An Efficient Deep Learning Approach to Classify White Blood Cells

by

Afif Ibna Kadir Khan Turja
18101407
Ahsan Habib
22241135
Kefaiat Lamia Ehsani
17201097
Zahin Shabab
20101165
Anika Nower Tamanna
23241044

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 and Engineering

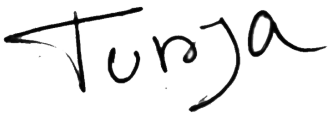
Department of Computer Science and Engineering
School of Data and Sciences
BRAC University
September 2023

Declaration

It is hereby declared that

1. The thesis submitted is 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.

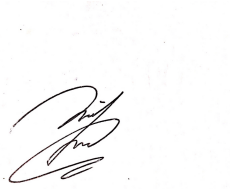
Student's Full Name & Signature:



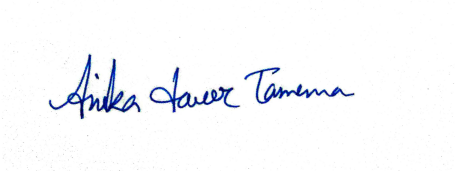
Afif Ibna Kadir khan Turja
18101407

Ahsan Habib
22241135

Kefaiat Lamia Ehsani
17201097



Zahin Shabab
20101165



Anika Nower Tamanna
23241044

Approval

The thesis/project titled "An Efficient Deep Learning Approach to Classify White Blood Cells" submitted by

1. Afif Ibna Kadir Khan Turja (18101407)
2. Ahsan Habib (22241135)
3. Kefaiat Lamia Ehsani (17201097)
4. Zahin Shabab (20101165)
5. Anika Nower Tamanna (23241044)

of SUMMER, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of of B.Sc. in Computer Science on September, 2023.

Examining Committee:

Supervisor:
(Member)

Shakila Zaman
Lecturer
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Dr. Md. Ashraful Alam, PhD
Assistant Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator :
(Member)

Dr. Md. Golam Rabiul Alam, PhD
Professor
Department of Computer Science and Engineering
Brac University

Departmental Head:
(Chair)

Dr. Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

The human immune system's white blood cells (WBCs) fight against infection and prevent the body from potentially harmful substances. They consist of neutrophils, eosinophils, basophils, monocytes, and lymphocytes, each of which comprises a various amount and has a specific task to do. Identifying white blood cells has been one of the most critical parts in medical science because it helps in diagnosing and monitoring various diseases and disorders. The manual microscopic analysis is difficult and subjective, and its time-consuming nature reduces the statistical dependability of the results. Problem with existing deep learning methods is that they can be heavy on computation. In this paper, we have proposed a very lightweight and efficient methodology which is called L100K-NetV2 with only 97,704 trainable parameters to classify white blood cells. The experiment, done with the Raabin-WBC (R-WBC) dataset, managed to achieve an accuracy of 98.11% in the TestA set. The proposed deep-learning methodology outperformed many other pre-trained deep learning models in terms of accuracy and parameter counts which helps to decrease the computational cost and training time.

Keywords: Deeplearning; Machine Learning; Convolutional Neural Network

Table of Contents

List of Figures

Chapter 1

Introduction

In the immune system of human body, white blood cells(WBCs) are critical components which act to defend the body against diseases, infections, and foreign substances. WBC classification is essential for diagnosing and treating a variety of medical conditions, affecting over 1 billion people worldwide each year [1]. Traditional manual classification techniques require significant time and are error-prone. It requires skilled technicians to identify and count the different types of WBCs under a microscope. Advancements of deep learning and computer vision techniques recently changed the sector by enabling the accurate and automated classification of white blood cells using image data. Deep learning models can be trained to learn the distinctive features of different types of WBCs from microscopic images. Once trained, these models can be used to classify WBCs with high accuracy and efficiency. However, these techniques can also be computationally expensive. With this thesis we aim to develop an lightweight and efficient deep learning approach for the classification of WBCs. Our primary goal is to develop computationally efficient deep learning model that can achieve accurate and consistent results for WBC classification. The proposed approach will use the advancements in state-of-the-art deep learning models, such as ResNet50, Inception v3 and DenseNet-121, to effectively extract and learn the distinctive features of white blood cells. We will start by analyzing the performance of different deep learning models on white blood cell classification. Specifically, ResNet50, InceptionV3, and DenseNet121 architectures will be investigated due to their proven capabilities in image classification tasks. To train and evaluate the deep learning models, a dataset of white blood cell images will be utilized, which in this case is the Raabin-WBC dataset [2]. The dataset consists of high-resolution microscopic images of different types of WBCs, which are categorized into the following classes: Neutrophils, Monocytes, Lymphocytes, Eosinophils, and Basophils. Appropriate preprocessing techniques, such as image augmentation, will be used to improve the diversity and generalization capabilities of the dataset. The performance of the developed deep learning approach will be assessed using standard evaluation metrics, including accuracy, recall, precision, and F1-score. Comparative analysis will be conducted to benchmark the performance of the proposed approach against existing manual classification methods and other machine learning algorithms, if applicable. Overall, by harnessing the power of deep learning, the contributions of this research will provide a significant step forward in the automated classification of WBC. This will enable faster and more reliable disease diagnoses and facilitate timely medical interventions.

1.1 Research Problem

White blood cell classification in medical diagnostics is of great importance due to its role in disease diagnosis, personalized treatment strategies, and early detection of diseases. Accurate and efficient classification methods help identify medical conditions, monitor disease progression, and serve as prognostic indicators. Additionally, they play a vital role in medical research, drug development, and point-of-care diagnostics. These methods enhance laboratory efficiency, provide valuable insights into disease mechanisms, and contribute to improved patient outcomes and healthcare practices.
Despite the advancements in white blood cell classification techniques, traditional methods often struggle to achieve high accuracy and efficiency due to the complexity and variability of white blood cell morphology. To overcome these limitations, there is a need for an efficient automated approach that can accurately classify white blood cells using deep learning techniques. However, the deep learning methods, those already exist, are heavy and require much more computational cost.
This research aims to develop an efficient deep learning approach which is lightweight, can accurately classify white blood cells into their respective subtypes, contributing to improved disease diagnosis and treatment. An efficient deep learning model accurately classifying white blood cells can revolutionize disease diagnosis and treatment, enabling faster and more accurate diagnoses, personalized treatment plans and valuable insights for medical research. This advancement can lead to improved patient care, enhanced healthcare accessibility, and better treatment outcomes.

1.2 Research Objective

With this research, we aim to develop an efficient deep-learning approach for the white blood cells (WBC) classification. The specific goals include:

- Investigating the performance of deep-learning models, such as ResNet50, InceptionV3 and DenseNet121, for WBC classification.
- Implementing and adapting the selected deep learning model (e.g., Inception v3) to the dataset containing white blood cell images.
- Implementing and adapting the suggested model L100KV2 in the dataset containing white blood cell images.
- Preprocessing the WBC images, including augmentation techniques, to enhance the diversity and robustness of the training data.
- Analyzing the interpretability and explainability of the deep learning model to gain insights into the features and characteristics used for white blood cell classification.
- Investigating the computational efficiency and resource requirements of the deep learning approach to ensure its practical applicability in real-time or resource-constrained scenarios.

- Conducting experiments and evaluations on a diverse and representative dataset of white blood cell images to validate the productivity and generalizability of the proposed methodology.

By addressing the research objective, this thesis aims to contribute to the development of an efficient, light-weight and accurate deep-learning approach for white blood cell classification, which can significantly improve the diagnostic process and aid in the timely detection and management of diseases.

Chapter 2

Related Works

In this research[3], automated medical image processing is used using various algorithms for preprocessing and segmentation to classify different categories of WBC. After noise reduction of the images, which could be associated with the light of a microscope or the non-compact blood contours, the images are ready for further processing. During preprocessing, noise is removed using a median filter, improving the quality of the images. Using thresholding, those images are separated from the background. This step is one of the most crucial steps in the entire procedure. The histogram entropy classification helps solve the difficulties faced in the previous step. After being done with classification, they are divided into five classes. After summarizing the classifications which are correct and incorrect using the matrix, the WBC were successfully classified with a success rate of 91%. It is advantageous because of its low computational complexity.
In this work [4], Convolutional Neural Network (CNN) is used for the classification and categorization of the WBC. The aim is to avoid pre-processing which could be complex at times. This CNN architecture implements a specific model for the classification known as WBCNet. To find better weights, the Random Gradient Descent Algorithm is used. The BN (Batch Normalization) algorithm is used for solving the issues with training the model. After testing different activation functions, WBCNet selected the PReLU activation function. The highest accuracy rate for WBCNet was reached at one point, top-1 rate of accuracy of 77.65%, and a top-5 rate of accuracy of 98.65%. During the training process, to prevent the over-fitting phenomenon, they added the "dropout" strategy.
In the paper [5], WBC from 3 distinct datasets were used to be classified which are categorized after detecting the nucleus and then the cytoplasm. Later on, the features are extracted and the WBC are classified. Nucleus segmentation occurs in 3 ways. Firstly, color balancing is done on an RGB image. Next, the HLS and CMYK color spaces are combined and computed, resulting in a soft map. Nucleus is segmented after implementation of an algorithm known as Otsu's thresholding. Finally, cytoplasm is segmented after obtaining the convex hull of the nucleus. RGB, LAB, HSV and YCrCb color spaces were the components used to extract the color features. During feature extraction, first the shape features (3 in number) are extracted and then the color features (48 in number) are extracted. After extracting the features, a properly trained SVM model gives the best results after data is fed to it.
The authors [6] of this study explained numerous strategies and procedures that

had been applied to accurately categorize white blood cells. The authors also gave a summary of pertinent research on deep learning and how it could be used for tasks in medical imaging, like classifying images of white blood cells. Additionally, they provided a thorough examination of how various topologies could be employed to increase classification accuracy for photos including leukocytes (white blood cells). The authors' methodology involved the deployment of a number of convolutional neural networks (CNNs) that had been trained using labeled datasets derived from microscopy pictures obtained from peripheral whole-blood samples of human subjects. They then compared their models to other conventional machine learning methods like kNN or SVM for comparison, evaluating them using common measures like accuracy and F1 score. In order to boost robustness against variance within training sets due to varying sizes and orientations between individual sample pictures taken by microscope cameras throughout capture sessions, they also tested out various network architectures, which included ResNet50 and VGG16 along with data augmentation techniques like random editing or flipping images horizontally or vertically before feeding them into their model architecture.The overall results demonstrated that these models were superior to traditional machine learning techniques when it came to accomplishing better performance regarding precise classification rates among leukocyte subtypes.

In this research [7], WBC are identified using a spectral and morphologic method which is based on hyperspectral blood images. The accuracy of previous methods were suboptimal which is why the new method helps to overcome this problem. This method focuses on the specific, distinct spectrum of each class. It combines the spectral and spatial features and optimizes both of their advantages. They used hyperspectral imaging system to collect the photos. The specific morphology of different white blood cells are used for special analysis to build appropriate features. This paper uses mathematical morphology which is a very reasonable alternative, useful for various image processing methods such as image filtering, segmentation, analyzing shape etc. Another data mining method, SVM, is used for training on spatial and spectral features combined which can deal with the regression problem along with the pattern identification problem.

In another work [8], the author of the paper proposed that they can classify WBC by gathering the information of neucluas. Only four features were extracted in this method from each nucleus of the blood cells. The key here was information based on morphology called pattern spectrum. Two classifiers are applied in this experiment, one is an artificial neural network classifier and the other is a Bayes classifier. Mathematical morphology is used here which is a branch of nonlinear image processing method used here. A single cell image of a nucleus would be manually segmented into cytoplasm, nucleus and background regions. After supervised learning, which is training and testing both Bayes classifier and artificial neural network, using the five fold cross validation method the experiments are performed. This research gets better results when using the Bayes classifier to classify the nucleus-based features in comparison with the traditional classification that is.

In this study [9], a technique known as transfer learning is used to identify different types of white blood cells. After modifying a CNN, it is used to classify various objects. Recent models have proven to be useful and more efficient on improving the classification of the images specifically on the tasks of segmentation and object detection. Using models which are already trained like the GoogLeNet, Resnet-

101 and the AlexNet highly accurate results are obtained because these models are considered to be robust and provide very accurate results apart from saving a lot of time. After making sure that the dimensions of the image to be fed to the model is of the same requirement of the proposed CNN.
This thesis [10] proposes a novel hybrid approach for efficient white blood cell subtype classification. The proposed approach consists of two main stages: (1) feature extraction and (2) feature selection. In the feature extraction stage, deep features are extracted from enhanced and segmented WBC images using transfer learning on pre-trained deep neural networks, DenseNet201 and Darknet53. In the feature selection stage, an entropy-controlled marine predator algorithm (ECMPA) is used to select the most dominant features while discarding the weak ones. The reduced feature vector is then classified with multiple baseline classifiers with various kernel settings.The proposed methodology is evaluated on a public dataset of 5000 synthetic WBC images corresponding to five different subtypes: neutrophils, eosinophils, basophils, lymphocytes, and monocytes. The experimental results demonstrate that the proposed approach achieves an overall average accuracy of 99.9% with more than 95% reduction in the size of the feature vector.
This thesis[11] proposes a novel multi-level convolutional neural network (CNN) approach for white blood cell (WBC) classification. The proposed approach consists of two stages: (1) WBC detection and separation, and (2) WBC subtype classification. In the first stage, a Faster R-CNN network is used to detect and separate WBCs into mononuclear and polymorphonuclear cells. In the second stage, two parallel CNNs with the MobileNet architecture are used to classify the subclasses. The proposed approach is evaluated on a public dataset of 2000 blood smear images with 40,000 labeled WBCs. The experimental results demonstrate that the proposed approach achieves an overall accuracy of 98.4% for WBC subtype classification.
This study [12] explores the adoption of deep learning (D.L) techniques to address the challenges associated with blood cell count and classification. Specifically, the study implements a deep learning model utilizing the DenseNet121 architecture for the classification of different white blood cell (WBC) types. By optimizing the DenseNet121 model with preprocessing techniques such as normalization and data augmentation, this approach seeks to enhance accuracy and efficiency. The dataset utilized for experimentation comprises 12,444 images. The study[12] further investigates the influence of various factors on model performance. Specifically, it explores the impact of different batch sizes in conjunction with the Adam optimizer and 10 epochs. The results demonstrate that the DenseNet121 model's performance excels with a batch size of 8 compared to other batch sizes.The results of this study indicate that the proposed Deep Learning model achieved notable performance metrics, including an accuracy rate of 98.84%, precision of 99.33%, sensitivity of 98.85%, and specificity of 99.61%.

Chapter 3

Methodology

3.1 Workplan

First, we have to select a dataset consisting of WBC images. In this case, our preferred dataset is 'Raabin-WBC'. We have to preprocess the data, from the dataset we are using, by splitting, resizing, normalising and implementing augmentation. Then, we will use the data and send it through our preferred 3 models, ResNet50, Inceptionv3 and DenseNet121. We will then analyze the results and compare the results between these 3 models. Then we suggested a very lightweight and efficient deep-learning L100KV2 model which is more efficient and therefore can classify and characterize the WBC more effectively.

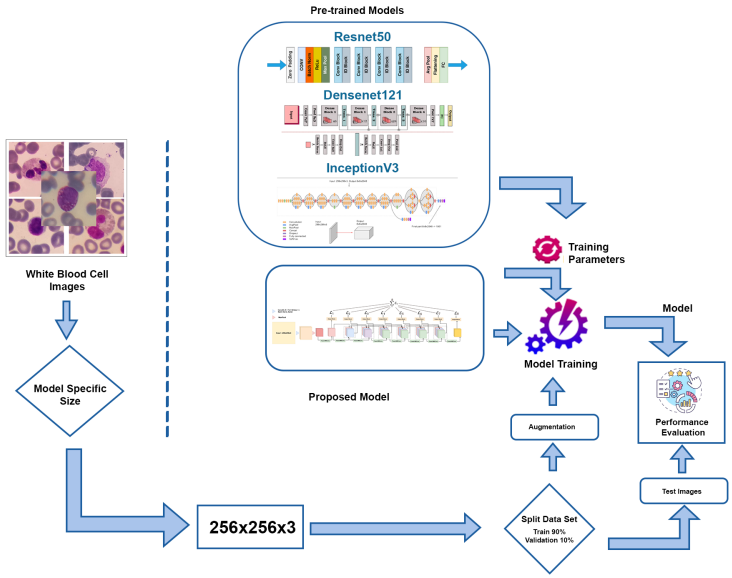


Figure 3.1: WorkFlow diagram

3.2 Dataset and Data analysis

3.2.1 Dataset

In this work, We have used a well-known benchmark data set: Raabin-WBC (R-WBC) [2], which is a comparatively newer proposed dataset for WBC subtypes classification. This dataset is contributed by many renowned laboratories. The train section of the dataset contains 10,175 images of white blood cells of 5 classes. The images were captured using 100x Olympus CX18 and 100x Zeiss microscopes, and Samsung Galaxy S5 and LG G3 camera phones. A collage of 2 sample images from each WBC sub-type taken from the RaabinWBC is shown in the Figure 3.2. We used the Train part of the dataset among the classes of Train,Test A and Test B.

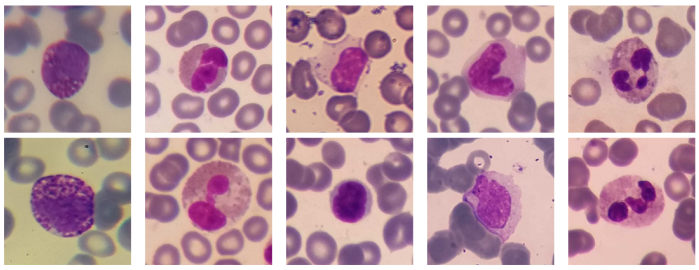


Figure 3.2: 2 images from each class of the dataset

3.2.2 Data Preprocessing

The train section of the dataset contains 10,175 images of white blood cells of 5 classes. We have split the dataset into 2 parts. Among the images we have used 90% for training, 10% for validation purposes. For testing, we used the TestA set. And the images were resized into 256*256 px. Then we normalized the data by converting the pixels in the range [0,255] to the range [0,1]. The loss will be allocated more equally if all of the images are scaled to the same range [0,1].To improve the model's robustness and ability to withstand variations we have used the 'ImageDataGenerator' class from Keras and defined the augmentation [13] settings. The images are randomly rotated between 0 and 40 degrees. We set the both vertical and horizontal flip to true. The fill mode fills any gaps in the image with the nearest pixel, effectively stretching it. We have set the height and width of the images to 224.

3.3 Model Architecture

3.3.1 ResNet50

Resnet 50 [14] stands for Residual Network which is a classic convolutional neural network consisting of 50 layers and it maintains two essential elements of design.

First, each layer has the same number of filters regardless of the output feature map's size. Secondly, it has twice as many filters in order to preserve the temporal complexity of every single layer even though the size of the feature map is cut by half. The 50-layer ResNet's building block has a bottleneck design. By using 11 convolutions, a "bottleneck" residual block reduces the number of parameters and matrix multiplications. This makes training each layer considerably faster. Instead of using two layers, it employs a stack of three layers. Resnet 50 allows for the addition of more convolutional layers in a convolutional neural network and it has demonstrated excellent performance on the ImageNet challenge. We have chosen Resnet50 [Figure : 3.3] as it reduces the training error with a deeper network through connection skip.

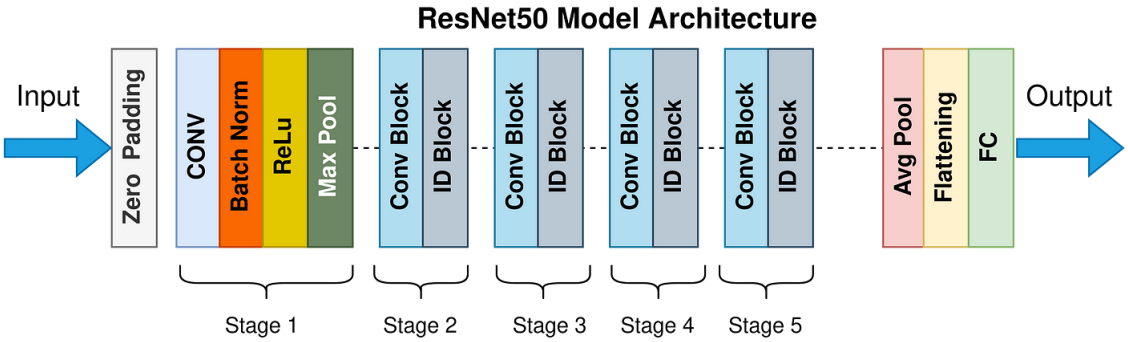


Figure 3.3: Architecture of ResNet50 [15]

3.3.2 InceptionV3

Inception V3 [16] is a deep convolutional neural network model consisting of 42 layers and it has a lower error rate than its preprocessor Inception V1 and V2. The model uses an auxiliary classifier as a regularizer and it's computationally less expensive. To help stabilize and accelerate the training process, Inception v3 uses a technique that normalizes the activations of each layer across mini-batches. Historically, average and maximum pooling have been employed to reduce the grid area of the feature maps. In the Inception V3 model, the activation aspect of the network's filters has been improved to decrease the grid size. Figure 3.4 shows the architecture of InceptionV3. This model was chosen because of its excellent performance and computational efficiency.

3.3.3 DenseNet121

Densenet121 [17] [Figure 3.5], a deep convolutional neural network architecture, has 4 average pools and 120 convolutional layers. The basic concept of DenseNet is the idea of dense connections. Because they require fewer parameters and allow feature reuse, DenseNets have achieved cutting-edge performance and greater efficiency across competitive datasets compared to their conventional CNN or ResNet competitors. We utilized DenseNet121 because it is impractical to apply the concatenation procedure when the size of feature maps varies.A key element of CNNs is downsampling of layers, which reduces the dimensionality of feature maps to boost computing speed.

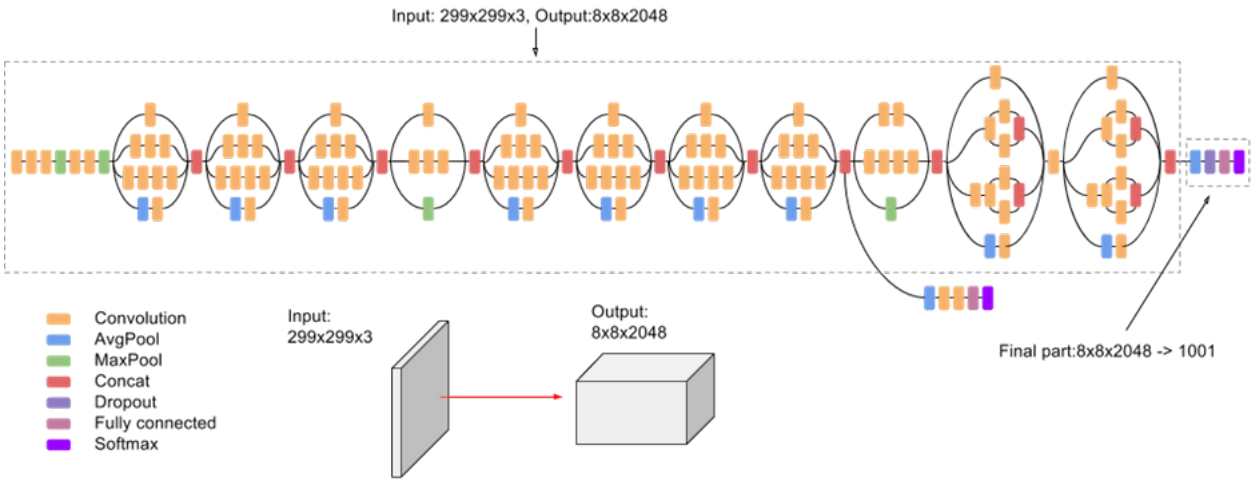


Figure 3.4: Structure of Inception V3

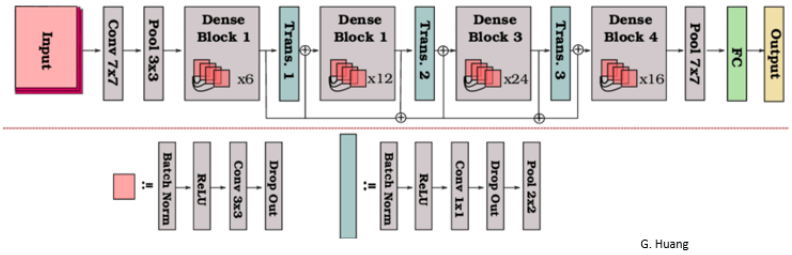


Figure 3.5: Structure of DenseNet121 [18]

3.3.4 L100KV2

As it integrates approaches suggested on various models, The proposed model architecture is inspired by Resnet [19] [20], EfficientNetV2 [21], MobileNetV2 [22] and DenseNet[23]. At first the CNN model performs 2D convolution operation of 7x7 kernel size and the strides set to 1 to keep the image in 256x256 px which helps to improve the performance of the model. Afterwards, Through a max-pooling2D operation with a pool size of 2, the feature maps produced by the Convolution2D operation are downscaled once. We used Fused-MBConv blocks in the proposed model which has shown promising results in EfficientNetV2. The authors pointed out that early use with the Fused-MBConv block [24] could reduce training times so we used Fused-MBConv blocks in the proposed model. Consequently, before delivering the feature maps to the output blocks, we implemented the Fused-MBConv blocks.The proposed model's Fused-MBConv blocks are comparable to those found in EfficientNetV2, which performs a convolution operation with a kernel size of 3x3 before passing the output through a Squeeze-and-Excitation [25] block or SE Block and performing another convolution operation with a kernel size of 1x1 to match the initial input's dimension so that the final output and initial input can be added together. According to the proposed model, the initial 3x3 Convolution2D operation on Fused-MBConv would generate a total of 6 feature maps, and the Squeeze-and Excitation layer would reduce all of the features to 50% of the initial features and perform ReLU activation, while the feature maps produced by Convolution2D operations would proceed through batch-normalization [26] and Swish activation [27] [28] function. The architecture of the Fused-MBConv blocks utilized in the suggested model is depicted in Figure 3.6 .

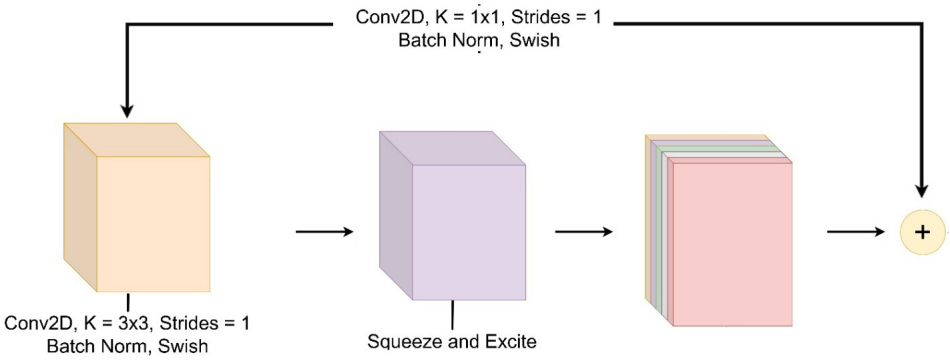


Figure 3.6: Fused-MBConv block used on the proposed model

As the model is highly inspired from MobileNetV2, we have used slightly modified Inverted Residual Block to reduce the total parameter counts. The figure 3.7 shows the inverted residual block used in the proposed model. We know that the Swish activation function is computationally more expensive but it provides a higher precision compared to ReLU6 [29] . Therefore, we have replaced the ReLU6 and linear activation function with the Swish activation function. The rest of the inverted residual blocks are almost the same as the one that is used in MobileNetV2. Firstly, a convolution 2D operation using a kernel size of 1x1 is performed on the initial

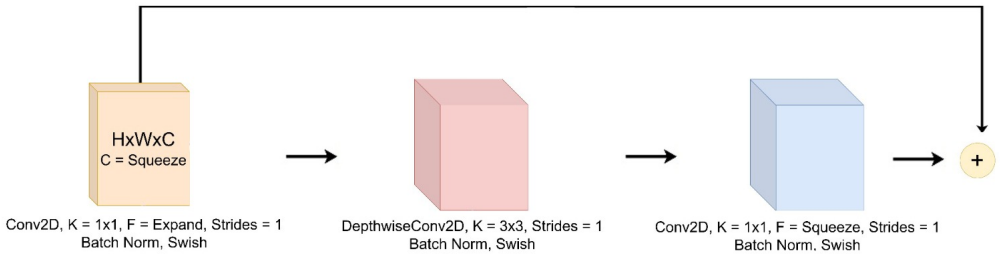


Figure 3.7: Inverted Residual block used on the proposed model

input to extend the dimension channel-wise on the resulting feature map, then a depth-wise 2D convolution operation would be done. The features then go through a convolution 2D operation utilizing a kernel size of 1x1 where the number of filters is set to the size of the initial input channel. In order to fit the features and add more, the number of channels was reduced or decreased. The Swish activation function was chosen because compared to ReLU [30], it enables smaller negative values to be processed and not be converted to zero which has been shown to increase deep learning model performance. Figure 3.8 shows the model architechture proposed in this work.

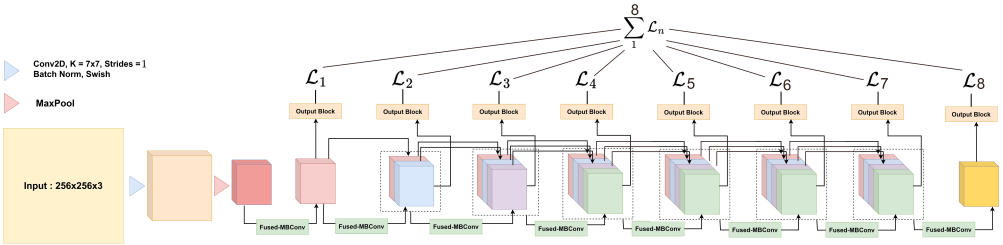


Figure 3.8: Proposed Model Architecture

A batch-normalization method is used just before forwarding the generated features from the convolution operation. The proposed model used a total of eight Fused-MBConv-blocks and the feature maps created by these blocks are concatenated to the following layers in an arrangement reminiscent of the DensNet architecture. The authors of the DensNet architecture explained in their work that this approach improves the training of deeper networks. This is particularly relevant for our proposed model, which has a total of 1172 convolutional layers. Additionally,the authors pointed out that this strategy can have a regularizing effect and a reduced overfitting.This is the motivation for using such a strategy. Moreover, the DensNet authors claimed that their approach enables implicit deep-supervision which is additional justification for applying deep-supervision in the suggested model. The output box would receive the feature maps created by the Fused-MBConv block.
Figure 3.9 illustrates the output block's layout. The output block is made out of layers of inverted residual block, several 1x1 kernel size convolution 2D operations and max-pooling 2D operations. In figure 3.9 the block receives the initial input feature maps and passes them through 1 inverted residual block where the features are extended to 8 channels and then squeezed again to 6 channels before being

delivered to the max-pooling 2D operations.

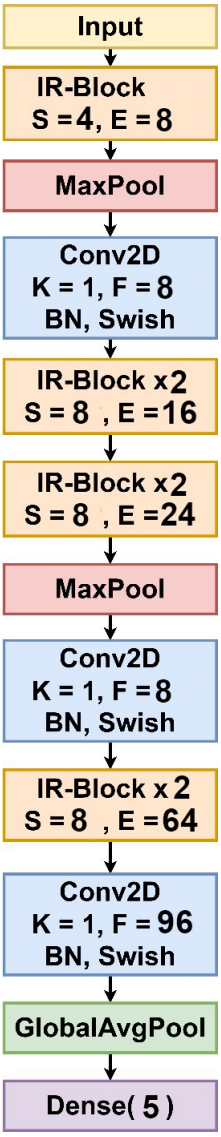


Figure 3.9: Structure of the Output Block

$$\sum_{1}^{8} \mathcal{L}_n = \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3 + \mathcal{L}_4 + \mathcal{L}_5 + \mathcal{L}_6 + \mathcal{L}_7 + \mathcal{L}_8 \tag{3.1}$$

Then a convolution2D operation with a kernel size of 1x1 and a number of filters set to eight would expand the dimension channel wise. Convolution2D operation generated feature maps will be subjected to batch-normalization and Swish activation function. 2 inverted residual blocks would be used while maintaining the same squeezing size. Similarly these squeezed channels would be 8 and expanded channels would be 16. Again, 2 Inverted Residual Blocks were added and squeezed channel would be 8 and expanded channels would be 24 before executing the max-pooling operation and another 1x1 convolution 2D operation without increasing the number of feature maps. Once more, 2 Inverted Residual Blocks would be used with this squeeze size remaining unchanged and expand size being increased to 64. The final steps included a GlobalAveragePooling2D operation, a Dense layer with a total of 5

layers and a softMax activation function for classifying the 5 different classes. The final softMax activation, which generates the probabilities for all five classes and is basically the output of the final prediction for the CNN Model, is the reason this block is known as the output block. Through the final softMax activation function, the 8 output blocks would use feature maps produced at various levels of the layers to produce 8 distinct predictions. Combined loss would be determined based on all of the predictions, as stated in equation 3.1, where $\mathcal{L}_n$ stands for the n loss calculated based on the n Output Block's prediction. The suggested CNN model becomes deeply-supervised by computing loss based on the characteristics of several levels. This enables the model to train based on both the hidden layers of the suggested model as well as the final output, which we would define as the one produced by the final Output Block that, if deep-supervision is not employed, takes the output from the final Fused-MBConv block as input. The authors of the Deeply-Supervised Nets [31], who proposed the methodology, inspired us to adopt this strategy. In their opinion, applying deep supervision reduces testing error due to its regularization effect and promotion of convergence, both of which would enhance the performance of the proposed model.

Chapter 4

Experiment and Results

We have done this experiment by using a Nvidia GTX1660 GPU with a memory of 6GB and compute compatibility of 7.0 Moreover, we used the Tensorflow library to enable Mixed Precision and only allocated 512MB of graphics processing unit memory to train the model.

4.0.1 Experiment with ResNet50

Initially, we experimented with the ResNet50 architecture. We set the batch size to 32 and trained the ResNet50 model for 50 epochs. We set the learning rate to 0.001 with Adam Optimizer[32]. Followed by the GlobalAveragePooling2D layer, we added a dense layer with 512 units which passed through a rectified linear unit. The dropout rate was then set to 0.5, a dense layer of 256 units was added, and a ReLu activation function was then applied. Another layer of 5 units was also added that

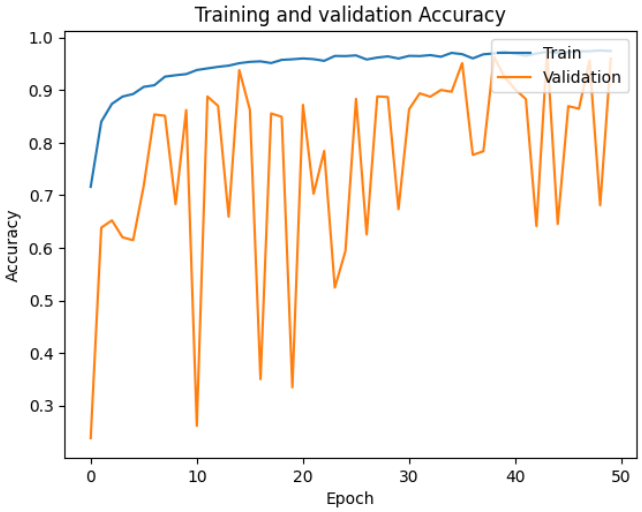


Figure 4.1: Training and validation accuracy of ResNet50

underwent a softmax activation function. For the loss function, we used the Sparse categorical cross Entropy. We set the input shape to (224,224,3). The results were gained from the Train part of the dataset. In the attempt, the validation accuracy reached 96.28% on the 39th epoch and the highest validation accuracy reached 97.42% on the 44th epoch and the training accuracy reached 97.43% on the 50th epoch. Figure 4.1 shows the training accuracy and validation accuracy achieved for

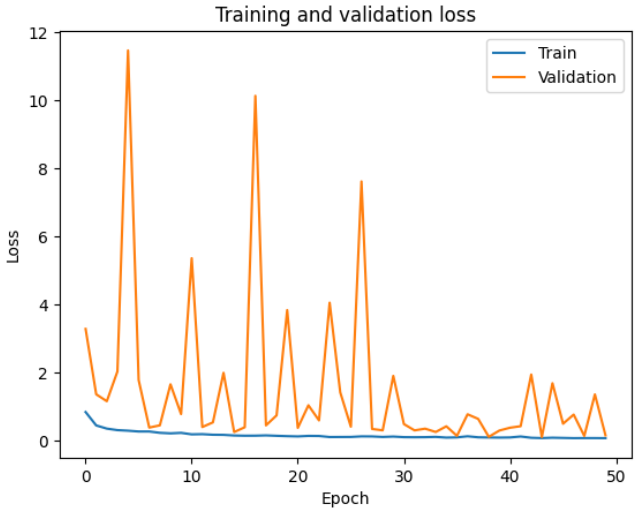


Figure 4.2: Training and validation loss of ResNet50

the ResNet50 model and Figure 4.2 shows the training loss and validation loss of the Resnet50 model. Table 4.1 shows the classification report of the Resnet50 model and it shows the Precision, Recall and F1 score. The ResNet50 model achieved 97.55% accuracy in the dataset. Figure 4.3 shows the confusion matrix which we used to explain the classification algorithm.

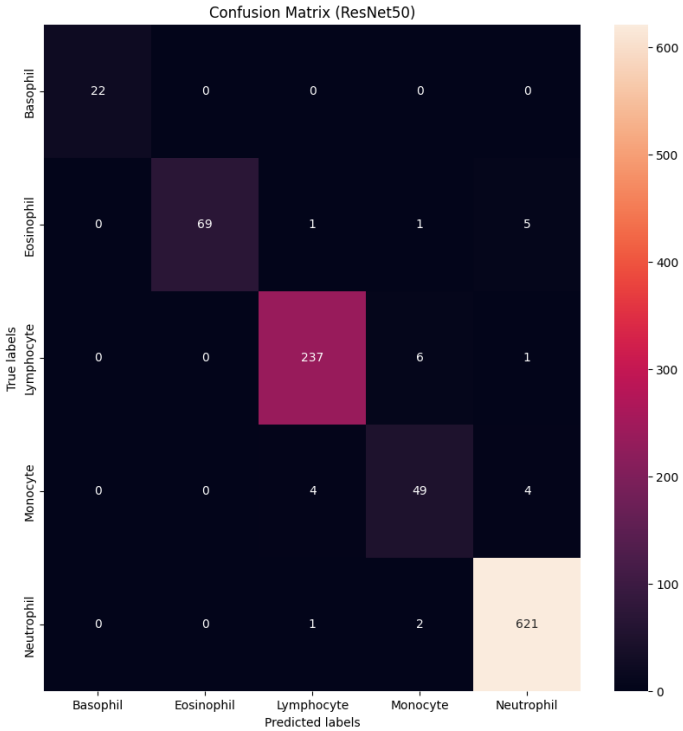


Figure 4.3: Confusion matrix of ResNet50

Class	Precision	Recall	F1
Basophil	1.00	1.00	1.00
Monocyte	0.84	0.86	0.85
Eosinophil	1.00	0.91	0.95
Lymphocyte	0.98	0.97	0.97
Neutrophil	0.98	0.1.00	0.99

Table 4.1: Classification report (ResNet50)

4.0.2 Experiment with InceptionV3

In the second phase, we experimented with the Inception V3 architecture. We kept the same batch size of 32 and trained the pre-trained model for 50 epochs as earlier. Again, we set the learning rate to 0.001 with Adam Optimizer. Followed by the GlobalAveragePooling2D layer, we added a dense layer with 512 units which passed through a rectified linear unit. Like in the Resnet, the dropout rate was then set

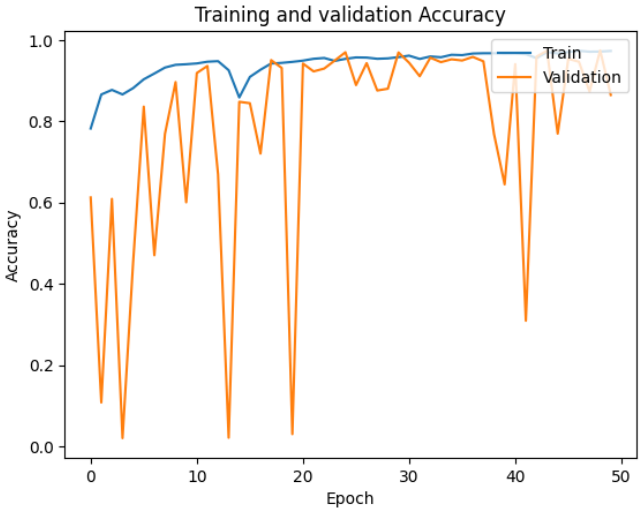


Figure 4.4: Training Accuracy and validation Accuracy of Inception V3

to 0.5, a dense layer of 256 units was added, and a ReLu activation function was then applied. We also added another layer of 5 units, which underwent a softmax activation function. For the loss function, we used the sparse categorical cross entropy. We have chosen the input shape as (224,224,3). The results were gained from the train part of the dataset. In this attempt, the validation accuracy reached 97.47% on the 49th

Class	Precision	Recall	F1
Basophil	0.92	1.00	0.96
Lymphocyte	0.96	0.99	0.98
Monocyte	0.89	0.82	0.85
Eosinophil	0.97	0.91	0.94
Neutrophil	0.99	0.99	0.99

Table 4.2: Classification report (Inception V3)

epoch and the training accuracy reached 97.42% on the 50th epoch. Figure 4.4 shows the training and validation accuracy achieved for the Inception V3 model and Figure 4.5 shows the training and validation loss of the Inception V3 model. Table 4.2 shows the classification report of the Inception V3 model and it shows the Precision, Recall and F1 score. The Inception V3 model achieved 97.26% accuracy in the dataset. Figure 4.6 shows the confusion matrix that we used to illustrate the classification algorithm.

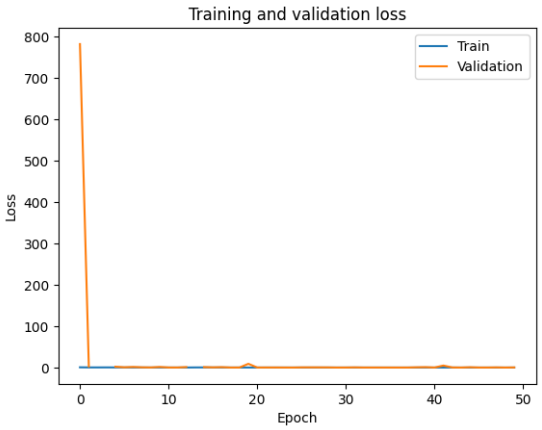


Figure 4.5: Train and validation loss of InceptionV3

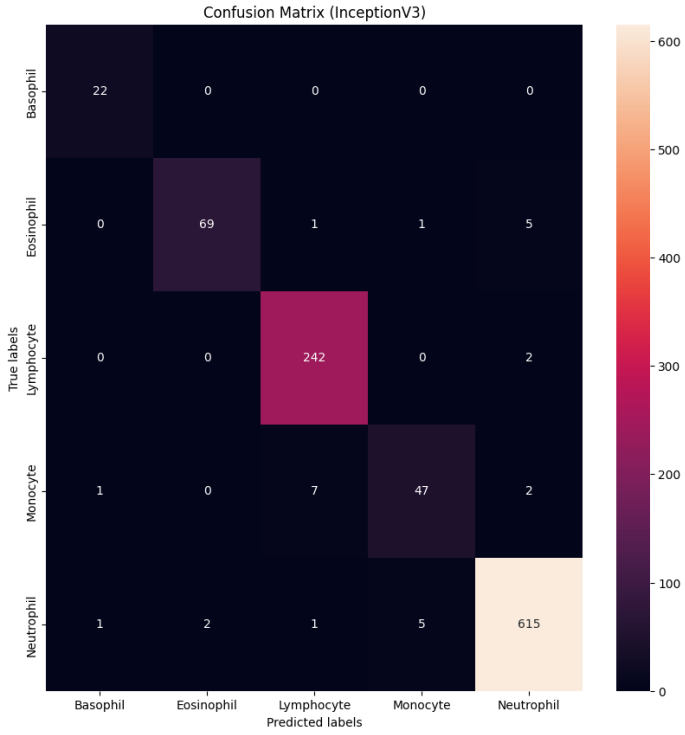


Figure 4.6: Confusion matrix of InceptionV3

4.0.3 Experiment with DenseNet-121

Lastly, we evaluated DenseNet121 using the same parameters and train-validation-test split. We used Adam Optimizer with a learning rate of 0.001 for the DenseNet121

model, which is the same value we used in the ResNet and InceptionV3 models. We created a dense layer with 512

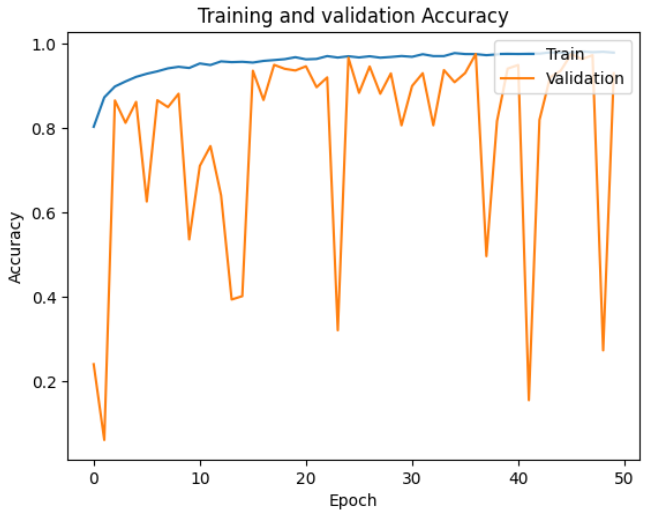


Figure 4.7: Training and validation accuracy of DenseNet-121

units that went through a Relu activation function after the GlobalAveragePooling2D layer. Then, a dense layer of 256 units was created, the dropout rate was adjusted to 0.5, and a ReLu activation function was used. Yet again, another layer with 5 units went through a softmax activation function. We employed the sparse categorical cross entropy for the loss function like the previous models.The model

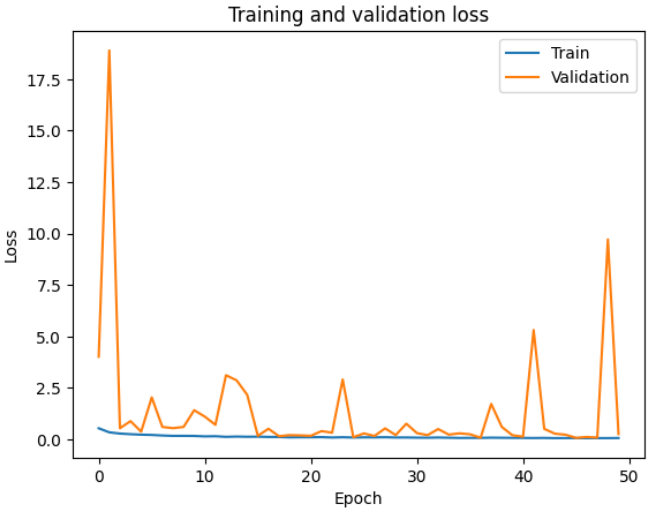


Figure 4.8: Training and validation loss of DenseNet-121

gained 97.57% validation accuracy on the 46th epoch and 98.14% training accuracy in the 47th epoch. Figure 4.7 shows the training and validation accuracy gained from the DenseNet121 model. And from Figure 4.7 we can see the training and validation loss from DenseNet121. Table 4.3 shows the classification report of DenseNet121 architecture. The model has achieved 97.36% accuracy. Figure 4.9 shows the confusion matrix generated from the DenseNet-121 model.

Class	Precision	Recall	F1
Basophil	1.00	1.00	1.00
Eosinophil	0.91	0.96	0.94
Monocyte	0.95	0.70	0.81
Lymphocyte	0.97	1.00	0.98
Neutrophil	0.98	0.99	0.99

Table 4.3: Classification report (DenseNet-121)

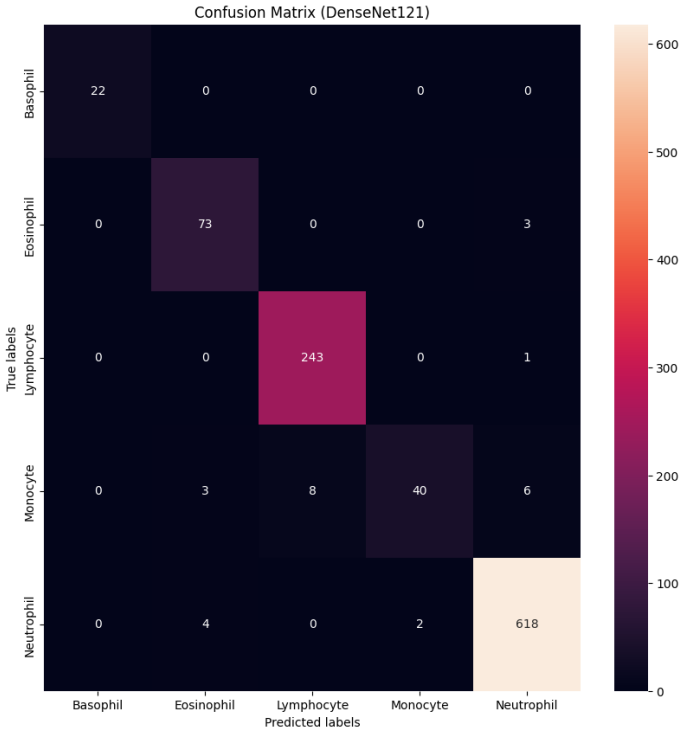


Figure 4.9: Confusion matrix of DenseNet-121

4.0.4 Experiment with L100kV2

We trained the proposed CNN model with Adam Optimizer [32] and used a learning rate scheduler [33]. Initially, we set up the learning rate to 0.01. To calculate the loss function, we used sparse categorical cross-entropy for multi-class classification. We trained the model for 100 epochs and set the batch size to 32. We set the learning rate to 0.01 initially and after 10 epochs set the decay to 0.001 for the next 40 epochs. After 50 epochs, the learning rate would decay to 0.0001 for the subsequent 50 epochs. Additionally, we use the ModelCheckpoint function from Keras to save

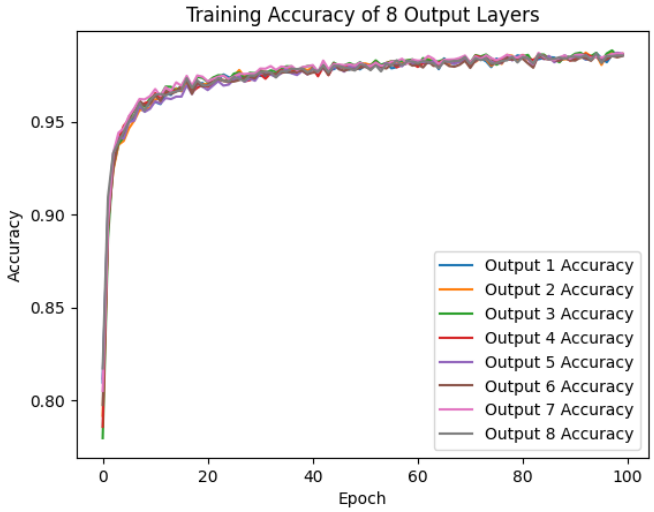


Figure 4.10: Training accuracy of 8 output layers

the model with the lowest validation loss, which includes the validation loss from eight separate output layers. For evaluation on the testing set, we would consider the model with the lowest validation loss. Figure 4.10 shows the training accuracy and figure 4.11 shows the training loss of the eight output layers. On the 98th epoch, the training accuracy reached 0.9884 in the 4th output layer. When the learning rate was set to 0.01 for the first 10 epochs there was more instability.

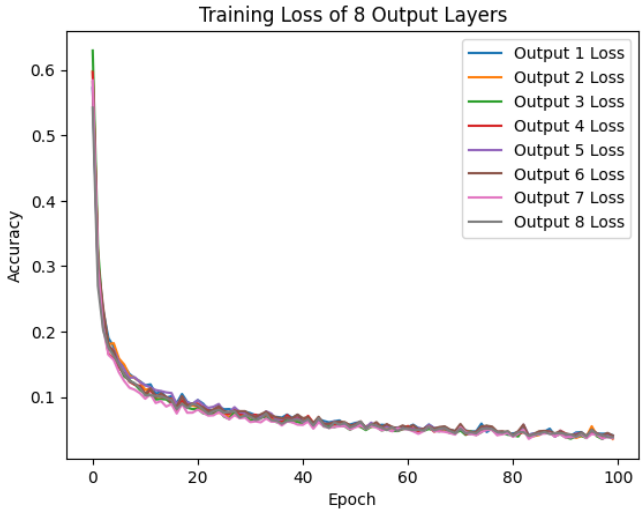


Figure 4.11: Training loss of 8 output layers

However, the graph became more stable when the learning rates were decrased to 0.001 or 0.0001. Despite the instability, training at a faster learning rate for the first 10 epochs took less time overall than training at a slower learning rate because the model converged more slowly at the slower learning rate. We also experimented with a low learning rate from the beginning with no decline. The lowest combined validation loss was reached on 100th epoch and the highest validation accuracy was reached 98.08% in the Output layer 6 in the 100th epoch.
Moreover, the figure 4.13 shows the validation accuracy and figure 4.14 shows the validation loss of the eight output layers. Additionally, we displayed the results of

averaging all of the probabilities for the classes from the output layers to generate a final value based on all of the outputs in Table 4.4 along with the weighted average of Precision, Recall, F1 scores, and Accuracy of all the output layers.

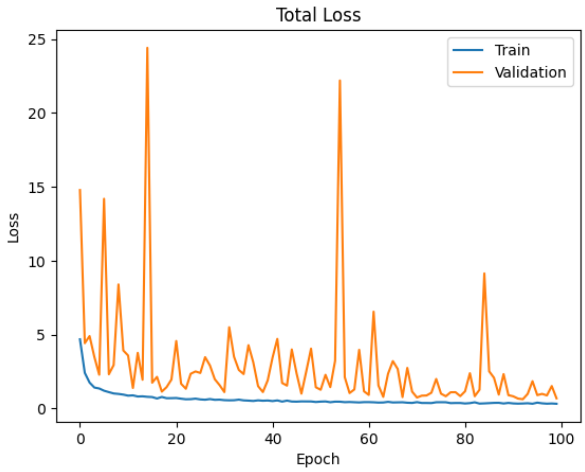


Figure 4.12: Total Loss

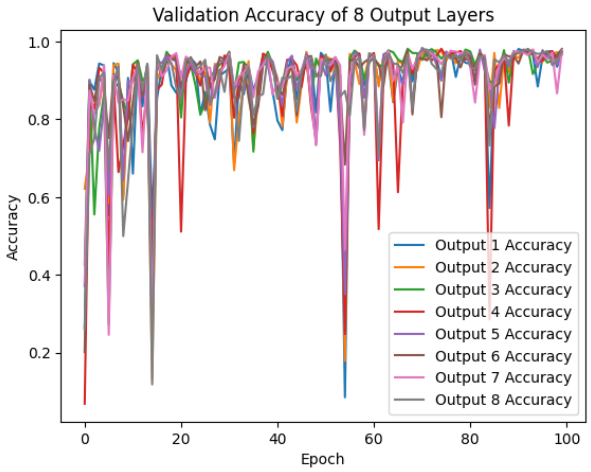


Figure 4.13: Validation accuracy of 8 output layers

A maximum accuracy of 98.11% could be attained by summing the probability from all of the Softmax output from eight separate layers.Figure 4.12 shows the total loss of the proposed model. Figure 4.15 shows the Confusion Matrix generated from L100KV2.

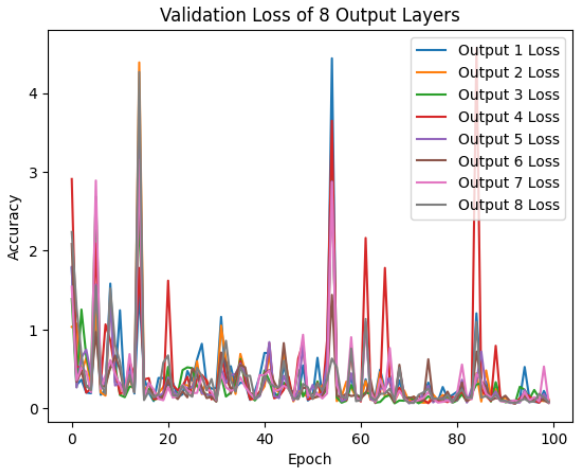


Figure 4.14: Validation loss of 8 output layers

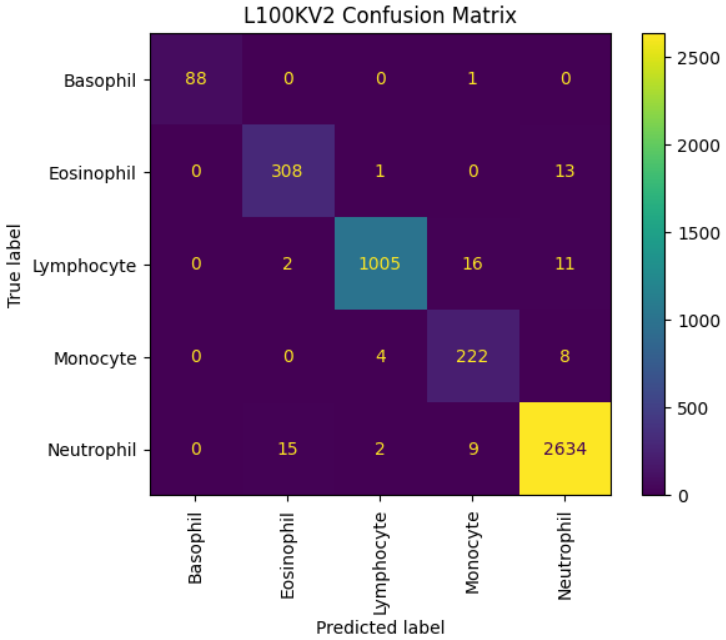


Figure 4.15: Confusion Matrix generated from L100KV2

Class	Precision	Recall	F1
Basophil	1.00	0.99	0.99
Eosinophil	0.95	0.96	0.95
Monocyte	0.89	0.95	0.92
Lymphocyte	0.99	0.97	0.98
Neutrophil	0.99	0.99	0.99
accuracy			0.98
macro avg	0.96	0.97	0.97
weighted avg	0.98	0.98	0.98

Table 4.4: Classification report (L100KV2)

4.1 Comparison of results

In this work, we have implemented three pretrained models: ResNet50, Inception V3, and DenseNet-121. We trained these models with the Raabin-WBC dataset. The ResNet-50 model achieved an accuracy of 98.04%. The Inception V3 and DenseNet-121 models achieved accuracies of 97.26% and 97.36%, respectively. Our proposed model, L100KV2, achieved an accuracy of 98.11% in the testA set.
By analyzing the figures we can see that The Resnet50 model achieved the highest accuracy among those models but in the InceptionV3 model the training and validation improved a lot compared to the ResNet50 and DenseNet121 model.

Method	RaabinWBC Testing Set	Trainable Parameters
InceptionV3	97.26	23.83M
DenseNet-121	97.36	10.4M
Support Vector Machine	94.65	0.015M
Resnet50	98.04	23M
MobileNetV2	98.48	2.2M
L100KV2 (Proposed Model)	98.11	0.097M

Table 4.5: Comparison of Results

We compared our work with various pre-trained deep learning models. Table 4.5 shows the results between various models and our model L100KV2. In most of the cases our model outpeformed the pre-trained CNN models with higher parameter counts since the study prioritizes a lightweight CNN with fewer parameters while focusing on the same dataset for the classification of white blood cells. Our model, which has a total of 107,880 parameters, is able to operate with less than 0.1 million parameters compared to other models that need millions of parameters to function. Of these parameters, 97,704 are trainable and 10,176 are non-trainable. Lastly, compared with different pre-trained CNN models, our models achieved a 0.07% improvement than most of the pre-trained model in accuracy while using only 97,704 parameters. This can significantly reduce memory usage and training time which helps in absolute classification of White blood cells.

Chapter 5

Conclusion

In this research, we developed and proposed a lightweight and efficient deep learning approach for the classification of white blood cells. The objective was to overcome the limitations of manual classification and leverage the power of deep learning to achieve accurate and consistent results with minimal computational cost. We looked into the deep learning approach's computational efficiency and resource usage. In order to improve the diversity and quality of the training data, we have included augmentation techniques. We have analyzed our proposed model with various other pre-trained deep learning models based on their F1 score, recall, and precision. We employed three existing models known as ResNet50, InceptionV3, and DenseNet121 throughout the process. We selected the ResNet50 model because it lowers training error with a deeper network by skipping connections. The ResNet50 model had an accuracy rate of 97.55%. InceptionV3, on the other hand, was chosen for its outstanding functionality and computational efficiency, achieving 97.26% accuracy. DenseNet121 achieved a precision of 97.36%. However, our proposed model outperformed these lightweight deep learning models in terms of both performance and computational efficiency. The ResNet50 model achieved the highest accuracy among the pre-trained models, but our proposed model outperformed the pre-trained models with an accuracy of 98.11% and reduced the computational cost due to its 97,704 trainable parameters. This demonstrates that our model is a promising solution for WBC classification due to its high accuracy and low computational cost. One limitation of our study is that we only evaluated our model on a single dataset. In the future, we plan to evaluate our model on larger and more diverse datasets to assess its generalizability. We also plan to develop an even more lightweight model with fewer parameters to reduce the computational cost while maintaining or improving accuracy. We believe that our work has the potential to make a significant impact on the field of WBC classification and improve the diagnosis and monitoring of various diseases and disorders.

Bibliography

[1] W. H. Organization, "Global health estimates 2021," in *Global Health Estimates 2021: WHO Statistical Information System*, 2022.

[2] Z. M. Kouzehkanan, S. Saghari, E. Tavakoli, *et al.*, "Raabin-wbc: A large free access dataset of white blood cells from normal peripheral blood," *bioRxiv*, pp. 2021–05, 2021.

[3] S. F. Bikhet, A. M. Darwish, H. A. Tolba, and S. I. Shaheen, "Segmentation and classification of white blood cells," in *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100)*, IEEE, vol. 4, 2000, pp. 2259–2261.

[4] R. R. J. Mayank Sharma Aishwarya Bhave, "White blood cell classification using convolutional neural network," *32*, 2018.

[5] S. Tavakoli, A. Ghaffari, Z. M. Kouzehkanan, and R. Hosseini, "New segmentation and feature extraction algorithm for classification of white blood cells in peripheral smear images," *Scientific Reports*, vol. 11, no. 1, p. 19 428, 2021.

[6] V. P. Pullanji and J. Muthuswamy, "White blood cell image classification using deep learning method," in *AIP Conference Proceedings*, AIP Publishing, vol. 2494, 2022.

[7] Q. Wang, L. Chang, M. Zhou, Q. Li, H. Liu, and F. Guo, "A spectral and morphologic method for white blood cell classification," *Optics & Laser Technology*, vol. 84, pp. 144–148, 2016.

[8] S. D. Nipon Theera-Umpon, "Morphological granulometric features of nucleus in automatic bone marrow white blood cell classification," vol. 11, pp. 353–359, 2007.

[9] M. J. Macawile, V. V. Quiñones, A. Ballado, J. D. Cruz, and M. V. Caya, "White blood cell classification and counting using convolutional neural network," in *2018 3rd International conference on control and robotics engineering (ICCRE)*, IEEE, 2018, pp. 259–263.

[10] R. Ahmad, M. Awais, N. Kausar, and T. Akram, "White blood cells classification using entropy-controlled deep features optimization," *Diagnostics*, vol. 13, no. 3, p. 352, 2023.

[11] 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.

[12] 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.

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

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[15] S. Mukherjee, *The annotated resnet-50*, Aug. 2022. [Online]. Available: https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758.

[16] 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*, 2016, pp. 2818–2826.

[17] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks. arxiv 2016," *arXiv preprint arXiv:1608.06993*, vol. 1608, 2018.

[18] N. Radwan, "Leveraging sparse and dense features for reliable state estimation in urban environments," Ph.D. dissertation, University of Freiburg, Freiburg im Breisgau, Germany, 2019.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[20] I. C. Duta, L. Liu, F. Zhu, and L. Shao, "Improved residual networks for image and video recognition," in *2020 25th International Conference on Pattern Recognition (ICPR)*, IEEE, 2021, pp. 9415–9422.

[21] M. Tan and Q. Le, "Efficientnetv2: Smaller models and faster training," in *International conference on machine learning*, PMLR, 2021, pp. 10 096–10 106.

[22] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[23] 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*, 2017, pp. 4700–4708.

[24] S. Gupta and B. Akin, "Accelerator-aware neural network design using automl," *arXiv preprint arXiv:2003.02838*, 2020.

[25] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[26] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pmlr, 2015, pp. 448–456.

[27] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[28] M. A. Mercioni and S. Holban, "P-swish: Activation function with learnable parameters based on swish activation function in deep learning," in *2020 International Symposium on Electronics and Telecommunications (ISETC)*, IEEE, 2020, pp. 1–4.

[29] H. Kim, J. Park, C. Lee, and J.-J. Kim, "Improving accuracy of binary neural networks using unbalanced activation distribution," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 7862–7871.

[30] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[31] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Artificial intelligence and statistics*, Pmlr, 2015, pp. 562–570.

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[33] H. Hazimeh and N. Ponomareva, "Mind the (optimality) gap: A gap-aware learning rate scheduler for adversarial nets," in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2023, pp. 3018–3033.