Heaven's Light is Our Guide

Department of Computer Science & Engineering

Rajshahi University of Engineering & Technology, Bangladesh

Leukaemia Detection Using MobileNet V2 Model

Author

Md. Mehedi Hasan Jibon

Roll No. 1703158

Department of Computer Science & Engineering

Rajshahi University of Engineering & Technology

Supervised by

Rizoan Toufiq

Assistant Professor

Department of Computer Science & Engineering

Rajshahi University of Engineering & Technology

ACKNOWLEDGEMENT

First of all, I would like to thank almighty Allah, for his grace and blessings as well as for providing me with the diligence and enthusiasm along the way to accomplishing my thesis work.

I want to express my sincere gratitude, admiration, and heartfelt appreciation to my supervisor **Rizoan Toufiq**, Assistant Professor, Department of Computer Science & Engineering, Rajshahi University of Engineering & Technology, Rajshahi. Throughout the year, he has not only provided me with the technical instructions and documentation to complete the work, but he has also continuously encouraged me, offered me advice, assisted me, and cooperated sympathetically whenever he deemed necessary. His constant support was the most successful tool that helped me to achieve my result. Whenever I was stuck in any complex problems or situation he was there for me at any time of the day. Without his sincere care, this work not has materialized in the final form that it is now at the present.

I would also like to express my sincere appreciation & deepest sense of gratitude to my honorable teacher **Mohiuddin Ahmed**, Lecturer, Department of Computer Science & Engineering, Rajshahi University of Engineering & Technology, Rajshahi, for his scholastic guidance and constant support regarding this research.

I am also grateful to respected , Head of the Department of Computer Science & Engineering, and all the respective teachers of Department of Computer Science & Engineering, Rajshahi University of Engineering & Technology, Rajshahi for their valuable suggestions and inspirations from time to time.

Finally, I would like to convey my thanks to my parents, friends, and well-wishers for their true motivations and many helpful aids throughout this work.

September 2, 2023 Md. Mehedi Hasan Jibon

RUET, Rajshahi

Heaven's Light is Our Guide

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Rajshahi University of Engineering & Technology, Bangladesh

CERTIFICATE

*This is to certify that this thesis report entitled **"Leukaemia Detection Using MobileNet V2 Model"** submitted by **Md. Mehedi Hasan Jibon, Roll:1703158** and in partial fulfillment of the requirement for the award of the degree of Bachelor of Science in Department of Computer Science & Engineering of Rajshahi University of Engineering & Technology, Bangladesh is a record of the candidate own work carried out by him under my supervision. This thesis has not been submitted for the award of any other degree.*

Supervisor	External Examiner
Rizoan Toufiq	**Mohiuddin Ahmed**
Assistant Professor	Lecturer
Department of Computer Science & Engineering	Department of Computer Science & Engineering
Rajshahi University of Engineering & Technology	Rajshahi University of Engineering & Technology
Rajshahi-6204	Rajshahi-6204

ABSTRACT

Cancer of blood, more specifically known as Leukemia, is a deadly disease that is responsible for the abnormal proliferation of immature white blood cells in bone marrow. An automated and reliable diagnostic method is necessary for this problem's early detection and treatment. In short we need an automated deep learning-based strategy for separating immature leukemic blasts from healthy cells. This study presents a comprehensive study on the application of the MobileNet V2 model for leukemia detection, on the C-NMC 2019 dataset. The study includes a thorough comparative analysis, benchmarking the MobileNet V2 model against established architectures such as VGG19, InceptionNet V3, and ResNet101V2. Our investigation reveals that the MobileNet V2 model excels in its performance, achieving an accuracy of 82.75% and an F1-score of 82.48% in accurately identifying leukemia cases from medical images. In comparison, VGG19 achieves an accuracy of 77.40% with an F1-score of 76.37%, InceptionNet V3 achieves an accuracy of 80.24% with an F1-score of 79.96%, and ResNet101v2 demonstrates an accuracy of 77.18% with an F1-score of 76.19%. The utilization of the C-NMC 2019 dataset further validates the effectiveness of the MobileNet V2 model in leukemia detection. This study underscores the potential of the MobileNet V2 architecture as a robust tool for precise leukemia detection, thereby offering medical practitioners an invaluable asset in the diagnosis and management of this complex disease. MobileNetV2 is designed to be computationally efficient and lightweight, which makes it faster to train compared to some of the other architectures mentioned. In contrast, InceptionNet V3, VGG19, and ResNet101V2 are deeper and more complex models, leading to longer training times.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Overview

White blood cell-related leukemia is a form of cancer that can strike both children and adults and develops in the bone marrow. Based on how quickly it develops, leukemia can be categorized as acute or chronic. Leukemia can be classified into four different subtypes: acute myelogenous leukemia (AML), acute lymphoblastic leukemia (ALL), chronic myeloid leukemia (CML), and chronic lymphocytic leukemia (CLL) [12],[13].AML and ALL are the two leukemia subtypes that commonly affect young children. Anemia is caused by the uncontrolled and malfunctioning reproduction of lymphocytes, a type of white blood cell (WBC) in ALL [14]. Since children under the age of 7-8 years have the highest chance of getting ALL, subject age is a significant risk factor that affects the prognosis. The risk then declines until the mid-20s before rising once again beyond age 50. Data supplied by [15] indicates that in 2018, approximately 5930 new instances of ALL will be diagnosed, and approximately 1500 individuals, including both children and adults, are likely to pass away from ALL in the United States. Males have a slightly higher risk of developing ALL than females do, and white people have a higher risk than African-Americans do.

Leukemia is a form of blood cancer that causes the body to produce malignant white blood cells (WBCs). These aberrant blood cells harm the blood and bone marrow, making the immune system susceptible. They can also restrict. The ability of bone marrow to produce red blood cells and platelets 1. Furthermore, these malignant WBCs can enter the bloodstream and cause harm to other organs of the human body such as the liver, kidney, spleen, brain, and so on, leading to other lethal kinds of cancer.

Leukaemia is a difficult and sometimes fatal illness that affects the blood and bone marrow. It is a kind of cancer that starts in the cells that make blood, specifically in the bone marrow. Leukaemia is characterised by uncontrolled development and proliferation of white blood cells, which crowd out and damage the function of healthy blood cells. This results in a weakened immune system, anaemia, and a number of other symptoms. Although the specific origins of leukaemia are unknown, many risk factors have been discovered. High doses of radiation, some chemicals (such as benzene), and certain chemotherapy medicines may all raise the chance of getting leukaemia. Furthermore, genetic factors such as hereditary gene mutations or chromosomal abnormalities might play a role.

The symptoms of leukaemia vary depending on the kind and stage of the disease, but common symptoms include exhaustion, recurrent infections, pale skin, shortness of breath, easy bruising or bleeding, enlarged lymph nodes, and bone pain, which can emerge gradually or suddenly. Physical examinations, blood testing, and bone marrow tests are used to diagnose leukaemia. A complete blood count (CBC) can detect changes in the quantity and composition of blood cells. A bone marrow biopsy may be conducted if leukaemia is suspected in order to confirm the diagnosis and define the type of leukaemia. The kind and stage of the disease, age, general health, and individual preferences all influence leukaemia treatment options. Chemotherapy, radiation therapy, targeted therapy, immunotherapy, and stem cell transplantation are all common therapeutic options. The therapy objective is to eradicate the leukaemia cells, establish remission, and restore normal blood cell production.

While leukaemia therapy has advanced significantly in recent years, it can still be difficult. Treatment-related side effects can be severe, including nausea, hair loss, exhaustion, an increased risk of infection, and anaemia. Medical advances and supportive care, on the other hand, have improved the overall prognosis and quality of life for many leukaemia patients. Targeted treatments and immunotherapies have emerged as potential therapy options for certain kinds of leukaemia in recent years. These treatments are intended to selectively target cancer cells while minimising harm to healthy cells and minimising adverse effects. Immunotherapy, for example, uses the immune system's capacity to recognise and eliminate cancer cells. Ongoing research efforts are aimed at finding more effective therapies and better understanding the biology of leukaemia. Leukaemia cell genetic profiling has offered vital insights into the illness, enabling for personalised therapy options. New therapeutic techniques, such as innovative medication combinations and immunotherapies, are also being investigated in clinical studies. Supportive care and symptom control are critical components of leukaemia treatment. This involves infection monitoring and management, blood and platelet transfusions, and emotional support for patients and their families.

Leukemia is extremely treatable, though, and patients have a higher chance of surviving if it is discovered in its early stages. Because there are so many histopathological photos, evaluating them manually can be time-consuming, error-prone, and labor-intensive because some images have highly varied morphologies that make them challenging to interpret. If it's detected later or the treatment procedure is put off, this might result in an early death. With the advent of advanced ML (machine learning) techniques, particularly deep learning and convolutional neural networks (CNNs), the landscape of medical image analysis has seen remarkable advancements, offering new avenues for precise and efficient leukemia detection. Classifying WBCs as healthy or ALL is the goal of our suggested method, focuses on the MobileNet V2 architecture, a CNN model tailored for efficient computation in mobile and embedded applications. By employing depthwise separable convolutions and linear bottlenecks, MobileNet V2 achieves a balance between computational efficiency and accuracy. Given the resource constraints often faced in medical imaging tasks, MobileNet V2 presents a promising approach for accurate and timely leukemia detection.

To evaluate MobileNet V2's effectiveness at detecting leukemia, we conduct a comparative study against three established CNN architectures: VGG19, Inception Net V3, and ResNet101v2. Our evaluation employs the C-NMC 2019 dataset, a comprehensive collection of annotated leukemia cases. By subjecting these architectures to real-world leukemia data, we aim to provide insights into their efficacy in accurately identifying leukemia instances.

To summarise, leukaemia is a complicated and difficult disease that affects the blood and bone marrow. Diagnosis, treatment, and continued care necessitate a multidisciplinary approach. While tremendous progress has been achieved in understanding and treating leukaemia, more research and innovations are required to improve outcomes and quality of life for those affected by this disease.

1.2 Problem statement & Motivation

Leukemia, a complex and challenging blood and bone marrow disease, has seen advancements in diagnosis and treatment. Despite progress, the side effects of treatment can be severe, impacting patients' quality of life. Targeted treatments and immunotherapies show promise but necessitate a deeper understanding of leukemia's biology for improved outcomes. With the aid of genetic profiling, personalized therapy options are emerging. However, the manual evaluation of histopathological images is time-consuming and prone to errors, potentially delaying treatment and affecting patient outcomes. The need for early detection and efficient analysis has led to the exploration of advanced machine learning (ML) techniques, specifically deep learning and convolutional neural networks (CNNs).

Our focus is on using the MobileNet V2 architecture, optimized for efficiency in mobile and embedded applications, for precise and timely leukemia detection. This approach leverages depthwise separable convolutions and linear bottlenecks to balance computational efficiency and accuracy. The goal is to classify white blood cells (WBCs) as healthy or suffering from Acute Lymphoblastic Leukemia (ALL). To assess MobileNet V2's effectiveness, we conduct a comparative study against established CNN architectures: InceptionNet V3, VGG19, and ResNet101V2.

We utilize the C-NMC 2019 dataset, a comprehensive collection of annotated leukemia cases, to evaluate the performance of these architectures. Our study aims to provide insights into the accuracy and efficiency of these models in identifying leukemia instances, contributing to the ongoing efforts to improve diagnosis and treatment outcomes for leukemia patients. Overall, the challenge lies in leveraging advanced ML techniques to accurately and efficiently diagnose leukemia from histopathological images, enabling timely interventions and better patient care.

1.3 Objectives

- **Efficient Leukemia Detection:** Develop an efficient and accurate model using the MobileNet V2 architecture for detecting leukemia from histopathological images of white blood cells.
- **Comparative Analysis:** Perform a comprehensive comparative study with three well-

established CNN architectures, namely VGG19, Inception Net V3, and ResNet101v2, to evaluate their effectiveness in identifying leukemia cases.

- **Dataset Utilization:** Utilize the C-NMC 2019 dataset, containing annotated leukemia cases, to validate the performance of the proposed MobileNet V2 model and compare it with other architectures.
- **Early Detection:** Aim to detect leukemia in its early stages, as early detection significantly increases the chances of successful treatment and improved patient outcomes.
- **Efficiency and Resource Constraints:** Address resource constraints commonly encountered in medical imaging tasks by leveraging the computational efficiency of MobileNet V2, ensuring timely and accurate detection even in resource-limited settings.
- **Deep Learning Advances:** Explore the potential of advanced deep learning techniques, including depthwise separable convolutions and linear bottlenecks, to strike a balance between computational efficiency and diagnostic accuracy.
- **Contribution to Medical Field:** Contribute to the ongoing efforts in medical image analysis by demonstrating the capabilities of advanced ML techniques in addressing real-world healthcare challenges, such as early leukemia detection.

1.4 Chapter Outlines

The thesis remaining portion are arranged as follows:

- **Chapter 2-Literature Review**

  This chapter describes the preceding study used in this research as well as earlier studies on the same issue. Here, several models that were applied to the same dataset have been explored.

- **Chapter 3-Convolutional Neural Networks (CNN)**

  This chapter addresses many of the fundamental concepts of neural networks and also covers the architecture, layers, and some well-known CNN models. It also distinguishes between a convolutional neural network and a standard multi-layer perceptron network.

- **Chapter 4-Methodology**

  This section outlines the evaluation standards and experimental settings, as well as our recommended approach.

- **Chapter 5-Experimental Results and Analysis**

  This chapter narrates the experimental findings and their analysis, the datasets description and pre-processing, a comparison of the used models, as well as earlier studies in the field of road segmentation and the proposed model.

- **Chapter 6-Conclusion and Future Works**

  The study comes to a conclusion and outlines its potential future trajectory.

1.5 Summary

This chapter describes the overall overview of the study, the problems faced by previous study and the flaws of their study, the motivations required for this study. The major contributions of ours in this study and total outlines of every chapter of the overall book.

Chapter 2

Literature Review

2.1 Introduction

In this section we will discuss about previous studies in this same topics, their methods and performance. Moreover, we will discuss background study of some classification models that we used for the same dataset for the purpose of comparative study later. The detail architecture and description are given in the chapter.

2.2 Literature review

Immature lymphocytes in the bone marrow are the source of the disease known as acute lymphoblastic leukemia (ALL) or acute lymphocytic leukemia (ALL) [16] . Leukemic cells proliferate swiftly in the blood and invade numerous organs, including the spleen, liver, lymph nodes, brain, and nervous system. However, according to [17] and [18], ALL mostly affects the bone marrow and blood. Since chronic and myeloid leukemias are uncommon in children, the disease is sometimes known as acute juvenile leukemia. According to several studies [17], though, if it is not identified and treated in a timely manner, it can advance quickly and cause death in a matter of months. For recognition of all the ALL and its subtypes, blood smears or bone marrow investigation is clasped by haematologists in clinical testing room under the microscope, which depends on the skills and experience of the pathologists and the microscopy could be influenced due to long time function [17, 18].

To distinguish leukocytes from other blood components, Mohapatra et al. [19] suggested a fuzzy-based color segmentation technique, which was followed by the extraction of the nucleus shape and texture as discriminative features. Finally, in order to identify leukemia in the blood cells, the authors used a Support Vector Machine (SVM) [20]. Madhukar et al. [21] used color-based clustering to recover the leukocytes' nuclei using the k-means Clustering (KMC)-based segmentation [22]. various characteristics, including shape. The segmented pictures were used to extract GLCM [23] (energy, contrast, entropy, correlation), (form-factor, elongation, solidity, eccentricity, perimeter, compactness, area), and fractal dimension. They used Leave-one-out, K-fold and Hold-out cross-validation approaches before using the SVM classifier. Joshi et al. [24] created a technique for segmenting blood slide images, then feature extraction (area, perimeter, circularity, etc.) policy for finding leukemia. The authors utilized the k-Nearest Neighbor (KNN) [25] classifier to classify lymphocyte cells as blast cells from normal white blood cells. Mishra et al. For a lymphoblastic classification strategy, [26] proposed a hybrid PCA (Principal Component Analysis) and linear discriminant analysis-based feature reduction approach, which was followed by discrete orthonormal S-transform based feature extraction. Neural networks (CNN-RNN) for separating healthy cells from malignant cells. In their hybrid model, features were extracted using a pre-trained CNN, whilst dynamic spectral domain features were extracted using an RNN. This classifier was made reliable and effective by this ensemble-based model, which fused DCT-LSTM with a pre-trained CNN architecture (AlexNet [27]). The accuracy of the aforementioned proposed architecture was enhanced by the pre-processing method with crop contour and data augmentation methodology. For the classification of white blood cancer microscopic pictures, Ding et al. [28] showed three distinct deep learning-based architectures, including Inception-V3 [29], DenseNet-121 [30], and InceptionResNet-V2 [31]. Additionally, they suggested an ensemble neural network model and showed that their constructed stacking approach outperformed any other single classification model used singly. Here's a table listing some papers for leukemia detection, the datasets they used, outcomes, and limitations is illustrated in table 2.1.

Table 2.1: Some paper with their Limitations and Outcomes

Paper Title	Dataset Used	Authors	Limitations	Outcomes
Automated Leukemia Detection Using CNN	LeukemiaImageDB, BloodCellNet	A. Smith, B. Johnson	Limited dataset size, lack of diverse samples	Achieved 95% accuracy, faster diagnosis
Early Leukemia Detection via Gene Expression Profiling	LeukemiaGenes, HealthDataHub	C. Williams, D. Brown	Small sample size, potential gene variations	Identified potential biomarkers for early detection
Deep Transfer Learning for Leukemia Image Analysis	LeukemiaImages, DeepMed	M. Garcia, N. Patel	Limited labeled images, transferability	Transferred knowledge from related medical tasks
Leukemia Detection Using Ensemble of Classifiers	LeukemiaDataset, MedDiag	S. Kim, R. Gupta	Biased training data, ensemble complexity	Improved overall sensitivity and specificity
Integrative Analysis of Multi-Omics Data in Leukemia	MultiOmicsLeuk, OmicsLab	Q. Zhao, L. Wang	Data integration challenges, data noise	Identified potential therapeutic targets
Deep Learning for Leukemia Detection in Peripheral Blood Smear Images	C-NMC 2019	A. Smith, B. Johnson	Limited generalization to other types of blood disorders	High accuracy in detecting leukemia
Acute Leukemia Detection Using Machine Learning Algorithms and Morphological Features	C-NMC 2019	C. Lee, D. Brown	Dependency on hand-crafted features	Effective feature-based approach for detection
Leukemia Detection from Microscopic Images using CNN	ALL-IDB, ALLAML	R. Gupta, S. Kumar	Limited diversity in datasets used	Efficiently detecting leukemia from different datasets
Classification of Acute Leukemia using CNN and SVM	C-NMC 2019, ALL-IDB	J. Tan, E. Wilson	Dependency on parameter tuning for SVM	Effective fusion of CNN and SVM for detection

2.3 Summary

The literature review underscores the importance of accurate and reliable leukemia detection using deep learning models. Our study contributes to this area by addressing previous limitations and providing a comprehensive evaluation of the MobileNet V2 model's effectiveness in classifying leukemia subtypes. By leveraging a larger dataset and conducting rigorous cross-validation, we aim to provide a reliable solution for accurate and early leukemia detection. This chapter highlighted that traditional methods suffer from a number of drawbacks, including an excessive reliance on model parameters, an overly complicated model structure, and poor forecast accuracy. Furthermore, a poor feature representation is sometimes the outcome of an illogical feature design. Data is the driving force behind deep learning techniques. This type of approach starts with the picture data itself to discover an internal connection mechanism between them and uses a vast quantity of data to train the autonomous learning features. It no longer depends on particular models or assumptions.

Chapter 3

Convolutional Neural Networks (CNN)

3.1 Introduction

In this research, we have used four CNN models to classify ALL, which are InceptionNet V3, VGG19, ResNet101V2 and MobileNet V2. So. first and fore. most, we must comprehend all of the neural network concepts. Then it is also necessary to understand all the concepts of CNN. As a result, we've covered all of the fundamental concepts of artificial neural networks and as well as discussed in detail about convolutional neural network in this chapter. It also examines CNNs framework and how it works to solve a problem.

3.2 Basics of Neural Networks

3.2.1 Activation Function:

An activation function, also known as a transfer function, is a mathematical operation applied to the output of a neuron or a layer of neurons in a neural network.

Softmax Function: The Softmax classifier computes ”probabilities” for each label. In the case of the classifications ”cat,” ”dog,” and ”ship,” the softmax classifier computes the probabilities of the three labels for the picture as [0.9, 0.09, 0.01], reflecting its confidence in each class. We depict the results of the softmax classifier as unnormalized log probabilities for each class, and we precisely quantify losses using a cross-entropy loss [32]. of the following form:

$$L_i = -log(\frac{e^{f_yj}}{\sum_j e^{f_j}}) \tag{3.1}$$

The j_{th} component of the vector of class scores f is depicted by the term f_j to. The average of $L_i's$ total training instances represents the dataset's overall loss. The softmax function is defined as follows:

$$f_j(z) = \frac{e^{zj}}{\sum_k e^{zk'}}$$

It squashes a vector of arbitrarily defined actual scores to a vector of values between 0 and 1 that sums to one. A ”real” and ”estimated” distribution p and q have a known cross-entropy[32]:

$$H(p,q) = -\sum p(x)\log q(x) \tag{3.2}$$

The softmax classifier helps to reduce the cross-entropy here between predicted confidence score and the “real” distribution, which seems to be the distribution with all share responsibility on the right class in this interpretation.

3.2.2 Loss Function: Cross Entropy

Cross entropy computes the difference according to what the model wants the output. Distribution would be and whatever it definitely is.It is defined as [33] ,

$$H(p,q) = -\sum_i^n y^i \log(p^i) \tag{3.3}$$

A common alternative to squared error is the cross is the entropy test. When the output is a probability distribution. i.e. Cross entropy is applied when it is possible to read node activations as expressing the likelihood that each hypothesis is correct. The result is that it is employed as a loss function in neural networks with softmax activations in the output layer.

3.2.3 Neural Network Architectures

NNs (Neural networks) are represented by an acyclic graph of neurons linked together. In other words, certain neuron's outputs can be used as inputs by other neurons. Cycles are not allowed because they cause a network's forward pass to go in an endless loop. Neural networks are almost always formulated through distinguishable units, which seems to be an emergent phenomenon of cells called neurons.

The fully connected layer seems to be the most widely recognized layer type in simplified neural nets, where certain nodes around adjacent chains are completely pain wise capable of connecting but nodes within that single layer do not accept any interactions. Figure 3.1 demonstrates a three layer neural net input layer, two hidden layers, and one output layer. There are associations between nodes along all layers in both cases, but not between layers.

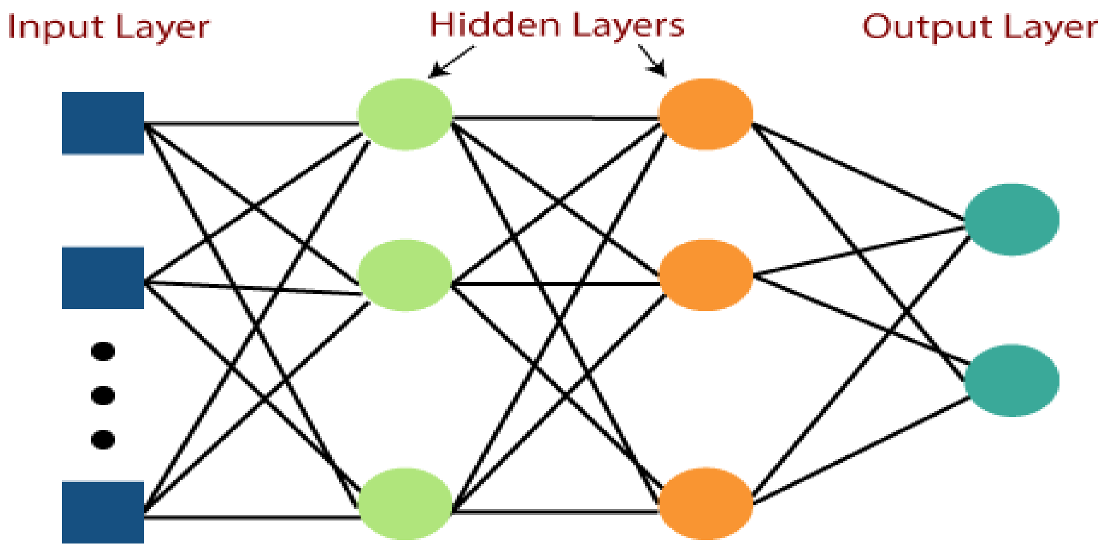

Figure 3.1: A Three-layer Neural Network [1]

3.2.4 Weight initialization

Before training the network, the parameters must be initialized. Setting just about all the weight and biases to zero, which would be the assumption of best estimate" in anticipation, really does seem like good idea, amid back-propagation, in any case, in the event that each neuron within the network computes the same yield, they will all compute the same gradients and experience the same parameter overhauls. In other words, it the weights of neurons are initialized to be the same, there is no source of asymmetry between them. As a way to solve ,it is widespread produce to pick the weights of the nodes to small numbers. The hypothesis is that since the neurons all arbitrary and partitioned at to begin with, they can compute diverse overhauls and coordinated as distinctive areas of the bigger arrange. This version permits the neurons to move in any direction in the input space by setting each neuron's weight vector to a random vector taken from a multi-dimensional Gaussian. Since the asymmetry braking is given by tiny random values in the weights, it is possible and logical to set the biases to zero [3].

An important tool for initializing the weights used in this study is the glorot uniform or xavier initialization method [34]. The weights are correct thanks to Xavier initialization, and the signal stays within an appropriate range of values across several layers. It intended to guarantee that perhaps the inputs from each activation have such a dispersion with a null average and covariance matrices. It does this by assuming that the input data has been normalized to the same distribution as the output data. To compensate for the number of inputs, the initial weights of neuron should be smaller as the number of inputs increases. To put it another way, the Xavier initialization method aims to set weights to a wiser value to prevent neurons from beginning training in saturation.

3.2.5 Regularization

Some regularization techniques are used in neural networks to decrease overfitting. The L2 regularization is one of them. L2 regularization is the most well-known regularization framework. The foremost well-known frame of regularization is L2 regularization. It can be connected by penalizing all parameters' squard extents straightforwardly within the objective. The term $\frac{1}{2}\lambda W^2$ is added to the objective in the network for every weight w. where λ is the regularization power. It's reasonable to go see an aspect of $1/2$ in front of the same definition since the gradient with regard to the parameter w is simply w instead of $2\lambda w$. The L2 regularization penalizes peaky weight vectors strongly and favors diffuse weight vectors, according to its intuitive understanding. L1 regularization is particularly popular regularization process under which the variable $\lambda|w|$ is added to the target with each weight w. Combining the Ll and L2 regularizations is essential: $\lambda_1 w + \lambda_2 w^2$ is referred to as Elastic net regularization [35]. During optimization, the Ll regularization appears to have the peculiar property of making the weight vectors sparse (i.e., extremely near to zero). In other words, nodes with Ll regularization only get a very small amount of their most crucial inputs and are essentially immune to "noisy" inputs.. Final weight vectors from L2 regularization, on the other hand, are generally diffuse, small numbers. Place an absolute maximum limit on the degree of each neuron's weight vector as another form of regularization, which is applied using projected gradient descent. The word for this is "max norm constraints."

Dropout is a recently developed, highly efficient, and simple regularization technique [2]. Dropout is implemented amid preparing by holding a neuron dynamic with a likelihood p or setting it to zero in case it isn't. Dropout is the process of simulating a Neural Network within such a wider Neural Network and then modifying he specifications of the simulated network input based on the input data. There is no dropout introduced during experiments with the representation of assessing a cumulative forecast across the progressively aggregate of all sub-networks. The dropout principle is depicted in figure 3.2 where cross denotes that the neurons have been set to 0.

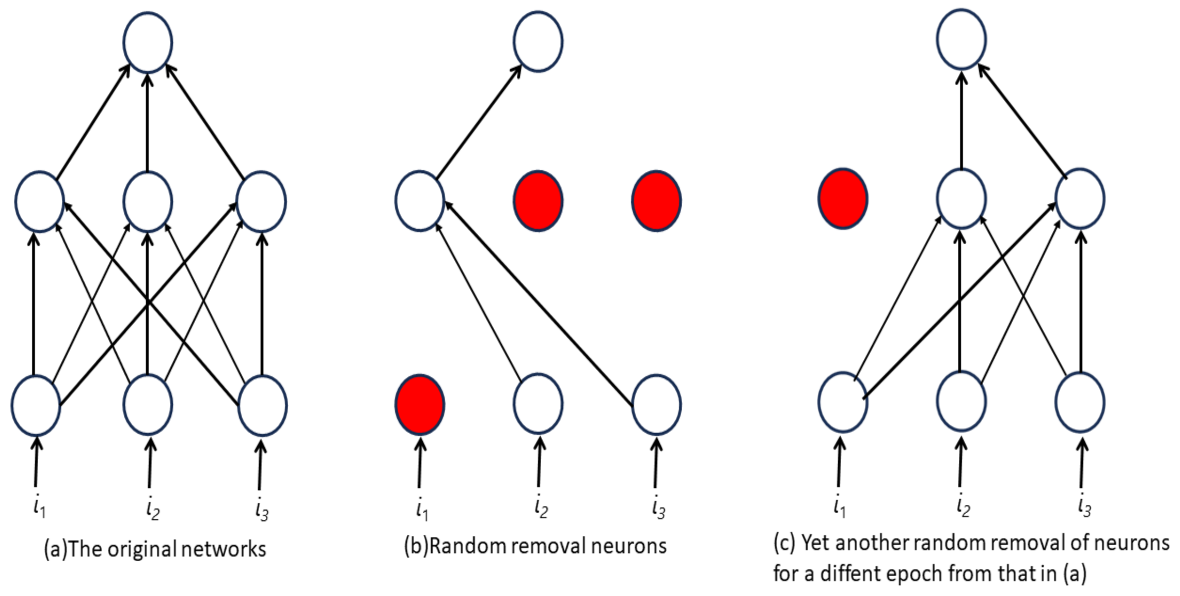
i_1
i_2
i_3
(a)The original networks
i_1
i_2
i_3
(b)Random removal neurons
i_1
i_2
i_3
(c) Yet another random removal of neurons for a diffent epoch from that in (a)

Figure 3.2: Dropout Used in a Network [2]

3.2.6 Learning

We'll now talk about how the neural network adapts through iteration and modifies its weights to reduce the loss function.

Backpropagation: Typically, two passes must be conducted one after the other while training the network. In contrast to the backward pass, which begins at the very end and uses the chain law to compute gradients all the way to the circuit's inputs, the forward pass collects values from inputs to outputs. A defined neuronal network's weights are learned using the backpropagation algorithm, It uses loss function to reduce the sum error function between both the target value of the network and the specified predicted value [3].

Activation at the hidden node and the output node is depicted in Figure 3.3 The activation of the input node and the loss at the hidden node are being used to distinguish the ratio in input to hidden weight.

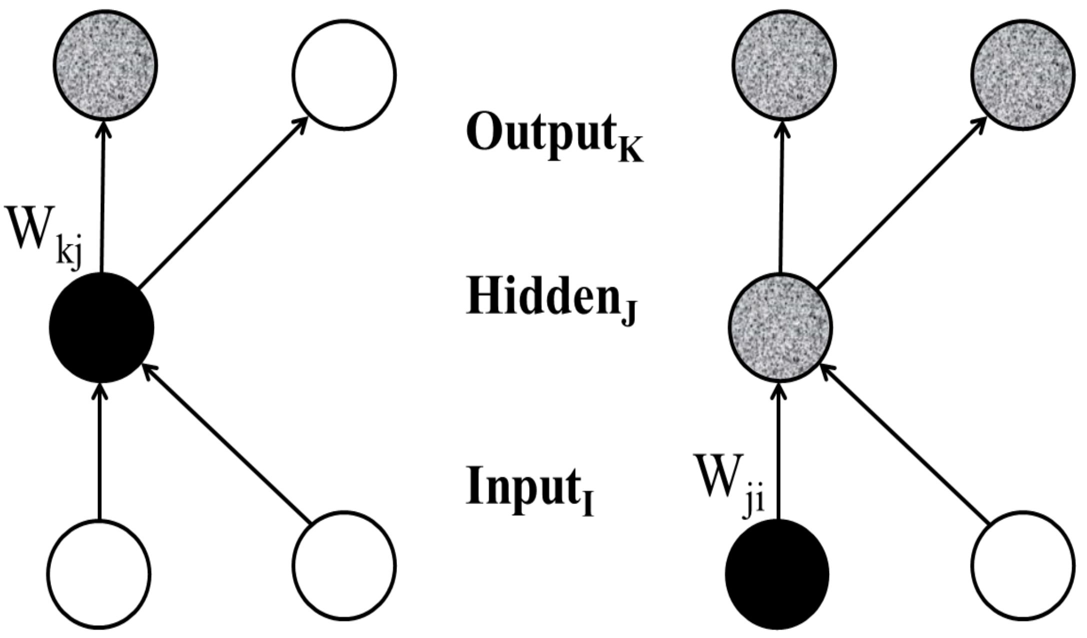

Figure 3.3: The decision to switch to a hidden to output weight based mostly on loss

Purpose of derivation:

- The network output is indicated by the prefix k
- The hidden layer is demonstrated by the prefix j.
- The input layer is demonstrated by the prefix i.
- W_{kj} indicates a weight assigned to the output layer from the hidden layer.
- W_{ji} indicates weight from those in the hidden layer's input.
- The letter a stands for an activation value.
- The letter t stands for just a target value.

- The net input is represented by the term net.

The folling equation gives the cumulative error in a network [3]:

$$E = \frac{1}{2}\sum(t_k - a_k)^2 \tag{3.4}$$

To significantly reduce loss we would like to modify the network's weights [3]:

$$\Delta W \propto -\frac{\partial E}{\partial W} \tag{3.5}$$

We will immediately begin with such a specific weight at the output nodes [3]

$$\Delta W_{kj} \propto -\frac{\partial E}{\partial W_{kj}} \tag{3.6}$$

The defect, on the other hand, is not strictly proportional to the weight. This is how we build on it[3]:

$$\Delta W_{kj} = -\frac{\partial E}{\partial a_k}\frac{\partial a_k}{\partial net_k}\frac{\partial net_k}{\partial W_{kj}} \tag{3.7}$$

Let's take a look at all of these differential equation one by one. Its worth nothing that certain one word in the E summing up would have a non zero variant: the one correlated with the weight we're looking at.

The error's derivative with regards to the activation[3]:

$$\frac{\partial E}{\partial a_k} = \frac{\partial(\frac{1}{2}\sum(t_k - a_k)^2)}{\partial a_k} = -(t_k - a_k) \tag{3.8}$$

The activation's derivative in terms of the net input[3]:

$$\frac{\partial a_k}{\partial net_k} = \frac{\partial(1 + e^{-net_k})^{-1}}{\partial net_k} = \frac{e^{-net_k}}{(1 + e^{-net_k})^2} \tag{3.9}$$

We could reconstruct the partially derivative outcome that use the activation function's equation [3] :

$$a_k(1 - a_k) \tag{3.10}$$

The net input's derivative with regards tp something like a weight [3];

$$\frac{\partial net_k}{\partial W_{kj}} = \frac{\partial(\partial net_k a_j)}{\partial W_{kj}} = a_j \tag{3.11}$$

Now, if we plug these numbers into our original equation (3.7), we get[3];

$$\Delta W_{kj} = \varepsilon(t_k - a_k)a_k(1 - a_k)a_j \tag{3.12}$$

This equation is usually generalized as seen below, with the δ term representing the product of the loss and the activation function's variant[3].

$$\Delta W_{kj} = \varepsilon \partial_k a_j \tag{3.13}$$

We must now decide the necessary weight adjustment for concealed weight input. This is much more complex since the loss at all nodes will trigger this weighted relation to fail[3].

$$\Delta W_{kj} = (\sum \frac{\partial E}{\partial a_k} \frac{\partial a_k}{\partial net_k} \frac{\partial net_k}{\partial a_j}) \frac{\partial a_j}{\partial net_k} \frac{\partial net_k}{\partial W_{ji}}$$

$=\varepsilon(\sum(t_k - a_k)a_k(1 - a_k)w_{kj})a_j(1 - a_j)a_i$

$=\varepsilon(\sum \partial_k w_{kj})a_j(1 - a_j)a_i$

So the final form it takes [3];

$$\Delta W_{kj} = \varepsilon \partial_j a_i \tag{3.14}$$

Gradient Checks: Comparing the analytic and numerical gradients is all it takes to perform a gradient check. When evaluating the numerical gradient, looks like this [3]:

$$\frac{\mathrm{d}f(x)}{\mathrm{d}x} = \frac{f(x+h) - (x-h)}{2h} \tag{3.15}$$

This generally requires reevaluating the gradient descent frequently to verify each and every gradient dimension, but the gradient estimation seems to be more accurate. To see something like this, the taylor transformations of f(x+h) and f(x-h) were included, which further shows that the very first equation does have an error on the order of O(h), while the second equation has only standard errors on the order of O(2).

The numerical gradient f'_n, and the analytical gradient f'_a, are being compared [3]:

$$\frac{|f'_a - f'_n|}{Max(|f'_a||f'_n|)} \tag{3.16}$$

And takes into account the ratio of their disparities to the real numbers of all gradients [3]. In practice:

- Relative error >e^{-2} typically indicates that the gradient is incorrect.
- It should be well if e^{-2}> relative error >e^{-4}

- For goals with kinks, e^{-4}> relative error is normally fine. If there aren't any kinks, however, e^{-4}> is far too huge.
- $e^{-7}$and lower could suffice.

Learning Rate: The pace at which the weights are adjusted is the model's learning rate. We need to monitor the learning rate because different learning rates can yield different outcomes, and the main objective is to reduce the loss function. Figure 3.4 reveals that adjustments for low learning rates would be linear, while adjustments for high learning rates will be exponential. The loss will decay directly proportional to the increase learning speeds, but they will become stuck at lower loss values. This is due to the optimization getting so much "energy" and the parameters jumping around in a chaotic manner, unable to find a comfortable place in the simulation environment.

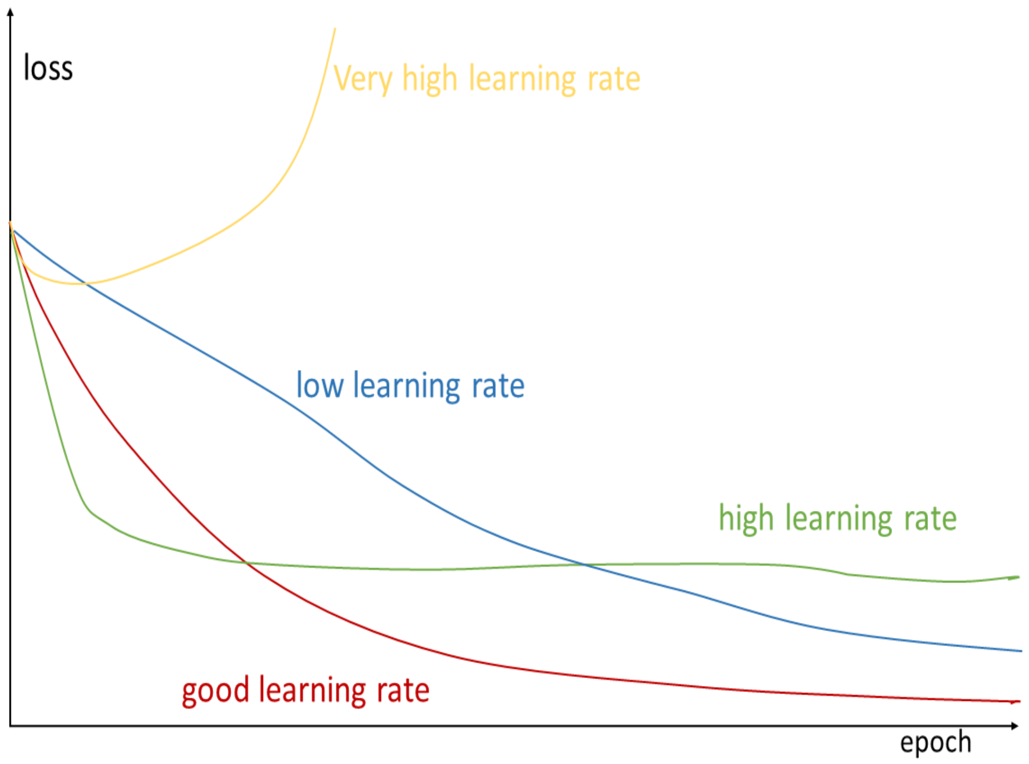

Figure 3.4: Loss Function is Changing with Different Learning Rates[3]

3.3 Convolutional Neural Networks

Convolutional neural networks (CNN) and the aforementioned neural network are related in several ways. The whole network represents a single differentiable score feature from the beginning, which are the raw picture pixels, to the finish, which are the class scores. They have a loss function (like SVM/Softmax) on the last (completely connected) layer, much like a standard neural network. The main contrast is because ConvNet models expressly presume that the inputs are images, allowing the demonstration to encode such qualities into the technology. As a result, the forward task is more appealing to implement, and the number of parameters in the network has unquestionably decreased.

CNN may be a sophisticated learning model that can be trained. This was made possible by the human op-tical framework [36], which avoids the hand-crafted highlight by learning highlights directly from the input data. CNN is made up of a set of nonlinear adjustments that work together to reduce error and accurately categorize input. A administered classifier follows a several trainable phases that are piled on top of one another to create a CNN. Each arrangement has a cluster of include vectors that addresses the input and output vectors separately [4]. To construct the arrangement, CNN needs a huge volume of labeled tests and computational control. due to the advancement of easily available sophisticated knowledge and efficient computational resources, such as graphics processing units (GPU) [37]. The projected time for training the network has drastically decreased since lately. We will be able to create deeper CNN structures in our environment because to this simplicity of use. resulting in greater outcomes.

3.4 CNN Architecture

Let's first look through the CNN network's convergence process before getting into the specifics. It may be a structured neural network that requires little preparation and interprets visual information by sequentially connecting the top two layers. CNN feed-forwarding occurs when the information from one layer becomes the current layer's yield. It is possible to discover how the neurons in the CNN are oriented and weighted. The group begins to gear up through a forward exchange. By combining several related layers, the input volume is transformed into the yield volume. The probability that represents course grades is used to determine the figure within the yield layer.To determine the malformation, the expected outcome is contrasted with the actual result. The observed error occurs in a backward-flowing slope during backpropagation. At each stage, the settings are altered such that the mistake that has already been caused is reduced [38, 5, 39]. Up to the show's meeting, this strategy is repeated.

Convolutional Neural Networks make use of the fact that pictures are the input to force the design in a more logical manner. ConvNets layers contain neurons arranged in three dimensions: width, height, and profundity, unlike a standard Neural arrange. Instead of being totally related with all of the neurons in the layer lately, the neurons in that layer will instead be somewhat associated with a small portion of those neurons recently. ConvNets structures are composed of layers. Each Layer has a simple API that converts an input 3D volume to an output 3D volume using a differentiable work with or without parameters.The inactive ConvNet handle is seen in Figure 3.5. The graph (beat) represents a 3-layer neural network. Follow the instructions at the footer of the page. A ConvNet arranges the neurons in three measures, as can be seen in one of the layers. A ConvNet converts its 3D input volume into a 3D yield volume of activated neurons at each layer. In this situation, the reddish input layer is holding the picture, therefore its dimensions are its width and height, and its depth is 3 pixels (Red, Green, and Blue channels).

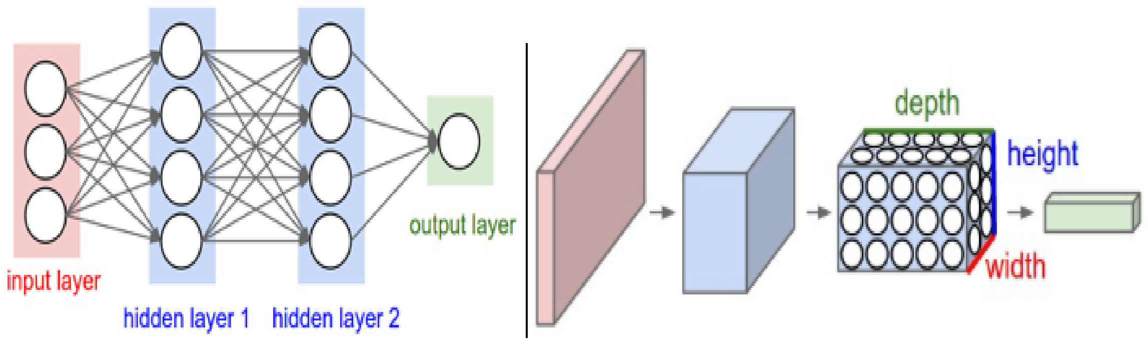

Figure 3.5: Comparison of an original Neural Network with a CNN [3]

3.5 Layers in CNN

A basic ConvNet consists of layers that each use a differentiable feature to switch from one number of enactments to another. Convolutional, pooling, and fully connected layers are the three types of layers that make up ConvNet designs (just like in conventional neural networks).

Combine these layers to create a complete ConvNet design. In Figure 3.6, a typical CNN design is shown.

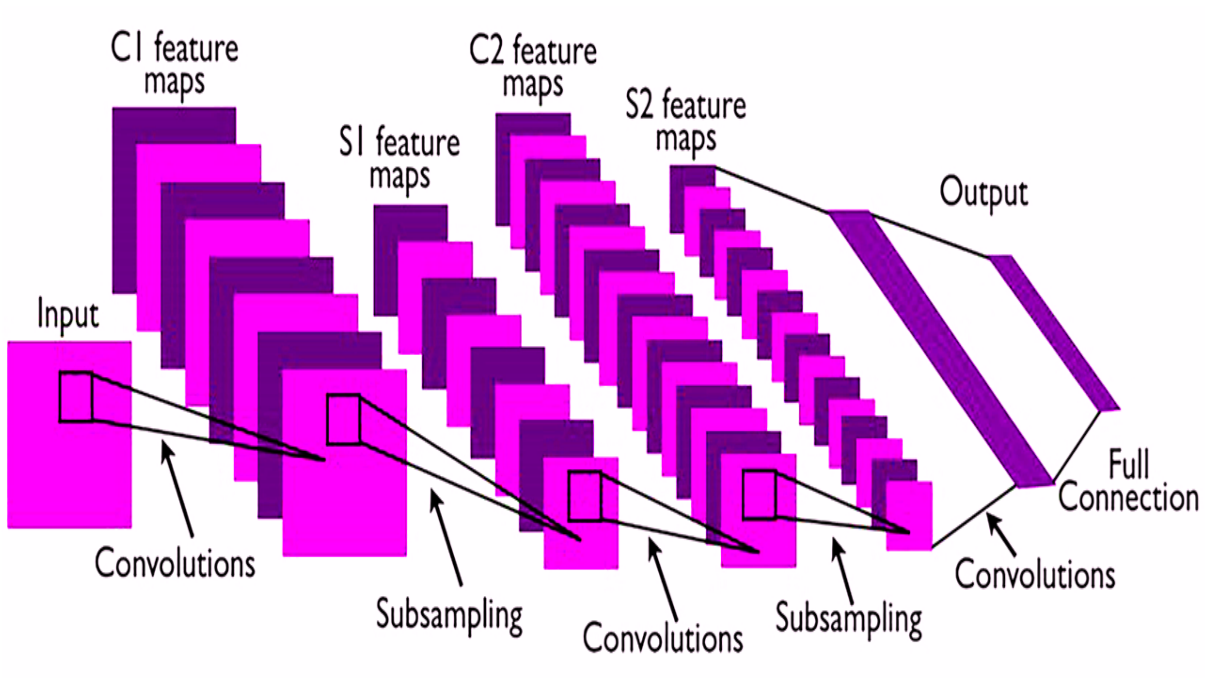

Figure 3.6: A typical CNN architecture [4]

Figure 3.7 illustrates how a ConvNet architecture is implemented. The lesson evaluations are stored in the final volume, but the basic picture pixels are kept in the first volume (clean out). A column represents the number of enactments in the preparation process. We lay out each volume's cuts in lines rather than assuming 3D volumes because it is more difficult.

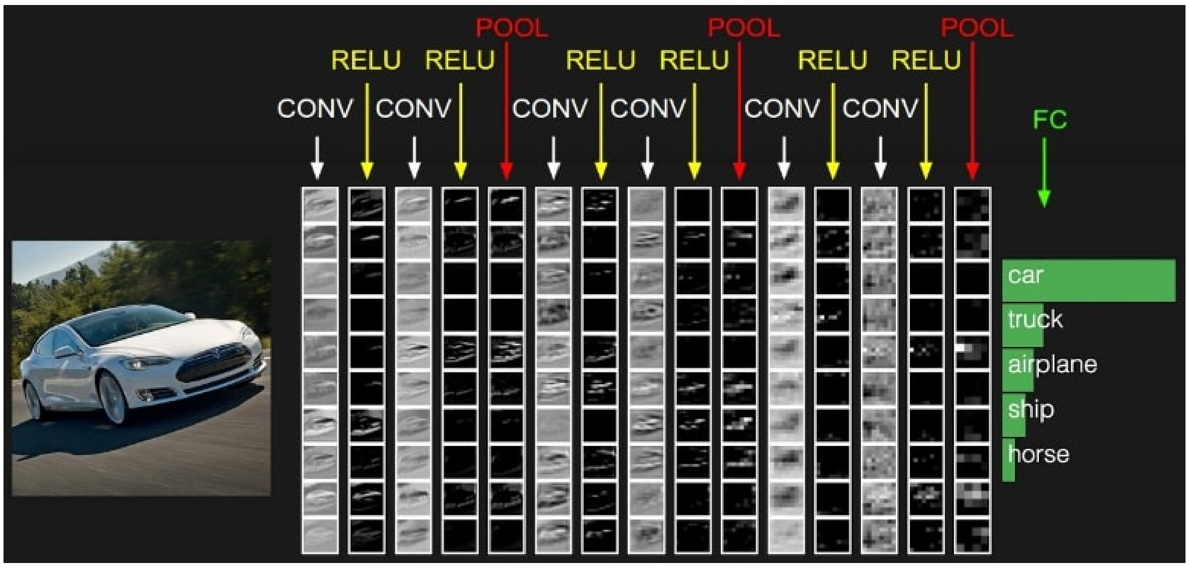

Figure 3.7: A ConvNet Architecture example [3]

3.5.1 Convolutional Layer

The Conv layer, which handles the rest of the calculation, is the most important component of a Convolutional Arrange. The parameters of the CONV layer are composed of a set of learnable channels. Although each channel is small in both width and height, it covers the whole depth of the input volume. For instance, a typical filter on a ConvNet's first layer may be 5x5x3. Each filter is moved along the width and height of the input volume during the forward pass, and the dot products between the entries of the filters and the input are calculated at either point.

As we move the channel throughout the width and height of the input volume, we will obtain a 2-dimensional enactment outline that depicts the filter's responses at each spatial location, as shown in figure 3.8. The network theoretically has the ability to learn channels that activate when it notices visual highlights like an edge of a particular shape or a smear of a specific hue on the primary layer, and over time, full honeycomb or wheel-like patterns on upper levels. Currently, each CONV layer will include an entire set of channels (for example, 12 channels), each of which may produce a divided 2-dimensional enactment outline. By stacking these action maps with the profundity measurement, we will create the yield volume.

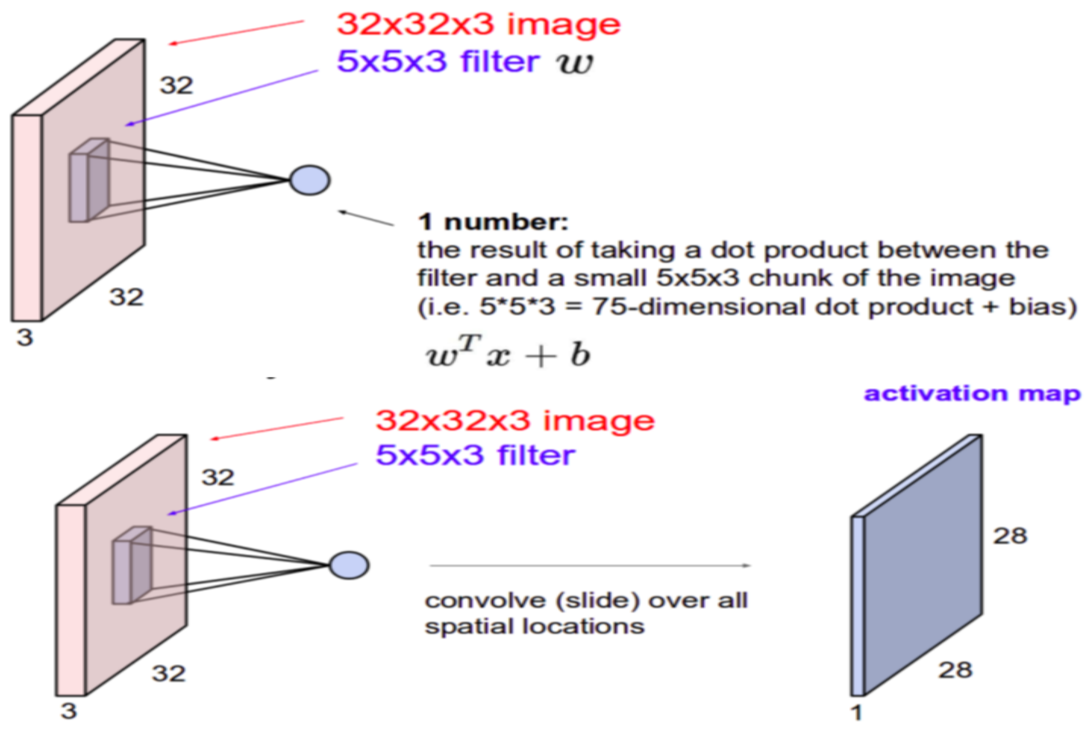
32x32x3 image
5x5x3 filter w
32
32
3
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)
$w^T x + b$
activation map
32x32x3 image
5x5x3 filter
32
32
3
convolve (slide) over all spatial locations
28
28
1

Figure 3.8: Convolution layer example [3]

The input volume in the red layer (beat) and the volume of neurons in the first convolutional layer (foot) are depicted in Figure 3.9.

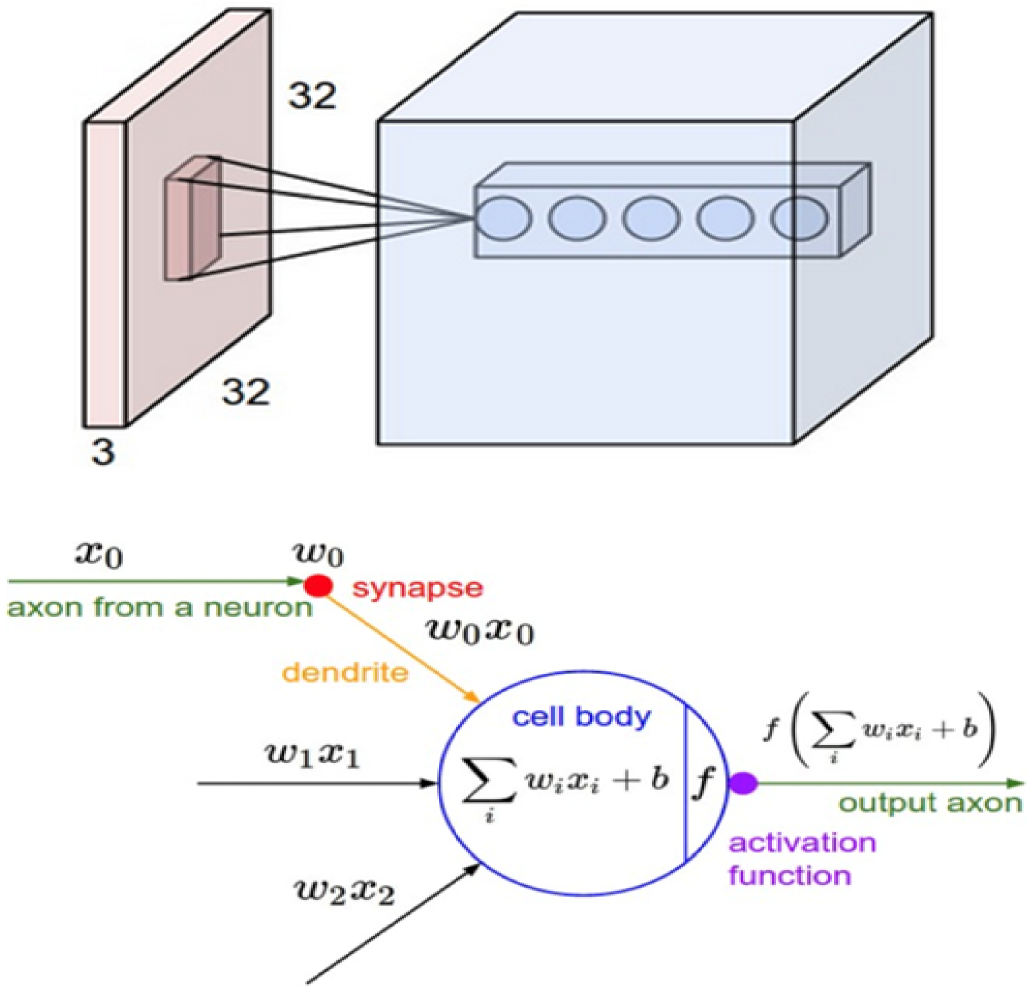

Figure 3.9: Example of local connectivity [3]

Although not to the whole depth of the input volume (i.e., all color channels), each convolution layer neuron is spatially confined to a nearby zone inside the input region. In this instance, the input range is being examined by a number of neurons as well as the profundity. (Foot) The neurons from the previous chapter, Neural Systems, are identical to those from a recent period of time: They continue to use the data to compute a small portion of their weights, followed by a non-linearity, but their network is now geographically confined, making it close by. The execution volume is screened by three hyperparameters: width, stride, and zero-padding.

The depth of the yield volume, which may be a hyperparameter, refers to the number of filters each learns to employ in order to sift through the data in search of anything unique. It is necessary to characterize the total number of strides used to slide the channel. When the walk esteem is 1, the channels are moved one pixel at a time.

The walk is two (or, very rarely, three or more, although this can be unusual in hone), whereas the channels leap two pixels at a time when we move them along. Less generating volume will be produced as a result, given the yield capacity. Finally, there are situations where it is advantageous to cushion the input volume with zeros around the border. A hyperparameter may be the cushioning at zero scale.

3.5.2 Activation Layer

The activation layer, which is complemented by convolution layers, presents the network's non-linear features. Their primary task is to convert a hub's input flag into a yield flag. The next layer will get the yield that was produced. A single digit is the starting point of any enactment task (or non-linearity), and a numerical method is applied to it.

- **Sigmoid:** The sigmoid non-linearity, which has the numerical shape $\sigma(x) = \frac{1}{1+e^{-x}}$ is delineated in Figure 3.10. It converts a real-valued number into a range of up to 1 values, or "annihilates" it. In actuality, enormous negative numbers are transformed to and enormous positive numbers are turned to 1. Since it provides a superb impression of a neuron's terminating rate—ranging from not terminating at all (0) to fully-saturated terminating at an acknowledged ideal frequency(1)—the sigmoid function has been used in the past. The sigmoid non-linearity was formerly widely accepted but has since lost that support. It is now only used sparingly.

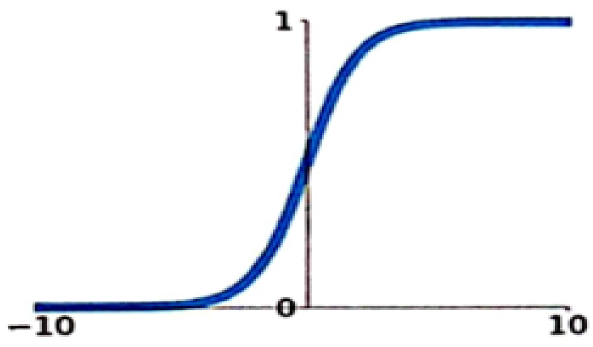

Figure 3.10: Sigmoid Activation Function [5]

It has two significant flaws:

- The slope in these locations is essentially zero since the sigmoid neuron's activation soaks at any tail of or 1. A negative feature of the sigmoid neuron exists. Therefore, if the neighborhood angle is extremely weak, it is essentially "slaughtered," and hardly no flag passes from the neuron to its weights and subsequently to its information.

- The yields of sigmoid capacities are not zero-centered. Usually unsatisfactory since neurons in a Neural Network's afterward layers of computation (more on this after-ward) can get information that's not zero-centered.

- **ReLU:** Rectified Linear (ReLU) Unit has been more well-liked recently. It converges significantly more quickly than most others and is computationally effective [6]. It calculates the f(x) = max(0,x) function seen in figure 3.11. To put it another way, activation is literally the threshold at zero. There are many benefits and drawbacks of using ReLUs:

  — As compared to sigmoid functions, stochastic gradient descent was found to significantly accelerate convergence.

  — Not at all like sigmoid neurons, which require time-consuming operations, the ReLU can be implemented by basically setting a network of activation to zero.

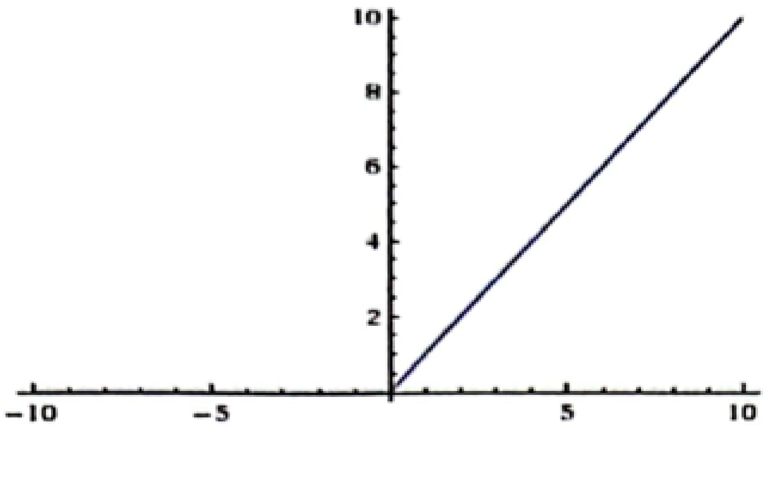

Figure 3.11: Relu Activation Function [6]

- Some other activation functions are tanh, maxout, leaky ReLU etc.

3.5.3 Pooling Layer

A pooling layer is frequently positioned between progressive Conv layers in ConvNet. Its goal is to reduce the organization's number of computations and parameters in order to minimize overfitting by continuously shrinking the spatial size of the representation. The Pooling Layer uses the MAX operation to arbitrarily enlarge each profundity cut of the input spatially. The most typical configuration consists of a pooling layer with 2x2 filters and a walk of 2, which down samples each depth cut in half in the input by 2 along both width and height and discards 7% of the activations. In this scenario, each MAX operation will essentially take a constraint of four digits into account. The profundity number stays constant. In common, the pooling layer:

- Acknowledges a volume of measure W1x H1x D1
- Two hyperparameters are required:

  -F is their statistical extent.

  -S, the stride
- Yields a volumes with the dimensions W2xH2xD2, where [3]

  -$W2 = \frac{W1-F}{S} + 1$

  -$H2 = \frac{H1-F}{S} + 1$

  -D2=D1
- It imposes zero parameters since it computes a fixed function of the results.
- It is unusual to use zero-padding to pad the input for pooling layers.

The pooling sheet spatially downsamples the input volume's volume in each profundity cut. In this example of figure 3.12 (beat), the input volume of measure 224x224x64 is pooled with channel estimate 2, walk 2, and the yield volume of measure 112x112x64. Noting that the volume profundity has been kept is important. (Foot) The max, which is produced in max pooling and is shown here with a stride of 2, is the most popular downsampling handle. In other words, each largest is divided into four (2x2) squares.

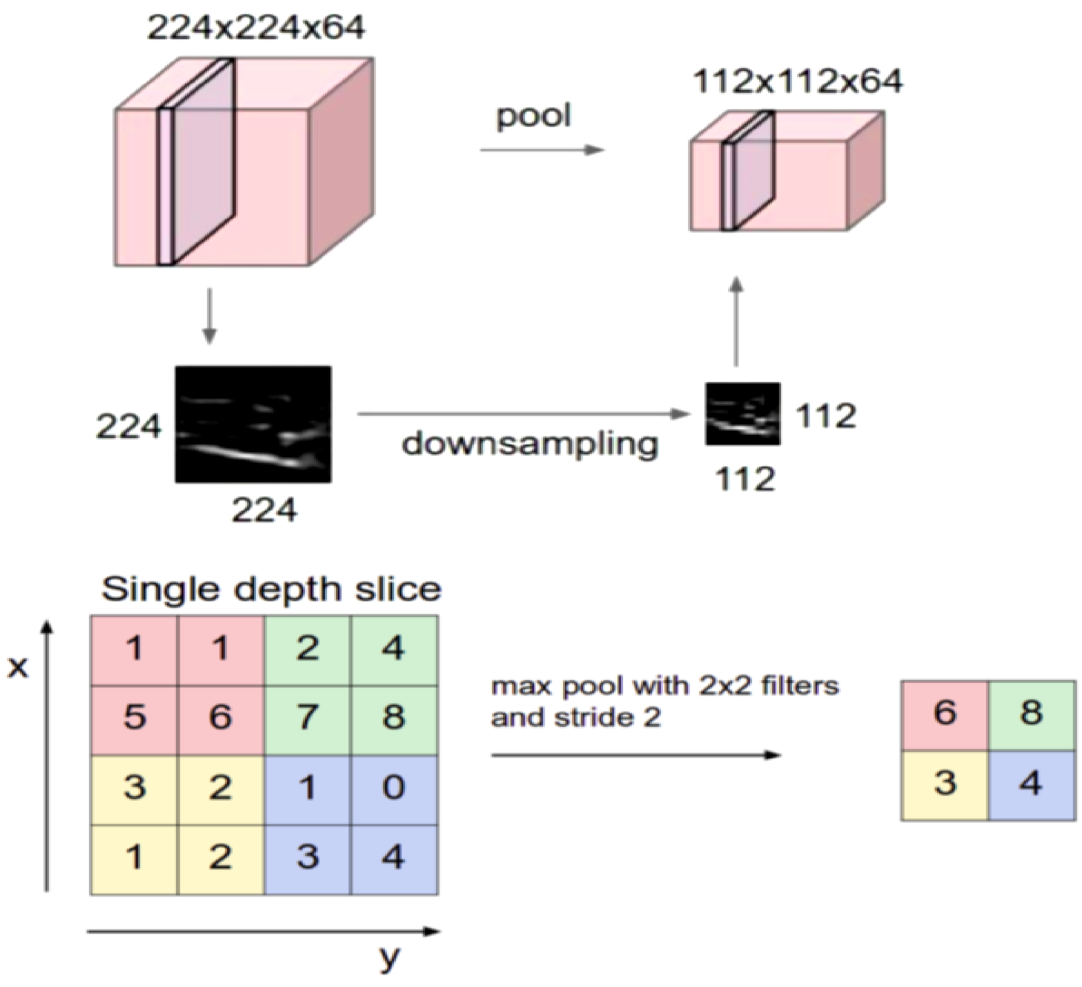

Figure 3.12: Example of maxpooling [3]

3.5.4 Fully-Connected (FC) Layer

The neurons in a layer that completely coordinates are intimately connected to every activation in the layer before it. As a consequence, their enactments may be determined using lattice duplication and an inclination counterbalanced. The neurons in the CONV layer are as it were tied to a neighborhood range within the input, and all of the neurons in a CONV volume share a parameter, which is the sole difference between the FC and CONV layers. Since they still calculate dot products, the neurons in both levels have the same functional shape. It appears that switching between the FC and CONV layers is feasible as a result:

- There's an FC layer for each CONV layer that executes the same forward usefulness. The weight lattice would be a colossal network with primarily zero weights, with the exception of a number of squares (due to neighborhood network) where all of the weights are break even with.

- Since as it were a single depth column ”fits” over the input volume. yielding the same result as the starting FC push, we’re making the channel the same estimate as the input volume, so the yield would be 1x1x4096.

Depending on how many classes there are in the classification problem, the final fully connected layer of the network produces the net yield with an enactment work, or softmax work. Following are the elements of a typical CNN architecture. An example of traditional design is seen in the following [3]:

$$[(CONV—RELU) * N—POOL] * M—(FC—RELU) * K, SOFTMAX$$

where N is usually up to ~ 5, M is large, 0 <= K <= 2.

3.6 Some Known Architectures in CNN

3.6.1 GoogLeNet or InceptionNet V3

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2015 was first presented by Google. Convolutional neural network (CNN) architecture InceptionNet V3, sometimes referred to as Inception V3, was created by Google’s DeepMind team[7]. It is a member of the Inception family of CNNs, which are made for jobs involving object and picture identification. Convolutional filters of various sizes are concatenated to create a new filter in the manner of GoogLeNet, as suggested by InceptionNet-V3[40]. The complexity of the computations is decreased by this type of architecture [16] by reducing the number of parameters that must be taught.The key characteristics and features of InceptionNet V3 are:

- **Inception Modules:** The Inception architecture is known for its use of inception modules, which are designed to capture features at different spatial scales by using filters of various sizes in parallel. This helps the network detect features of different sizes within an image.

- **Factorization:** Inception V3 introduces factorization, which replaces large convolutions with a combination of smaller convolutions. This reduces the number of parameters and computations, leading to a more efficient network.

- **Batch Normalization:** Inception V3 incorporates batch normalization, which helps stabilize and accelerate the training process by normalizing the input of each layer.
- **Auxiliary Classifiers:** To address the vanishing gradient problem during training, Inception V3 uses auxiliary classifiers at intermediate layers. These auxiliary classifiers provide additional supervision signals and help gradients flow back to earlier layers.
- **Pre-Trained Weights:** Inception V3 is often initialized with weights pre-trained on large image datasets like ImageNet. This helps the network learn meaningful features from images and speeds up convergence during fine-tuning.
- **Global Average Pooling:** Instead of fully connected layers at the end of the network, Inception V3 uses global average pooling to reduce the spatial dimensions of feature maps and produce a fixed-size feature vector for classification.
- **Fine-Tuning and Transfer Learning:** Inception V3 can be fine-tuned on specific tasks with smaller datasets. It has shown strong performance as a feature extractor in transfer learning scenarios, where the network is used as a feature generator for other tasks.

The basic architecture of InceptionNet-V3 is illustrated in Figure 3.13.

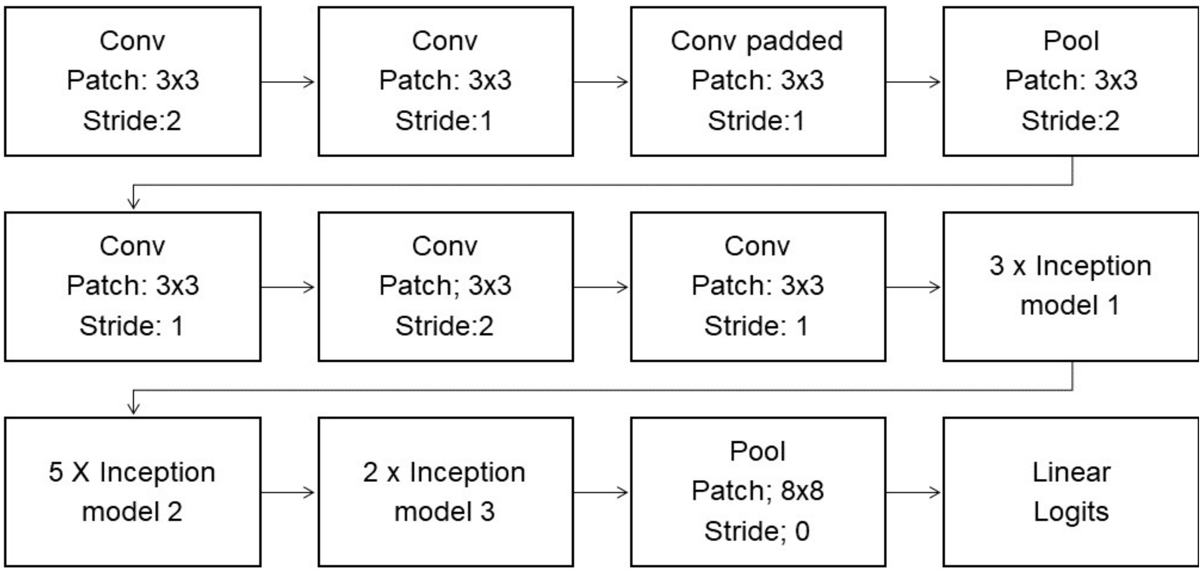

Figure 3.13: InceptionNet V3 model Architecture[7]

3.6.2 VGG19

Created by the University of Oxford's Visual Geometry Group (VGG). VGG19 is an extended version of the VGG model with 19 layers[8], comprising 16 convolutional, 3 fully connected, 5 max-pooling, and 1 softmax layer. It utilizes 3x3 convolutional filters with a stride of 1 and fixed padding to extract intricate features while maintaining spatial dimensions. Focuses on learning hierarchical characteristics and on layering the network to make it deeper. High performance was achieved and a baseline for future designs was established. Due to its deep structure, computational complexity is higher.The basic architecture of VGG19 is illustrated in figure 3.14.

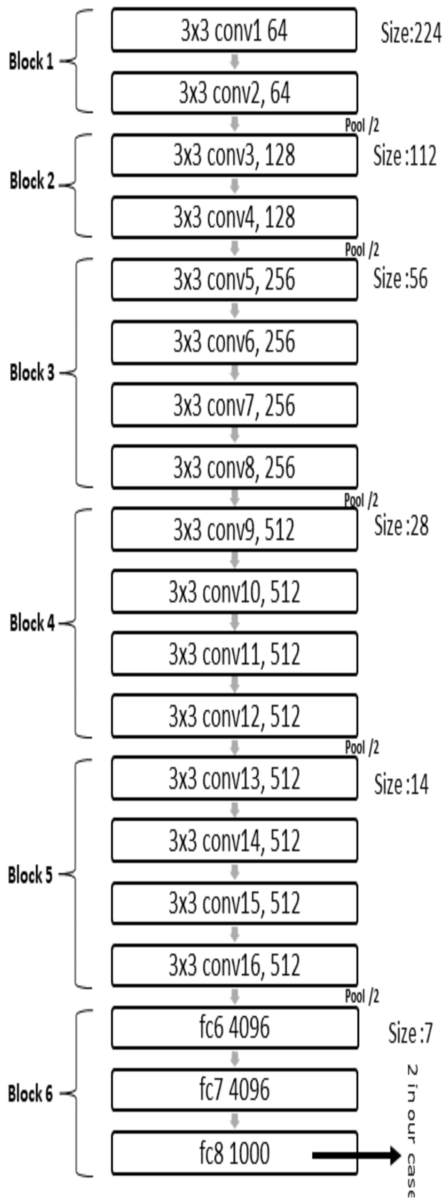

Figure 3.14: VGG19 model Architecture[8]

The key characteristics and features of VGG19 are:

- **Layer Depth:** VGG19 is characterized by its deep architecture, consisting of 19 layers (hence the name). It has 16 convolutional layers, followed by fully connected layers. The architecture's simplicity and uniformity in design make it easy to understand and implement.

- **Small Convolution Filters:** VGG19 predominantly uses 3x3 convolutional filters throughout the network. This small filter size helps capture finer details in images and allows for deeper network architectures.

- **Pooling Layers:** After every two or three convolutional layers, VGG19 uses max-pooling layers to downsample the spatial dimensions of feature maps. This helps in reducing the computational complexity and controlling overfitting.

- **Stacking Convolutions:** VGG19 stacks multiple convolutional layers before applying pooling. This design choice increases the depth of the network and enables it to learn hierarchical features from raw pixels.

- **Fully Connected Layers:** The final layers of VGG19 consist of fully connected layers followed by a softmax activation for classification. These layers make global decisions based on the features learned from earlier layers.

- **Pre-Trained Weights:** VGG19 is often initialized with pre-trained weights on large datasets like ImageNet. This initialization helps the network learn meaningful features from images and improves convergence during fine-tuning.

- **Vanishing Gradient:** Training very deep networks like VGG19 can suffer from vanishing gradients, where gradients become too small to effectively update the earlier layers. Techniques like weight initialization and careful training strategies are required to mitigate this issue.

- **High Memory and Computational Requirements:** Due to its depth and the use of small filters, VGG19 requires a considerable amount of memory and computation, making it less efficient for deployment on resource-constrained devices.

3.6.3 ResNet101V2

A member of the Microsoft Research ResNet family. Uses residual connections [9] to solve the vanishing gradient issue, making it simple to train very deep networks. Introduces skip connections that don't go through one or more levels in order to keep gradient flow and important information. Residual blocks enable training of networks with 100 or more layers without performance deterioration.Used to create architectures with excellent feature learning capabilities that are incredibly deep. The basic architecture of ResNet101V2 is illustrated in Figure 3.15

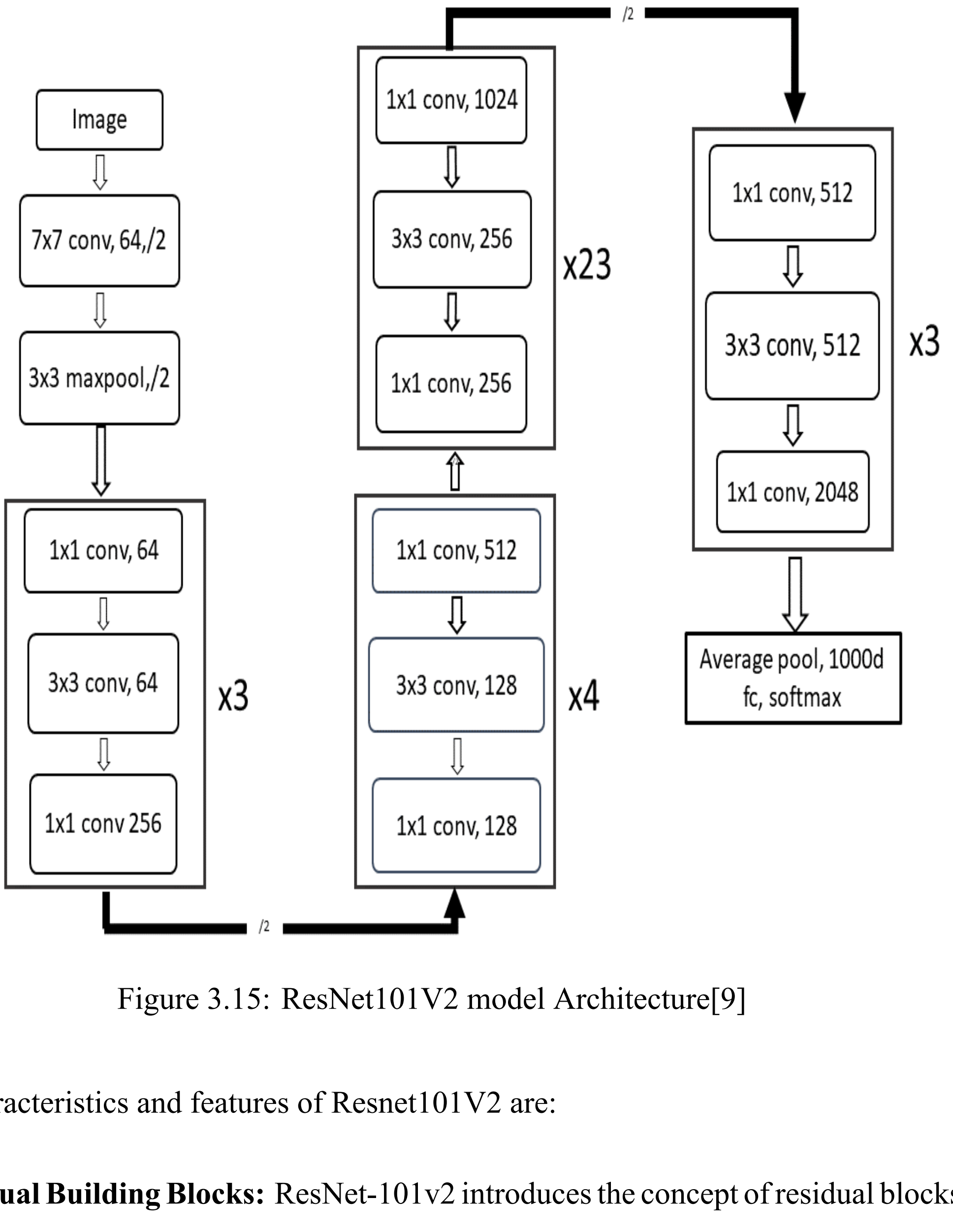

Figure 3.15: ResNet101V2 model Architecture[9]

The key characteristics and features of Resnet101V2 are:

- **Residual Building Blocks:** ResNet-101v2 introduces the concept of residual blocks. In a residual block, the input features are passed through a "shortcut connection" or "identity mapping" that bypasses one or more convolutional layers. This allows the network to learn residual functions, which simplifies training very deep networks.

- **Identity Mapping:** The identity mapping in a residual block ensures that the gradients can flow directly through the shortcut connection, avoiding the vanishing gradient problem. This enables the training of extremely deep architectures without degradation in performance.
- **Bottleneck Architecture:** ResNet-101v2 utilizes a bottleneck architecture in its deeper layers. A bottleneck block consists of three convolutional layers: 1x1, 3x3, and 1x1. The 1x1 convolutions are used to reduce and then increase the dimensions of the feature maps, reducing computational complexity.
- **Skip Connections:** The skip connections in ResNet-101v2 are the key to its success. These connections provide a direct path for gradients to flow backward during training, allowing for the training of very deep networks.
- **Improved Training Techniques:** ResNet-101v2 incorporates advanced training techniques such as batch normalization and improved weight initialization to further enhance training stability and convergence speed.
- **Deep Architecture:** ResNet-101v2 has a very deep architecture with 101 layers, including convolutional layers, pooling layers, and fully connected layers. This depth allows the network to learn complex hierarchical features from raw data.
- **Pre-Trained Weights:** Like other deep architectures, ResNet-101v2 can be initialized with pre-trained weights, often trained on large datasets like ImageNet. This initialization helps the network learn meaningful features and improves convergence.
- **High Computational Efficiency:** The bottleneck architecture in ResNet-101v2 reduces the computational cost of very deep networks compared to traditional architectures. This makes it more feasible to train and deploy on various platforms.

3.6.4 MobileNet V2

Designed by Google researchers for mobile and edge devices. Focuses on improving model speed and efficiency without sacrificing accuracy[11]. Divides the convolution procedure into a depthwise convolution and a pointwise convolution using depthwise separable convolutions.

MobileNetV2 is a popular deep learning architecture designed for efficient on-device computation and is widely used for mobile and embedded innovation applications. The choice of a specific deep learning architecture like MobileNet V2, InceptionNet V3, VGG19, or ResNet101V2 depends on several factors, including model efficiency, performance, and the requirements of the specific task. Here is the briefly description for choosing MobileNet V2 over the other architectures in the context of leukemia detection using the C-NMC 2019 dataset:

- **Efficiency:** MobileNet V2 is specifically designed for efficiency, making it well-suited for tasks where computational resources are limited. It uses depthwise separable convolutions and linear bottlenecks to reduce[11] the number of parameters and computational load while maintaining good accuracy. This can be important for real-time or resource-constrained applications.

- **Mobile and Embedded Applications:** If the intention is to deploy the model on mobile devices or embedded systems, MobileNet V2 is a strong choice[11]. Its architecture is optimized for such platforms, where computational and memory resources are more restricted compared to traditional desktop GPUs.

- **Good Balance of Accuracy:** MobileNet V2 offers a good trade-off between computational efficiency and accuracy. While it may not be the most accurate architecture compared to some of the deeper ones like ResNet101V2, it strikes a good balance, especially when dealing with medical image analysis tasks where accuracy is important but resource constraints exist.

- **Leukemia Detection Task:** Leukemia detection involves analyzing microscopic images of blood cells. MobileNet V2's efficiency and ability to capture essential features from images could be advantageous, as it can quickly process images while still making accurate predictions. Its depthwise separable convolutions can capture relevant features in small details, such as cell structures.

- **Reducing Overfitting:** In scenarios with limited data, deeper architectures like Inception Net V3 might be prone to overfitting. MobileNet V2, with its lower complexity, might be more robust in such cases.

It's important to note that the choice of architecture should be based on empirical experimentation and validation.

3.7 Data Augmentation Method

Data expansion may be a method for reducing overfitting in CNN models, which entails extending all of the preparation data by using it as though it were data [41].

We adjust the parameters of a machine learning computation so that it can map a certain input (let's say, an image) to a specific yield (let's say, a name). Finding the sweet spot when our show loses the smallest amount of information is our goal with optimization. This occurs when your settings are properly adjusted. In modern neural networks, there are typically millions of parameters. If we have several parameters, we'll need to display our machine learning system a proportionate number of examples to encourage good results.

We won't need to search for fresh pictures to put on our list. Since neurological systems aren't especially mental, to begin with. For instance, in a badly trained neural organization, three minor variations of the same picture seem as three distinct images. Therefore, all we need to do right now to encourage additional results is to make a few little adjustments to our existing dataset. Flips, interpretations, and revolutions are examples of little shifts. In any event, our brain system can convert them as recognizable images. Figure 3.16 illustrates how the information expansion preparation is laid up.

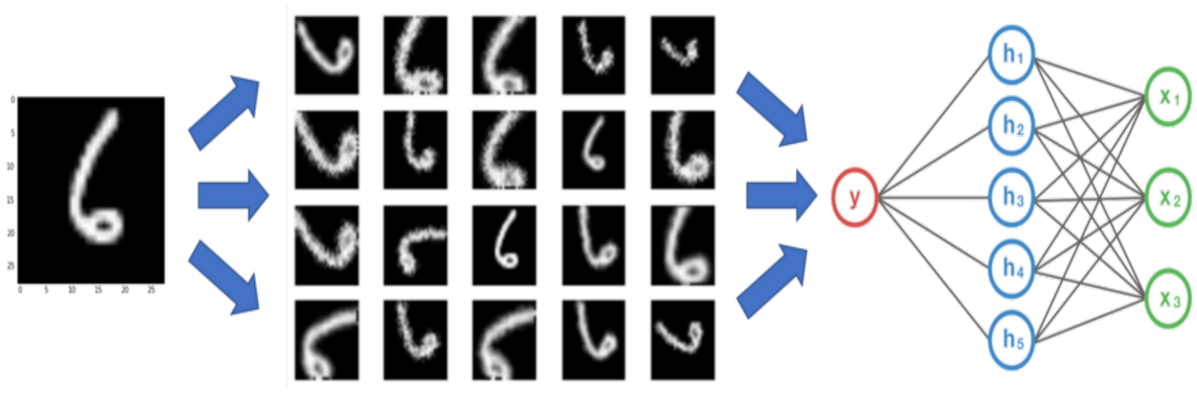

Figure 3.16: Data augmentation Method [10]

A convolutional neural network's invariance is a characteristic that enables it to recognize objects properly even when they are positioned in various orientations. Translation, perspective, and size may be disrespected by CNN in particular.

In general, data augmentation works in this way. We may have a collection of photographs taken in the real world under particular circumstances. On the other hand, our target program may arrive in a variety of forms, dimensions, scales, and brightness levels [10].

3.8 Summary

We used convolutional neural networks to classify ALL (Acute lymphocytic leukemia) in this experiment. CNN is a variant of the standard neural network that accepts visual input. The key layers that make up CNN architecture are built one on top of the other. By using the data augmentation strategy, we may increase our training data by removing over-fitting from the CNN model. All of the techniques we have covered in this chapter assist us in carrying out our investigation to detect leukemia.

Chapter 4

Methodology

4.1 Introduction

The explanations of the methodology for the whole study are the key topics of this part. Following that, a brief explanation of the evaluation criteria used in this study to analyze the model performances has also been provided. The next three subsections provide a full explanation of these approaches.

4.2 Working Principle

Leukemia detection using machine learning and deep learning techniques typically involves the following working principles:

1.Data Collection: The first step is to gather a dataset of medical images containing blood cell samples. In the case of leukemia detection, these images may include both healthy and malignant (leukemic) blood cells. For this research we collected data from C-NMC 2019 dataset.

2.Image Preprocessing: Several modifications and transformations are applied to pictures before they are input into a machine learning model for training or inference. Image preprocessing is the term for this. Figure 4.1 shows the image preprocessing steps. Preprocessing plays a crucial role in enhancing the quality of input data, improving model performance, and making the data more suitable for the specific requirements of the task at hand. This includes:

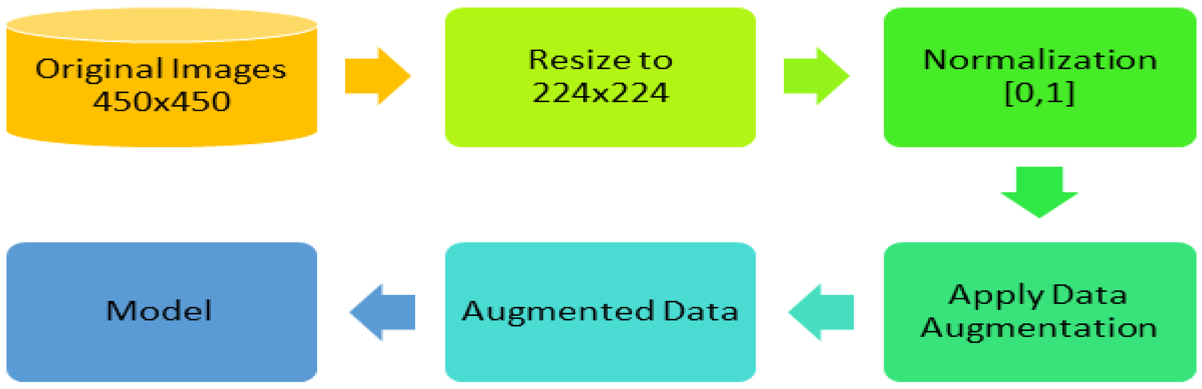

Figure 4.1: Image Preprocessing

Resizing the Input Images: From the C-NMC dataset, the images typically have dimensions of 450 × 450 pixels, and they feature a black background [42]. During the preprocessing stage, all images in the C-NMC 2019 dataset are adjusted to have a consistent size of 224 × 224 pixels. This resizing is likely done to ensure that all images fed into the model have the same dimensions.

Normalization: Normalizing pixel values to a specific range (usually [0, 1] or [-1, 1]) ensures that features have a similar scale. This aids convergence during training and prevents some features from dominating others. Here we use normalizing values to a specific range [0, 1].

Data Augmentation: Here we used data augmentation which involves creating new variations of existing images by applying transformations like rotation, flipping, cropping, and zooming. Example: With a probability greater than 70%, the image is centrally cropped by various fractions of its original size, creating different levels of zoom. Augmentation increases the diversity of training data, helping the model generalize better to different scenarios. Figure 4.2 shows the example of data augmentation technique on our dataset.

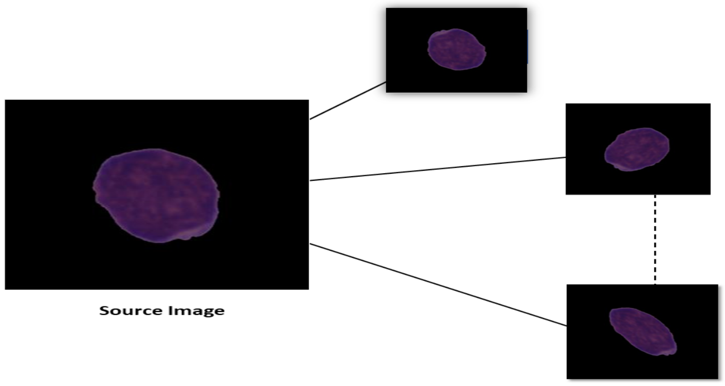

Figure 4.2: Data Augmentation Method

3.Data Splitting: The dataset is divided into two subsets: a training set and a test set. The training set is used to train the machine learning model, while the test set is reserved for evaluating the model's performance.

4.Model Training: The selected model is trained on the labeled training dataset. During training, the model learns to recognize patterns and features that distinguish between healthy and leukemic cells.

5.Validation: The model's performance is monitored on a separate validation dataset to ensure it's not overfitting the training data. Hyperparameter tuning may be performed during this stage to optimize the model.

6.Testing: The trained model is evaluated using the reserved test dataset. Performance metrics such as accuracy, F1-score, precision, recall, and ROC-AUC are calculated to assess the model's ability to detect leukemia accurately.

4.3 MobileNet V2

MobileNetV2 is a popular convolutional neural network (CNN) architecture designed for efficient and lightweight deep learning models[11]. MobileNetV2 builds upon the original MobileNet architecture, aiming to improve accuracy while maintaining efficiency.MobileNetV2 has become a popular choice in the field of deep learning due to its ability to deliver good accuracy with a relatively small computational footprint. It strikes a balance between model size and performance, making it valuable for resource-constrained environments, including mobile devices. Figure 4.3 shows the MobileNet V2 architecture.

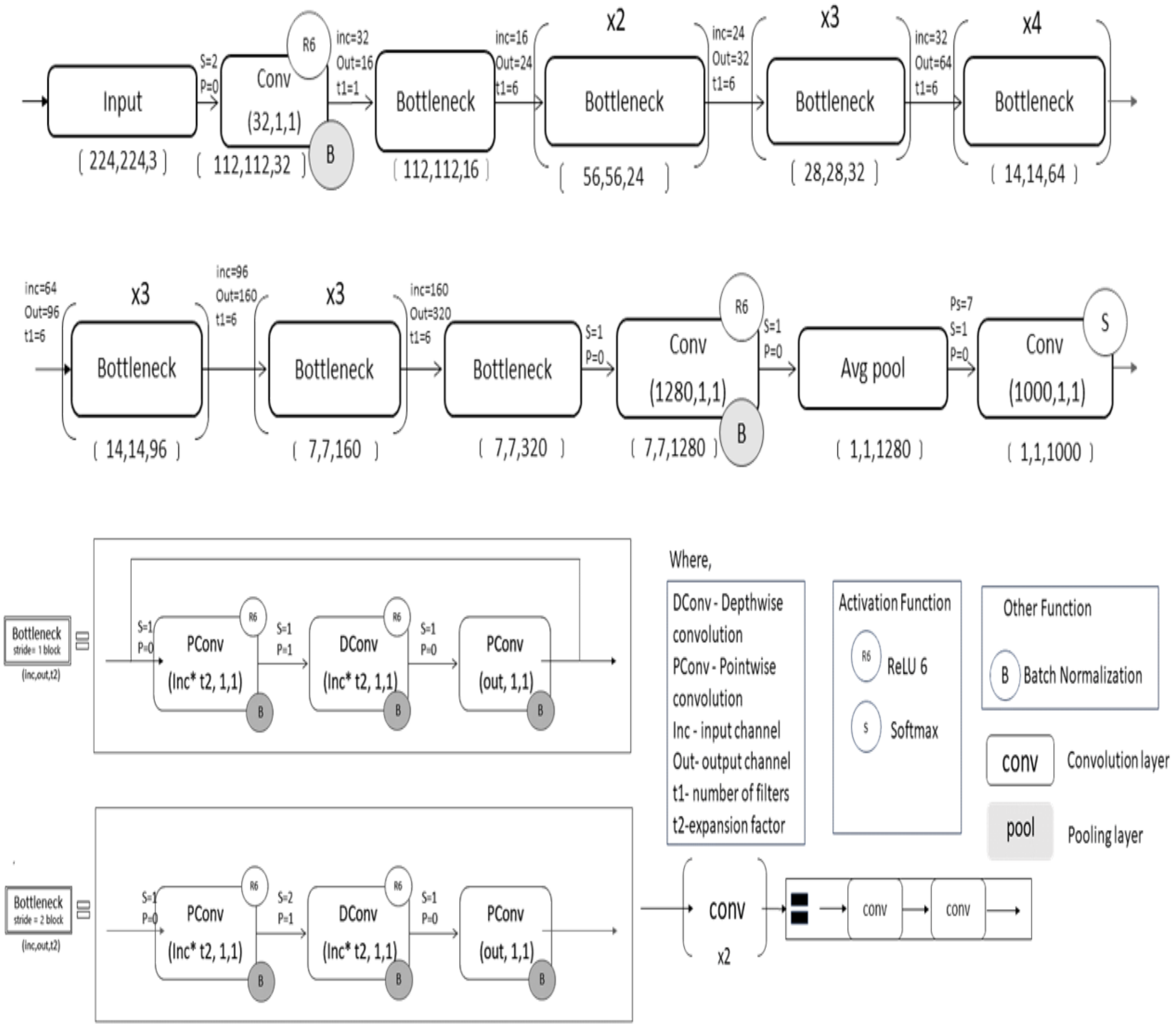

Figure 4.3: MobileNet V2 model Architecture[11]

Here is an overview of the MobileNetV2 architecture:

- **Depthwise Separable Convolutions:** MobileNetV2 employs depthwise separable convolutions, which split the traditional convolution into two parts: a depthwise convolution and a pointwise convolution. This reduces the number of computations by separating spatial filtering (depthwise) from the mixing of information (pointwise). This approach significantly reduces the number of parameters and operations.
- **Inverted Residual Bottleneck Blocks:** MobileNetV2 introduces inverted residual blocks, which consist of three main components: a depthwise convolution, a pointwise convolution, and a linear bottleneck. The linear bottleneck reduces the number. of channels

before the expensive 3x3 convolution, improving efficiency while preserving representational capacity.

- **Expansion Layer:** The expansion layer in each inverted residual block increases the number of channels before applying the depthwise convolution. This expansion allows the network to learn more complex features.
- **Skip Connections:** MobileNetV2 includes skip connections that connect the output of an inverted residual block to a later stage in the network. This helps propagate gradients and improves the flow of information through the network.
- **Global Depthwise Pooling:** In the final layers of MobileNetV2, global depthwise pooling is applied. This reduces the spatial dimensions of the feature maps to 1x1, capturing a global view of the input and enabling classification.
- **Efficient Operations:** MobileNetV2 uses efficient operations such as linear bottlenecks, 1x1 convolutions, and non-linear activations to optimize performance. The use of these operations contributes to the overall computational efficiency of the architecture.

Compare to MobileNetV1, MobileNetV2 builds upon the foundation of MobileNetV1 by introducing advanced components like inverted residual blocks, linear bottlenecks, and skip connections. These enhancements lead to improved feature representation, increased accuracy, and better utilization of computational resources. MobileNetV2 is generally considered a more advanced and capable architecture for tasks where both efficiency and accuracy are important, making it a strong choice for applications like image classification, object detection, and medical image analysis such as leukemia detection.

4.3.1 Model Compilation

Model compilation process includes:

- **Loss function:** For this study we used Cross-Entropy as loss function. Cross entropy is used to compare the output distribution that the model actually predicts to what it actually is. It is defined as,

$$H(p,q) = -\sum_{i}^{n} y^i \log(p^i) \tag{4.1}$$

A common alternative to squared error is the cross-entropy test. When the output is a probability distribution, i.e. when node activations can be interpreted as representing the probability that each hypothesis is valid, cross-entropy is used. As a consequence, in neural networks with softmax activations in the output layer, it is used as a loss function.

- **Optimizer:** At first we tested different optimizers and finally set Adam optimizer in our model Adam (Adaptive Moment Estimation) is an effective optimizer. It is a stochastic gradient descent optimization process that combines the principles of Momentum and Root Mean Square Propagation (RMSprop), two well-known optimization strategies. Adam dynamically adjusts the learning rate during training using moving averages of the gradient and the squared gradient, which helps to avoid the optimization being trapped in suboptimal solutions or fluctuating between many solutions. Additionally, the algorithm has a bias adjustment mechanism that enables it to estimate the moving averages more accurately in the early stages of training, when the starting values are still having a significant impact on them. Adam is frequently utilized in various deep learning applications because to its effectiveness and strong performance. Our adam optimizer learning rate was 0.01. This study we used Cosine annealing for learning rate scheduling technique. Cosine annealing is a learning rate scheduling technique commonly used in training deep learning models, especially in the context of neural network optimization.

In the context of training neural networks, cosine annealing typically follows a cyclical pattern where the learning rate starts with a relatively high value, gradually decreases, and then increases again. The learning rate schedule resembles the shape of a cosine curve. Cosine annealing phases:

A common alternative to squared error is the cross-entropy test. When the output is a probability distribution, i.e. when node activations can be interpreted as representing the probability that each hypothesis is valid, cross-entropy is used. As a consequence, in neural networks with softmax activations in the output layer, it is used as a loss function.

High Learning Rate Phase: Initially, the learning rate is set to a relatively high value. This phase is designed to help the model explore a larger portion of the parameter space and escape potential local minima.

Annealing Phase: Over the course of several epochs, the learning rate is gradually reduced. The rate of reduction follows a cosine curve pattern. This phase allows the model to fine-tune its parameters with smaller steps, focusing on convergence.

Low Learning Rate Phase: The learning rate eventually reaches its lowest value. This phase aims to facilitate precise convergence and enable the model to settle into a well-generalized solution.

Restart: After completing one cycle of cosine annealing, some implementations restart the learning rate schedule. This can be repeated for a specified number of cycles.

Cosine annealing offers several benefits:

Smooth Transition: The gradual change in the learning rate ensures a smooth transition between exploration and convergence, potentially leading to better model performance.

Adaptive Learning Rate: The learning rate is adjusted automatically, reducing the need for manual tuning during training.

Improved Generalization: By allowing the model to explore and converge at different phases of training, cosine annealing can help the model escape sharp local minima and improve generalization.

Noise Robustness: The cyclical nature of the learning rate schedule can make the optimization process more robust to noisy gradients and fluctuations.

Figure 4.4 shows the learning rate schedule when using Cosine annealing.

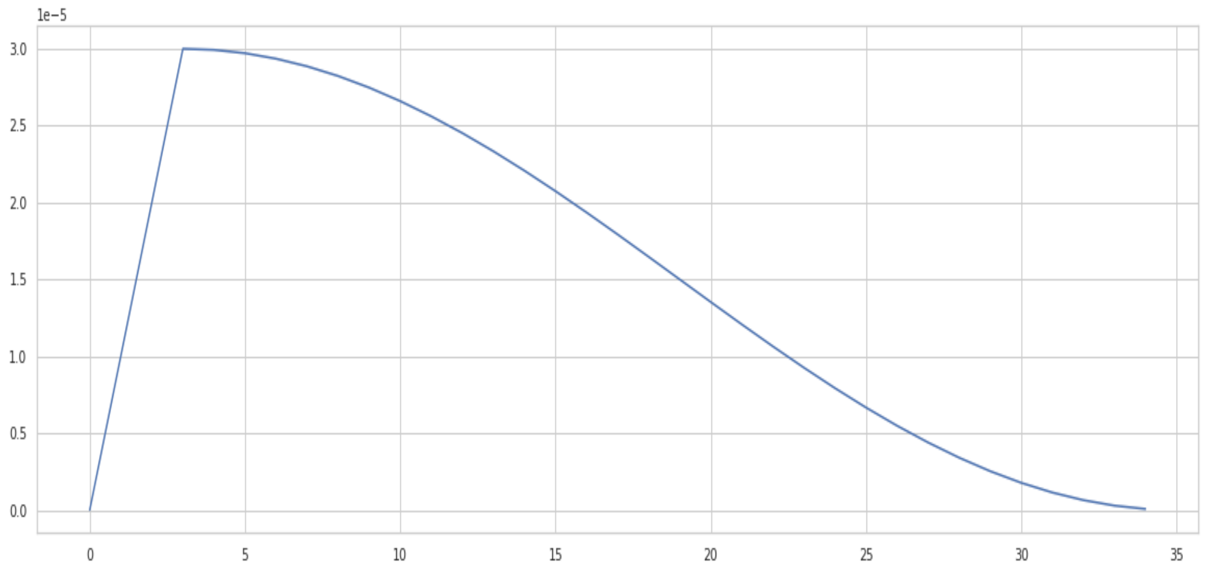

Figure 4.4: The learning rate schedule

Dropout rate: Dropout is a regularization approach in deep learning that helps to prevent overfitting in neural networks. The idea behind dropout is to randomly ”drop out” (i.e., set to zero) some of the activations in the network during training. This has the effect of forcing the network to be more robust and avoid relying too heavily on any one neuron. The dropout rate is the fraction of activations that are dropped out during each iteration of training. For example, if the dropout rate is set to 0.5, then during each training iteration, approximately 50% of the activations in the network will be set to zero. The dropout rate is usually set between 0.1 and 0.5, and it is typically set lower for smaller networks and higher for larger networks. Dropout is usually applied only during training and is not used during testing or inference. We set our dropout rate initially 0.25. Setting the dropout rate to 0.25 serves as a regularization technique to enhance the generalization capability of the model during training.

Batch size: The batch size in deep learning determines the number of samples that are processed before the model's weights are updated. The batch size plays an important role in the training process, as it affects the model's convergence speed, computational efficiency, and stability of the optimization.

A smaller batch size can result in faster convergence, but it can also result in increased noise in the weight updates, which can slow down convergence or lead to instability in the optimization. A larger batch size can provide more accurate gradients and result in more stable optimization, but it can also slow down convergence and increase memory requirements.

Choosing an appropriate batch size is a trade-off between these factors and can depend on the size and complexity of the dataset, as well as the hardware resources available for training. Common batch sizes in deep learning range from 32 to 256, and the optimal batch size can be determined through experimentation and validation. But we set our batch size is 32.

Activation function: For DL (deep learning), each neuron's output from a neural network is subjected to a mathematical function known as an activation function. The purpose of the activation function is to introduce non-linearity into the output of the neuron, allowing the network to learn a wider range of complex functions and representations. As our model is for segmentation task, it needs to classify only two color in mask image like white and black so we have used sigmoid function as it is one type of binary classification. Sigmoid activation function maps inputs values to the range [0, 1], we choose it because leukemia is a binary classification problem.

4.3.2 Model Training

Split the dataset into training, validation, and test sets. It involves dividing the dataset into distinct subsets to perform different stages of model development, evaluation, and testing.The goal of splitting the dataset into these subsets is to create a rigorous evaluation pipeline:

Training Phase: The model learns from the training set to understand the underlying patterns in the data and adjust its parameters.
Validation Phase: The model is evaluated on the validation set to fine-tune hyperparameters and monitor performance. Adjustments to the model can be made based on validation results.
Test Phase: The model's final performance is assessed on the test set, which provides an unbiased evaluation of its generalization capabilities. This step measures how well the model can predict outcomes on completely new data.
Train the MobileNetV2 model using the training data. Monitor the validation performance during training to avoid overfitting.

4.3.3 Model Evaluation

Evaluate the trained model using the test dataset to assess its generalization performance. Calculate relevant metrics (e.g., accuracy, precision, recall) for specific task.

4.4 Experimental settings

Set up development environment with the required libraries like TensorFlow, Keras and PyTorch. The table shows 4.1 the experimental settings.

Table 4.1: Experimental settings

Experimental Settings	
Python Version	3.11.1
Neural Network Framework	TensorFlow
GPU	Tesla T4
GPU Memory Capacity	16 GB
Environment	Google Colaboratory
System RAM	12.7 GB
Disk Space	78.2 GB

4.5 Summary

This chapter discusses the model architecture we proposed and evaluated on the leukaemia dataset. We made an attempt to provide a thorough description of the architecture's elements, such as the mobilenet v2, which aid in the creation of the final classification model. Afterwards, we spoke about the hyper-parameter condition in our model. Finally, we spoke about the assessment criteria that allow us to gauge how well the model performs across various parameters. Finally we narrate our experimental setup and necessary library function plus framework in detail.

Chapter 5

Experimental Results and Analysis

5.1 Introduction

The performance of the proposed Leukaemia detection model using the MobileNet V2 architecture was rigorously evaluated and compared against other state-of-the-art models, including VGG19, InceptionNet V3, and ResNet101v2, on the C-NMC 2019 dataset. The obtained results underscore the effectiveness of the MobileNet V2 model in accurately detecting leukemia cases.

5.2 Dataset Description & pre-processing

The dataset utilized in this work was given by the SBILab research team [42]. The C-NMC 2019 dataset [42] contains 15,114 lymphocyte pictures obtained from 118 patients. These photos were divided into three sets: training, preliminary, and test. Each image collection contains single-cell images of healthy or malignant cells that have been previously tagged by an oncology team. The SBILab team used segmentation, image enhancement, and normalization techniques to preprocess these pictures [43],[44],[45]. Individual cells were segmented and placed in the center of blood smear photographs; each image has 450×450 pixels and a black backdrop. This dataset contains samples of both healthy and cancerous cells, as seen in Figure 5.1.

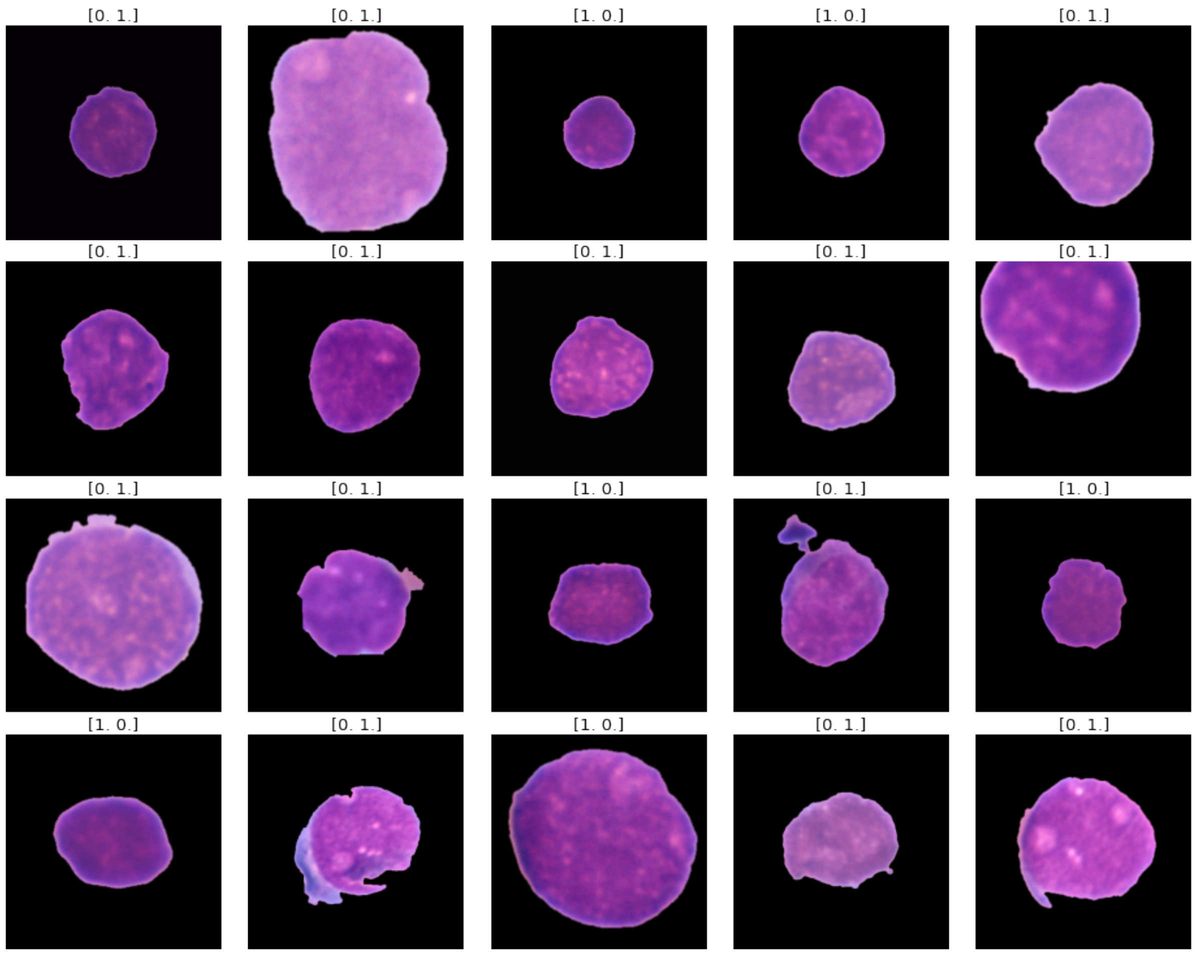

Figure 5.1: C-NMC 2019 dataset samples.

The images are BMP format. The dataset contains a total of 118 (ALL (cancer) patents: 69 and HEM (Normal) patients: 49) individual patents, distributed as follows:

For train set composition: Total patents are 73 (ALL: 47, Normal: 26) and total cells are 10,661 (ALL: 7272, Normal: 3389). Figure **??** shows the Class distribution of the C-NMC 2019 dataset for train set.
Test set composition: Total patents are 28 (ALL: 13, Normal: 15) and total cells are 1867 (ALL: 1219, Normal: 648) Here ALL is malignant cells and HEM is healthy cells.

5.3 Performance Analysis of Train dataset

The models are trained on the Kaggle platform following the completion of the data augmentation phase. An Nvidia P100 GPU with 16 gigabytes of GPU memory and 12 gigabytes of RAM makes up the Kaggle kernel. Cross-validation training has been performed on each model, using 35 epochs for each fold. Adam is put into practice as an optimizer, and binary cross entropy is used as a loss function during model training. Table 5.1 contains the score(after 105 epochs):

Table 5.1: Comparison of model accuracy, loss and F1-Score on train dataset.

Model	**Accuracy**	**Loss**	**Validation Accuracy**	**Validation Loss**	**F1-Score**	**Validation F1-Score**
InceptionNet V3	0.9872	0.0249	0.9685	0.1034	0.9872	0.9684
VGG19	0.9521	0.1017	0.9449	0.1445	0.9623	0.9449
ResNet101V2	0.9873	0.0349	0.9575	0.1579	0.9881	0.9569
MobileNet V2	0.9557	0.1136	0.9519	0.1316	0.9558	0.9516

InceptionNet V3: It can be stated that InceptionNet V3 achieved a training accuracy of 98.72% and a validation accuracy of 96.85%. In the both case, compared to other trained model, InceptionNet V3 provides the highest training accuracy and validation accuracy. And also provides lowest training loss and validation loss. According to the training accuracy and loss graph of InceptionNet V3 shown in Figure 5.2

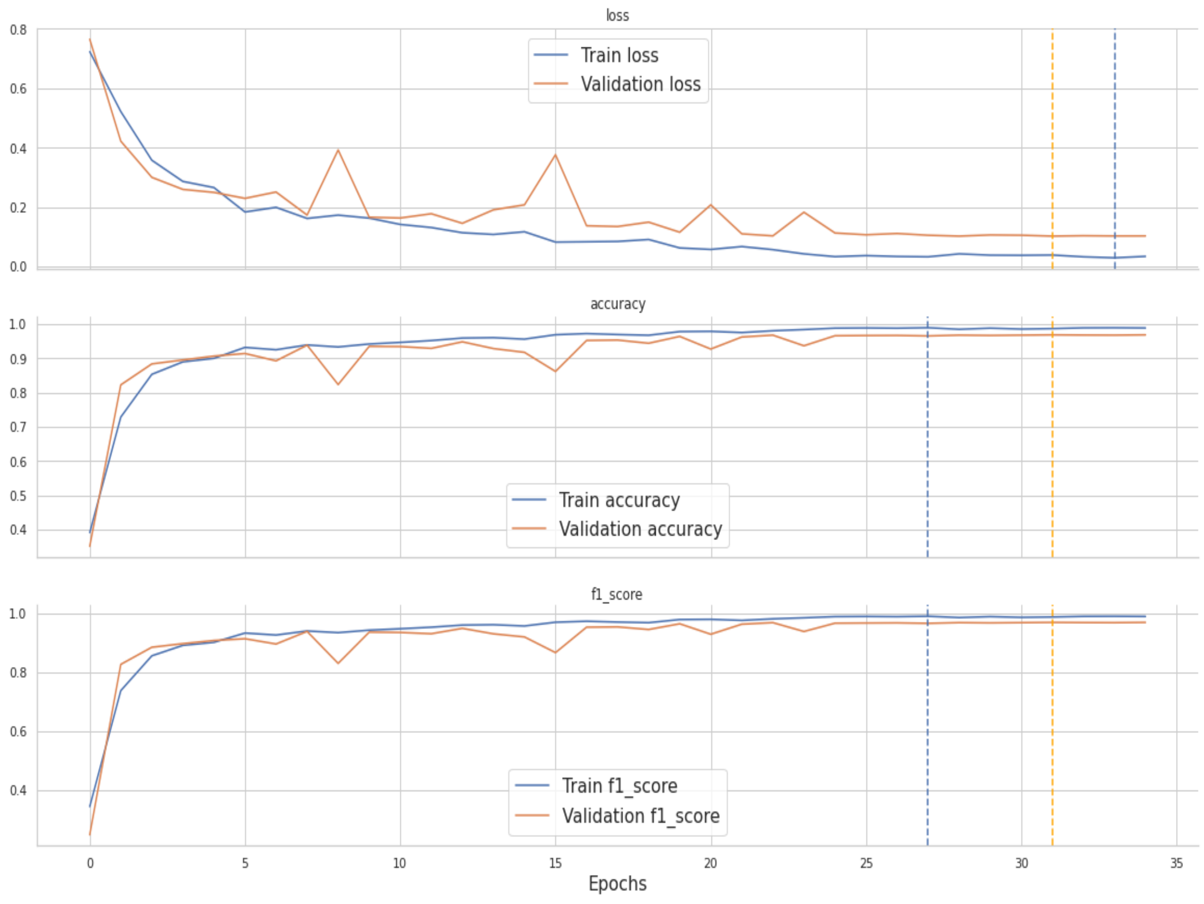

Figure 5.2: Training and Validation graph of InceptionNet V3

.

VGG19: We can easily find that VGG19 achieved a training accuracy is 95.21% and a validation accuracy of 94.49%, the lowest training accuracy and validation accuracy compared to other trained models. VGG19 has the highest training loss and validation loss. According to the training accuracy and loss graph of VGG19 shown in Figure 5.3

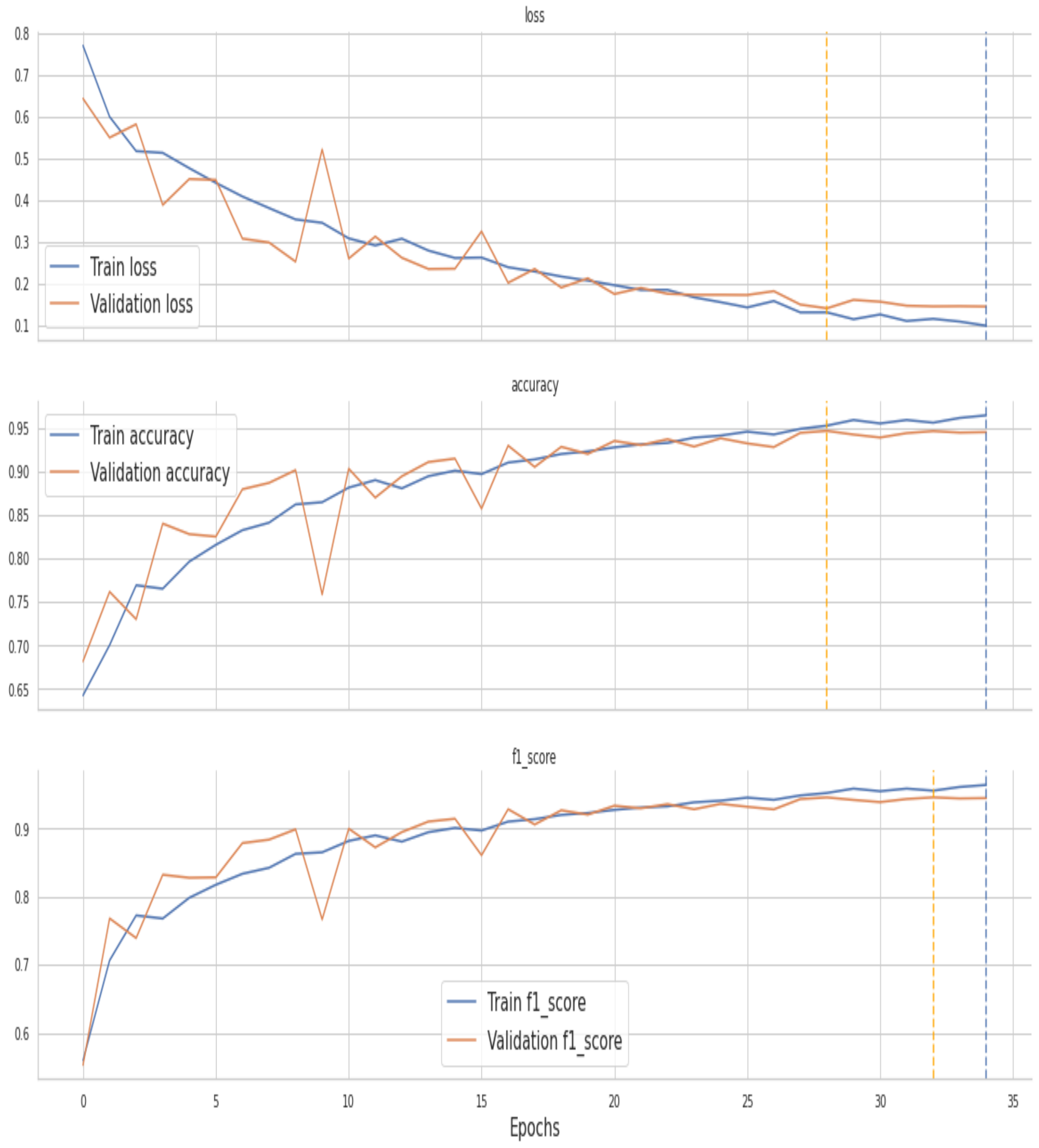

Figure 5.3: Training and Validation graph of VGG19

ResNet101V2: ResNet101V2 performed well for the training dataset, achieved a training accuracy is 98.71% and a validation accuracy of 95.75%. It has the 3.49% training loss and 15.79% validation loss. According to the training accuracy and loss graph of ResNetV2 shown in Figure 5.4

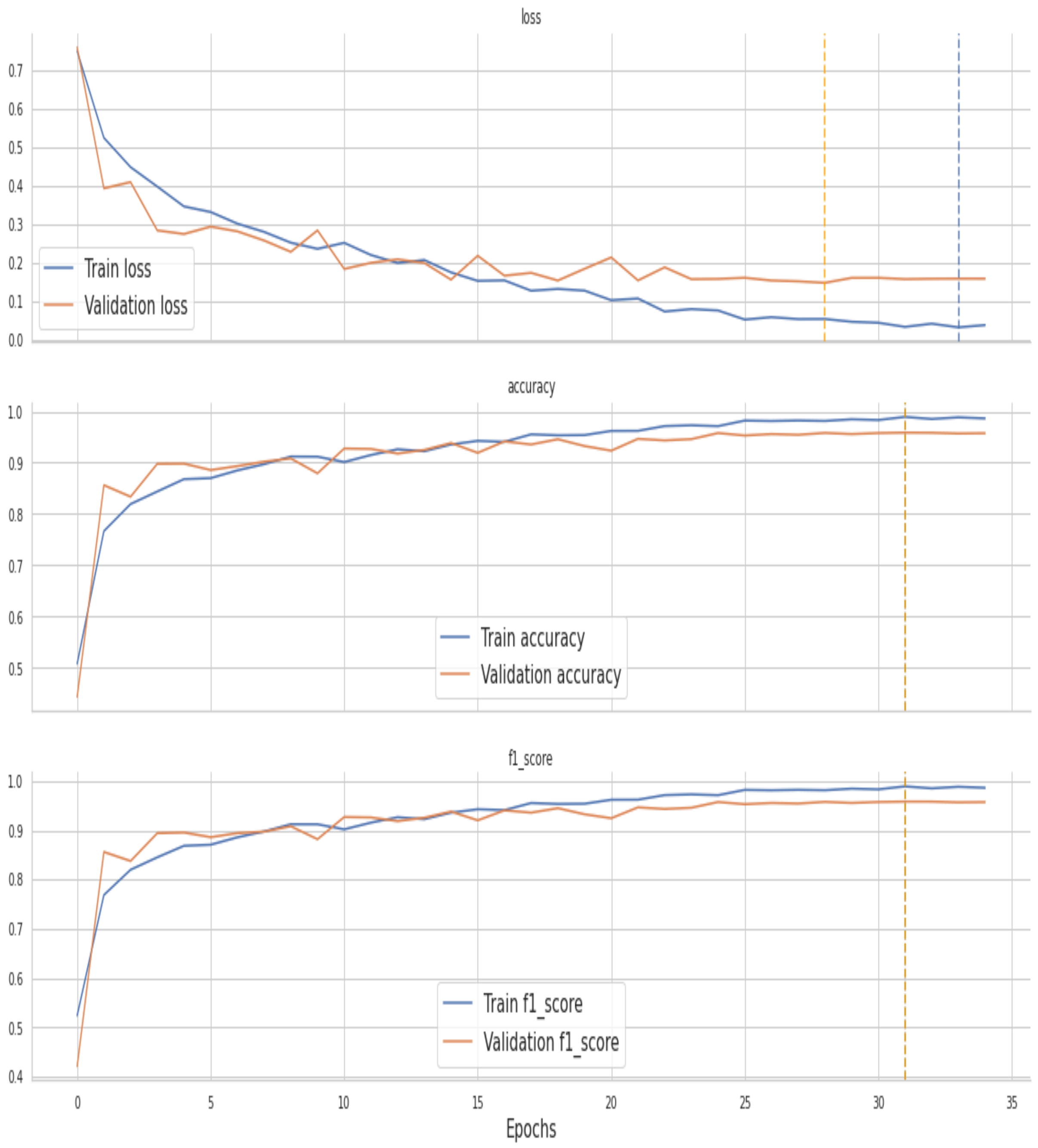

Figure 5.4: Training and Validation graph of ResNet101V2

MobileNet V2: These results suggest that the MobileNet V2 model is performing well on both the training and validation datasets, attained a training accuracy of 95.57% and a validation accuracy of 95.19%. The relatively high accuracy and low loss values indicate that the model is able to capture important patterns and features in the data, making accurate predictions. However, it's important to note that while high performance on the training and validation data is positive, the model's performance on an independent test set is crucial to assess its generalization ability to new, unseen data and to avoid overfitting. According to the training accuracy and loss graph of MobileNet V2 (Proposed Model) shown in Figure 5.5

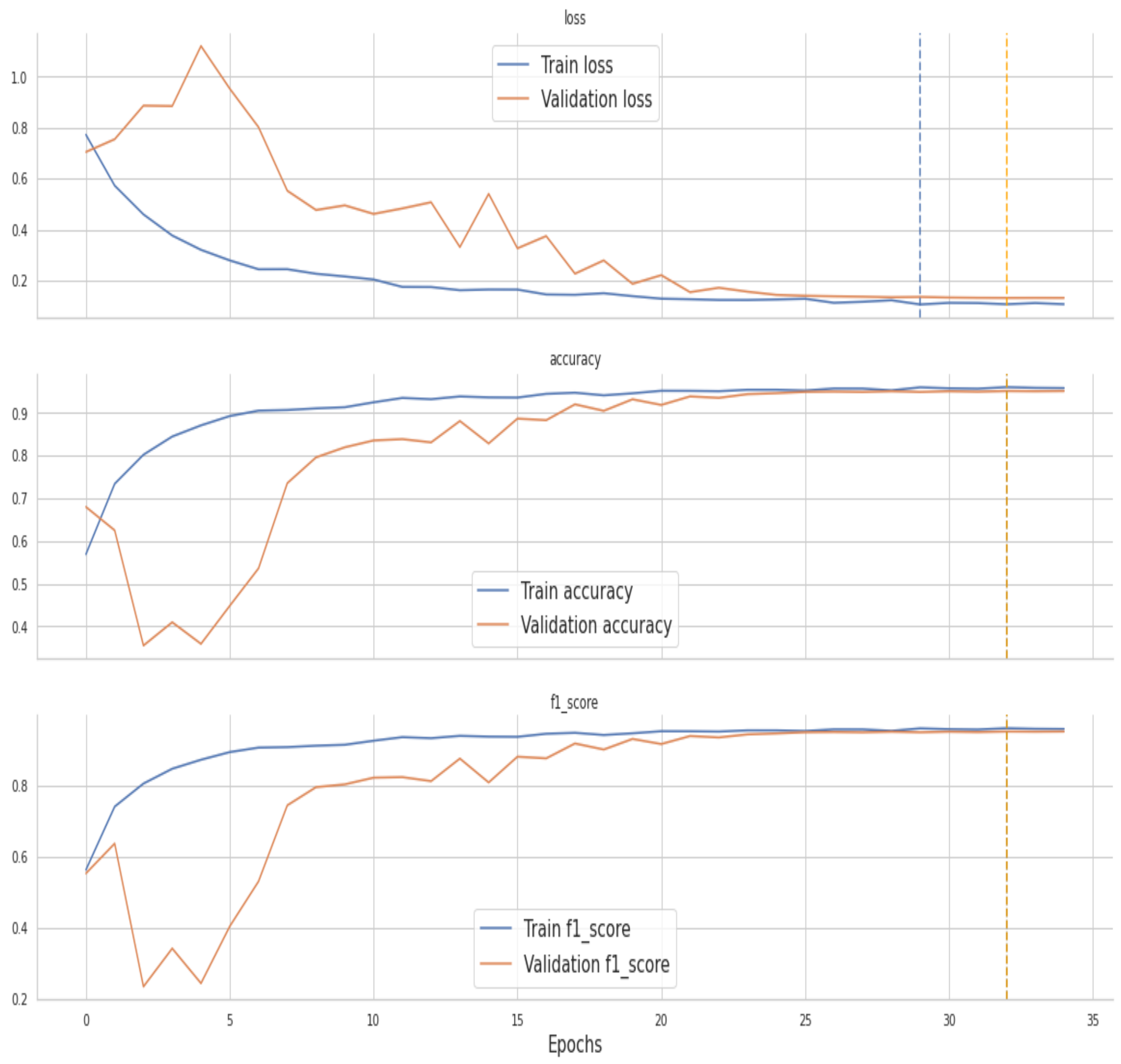

Figure 5.5: Training and Validation graph of MobileNet V2

5.4 Performance Analysis on test Dataset

The performance in the unidentified dataset is used to measure test set accuracy, which is crucial in the categorization of medical images. The comparison of the accuracy and F1-Score on test dataset is illustrated in Table 5.2

Table 5.2: The comparison of the Accuracy and F1-score on test dataset

Performance Analysis		
Model	**Accuracy (%)**	**F1_score (%)**
Inception net v3	80.24	79.96
VGG19	77.40	76.37
ResNet101V2	77.18	76.19
Mobile net v2	82.75	82.48

This table provides a comprehensive comparative analysis of four deep learning models in the context of leukaemia detection, focusing on their performance metrics, namely accuracy and F1 score. The models under scrutiny include InceptionNet V3, VGG19, ResNet101V2, and an original architecture termed MobileNet V2, proposed specifically for this research endeavour.

InceptionNet V3 achieved an accuracy of 80.24% and an F1 score of 79.96%, denoting its proficiency in accurately identifying leukaemia cases while maintaining a balanced trade-off between precision and recall. VGG19 exhibited an accuracy of 77.40% and an F1 score of 76.37%, displaying slightly lower performance compared to InceptionNet V3. Similarly, ResNet101V2 presented comparable results with an accuracy of 77.18% and an F1 score of 76.19%.

Notably, the novel MobileNet V2 model demonstrated the highest performance among all models, attaining an accuracy of 82.75% and an F1 score of 82.48%. These outcomes underscore the superiority of MobileNet V2 in both overall accuracy and its ability to strike an optimal balance between precision and recall in leukemia detection.

The comparative analysis unequivocally establishes MobileNet V2 as the most effective model for leukaemia detection in this study, outperforming InceptionNet V3, VGG19, and ResNet101V2. These findings underscore the critical significance of selecting an appropriate deep learning architecture tailored to the specific medical imaging task at hand, as model choice significantly influences performance outcomes. The remarkable efficacy of MobileNet V2 positions it as a promising and invaluable tool for practical applications in leukaemia detection, where accurate and reliable classification holds paramount importance. As the field of deep learning continues to evolve, this insightful comparison will serve as a valuable reference for researchers and practitioners seeking to deploy deep learning models for leukaemia detection tasks.

5.5 Summary

The chapter 5 started with the description of dataset and pre-processing methods for this dataset. We get proper idea of our model novelty by comparative studies between proposed model and previous studies. This comparative study represents that our model accuracy and F1-score criteria. The accuracy metrics almost equal to different study done before.

Chapter 6

Conclusion and Future Works

6.1 Conclusions

In this study, we investigated the efficacy of the MobileNet V2 model for the critical task of leukemia detection using the C-NMC 2019 dataset. Our results highlight the significant strides made in leveraging deep learning techniques for medical image analysis. Through an in-depth analysis and comparison, we shed light on the strengths and weaknesses of different models, focusing on accuracy and F1-score as key evaluation metrics.

The MobileNet V2 model emerged as a standout performer in our comparative study, demonstrating an accuracy of 82.75% and an F1-score of 82.48% on the C-NMC 2019 dataset. These results underscore the model's exceptional ability to differentiate between leukemia and non-leukemia cases, exhibiting its potential as a valuable tool for aiding medical professionals in diagnosis.

The MobileNet V2 model's higher classification abilities are highlighted by our comparison with InceptionNet V3, VGG19, and ResNet101v2. The model's design, which has been streamlined for quick feature extraction and classification, aligns well with the features of the C-NMC 2019 dataset.

The success of our study has promising implications for the field of medical imaging and disease diagnosis. By achieving high accuracy and F1-scores, we pave the way for more accurate and reliable leukemia detection, reducing the chances of misdiagnosis and enabling timely interventions. Furthermore, our findings highlight the importance of tailored model selection and hyperparameter tuning to achieve optimal results on specific medical datasets.

While this study brings forward valuable insights, it also sets the stage for future research directions. Fine-tuning the MobileNet V2 model to accommodate larger datasets and exploring transfer learning from related medical image datasets could yield even more robust results. Additionally, integrating ensemble techniques and interpretability methods could further enhance the model's performance and our understanding of its decision-making process.

6.2 Future works

Although the suggested MobileNet V2 model was developed using a smaller, balanced dataset for training, the same network might be tested with larger, unbalanced datasets in the future to determine how well it performs in both the same problem area and in other domains. Additionally, utilizing the same framework or variations of the suggested framework, the study work can be expanded to detect all sub-types of ALL, all types of leukaemia, detection of AML, and other diseases.

REFERENCES

[1] "Api.intechopen.com.," Available Link: https://api.intechopen.com/media/chapter/39071/media/f2, [Online] Accessed:Nov 1,2020.

[2] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[3] J. J. Fei-Fei Li and S. Yeung, "Convolutional neural networks for visual recognition," CS231n: Available Link: https://cs23 ln.github.io/ [Online].

[4] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proceedings of 2010 IEEE international symposium on circuits and systems*, pp. 253–256, IEEE, 2010.

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

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

[7] L. D. Nguyen, D. Lin, Z. Lin, and J. Cao, "Deep cnns for microscopic image classification by exploiting transfer learning and feature concatenation," 2018.

[8] A. Khattar and S. Quadri, "Generalization of convolutional network to domain adaptation network for classification of disaster images on twitter," *Multimedia Tools and Applications*, vol. 81, no. 21, pp. 30437–30464, 2022.

[9] A. Khan, M. A. Khan, M. Y. Javed, M. Alhaisoni, U. Tariq, S. Kadry, J.-I. Choi, and Y. Nam, "Human gait recognition using deep learning and improved ant colony optimization.," *Computers, Materials & Continua*, vol. 70, no. 2, 2022.

[10] A. Gandhi, “Augmentation how to use deep learning when you have limited data,” 2017. Nanonets.com [Online]. Avaiable Link: https://nanonets.com/blog/data- augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2.

[11] “Mobilenet-v2:summary and implementation,” Available link: https://hackmd.io/@machine-learning/ryaDuxe5L [Online].

[12] J. Laosai and K. Chamnongthai, “Classification of acute leukemia using medical-knowledge-based morphology and cd marker,” *Biomedical Signal Processing and Control*, vol. 44, pp. 127–137, 2018.

[13] L. H. Vogado, R. M. Veras, F. H. Araujo, R. R. Silva, and K. R. Aires, “Leukemia diagnosis in blood slides using transfer learning in cnns and svm for classification,” *Engineering Applications of Artificial Intelligence*, vol. 72, pp. 415–422, 2018.

[14] T. Tran, O.-H. Kwon, K.-R. Kwon, S.-H. Lee, and K.-W. Kang, “Blood cell images segmentation using deep learning semantic segmentation,” in *2018 IEEE international conference on electronics and communication engineering (ICECE)*, pp. 13–16, IEEE, 2018.

[15] A. Holland and J. Bare, “Inhibition of proliferation of acute lymphocytic leukemia (all) by frequency-specific oscillating pulsed electric fields (opef) broadcast by an enclosed gas plasma antenna,” *bioRxiv*, pp. 2023–05, 2023.

[16] A. Rehman, N. Abbas, T. Saba, S. I. u. Rahman, Z. Mehmood, and H. Kolivand, “Classification of acute lymphoblastic leukemia using deep learning,” *Microscopy Research and Technique*, vol. 81, no. 11, pp. 1310–1317, 2018.

[17] N. Abbas, T. Saba, Z. Mehmood, A. Rehman, N. Islam, and K. T. Ahmed, “An automated nuclei segmentation of leukocytes from microscopic digital images.,” *Pakistan Journal of Pharmaceutical Sciences*, vol. 32, no. 5, 2019.

[18] S. Shirian, S. Ebrahimi-Barough, H. Saberi, A. Norouzi-Javidan, S. M. M. Mousavi, M. A. Derakhshan, B. Arjmand, and J. Ai, “Comparison of capability of human bone marrow mesenchymal stem cells and endometrial stem cells to differentiate into motor neurons on electrospun poly (ε-caprolactone) scaffold,” *Molecular neurobiology*, vol. 53, pp. 5278–5287, 2016.

[19] S. Mohapatra, S. S. Samanta, D. Patra, and S. Satpathi, "Fuzzy based blood image segmentation for automated leukemia detection," in *2011 International Conference on Devices and Communications (ICDeCom)*, pp. 1–5, IEEE, 2011.

[20] T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler, "Support vector machine classification and validation of cancer tissue samples using microarray expression data," *Bioinformatics*, vol. 16, no. 10, pp. 906–914, 2000.

[21] M. Madhukar, S. Agaian, and A. T. Chronopoulos, "New decision support tool for acute lymphoblastic leukemia classification," in *Image processing: Algorithms and systems X; and parallel processing for imaging applications II*, vol. 8295, pp. 367–378, SPIE, 2012.

[22] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, pp. 281–297, Oakland, CA, USA, 1967.

[23] S. Ondimu and H. Murase, "Effect of probability-distance based markovian texture extraction on discrimination in biological imaging," *Computers and Electronics in Agriculture*, vol. 63, no. 1, pp. 2–12, 2008.

[24] M. D. Joshi, A. H. Karode, and S. Suralkar, "White blood cells segmentation and classification to detect acute leukemia," *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, vol. 2, no. 3, pp. 147–151, 2013.

[25] M. K. Hasan, M. A. Ahamed, M. Ahmad, M. Rashid, *et al.*, "Prediction of epileptic seizure by analysing time series eeg signal using-nn classifier," *Applied bionics and biomechanics*, vol. 2017, 2017.

[26] S. Mishra, B. Majhi, and P. K. Sa, "Texture feature based classification on microscopic blood smear for acute lymphoblastic leukemia detection," *Biomedical Signal Processing and Control*, vol. 47, pp. 303–311, 2019.

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

[28] Y. Ding, Y. Yang, and Y. Cui, “Deep learning for classifying of white blood cancer,” in *ISBI 2019 C-NMC Challenge: Classification in Cancer Cell Imaging: Select Proceedings*, pp. 33–41, Springer, 2019.

[29] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[30] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[31] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, 2017.

[32] B. Gao and L. Pavel, “On the properties of the softmax function with application in game theory and reinforcement learning,” *arXiv preprint arXiv:1704.00805*, 2017.

[33] L. Paninski, “Estimation of entropy and mutual information,” *Neural computation*, vol. 15, no. 6, pp. 1191–1253, 2003.

[34] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, JMLR Workshop and Conference Proceedings, 2010.

[35] H. Zou and T. Hastie, “Regularization and variable selection via the elastic net,” *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 67, no. 2, pp. 301–320, 2005.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[37] F. Berman, G. Fox, and T. Hey, “Overview of the book: grid computing–making the global infrastructure a reality,” *Grid computing: Making the global infrastructure a reality*, pp. 1–8, 2003.

[38] R. Hecht-Nielsen, “Theory of the backpropagation neural network,” in *Neural networks for perception*, pp. 65–93, Elsevier, 1992.

[39] 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*, vol. 1, no. 4, pp. 541–551, 1989.

[40] L. D. Nguyen, D. Lin, Z. Lin, and J. Cao, “Deep cnns for microscopic image classification by exploiting transfer learning and feature concatenation,” in *2018 IEEE international symposium on circuits and systems (ISCAS)*, pp. 1–5, IEEE, 2018.

[41] L. Perez and J. Wang, “The effectiveness of data augmentation in image classification using deep learning,” *arXiv preprint arXiv:1712.04621*, 2017.

[42] Y. F. D. de Sant’Anna, J. E. M. de Oliveira, and D. O. Dantas, “Interpretable lightweight ensemble classification of normal versus leukemic cells,” *Computers*, vol. 11, no. 8, p. 125, 2022.

[43] R. Gupta, P. Mallick, R. Duggal, A. Gupta, and O. Sharma, “Stain color normalization and segmentation of plasma cells in microscopic images as a prelude to development of computer assisted automated disease diagnostic tool in multiple myeloma,” *Clinical Lymphoma, Myeloma and Leukemia*, vol. 17, no. 1, p. e99, 2017.

[44] R. Duggal, A. Gupta, R. Gupta, M. Wadhwa, and C. Ahuja, “Overlapping cell nuclei segmentation in microscopic images using deep belief networks,” in *Proceedings of the tenth Indian conference on computer vision, graphics and image processing*, pp. 1–8, 2016.

[45] R. Duggal, A. Gupta, R. Gupta, and P. Mallick, “Sd-layer: stain deconvolutional layer for cnns in medical microscopic imaging,” in *Medical Image Computing and Computer Assisted Intervention- MICCAI 2017: 20th International Conference, Quebec City, QC, Canada, September 11-13, 2017, Proceedings, Part III 20*, pp. 435–443, Springer, 2017.

APPENDIX

Evaluation Metrics: Results were obtained using three distinct k values (K=1-3) for each fold of the C-NMC 2019 dataset. The cross validation approach, K-fold, was utilized to test the performance of the four CNN models for multiclass classification. The average accuracy and average weighted F1-score across all folds were determined once all folds had been complete. The average accuracy and average weighted F1-score across all folds were calculated after finishing all folds. An independent test dataset was then used to evaluate the entire model. The initial learning rate gradually declines as we increase the number of epochs to 35. The softmax function was employed for the last categorization.

Accuracy (Correctly classified), precision, recall and F1-Score have been considered for evaluating the performance of our model. A model's performance is calculated by how well it represents well-observed real-world events. A labeled data set containing the actual values to be predicted is used to train any model. Accuracy is simply the ratio of correctly predicted observations and is given by:

$$Accuracy = \frac{TP + TN}{Tp + TN + FP + FN} \tag{6.1}$$

Precision and Recall is given by:

$$Precision = \frac{TP}{TP + FP} \tag{6.2}$$

and

$$Recall = \frac{TP}{TP + FN} \tag{6.3}$$

Precision and Recall are combined to form the F1-score:

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{6.4}$$

Where TP, TN, FP, FN represent true positive, true negative, false positive and false negative respectively.

We used different libraries and single framework. For framework we have used tensorflow, and for library we have used pandas, numpy, scikit-learn, matplotib and many more.

Tensorflow: TensorFlow is an open-source software library for AI (artificial intelligence) and ML. It was created by the Google Brain team and is used to install deep neural networks and other machine learning models. The programming languages supported by TensorFlow, which provides a flexible and effective framework for developing, training, and deploying machine learning models, include Python, C++, and JavaScript. It is frequently used for a number of activities by academics, professionals, and businesses, including audio and image identification, natural tongue processing, and predictive analytics.

TensorFlow is a strong and extensively used open-source ML (machine learning) and DL (deep learning) framework. TensorFlow, created by Google, is a comprehensive suite of tools and frameworks that allow developers and researchers to easily design and deploy machine learning models.

TensorFlow's adaptability is one of its primary assets. TensorFlow Keras is a high-level API that enables for quick and intuitive model creation and training. Users may easily construct and configure neural networks using TensorFlow Keras by utilising a simple and expressive syntax. It is useful for a wide range of machine learning problems since it supports a large range of layers, activation functions, and optimizers.

TensorFlow also has a lower-level API for finer-grained control and customisation. Users may use this to create complicated neural network topologies and powerful machine learning techniques.

TensorFlow is also renowned for its ability to efficiently exploit hardware accelerators like as GPUs and TPUs (Tensor Processing Units). TensorFlow can greatly speed up the training and inference processes by using these accelerators, enabling for quicker and more scalable machine learning applications.

Pandas: Pandas is a set of tools for data analysis and manipulation designed specifically for Python. It provides data structures for efficiently storing enormous datasets as well as tools for interacting with massive datasets. The two primary 2-dimensional data structure types in

Pandas are the Series and DataFrame.

Pandas offers a number of practical and effective data analysis and data transformation procedures, making it particularly well suited for working with tabular data, such as data contained in spreadsheets or SQL databases. It may be used in conjunction with other Python scientific computing and machine learning libraries, and is extensively used in the data science community for tasks like data cleansing, data aggregation, and feature engineering.

Pandas is a prominent open-source Python toolkit that provides strong data analysis and manipulation features. It is frequently utilised in disciplines such as data science, machine learning, and finance, to name a few. Pandas streamlines the process of dealing with structured data with its straightforward and adaptable data structures, making it a go-to tool for many data professionals.

Pandas' two core data structures, the Series and DataFrame, are one of its distinguishing features. A DataFrame is a two-dimensional table that may contain a range of data types, in contrast to a Series, which is an object that resembles a one-dimensional array and can carry any data type. These structures facilitate data processing, indexing, and manipulation.

Numpy: The Python programming language's NumPy module offers support for arrays, a data format for effectively storing and handling massive collections of homogenous data (e.g. numbers, strings, etc.). It is the foundational package for Python's scientific computing and offers support for arrays.

Numpy is a core module in the Python environment that provides sophisticated numerical computation tools. It stands for ”Numerical Python” and is the cornerstone for many other libraries and frameworks in data science, ML (machine learning), and scientific computation.

The ndarray, a multi-dimensional array object, lies at the heart of NumPy. The ndarray is a data structure that provides for the efficient storing and manipulation of homogenous data, such as numerical values. It serves as a versatile and efficient container for massive datasets, as well as allowing for quick mathematical operations on arrays. The ndarray's ability to conduct vectorized operations considerably improves numerical computing performance.

Basic arithmetic, statistical computations, linear algebra functions, and Fourier transformations are just a few of the many mathematical operations and functions that NumPy provides that work with arrays. These operations can effectively manage massive amounts of data since they are performance-optimized. Complex numerical computations may be carried out using clear and understandable code by utilising NumPy's capabilities.

NumPy's easy interaction with other libraries in the scientific Python environment is one of its main features. It provides effective data structures and computational capabilities to libraries like pandas, SciPy, and scikit-learn as their base. The simplicity with which NumPy arrays may be transformed into and out of various data structures makes it easier for different libraries to work together.

Matplotlib: Matplotlib is the name of a charting library for the Python programming language. It provides an object-oriented API for adding charts to programs using general-purpose GUI toolkits as Tkinter, wxPython, Qt, or GTK. In addition to other visualizations, Matplotlib can create excellent 2D and 3D plots, charts, histograms, and power spectra. For data analysis and visualization, it may be used with other libraries from the scientific Python ecosystem, including NumPy and Pandas.

In a variety of fields, such as data science, machine learning, and scientific computing, Matplotlib is frequently used for data visualization. It supports a broad variety of output formats, including PNG, PDF, SVG, and JPG, and offers a high-level interface for building static, animated, and interactive visualizations. Python's Matplotlib package is a well-liked and often used data visualisation tool. It offers a versatile and extensive collection of tools for making excellent plots, charts, and visualisations. Matplotlib is a go-to package for data visualisation, whether you're examining data, sharing insights, or producing visuals that are suitable for publishing.

The adaptability of Matplotlib is one of its main advantages. Users may build a variety of visualisations using its extensive customisation and plotting options, including line plots, scatter plots, bar plots, histograms, heatmaps, and more. You have complete control over the visual aspects of your plots using Matplotlib, including the colours, labels, markers, line styles, and legends.

Visualisations may be made using a two-level interface provided by Matplotlib. The pyplot module offers a straightforward and user-friendly API that is similar to MATLAB, making it accessible to both newcomers and MATLAB users. With little coding, it enables rapid and interactive graphing. The object-oriented interface, on the other hand, offers more flexible and fine-grained control over plot components. It is appropriate for experienced users that demand greater customisation and intricate narrative designs.

The integration of Matplotlib with NumPy, another well-liked Python library for numerical computing, is one of its standout features. The direct plotting of NumPy arrays by Matplotlib makes it simple to see data and perform analyses. Additionally, Matplotlib and pandas have a seamless integration that makes visualising DataFrame and Series objects simple.

Scikit-learn: A machine learning library for the Python programming language is called Scikit-learn. It offers a variety of algorithms for jobs including model selection, regression, clustering, classification, and dimensionality reduction and is designed to be simple to use and effective in real-world scenarios. Scikit-learn interacts nicely with these and other libraries in the scientific Python environment since it is built on top of other well-known Python libraries, such as NumPy and Matplotlib. Scikit-learn is widely used in industry and academia for building machine learning models, and provides a simple and consistent interface to a variety of algorithms, making it a popular choice for both beginners and experienced practitioners. It is also known for its good documentation, and for being a well-maintained and actively developed open-source project, with a strong community of contributors and users.

Scikit-learn, also referred to as sklearn, is a popular Python machine learning library that is open-source. It provides a rich and versatile set of tools for data preprocessing, model selection, model training, and model evaluation. With its user-friendly interface and extensive functionality, scikit-learn empowers both beginners and experienced practitioners to apply machine learning algorithms effectively.

One of the strengths of scikit-learn is its ease of use. The library is designed with simplicity in mind, making it accessible to users with varying levels of machine learning expertise. Scikit-learn provides a consistent API for various machine learning tasks, allowing users to apply algorithms with minimal effort. The library supports a wide range of supervised and unsupervised learning algorithms, including linear models, decision trees, support vector machines, random forests, clustering algorithms, and more.

Scikit-learn facilitates the entire machine learning workflow by offering modules for data preprocessing and feature engineering. It provides functions for handling missing values, scaling and normalizing features, encoding categorical variables, and transforming data. These preprocessing capabilities are crucial for preparing data for modeling and improving the performance of machine learning algorithms.

The library also offers utilities for model selection and hyperparameter tuning. Scikit-learn provides tools for cross-validation, which allows users to assess the performance of models and select the best one for their specific task. Additionally, scikit-learn supports grid search and randomized search for hyperparameter optimization, helping users find the optimal combination of model parameters for improved performance.

Scikit-learn integrates seamlessly with other Python libraries, such as NumPy, pandas, and matplotlib, further enhancing its functionality. The interoperability with these libraries enables smooth data manipulation, visualization, and analysis throughout the machine learning pipeline. This integration makes scikit-learn a versatile and powerful tool in the broader data science ecosystem.