CLASSIFICATION OF WHITE BLOOD CELL USING CONVOLUTIONAL NEURAL NETWORK(CNN)

Report submitted to the SASTRA Deemed to be University as the requirement for the course

CSE300 - MINI PROJECT

Submitted by

JANARTHANAN N
(123003084, B. TECH Computer Science and Engineering)

KISHORE Y
(123003112, B. TECH Computer Science and Engineering)

NAVANEETH M
(12, B. TECH Computer Science and Engineering)

January 2022

SCHOOL OF COMPUTING
THANJAVUR, TAMIL NADU, INDIA – 613 401

SCHOOL OF COMPUTING

THANJAVUR– 613 401

Bonafide Certificate

This is to certify that the report titled "**Classification of White Blood Cell using Convolutional Neural Network**" submitted as a requirement for the course, CSE300: **MINI PROJECT** for B.Tech. is a bonafide record of the work done by **Mr. Janarthanan N (123003084, B. Tech Computer Science and Engineering**), **Mr. Kishore Y (123003112, B. Tech Computer Science and Engineering), Mr. Navaneeth M (123003159, B. Tech Computer Science and Engineering)**during the academic year 2021-22, in the School of Computing, under my supervision.

Signature of Project Supervisor **:**

Name with Affiliation **:** Dr V S Shankar Sriram, Associate Dean – CSE, School of Computing, SASTRA Deemed University

Date **:**

Mini Project *Viva voc*e held on _____

Examiner 1 **Examiner 2**

Acknowledgements

We would like to thank our Honorable Chancellor **Prof. R. Sethuraman** for providing us with an opportunity and the necessary infrastructure for carrying out this project as a part of our curriculum.

We would like to thank our Honorable Vice-Chancellor **Dr. S. Vaidhyasubramaniam** and **Dr. S. Swaminathan,** Dean**,** Planning & Development, for the encouragement and strategic support at every step of our college life.

We extend our sincere thanks to **Dr. R. Chandramouli,** Registrar, SASTRA Deemed to be University for providing the opportunity to pursue this project.

We extend our heartfelt thanks to **Dr. A. Umamakeswari**, Dean, School of Computing, **Dr. V. S. Shankar Sriram**, Associate Dean, Department of Computer Science and Engineering, **Dr. R. Muthaiah**, Associate Dean, Department of Information Technology and Information & Communication Technology and **Dr. B. Santhi**, Associate Dean, Department of Computer Application.

Our guide **Dr. K. Kannan**, Associate Dean - CSE, School of Computing was the driving force behind this whole idea from the start. His deep insight in the field and invaluable suggestions helped us in making progress throughout our project work. We also thank the project review panel members for their valuable comments and insights which madethis project better.

We would like to extend our gratitude to all the teaching and non-teaching faculties of the School of Computing who have either directly or indirectly helped us in the completion of the project.

We gratefully acknowledge all the contributions and encouragement from my family and friends resulting in the successful completion of this project. We thank you all for providing me an opportunity to showcase my skills through project.

List of Figures

Figure No.	Title	Page No.
1	CNN Architecture	8

List of Tables

Table No.	Table name	Page No.
1	Accuracy using ResNet	24

Abstract

The sudden boom in the field of Machine Learning is because of its ability to learn from the data in the real-world. In the recent times, people have identified its extensive use in the biomedical field. This paper classifies the White Blood Cells using Convolution Neural Network So, a better and accurate analysis of the white blood cells can be of great help to the doctors in diagnosing cases such as cancer and viral infections. The proposed approach in this paper will be able to classify the different types of WBC in an efficient way which is better than the other approaches. We use a public dataset which is available on Kaggle. For a better understanding of the result, we compare the performance of the proposed framework with another four techniques in terms of accuracy, specificity and sensitivity.

KEY WORDS: Convolution Neural Network, Machine Learning, Accuracy, Specificity, Sensitivity.

Table of Contents

Title	Page No.
Bonafide Certificate	ii
Acknowledgements	iii
List of Figures	iv
List of Tables	iv
Abstract	v
1. Summary of the base paper	7
2 Merits and Demerits of the base paper	9
3. Source Code	10
4. Snapshots	21
5. Conclusion and Future Plans	25
6. References	26
7. Base Paper	27

CHAPTER 1

SUMMARY OF THE BASE PAPER

Classification of White Blood cell using Convolution Neural Network

https://doi.org/10.1016/j.bspc.2021.103156

In this base paper, it is found that they have used Adam optimizer with Relu activation function and has been run on 20 epochs. The main contribution of the base paper was the usage of Convolution Neural Network to classify and identify the type of White Blood Cell, and that the technique used is evaluated on Blood Cell Count and Classification Dataset. Along with this, to identify from the blood smear image, they have used certain performance metrics such as accuracy, specificity and sensitivity.

Similar works have been proposed in the past using various techniques and different datasets. In this base paper, they have used a publicly available dataset, whereas in the past works, locally available datasets have been used.

This paper was based on the fact of increasing number of COVID-19 cases. In order to produce an effective vaccination, it is necessary to identify the types of White Blood Cell that acts with the vaccination and produces positive and efficient results against the virus. With a massive ratio of 600:1, it is very difficult to find a White Blood Cell amongst heavy cells of Red Blood Cells. So to find the White Blood Cell and to identify it's types is even more difficult among a large group of Red Blood Cells. Hence, this paper was introduced.

Various layers have been used in the given base paper, such as Batch Normalization, Convolutional, Dropout etc. The data has been pre-processed by cleaning the data, resizing the images, reduction of noise, augmenting and train-test division.

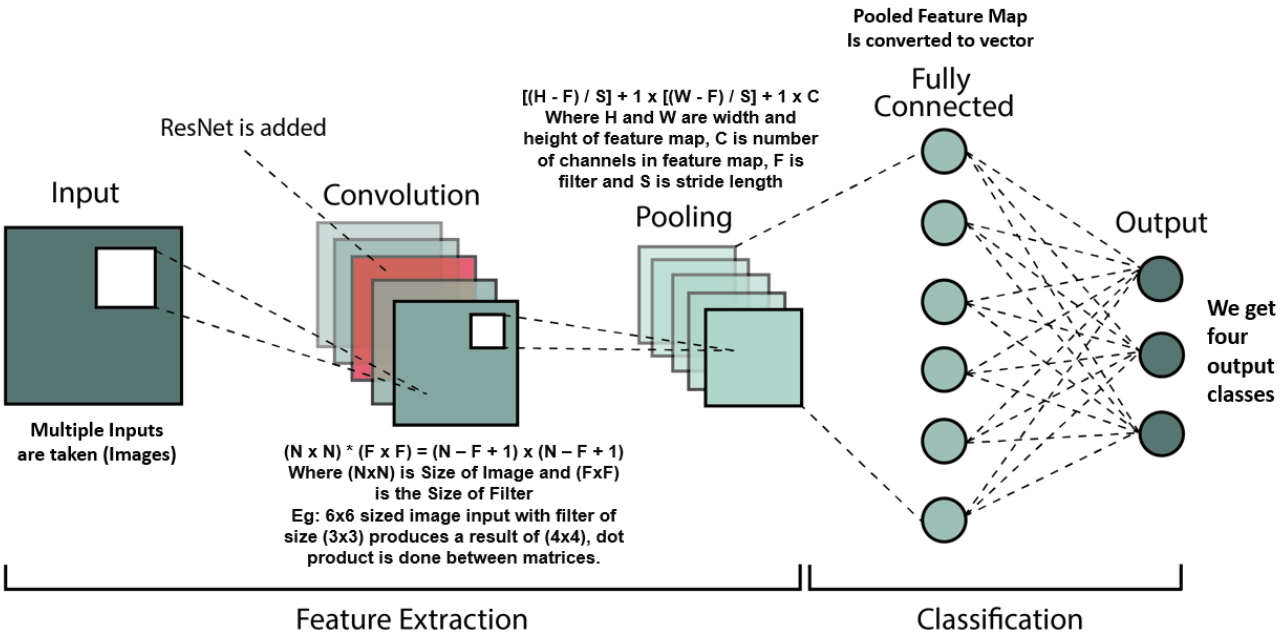


Figure 1: CNN Architecture using ResNet

CHAPTER 2

MERITS AND DEMERITS OF THE BASE PAPER

2.1 Merits of Proposed Methodology:

- Public dataset has been used. In the past works, locally available dataset has been used. This reduces the chance of identifying a type of White Blood Cell from a random sample, which will be present in practical situations. Since public dataset has been used, this model can be used for other samples in order to identify the type of White Blood Cell.
- Identification of type of White Blood Cell helps in understanding the working of vaccination that is being created for the disease for which protection is needed.
- The same classification can be used to solve other classification problems such as Red Blood Cell classification.

2.2 Demerit:

- Test accuracy is less.
- Large epochs is needed to produce higher accuracy, compared to ResNet.
- Novelty of base paper is not specific.

CHAPTER 3
SOURCE CODE

3.1 Using CNN:

```
import numpy as np
import pandas as pd

import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
```

```
train_datagen=ImageDataGenerator(
                                    rescale=1./255,
                                    shear_range=0.2,
                                    width_shift_range=0.1,
                                    height_shift_range=0.1,
                                    zoom_range=0.2,
                                    horizontal_flip=True,
                                    validation_split=0.2)
```

```
training_set=train_datagen.flow_from_directory(
        r'../input/dataset1/archive/dataset2-master/dataset2-master/images/TRAIN',
        target_size=(128,128),
        batch_size=32,
        class_mode='categorical',
        seed=42,
        shuffle=True,
        subset='training')

validation_set=train_datagen.flow_from_directory(
        r'../input/dataset1/archive/dataset2-master/dataset2-master/images/TRAIN',
        target_size=(128,128),
        batch_size=32,
        class_mode='categorical',
        seed=42,
        shuffle=False,
        subset='validation')
```

```
test_datagen=ImageDataGenerator(
        rescale=1./255,
        featurewise_center=False,
        samplewise_center=False,
        featurewise_std_normalization=False,
        samplewise_std_normalization=False)
test_set=test_datagen.flow_from_directory(
        r'../input/dataset1/archive/dataset2-master/dataset2-master/images/TEST',
        target_size=(128,128),
        batch_size=32,
        class_mode='categorical')
```

```
training_set.next()[1]
```

```
cnn=tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(filters=6,kernel_size=3,input_shape=[128,128,3]),
   tf.keras.layers.BatchNormalization(input_shape=[128,128,3]),
   tf.keras.layers.Activation('elu')*,
   tf.keras.layers.MaxPool2D(pool_size=2,strides=2),

   tf.keras.layers.Conv2D(filters=16,kernel_size=3),
   tf.keras.layers.BatchNormalization(),
   tf.keras.layers.Activation('elu'),
   tf.keras.layers.MaxPool2D(pool_size=2,strides=2),
   tf.keras.layers.BatchNormalization(),
   tf.keras.layers.Activation('elu'),
   tf.keras.layers.Conv2D(filters=64,kernel_size=3),
   tf.keras.layers.BatchNormalization(),
   tf.keras.layers.Activation('elu'),
   tf.keras.layers.MaxPool2D(pool_size=2,strides=2),
   tf.keras.layers.Conv2D(filters=128,kernel_size=3),
   tf.keras.layers.BatchNormalization(),
   tf.keras.layers.Activation('elu'),
   tf.keras.layers.MaxPool2D(pool_size=2,strides=2),

   tf.keras.layers.Dropout(0.25),
   tf.keras.layers.Flatten(),
   tf.keras.layers.Dense(units=128),
   tf.keras.layers.Activation('elu'),
   tf.keras.layers.Dense(units=4),
   tf.keras.layers.Activation('softmax')
])
cnn.summary()
```

*The activation function can be replaced with relu,selu and tanh

```
from keras import optimizers
opt =tf.keras.optimizers.Nadam(learning_rate=0.001)*
cnn.compile(loss='categorical_crossentropy',
                optimizer=opt,
                metrics=['accuracy'])
```

*The optimizer is replaced with Adagrad, Adam and RMSprop

```
r=cnn.fit(
   x=training_set,
   validation_data=validation_set,
   epochs=50,
   #callbacks=[
     #  tf.keras.callbacks.EarlyStopping(
      # monitor='val_loss',
      # patience=3,
      # restore_best_weights=True)
   # ]
)
```

```
loss, acc = cnn.evaluate(test_set,verbose=0)
print(f"test accuracy {acc*100}")
```

```
import numpy as np
from keras.preprocessing import image
test_image=image.load_img(r'../input/sample/eos.jpeg',target_size=(128,128))
test_image=image.img_to_array(test_image)
test_image=np.expand_dims(test_image,axis=0)
result=cnn.predict(test_image)
result
```

```
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
```

```
CLASS_NAMES=list(training_set.class_indices.keys())
CLASS_NAMES
```

```
predictions=np.argmax(cnn.predict(test_set),axis=1)

cm=confusion_matrix(test_set.labels,predictions)
print(cm)

clr=classification_report(test_set.labels,predictions,target_names=CLASS_NAMES)
print("classification_report:\n----------------\n",clr)
```

```
import matplotlib.pyplot as plt

accuracy = r.history['accuracy']
loss = r.history['loss']
val_accuracy = r.history['val_accuracy']
val_loss =r.history['val_loss']

print(f'Training Accuracy: {np.max(accuracy)}')
print(f'Training Loss: {np.min(loss)}')
print(f'Validation Accuracy: {np.max(val_accuracy)}')
print(f'Validation Loss: {np.min(val_loss)}')
```

```
plt.plot(r.history['accuracy'])
plt.plot(r.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

plt.plot(r.history['loss'])
plt.plot(r.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```

```
from tensorflow.keras.models import load_model

cnn.save('gfgModel.h5')
print("Model Saved")

savemodel=load_model('gfgmodel.h5')
savemodel.summary()

cnn.save_weights('gfgModelWeights')
print('Model Saved!')

savedModel = cnn.load_weights('gfgModelWeights')
print('Model Loaded!')
cnn.summary()
```

3.2 Using ResNet:

```
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.optimizers import Adam,SGD,RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np
import tensorflow as tf
```

```
cnn = ResNet50(include_top=False,weights="imagenet")
cnn.summary()
```

```
print("There are {} layers in model".format(len(cnn.layers)))
```

```
for layer in cnn.layers[:140]:
   layer.trainable = False

cnn.summary()
```

```
test_datagen = ImageDataGenerator()
train_datagen = ImageDataGenerator(validation_split=0.15,
                         vertical_flip=True,
                         horizontal_flip=True,
                         rotation_range=0.2
                        )

test_set = test_datagen.flow_from_directory(r'C:\Users\jana\Desktop\dataset2\TEST',
                                    target_size=(224,224),
                                    shuffle=False

                                   )
training_set = train_datagen.flow_from_directory(r'C:\Users\jana\Desktop\dataset2\TRAIN',
                                     subset="training",
                                     target_size=(224,224),

                                     )

validation_set = train_datagen.flow_from_directory(r'C:\Users\jana\Desktop\dataset2\TRAIN',
                                       subset="validation",
                                       target_size=(224,224)
                                      )
model.summary()
```

```
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
```

```
CLASS_NAMES=list(training_set.class_indices.keys())
CLASS_NAMES
```

```
import numpy as np
from keras.preprocessing import image
test_image=image.load_img(r'C:\Users\jana\Desktop\sample3\neutro.jpg',target_size=(224,224))
test_image=image.img_to_array(test_image)
test_image=np.expand_dims(test_image,axis=0)
result=np.argmax(model.predict(test_image))
result
model.summary()
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score,confusion_matrix

y_pred = model.predict_classes(test_set)

y_true = test_set.labels

print("Final test accuracy is {}%".format(accuracy_score(y_pred=y_pred,y_true=y_true)))

confMatrix = confusion_matrix(y_pred=y_pred,y_true=y_true)
confMatrix

plt.subplots(figsize=(6,6))
sns.heatmap(confMatrix,annot=True,fmt=".1f",linewidths=1.5)
plt.xlabel("Predicted Label")
plt.ylabel("Actual Label")
plt.show()
```

```
import matplotlib.pyplot as plt

accuracy = history.history['acc']
loss = history.history['loss']
val_accuracy = history.history['val_acc']
val_loss =history.history['val_loss']

print(f'Training Accuracy: {np.max(accuracy)}')
print(f'Training Loss: {np.min(loss)}')
print(f'Validation Accuracy: {np.max(val_accuracy)}')
print(f'Validation Loss: {np.min(val_loss)}')
```

```
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```

```
from tensorflow.keras.models import load_model

cnn.save('gfgModel.h5')
print("Model Saved")

savemodel=load_model('gfgmodel.h5')
savemodel.summary()

cnn.save_weights('gfgModelWeights')
print('Model Saved!')

savedModel = cnn.load_weights('gfgModelWeights')
print('Model Loaded!')
cnn.summary()
```

4.3 Interface Code:

#

```
import keras
from PIL import Image, ImageOps
import numpy as np

def teachable_machine_classification(img, weights_file):
    # Load the model
    model = keras.models.load_model(weights_file)

    # Create the array of the right shape to feed into the keras model
    data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)
    image = img
    #image sizing
    size = (224, 224)
    image = ImageOps.fit(image, size, Image.ANTIALIAS)

    #turn the image into a numpy array
    image_array = np.asarray(image)
    # Normalize the image
    normalized_image_array = (image_array.astype(np.float32) / 127.0) -
1

    # Load the image into the array
    data[0] = normalized_image_array

    # run the inference
```

```
    prediction = model.predict(data)
    return np.argmax(prediction), prediction # return position of the
highest probability



#

import streamlit as st
from img_classification import teachable_machine_classification
from PIL import Image
import numpy as np
import pandas as pd

col1Head,col2Head,col3Head = st.columns([1,2,1])
with col2Head:
  imageheader = Image.open("satralogo.png")
  st.image(imageheader,width=500)
st.title("Image Classification for White Blood Types using CNN")
st.header("Select an Optimizer, Activation Function and Number of
Epochs to get accuracy")
st.write("")
st.write("")

model_list = ["ResNet50", "Normal"]
activation_list = ["RELU", "ELU", "SELU", "TANH"]
opti_list = ["ADAM", "NADAM","RMSProp", "ADAGRAD",]
epoch = [30, 50]

train30 =   {
     "ADAM":[98.35,97.65,96.69,95.04],
     "NADAM":[98.02,97.87,96.88,95.18],
     "RMSProp":[98.54,97.92,96.19,95.60],
     "ADAGRAD":[76.53,78.85,76.65,77.77],
    }
train50 =   {
     "ADAM":[99.32,98.63,97.48,96.22],
     "NADAM":[98.94,98.79,98.04,78.17],
     "RMSProp":[98.8,98.75,97.91,97.07],
     "ADAGRAD":[86.45,84.96,79.91,80.09],
```

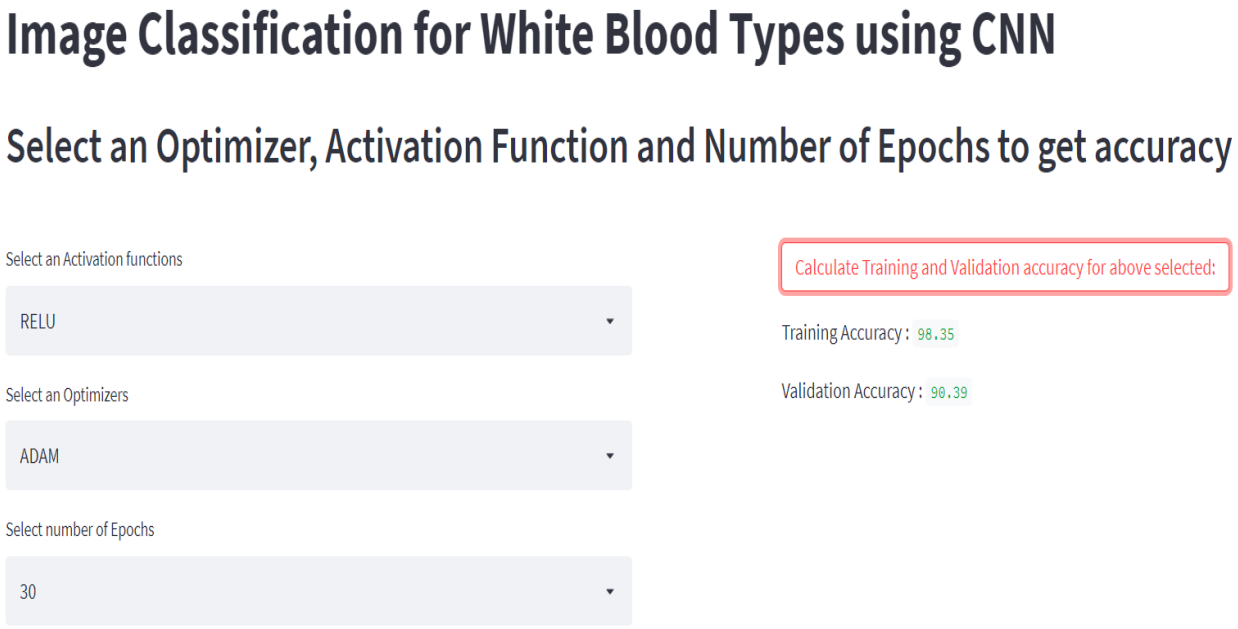
```
    }
valid30 =    {
      "ADAM":[90.39,93.11,89.54,85.11],
      "NADAM":[82.6,73.3,92.25,81.19],
      "RMSProp":[91.2,86.97,89.14,79.98],
      "ADAGRAD":[79.93,81.39,79.13,80.49],
    }
valid50 =  {
      "ADAM":[90.39,93.11,89.54,85.11],
"NADAM":[82.6,73.3,92.25,81.19],
      "RMSProp":[91.2,86.97,89.14,79.98],
      "ADAGRAD":[79.93,81.39,79.13,80.49],
    }

col1,col2,col3 = st.columns([1,0.2,1])
colTab1,coldTabMid, colTab2 = st.columns([1,0.2,1])

with col1:
  actv_selected =st.selectbox("Select an Activation functions",
activation_list)
  optm_selected =st.selectbox("Select an Optimizers", opti_list)
  epoch_selected =st.selectbox("Select number of Epochs", epoch)

if 'count1' not in st.session_state:
  st.session_state.count1 = 0
if 'count2' not in st.session_state:
  st.session_state.count2 = 0

def calcTrainAcc(actv_selected,optm_selected,epoch_selected):
    if actv_selected == "RELU":
        index = 0
    if actv_selected == "ELU":
        index = 1
    if actv_selected == "SELU":
        index = 2
    if actv_selected == "TANH":
        index = 3

    if(epoch_selected == 30):
```

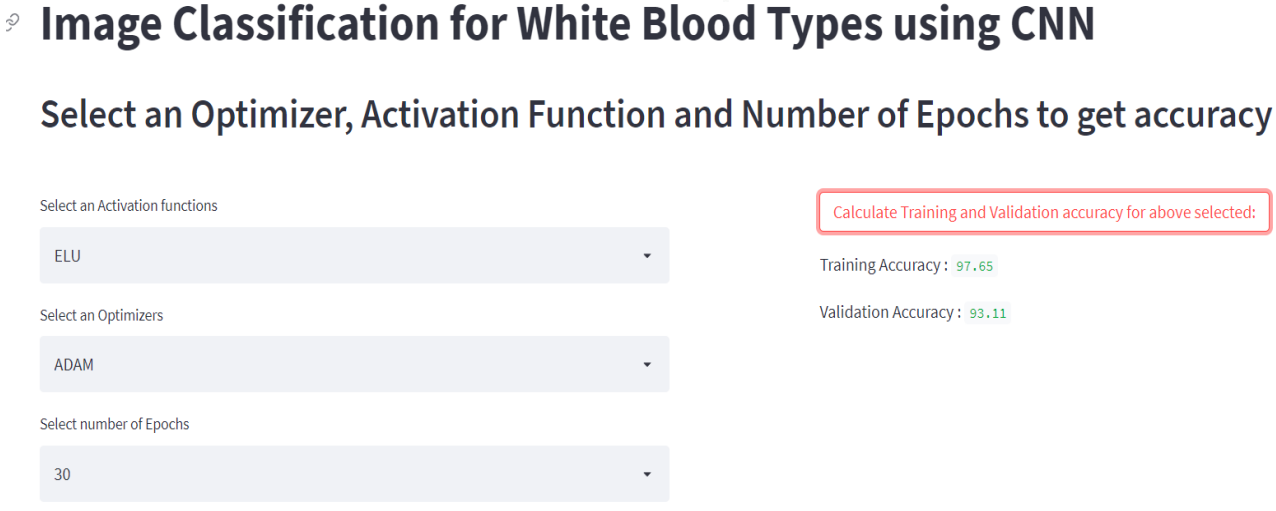
```
        st.session_state.count1 =  train30[optm_selected][index]
        st.session_state.count2 =  valid30[optm_selected][index]

    if(epoch_selected == 50):
        st.session_state.count1 =  train50[optm_selected][index]
        st.session_state.count2 =  valid50[optm_selected][index]

with col3:
     st.button('Calculate Training and Validation accuracy for above
selected: ',
on_click=calcTrainAcc,args=(actv_selected,optm_selected,epoch_select
ed))
     st.write("Training Accuracy : ",st.session_state.count1)
     st.write("Validation Accuracy : ",st.session_state.count2)

with colTab1:
    st.write("")
    st.write("")
    st.write("")

    st.write("30 epochs training accuracy")
    df = pd.DataFrame(train30,
    index = ('RELU','ELU','SELU','TANH'))
    st.table(df)

    st.write("50 epochs training accuracy")
    df = pd.DataFrame(train50,
    index = ('RELU','ELU','SELU','TANH'))
    st.table(df)

with colTab2:
    st.write("")
    st.write("")
    st.write("")

    st.write("30 epochs validation accuracy")
    df = pd.DataFrame(valid30,
    index = ('RELU','ELU','SELU','TANH'))
    st.table(df)
```

```
    st.write("50 epochs validation accuracy")
    df = pd.DataFrame(valid50,
    index = ('RELU','ELU','SELU','TANH'))
    st.table(df)

st.write("")
st.write("")
st.header("Upload your image below for classifying the type of WBC")
st.write("")
model_selected = st.selectbox("Select the Model", model_list)
st.write("Drag any image from bottom to classify")

col1Image,col2Image,col3Image,col4Image = st.columns([1,1,1,1])
with col1Image:
  imageheader = Image.open("eos-both.jpg")
  st.image(imageheader,width=200)
with col2Image:
  imageheader = Image.open("eos-model1(2).jpg")
  st.image(imageheader,width=200)
with col3Image:
  imageheader = Image.open("mono-model2.jpg")
  st.image(imageheader,width=200)
with col4Image:
  imageheader = Image.open("neut-both.jpg")
  st.image(imageheader,width=200)

st.write("")
st.write("")

uploaded_file = st.file_uploader("Choose a WBC blood smear sample in
JPG form ...")

if uploaded_file is not None:
    image = Image.open(uploaded_file)
    st.write(uploaded_file)
    st.image(image, caption='Uploaded Image',width=300)
    st.write("")
```

```
        if model_selected == "ResNet50":
          label, resArr = teachable_machine_classification(image, 'model.h5')
        if model_selected == "CNN":
          label, resArr = teachable_machine_classification(image,
      'model2.h5')
        lis = resArr.tolist()


        st.write(lis[0])
        newLis = []
        for x in lis[0]:
            x = (x - np.min(lis[0])) / (np.max(lis[0]) - np.min(lis[0]))
            newLis.append(x)
        st.table(newLis)
        st.write("Class label:\t ", label)
        if label == 0:
          st.write("WBC type is EOSINOPHIL")
        if label == 1:
          st.write("WBC type is LYMPHOCYTE")
        if label == 2:
          st.write("WBC type is MONOCYTE")
        if label == 3:
          st.write("WBC type is NEUTROPHIL")


        if label:
    st.success('Blood smear image is classified and WBC type is returned
successfully')
```

CHAPTER 4
SNAPSHOTS





30 epochs training accuracy

	ADAM	NADAM	RMSProp	ADAGRAD
RELU	98.3500	98.0200	98.5400	76.5300
ELU	97.6500	97.8700	97.9200	78.8500
SELU	96.6900	96.8800	96.1900	76.6500
TANH	95.0400	95.1800	95.6000	77.7700

30 epochs validation accuracy

	ADAM	NADAM	RMSProp	ADAGRAD
RELU	90.3900	82.6000	91.2000	79.9300
ELU	93.1100	73.3000	86.9700	81.3900
SELU	89.5400	92.2500	89.1400	79.1300
TANH	85.1100	81.1900	79.9800	80.4900

50 epochs training accuracy

	ADAM	NADAM	RMSProp	ADAGRAD
RELU	99.3200	98.9400	98.8000	86.4500
ELU	98.6300	98.7900	98.7500	84.9600
SELU	97.4800	98.0400	97.9100	79.9100
TANH	96.2200	78.1700	97.0700	80.0900

50 epochs validation accuracy

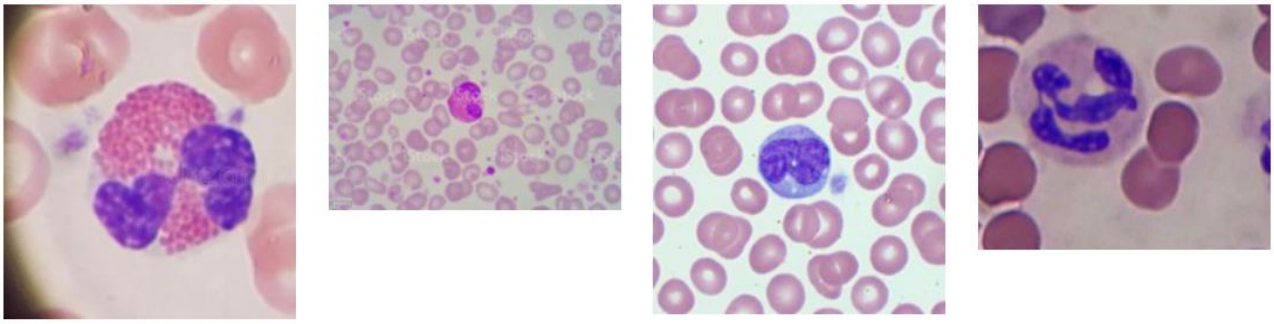
	ADAM	NADAM	RMSProp	ADAGRAD
RELU	90.3900	82.6000	91.2000	79.9300
ELU	93.1100	73.3000	86.9700	81.3900
SELU	89.5400	92.2500	89.1400	79.1300
TANH	85.1100	81.1900	79.9800	80.4900

Upload your image below for classifying the type of WBC

Select the Model

ResNet50

Drag any image from bottom to classify



Choose a WBC blood smear sample in JPG form ...

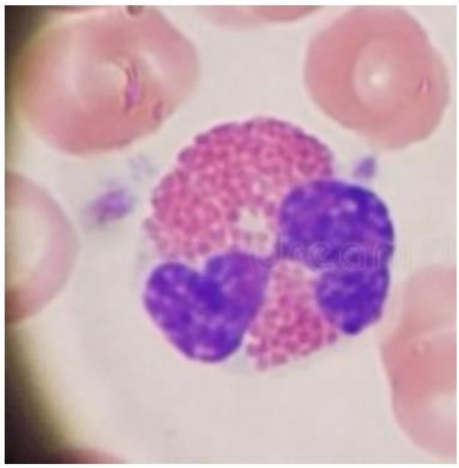
Drag and drop file here

Limit 200MB per file

Browse files

e20cbd7afc85e5cab72d0804febc14ed0775fbc77bee306d6203994f.jpg 8.8KB

```
UploadedFile(id=1, name='e20cbd7afc85e5cab72d0804febc14ed0775fbc77bee306d6203994f.jpg', type='image/jpeg', size=9056)
```



Uploaded Image

```
[
  0 : 1
  1 : 7.633455406289133e-35
  2 : 1.8153510120104732e-23
  3 : 2.3646337117343857e-21
]
```

	0
0	1.0000
1	0.0000
2	0.0000
3	0.0000

Class label: `0`

WBC type is EOSINOPHIL

DATASET	TRAINING ACCURACY	VALIDATION ACCURACY
1st dataset (5 epochs)	99.60	74.92
2nd dataset (10 epochs)	99.38	85.71

Table 1: Accuracy using ResNet

CHAPTER 5
CONCLUSION

By using a powerfully trained model called ResNet50, we are able to achieve an accuracy of 99%, whilst the accuracy of the base paper is 98.55%. We have also used different optimizers such as Adagrad, Adam, Nadam, SGD and RMSprop, along with different activation functions such as tanh, relu, selu and elu. This has helped in understanding the efficient way to classify White Blood Cell, which in turn will be useful for understanding the different White Blood Cell for the creation of vaccination.

CHAPTER 6
REFERENCES

References:

1. M. Saraswat, K. Arya, Automated microscopic image analysis for leukocytes identification: a survey, Micron 65 (2014) 20–33.

2. D.M. Sabino, L. Da Fontoura Costa, E. Gil Rizzatti, M. Antonio Zago, A texture approach to leukocyte recognition, Real-Time Imaging 10 (4) (2004) 205–216.

3. P. Ghosh, D. Bhattacharjee, M. Nasipuri, Blood smear analyzer for white blood cell counting: a hybrid microscopic image analyzing technique, Appl. Soft Comput. 46 (2016) 629–638.

4. R.B. Hegde, K. Prasad, H. Hebbar, B.M. Singh, Comparison of traditional image processing and deep learning approaches for classification of white blood cells in peripheral blood smear images, Biocybern. Biomed. Eng. 39 (2) (2019) 382–392.

5. A. Patil, M. Patil, G. Birajdar, White blood Cells image classification using deep learning with Canonical correlation analysis, IRBM (2020).

CHAPTER 7
BASE PAPER

Classification of White blood cell using Convolution Neural Network

Gaurang Panchal, *Student Member, IEEE*, Debasis Samanta, *Senior Member, IEEE*, Ashok Kumar Das, *Senior Member, IEEE*, Neeraj Kumar, *Senior Member, IEEE*, Kim-Kwang Raymond Choo, *Senior Member, IEEE*

***Abstract*—**The human immune system consists of White Blood Cells that are responsible for fighting of disease pathogens. In the field of medical imagining, white blood cells is of great importance. Analysis of white blood cells can be helpful to medical experts in many of the cases such as viral infection or cancer infection. In this paper, the classification of White Blood Cell using a Convolution Neural Network (CNN) is proposed. The proposed approach is able to classify the type of cell in much less epochs/time than other approaches. The performance of the proposed approach is evaluated on Kaggle dataset. The overall accuracy obtained from the proposed approach is 98.55%.

Keywords: Deep learning Feature selection White Blood Cell classification Convolution Neural Network (CNN)

I. INTRODUCTION

Human blood contains different cellular entities that are responsible for various functions of body such as carrying oxygen, regeneration, clotting, immunity, etc. The four major components of blood are plasma, red blood cells (RBC), platelets, and white blood cells (WBC). WBCs are responsible for fighting with foreign pathogens and protect the body from infection. In today's times, when an uphill battle against Coronavirus [24] is going on, producing antibodies against such types of deadly virus is critical task. WBCs are an integral part of our fight against such deadly infection. The classification of WBC in blood is a most important task. WBC consists of cytoplasm, cell wall, and a nucleus. There are five types of WBC namely, Monocyte, Eosinophil, Basophil, Lymphocytes and Neutrophil (see Fig. 1). All these types of WBC have a function to perform in the fight against virus, bacteria, fungus and all other kinds of infections in our body. The ratio of RBC and WBC in a normal human body is 600:1 [5]. This implies that over 600 RBCs a single WBC would be found. Further, the identification of WBC type in a blood smear sample is even more difficult. In the recent years, deep learning techniques are widely used in medical field. This fact motivate us to utilizes these techniques in the classification of WBC in the given blood sample. In this paper, a Convolution Neural Network (CNN) based classification technique is constructed to determine the type of WBC. The main contributions of this paper are as follows:

1. A convolution neural network based classification technique is proposed to identify the types of WBC in the given blood sample.

2. The proposed technique is evaluated on Blood Cell Count and Classification Dataset.

3. The performance of the proposed technique is compared with four well-known techniques in terms of accuracy, specificity, and sensitivity.

The structure of this paper is organised as follows. Section 2 discusses research works done in classifying type of WBC from blood smear images. In Section 3, the proposed approach is discussed in detail. Experimental results and discussions are mentioned in Section 4. The concluding remarks are drawn in Section 5.

II. RELATED WORK

Many approaches have been made for classifying type of WBC in blood smear images. Most popular ones among these are based on either Machine Learning, Deep Learning, Fuzzy Logic or a mix of all [6,7]. Surveys done by Saraswat et al. [6] in 2014 and by Kumar et al. [7] in 2020 reveal that mostly locally available datasets are used by the research-works which makes the algorithm tailor-made classifier for the dataset. Fig. 2 shows the distribution of types of datasets used in the literature. As depicted in the figure, most studies have used locally available datasets and some research-works have not revealed their datasets. The proposed work used a publicly available dataset for drawing comparisons with the previous works. Classification using machine learning can be done through Support Vector Machine (SVM) [2] and Bayesian classifiers [8]. These classifiers do not need high volumes of training data. However, some pre-processing techniques are required. Hegde et al. [11] used both traditional image classification and Convolution Neural Network (CNN) for the classification. They also compared the results of the two approaches.

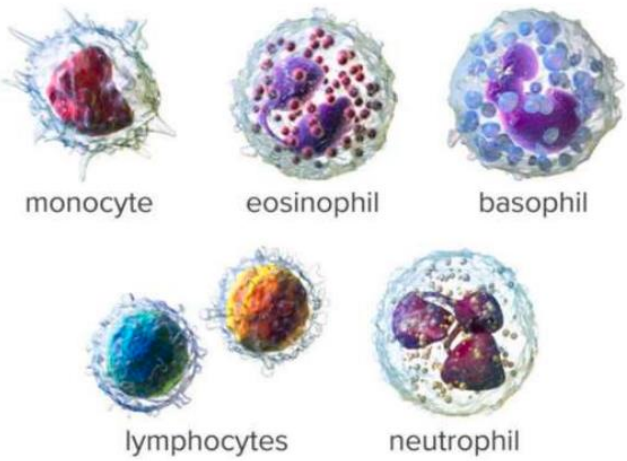


Fig. 1. Five types of White Blood Cell [5].

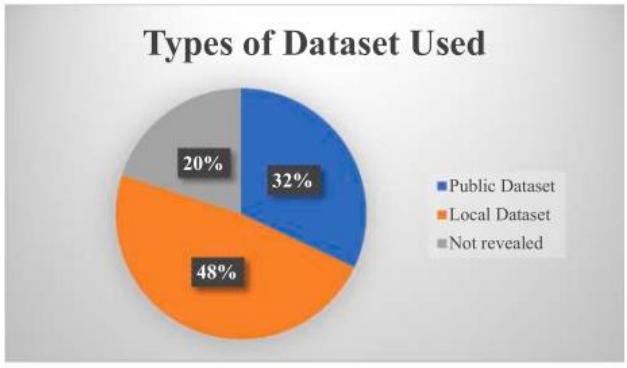


Fig. 2. Distribution of types of Datasets used by various researchers.

While the results obtained were similar; traditional method relied heavily on segmentation and feature extraction. CNN is not dependent on these parameters. Nevertheless, CNN approach requires a large amount of labelled data and traditional image processing has no such dependencies. Also, traditional image processing approaches are easy to implement for image classification. Singh et al. [12] proposed a CNN model for WBC classification. CNN model was trained using 200 epochs. Ma et al. [13] combined ResNet and DC-GANs to construct a classifier with improved accuracy. Sengür et al. [14] proposed a WBC classification approach in which first Region-Of-Interest (ROI) is extracted from blood smear images in HSV space. Afterwards, feature extraction was done using ResNet50. The feature selection was performed using principal component analysis (PCA). The classification job was carried out by using Long Short-Term Memory (LSTM) classifier. This approach attained the overall accuracy of 85.7%. Patil et al. [15] used Canonical Correlation Analysis (CCA) [16] to enhance accuracy. They used the concepts of both CNN and RNN. Both [14] and [15] used the same dataset in their research work. Wijesinghe et al. [29] used k-means clustering for WBC nuclei segmentation from images. Thereafter, VGG16 classifier [30] was used for putting WBC nuclei into their respective class. Another CNN model proposed by Jung et al. [31]. It was first trained over a local dataset and then it was used to classify the images from dataset reference [3]. Ucar proposed a Deep learning model [32] based on ShuffleNet [33]. The overall accuracy obtained from this model was 97%. Karthikeyan et al. [34] proposed a CNN for detection and classification of WBC using Euler's Jenks Optimisation. The elimination of RBCs from the peripheral blood samples was done by using Jenks Optimised pooling [3].

Apart from deep learning techniques, Fuzzy Systems have also been used in the literature. Ghosh et al. [9] proposed a fuzzy system to count WBCs in blood smear images. Rawat et al. [10] proposed an adaptive neuro-fuzzy system to classify WBC type on an online dataset and applied Chan-Vese [28] algorithm for segmentation of WBC nucleus from blood smear images. The summarization of related work done in the field of classification of WBC is depicted in Table1.

From this literature review, the authors concluded following observations:

1. Both traditional image processing and Neural Networks (NN) have been used to classify WBC types.
2. For WBC classification problem, Convolution Neural Network (CNN) can be constructed to give good results.
3. Time taken to train CNNs for WBC classification and number of epochs is still very high. The above-mentioned facts motivated us to design a classification technique using a CNN.

III. PROPSED APPROACH

In this paper, a Convolution Neural Network (CNN) is constructed to classify the type of WBC in blood smear images obtained from Blood Cell Count and Detection (BCCD) dataset. A CNN is the designed neural network itself "learns" how to classify images from the repeated "epochs" of training on a large volume of data. In this section, the designed neural network and its layers are described in the succeeding subsections.

3.1. Convolution Neural Network

A CNN is a type of ANN (Artificial Neural Network) in which convolution is done in order to extract vital features from input images as shown in Fig. 3. Convolution operation is done using filter (also called kernels) which can perform operations such as stride, edge detection, sharpen, blur, etc. In the CNN, all the convolution layers perform stride operation. In stride operation, the amount of movement between applications of filter to input image is adjusted. It is used for down sampling the input image. The output from a convolution layer is a feature map, which is then given to a pooling layer. Pooling layer reduces the parameters when the image size is huge. After that feed-forward layer is set up in which all the neurons of one layer are fully connected to other layer; which is why this layer is called fully-connected or 'fc' layer. Lastly, classification layer is present which gives a probability of the presence of an object in the given input image. A dropout layer is also sometimes added to the network to avoid the problem of overfitting of the neural network.

3.2. Methodology

Figure 4 shows the proposed approach for classification of WBCs in blood smear images. Convolution layer followed by normalisation, leaky Relu and pooling layer is repeated four times in the proposed approach. A dropout layer is attached in the last in order to prevent overfitting problem. A fully-connected layer and softmax layer are added afterwards. In the

end, the classification layer is used to classify WBC type from the features extracted by all the previous layers.

Table 1
Brief Literature Review in chronological order.

Year	Ref.	Title of the Research Work	Approach
2004	[8]	A texture approach to leukocyte recognition	Bayesian Classifier
2016	[9]	Blood smear analyzer for white blood cell counting: A hybrid microscopic image analyzing technique	Fuzzy Logic, Random Forest
2018	[2]	Fast and robust segmentation of white blood cell images by self-supervised learning	Support Vector Machine (SVM)
2019	[11]	Comparison of traditional image processing and deep learning approaches for classification of white blood cells in peripheral blood smear images.	Neural Network, Autoencoders, Convolution Neural Network
2019	[14]	White blood cell classification based on shape and deep features.	Long Short-Term Memory Network
2019	[31]	W-net: a CNN-based architecture for white blood cells image classification	Convolutional Neural Network
2020	[12]	Blood Cell Types Classification Using CNN	Convolution Neural Network
2020	[13]	Combining DC-GAN with ResNet for blood cell image classification	Deep Convolution Generative Adversarial Network
2020	[15]	White Blood Cells Image Classification Using Deep Learning with Canonical Correlation Analysis	Convolutional Neural Network and Recurrent Neural Network
2020	[29]	Fully Automated Detection and Classification of White Blood Cells	K-means clustering and VGG-16
2020	[31]	Deep Learning Approach to Cell Classification in Human Peripheral Blood	ShuffleNet
2020	[34]	White blood cell detection and classification using Euler's Jenks optimized multinomial logistic neural networks	Jenks Optimization, Convolution Neural Network

The detail description of layers and functions used in the proposed approach is given below.

1. Input Layer: It is the layer where image input is received and the imageInputLayer function is utilized. The size of image can be specified by InputSize argument.

2. Convolutional Layer: convolution2dLayer function is utilized for creating 2-D convolutional layer. This layers automatically learns features while sliding on image in a definite way. To set the con volutional filters, filterSize argument is used. The movement of filter horizontally and vertically happens in a step and its size is called as stride. Dilation Factor argument can be used for determining step size. Convolutional operation results in feature maps. Padding can be used to increase output size of layer. The output size is shown as an integer. Neurons in a feature map are the number of neurons in Convolutional layer. Number of layers can be defined depending upon the proposed design.

3. Batch Normalization Layer: The batch normalization layer is responsible for normalizing the propagation gradients and activations. This layer can alter the speed of training. It can be used for optimization. For this layer, batchNormalizationLayer function is utilized.

4. ReLU Layer: This layer is responsible for applying a thresholding operation to each input layer. To create this layer, ReLU Layer function is used.

5. Average and Max Pooling Layers: The task of these layers is down sampling. It can help in increasing number of filters in deeper convolutional layers without increasing the computation. Redundant information and spatial size is reduced. For these layers, maxPooling2dLayer function is used in MATLAB.

6. Dropout Layer: This is used to prevent overfitting.

7. Fully Connected Layers: This layer is connecting all the neurons to the neurons of previous layers and is responsible for merging the features learned by previous layers and the larger patterns can thus by identified. To employ this layer, fullyConnectedLayer function is utilized.

8. Output Layers: These consists of softmax and classification layer. The softmax layer assigns probabilities to each class and sum of these probabilities will be equal to one. Functions used for creating these layers are softmax and classificationLayer.

Training options such as initial learning rate, optimizer used, batch size, maximum number of epochs etc. are specified. Afterwards, training of the neural network is done and then trained neural network is validated and accuracy is measured. All the layers and their parameters are mentioned in Table2

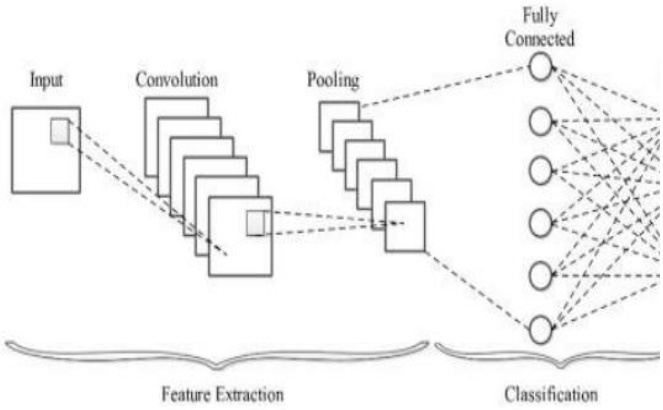


Fig. 3. CNN-based classification approach [17].

IV. EXPERIMENTAL RESULTS AND DISCUSSIONS

The performance of the proposed classification approach is validated on the publicly available dataset.

4.1. Experimental setup

In the designed architecture, initial learning rate is set to 0.001. Gradient decay factor is 0.10 while optimizer used has been set to Adam. Maximum number of epochs is set to 20. Simulation has been carried out on Intel Core i7 processor, 16 GB of RAM, 1 TB FireCuda hard disk and NVIDIA GeForce GTX 1060 6 GB graphics card. The dataset has been taken from publicly available source [4]. This dataset consisted of 364 colored images which were taken from a microscope of dyed WBC. 1. For this work, some blood datasets were considered [1–4]. Some research works are based upon the locally available datasets while others use a publicly available dataset. From these datasets, has been chosen as dataset for training and validation purposes. This dataset was chosen in this work for the following reasons:

1. [4] is a balanced dataset containing almost equal number of data points in all the four classes, unlike [1].

2. Availability of more data points in [4] as compared to [2] and [3] further enhanced our interest in using it.

This dataset consists of 12,500 images of blood cells. These are classified into four different categories such as Eosinophil, Lymphocytes, Monocytes and Neutrophils. It has almost equal distribution of images i.e., approximately 3000 in each class. Number of images of different types of WBCs are shown in Table 3.

For pre-processing of data, following operations were carried out.

1. Cleaning of data: In the dataset, it was found that it has only 3 images of Basophil type WBC, which are negligible as compared to other types. So, this type was removed for this study for training the network.

2. Resizing of images: All the images in dataset were converted to 128*128 size.

3. Reduction of noise: Median filter was used for salt and pepper type noise reduction. This filter was applied to each image with neighbourhood size 2*2 and zero padding.

4. Augmentation: Augmentation techniques are used to increase the size of dataset using operations of rotations, flips and shears which resulted in total number of 12,444 images which are shown in Table 4.

5. Train-test division: The images in dataset are divided in the ratio of 80–20 for training and testing of the proposed model.

It is to noted here that dataset available online has two folders and in one of these folders steps 1–4 have already been implemented. This makes the dataset with a total of 12,444 images of WBCs ready for the classification task.

Some screenshots of training of designed CNN have been captured and shown in Fig. 5.

It can be seen in Fig. 5(b), a sudden fall in accuracy is observed after seventh epoch due to Dropout Layer for prevention of the overfitting problem. In Fig. 5(c), the training of neural network is about to be completed which gets completed in Fig. 5(d). Batch Size was set to be 128 and total time taken by the proposed CNN model was around 8 min

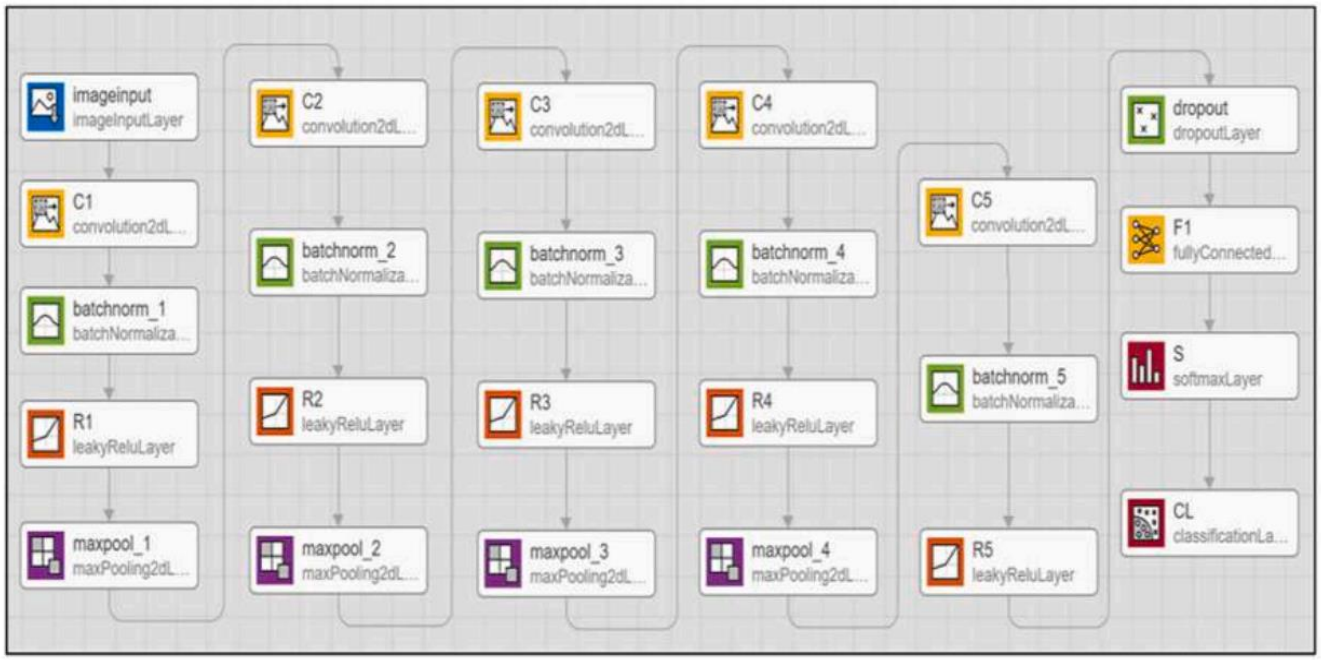


Fig. 4. Proposed CNN for WBC classification.

Table 2
Layers and their parameters of the proposed CNN model.

Layer number	Layer detail	Activation
1	Input of image	128*128*3
2	Convolution Layer 1	128*128*6
3	Batch Normalization Layer 1	128*128*6
4	Leaky ReLU 1	128*128*6
5	Max Pooling Layer 1	64*64*6
6	Convolution Layer 2	60*60*16
7	Batch Normalization Layer 2	60*60*16
8	Leaky ReLU 2	60*60*16
9	Max Pooling Layer 2	30*30*16
10	Convolution Layer 3	26*26*64
11	Batch Normalization Layer 2	26*26*64
12	Leaky ReLU 3	26*26*64
13	Max Pooling Layer 3	13*13*64
14	Convolution Layer 4	9*9*128
15	Batch Normalization Layer 4	9*9*128
16	Leaky ReLU 4	9*9*128
17	Max Pooling Layer 4	4*4*128
18	Convolution Layer 5	1*1*512
19	Batch Normalization Layer 5	1*1*512
20	Leaky ReLU 5	1*1*512
21	Dropout Layer	1*1*512
22	Fully Connected Layer	1*1*4
23	Softmax Layer	1*1*4
24	Classification Layer Output	–

4.2. Performance evaluation metrics

The performance of a classifier is tested through the performance metrics such as accuracy, precision, specificity, recall or sensitivity [21] and F1-score [22]. The mathematical formulation of these performance measures is given below [23]:

$$Precision = \frac{TruePositive}{(TruePositive + FalsePositive)} \times 100 \quad (1)$$

$$Recall = \frac{True\ Positive}{(True\ Positive + False\ Negative)} \times 100 \quad (2)$$

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + False\ Positive + True\ Negative + False\ Negative} \times 100 \quad (3)$$

$$Specificity = \frac{True\ Negative}{True\ Negative + False\ Positive} \times 100 \quad (4)$$

$$F1 = 2 \times \frac{Precision \times Recall}{(Precision + Recall)} \quad (5)$$

All these formulae require True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). These values can be calculated using a confusion matrix. Fig. 6 depicts the confusion matrix for binary classification. Confusion matrix of the four classifications (or classes) of WBC types is shown in Fig. 7. The rows of confusion matrix show the predicted output class and columns indicate true output class. Red colored columns indicate the mismatches while green colored boxes indicate rightly matched labels. Ground truth of the blood smear images are labelled as target class while the predicted

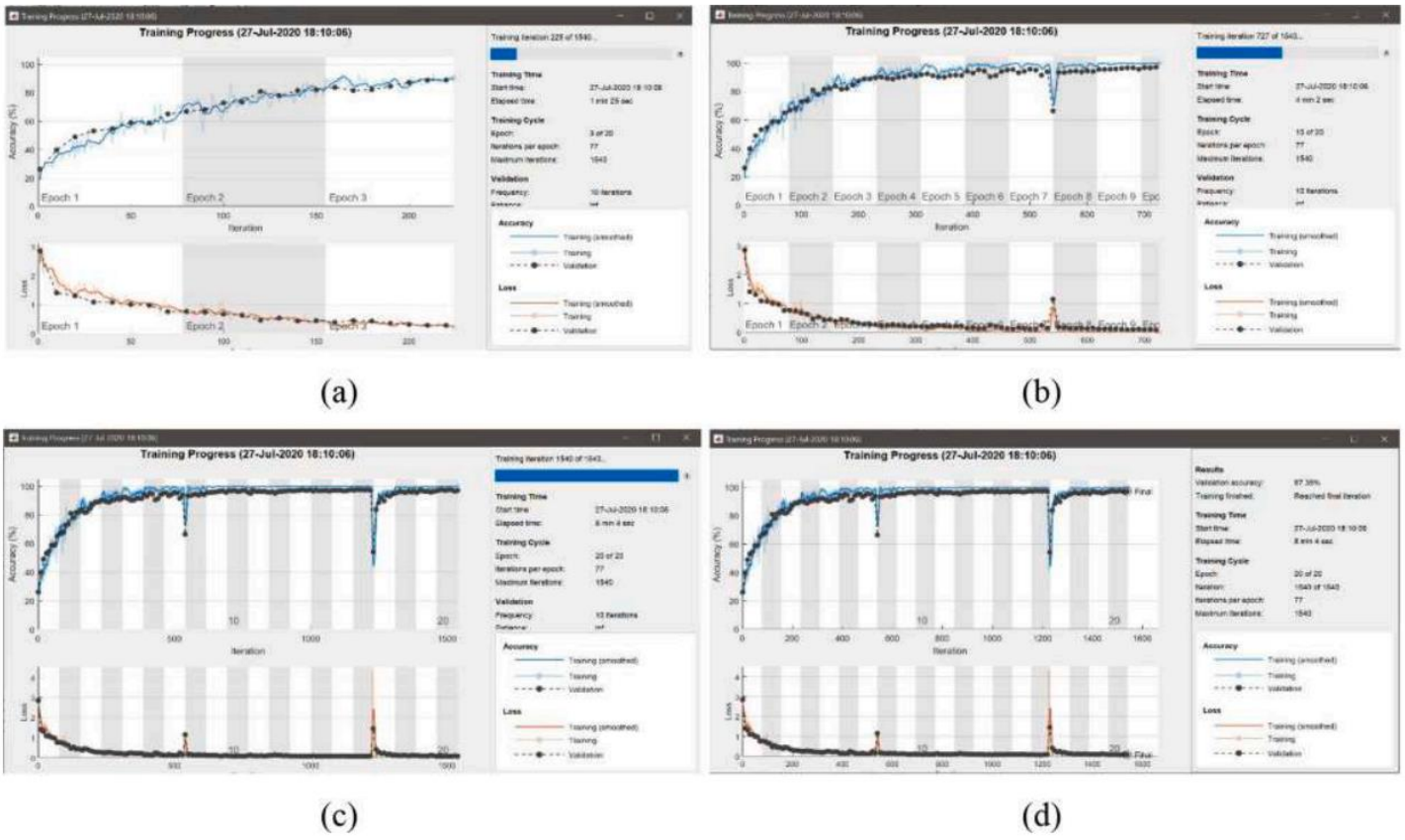


Fig. 5. Screen grabs of (a) Initial training of neural network; (b) In mid-way of training; (c) Finalizing training of network and (d) Training complete.

output class is denoted by label output class in confusion matrix. Maximum mismatches are found in case of labelling Neutrophils as 24 images of Neutrophils have been incorrectly labelled as Eosinophils.

Using the confusion matrix, the above-mentioned parameter metric values have been calculated and mentioned in Table 5. While Monocytes have been classified with the highest accuracy and F1-score; Neutrophil classification is mainly responsible for bringing overall accuracy rates low.

A graph has been plotted to show sensitivity, specificity and accuracy obtained for the four classes in Fig. 8. First bar shows accuracy, second bar indicates specificity values and third bar is for sensitivity measures of the four types of WBC types.

Figure 9 depicts the classification accuracy obtained for the designed CNN and the existing techniques. It can be observed that the designed CNN model outperforms other works in terms of accuracy. Table 6 shows the performance evaluation of proposed approach and the compared techniques over BCCD dataset.

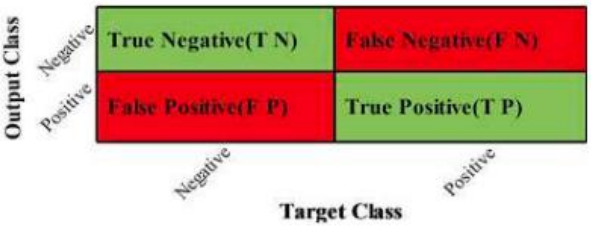


Fig. 6. Confusion Matrix for binary classification.

V. CONCLUSION

In this paper, a convolutional neural network-based classification technique was proposed to classify the type of WBC in blood smear images. The overall accuracy of the proposed CNN is 98.55% in classifying type of WBC, which is compared with other classifiers. The proposed CNN was designed from scratch and it has been able to achieve 98.55% accuracy in about 20 epochs. The proposed CNN based classification technique can be used to solve other classification problems such as RBC classification [26]. Transfer learning approaches can also be applied to attain higher accuracy rates. Though the proposed CNN model gives an overall accuracy rate above 98.55%, it is trained and tested

over single celled images which is quite unlikely to be found in a blood sample. The authors intend to apply the model on multi-celled, overlapped and occluded WBC images in their future work.

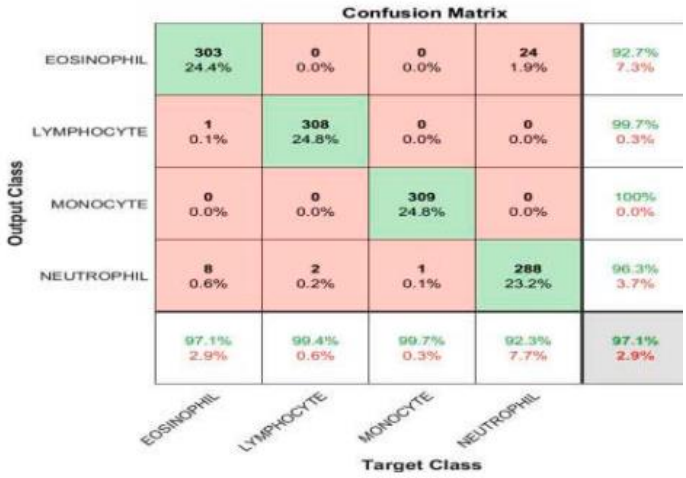


Fig. 7. Confusion matrix for all the four classes of WBCs.

Table 5
Precision, Recall, Specificity, F1-score and Accuracy obtained from propos approach.

Label	Precision	Recall	Specificity	F1-score	Accurac
Eosinophil	92.66	97.12	97.42	94.84	97.35
Lymphocyte	99.67	99.35	99.89	99.51	99.75
Monocyte	100	99.68	100	99.84	99.92
Neutrophil	96.32	92.31	98.82	94.27	97.19

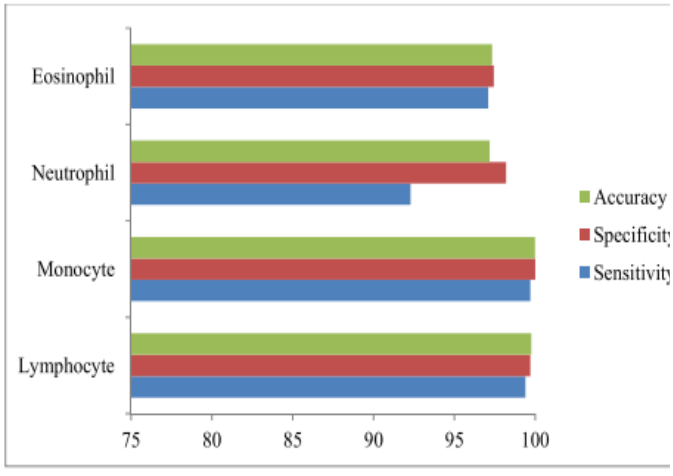


Fig. 8. Accuracy, Specificity and Sensitivity of the four classes.

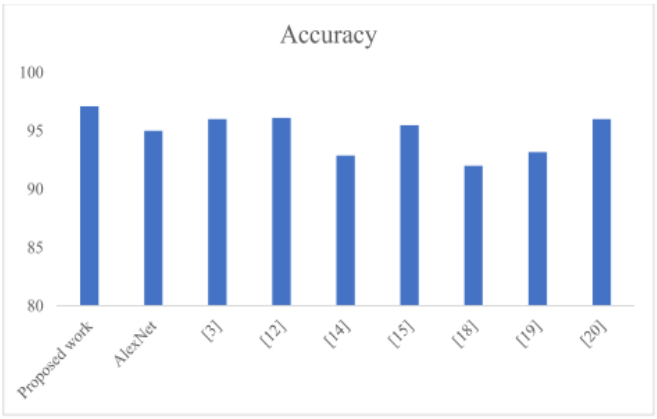


Fig. 9. Comparison of Proposed CNN with previous works.

Table 6
Performance comparison of different classification techniques on BCCD dataset.

	Number of epochs	Accuracy	Precision	Recall	Specificity	F1-measure
[14]	50	92.89	62.36	68.25	92.900	64.74
[15]	100	95.48	91.03	90.96	96.99	90.96
[18]	20	92	86	86	95	86
[25]	50	95.70	91.43	91.60	97.12	91.40
Ours	20	98.55	97.16	97.11	99.03	97.11

CRediT authorship contribution statement

Ashish Girdhar: Conceptualization, Methodology, Software, Writing - review & editing. **Himani Kapur:** Data curation, Software, Writing - original draft. **Vijay Kumar:** Investigation, Supervision, Validation, Visualization.

Declaration of Competing Interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] A. Acevedo, A. Merino, S. Alf´erez, A. ´ Molina, L. Boldú, J. Rodellar, A dataset of microscopic peripheral blood cell images for development of automatic recognition systems, Data Brief 30 (2020), 105474.

[2] X. Zheng, Y. Wang, G. Wang, J. Liu, Fast and robust segmentation of white blood cell images by self-supervised learning, Micron 107 (2018) 55–71.

[3] S.H. Rezatofighi, H. Soltanian-Zadeh, Automatic recognition of five types of white blood cells in peripheral blood, Comput. Med. Imaging Graph. 35 (4) (2011) 333–343.

[4] P. Mooney, Blood cell images,

https://www.kaggle.com/paultimothymooney/blood-cells, Accessed 25/07/2020.

[5] White blood cells: function, ranges, types, and more, https://www.medicalnewstoday.com/articles/327446. Accessed 26/07/2020.

[6] M. Saraswat, K. Arya, Automated microscopic image analysis for leukocytes identification: a survey, Micron 65 (2014) 20–33.

[7] K. Anilkumar, V. Manoj, T. Sagi, A survey on image segmentation of blood and bone marrow smear images with emphasis to automated detection of leukemia, Biocybern. Biomed. Eng. 40 (4) (2020) 1406–1420.

[8] D.M. Sabino, L. Da Fontoura Costa, E. Gil Rizzatti, M. Antonio Zago, A texture approach to leukocyte recognition, Real-Time Imaging 10 (4) (2004) 205–216.

[9] P. Ghosh, D. Bhattacharjee, M. Nasipuri, Blood smear analyzer for white blood cell counting: a hybrid microscopic image analyzing technique, Appl. Soft Comput. 46 (2016) 629–638.

[10] J. Rawat, A. Singh, H.S. Bhadauria, J. Virmani, J.S. Devgun, Leukocyte classification using adaptive Neuro-Fuzzy inference system in microscopic Blood images, Arab. J. Sci. Eng. 43 (12) (2017) 7041–7058.

[11] R.B. Hegde, K. Prasad, H. Hebbar, B.M. Singh, Comparison of traditional image processing and deep learning approaches for classification of white blood cells in peripheral blood smear images, Biocybern. Biomed. Eng. 39 (2) (2019) 382–392.

[12] I. Singh, N.P. Singh, H. Singh, S. Bawankar, A. Ngom, Blood cell types classification using CNN, in: International work-conference on bioinformatics and biomedical engineering, Springer, Cham, 2020, pp. 727–738.

[13] L. Ma, R. Shuai, X. Ran, W. Liu, C. Ye, Combining dc-gan with resnet for blood cell image classification, Med. Biol. Eng. Comput. 58 (6) (2020) 1251–1264.

[14] A. Sengur, Y. Akbulut, U. Budak, Z. Comert, White blood CELL classification based on shape and deep features. 2019 International Artificial Intelligence and Data Processing Symposium (IDAP), 2019.

[15] A. Patil, M. Patil, G. Birajdar, White blood Cells image classification using deep learning with Canonical correlation analysis, IRBM (2020).

[16] H. Hotelling, Relations between two sets of variates, Biometrika 28 (3/4) (1936) 162–190.

[17] E.J. Rhee, A deep learning approach for classification of cloud image patches on small datasets, J. Inf. Commun. Converg. Eng. 16 (3) (2018) 173–178.

[18] H.E. Mohamed, H.W. El-Behaidy, G. Khoriba, J. Li, Improved white blood cells classification based on pre-trained deep learning models, J. Commun. Softw. Syst. 16 (1) (2020) 37–45.

[21] D.L. Olson, D. Delen, Performance evaluation for predictive modelling, Adv. Data Min. Tech. (2008) 137–147.

[22] Y. Sasaki, The truth of the F-measure, https://www.toyota-ti.ac.jp/Lab/ Denshi/COIN/people/yutaka.sasaki/F- measure- YS-26Oct07.pdf/, 2007, pp. 1–5, Accessed 14-08-2020.

[23] P.P. Banik, R. Saha, K. Kim, An automatic Nucleus segmentation and Cnn model based classification method of white blood cell, Expert Syst. Appl. 149 (2020), 113211.

[24] A. Girdhar, H. Kapur, V. Kumar, M. Kaur, D. Singh, R. Damasevicius, Effect of COVID-19 outbreak on urban health and environment, Air Qual. Atmos. Health 14 (3) (2020) 389–397.

[25] O. Dekhil, Computational techniques in medical image analysis application for white blood cells classification, Electronic Theses and Dissertations. Paper 3424, 2020.

[26] L. Alzubaidi, M.A. Fadhel, O. Al-Shamma, J. Zhang, Y. Duan, Deep learning models for classification of red blood cells in microscopy images to aid in sickle cell anemia diagnosis, Electronics 9(3) (2020) 427, 1–18.

[28] T.F. Chan, L.A. Vese, Active contours without edges, IEEE Trans. Image Process. 10 (2) (2001) 266–277.

[29] C.B. Wijesinghe, D.N. Wickramarachchi, I.N. Kalupahana, L.R. De Seram, I.D. Silva, N.D. Nanayakkara, Fully automated detection and classification of white blood cells. 2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), 2020.

[30] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, p. arXiv:1409.1556.

[31] C. Jung, M. Abuhamad, J. Alikhanov, A. Mohaisen, K. Han, D. Nyang, W-net: a CNN-based architecture for white blood cells image classification, 2019, arXiv preprint arXiv:1910.01091.

[32] F. Ucar, Deep learning approach to cell classification in human peripheral blood. 2020 5th International Conference on Computer Science and

Engineering (UBMK), 2020.

[33] X. Zhang, X. Zhou, M. Lin, J. Sun, Shufflenet: an extremely efficient convolutional neural network for mobile devices, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 6848–6856.

[34] M.P. Karthikeyan, R. Venkatesan, V. Vijayakumar, L. Ravi, V. Subramaniyaswamy, White blood cell detection and classification using Euler's Jenks optimized multinomial logistic neural networks, J. Intell. Fuzzy Syst. 39 (6) (2020) 8333–83