were trained on the ImageNet Database, which is significantly different from the C_NMC_2019 Dataset used for this work.



(a) Train Accuracy

(b) Test Accuracy

(c) Train Loss

(d) Test Loss

Figure 3.8: Cross Comparison. Dense Net (Green), ResNet (Cyan), Model From Scratch (Black). The model trained from scratch achieved the highest scores with regard to every performance metric.

4. Conclusion

The models developed throughout this work show great potentials in their application to computer aided diagnostic systems for the early diagnosis of B-Cell Acute Lymphoblastic Leukemia. The best performing configuration was able to distinguish between Leukemic Blasts and Normal B-Lymphoid Precursors cells with 82% accuracy, 96% specificity and 87% precision. Unfortunately, none of the models was capable of developing a high sensitivity towards the disease. As a matter of fact, the best performing model was only able to achieve a sensitivity of 50%, which is extremely low and does not match the standards required for real world applications. The ineptitude towards sensitivity indicates that the model has a very strong tendency to mistake healthy cells for cancerous cells, resulting in a very high number of false negatives.

There may be many reasons behind the bias of the model towards cancerous cells. Firstly, the small sample of images on which the network was trained and tested probably undermined its ability to extract and generalize the feature necessary to generate a good classifier. Secondly, the unbalanced sample probably contributed to the network bias towards leukemic cells. Thirdly, the likeness between leukemic blasts and lymphoid precursors, coupled with a possibly inaccurate ground truth, may have predisposed the network to classify the cells as cancerous. Several solutions could be implemented to abate the aforementioned network bias. For instance, data augmentation techniques could be used to increase the variability of the sample and enhance the generalizing ability of the network. Moreover, as General Adversarial Networks (GAN) demonstrate potential in synthesizing medical images (Skandarani et al., 2021), they could be deployed as an advanced data augmentation technique to fabricate artificial images and increase the number of training samples.

A few important remarks can also be made with regards to the low performance achieved by the pre-trained neural networks. Their unsatisfactory results suggests that these models failed to adapt their prior knowledge to the new task. The lack of success of this approach highlights the necessity to develop large and complex neural networks that are able to handle medical image data and that can easily transfer their knowledge across the medical domain.

Furthermore, as the research is shifting from the development of new architectures to the application of already established configurations to new problems, it is fundamental to create very large medical databases on which these pre-existing networks can be trained and tested.

Altough the deep learning models presented in this work failed to meet the standards necessary for real world application, the potentiality of these tools as diagnosis systems cannot be overlooked. As the field of Deep Learning-Based CAD systems is at its early stages of development, a few unsatisfactory results should not discourage the use of these technologies in the clinical settings. Contrarily, the careful examination of the causes behind these inadequate performances could provide useful guidelines to improve future research and implementation.

Hopefully, this dissertation was able to showcase the potential of deep learning technologies applied to the medical domain, and also provide a few, but possibly useful, recommendations for future work.

Bibliography

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org, [Last Accessed: 29-Aug-2022].

Charu C. Aggarwal. *Neural Networks and Deep Learning*. Springer, Cham, 2018. ISBN 978-3-319-94462-3. doi: 10.1007/978-3-319-94463-0.

Mohammed A Al-Masni, Mugahed A Al-Antari, Jeong-Min Park, Geon Gi, Tae-Yeon Kim, Patricio Rivera, Edwin Valarezo, Mun-Taek Choi, Seung-Moo Han, and Tae-Seong Kim. Simultaneous detection and classification of breast masses in digital mammograms via a deep learning YOLO-based CAD system. *Comput. Methods Programs Biomed.*, 157:85–94, April 2018. URL https://pubmed.ncbi.nlm.nih.gov/29477437/. [Last Accessed: 27-July-2022].

American Cancer Society. Cancer facts & figures 2022. Technical report, Atlanta: American Cancer Society, 2022. Available from https://www.cancer.org/content/dam/cancer-org/research/cancer-facts-and-statistics/annual-cancer-facts-and-figures/2022/2022-cancer-facts-and-figures.pdf [Last Accessed: 11-Aug-2022].

Jordan A. Baeker Bispo, Paulo S. Pinheiro, and Erin K. Kobetz. Epidemiology and etiology of leukemia and lymphoma. *Cold Spring Harbor Perspectives in Medicine*, 10(6), 2020. doi: 10.1101/cshperspect.a034819. URL http://perspectivesinmedicine.cshlp.org/content/10/6/a034819.abstract. [Last Accessed: 16-July-2022].

Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*,

volume 20. Curran Associates, Inc., 2007. URL https://proceedings.neurips.cc/paper/2007/file/0d3180d672e08b4c5312dcdafdf6ef36-Paper.pdf. [Last Accessed: 27-Aug-2022].

Kenneth Clark, Bruce Vendt, Kirk Smith, John Freymann, Justin Kirby, Paul Koppel, Stephen Moore, Stanley Phillips, David Maffitt, Michael Pringle, Lawrence Tarbox, and Fred Prior. The cancer imaging archive (tcia): Maintaining and operating a public information repository. *Journal of Digital Imaging*, 26(6):1045–1057, Dec 2013. ISSN 1618-727X. doi: 10.1007/s10278-013-9622-7. URL https://doi.org/10.1007/s10278-013-9622-7. [Last Accessed: 11-Aug-2022].

George V. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989. URL https://www.semanticscholar.org/paper/Approximation-by-superpositions-of-a-sigmoidal-Cybenko/8da1dda34ecc96263102181448c94ec7d645d085. [Last Accessed: 25-Aug-2022].

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de las Casas, Craig Donner, Leslie Fritz, Cristian Galperti, Andrea Huber, James Keeling, Maria Tsimpoukelli, Jackie Kay, Antoine Merle, Jean-Marc Moret, Seb Noury, Federico Pesamosca, David Pfau, Olivier Sauter, Cristian Sommariva, Stefano Coda, Basil Duval, Ambrogio Fasoli, Pushmeet Kohli, Koray Kavukcuoglu, Demis Hassabis, and Martin Riedmiller. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, Feb 2022. ISSN 1476-4687. doi: 10.1038/s41586-021-04301-9. URL https://doi.org/10.1038/s41586-021-04301-9. [Last Accessed: 10-Aug-2022].

Kunio Doi. Computer-aided diagnosis in medical imaging: historical review, current status and future potential. *Comput. Med. Imaging Graph.*, 31(4-5):198–211, June 2007. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1955762/. [Last Accessed: 20-July-2022].

Ying Dong, Oumin Shi, Quanxiang Zeng, Xiaoqin Lu, Wei Wang, Yong Li, and Qi Wang. Leukemia incidence trends at the global, regional, and national level between 1990 and 2017. *Exp. Hematol. Oncol.*, 9(1):14, June 2020. URL https://ehoonline.biomedcentral.com/articles/10.1186/s40164-020-00170-6. [Last Accessed: 12-July-2022].

Babak Ehteshami Bejnordi, Mitko Veta, Paul Johannes van Diest, Bram van Ginneken, Nico Karssemeijer, Geert Litjens, Jeroen A W M van der Laak, the CAMELYON16 Consortium, Meyke Hermsen, Quirine F Manson, Maschenka Balkenhol, Oscar Geessink, Nikolaos Stathonikos, Marcory Crf van Dijk, Peter Bult, Francisco Beca, Andrew H Beck, Dayong Wang, Aditya Khosla, Rishab Gargeya, Humayun Irshad, Aoxiao Zhong, Qi Dou, Quanzheng Li, Hao Chen, Huang-Jing Lin, Pheng-Ann Heng, Christian Haß, Elia Bruni, Quincy Wong, Ugur Halici, Mustafa Ümit Öner, Rengul Cetin-Atalay, Matt Berseth, Vitali Khvatkov, Alexei Vylegzhanin, Oren Kraus, Muhammad Shaban, Nasir Rajpoot, Ruqayya Awan, Korsuk Sirinukunwattana, Talha Qaiser, Yee-Wah Tsang, David Tellez, Jonas Annuscheit, Peter Hufnagl, Mira Valkonen, Kimmo Kartasalo, Leena Latonen, Pekka Ruusuvuori, Kaisa Liimatainen, Shadi Albarqouni, Bharti Mungal, Ami George, Stefanie Demirci, Nassir Navab, Seiryo Watanabe, Shigeto Seno, Yoichi Takenaka, Hideo Matsuda, Hady Ahmady Phoulady, Vassili Kovalev, Alexander Kalinovsky, Vitali Liauchuk, Gloria Bueno, M Milagro Fernandez-Carrobles, Ismael Serrano, Oscar Deniz, Daniel Racoceanu, and Rui Venâncio. Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer. *JAMA*, 318(22):2199–2210, December 2017. URL https://pubmed.ncbi.nlm.nih.gov/29234806/. [Last Accessed: 23-July-2022].

Enas M. F. El Houby. Framework of computer aided diagnosis systems for cancer classification based on medical images. *Journal of Medical Systems*, 42(8):157, Jul 2018. ISSN 1573-689X. doi: 10.1007/s10916-018-1010-x. URL https://doi.org/10.1007/s10916-018-1010-x. [Last Accessed: 15-July-2022].

Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, February 2017. URL https://pubmed.ncbi.nlm.nih.gov/28117445/. [Last Accessed: 23-July-2022].

Jacques Ferlay, Murielle Colombet, Isabelle Soerjomataram, Donald M Parkin, Marion Piñeros, Ariana Znaor, and Freddie Bray. Cancer statistics for the year 2020: An overview. *Int. J. Cancer*, 149(4):778–789, April 2021. URL https://pubmed.ncbi.nlm.nih.gov/33818764/. [Last Accessed: 17-Aug-2022].

Joaquín Figueroa Barraza, Enrique López Droguett, and Marcelo Ramos Martins. Towards interpretable deep learning: A feature selection framework for prognostics and health management using deep neural networks. *Sensors (Basel)*, 21(17):5888, September 2021. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8433983/. [Last Accessed: 27-July-2022].

K Fukushima. Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybern.*, 36(4):193–202, 1980. URL https://www.rctn.org/bruno/public/papers/Fukushima1980.pdf. [Last Accessed: 20-Aug-2022].

Yiming Gao, Krzysztof J Geras, Alana A Lewin, and Linda Moy. New frontiers: An update on computer-aided diagnosis for breast imaging in the age of artificial intelligence. *AJR Am. J. Roentgenol.*, 212(2):300–307, February 2019. URL https://pubmed.ncbi.nlm.nih.gov/30667309/. [Last Accessed: 17-July-2022].

Shiv Gehlot, Anubha Gupta, and Ritu Gupta. Sdct-auxnet: Dct augmented stain deconvolutional cnn with auxiliary classifier for cancer diagnosis. *Medical Image Analysis*, 61: 101661, 2020. ISSN 1361-8415. doi: https://doi.org/10.1016/j.media.2020.101661. URL https://www.sciencedirect.com/science/article/pii/S136184152030027X. [Last Accessed: 11-Aug-2022].

Fiona J Gilbert, Susan M Astley, Magnus A McGee, Maureen G C Gillan, Caroline R M Boggis, Pamela M Griffiths, and Stephen W Duffy. Single reading with computer-aided detection and double reading of screening mammograms in the united kingdom national breast screening program. *Radiology*, 241(1):47–53, October 2006. URL https://pubmed.ncbi.nlm.nih.gov/16990670/. [Last Accessed: 30-June-2022].

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Shubham Goswami, Suril Mehta, Dhruva Sahrawat, Anubha Gupta, and Ritu Gupta. Heterogeneity loss to handle intersubject and intrasubject variability in cancer. *CoRR*, abs/2003.03295, 2020. URL https://arxiv.org/abs/2003.03295. [Last Accessed: 11-Aug-2022].

Anubha Gupta and Ritu Gupta. All challenge dataset of isbi 2019 [data set]. The Cancer Imaging Archive. Available from https://doi.org/10.7937/tcia.2019.dc64i46r [Last Accessed: 11-Aug-2022], 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385. [Last Accessed: 27-Aug-2022].

Emilie L Henriksen, Jonathan F Carlsen, Ilse Mm Vejborg, Michael B Nielsen, and Carsten A Lauridsen. The efficacy of using computer-aided detection (CAD) for detection of breast cancer in mammography screening: a systematic review. *Acta Radiol.*, 60(1):13–18, January 2019. URL https://pubmed.ncbi.nlm.nih.gov/29665706/. [Last Accessed: 10-July-2022].

Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6:107–116, 04 1998. doi: 10.1142/S0218488598000094. URL https://www.worldscientific.com/doi/abs/10.1142/S0218488598000094. [Last Accessed: 30-Aug-2022].

Michael Holinstat. Normal platelet function. *Cancer Metastasis Rev.*, 36(2):195–198, June 2017. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5709181/. [Last Accessed: 29-July-2022].

Kurt Hornik, Maxwell B. Stinchcombe, and Halbert L. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989. URL https://www.cs.cmu.edu/~epxing/Class/10715/reading/Kornick_et_al.pdf. [Last Accessed: 18-Aug-2022].

Kurt Hornik, Maxwell B. Stinchcombe, and Halbert L. White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3:551–560, 1990. URL https://www.semanticscholar.org/paper/Universal-approximation-of-an-unknown-mapping-and-Hornik-Stinchcombe/37807e97c624fb846df7e559553b32539ba2ea5d. [Last Accessed: 20-Aug-2022].

Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016. URL http://arxiv.org/abs/1608.06993. [Last Accessed: 27-Aug-2022].

Zhimin Huo, Ronald M Summers, Sophie Paquerault, Joseph Lo, Jeffrey Hoffmeister, Samuel G Armato, 3rd, Matthew T Freedman, Jesse Lin, Shih-Chung Ben Lo, Nicholas Petrick, Berkman Sahiner, David Fryd, Hiroyuki Yoshida, and Heang-Ping Chan. Quality assurance and training procedures for computer-aided detection and diagnosis systems in clinical use. *Med. Phys.*, 40(7):077001, July 2013. URL https://pubmed.ncbi.nlm.nih.gov/23822459/. [Last Accessed: 17-July-2022].

Ferlay J, Ervik M, Lam F, Colombet M, Mery L, Piñeros M, Znaor A, Soerjomataram I, and Bray F. Global cancer observatory: Cancer today. Lyon, France: International Agency for Research on Cancer. Available from https://gco.iarc.fr/today, 2020. [Last Accessed: 11-Aug-2022].

Kenneth Kaushansky, Marshall Al Lichtman, Josef Prchal, Marcel M Levi, Oliver W. Press, Linda J Burns, and Michael A Caligiuri. *Williams Hematology, 9E.* McGraw-Hill Professional, New York, NY, 9 edition, December 2015.

John D Keen, Joanna M Keen, and James E Keen. Utilization of computer-aided detection for digital screening mammography in the united states, 2008 to 2016. *J. Am. Coll. Radiol.*, 15(1 Pt A):44–48, January 2018. URL https://pubmed.ncbi.nlm.nih.gov/28993109/. [Last Accessed: 7-July-2022].

J.D. Kelleher. *Deep Learning.* The MIT Press Essential Knowledge series. MIT Press, 2019. ISBN 9780262537551. URL https://books.google.it/books?id=b06qDwAAQBAJ.

Eun-Kyung Kim, Hyo-Eun Kim, Kyunghwa Han, Bong Joo Kang, Yu-Mee Sohn, Ok Hee Woo, and Chan Wha Lee. Applying data-driven imaging biomarker in mammography for breast cancer screening: Preliminary study. *Sci. Rep.*, 8(1):2762, February 2018. URL https://www.nature.com/articles/s41598-018-21215-1. [Last Accessed: 29-July-2022].

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International*

Conference on Learning Representations, 12 2014. URL https://arxiv.org/abs/1412.6980. [Last Accessed: 25-Aug-2022].

Thijs Kooi, Geert Litjens, Bram van Ginneken, Albert Gubern-Mérida, Clara I Sánchez, Ritse Mann, Ard den Heeten, and Nico Karssemeijer. Large scale deep learning for computer aided detection of mammographic lesions. *Med. Image Anal.*, 35:303–312, January 2017. URL [https://pubmed.ncbi.nlm.nih.gov/27497072/]. [Last Accessed: 13-Aug-2022].

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf. [Last Accessed: 18-Aug-2022].

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989a. doi: 10.1162/neco.1989.1.4.541. URL https://ieeexplore.ieee.org/document/6795724. [Last Accessed: 17-Aug-2022].

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791. URL https://ieeexplore.ieee.org/document/726791. [Last Accessed: 20-Aug-2022].

Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, R. Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989b. URL https://proceedings.neurips.cc/paper/1989/file/53c3bce66e43be4f209556518c2fcb54-Paper.pdf. [Last Accessed: 18-Aug-2022].

Yann LeCun, Koray Kavukcuoglu, and Clement Farabet. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 253–256, 2010. doi: 10.1109/ISCAS.2010.5537907. URL [https://koray.kavukcuoglu.org/publis/lecun-iscas-10.pdf]. [Last Accessed: 18-Aug-2022].

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, May 2015. URL https://www.researchgate.net/publication/277411157_Deep_Learning. [Last Accessed: 25-Aug-2022].

Constance D Lehman, Robert D Wellman, Diana S M Buist, Karla Kerlikowske, Anna N A Tosteson, Diana L Miglioretti, and Breast Cancer Surveillance Consortium. Diagnostic accuracy of digital screening mammography with and without computer-aided detection. *JAMA Intern. Med.*, 175(11):1828–1837, November 2015. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4836172/. [Last Accessed: 8-July-2022].

Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a non-polynomial activation function can approximate any function. *New York University Stern School of Business Research Paper Series*, 1993. URL https://www.sciencedirect.com/science/article/abs/pii/S0893608005801315. [Last Accessed: 25-Aug-2022].

Shih-Chung B. Lo, Jyh-Shyan Lin, Matthew T. Freedman, and Seong K. Mun. Computer-assisted diagnosis of lung nodule detection using artificial convoultion neural network. In Murray H. Loew, editor, *Medical Imaging 1993: Image Processing*, volume 1898 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 859–869, September 1993. doi: 10.1117/12.154572. URL https://ui.adsabs.harvard.edu/abs/1993SPIE.1898..859L. [Last Accessed: 21-July-2022].

National Cancer Institute. Surveillance, epidemiology, and end results (seer) program. seer*stat database: Incidence - seer research data, 22 registries (2000-2019). Technical report, National Cancer Institute Division of Cancer Control and Population Sciences, 2022a. Released April 2022, based on the November 2021 submission. Available from https://seer.cancer.gov/statistics-network/explorer/application.html [Last Accessed: 11-Aug-2022].

National Cancer Institute. Surveillance, epidemiology, and end results (seer) program. seer*stat database: Incidence - seer research data, 8 registries (1979-2019). Technical report, National Cancer Institute Division of Cancer Control and Population Sciences, 2022b. Released April 2022, based on the November 2021 submission. Available from

https://seer.cancer.gov/statistics-network/explorer/application.html [Last Accessed: 11-Aug-2022].

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf. [Last Accessed: 19-Aug-2022].

Nicholas Petrick, Berkman Sahiner, Samuel G Armato, 3rd, Alberto Bert, Loredana Correale, Silvia Delsanto, Matthew T Freedman, David Fryd, David Gur, Lubomir Hadjiiski, Zhimin Huo, Yulei Jiang, Lia Morra, Sophie Paquerault, Vikas Raykar, Frank Samuelson, Ronald M Summers, Georgia Tourassi, Hiroyuki Yoshida, Bin Zheng, Chuan Zhou, and Heang-Ping Chan. Evaluation of computer-aided detection and diagnosis systems. *Med. Phys.*, 40(8):087001, August 2013. URL https://pubmed.ncbi.nlm.nih.gov/23927365/. [Last Accessed: 10-Aug-2022].

A. Rad and M. Häggström. Simplified hematopoiesis, July 2009. CC-BY-SA 3.0 license. Available at https://commons.wikimedia.org/wiki/File:Hematopoiesis_simple.svg [Last Accessed 10-Sept-2022].

M. A. Rieger and T. Schroeder. Hematopoiesis. *Cold Spring Harb Perspect Biol*, 4(12), Dec 2012. URL https://pubmed.ncbi.nlm.nih.gov/23209149/. [Last Accessed: 23-July-2022].

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. ISSN 0033-295X. doi: 10.1037/h0042519. URL http://dx.doi.org/10.1037/h0042519. [Last Accessed: 18-Aug-2022].

Sarmad Shafique and Samabia Tehsin. Computer-aided diagnosis of acute lymphoblastic leukaemia. *Computational and Mathematical Methods in Medicine*, 2018:6125289, Feb

2018. ISSN 1748-670X. doi: 10.1155/2018/6125289. URL https://doi.org/10.1155/2018/6125289. [Last Accessed: 15-July-2022].

Rebecca L. Siegel, Kimberly D. Miller, Hannah E. Fuchs, and Ahmedin Jemal. Cancer statistics, 2022. *CA: A Cancer Journal for Clinicians*, 72(1):7–33, 2022. doi: https://doi.org/10.3322/caac.21708. URL https://acsjournals.onlinelibrary.wiley.com/doi/abs/10.3322/caac.21708. [Last Accessed: 15-July-2022].

Youssef Skandarani, Pierre-Marc Jodoin, and Alain Lalande. Gans for medical image synthesis: An empirical study, 2021. URL https://arxiv.org/abs/2105.05318. [Last Accessed: 2-Sept-2022].

Hyuna Sung, Jacques Ferlay, Rebecca L Siegel, Mathieu Laversanne, Isabelle Soerjomataram, Ahmedin Jemal, and Freddie Bray. Global cancer statistics 2020: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries. *CA Cancer J. Clin.*, 71(3):209–249, May 2021. URL https://pubmed.ncbi.nlm.nih.gov/33538338/. [Last Accessed: 9-Aug-2022].

Stephen Waite, Jinel Scott, Brian Gale, Travis Fuchs, Srinivas Kolla, and Deborah Reede. Interpretive error in radiology. *AJR Am. J. Roentgenol.*, 208(4):739–749, April 2017. URL https://pubmed.ncbi.nlm.nih.gov/28026210/. [Last Accessed: 13-July-2022].

Jian Wu, Wanli Liu, Chen Li, Tao Jiang, Islam Mohammad Shariful, Hongzan Sun, Xiaoqi Li, Xintong Li, Xinyu Huang, and Marcin Grzegorzek. A state-of-the-art survey of u-net in microscopic image analysis: from simple usage to structure mortification, 2022. URL https://arxiv.org/abs/2202.06465. [Last Accessed: 20-July-2022].

Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9(4): 611–629, Aug 2018. ISSN 1869-4101. doi: 10.1007/s13244-018-0639-9. URL https://doi.org/10.1007/s13244-018-0639-9. [Last Accessed: 23-July-2022].

Tsila Zuckerman and Jacob M Rowe. Pathogenesis and prognostication in acute lymphoblastic leukemia. *F1000Prime Rep.*, 6:59, July 2014. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4108947/. [Last Accessed: 16-July-2022].