

# Classification of Peripheral Blood Cell Images using Deep Learning

by

Oyshik Ahmed Aadi  
18201052

Md.Meghdad Hossain Akash  
19201138

Fahim Ishraq  
18301077

Asif Hossain  
22241039

Abdullah Al Fahim  
18201099

A thesis submitted to the Department of Computer Science and Engineering  
in partial fulfillment of the requirements for the degree of  
B.Sc. in Computer Science

Department of Computer Science and Engineering  
Brac University  
January 2023

© 2023. Brac University  
All rights reserved.

# Declaration

It is hereby declared that

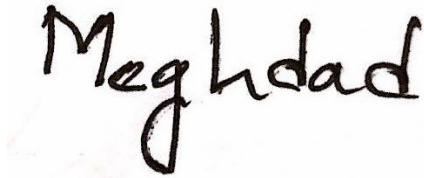
1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

**Student's Full Name & Signature:**



---

Oyshik Ahmed Aadi  
ID: 18201052



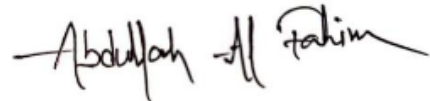
---

Md. Meghdad Hossain Akash  
ID: 19201138



---

Fahim Ishraq  
ID: 18301077



---

Abdullah Al Fahim  
ID: 18201099



---

Asif Hossain  
ID: 22241039

# Approval

The thesis/project titled “Classification of Peripheral Blood Cell Images using Deep Learning” submitted by

1. Oyshik Ahmed Aadi (18201052)
2. Md.Meghdad Hossain Akash (19201138)
3. Fahim Ishraq (18301077)
4. Asif Hossain (22241039)
5. Abdullah Al Fahim (18201099)

Of Fall, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 19, 2023.

## Examining Committee:

Supervisor:  
(Member)



---

Dewan Ziaul Karim  
Lecturer  
Department of Computer Science and Engineering  
BRAC University

Program Coordinator:  
(Member)

---

Dr. MD. Golam Rabiul Alam  
Professor  
Department of Computer Science and Engineering  
BRAC University

Head of Department:  
(Chair)

---

Dr. Sadia Hamid Kazi  
Associate Lecturer  
Department of Computer Science and Engineering  
BRAC University

## **Ethics Statement**

Writing the thesis is a crucial step strictly in compliance with the university's rules and regulations, as well as ethical principles for doing research. Original data has been incorporated into our thesis. We double-checked our citations and references. Each of the paper's five co-authors accepts responsibility for any violations of the thesis rule. There were several questionnaire-free tools, articles, and YouTube videos that helped us address problems. In addition, we would want to take this opportunity to thank everyone who has assisted us along the journey. Our thesis was completed without the use of unethical tactics. The Brac University's code of ethics guides our activities.

## Abstract

Diagnosis and Identification of cells and disease infected cells are an important part of medical science that bears huge significance even today. There are health implications can often be identified by observing the morphological changes of cells as well as the quantity of cells. The traditional methods of counting blood and chemically identifying diseases can be expensive and/or time consuming to the extent that only certain medical centres can perform the task at hand, or take days to receive a report of. However, we believe Deep Learning with Convolutional Neural Networks (CNNs) can take over most of this tedious process. In this work, we aim towards creating a custom CNN model that can quickly classify different kinds of peripheral blood cells such as the 7 different white blood cell types (basophils, erythroblasts, ig, eosinophils, lymphocytes, monocytes, neutrophils) and platelets. Such a model can be used in blood cell counts which can be used to identify cases like leukemia. Moreover, such a method can be extended into other fields such as red blood cell detection or even infected cell detection, which includes identifying diseases from Sickle Cell Anemia to cells affected by Covid19. Our custom CNN model has performed exceptionally well, achieving accuracies as high as 99.1% and 98.9% in training and validation respectively, which is significantly higher than using pre-trained models such as DenseNet or NasNet. In more ways than one, we show how our model is better suited for the task at hand.

**Keywords:** Convolutional Neural Network, Classification of Blood Cells, Deep learning.

## Dedication

To our teammates and family members, whose unwavering encouragement and the team's engrossed pursuit of the goal both played crucial roles in the successful completion of this paper. It is to them that we dedicate this paper.

## **Acknowledgement**

Firstly, all praise to the Great Allah for whom our thesis have been completed without any major interruption.

Secondly, to our supervisor Dewan Ziaul Karim sir for his kind support and advice in our work. He helped us whenever we needed help.

And finally to our parents without their throughout support it may not be possible. With their kind support and prayer we are now on the verge of our graduation.



# Table of Contents

Declaration	i
Approval	ii
Ethics Statement	iv
Abstract	v
Dedication	vi
Acknowledgment	vii
Table of Contents	viii
List of Figures	x
List of Tables	xi
Nomenclature	xii
<b>1 Introduction</b>	<b>1</b>
1.1 Background Information . . . . .	1
1.2 Problem Statement . . . . .	1
1.3 Research Objectives . . . . .	2
<b>2 Literature Review</b>	<b>3</b>
2.1 Related Works . . . . .	3
<b>3 Work Plan</b>	<b>8</b>
3.1 Workflow and Methodology . . . . .	8
3.2 Dataset . . . . .	10
3.3 Data Source . . . . .	10
3.4 Data Classification . . . . .	11
3.5 Data Pre-processing . . . . .	11
3.5.1 Image Resizing . . . . .	12
3.5.2 Image Augmentation . . . . .	12
3.5.3 Image Scaling and Normalization . . . . .	12
3.5.4 Image Thresholding . . . . .	12
3.6 CNN and Non-linearity . . . . .	13

<b>4</b>	<b>Description of the Model</b>	<b>14</b>
4.1	Model . . . . .	14
4.2	CNN Model . . . . .	14
4.2.1	Convolutional Layer . . . . .	15
4.2.2	Pooling Layer . . . . .	15
4.2.3	Fully Connected Layer . . . . .	16
4.2.4	CNN Model Summary . . . . .	16
4.3	VGG-19 . . . . .	18
4.4	MobileNetV3 . . . . .	19
4.5	ResNet50 . . . . .	20
4.6	EfficientNetV2B1 . . . . .	21
4.7	DenseNet121 . . . . .	22
4.8	NASNet . . . . .	23
4.9	Inception Network . . . . .	24
4.10	A Custom Model . . . . .	25
<b>5</b>	<b>Preliminary Analysis</b>	<b>28</b>
5.1	Result and Analysis . . . . .	28
5.2	Performance Analysis . . . . .	29
5.3	Result Comparison . . . . .	33
<b>6</b>	<b>Challenges</b>	<b>34</b>
<b>7</b>	<b>Conclusion and Future work</b>	<b>35</b>
7.1	Conclusion . . . . .	35
7.2	Future work . . . . .	35

# List of Figures

3.1	Workflow . . . . .	9
3.2	Cell images . . . . .	11
4.1	CNN Architecture . . . . .	15
4.2	Regular Neural Network vs CNN . . . . .	15
4.3	Layers . . . . .	17
4.4	Network Architecture of VGG19 Model . . . . .	18
4.5	MobieNetV3 . . . . .	19
4.6	ResNet50 . . . . .	20
4.7	EfficientNet Architecture . . . . .	21
4.8	DenseNet121 . . . . .	22
4.9	NASNet Architecture . . . . .	23
4.10	Inception Network . . . . .	24
4.11	Custom Model Architecture . . . . .	27
5.1	EfficientNetV2B1 accuracy graph . . . . .	29
5.2	NasNet accuracy graph . . . . .	29
5.3	InceptionNet accuracy graph . . . . .	30
5.4	DenseNet121 accuracy graph . . . . .	30
5.5	Custom Model Accuracy Graph . . . . .	31
5.6	Custom Model Loss Graph . . . . .	31
5.7	Confusion Matrix Based on Custom CNN Model . . . . .	32
5.8	Graphical Representation of different models . . . . .	33

# List of Tables

3.1	Table for Individual Class . . . . .	11
5.1	Classification Report Based on Custom CNN Model . . . . .	32
5.2	Accuracy and Validation percentages of the models used. . . . .	33

# Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

*CDSS* Clinical Decision Support System

*CNN* Convolutional Neural Networks

*CPU* Control Processing Unit

*CUDA* Compute Unified Device Architecture

*DSSCSCNNs* Disruption-based Salp Swarm and Cat Swarm-based optimized Convolutional Neural Networks

*DT* Decision Tree

*GPU* Graphics Processing Unit

*LSTM* Long Short Term Memory

*MLP* Multilayer Perceptron

*PBC* Peripheral Blood Cell

*PCA* Principal Component Analysis

*RBC* Red Blood Cells

*RF* Random Forest

*RNN* Recurrent Neural Networks

*SNN* Segmentation Neural Network

*SVM* Support Vector Machine

*TL* Transfer Learning

*VGG* Visual Geometry Group

*WBC* White Blood Cells

# Chapter 1

## Introduction

### 1.1 Background Information

AI and Deep Learning have made great advancements in many areas of science and research. One such area is the healthcare industry, in which it allows for the continuous growth and development of medicines as well as the understanding of the human body. Images may be utilized to train a Machine Learning Algorithm using Deep Learning for object categorization. This is achieved through the use of convolutional neural networks. One such object is Blood Cells.

Blood cells are mainly of 3 kinds. Red Blood Cells (RBC), which are red, round and concave disc shaped cells with no nucleus. It carries oxygen around the body. White Blood Cells (WBC) often vary in shape and structure, and each has a very distinct nucleus. Their function is to guard the body against pathogens and remove harmful substances from the body. Involved in the same area, Platelets are also a part of the Peripheral Blood Cells. They help in creating blood clots which are used to seal wounds to prevent bleeding.

The tasks that blood cells carry out are extremely vital. So, any infections or diseases involving them can be extremely deadly and it is important that any such maladies are diagnosed and treated as fast as possible. However, this is not an easy task. In most cases, diagnosis often is performed when it's too late, and in other cases, diseases are not detected until much later. For example, people only take blood cell count after it has dropped low enough for symptoms to appear, and even then, the process of manually counting blood cells is long and tedious. Blood cell counts and the detection of diseases like Sickle Cell Anemia and Malaria, which are known to cause physical changes to the appearance of the cell, can be sped up significantly with the help of Deep Learning's ability to classify blood cells based on their shape and size.

### 1.2 Problem Statement

In regard to peripheral blood cells, there are a lot of tests that need to be done in order to identify the existence of certain cells or chemicals that can dictate a person's health condition. However, some of these tests may require additional resources (such as reagents) or require a lot of time to finish. Moreover, testing requirements vary depending on the scope of the tests. Some tests can require 3 minutes, while others, like blood count tests, can take 24 hours to 3 days. Moreover,

each test may even take 1 to 2 weeks depending on the waiting times or backlog of patients. The most recent example was the Covid tests where, while the test itself is not intensive, the amount of people taking the test resulted in delays in receiving the results in terms of weeks.

Some tests are difficult to do in general, as they often require a lot of manual labor. For example, Thrombocytopenia is the disease which results in very low platelet counts. This hampers the body's ability to seal wounds through blood clots and therefore cannot prevent bleeding. Platelets are often referred to as "cell fragments" and are extremely small in size. Thus, counting them manually is extremely difficult. Moreover, we generally move to other methods such as searching for (or the lack of) clotting factors to find signs of Thrombocytopenia which requires additional resources.

In regard to other peripheral blood cell counts, Leukocytosis is the term used to describe high white blood cell counts. However, this in and of itself is usually not a problem, as it can mean that the body is simply responding to an infection or an inflammatory response. However, it can be an indicator of the onset of leukemia, which is a type of blood cell cancer where white blood cells multiply uncontrollably. To properly count them, we would need to use Deep Learning Convolutional Neural Networks to classify different peripheral blood cells simultaneously. CNNs algorithms can be trained with microscope images and we would need to classify platelets, white blood cells, and red blood cells simultaneously as blood cell counts are more meaningful when they are taken as a ratio of the available cell samples.

### 1.3 Research Objectives

The objective of this thesis is to design a classification system to simultaneously classify peripheral blood cells from blood samples presented as microscope images such that diseases including but not limited to: Thrombocytopenia, Leukocytosis, Sickle Cell Anemia, and others that can be physically observed can be detected. Deep Learning with Convolutional Neural Networks will be used to classify and detect peripheral blood cells to determine the ratio of platelets, WBC, and RBC as well as determine abnormalities in the physical appearance of these cells after necessary image processing. Our objectives are -

- Understanding preprocessing techniques such as image processing, denoising and reshaping,
- Creating a Deep Learning model to detect and classify peripheral blood cells through CNN.
- Able to check for abnormal or normal blood cells after classification.
- Designing the Deep Learning model to aid doctors during diagnosis of patients that is also accurate

# Chapter 2

## Literature Review

### 2.1 Related Works

In this paper by Riya Roy and Ms. Swapna Sasi et al. [1] the authors had proposed several frameworks to aid with the classification the different types of white blood cell by utilizing multi-class support vector machine classification (abbreviated as SVM classifications) and convolutional neural networks (CNNs). Some of these white blood cells include, but not limited to: Neutrophils, Lymphocytes, Monocytes, Eosinophil and Basophils. These cells can be identified through their physical/morphological appearance and the utilization of deep learning. Such methods could be used for diagnosis of diseases related to morphological changes of blood cells.

In another paper by R F Rahmat, F S Wulandari, et al. [2] they also used support vector machines but also provided general details of their architecture. It involves using the shape and structure of cells, in this case: Red Blood Cells, through feature extraction after preprocessing the image, which involves Greyscaling, Thresholding, Erosion, and Dilation. The features in question were that what they were looking for was “roundness.” While their level of accuracy varies by a relatively large margin, they do average about a 93% level of accuracy.

This authors [3] suggested a CNN-based technique for WBC classification and performed a comparatative analysis of W-Net Algorithm against other alogrithms such as AlexNet, VGG16/19, SVM, variations of ResNet, and RNNs. Moreover, W-Net is tested on a large-scale real-world dataset with 6,562 real photos of the five WBC types. They demonstrated that W-Net outperforms all other algorithms in terms of accuracy. Furthermore, W-Net is made up of three convolutional layers and two fully connected layers that collect and, through the utilization of a softmax classifier, learn features from WBC pictures in order to accurately classify them into five classes. The W-Net architecture has an overall average accuracy of 97%. Meanwhile, other architectures that were compared with W-net results at average accuracy of AlexNet(84%), VGGNet(44%), RNN(83%), and ResNet(51%). Also, they used the hinge loss function to train a W-Net model with an SVM classifier (W-Net-SVM). The training time of W-Net-SVM was 33.79. Afterwards, they used two types of data: public datasets and trained datasets. They showed that training a model on a large-scale dataset gives better performance. Thus, they proved that the W-net



algorithm performs better than other CNN-based models.

Girdhar et al. [4] discussed large amounts of labeled data are necessary for CNN. Various datasets, including blood samples, were researched, including those that were known to be available locally (20%), publically (48%) and unknown (20%). After that, they suggested four convolution layers, normalization, leaky Relu, and a pooling layer. To avoid overfitting, dropout layers are used. After that, a softmax and fully-connected layer are implemented. WBC kinds are categorized by the categorization layer, which draws on the features learned in earlier layers to do so. Binary categories were given a confusion matrix. This dataset contains 12,444 blood-cell pictures. Each category has an equal number of photos: eosinophils, lymphocytes, monocytes, and neutrophils (approximately 3000). This dataset's 98.55% accuracy beats all previous records. The suggested CNN model was trained and assessed on single-celled pictures, which are unlikely in a blood sample. Future implementations include multi-celled, overlapping, and obscured WBC imaging.

This paper [5] presented blood cell automated recognition using convolutional neural networks (CNN). Two designs used Vgg-16 and Inceptionv3 convolutional neural networks. The networks initially extracted features to train an SVM classifier (SVM). Fine tuning improved the second time. Vgg-16 and Inceptionv3 architectures had 86% and 90% accuracy, respectively. In fine-tuning, Vgg-16 and Inceptionv3 architectures were 96% and 95% accurate. The Oxford University-trained Vgg-16 CNN recognizes objects. Google created Inceptionv3 to reduce CNN's computing cost. Unbalanced and balanced datasets are used to train the data. They proved a balanced, high-quality image improves results. CNN has layers. CNN's convolutional layer converts input to output using a convolutional algorithm. ReLu increases the network's learning speed and classification accuracy when applied to a convolutional layer's output. Pooling reduces map size. Fully connected and output layers produce functions for probability distributions. A confusion matrix with actual rows and expected columns was also introduced. Fine tuning yielded 96.2% accuracy overall.

The authors [6] proposed a hybrid and effective CNN strategy for PBC image identification and classification. SSPSO-CNN, OLPSO-CNN, BQ-CNN, and DNN-JOA were introduced to improve peripheral blood cell image identification and classification accuracy. These methods are compared to others based on each peripheral blood cell image's true positive rate. The overall accuracy was 98 percent after using confusion matrix. Fine-tuning on 10,674 medical images yield the greatest results. The approach enhanced precision, F1-score, and overall classification accuracy to 99 percent.

The authors [7] discussed about white blood cells: their classification, types, and functions in our blood, laboratory testing, and the article's topic. First, the authors discussed the D.L model in conjunction with DenseNet121 to classify white blood cells. The proposed model achieved very high levels of accuracy (98.84%), precision (99.53%), sensitivity (98.5%), and specificity (99.51%). The main problem with the proposed study is that both training and testing are done on the same dataset, which only has WBC samples. Additionally, the authors talked about their trained

dataset, proposed model, and findings from the study. Also, they specified how the proposed model was built and the modification of the training dataset. Later on, the way of preprocessing the dataset, basically its image source, image size, training part, data normalization, and data augmentation, was explained by the authors. Furthermore, in the feature extracting part, the authors basically clarified the features of the model, resizing the image, Adam optimizer, and convolutional block by using DenseNet121. Finally, the authors summed up the outcomes of the model.

This paper [8] demonstrated a microscopy-based dataset of peripheral blood cells for use in training automated recognition models. Here, they applied deep convolutional neural network (CNN) architectures that had been trained on the ImageNet dataset to the problem of classifying peripheral blood cells. Wide ResNet-50-2 has a higher rate of accuracy, at 99.32%, than VGG-19, which only manages 99.27%. By setting empirical benchmarks for blood cell classification with deep learning, their study provides a starting point for future work with specialized architectures and techniques.

Gavas et al. [9] presented the deep CNNs for peripheral blood cell classification. First of all, the cell's functionality in blood, elements of blood, types of cells, their functionalities, and the process of classifying the blood cells through CNNs model were explained in detail by the authors. Secondly, the authors mentioned various types of peripheral blood cell segmentation and classification for diagnosing diseases related to blood cells. The methodologies for classifying blood cells, such as CNN Architects and Transfer Learning (TL), using the ImageNet dataset, were mentioned by the writers. Furthermore, the authors described the experimental process and possible outcomes of the model. Moreover, later on, the authors discussed how the model can help to detect several blood-related diseases by classifying the blood cells based on the findings. Finally, the authors clarified the accuracy of their findings.

This paper [10] showed an efficient multi-level convolutional neural network approach for white blood cell classification. First of all, blood cells, their types, different methodologies of classifying blood cells, and naming processes were mentioned by the authors. Additionally, following the previous segment in the state of the art portion, the author mentioned the different models like CNN, DCNN, MFCNN, capsule networks, LSTM, and so on to classify the blood cells. Moreover, the authors acknowledged the materials used in this proposed model and the formulas to get the findings, accuracy, recall, precision, and  $F_{score}$ . Furthermore, the results and accuracy of the findings based on the previous segment were discussed by the authors. Finally, the authors concluded by mentioning the purpose of the model and how it could contribute further help.

Maria et al. [11] proposed a paper that uses a deep learning approach for segmentation of red blood cell images and malaria detection. Malaria is an endemic, deadly disease. There are more than 120 types of malarial species that affect animals, and more than six species affect humans. Deep learning using Convolution Neural network (CNN) and segmentation neural network (SNN) has immense potential in the field of digital pathology, by using this different anomalies on RBC can be identified. A CDSS (Clinical Decision Support System) is designed using a neural network to

detect malaria infection. Digital images of peripheral blood smears will be used as input, and the presence of infection in the RBCs will be labeled as an output. The data sets used were 80% to train and 20% for validation. The global accuracy of the test set for the data was 90.29% to 93.72%. The highest accuracy needed the most time. This model doesn't require any preprocessing or manual data extraction. It was proposed in a three-stage pipeline segmentation of a RBC cell.

Laura et al. [12] suggested a paper combining peripheral blood cell pictures and a deep learning model (ALNet) to diagnose acute leukemia lineage. Acute leukemia is heterogeneous. It replaces cells and decreases 3 haematopoietin lines in peripheral blood cells. In these model, AI decision making helps to a great extent in detecting haematology malignancy. 16,450 single RBC cells were analyzed 85% images were used to train and rest 15% was used to test. These digital blood cells were collected by CellaVision DM96 (363 X 360 pixels). To identify the acute leukemia lineage, VGG16, RESNET 101, and SENNET154, these architectures were used. ALNET gave correct diagnostic predictions with a value of 100%, the precision value of 93.7%, and the specificity accuracy of the leukemia lineage is 92.3%. Alnet is a predictive model made up of a few convolutions neural networks (CNN).

The authors [13] proposed a paper where they were able to automatically classify cells in a peripheral blood image, which is based on morphological image processing. Pathologists employ RBC and WBC information to diagnose and identify illnesses, bacteria, and viruses in blood cells. Computer aided inspection of RBC is done by using a segmentation algorithm to extract the cell classification. Methods like preprocessing, segmentation, principal component analysis (PCA), and feature extraction are used to construct more efficient classifiers to identify components in a blood cell. The images are collected using a CCD color camera attached to a microscope that is zoomed to 400 times of 640 x 480 pixels. In this system, UNL (Universidade Nova de Lisboa) Fourier transformation is used to manage open curves and lines in the image. Using this model provides faster recognition in detecting components insides RBC and WBC.

This paper [14] classifies WBC using machine learning. White blood cell classification uses TWO-DCNN. Two-model transfer learning and deformable convolution are used. VGG16, VGG19, RESNET-50, SVM, MLP, DT, and random forest architectures were employed (RF). Two-DCNN performs best. With exact feature extraction and improved network weights, precision is 91.6% to 95.7%. For low-resolution, noisy data sets, the TWO-DCNN performs best.

The authors [15] used Hybrid DSSCS and convolutional neural network to recognize peripheral blood cells. DSCSCNNs are used in this model. This solves CNN's hyperparameter issue. The training set uses the VGG-16 model architecture. This approach provides 100% accuracy, sensitivity, and specificity. Because of this, it has the highest peripheral blood cell detection accuracy and 99% blood cell categorization accuracy.

The authors [16] said that WBC can detect infections, allergic reactions, inflammation, and blood cancers including leukemia and lymphoma (WBCs). Semantic

segmentation, database update, and classification via transfer learning comprise the bulk of the suggested approach. Deep learning-based categorization of five WBC subtypes from whole blood smear pictures is a unique feature of our study, which we want to use in the near future. According to the authors, transfer learning strategies yield improved accuracy with less datasets. The authors also covered WBC localization, semantic segmentation, and database update, as well as classification and transfer learning using AlexNet. In addition, the author went into further depth on the data base, the photos' origins, and other related topics. The technique uses deep learning-based semantic segmentation for WBC localisation and transfer learning-based categorization. Segmentation accuracy is 98.22%, mean IoU is 84.22% (@IoU = 0.7), and classification accuracy is 98.87%. Finally, the author presented the findings of his investigation.

This paper [17] provided a background on blood and RBCs. Second, they went through the CNN aspect of the proposal and how it would function. It is possible to resolve and analyze the fine features in an input picture using convolutional neural networks (CNNs). RBCs in flow may be distinguished by their form properties, and outliers can be identified using a CNN. In addition, the author discussed ethical statements, the architecture and training of CNN, and datasets in the materials and methods section (total classified cells, images, etc). When presenting their findings, the authors included a table with the output values of all the cells, an image montage of all the incorrectly categorized croissants, a confusion matrix with true values and relative percentages, and a commentary on the results. When it comes down to it, pre-trained and freely accessible artificial neural networks don't fit the job at hand, the author found. To train the CNN, only RBC forms are used in the flow, so any cell class limitations are completely artificial. Its function will be better understood via more experiments involving varied channel geometries and flow conditions.

The authors [18] have discussed the white blood cell or leukocytes. They also suggested a CNN-based model with 94 percent accuracy for polynuclear and mononuclear and 78 percent accuracy for eosinophil, lymphocyte, neutrophil, and monocyte. Second, they described baseline models to compare with our suggested model. Naive Bayes and SVM methods follow. They have further included the concept of convolutional neural networks. In this part, approaches are presented, followed by architecture and algorithms. In the following part, they displayed the dataset. This dataset comprises around 13k enhanced images of 170 blood cells labeled with their kind. There are around 3000 images for each blood cell type, divided into 5 groups based on the kind of cell, i.e. eosinophil, basophil, lymphocyte, neutrophil, and monocyte. In addition, they addressed the materials utilized for setup and outcome analysis. Finally, the writers concluded their discussion on the potential of this project and future efforts. This model is useful for 220 blood diagnoses in the medical industry, which may save a significant amount of time.

# Chapter 3

## Work Plan

### 3.1 Workflow and Methodology

In this specific portion, the methodologies used in this research is illustrated. Initially data collection is done then the dataset is sorted out by putting in different pre-processing methods to it. The total advancement of this research incorporates a conventional CNN model which constitutes of an architecture that has 7 convolutional layers in the custom model and also has a transfer learning technique for 7 other pre-trained convolutional neural network models (VGG-19, MobileNetV3, EfficientNetV2B1, Dense Net121, Res Net50, NasNet, Inception Network) and then, after comparing their achievement characterize by validity, correctness, F1score, Confusion Matrix, and AUC curve the optimal structure for the blood cell identification and classification on blood cell pictures and blood cell components will be determined. The process is described progressively in the phases listed below:

- Collection of Data
- Pre-Processing of Data
- Conventional CNN model
- Pre-Trained Convolutional Neural Network models for Transfer Learning
- Asses the CNN model's accomplishment

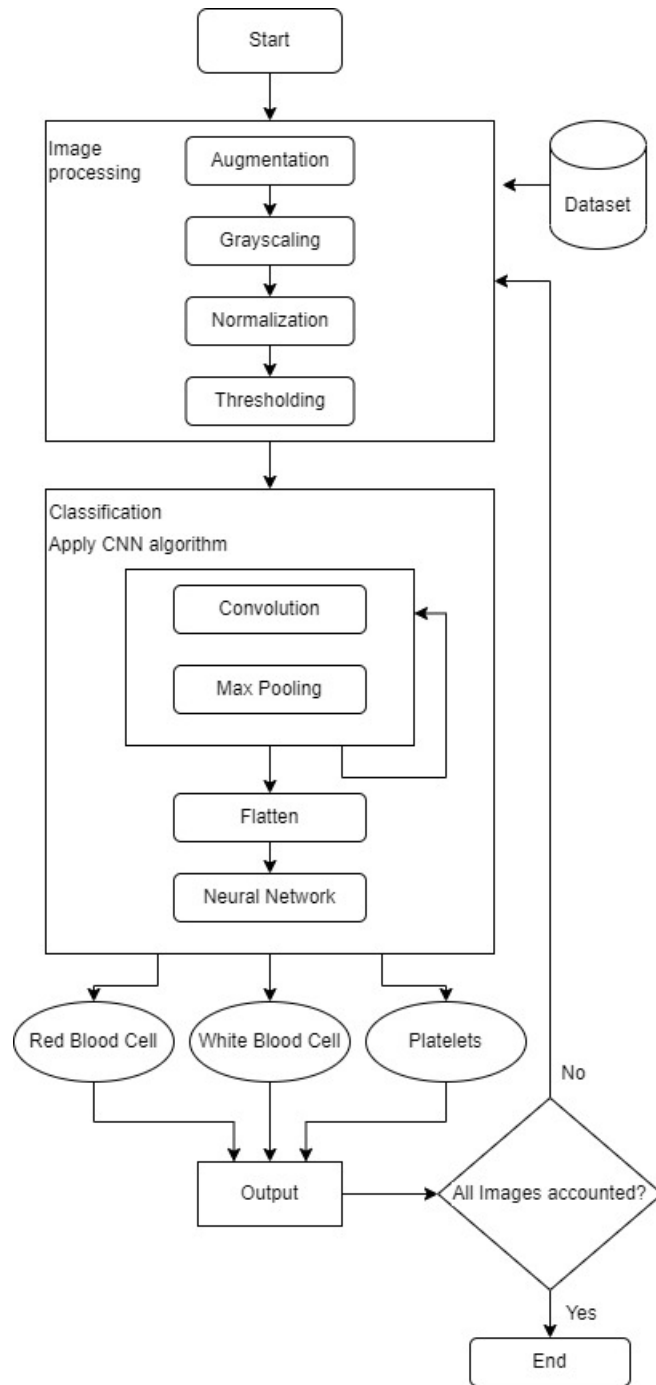


Figure 3.1: Workflow

## 3.2 Dataset

Here, we will go over the strategies that we will employ to accomplish our goal. We'll start by utilizing the tagged photo-containing kaggle PBC dataset. Deep learning frameworks favor using a lot of high-quality photos for model testing and training. Data reasoning can be used to create a solid PBC image. This will increase the accuracy and rate of the classification and picture differentiation process. By using an approach called data augmentation, we can considerably increase the amount of variety of images that we can use in training models without the need for actually acquire new data. By adjusting orientation, brightness, scale, position, and other factors using the available dataset, it is possible to create images artificially. Without significantly altering the model itself, it is simple to improve accuracy. In our experiment, we want to employ approximately 4,096 photos per class. Our training dataset has undergone certain changes. Setting the image sizes, batch size, rescaling, rotation, horizontal and vertical flipping are all included in this process. All of these have the following values set:

- Target size: 224 X 224
- Batch size: 16
- Rescaling size: 1 / 255
- Vertical Flip: True
- Horizontal Flip: True
- Shear range: 0.1
- Rotation range: 45

## 3.3 Data Source

Acevedo's et al.[8] PBC photos are included in this dataset. The dataset is split into eight folders, one for each of the image categories (basophil, erythroblast, ig, platelet, eosinophil, lymphocyte, monocyte and neutrophil). There is a total of 32,768 JPG pictures belonging to 8 classes. Here are pictures of 8 different classes:

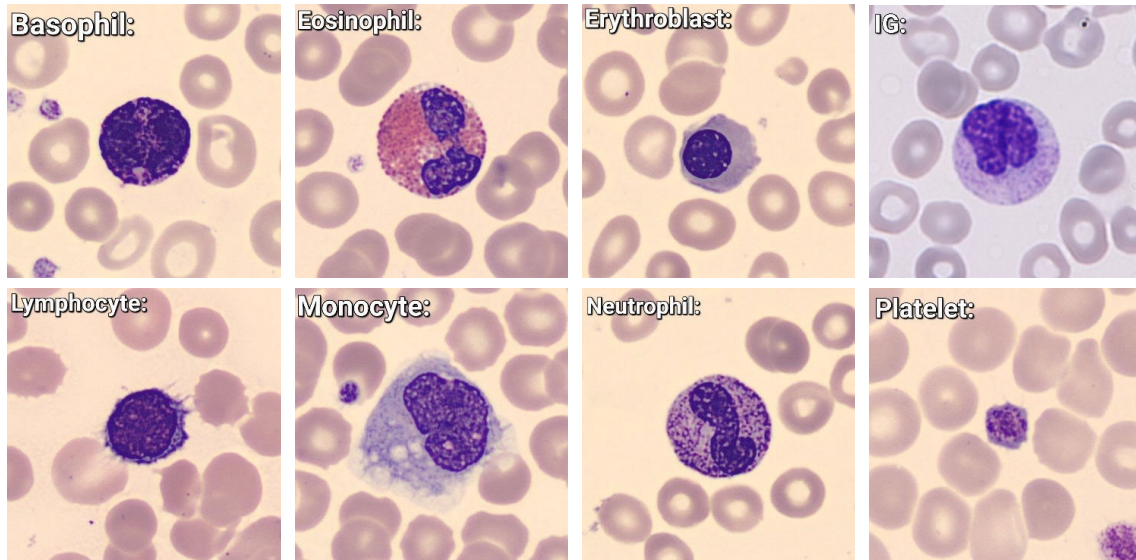


Figure 3.2: Cell images

### 3.4 Data Classification

A training dataset is used to update the model weights for a successful input-to-output mapping. Neural networks are trained using data and solutions from the actual world, which enables them to transform data into a consistent input-output relationship. A source of input for a system could be output labels or algorithms. An optimization strategy is utilized to manage the training phase of the neural network model, searching through a space of possible weight values for a set of weights that exhibit good performance on the training dataset. The algorithm that we used will learn from the training set in order to benefit from the performance of real-world samples. The algorithm's trust in the present will increase as a result of the successful findings for an unidentified test collection. An objective evaluation of model fit is used to adjust the model's parameters on our training dataset. When the validation dataset's effectiveness is chosen and the chosen model is mentioned in the design model, an unfair measurement happens. As demonstrated in the Data Sample image, we used 80% of our data to train the model and 20% of our data to validate it for each class. For the project, we ultimately decided on an asymmetrical ratio of 8:2.

Train	Validation	Total
3277	819	4096

Table 3.1: Table for Individual Class

### 3.5 Data Pre-processing

Before implementing CNN algorithm, we need to apply appropriate image processing techniques to our dataset if needed. As images from datasets may vary in quality, we would need our RBC, WBC and platelets to stand out in particular before applying to the CNN and generating an output.



### 3.5.1 Image Resizing

Multiple models will be used in our study. As a result, during the network training process, each blood cell image must be scaled to a size of 224 x 224 pixels (However, some images had to be changed to other resolutions due to model compatibility). TensorFlow and the Keras libraries were used to achieve this.

### 3.5.2 Image Augmentation

Our image would first need to be augmented. This means that, for our neural network, we will have to change our source images through flips, zooms or crops to select out a specific portion of the image, such as a portion containing a White Blood Cell. Since one image from the dataset contains multiple blood cells, it can allow us to generate fresh training data artificially using preexisting training data. It is useful for training our neural network for identifying/classifying peripheral blood cells. Additionally, image augmentation is used to increase the size of our dataset, scaling, cropping, and rotation. With image augmentation settings, flipping, translation, affine transformation zooming, and photo sharing were also possible. This was used in order to increase the recognition of our trained models by producing additional images and changing their placements.

### 3.5.3 Image Scaling and Normalization

Grayscale is a technique where we can change the color code of our images by equalizing the RGB pixel values and converting them to a singular value in the grayscale. This can be achieved by  $[ \text{GreyScaleValue} = ( \text{RedValue} + \text{BlueValue} + \text{GreenValue} ) / 3 ]$ , which is taking the average of the RGB values. Afterwards, we can apply Normalization. This is a technique that can change the maximum and minimum of pixel intensity values. The motivation to do this is that sometimes, images may not have the same consistency as each other, and that normalization is a means to achieve said consistency. We apply this after Grayscale to make sure that the images of the same type of blood cell do not have different grayscale values, or else the neural network may misclassify it into a different entity. It is possible to give system data to either the inputs themselves or the activation functions of a preceding layer. This procedure is known as batch normalization. The training is expedited, and some generalized linear is provided, which lowers generalization mistakes. Internal covariate shift is one of the main issues that batch normalization solves. The output of the earlier levels can be equalized using batch normalization, a network architecture that enables each layer to adapt more independently.

### 3.5.4 Image Thresholding

Thresholding is a process that generates a binary image of black and white. This is done so that, after a certain threshold, which is a grayscale value, a pixel in the image will be counted as black, if it is gray enough, or white if it is not gray enough. We apply image thresholding after converting the images to black and white in an attempt to improve training speed and observe potential changes to accuracy.

## 3.6 CNN and Non-linearity

Afterwards, we can apply our convolutional deep learning algorithm to train our neural network and then subsequently use it to classify our peripheral blood cells. We may repeat the convolution and max pool segments several times to lower our node counts during the flattening layer. In addition, a confusion matrix may be implemented, where the columns indicate the expected values and the rows reflect the actual ones.

# Chapter 4

## Description of the Model

### 4.1 Model

First of all, we will gather and prepare the data. We will partition our dataset in a 8:2 ratio after preprocessing. Whereas, we will use 80% for training and the remaining 20% for validating the models we've already used. Our study will employ a number of deep learning models, including VGG19, MobileNetV3, ResNet50, ResNet101, EfficientNetV2B1, DenseNet121, NASNet, Inception Network and Custom CNN Model. The performance of our models' prediction will then be compared. The top two performing structures will be determined through comparison, and we'll combine them for an even more effective outcome.

### 4.2 CNN Model

CNN has multiple layers. CNN's convolutional layer converts input to output using a convolutional algorithm. As the product of the convolutional layer is provided to the ReLu function, the network is able to train more quickly and make more accurate classifications. The layer which is responsible for pooling is used to reduce the map dimension. Fully connected and output layers produce functions for probability distributions. Moreover, CNN is a hierarchical neural network which is widely functioned as a tool for processing images, segmentation, anomaly detection, classification, etc. It has several convolutional layers. A neural network is used to classify various items in a picture or the whole picture itself. It is a very popular deep learning tool used in every field. CNN mainly has three levels: convolution, pooling, and the fully connected layers. Convolution and pooling layers helps to identify an image or any feature of the image. Feature Extraction uses the convolution layer result to determine the class of the image. In a neural network, there are mainly three components that are an input layer, a processing layer, and an output layer.

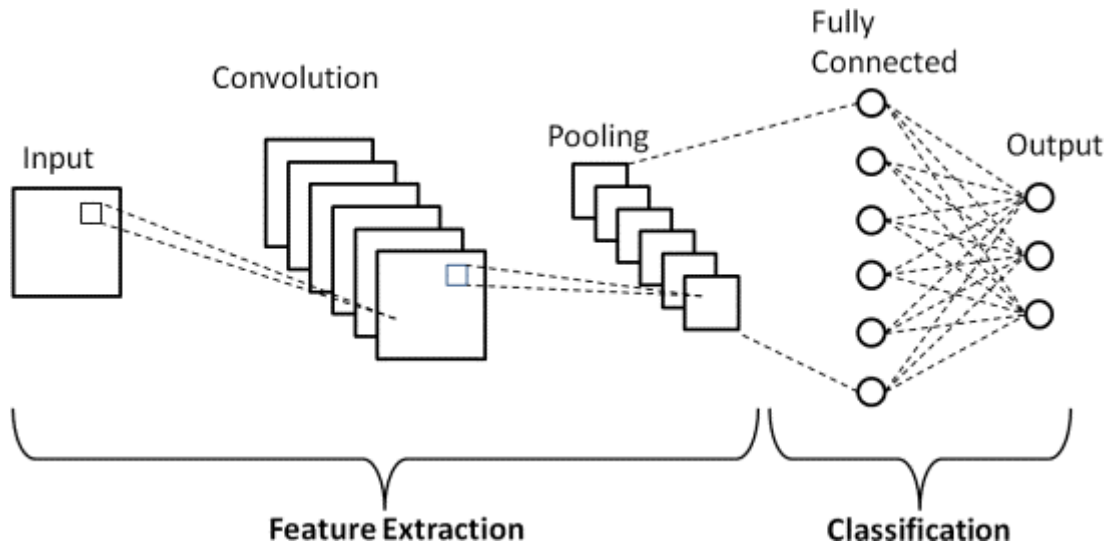


Figure 4.1: CNN Architecture

### 4.2.1 Convolutional Layer

A CNN's primary component is its convolution layer. In this layer there are set of kernels or filters, parameters which are used to learn by training. Filters are applied to the input image or other feature maps in the convolutional layers of a deep CNN. The majority of the network's configuration parameters may be located in this section.

$$Equation = (((m * n * d) + 1) * k) \quad (4.1)$$

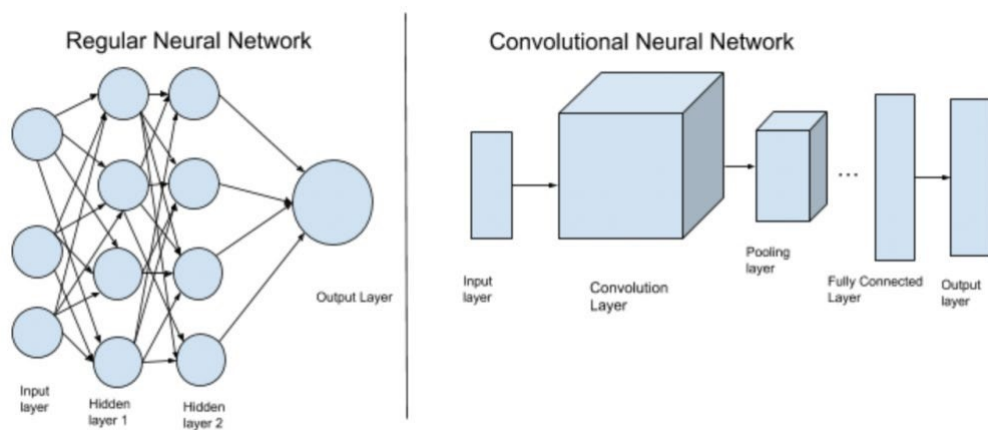


Figure 4.2: Regular Neural Network vs CNN

### 4.2.2 Pooling Layer

In a convolutional neural network, in most cases, the fully connected layers and the convolutional layers are connected with a pooling layer. We applied max pooling to our model also. The pool size was 2 x 2.

### 4.2.3 Fully Connected Layer

Later on, we completed the full connection, often known as connecting the layers. The Fully connected layer constitutes of neurons, weights, and biases. It serves as the connection between the two layers. Basically, this layer is addressed to facilitate in the classification process. In this stage, we fused two dense layers of units of 256 and 128 after the flattening layer. These are done before the final output layer (of which there are 8 units since we are working with 8 classes)

$$\text{Equation} = ((\text{current layer neurons } c * \text{previous layer neurons } p) + 1 * c) \quad (4.2)$$

### 4.2.4 CNN Model Summary

Essentially, the quantity of parameters in a given layer is the total number of "learnable" elements for a filter, or parameters for that layer's filter, if such a term exists. After dividing the dataset into train data and validation data, we used a neural network framework called Keras to create a sequential CNN model for the suggested system. This was done in order to evaluate the model's precision. As well as the same amount of max pooling layers, our model had a total of 7 2D convolutional layers. Then we applied two layers of density after flattening and then the output layer.

Model: "sequential\_1"

Layer (type)	Output Shape	Param #
conv2d_7 (Conv2D)	(None, 224, 224, 32)	896
batch_normalization_7 (Batch Normalization)	(None, 224, 224, 32)	128
max_pooling2d_7 (MaxPooling2D)	(None, 112, 112, 32)	0
conv2d_8 (Conv2D)	(None, 112, 112, 64)	18496
batch_normalization_8 (Batch Normalization)	(None, 112, 112, 64)	256
max_pooling2d_8 (MaxPooling2D)	(None, 56, 56, 64)	0
conv2d_9 (Conv2D)	(None, 56, 56, 128)	73856
batch_normalization_9 (Batch Normalization)	(None, 56, 56, 128)	512
max_pooling2d_9 (MaxPooling2D)	(None, 28, 28, 128)	0
conv2d_10 (Conv2D)	(None, 28, 28, 128)	147584
batch_normalization_10 (Batch Normalization)	(None, 28, 28, 128)	512
max_pooling2d_10 (MaxPooling2D)	(None, 14, 14, 128)	0
conv2d_11 (Conv2D)	(None, 14, 14, 128)	147584
batch_normalization_11 (Batch Normalization)	(None, 14, 14, 128)	512
max_pooling2d_11 (MaxPooling2D)	(None, 7, 7, 128)	0
conv2d_12 (Conv2D)	(None, 7, 7, 128)	147584
batch_normalization_12 (Batch Normalization)	(None, 7, 7, 128)	512
max_pooling2d_12 (MaxPooling2D)	(None, 3, 3, 128)	0
conv2d_13 (Conv2D)	(None, 3, 3, 128)	147584
batch_normalization_13 (Batch Normalization)	(None, 3, 3, 128)	512
max_pooling2d_13 (MaxPooling2D)	(None, 1, 1, 128)	0
flatten_1 (Flatten)	(None, 128)	0
dense_3 (Dense)	(None, 256)	33024
dense_4 (Dense)	(None, 128)	32896
dense_5 (Dense)	(None, 8)	1032

=====  
Total params: 753,480; Trainable params: 752,008; Non-trainable params: 1,472

Figure 4.3: Layers

### 4.3 VGG-19

We also used the VGG-19 layer as the output of the data was giving a low accuracy rate. One recent groundbreaking development is layered convolutional neural networks (VGG19) that have already been trained and a great understanding of what shapes, colors, and structures make up an image. In-Depth Learning System In order to perfect its categorization abilities, VGG19 has been exposed to millions of images from various sources. The architecture of VGG19 is almost the same as that of VGG16, but it has 19 layers, which is three more layers than VGG16. The model consists of 19 layers and is an enhanced convolution neural network. Convolutions are used to build the model, however the vanishing gradient problem limits the number of layers that may be used.

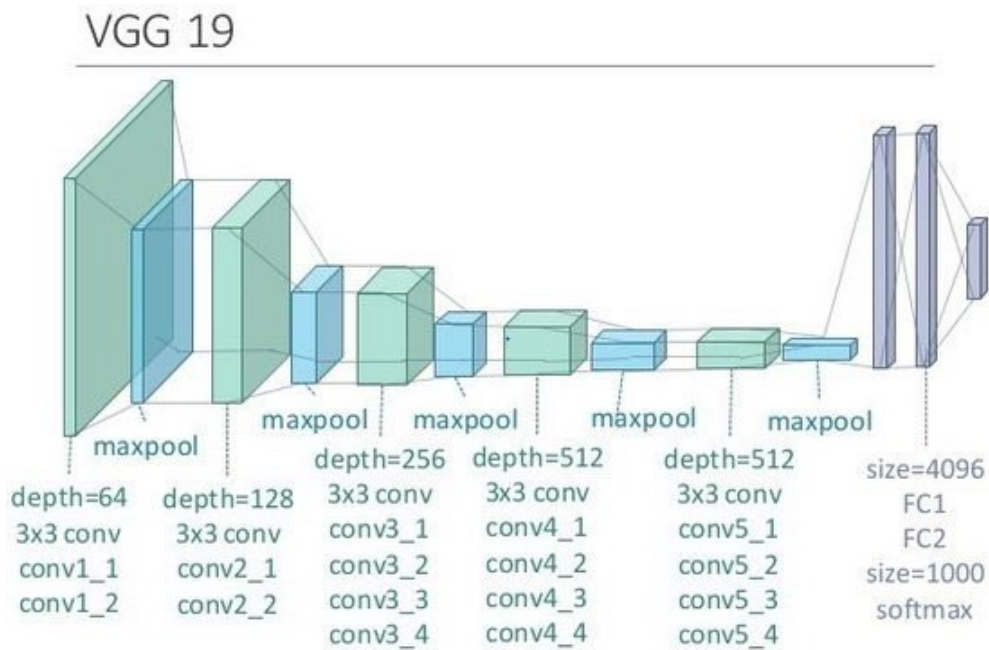


Figure 4.4: Network Architecture of VGG19 Model

## 4.4 MobileNetV3

MobileNetV3 is the most recent version of the neural network architecture used in artificial intelligence and machine learning. It employs AutoML to determine which neural network design is most effective for an agglomerative task. The building's earlier iterations were constructed by hand. MobileNetV3 utilizes NetAdapt and MnasNet. The optimal configuration is found by MobileNetV3, which employs reinforcement learning to build a course architecture. Any inactive activation channels are then disabled by NetAdapt. This aids in optimizing the design. Another novel concept in MobileNetV3 is the squeeze-and-excite block. Squeeze-and-excite blocks enhance network representations by highlighting the interdependence of convolutional channels. We want to alter how the network operates. Therefore, in order to choose which qualities to prioritize first, the network may employ global data. Additionally, MobileNetV3-Large, created for use with plenty of resources, is 20% faster and 3.2% more accurate at classifying images with ImageNet. MobileNet V3-Large is nearly 25% faster than MobileNet V2 at finding COCO while being almost as accurate. While still 6.6% more accurate than a MobileNetV2 model with the same latency, MobileNetV3-Small strives to use fewer resources.

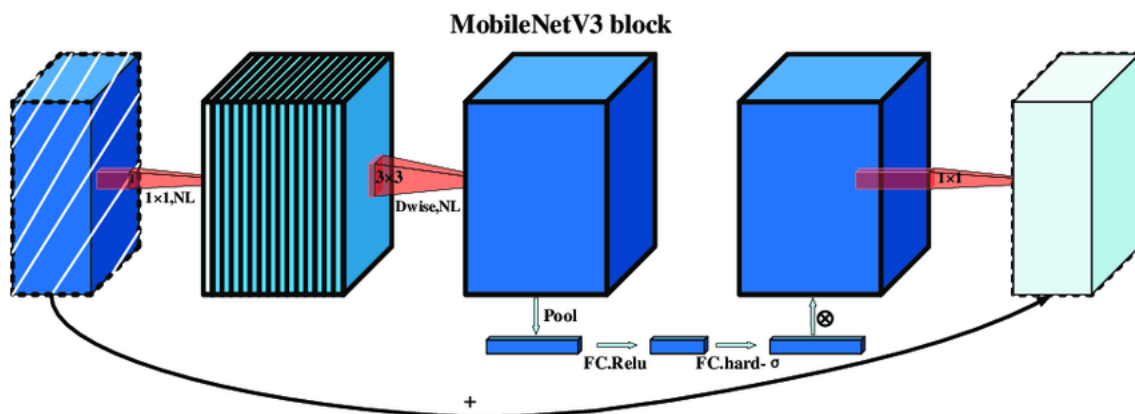


Figure 4.5: MobileNetV3



## 4.5 ResNet50

Resnet is a convolutional neural network that could be applied as a cutting-edge model for classifying images. The massive classification dataset known as ImageNet served as the pre- training data for the ResNet models. 50 layers make up ResNet-50, which was trained using 1 million images from 1000 different ImageNet categories. A sophisticated architecture is implied by the roughly 23 million trainable characteristics, which enhance photo identification. The vast ImageNet dataset contains a wide range of image classifications, thus there is a good chance that images very similar to yours have already been used for pre-training purposes. Using a model that has already been trained is more efficient than creating one from scratch and training it yourself. A 34-layer plain network that was influenced by VGG-19 is included in the design. In this system, certain connections have been omitted or shortened. These omitted connections or residual blocks subsequently change the architecture into the residual network. Each of the five phases of ResNet-50 has a residual block. There are 1\*1 and 3\*3 convolution layers in each residual block. Simple residual blocks exist. In typical neural networks, each layer feeds the following layer. Each layer feeds into the next tier and then directly onto the following tiers, which are spaced 2-3 hops apart, in a network with residual blocks.

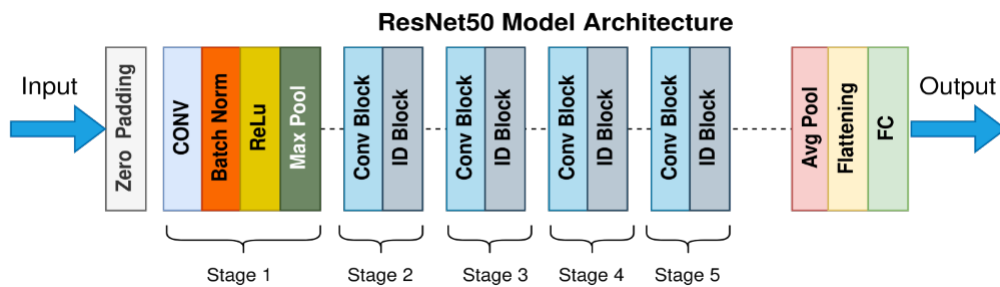


Figure 4.6: ResNet50

## 4.6 EfficientNetV2B1

EfficientNet is a method for designing and scaling convolutional neural networks that proportionally increases or decreases the depth, breadth, and resolution of the original data using a compound coefficient. To scale up models effectively and efficiently, EfficientNet uses a compound coefficient technique. Instead of arbitrarily expanding the width, depth, or resolution of each dimension, compound scaling applies a consistent set of scaling factors to each one. Consuming the scaling technique with AutoML, the creators of Efficient were able to create seven models with varying dimensions that outperformed the best convolutional neural networks of the time while using less energy. The baseline network found by the AutoML MNAS framework's neural architecture search serves as the basis of EfficientNet. The accuracy of networks is maximized, while those that need a lot of processing resources are punished. The inference time score decreases when the network makes predictions slowly.

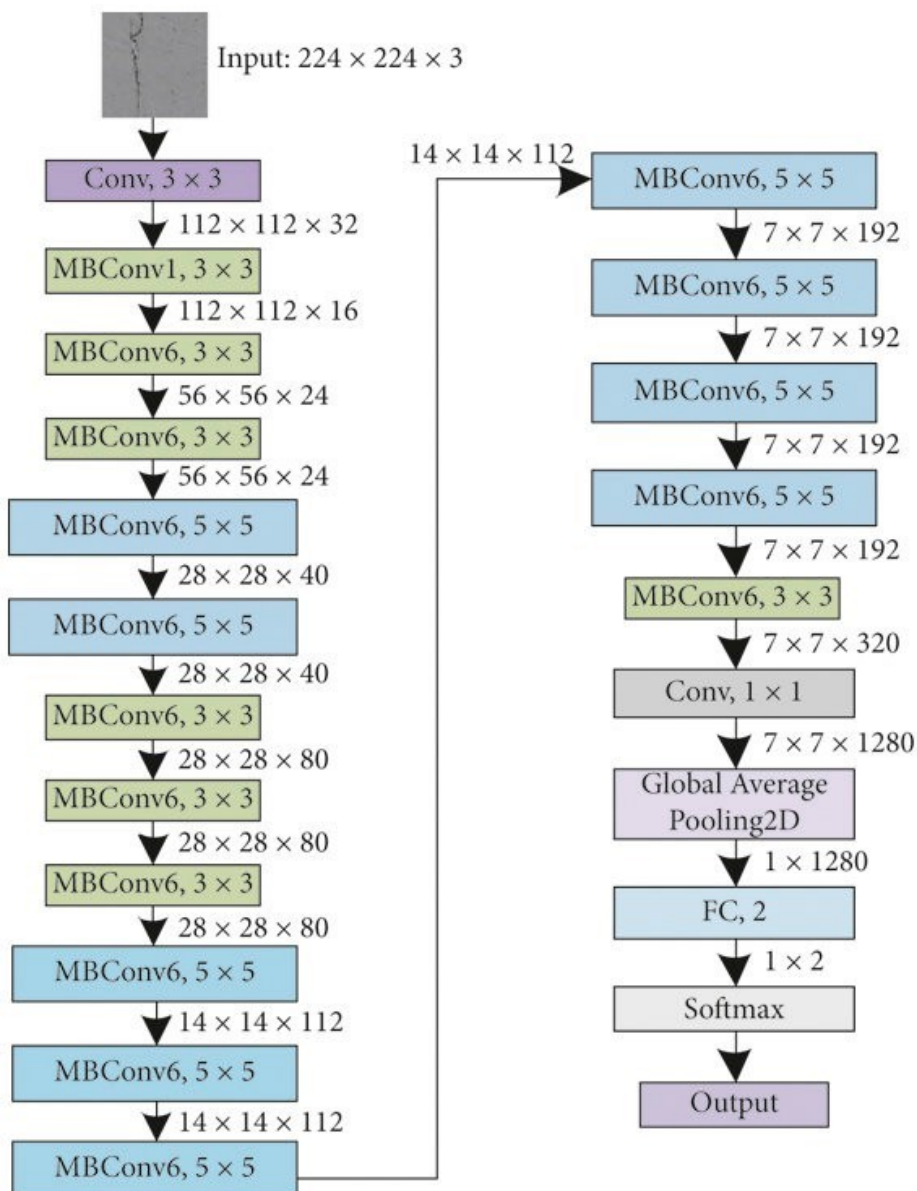


Figure 4.7: EfficientNet Architecture

## 4.7 DenseNet121

In a DenseNet design, each layer is immediately linked to each subsequent layers; therefore from this reason the name "Densely Connected Convolutional Network." was given. DenseNet-121 consists of 120 convolutions and 4 avgPools. To use prior layers' knowledge, all levels, in which dense block is included and also transition layers are included, across a spectrum of inputs their weights are spreaded. Since they contain the most redundant attributes, layers which are located in the second and in the third dense blocks weighs the transition layer output. Despite the fact that the final layers uses the weights from the total dense block, studies shows that the model can be farther created by implementing higher-level features. DenseNets outperform CNNs and ResNets on several benchmark datasets. This is because they may create models that are more compact and effective by using fewer parameters and reusing characteristics.

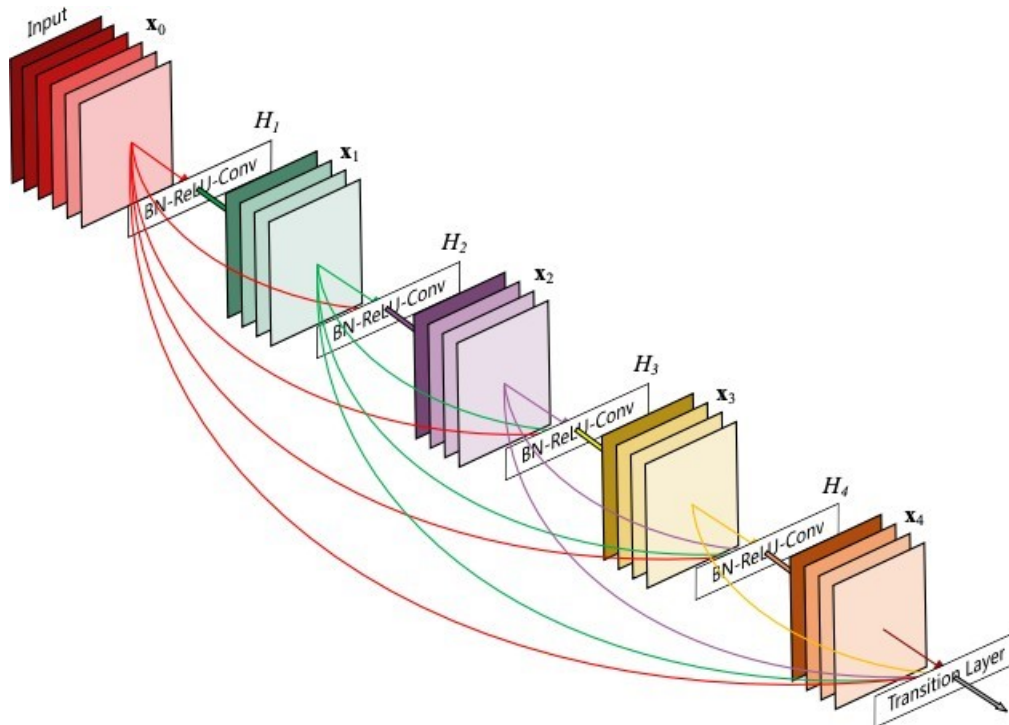


Figure 4.8: DenseNet121

## 4.8 NASNet

Neural Search Architecture Network (NASNet) is an example of a model for Machine Learning that goes by the name NASNet. The creation of network architectures is automated by the Neural Architecture Search (NAS) program. It is an algorithm that looks for the best algorithm to use in order to perform a specific task as effectively as possible. Neural Architecture Search (NAS) has received a lot of attention as a result of the groundbreaking studies carried out by Zoph and Le in 2017 and Baker et al. in 2017. This attention has resulted in the development of many intriguing ideas for NAS approaches that are better, faster, and more cost-effective. These three elements are present in the conventional NAS configuration:

1. Exploring Outer Space.
2. The Search Methodology.
3. Method for Estimating Future Performance.

Deeper neural networks and a range of architectures were produced by NAS using RL and evolutionary algorithms. This achievement was achieved with great success.

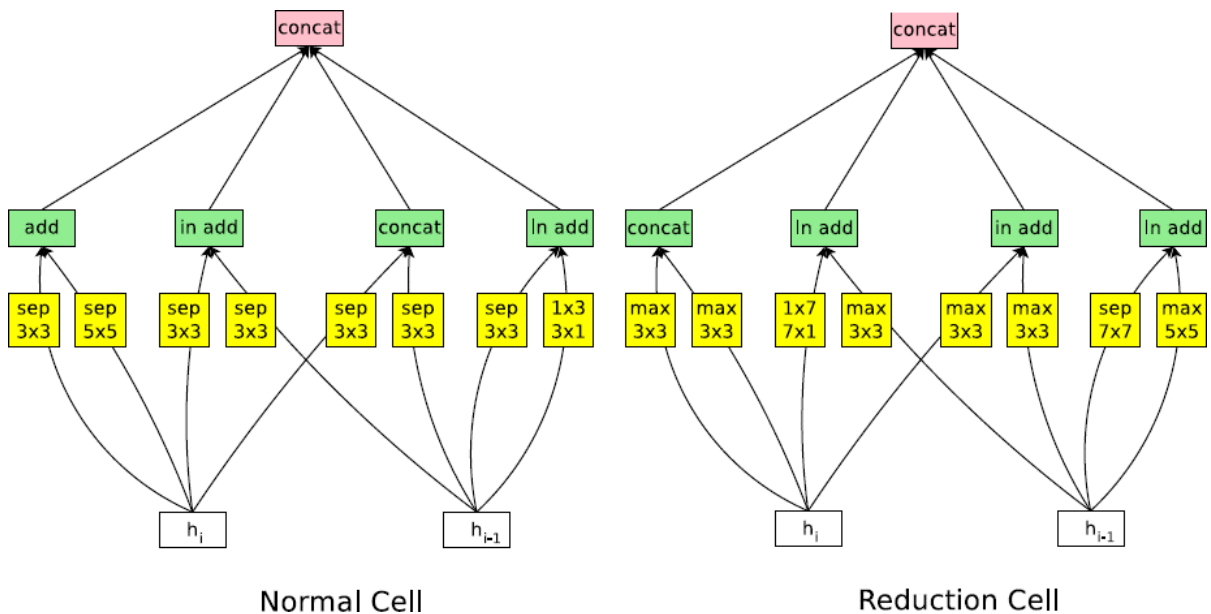


Figure 4.9: NASNet Architecture

## 4.9 Inception Network

A novel deep learning convolutional neural network design was introduced in a study published in 2014 by Google and other research organizations. It was the biggest and most effective deep neural network architecture at the time. A GoogleNet variant of the Inception Network, which served as the innovative architecture, later achieved the best performance possible in the classification computer vision problem of the 2014 ImageNet LargeScale Visual Recognition Challenge (ILVRC14). Inception modules are recurring parts in a deep neural network's architecture. This module's technical specifications are the main topic of this page, as was already mentioned. Here are some more details on this article's scope.

- The origination of the name given to the Inception Module.
- Source of data on the key concepts and notions that influenced the Inception module's architectural designs.
- Specifications of the Inception module's constituent parts.
- Illustrations that show the Inception module's internal architecture and structure in depth.
- Calculations to determine how many multiplier operations are performed by each component of the Inception module.

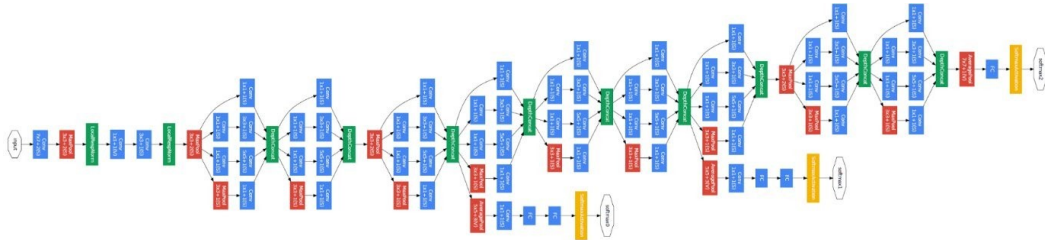


Figure 4.10: Inception Network

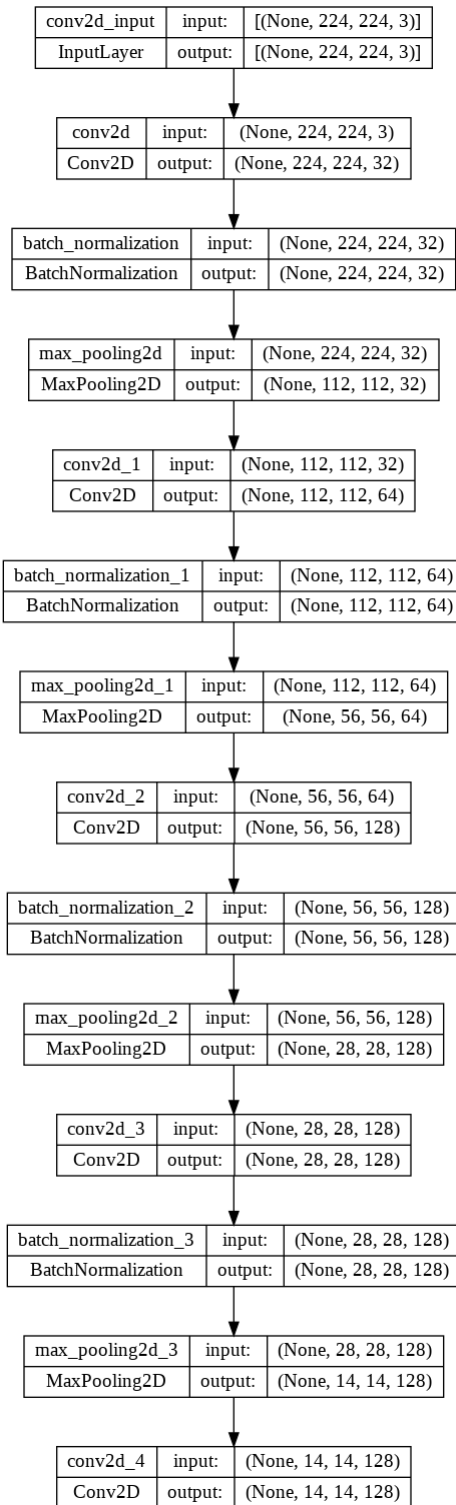
## 4.10 A Custom Model

Our neural network is composed of three primary layers: convolutional, maxpool, and batch-normalization. The convolutional layer contains the configurations of our kernels, which our model trains on to comprehend and categorize our input.. These kernels are also called filters, and these filters extract specific data regarding the images that can be used to classify said image through a dot product. For example, a filter can be used to capture the shape of a lymphocyte nucleus, which is round in nature. The filter will carry information for a rounded object. Likewise, other filters for other cell types will carry different information, ranging from shape, size to even color. We use 7 convolution layers, with and the filter are of 3x3 in size. We initially start with 32 filters and increase it do 64 and then to 128.

The next part is the MaxPool2D layer. MaxPooling improves the neural network's efficiency and lessens the features it must extract. The number of adjustable settings has shrunk, indicating this. With a 2x2 pool size, it finds the largest region in the input that the kernel from the aforementioned convolutional layer is using to create a feature. For example, a 4 x 4 matrix (such as an image) can be segmented into 4, 2x2 parts. It finds the maximum of each part and discards the other non-max values. The result is a 2x2 matrix comprised of only the maximums. This reduces the number of parameters (or details) we have the check and thus improving performance.

We then have a BatchNormalization layer. This is done as a way "stabalize" the layers by recentering and rescaling, as unstable filters (non-centered or out-of-focus) can cause variations in training.

We repeat the above 3 times together before finally flattening to 8 dense layers (though it is worth mentioning we have two dense layers), all the inputs of the dense layers are outputs from the previous layer, and are fully connected.



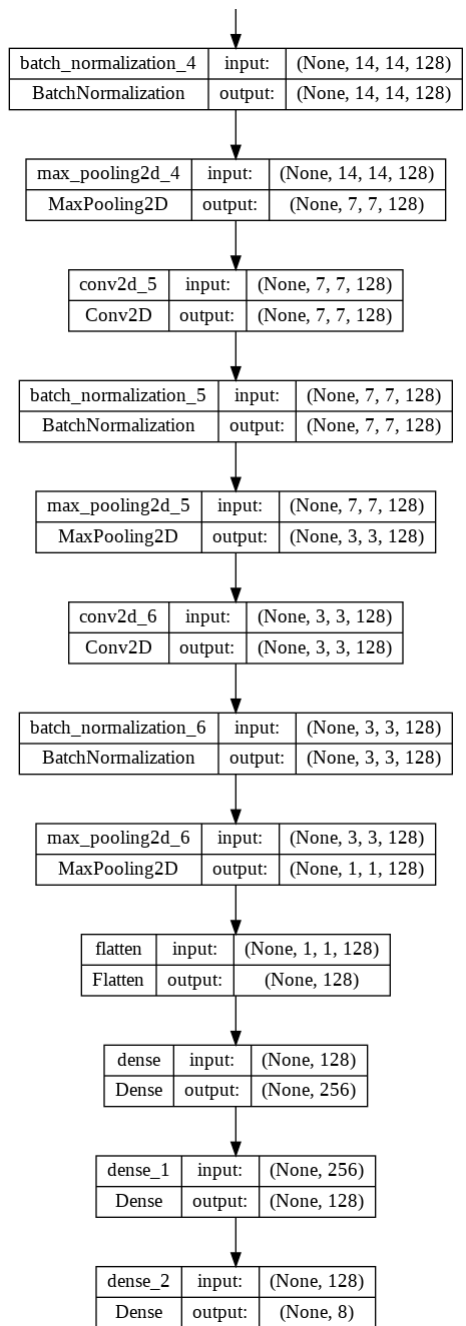


Figure 4.11: Custom Model Architecture



# Chapter 5

## Preliminary Analysis

### 5.1 Result and Analysis

For our analysis, we had run several different CNN pre-trained models with varying layer counts for 40 epochs. We have run EffecientNetV2B1, MobileNetV3Large, Inception, DenseNet121, NasNet, ResNet50, ResNet101, VGG19 and Custom CNN Model. One conclusion that we drew when running these models is that a specific value of the number of layers often provide better training and validation accuracy. The general sweet spot for the number of layers for this dataset appears to be around 300 to 600 layers. Anything lower generally results in poor training and validation accuracy. Anything higher on the other hand, while they do show good training accuracy, the validation accuracy was subpar, albeit still better compared to those with lower layer counts. Such models were very likely to be overfitting, despite having anti-overfitting measures in place. However, this does not always appear to be the case. Our custom model has performed the best, it has a training accuracy rate of 99.1% and a validation accuracy of 98%. Which is the highest among all these models but with a much lower layer count and better learning speed. Among these models Inception network produced a training accuracy of 99% and a validation accuracy of 97.3%, which is still almost 3 percent lower than our custom model. Similarly EffecientNetV2B1 yield a 99.2% and 93.8% training and validation accuracy and DenseNet12 model gave accuracy results of 80.8%, 72.7%, which significantly less accuracy than compared our custom model in training and validation respectively. Other models gave good training accuracy results but the validation accuracy was not good they had overfitting/underfitting problems. As a result, we decided to select DenseNet121, EffecientNetV2B1, Inception and NasNet as our models for discussion and performance analysis to compare with with our custom model. Models such as MobileNetV3Large, ResNet50, ResNet101, VGG19 produced rather poor results ranging from 10% to 15% accuracy in both training and validation. There is a possibility that these low accuracy values may be tied to the device being used than being an issue with the dataset or model itself.

It is also worth mentioning the stability of the models during training. Only the custom model stabilized before 36 epochs compared to the other models, which was still unstable even after 40 epochs.

## 5.2 Performance Analysis

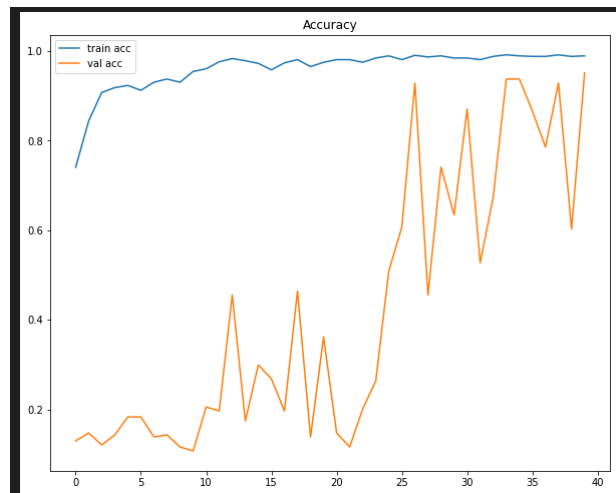


Figure 5.1: EfficientNetV2B1 accuracy graph

In the above graph, it can be seen that validation accuracy slowly converges at first then after some time it rapidly converges and gives good validation accuracy

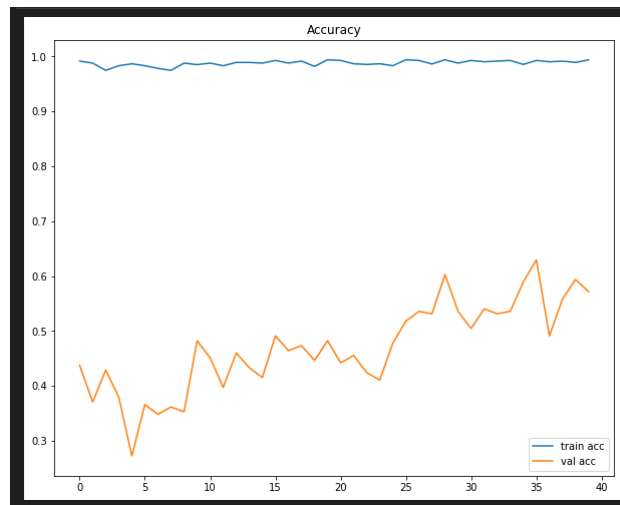


Figure 5.2: NasNet accuracy graph

In the following graph for NasNet, we can see that although training accuracy is good, validation accuracy doesn't converge all that much and occasionally diverges a bit, leading to an overfitting issue.

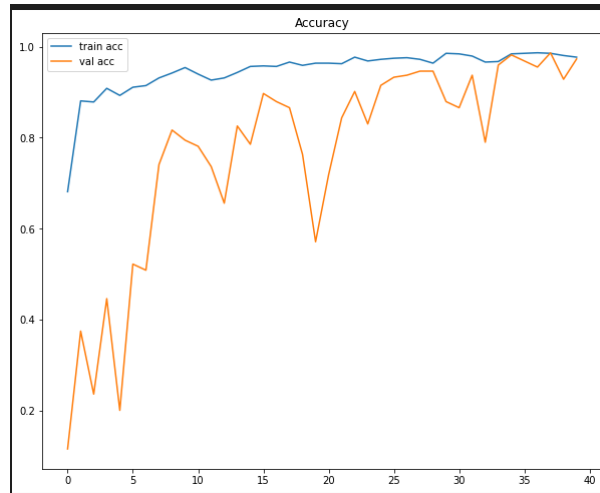


Figure 5.3: InceptionNet accuracy graph

Here, in the above graph of Inception model it can be seen that the validation accuracy converges in a very manner and also the accuracy is on par with the validation accuracy that's why it produces good accuracy and good validation accuracy results and no overfitting problem is seen.

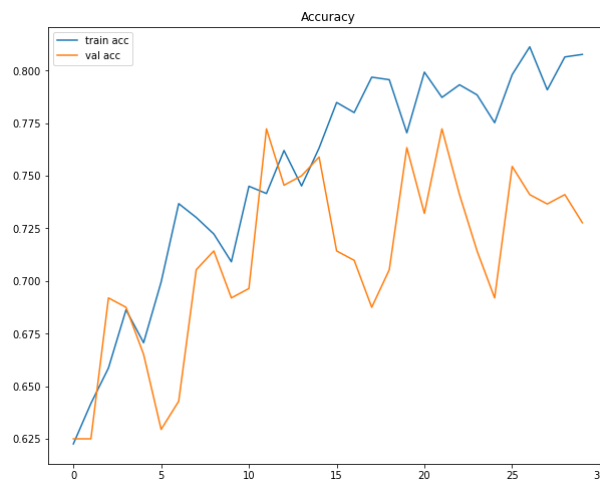


Figure 5.4: DenseNet121 accuracy graph

DenseNet121, while it did not achieve outstanding accuracy like Inception or EfficientNet, had still shown that its validation accuracy and training accuracy was rising. Perhaps, with a few more epochs, it might have reached over 90% accuracy. However, this also shows just how fast models like our custom model, Inception and EfficientNet are in terms of converging and stability during training.

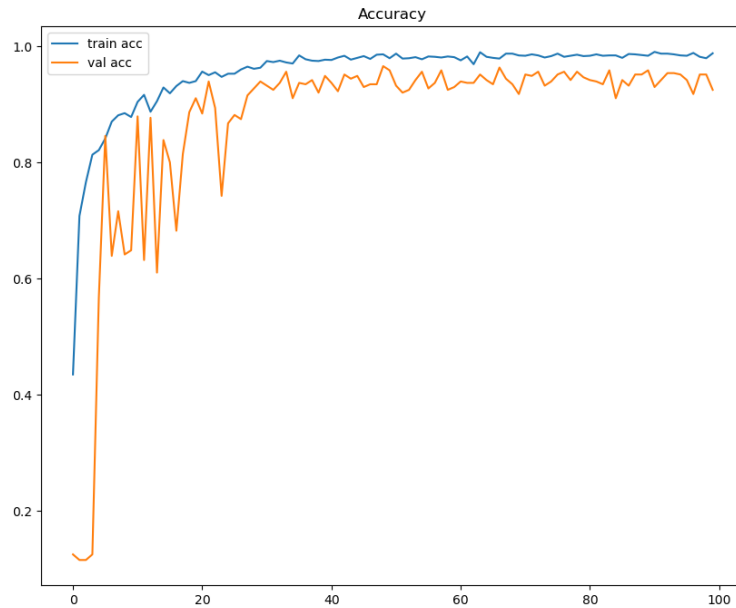


Figure 5.5: Custom Model Accuracy Graph

In the above custom model accuracy graph, training accuracy and validation accuracy converges almost immediately. After 40 epochs, the validation and training accuracy stabilizes and averages to around 98.9 to 99.1 percent.

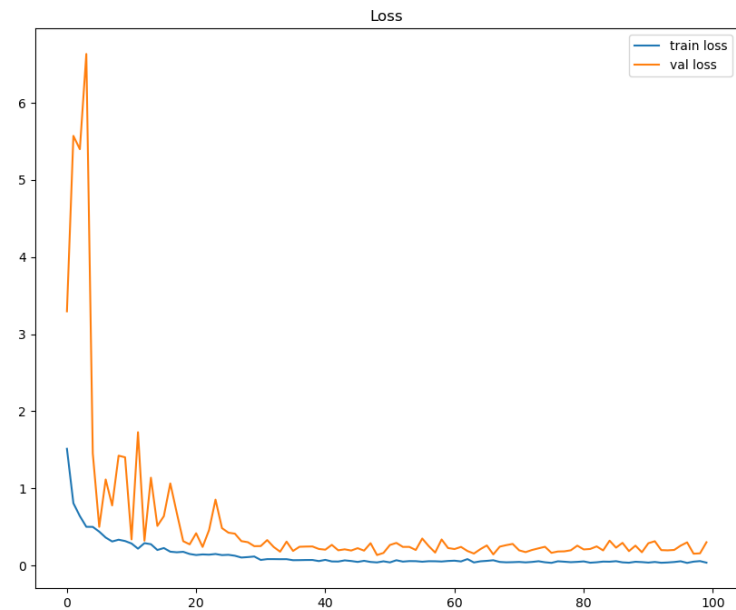


Figure 5.6: Custom Model Loss Graph

The loss graphs show that the fast initial converges causes the loss values to plummet and stabilizes entirely after 40 epochs, to near zero values.

Basophil	809	0	6	0	0	2	0	2
Erythroblast	0	807	4	0	0	2	5	1
I.G.	18	28	560	0	1	7	167	38
Platelet	0	0	0	819	0	0	0	0
Eosinophil	0	0	0	2	813	0	0	4
Lymphocyte	1	0	1	0	0	811	5	1
Monocyte	0	5	9	0	0	5	798	2
Neutrophil	3	3	40	2	4	1	1	765
	Basophil	Erythroblast	I.G.	Platelet	Eosinophil	Lymphocyte	Monocyte	Neutrophil

Figure 5.7: Confusion Matrix Based on Custom CNN Model

Class	precision	recall	f1-score	support
basophil	0.97	0.99	0.98	819
erythroblast	0.96	0.99	0.97	819
ig	0.90	0.68	0.78	819
platelet	1.00	1.00	1.00	819
eosinophil	0.99	0.99	0.99	819
lymphocyte	0.98	0.99	0.98	819
monocyte	0.82	0.97	0.89	819
neutrophil	0.94	0.93	0.94	819
accuracy			0.94	6552
macro avg	0.95	0.94	0.94	6552
weighted avg	0.95	0.94	0.94	6552

Table 5.1: Classification Report Based on Custom CNN Model

From the above confusion matrix and classification report, we can see how well our model performed when classifying the 7 different white blood cell types and the platelet. There are a few things of note here: Our model showed an average of 95% precision and recall and a 94% f1-score. Of these, our model is had achieved a 100% in all 3 aspects when identifying a platelet. This may be due to overfitting or bias towards the dataset, which would be indicative of a future generalisation issue. However, in the model’s defense, a platelet is very different from the white blood cell, being essentially a cell fragment rather than a whole cell, so it is missing all features that would distinguish it from a white blood cell.

On the other hand, the model struggled to differentiate between an ig from a monocyte, which is reflected by the confusion matrix (167 images were wrongly classified as a monocyte) and the low f1-score of 78% for ig. Nevertheless, it still performed considerably well for all other types.

### 5.3 Result Comparison

Our custom model yields a competitive 99.1% training accuracy and a very good 98% validation accuracy, which is the highest among all the models we have used. The EffecientNetV2B1 achieved a 99.2% accuracy during training and a 93.8% accuracy during validation. As a result, the model does not suffer from overfitting. However, NasNet cannot make the same claim. NasNet was able to complete its tasks more quickly than EffecientNetV2B1 and had a training accuracy of 99.4%, but its validation accuracy was just 59.4%, suggesting that it had been overfit. On the other hand, the Inception model gave promising result; the training accuracy was 99.0%, and the validation accuracy rate was 97.3%. The accuracy of the Inception model was close to that of our custom model, but our custom models produced better validation accuracy. DenseNet121’s accuracy of training was just 80.8%, with only 72.7% accuracy during validation, which is lower than both Inception and our custom CNN model. Moreover, in MobileNetV3 the training accuracy was 99% but the validation accuracy is very poor which is only 15% so here the overfitting problem is seen when using this model. ResNet50, ResNet101 and VGG19 was observed to not train at all with the dataset and produced very low accuracy values. It is also worth mentioning that although Inception produced similar accuracies to our custom CNN model, our custom model uses less parameters and therefore it runs much faster and is more lightweight. Among all of these, it’s clear that our custom model has the highest accuracy and is the best solution that we have.

Model	Training Accuracy	Validation Accuracy
Custom Model	99.1 %	98.0%
Inception	99.0%	97.3%
EffecientNetV2B1	99.2%	93.8%
NasNet	99.4%	59.4%
DenseNet121	80.8%	72.7%
MobileNetV3	99.0%	15.0%
ResNet50 and ResNet101	12.2%	12.1%
VGG19	15.3%	12.0%

Table 5.2: Accuracy and Validation percentages of the models used.

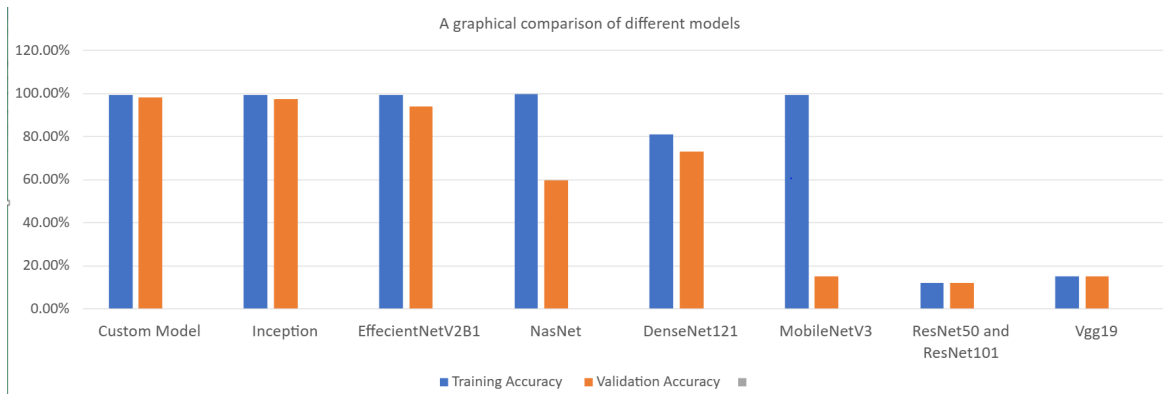


Figure 5.8: Graphical Representation of different models

# Chapter 6

## Challenges

Initially, we wanted our model to detect 8 different types of blood cell (7 white blood cell and platelets). But due to some hardware limitations and mainly different problems with the pre trained CNN models and their module we were getting less accuracy numbers. We then reduced our parameters a bit and started the classification with 3 classifications. As we used CNN and deep learning in our model, some typical challenges of CNN were seen at first there was a huge underfitting problem, indicated by a poor validation and training accuracy. Our code required several revisions until we found the solution to the underfitting problem and then reverted to classifying 8 classes instead of 3.

We used a lot of pre trained models which were very GPU demanding, or required a version of tensorflow which we did not have access to. Among these models, Inception model, MobileNetV3, DenseNet had more than 200 layers. Since we didn't initially have that specific version of tensorflow that can run these models, we had resorted to using google colab and its Nvidia CPUs which took time and was a challenge to train the image data of 8 classification under a CUDA environment. Eventually, we were able to afford a new device with an RTX 3050 as well as a fresh tensorflow installation to remove the need to use google colab for a CUDA enabled environment.

We had enough data to train our models, but one issue was that most of the data in the dataset were imbalanced, as a result we needed to cut some images so balance all data so that we had just enough to produce a reliable result on the models.

# Chapter 7

## Conclusion and Future work

### 7.1 Conclusion

Disease detection by analyzing blood cells has the potential to revolutionize the world and save lives. In this research, we attempt to design a solution-providing algorithm that is both efficient and superior. We have used CNN's deep learning algorithms to create a model to classify peripheral blood cells and intend to enhance the system further. Right now, our best models that we have run are the custom CNN model, Inception Network and EfficientNetV2B1. DenseNet121 was unstable, and was taking too long to reach a higher accuracy. Inception Network and EfficientNetV2B1 provided the good accuracy in training, but not against our custom model. Our proposed custom model for this study was able to perform competitively with other models at 98 percent validation accuracy, which is the highest of the 3 while also being relatively lightweight, as such we believe it is much better for this task at hand.

### 7.2 Future work

When it comes to classification of images of blood cell and to diagnose diseases it is becoming more popular and accepting in the medical field. Also, it is seen that many blood diseases (such as cancer cells) are different in quantity and shape compared to the healthy ones so by using our model and expanding the parameters / dataset more we can train our model to detect these abnormal cells. As a scope for future work, we have plans to add a feature that is it will be able to detect if a person is affected by a disease or not as usually the person will often have physically altered blood cells or have abnormal counts. Such a feature will have value for men, women and children where our model will just compare those numbers with the numbers or quality of cells of the disease affected person. As a result, it will be possible to diagnose the illness and also will be able to identify if any kinds of infection that is present on the blood cells. To put it simply, it will have the ability to determine whether or not a certain blood cell is infected, or where there is too many of a blood cell (ie leukemia). Moreover, we also plan to have better represented results with Explainable AI (XAI), such that predictions can be explained to staff such as doctors or patients.



# Bibliography

- [1] R. Roy and S. Sasi, “Classification of wbc using deep learning for diagnosing diseases,” in *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, IEEE, 2018, pp. 1634–1638.
- [2] R. Rahmat, F. Wulandari, S. Faza, M. Muchtar, and I. Siregar, “The morphological classification of normal and abnormal red blood cell using self organizing map,” in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, vol. 308, 2018, p. 012015.
- [3] C. Jung, M. Abuhamad, D. Mohaisen, K. Han, and D. Nyang, “White blood cells (wbc) images classification using cnn-based networks,” 2021.
- [4] A. Girdhar, H. Kapur, and V. Kumar, “Classification of white blood cell using convolution neural network,” *Biomedical Signal Processing and Control*, vol. 71, p. 103156, 2022.
- [5] A. Acevedo, S. Alférez, A. Merino, L. Puigvi, and J. Rodellar, “Recognition of peripheral blood cell images using convolutional neural networks,” *Computer methods and programs in biomedicine*, vol. 180, p. 105020, 2019.
- [6] R. Kumar, S. Joshi, and A. Dwivedi, “Cnn-sspso: A hybrid and optimized cnn approach for peripheral blood cell image recognition and classification,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 35, no. 05, p. 2157004, 2021.
- [7] S. Sharma, S. Gupta, D. Gupta, *et al.*, “Deep learning model for the automatic classification of white blood cells,” *Computational Intelligence and Neuroscience*, vol. 2022, 2022.
- [8] A. Acevedo, A. Merino, S. Alférez, Á. Molina, L. Boldú, and J. Rodellar, “A dataset of microscopic peripheral blood cell images for development of automatic recognition systems,” *Data in brief*, vol. 30, 2020.
- [9] E. Gavas and K. Olpadkar, “Deep cnns for peripheral blood cell classification,” *arXiv preprint arXiv:2110.09508*, 2021.
- [10] C. Cheuque, M. Querales, R. León, R. Salas, and R. Torres, “An efficient multi-level convolutional neural network approach for white blood cells classification,” *Diagnostics*, vol. 12, no. 2, p. 248, 2022.
- [11] M. Delgado-Ortet, A. Molina, S. Alférez, J. Rodellar, and A. Merino, “A deep learning approach for segmentation of red blood cell images and malaria detection,” *Entropy*, vol. 22, no. 6, p. 657, 2020.

- [12] L. Boldú, A. Merino, A. Acevedo, A. Molina, and J. Rodellar, “A deep learning model (alnet) for the diagnosis of acute leukaemia lineage using peripheral blood cell images,” *Computer Methods and Programs in Biomedicine*, vol. 202, p. 105999, 2021.
- [13] K. Kim, J. Jeon, W. Choi, P. Kim, and Y.-S. Ho, “Automatic cell classification in human’s peripheral blood images based on morphological image processing,” in *Australian Joint Conference on Artificial Intelligence*, Springer, 2001, pp. 225–236.
- [14] X. Yao, K. Sun, X. Bu, C. Zhao, and Y. Jin, “Classification of white blood cells using weighted optimized deformable convolutional neural networks,” *Artificial Cells, Nanomedicine, and Biotechnology*, vol. 49, no. 1, pp. 147–155, 2021.
- [15] S. Joshi, R. Kumar, and A. Dwivedi, “Hybrid dsscs and convolutional neural network for peripheral blood cell recognition system,” *IET Image Processing*, vol. 14, no. 17, pp. 4450–4460, 2020.
- [16] M. R. Reena and P. Ameer, “Localization and recognition of leukocytes in peripheral blood: A deep learning approach,” *Computers in Biology and Medicine*, vol. 126, p. 104034, 2020.
- [17] A. Kihm, L. Kaestner, C. Wagner, and S. Quint, “Classification of red blood cell shapes in flow using outlier tolerant machine learning,” *PLoS computational biology*, vol. 14, no. 6, e1006278, 2018.
- [18] P. Tiwari, J. Qian, Q. Li, *et al.*, “Detection of subtype blood cells using deep learning,” *Cognitive Systems Research*, vol. 52, pp. 1036–1044, 2018.