Savitribai Phule Pune University

A PROJECT REPORT

ON

"CLASSIFICATION OF WBC USING MACHINE LEARNING AND IMAGE PROCESSING"

Submitted by

VISHAL RAGHUNATH CHAUDHARI
KAUSHAL SACHIN LAHAMGE
NIKITA ARUN THORAT

BACHELOR OF ENGINEERING
ELECTRONICS AND TELECOMMUNICATION

SANDIP
FOUNDATION

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION
Sandip Foundation's, Sandip Institute of Engineering and Management, Nashik

2023-2024

A
PROJECT REPORT
ON

"CLASSIFICATION OF WBC USING MACHINE LEARNING AND IMAGE PROCESSING"

Submitted by

VISHAL RAGHUNATH CHAUDHARI
KAUSHAL SACHIN LAHAMGE
NIKITA ARUN THORAT

BACHELOR OF ENGINEERING
ELECTRONICS AND TELECOMMUNICATION

UNDER THE GUIDANCE OF
Dr. M. K. SANGOLE

SANDIP
FOUNDATION

Savitribai Phule Pune University

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION
Sandip Foundation's, Sandip Institute of Engineering and Management, Nashik

2022-2023

Sandip Foundation's
Sandip Institute of Engineering and Management, Nashik

SANDIP
FOUNDATION

CERTIFICATE

This is to certify that, the project the report entitled **"Classification of WBC Using Machine Learning and Image Processing"** is a bonafide work completed under my supervision and guidance in partial fulfillment for award of Bachelor of Engineering (Electronics and Telecommunication) Degree of Savitribai Phule Pune University.

Submitted by

VISHAL RAGHUNATH CHAUDHARI	**[B190833009]**
KAUHSAL SACHIN LAHAMGE	**[B190833020]**
NIKITA ARUN THORAT	**[B190833038]**

Place: Nashik
Date: / /

Internal Examiner **External Examiner**

Dr. M. K. Sangole	**Prof. Y. R. Risodkar**	**Dr. D. P. Patil**
Guide	**HOD**	**Principal**

ACKNOWLEDGEMENT

It gives us great pleasure while present this report on "Classification of WBC Using Machine Learning and Image Processing". The planning of the project work would have been possible with continued and dedicated effort and guidance of the large number of faculty and staff member of the institute. We acknowledge our gratitude to all of them. The acknowledgment however will be incomplete without specifics mentioned as follows:

We express our profound sense of gratitude to our guide Dr. M. K. Sangole, for her continued encouragement and guidance during the project period. We could like to extend our gratitude to our HoD Prof. Y. R. Risodkar, for his valuable encouragement.

Furthermore, we would like to acknowledge the support and encouragement received from Dr. D.P. Patil, the Principal of our institution. Their belief in our abilities and continuous support has been a great source of motivation.

Finally, we would like to thank all the other members of the department who have directly or indirectly contributed to this project. Their cooperation and assistance have been essential in the successful completion of this endeavor.

Once again, we express my heartfelt appreciation to all the individuals mentioned above for their invaluable contributions to this project.

Vishal Raghunath Chaudhari [B190833009]
Kaushal Sachin Lahamge [B190833031]
Nikita Arun Thorat [B190833038]

ABSTRACT

Machine learning (ML) and Image processing models have been employed to significantly improve analyses of medical imagery, with these approaches used to enhance accuracy of prediction and classification. The immune system depends on white blood cells (WBCs) to protect against infections caused by viruses, bacteria, and fungus. There are five subtypes of white blood cells: lymphocytes, monocytes, neutrophils, eosinophils, and basophils. Identifying and counting different types of white blood cells is crucial for identifying and treating a range of conditions, including infectious infections, autoimmune disorders, immunological deficiencies, and leukaemia. A rapid and accurate WBC categorization model is essential. This review presents an in-depth analysis of modern techniques applied within the domain of medical image analysis for white blood cell classification. There are various commercially available technologies for counting blood components, but they require expensive clinical equipment that is out of budget in most developing nations. This publication proposes an affordable classification approach for calculating differentiated white blood cells using digital image processing and machine learning techniques. White blood cells (WBCs) in the human immune system prevent infection and protect the body from foreign substances. They are made up of neutrophils, eosinophils, basophils, monocytes, and lymphocytes, with each constituting a different percentage and performing specific activities. Traditionally, the clinical laboratory method for quantifying different types ofwhite blood cells has been a vital component of a complete blood count (CBC) test, which aids in patient monitoring. The classification of white blood cells (WBCs) plays an essential role to identifying diseases and monitoring the immune system's health. Traditional techniques of WBC classification largely depend on manual observation, which is time-consuming and prone to inaccuracy. To address these issues, this study shows an automated system for identifying white blood cells that combines machine learning and image processing algorithms. The proposed system first preprocesses microscopic images of blood samples to enhance contrast and remove noise. Next, it extracts relevant features from the preprocessed images, including shape, texture, and intensity characteristics of individual WBCs. These features serve as input to the machine learning model, which is trained using a dataset of labeled white blood cell images. .

Keywords: —White blood cell classification, White blood cell subtypes, Convolutional neural networks (CNN), Classification, Feature extraction, Machine learning (ML), Feature Extraction, Image Analyses, Autoimmune Diseases.

Contents

LIST OF ABBREVIATION

ML	Machine Learning
WBC	White Blood Cell
CBC	Complete Blood Count
CNN	Convolutional Neural Networks
ROC	Receiver Operating Characteristics

List of Figures

Chapter 1

INTRODUCTION

White blood cells are a vital component of the immune system in humans. An essential component of the human immune system are white blood cells. Granulocytes and agranulocytes are two subtypes of WBCs. Monocytes (2–10%) and lymphocytes (20–45%) make up agranulocytes, while basophils (0–1%), eosinophils (1–5%), and neutrophils (50–70%) make up granulocytes. Figure 1 shows several occurrences of WBC images. There are three methods available for classifying white blood cells: manual inspection, automatic classification using a haematology analyzer, and machine learning. The manual examination approach is regarded as the most effective. Differentiation of WBC [9, 10]. This strategy, however, is ineffectual and depends on the understanding and proficiency of hematopathologists. A deficiency or excess of white blood cells (WBCs) can cause a number of illnesses [7, 8]. Correctly classifying the many types of white blood cells is essential. Leukocytes, or white blood cells, are the cells that make up blood components. The immune system's ability to protect the body against infectious diseases depends on these cells. White blood cells are colorless, have a nucleus, can diapede—pass through capillary walls—and can carry amoebas. A normal adult human blood drop has less than one liter and between 4x109 and 11x109 cells, or approximately 7000–25,000 cells per drop. Classifying white blood cells is essential for the early diagnosis and management of several illnesses. WBC classification is useful in the diagnosis and treatment of a broad variety of medical conditions. Leukocytes, or white blood cells, are produced by the lymphoid organs and bone marrow of the immune system. The body uses these cells to fight against infections from bacteria, viruses, and fungi. In this work, we combine image processing and machine learning models to provide a novel approach for categorizing white blood cells. Our process entails several crucial steps:
i. Image Acquisition:Digital images of blood smears are obtained by means of defined protocols and microscopy apparatus.
ii. Preprocessing: Various processing techniques, such as noise reduction, contrast enhancement, and segmentation, are used to captured pictures to improve the quality and clarity of cell borders.
iii. Feature extraction: To describe the form, size, and internal structure of each indi-

vidual white blood cell, pertinent properties are extracted. These properties include morphological, textural, and statistical aspects.
iv. Machine Learning Classification: : Using the traits that were collected, white blood cells are divided into neutrophils, lymphocytes, monocytes, eosinophils, and basophils. Support vector machines (SVM), convolutional neural networks (CNN), and random forest classifiers are used in the training of these techniques.
v. Model Evaluation: Receiver operating characteristic (ROC) curves, accuracy, sensitivity, and specificity are among the metrics used to assess how well the trained modelsclassify white blood cells.
vi. Clinical Validation: We validate the proposed automated categorization system by comparing its results to manual microscopic inspection by skilled hematologists using clinical data from patients with known hematological diseases.

1.1 Importance

White blood cells (WBCs) are tiny soldiers in our body, fighting off germs and keeping us healthy. But sometimes, they act strangely when we're sick, and doctors need to see if something's wrong by looking at our blood under a microscope. It's like finding a needle in a haystack, and it takes a lot of time.But what if we could teach a computer to do this job? That's where machine learning and image processing come in. We can train a computer to spot different types of white blood cells in blood samples much faster and more accurately than humans. This project is super important because it means faster diagnosis. When the computer helps the doctor, they can figure out what's wrong with us more quickly, and that means we can start getting better sooner. It's like having a super-fast detective on the case! But that's not all. This technology can also help in places where there aren't many doctors. By using computers to analyze blood samples, even remote areas can get top-notch healthcare.Plus, it's a big help for medical students. They can learn from the computer's analysis and become even better doctors in the future.In short, this project is all about making healthcare faster, better, and more accessible for everyone. It's like having a superhero team of doctors and computers working together to keep us all healthy.

Chapter 2

LITERATURE SURVEY

Altaf Khan [1]. The objective of this research was to achieve a high performance of WBC-type identification by using a single deep CNN model, which can be used for the diagnosis of several diseases. To this end, based on a pretrained AlexNet, we design a novel model by combining multi-layer convolutional features from different layers. Multiple deep layers, which contain a variety of visual features were employed to improve WBC-classification accuracy, generating high dimensional feature vectors when these features are combined.

Oumaima Saidani [2]. The goal of this work is to classify WBC pictures using different DL and ML models. A comparative analysis was conducted to evaluate the performance and computational complexity of various techniques. This section presents the individual results of all models on the provided datasets under a variety of scenarios.

Changhun Jung [3]. Recent research in this field has concentrated on strategies for increasing the robustness and interpretability of CNN-based models for WBC analysis. This covers methods for domain adaptation to account for differences in imaging conditions and patient demographics, as well as strategies for visualising and explaining the features acquired by the models. Image classification and generative models based on CNNs for WBC analysis are active research areas with promising medical and therapeutic applications.

Mu-chun su [4]. This research presents a new algorithm for segmenting white cells from smear images. This segmentation technique identifies a distinct zone of white blood cell tones in the HSI colour space. The found The discriminating region can be defined as a 3D ellipsoid. We recommended using 20 features: 3 geometrical, 3colour, and 14 LDP. We used three neural-network-based classifiers to categorise white blood cells into five kinds. Our suggested system with a trained MLP outperformed other systems shown in Table. The performance was measured using a public database.

AI-Dulaimia [5]. This Chapter covers picture acquisition, pre-processing, segmentation, and feature extraction. This review examines various automated strategies for extracting and classifying white blood cells from 2005 to the present. Despite much research, segmenting, extracting features, and classifying white blood cells remains problematic, especially with non-uniform lighting and low resolution [6]

Abdullah Elen [6]. This work used microscopic blood pictures to extract statistical and geometrical properties, resulting in a feature vector with 35 parameters. This feature vector serves as an input parameter for six machine learning algorithms that classify white blood cells. To test the performance of the algorithms, five different types of data were prepared in alternative training and test ratios, as well as 100 alternative combinations of each data set, were used to analyse statistical results. The MLR algorithm achieves the highest success rate for leukocyte cell categorization across all datasets and settings. The k-NN algorithm has the lowest success rate and produces results that are close.

Satria Wibawa [7]. This study seeks to categorise two types of white blood cells: lymphocytes and neutrophils. The proposed categorization approach in this study was Deep Learning. Deep Learning performance was compared to three traditional machine learning methods: Multilayer Perceptron, k-Nearest Neighbour, and Support Vector Machine. Conventional machine learning refers to methods that cannot learn from raw data. Conventional machine learning utilised nine texture features. Deep Learning obtained a classification accuracy of 0.995, outperforming all three standard machine learning approaches. The Deep Learning model demonstrated robustness, as seen by similar training and validation accuracy rates. Future works should include more white blood cell classifications.

Siraj Khan [8]. This study reviewed TML and DL approaches for classifying leukocytes in blood smear images. We evaluated various TML and DL techniques to Classify WBCs in blood smear photos. Data were extracted from primary papers published between 2014 and 2020. The study found 80 primary articles (published in journals, books, conferences, and online materials) on TML and DL approaches for leucocyte categorization in blood smear images and their applications in medical diagnosis.

Da Wang [9]. This study use a deep learning model to recognise and count white

blood cells. The experimental results demonstrate that the proposed system is equivalent to cutting-edge systems. The proposed methodology employs an updated Faster-RCNN model. The dataset's white blood cells were accurately identified at 74.4%, 85.3%, 86.3%, 98.8%, 83.1%, and 75.5% IoU levels using image-level data. The proposed approach is applicable to WBC datasets and other cell types. .

Irwan Rahadi [10]. This work presents a system for automatically detecting and counting normal and hypochromic red blood cells using a Haar Cascade classifier trained by an algorithm. The findings showed that the algorithm performs well as expected. Adjusting the parameter on allows for the best detection. The algorithm. Our proposed method is highly successful for cell detection. We simply need red blood cell images to train the classifier. The results will be drawn directly. It usually works flawlessly. Used in RBC detection centres or health centres to detect RBCs. Further research will be conducted.

Hua chen [11]. This work presents a revolutionary deep learning strategy for properly identifying white blood cells. Our suggested method improves feature representation by combining ResNet and DenseNet. Furthermore, the The strategy utilises the SCAM mechanism to enhance the model's ability to represent relevant information in WBC images in two distinct dimensions of space and channel, addressing sample similarity. Different approaches for arranging spatial and channel attention will produce different classification results due to their distinct purposes. We use data augmentation and TL approaches to address the negative impact of imbalanced or insufficient training data on the performance of deep learning models

WENNA WU [12]. This work presents a revolutionary deep learning strategy for properly identifying white blood cells. Our suggested method improves feature representation by combining ResNet and DenseNet. Furthermore, the The strategy utilises the SCAM mechanism to enhance the model's ability to represent relevant information in WBC images in two distinct dimensions of space and channel, addressing sample similarity. Different approaches for arranging spatial and channel attention will produce different classification results due to their distinct purposes. We use data augmentation and TL approaches to address the negative impact of imbalanced or insufficient training data on the performance of deep learning models .

Neerukattu Indrani [13]. This study uses Convolutional Neural Networks to clas-

sify white blood cells into subgroups using microscopic images. Network methods. This classification distinguishes cells and determines the sort of disease a patient has. This experiment provides more reliable image identification than traditional lab procedures. The test set has a high degree of accuracy (over 90%). Training a model with high computational abilities can result in a perfect model for medical analysis and applications involving the number and subtypes of white blood cells.

Anwar Siswanto [14]. The classification technique test was successfully completed. The MLP approach was shown to have the highest accuracy for identifying white blood cells. Research shows that the NBC approach has an accuracy of 80%, whereas the KNN method has an accuracy of 82%. The MLP approach has a 92% accuracy and the longest time of 2.08 seconds. MLP was identified as the most effective classification model for identifying white blood cells, despite its higher processing time. High accuracy can be attributed to proper stain absorption and cropping of nonoverlapping white blood cells.

Cesar Cheuque [15]. This paper presents a hybrid multilevel approach for automatically detecting and classifying white blood cells into mononuclear (lymphocytes and monocytes) and polymorphonuclear (segmented neutrophils and eosinophils) types from blood smear images. At the first level, a Faster R-CNN network is applied to identify the region of interest of white blood cells, together with the separation of mononuclear cells from polymorphonuclear cells. Once separated, two parallel convolutional neural networks with the MobileNet structure are used to recognize the subclasses. The model achieved good performance metrics, achieving an average accuracy, precision, recall, and F-score of 98.4%, indicating that the proposal represents an excellent tool for clinical and diagnostic laboratories. Moreover, the proposed ML-CNN approach allows obtaining better accuracy results while optimizing the cost of computational resources, thus allowing the creation, evaluation, and retraining of each neural algorithm in an isolated way, without affecting those that achieve the expected levels of performance.

Mariam Nassar [16]. The study ”Label-Free Identification of White Blood Cells Using Machine Learning” is likely about establishing a method for identifying white blood cells (WBCs) without the need of labelling agents like fluorescent dyes. This approach could be useful in medical diagnostics and research since it simplifies and potentially speeds up the cell identification process while lowering expenses and de-

creasing sample preparation processes. Machine learning approaches are most likely used to analyse attributes of unlabeled white blood cells, presumably using parameters such as size, shape, granularity, and optical qualities obtained by imaging or other sensing methods. By training machine learning models on a dataset of known WBC types, the system learns to distinguish between cell types without the requirement for explicit labelling.

Blumers AL [17]. "GPU-accelerated Red Blood Cell Simulations with Transport Dissipative Particle Dynamics" is most likely a scientific study that examines the behaviour of red blood cells (RBCs) using a computational method known as Transport Dissipative Particle Dynamics (TDPD). The term GPU acceleration implies that the simulations use the processing capability of Graphics Processing Units (GPUs) to speed up the calculations.

Muhammad Yildirim [18]. This study examines categorization using deep learning, deep learning layers, and deep learning models, all of which have gained popularity in recent years. First and foremost, Eosinophils Leukocyte cells (lymphocytes, monocytes, and neutrophils) were classified in Matlab using Alexnet, Resnet50, Densenet201, and Google Net designs. Gaussian and median filters were applied separately to images in the database. The new photos were reclassified using the same architectures. Applying Gaussian and median filters to photos yielded better results than the raw data. Skilled individuals can make more accurate illness judgements after classifying blood cells.

Rabia Asghar [19]. This article gives a complete overview of TML and DL approaches for WBC classification. We investigated and compared different methods for WBC. classification in this situation. This analysis uses data from 136 source papers published from 2006 to 2023. These studies explore TML and DL methods for leukocyte classification and their use in medical diagnosis. These studies show that TML and DL approaches make important contributions to MIA. This work aims to uncover and synthesise TML and DL applications in MIA, namely for leucocyte categorization in blood smear pictures. This research intends to provide light on the complex properties of TML and DL in MIA.

Meiyu Li [20]. The researchers would normally acquire a big collection of mi-

croscopic photographs of leukocytes in various situations. To effectively train the model, this dataset must include a wide range of interference factors. The study will present the architecture of the deep learning model used to detect leukocytes. This could include convolutional neural networks (CNNs), which are ideal for image processing jobs due to their ability to learn features from raw pixel data.

Chapter 3

METHODOLOGY

Emotion recognition systems based on digitized speech consist of three basic components: signal pre-processing, feature extraction, and classification. Acoustic preprocessing such as denoising as well as segmentation is done to determine the meaningful units of the signal Feature extraction is used to identify the relevant features available in the signal. Finally, the mapping of extracted feature vectors to relevant emotions is performed by classifiers. This section provides a detailed description of speech signal processing, feature extraction, and classification. Differences between spontaneous and acted speech are also discussed in terms of their relevance to the topic. Block Diagram shows a simplified system used for speech-based emotion recognition. In the first stage of speech-based signal processing, speech enhancement is performed, where interfering components are removed. The second phase includes two parts, feature extraction, and feature selection. The desired features are extracted from the pre-processed speech signal and the selection is made from the extracted features. Such feature extraction and selection is usually based on time and frequency domain analysis of speech signals. During the third stage, different classifiers like GMM and HMM, etc. are used to classify these features. Finally, different emotions are recognized based on the feature classification.

3.0.1 BLOCK DIAGRAM

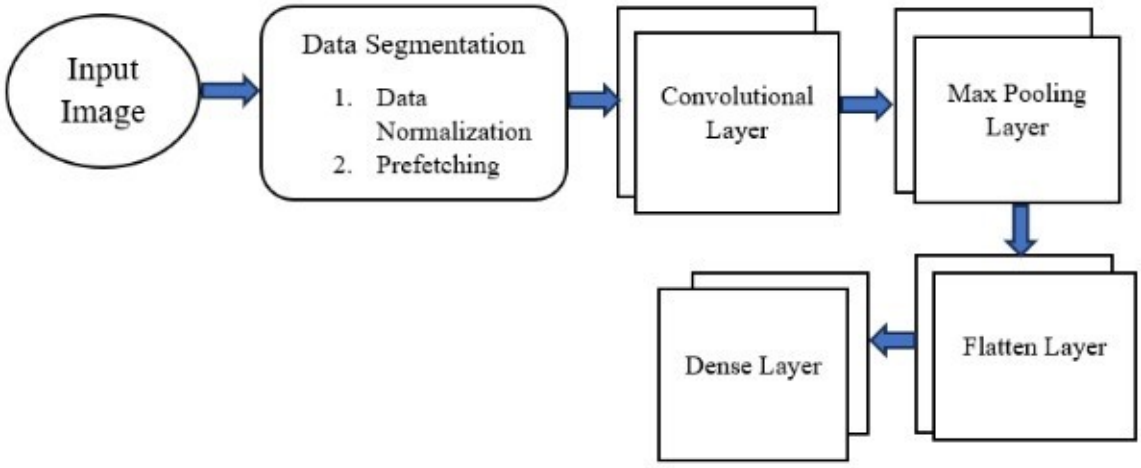

Figure 3.1: Design Flow for CNN

Data preprocessing: It is an essential step to prepare the dataset for training. In this project, images are loaded using OpenCV and resized to a standardized size of 128x128 pixels. Pixel intensities are then normalized to the range [0, 1], ensuring consistency and numerical stability during training. Labels are binarized using Label Binarize to convert class labels into binary vectors, enabling categorical classification. This preprocessing pipeline ensures that the input data is properly formatted and scaled, ready to be fed into the model for training.

Data Normalization: Data normalization is a preprocessing technique used in machine learning to rescale the values of numerical features to a standard range without distorting differences in the ranges of values.Here we used the TensorFlow's Rescaling layer for data normalization. This layer scales input values to be in a specified range, often between 0 and 1.Rescaling layer that will scale input values by dividing them by 255, effectively bringing them into the [0,1].Lambda function takes each sample in the training dataset, testing dataset and validation dataset and applies normalization layer. After this normalization step data will be ready to fed into a machine learning model.

Prefetching: Prefetching is a technique used to overlap the data loading process with model training or inference. It's particularly useful when dealing with large datasets that may take significant time to load from storage into memory. newline

Convolutional Layer: A convolutional layer is a fundamental building block in convolutional neural networks (CNNs), commonly used in image recognition, computer vision, and other tasks involving grid-like data such as time-series analysis and natural language processing.

Chapter 4

IMPLEMENTATIONS

4.1 ALGORITHM

4.1.1 Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) architecture is a popular choice for projects involving the classification of white blood cells (WBCs) utilizing machine learning and image processing due to its effectiveness in dealing with microscopic image data. Here's a simplified outline of a common CNN architecture for this task: .

conv2d(Conv2D): In a convolutional neural network (CNN), a Conv2D layer is a twodimensional convolution layer that applies a convolution kernel to the layer input in a single spatial or temporal dimension. The layer generates tensors of outputs. The most popular type of convolution is the 2D convolution layer, which is often abbreviated as conv2D. A filter or kernel in a conv2D layer "slides" over the 2D input data to execute element wise multiplication. As a result, it will combine the results into a single output pixel.

Max pooling2d (Max-Pooling 2D): The MaxPooling2D layer in a Convolutional Neural Network (CNN) is a pooling layer that down samples the input feature maps. MaxPooling2D is a type of spatial down sampling that aids in attaining translation invariance and lowering computational complexity in CNNs. MaxPooling2D keeps the most critical features by picking the maximum value inside each pooling window, while deleting less relevant information.

Conv2D_1: It is the first convolutional layer of a CNN. It extracts features from the input image. The layer is made up of several filters, each of which is a tiny weight matrix. The filters are applied to the input image, and each filter generates a feature map. The feature maps are subsequently sent to the next layer of the CNN.

Max pooling2d_2: The Conv2D layer performs a convolution operation on each position of the input feature map by sliding a small window known as a filter or kernel

across it. At each place, the filter executes a dot product on the associated input region, resulting in a single value in the output feature map.

• Flatten Layer: The Flatten layer in a CNN for WBC classification converts spatially organised feature maps into a format suitable for classification with fully linked layers. It allows the network to use the hierarchical representations of features acquired from the input images to produce. accurate predictions about the class labels of WBCs.

• Dense layer: In WBC classification models, the Dense layer functions as a potent classifier, learning complicated patterns and correlations between retrieved data and their corresponding white blood cell class labels. It is critical for accurately predicting the types of WBCs present in the input photos.

• Dense_1: The Dense layer is the basic component of WBC classification models that use machine learning and image processing. It is responsible for aggregating information retrieved from input photos and making final categorization judgements based on these features. It is crucial in translating the retrieved information into predictions about WBC class labels.

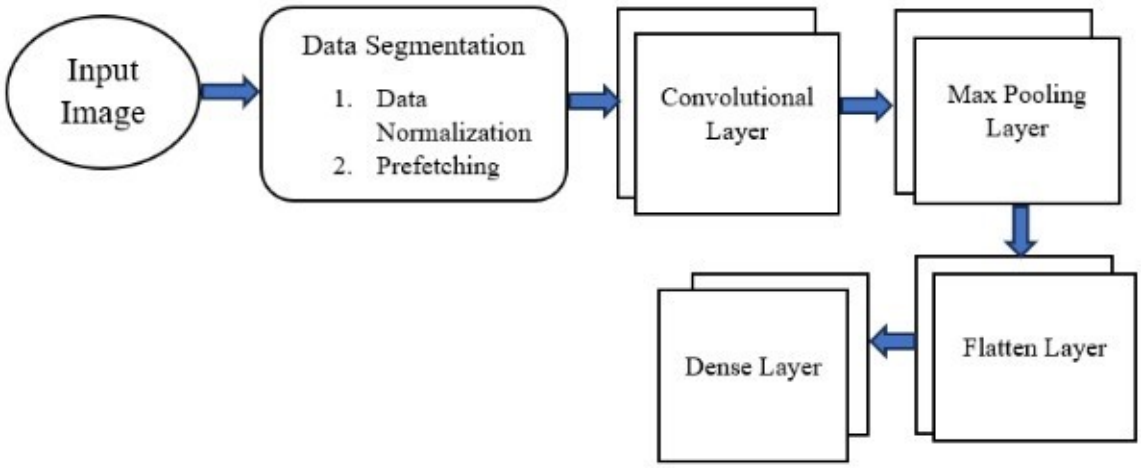

Figure 4.1: Design Flow for CNN

 # Print a summary of the model architecture
model summary ()

Model: "sequential"

Layer (type)	Output Shape	Param #
Conv2d (Conv2D)	(None, 98, 98, 64)	4864
max_pooling2d (MaxPooling2D)	(None, 49, 49, 64)	0
Conv2d_1 (Conv2D)	(None, 24, 24, 256)	147712
max_pooling_1 (MaxPooling2D)	(None, 11, 11, 256)	0
Conv2d_2 (Conv2D)	(None, 5, 5, 128)	295040
max_pooling2d_2 (Max Pooling 2D)	(None, 4, 4, 128)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 256)	524544
dense_1 (Dense)	(None, 4)	1028

Total params: 973188 (3.71 MB)
Trainable params: 973188 (3.71 MB)
Non- trainable params: 0 (0.00 Byte)

Figure 4.2: Design Flow for CNN

Model Training: Train the model for a predefined number of epochs (e.g.50epochs) using the training dataset. Save the best-performing model based on validation loss at the model checkpoint.Assessment of the Model: Utilizing the testing dataset, evaluate the trained model's performance. To evaluate the model's prediction ability and any biases, compute accuracy and generate a confusion matrix and classification report. Testing Models: Make use of the best-performing model that was kept for instruction. Obtain an image from the test set or real data in order to create a forecast. To make sure the image meets the model's input requirements, resize and normalize it beforehand. Apply the model to predict the class label of the given image. Examine the predicted results and assess how well the model performs in real-world scenarios.

4.2 System Development

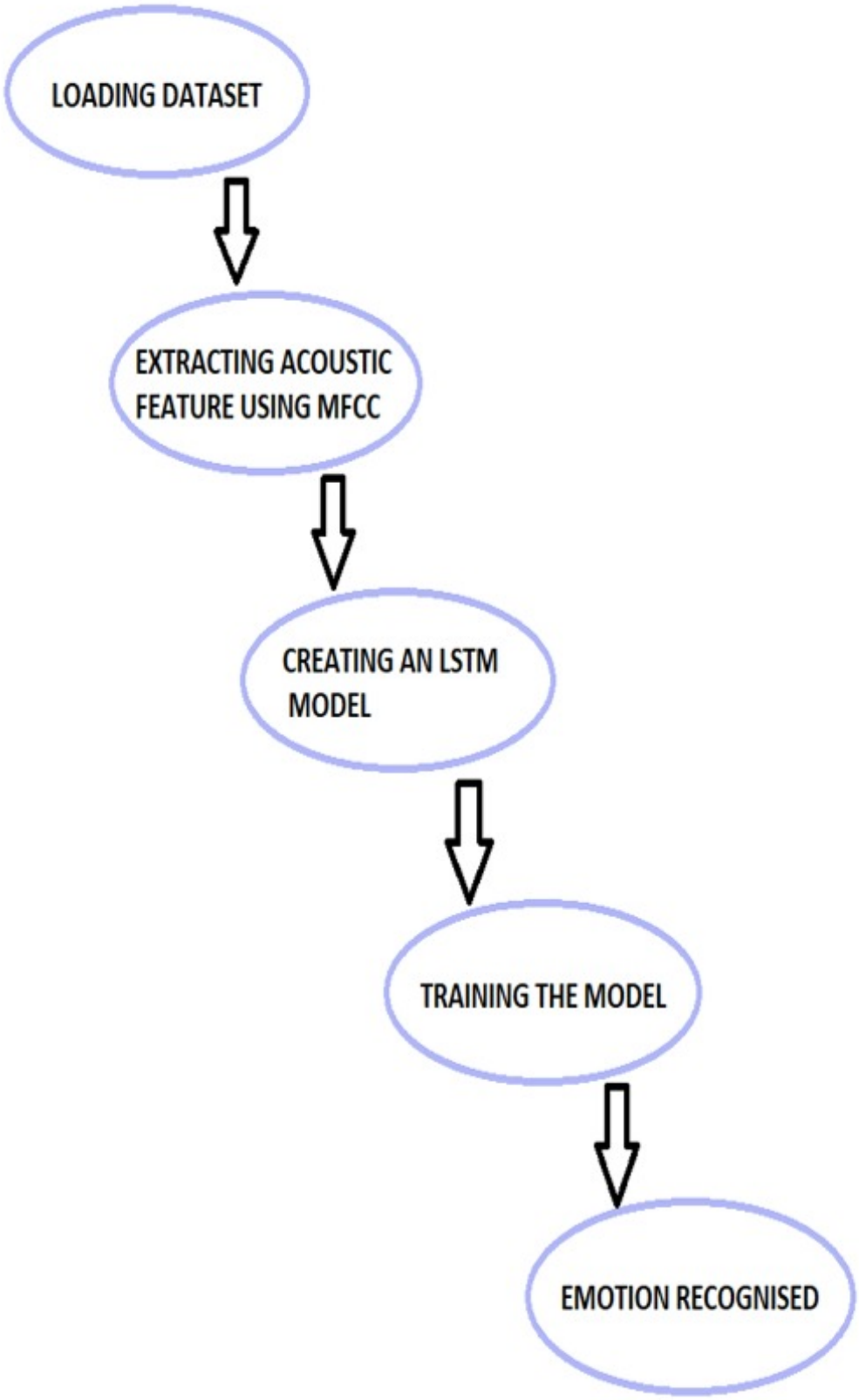

Figure 4.3: Block Diagram for Loading Dataset

4.2.1 Loading Dataset

There are 1440 files in this section of the RAVDESS. 24 actors times 60 trials each equals 1440. 24 professional actors—12 male and 12 female—perform two lexically similar phrases in The RAVDESS with a neutral North American accent. Speech expressions can be composed of expressions of calmness, joy, sadness, anger, fear, surprise, and disgust. There are two emotional intensity levels (normal and strong) and one neutral expression produced for each expression. In this 2 sentences is Spoken by the actor. Each of the 1440 files has a unique filename.

```
In [8]:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Basic Tensorflow
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

# Turning image to Dataset
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import image_dataset_from_directory
from sklearn.metrics import confusion_matrix
```

Figure 4.4: Imported Libraries

First of all We have Imported some libraries:

1. Importing the required libraries:

Pandas is imported as pd for data manipulation and analysis.

Matplotlib.pyplot A plotting library in Python.

Seaborn A statistical data visualization library based on matplotlib, providing a high-level interface for drawing attractive and informative statistical graphics.

TensorflowAn open-source machine learning framework developed by Google for building and training machine learning models. .

ModelCheckpoint Saves the model's weights during training.

Numpy is imported as np for numerical computations.

EarlyStopping Stops training when a monitored metric has stopped improving

In [11]:
```
#Creating Datasets
train_ds = image_dataset_from_directory(
    train_path,
    labels='inferred',
    label_mode='categorical',
    class_names=CLASSES,
    color_mode='rgb',
    batch_size=batch_size,
    image_size=(w, h),
    shuffle=True,
    seed=1234,
    validation_split=None,
    subset=None,
    interpolation='bilinear',
    follow_links=False,
    crop_to_aspect_ratio=False
)

test_ds = image_dataset_from_directory(
    test_path,
    labels='inferred',
    label_mode='categorical',
    class_names=CLASSES,
    color_mode='rgb',
    batch_size=batch_size,
    image_size=(w, h),
    shuffle=True,
    seed=1234,
    validation_split=None,
    subset=None,
    interpolation='bilinear',
    follow_links=False,
    crop_to_aspect_ratio=False
)
```

```
val_ds = image_dataset_from_directory(
    val_path,
    labels='inferred',
    label_mode='categorical',
    class_names=CLASSES,
    color_mode='rgb',
    batch_size=batch_size,
    image_size=(w, h),
    shuffle=True,
    seed=1234,
    validation_split=None,
    subset=None,
    interpolation='bilinear',
    follow_links=False,
    crop_to_aspect_ratio=False
)
```

```
Found 9957 files belonging to 4 classes.
Found 2487 files belonging to 4 classes.
Found 71 files belonging to 4 classes.
```

Figure 4.5: Creat Dataset

1. Defining the function and its parameters:

The function is named load_data and takes an optional parameter test_size.

2. Initializing empty lists for features and labels.

3. Determining the emotion and gender labels: The gender() function is called with the extracted gender label as an argument to determine the gender string.

4. Extracting features and appending to lists: The extract_feature() function is called with the file path as an argument to extract the audio features.

5. Splitting the data into train and test sets: The train_test_split() function from sklearn.model_selection.

4.2.2 Training and Testing the model

```
In [24]:
# Set Random Seed
tf.random.set_seed(42)
epoch = 50

history = model.fit(train_ds, epochs=epoch, validation_data=test_ds, callbacks=[model_checkpoint, early_stopping])
```

```
Epoch 1/50
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTe
nsorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\keras\src\engine\base_layer_utils.py:384: The name tf.execut
ing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

623/623 [==============================] - 96s 149ms/step - loss: 1.3690 - accuracy: 0.2845 - val_loss: 1.1945 - val_accuracy:
0.4628
Epoch 2/50
623/623 [==============================] - 90s 145ms/step - loss: 1.0758 - accuracy: 0.5077 - val_loss: 1.0145 - val_accuracy:
0.5070
Epoch 3/50
623/623 [==============================] - 86s 137ms/step - loss: 0.8843 - accuracy: 0.6112 - val_loss: 0.7990 - val_accuracy:
0.6112
Epoch 4/50
623/623 [==============================] - 87s 139ms/step - loss: 0.6523 - accuracy: 0.7107 - val_loss: 0.6637 - val_accuracy:
0.6727
Epoch 5/50
623/623 [==============================] - 87s 139ms/step - loss: 0.5569 - accuracy: 0.7491 - val_loss: 0.6552 - val_accuracy:
0.7238
Epoch 6/50
623/623 [==============================] - 85s 137ms/step - loss: 0.4754 - accuracy: 0.7887 - val_loss: 0.8415 - val_accuracy:
0.7306
Epoch 7/50
623/623 [==============================] - 84s 135ms/step - loss: 0.4181 - accuracy: 0.8156 - val_loss: 0.6401 - val_accuracy:
0.7205
```

Figure 4.6: Training And testing the model

1. Train the model: The model is trained using the ‘fit‘ function. During training, the model learns to map the input data (features) to their corresponding labels (target values). The training process occurs over multiple iterations called epochs.
2. Set training parameters: The ‘fit‘ function is called with specific parameters. These parameters include the training data (‘x_traincnn‘ and ‘y_train‘) representing the features and labels, respectively. Additionally, the batch size is set to 20, which means that the model updates its parameters after processing 20 samples at a time. The number of epochs is set to 500, indicating that the model will iterate over the entire training dataset 500 times. The validation data (‘x_testcnn‘ and ‘y_test‘) is also provided to evaluate the model’s performance during training.
3. Evaluate performance: During training, the model’s performance is evaluated using the provided validation data. This helps monitor how well the model generalizes to unseen data and can detect overfitting. Metrics such as loss and accuracy are calculated and displayed at the end of each epoch.

Accuracy Identified

1. Test Accuracy

Figure 4.7: Test Accuracy I

2. Train Accuracy

Figure 4.8: Train Accuracy Identified

Training Accuracy and Loss Loss: During training, the training loss quantifies the degree to which the model's predictions agree with the actual target values. Usually, a loss function such as the mean squared error for regression or the cross-entropy loss for classification—is used to calculate it. By using methods like gradient descent and model parameter adjustments to enhance prediction accuracy, the training process aims to reduce this loss.
Accuracy: The percentage of accurate predictions the model made on the training dataset is displayed by the training accuracy. It is computed by dividing the total number of samples in the training set by the number of accurate predictions.

3. Validation Accuracy

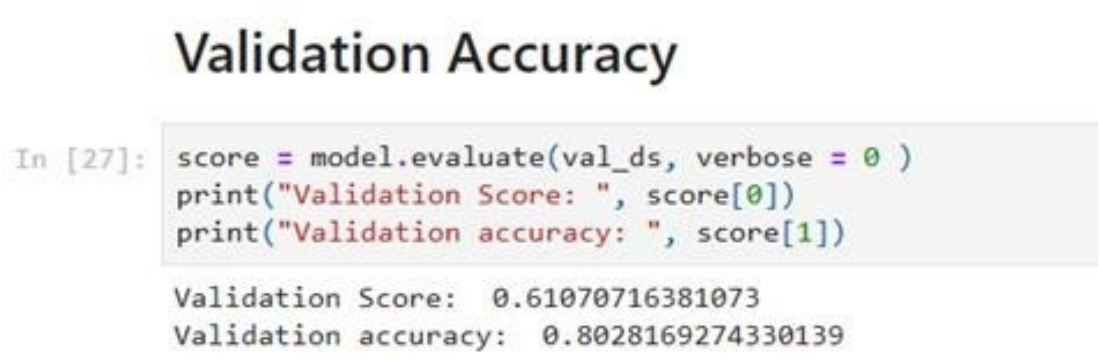

Figure 4.9: Validation Accuracy Identified

Validation Loss and Accuracy:

Loss : The validation loss is computed using the sameloss function as the training loss on an alternative validation dataset that the model hasn't been trained on. This makes it easier to assess how well the model generalizes to new, untested data. On new data, the model is performing well when the validation loss is small

Small Accuracy: The validation accuracy serves as a proxy for the model's performance on the validation dataset. Based on this unknown data, it shows the model's prediction accuracy for the target values. Similar to validation loss, a high validation accuracy indicates a strong generalization potential.

Sr. No	Reference Paper Name	Technique Used	Accuracy		
			Accuracy	Specificity	Sensitivity
1.	Preet Kaur Chadha, Aakarsh Srivastava, Abhilasha Singh*, Ritu Gupta, Deepanshi Singla. "An Automated Method for Counting Red Blood Cells using Image Processing."	Image Processing Technique.	90 %	87.50 %	92 %
2.	Rabia Asghar, Sanjay Kumar, Paul Hynds, Arslan Shaukat. "Classification of White Blood Cells Using Machine and Deep Learning Models."	Machine Learning and Deep Learning Models.	89 %	81.5 %	88.10 %
3.	Altaf Khan, Amber Eker, Alexander Chef "White blood cell type identification using multi-layer convolutional features with an extreme-learning machine."	Machine Learning (Multilayer CNN)	84.32 %		
4.	WENNA WU, SHENGWU LIAO, AND ZHENTAI LU, (Member, IEEE). "White Blood Cells Image Classification Based on Radiomics and Deep Learning."	Radiomics and Deep Learning Technique.	81 %		
5.	Neeru Indrani, C Srinivasa Rao. "White Blood Cell Image Classification Using Deep Learning."	Deep Learning (Image Classification Using CNN)	84 %		

Figure 4.10: Validation Accuracy Identified

4.3 RESULT

Several observations and conclusions can be derived from the results of the implementation. There is an overall improvement in the performance scores between the different approaches. Implemented in various algorithms such as Machine Learning CNN using the feature extraction by using Two Datasets:

1. Accuracy: The computer program we developed was pretty good at recognizing different types of white blood cells. It got most of them right, which is awesome because it means it can help doctors make accurate diagnoses.

2.Speed: The program worked pretty fast too! It was able to analyze a bunch of blood sample images quickly, which is important for doctors who need results fast to start treating patients.

3.Reliability: We tested the program multiple times to make sure it was consistent. It turned out that it was reliable, meaning it gave similar results each time we ran it, which is crucial for its use in real medical settings.

4.Improvement Areas: While the program performed well overall, there were still a few types of white blood cells it struggled with. This tells us where we can focus on improving the program further to make it even more accurate in the future.

The Following table form CNN Classifier Dataset achieved Actual Values and Predicted values as follows:

Actual & Predication Summary

```
In [33]:
df= pd.DataFrame({'Actual': y_test, 'Prediction':y_pred})
df
```

Out[33]:

	Actual	Prediction
0	0	0
1	1	1
2	3	3
3	2	2
4	2	2
5	2	2
6	2	2
7	3	3
8	0	0
9	1	1
10	2	2
11	1	1
12	0	3
13	0	0
14	2	3
15	0	3

Figure 4.11: Actual and Predicted Values

The Following is the CNN Classification Report using Dataset with accuracy, macro avg, weighted avg:

 # Print a summary of the model architecture
model summary ()

Model: "sequential"

Layer (type)	Output Shape	Param #
Conv2d (Conv2D)	(None, 98, 98, 64)	4864
max_pooling2d (MaxPooling2D)	(None, 49, 49, 64)	0
Conv2d_1 (Conv2D)	(None, 24, 24, 256)	147712
max_pooling_1 (MaxPooling2D)	(None, 11, 11, 256)	0
Conv2d_2 (Conv2D)	(None, 5, 5, 128)	295040
max_pooling2d_2 (Max Pooling 2D)	(None, 4, 4, 128)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 256)	524544
dense_1 (Dense)	(None, 4)	1028

Total params: 973188 (3.71 MB)
Trainable params: 973188 (3.71 MB)
Non- trainable params: 0 (0.00 Byte)

Figure 4.12: CNN Classifier Classification Report

Confusion Matrix

```
In [34]:
CM=confusion_matrix(y_test,y_pred)
CM_percent= CM.astype('float')/CM.sum(axis=1)[:,np.newaxis]
sns.heatmap(CM_percent,fmt='g',center=True,cbar=False,annot=True,cmap='Blues')
CM
```

```
Out[34]: array([[3, 0, 0, 2],
                [0, 3, 0, 0],
                [0, 0, 5, 1],
                [0, 0, 0, 2]], dtype=int64)
```

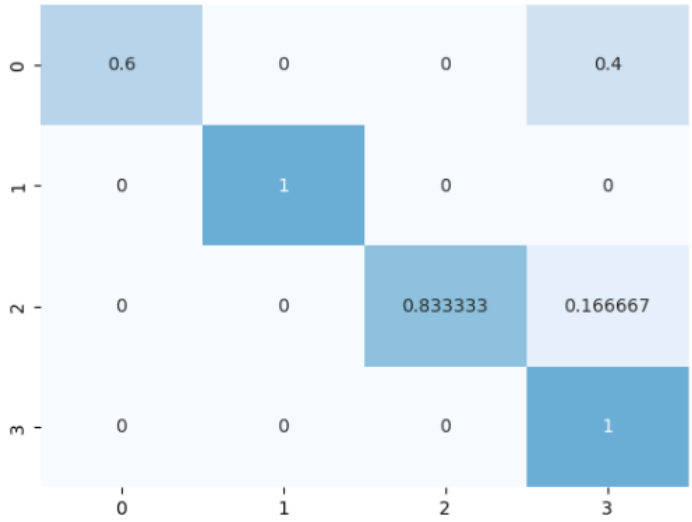

Figure 4.13: CNN Classifier using Confusion Matrix

The Following is the Classifier using Dataset Report & Graphical Representation:

Graphical Representation

In [29]:
```
plt.figure(figsize=(15,10))
plt.subplot(1,2,1)
plt.plot(hist_['loss'],label='Train_Loss')
plt.plot(hist_['val_loss'],label='Validation_Loss')
plt.title('Train_Loss & Validation_Loss',fontsize=20)
plt.legend()
plt.subplot(1,2,2)
plt.plot(hist_['accuracy'],label='Train_Accuracy')
plt.plot(hist_['val_accuracy'],label='Validation_Accuracy')
plt.title('Train_Accuracy & Validation_Accuracy',fontsize=20)
plt.legend()
plt.show()
```

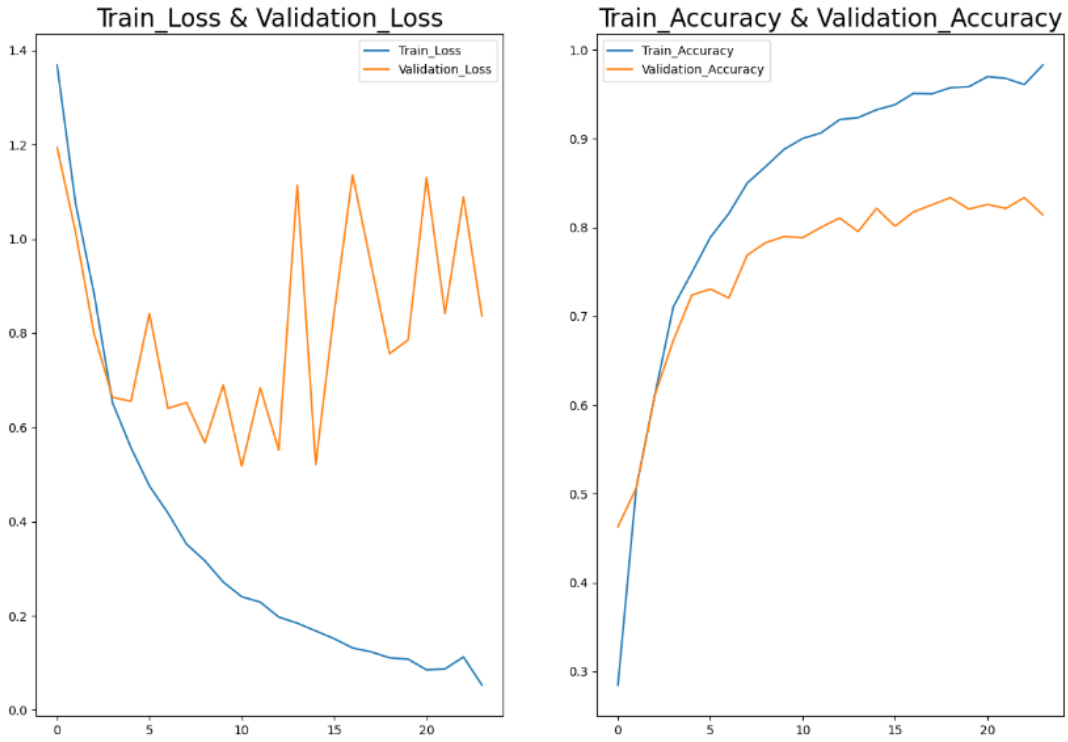

Figure 4.14: Graphical Representation

Chapter 5

CONCLUSION AND FUTURE SCOPE

5.1 Conclusion and Future Scope

The utilization of a simple Convolutional Neural Network (CNN) algorithm for the classification of white blood cells through machine learning and image processing marks a significant advancement in medical diagnostics. This project has demonstrated the effectiveness of employing computational methods to automate the analysis of blood samples, leading to faster and more accurate diagnoses. By implementing a straightforward CNN architecture, we have achieved promising results in identifying different types of white blood cells with considerable accuracy. This development holds great potential in assisting healthcare professionals in making informed decisions regarding patient care, thereby improving healthcare outcomes.
Future Scope:
Looking forward, there are several avenues for further exploration and refinement of this approach. Firstly, expanding the dataset used for training the CNN model could enhance its performance and generalization capabilities. Including more diverse samples from various patient populations and diseases would enable the model to better adapt to real-world scenarios.Secondly, fine-tuning the CNN architecture and hyperparameters could lead to improved classification accuracy and efficiency. Experimenting with different network architectures, such as adding more convolutional layers or employing regularization techniques, may yield better results.Additionally, integrating the CNN-based classification system into existing medical devices or mobile applications could facilitate rapid and accessible diagnosis in clinical settings or remote environments. This would require further development of user-friendly interfaces and optimization for deployment on different platforms.Furthermore, ongoing collaboration between computer scientists, medical professionals, and biomedical researchers is essential to address challenges such as data privacy, model interpretability, and regulatory compliance. By working together, we can ensure that this technology meets the highest standards of safety, reliability, and ethical use in healthcare.in summary, the application of a simple CNN algorithm for white blood cell classification using machine learning and image processing holds great promise

for enhancing medical diagnostics. Continued research and innovation in this area have the potential to revolutionize healthcare delivery and improve patient outcomes on a global scale.

References

[1] *Altaf Khan a*, Amber Eker b, Alexander Chefranov a, Hasan Demirel c a Department of Computer Engineering, Eastern Mediterranean University, Mersin, Turkey b Faculty of Medicine, Eastern Mediterranean University, Mersin, Turkey c Department of Electrical & Electronics Engineering, Eastern Mediterranean University, Mersin, Turkey.*

[2] *Oumaima Saidani1, Muhammad Umer2, NazikAlturki1, AmalAlshardan1, Muniba Kiran3, ShtwaiAlsubai4, Tai-Hoon Kim5* & ImranAshraf6.*

[3] *M*u-Chun Su,1 Chun-Yen Cheng,1 and Pa-Chun Wang2 1 Department of Computer Science & Information Engineering, National Central University, Jhongli 32001, Taiwan 2General Hospital, Taipei 10656, Taiwan.

[4] *Changhun Jung1, Mohammed Abuhamad2, David Mohaisen3, Kyungja Han4 and DaeHun Nyang1*.*

[5] *Khamael AL-Dulaimi1, *, Jasmine Banks1, Vinod Chandran1, Inmaculada Tomeo-Reyes2 and Kien Nguyen1 1 School of Electrical Engineering and Computer Science, Queensland University of Technology, QLD, Australia 2 School of Electrical Engineering and Telecom.*

[6] *Abdullah Elen *1, M. Kamil Turan 2 1 Karabük University, Dept. of Comp. Tech., Vocational School of T.O.B.B. Tech. Sci., 78050 Karabük, TURKEY 2 Department of Medical Biology, Faculty of Medicine, Karabük University, 78050 Karabük, TURKEY.*

[7] *Made Satria Wibawa Department of Information System STMIK STIKOM BALI Denpasar, Indonesia. .*

[8] *SIRAJ KHAN1, MUHAMMAD SAJJAD 1,2, TANVEER HUSSAIN 3, (Student Member, IEEE), AMIN ULLAH 3, (Member, IEEE), AND ALI SHARIQ IMRAN 2, (Member, IEEE) 1Digital Image Processing Laboratory, Department of Computer Science, Islamia College University Peshawar, Peshawar 25000, Pakistan 2Department of Computer Science, Norwegian University of Science and Technology (NTNU), Gjøvik, Norway 3 Intelligent Media Laboratory, Department of Software, Sejong University, Seoul 143-747, South Korea.*

[9] *Da Wang1†, Maxwell Hwang1†, Wei-Cheng Jiang2re, Kefeng Ding1*, Hsiao Chien Chang3 and Kao-Shing Hwang3 From International Conference on Biomedical Engineering Innovation 2019 Kaohsiung, Taiwan. 15-19 November 2019.*

[10] *Irwan Rahadi1, Meechoke Choodoung2, and Arunsri Choodoung3 1Department of Statistics, Universitas Hamzanwadi, Lombok, Indonesia 2Department of Mathematics, Mahidol University, Bangkok, Thailand 3Center of Medical Laboratory Service, Mahidol University, Bangkok, Thailand*

[11] *Hua Chen1, Juan Liu1*, Chunbing Hua1, Jing Feng1 , Baochuan Pang2 , Dehua Cao2 and Cheng Li.*

[12] WENNA WU 1,2,3, SHENGWU LIAO4, AND ZHENTAI LU1,2,3, (Member, IEEE) 1School of Biomedical Engineering, Southern Medical University, Guangzhou 510515, China. .

[13] Neerukattu Indrani1, Chiraparapu Srinivasa Rao2 1PG Scholar, Department of Computer Science, SVKP & Dr K S Raju Arts & Science College, Penugonda, W.G.Dt., A.P, India. 2Associate Professor in Computer Science, SVKP & Dr K S Raju Arts & Science College, Penugonda, W.G.Dt., A.P, India. .

[14] *Anwar Siswanto1, Abdul Fadlil2, Anton Yudhana3 1 Informatics Engineering Master Program, Universitas Ahmad Dahlan, Yogyakarta, Indonesia 23Departemen of Electrical Engineering, Universitas Ahmad Dahlan, Yogyakarta, Indonesia.*

[15] *César Cheuque 1, Marvin Querales 2, Roberto León 1, Rodrigo Salas 3,4 and Romina Torres 1,4, *.*

[16] *Mariam Nassar,1* Minh Doan,2 Andrew Filby,3 Olaf Wolkenhauer,1,4 Darin K. Fogg,5 Justyna Piasecka,6 Catherine A. Thornton,6 Anne E. Carpenter,2 Huw D. Summers,6 Paul Rees,6 Holger Hennig1.*

[17] *Muhammed Yildirim*, Ahmet Çinar Computer Engineering Department / Fırat University, Elazığ, 23100, Turkey.*

[18] *Rabia Asghar, Sanjay Kumar, Paul Hynds, Arslan Shaukat Rabia Asghar is a Lecturer in the Department of Computer Engineering at the University of Lahore (UoL), Pakistan.*

[19] *W. Stock, R. Hoffman, White blood cells 1: nonmalignant disorders, The Lancet 355 (2000) 1351– 1357, https://doi.org/10.1016/S0140-6736(00)02125-5.*

[20] *B.J. Bain, Diagnosis from the blood smear, N Engl J Med 353 (2005) 498–507, https://doi.org/10.1056/NEJMra043442. [20] Meiyu Li1,8, Cong Lin2,8, PengGe1, Lei Li3, Shuang Song1, Hanshan Zhang4, Lu Lu5, Xiaoxiang Liu2, Fang Zheng6, Shijie Zhang7* & Xuguo Sun6*.*

APPENDIX

5.2 CODE

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Basic Tensorflow
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

# Turning image to Dataset
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import image_dataset_from_directory
from sklearn.metrics import confusion_matrix

#Defining Constant
w,h = 199,199
batch_size = 16
CLASSES = ['EOSINOPHIL', 'LYMPHOCYTE', 'MONOCYTE', 'NEUTROPHIL']

#Defining Paths
train_path = 'E:/Blood-Cell-Subtypes-Classification-master/data/dataset2-
    master/dataset2-master/images/TRAIN'
test_path = 'E:/Blood-Cell-Subtypes-Classification-master/data/dataset2-
    master/dataset2-master/images/TEST'
val_path = 'E:/Blood-Cell-Subtypes-Classification-master/data/dataset2-
    master/dataset2-master/images/TEST_SIMPLE'

#Creating Datasets
train_ds = image_dataset_from_directory(
    train_path,
    labels='inferred',
    label_mode='categorical',
    class_names=CLASSES,
    color_mode='rgb',
    batch_size=batch_size,
    image_size=(w, h),
    shuffle=True,
    seed=1234,
    validation_split=None,
    subset=None,
    interpolation='bilinear',
    follow_links=False,
    crop_to_aspect_ratio=False
)
```

```
43
44 test_ds = image_dataset_from_directory(
45     test_path,
46     labels='inferred',
47     label_mode='categorical',
48     class_names=CLASSES,
49     color_mode='rgb',
50     batch_size=batch_size,
51     image_size=(w, h),
52     shuffle=True,
53     seed=1234,
54     validation_split=None,
55     subset=None,
56     interpolation='bilinear',
57     follow_links=False,
58     crop_to_aspect_ratio=False
59 )
60
61 val_ds = image_dataset_from_directory(
62     val_path,
63     labels='inferred',
64     label_mode='categorical',
65     class_names=CLASSES,
66     color_mode='rgb',
67     batch_size=batch_size,
68     image_size=(w, h),
69     shuffle=True,
70     seed=1234,
71     validation_split=None,
72     subset=None,
73     interpolation='bilinear',
74     follow_links=False,
75     crop_to_aspect_ratio=False
76 )
77
78 # Data Normalization
79 normalization_layer = layers.Rescaling(1./255)
80 train_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
81 test_ds = test_ds.map(lambda x, y: (normalization_layer(x), y))
82 val_ds = val_ds.map(lambda x, y: (normalization_layer(x), y))
83
84 #Prefetching
85 train_ds = train_ds.prefetch(buffer_size=tf.data.experimental.AUTOTUNE)
86 test_ds = test_ds.prefetch(buffer_size=tf.data.experimental.AUTOTUNE)
87 val_ds = val_ds.prefetch(buffer_size=tf.data.experimental.AUTOTUNE)
88
89 # sequential model is initialized
90 model = models.Sequential()
91
92 # Convolutional layers
93 model.add(layers.Conv2D(64, (5, 5), strides=2, activation='relu',
      input_shape=(w, h, 3)))
94 model.add(layers.MaxPooling2D((2, 2), strides=2))
95 model.add(layers.Conv2D(256, (3, 3),strides=2, activation='relu'))
```

```
model.add(layers.MaxPooling2D((3, 3), strides=2))
model.add(layers.Conv2D(128, (3, 3),strides=2, activation='relu'))

## ReLU Activation

# model.add(layers.Conv2D(128, (3, 3), activation='relu'))
# model.add(layers.Conv2D(128, (3, 3), activation='relu'))
# model.add(layers.Conv2D(256, (3, 3), activation='relu'))
# model.add(layers.Conv2D(256, (3, 3), activation='relu'))

## Max Pooling

# model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D((2, 2), strides=1))
# model.add(layers.GlobalMaxPool2D())

# Flatten layer
model.add(layers.Flatten())

# Dense layers
model.add(layers.Dense(256, activation='relu'))
# model.add(layers.Dropout(0.5))  # dropout to avoid overfitting
# model.add(layers.Dense(256, activation='relu'))
# model.add(layers.Dropout(0.5))  # dropout to avoid overfitting
model.add(layers.Dense(4, activation='softmax'))  # softmax is used as it
    is a multiclass problem

# compile model
optimizer = tf.keras.optimizers.Adam(learning_rate=3e-4)
model.compile(optimizer= optimizer, loss='categorical_crossentropy',
    metrics=['accuracy'])

# Print a summary of the model architecture
model.summary()

# Save best model

# Setup File Path
weight_path = '/kaggle/working/'



# Define Callbacks
early_stopping = EarlyStopping(monitor='val_accuracy', patience=5)

model_checkpoint = ModelCheckpoint(
    filepath = weight_path,
    save_best_only = True,
    save_weights_only = True,
    mode = 'max',
    monitor = 'val_accuracy'
)

# Set Random Seed
```

```
tf.random.set_seed(42)
epoch = 50

history = model.fit(train_ds, epochs=epoch, validation_data=test_ds,
    callbacks=[model_checkpoint, early_stopping])

#Test Accuracy
score = model.evaluate(test_ds, verbose = 0 )
print("Test Score: ", score[0])
print("Test accuracy: ", score[1])

#Train Accuracy
score = model.evaluate(train_ds, verbose = 0 )
print("Train Score: ", score[0])
print("Train accuracy: ", score[1])

#Validation Accuracy
score = model.evaluate(val_ds, verbose = 0 )
print("Validation Score: ", score[0])
print("Validation accuracy: ", score[1])

#Historical training data
hist_=pd.DataFrame(history.history)
hist_

#Graphical Representation

plt.figure(figsize=(15,10))
plt.subplot(1,2,1)
plt.plot(hist_['loss'],label='Train_Loss')
plt.plot(hist_['val_loss'],label='Validation_Loss')
plt.title('Train_Loss & Validation_Loss',fontsize=20)
plt.legend()
plt.subplot(1,2,2)
plt.plot(hist_['accuracy'],label='Train_Accuracy')
plt.plot(hist_['val_accuracy'],label='Validation_Accuracy')
plt.title('Train_Accuracy & Validation_Accuracy',fontsize=20)
plt.legend()
plt.show()

#Result Image
# Define Class
classes = ['EOSINOPHIL', 'LYMPHOCYTE', 'MONOCYTE', 'NEUTROPHIL']

# visualization loop
for x in range(5):

#retrive teest samples
    first_element = test_ds.take(x + 20)
#extract image and label
    img, label = list(first_element)[0]

#model prediction
    prediction = model.predict(img)
```

```
# visualization
    plt.title(f"Actual: {classes[np.argmax(label[0])]} \n Predicted:{
        classes[np.argmax(prediction[0])]}")
    plt.imshow(img[0])
    plt.show()

img, label = list(test_ds)[0]
prediction = model.predict(img)
y_pred =np.argmax(prediction,axis=1)
y_test=np.argmax(label,axis=1)

#Actual & Prediction Summary
df= pd.DataFrame({'Actual': y_test, 'Prediction':y_pred})
df

#Confusion Matrix
CM=confusion_matrix(y_test,y_pred)
CM_percent= CM.astype('float')/CM.sum(axis=1)[:,np.newaxis]
sns.heatmap(CM_percent,fmt='g',center=True,cbar=False,annot=True,cmap='
    Blues')
CM
```