



e-ISSN: 2319-8753 | p-ISSN: 2347-6710

# IJIRSET

International Journal of Innovative Research in  
**SCIENCE | ENGINEERING | TECHNOLOGY**



# INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 11, Issue 6, June 2022

**ISSN** INTERNATIONAL  
STANDARD  
SERIAL  
NUMBER  
INDIA

Impact Factor: 8.118



9940 572 462



6381 907 438



ijirset@gmail.com



www.ijirset.com

# Classification of Blood Cell Subtypes Using Deep Learning

D S Bhavani<sup>1</sup>

Assistant Professor, Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology  
Hyderabad, Telangana, India<sup>1</sup>

**ABSTRACT:** White Blood Cells also known as leukocytes play an important role in the human body by increasing the immunity by fighting against infectious diseases. The classification of White Blood Cells plays an important role in detection of a disease in an individual. The classification can also assist with the identification of diseases like infections, allergies, anemia, leukemia, cancer, Acquired Immune Deficiency Syndrome (AIDS), etc. that are caused due to anomalies in the immune system. This classification will assist the hematologist distinguish the type of White Blood Cells present in the human body and find the root cause of diseases. Currently there is a large amount of research going on in this field. Considering a huge potential in the significance of classification of WBCs, deep learning technique Convolution Neural Networks (CNN) which can classify the images of WBCs into its subtypes namely, Neutrophil, Eosinophil, Lymphocyte and Monocyte is used. The experimental results of various experiments executed on the Blood Cell Classification and Detection (BCCD) dataset using CNN are shown. This study presents an automated system for Blood Cell Classification. Using various image processing and classification techniques one can detect and classify images. This work points out different algorithms for classifying blood cell images. This work mainly involves three steps specially preprocessing of the image, extraction of features from the preprocessed image and the last one is classification of image using some deep learning techniques.

**KEYWORDS:** CNN, Blood cell classification, Basophils, Eosinophil, Monocytes, Lymphocytes, Neutrophil, TensorFlow, Keras, SoftMax function, Relu function

## I. INTRODUCTION

White Blood cells are key players in the immune system of the human body. There are three broad classifications of blood cells-Red Blood Cells (RBC) that transport oxygen, White Blood Cells (WBC) the face of the immune system and platelets that trigger blood clotting in damaged tissues. White Blood Cells make up 1% of the human blood in a healthy human adult. They are present throughout the body and each type of White Blood Cells have a certain functionality in the human body and serves by protecting the human body against various infections and diseases. If they detect any of these in the blood, they attack them to counter any potential damage these elements can cause in the body. The structure of the WBC, predominantly, comprises a large lobed nucleus that can be used to distinguish a WBC from other blood cell types. Apart from a nucleus, WBC consists of cytoplasm and cell wall.

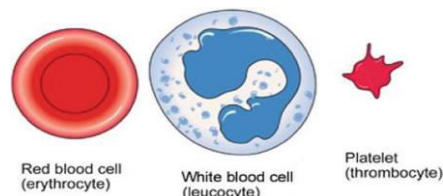


Fig. 1 Types of blood cells.

There are major five categories of WBC in the human body. However due to data set constraints we have classified data into four categories: Basophils (0.4% approximately), Eosinophil (2.3% approximately), Monocytes (5.3% approx.), Lymphocytes (30% approx.) and Neutrophils (62% approx.).

1. **Eosinophil:** The exact count of the Eosinophil in body keeps on changing during the day depending on the season

and during different phases of the human body. On an average, Eosinophil amounts to 2–4% of the total WBC count and can persist in the blood circulation for 8–12 days. These are found in the medulla, cortex region, lower gastrointestinal region and lymph nodes. For recognizing the Eosinophil from the images, the cell is rounded in the shape of skin-red color with a purple-colored bi-lobed nucleus. The lobes of the nucleus are connected by a thin strand.

2. **Monocytes:** The count of Monocytes present in a healthy human body varies between 6–9% of total WBC count. The lifetime of Monocytes varies from hours to a day. Monocytes are also responsible for presenting these pathogens to T cells so that they can be easily killed, and it helps reduce the response time of antibodies in humans. Monocytes can be recognized in the BCCD image dataset using certain features, the nucleus in it is a kidney shaped-roundish cell with skin red color and some purple color in it without any lobe.
3. **Lymphocytes:** Lymphocytes count present in a healthy human body is around 25–30% of total WBC count. These cells are more present in the lymphatic system than in blood. Lymphocytes consist of two types of cells, namely B-cells and T-cells. These cells are responsible for directly killing virus infected cells in the human body and eliminating cancer cells. Lymphocytes can be easily identified in the BCCD dataset by looking at the nucleus as it is clearly round purple colored potato and eccentric.
4. **Neutrophils:** Neutrophils are a part of the innate immune system. The WBCs consists of almost 60–70% of Neutrophils making it largest component of WBCs. The main targets of Neutrophils are bacterial and fungal pathogens. Neutrophils can be recognized as purple colored multi-lobed groundnut-shaped nucleus inside the skin-red colored cells. Generally, there are three to five lobes in the nucleus with a transparent looking cytoplasm (Figs. 2 and 3).

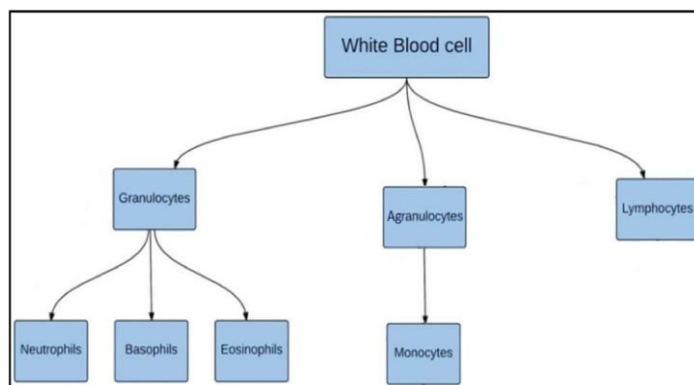


Fig. 2 Classification of WBCs.

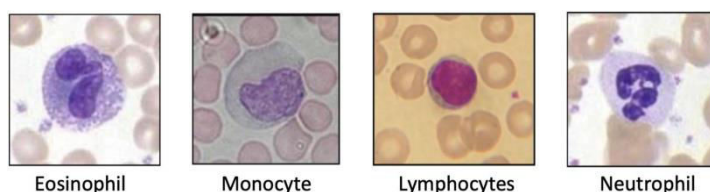


Fig. 3 Types of blood cells

In medical science, one of the challenges faced is the identification and determining the number of white blood cells. The major reason is the abundance of red blood cells. In a healthy adult, the WBCs make approximately 1% of the total blood volume. Due to this low proportion of the WBCs in blood, the recognition of the WBCs becomes a challenge which further toughens the job of classifying the subtypes of WBCs.



The change in number of any subtype of classification means that there is a problem in the body and the body is responding to a type of pathogen. The further prognosis of the disease can be determined by the disease type and can help in prescription of treatment for that specific ailment. The traditional method of classifying the WBC types includes studying the blood smeared slides under light and electron microscopes. The blood cells are stained prior to studying them under a microscope. For perfect identification, pathologists need to look for the shape of the nucleus and compare the size relative to RBCs. Since this is a manual process, it is prone to error and is time consuming. The biggest drawback with the manual classification is the aspect of human error associated with mechanical scanning of glass slide and the tradeoff between the image resolution and microscopic field of view (FOV). To overcome these disadvantages, there is a need for an automated method to classify the white blood cells using images of stained blood cells.

The traditional method of classifying the WBC types includes studying the blood smeared slides under light and electron microscopes. The blood cells are stained prior to studying them under a microscope. For perfect identification, pathologists need to look for the shape of the nucleus and compare the size relative to RBCs. Since this is a manual process, it is prone to error and is time consuming. To overcome these disadvantages, there is a need for an automated method to classify the white blood cells using images of stained blood cells.

The Automated system makes use of deep learning (computer vision). Take the blood smeared cell images of a person and image preprocessing is done on those images which are provided, and those images are given to the trained model. As a model is trained on a large number of images, it compares the features of images that are provided for prediction with the features of pretrained images and gives output as to which class it belongs to.

## **II. RELATED WORK**

Khamael AL-Dulaimi, Jasmine Banks<sup>1</sup>, Vinod Chandran, Inmaculada Tomeo-Reyes and Kien Nguyen [1] proposed the following approach. White blood cells (WBC) play a significant role in the immune system by protecting the body from infectious disease and foreign invaders. Therefore, an automatic identification of WBC from microscopic images is an essential importance to help the hematologist in diagnosing diseases, such as leukemia, AIDS, and certain types of blood cancer. Analysis of WBC structure from microscopic images and classification of cells into types and sub-types are challenging because of variations in maturation stage, and intra-class variations of the cell shape in images due to using different acquisition and staining processes. An overview of the structure of WBCs, the types and subtypes of WBC, and their features, including the shape of nuclei, size, function and colour are provided. Next, a detailed process of the identification of WBC in images, including image acquisition and consideration of the effect of staining to visualize changes in the colour and shape of the nucleus was mentioned. A survey of the recent history up to current state-of-the-art in automated identification of WBCs, including techniques such as image processing, signal processing, pattern recognition and deep learning techniques was provided.

Dixit Kolhatkar, Nisha Wankhede, [2] proposed the following approach. In the medical field blood testing is one of the most important clinical examination tests. In clinical laboratories counting of different types of blood cells is important for physicians to diagnose the diseases in particular patients. Manual microscopic inspection of blood cells is time consuming and requires more technical knowledge. Therefore, there is a need to research for an automated blood cell detection system that will help physicians to diagnose diseases in a fast and efficient way. Many researchers have done their research for counting blood cells using different methodologies. This work reviews different methodologies that have been used for blood cell counting. The objective is to study these methodologies and identify future research directions in order to get more accuracy.

Prayag Tiwari, Jia Qian, Qiuchi Li, Benyou Wang, Deepak Gupta, Ashish Khanna, Joel J.P.C. Rodrigues, C. Victor Hugo [3] proposed the following approach. Deep Learning has already shown power in many application fields and is accepted by more and more people as a better approach than the traditional machine learning models. In particular, the implementation of deep learning algorithms, especially Convolutional Neural Networks (CNN), brings huge benefits to the medical field, where a huge number of images are to be processed and analyzed. This work aims to develop a deep learning model to address the blood cell classification problem, which is one of the most challenging problems in blood diagnosis. A CNN-based framework is built to automatically classify the blood cell images into subtypes of the cells. Experiments are conducted on a dataset of 13k images of blood cells with their subtypes, and the results show that our proposed model provides better results in terms of evaluation parameters.

Ishpreet Singh, Narinder Pal Singh, Harnoor Singh, Saharsh Bawankar(B), and Alioune Ngom [4] proposed the following approach. White Blood Cells also known as leukocytes play an important role in the human body by increasing the immunity by fighting against infectious diseases. The classification of White Blood Cells plays an important role in detection of a disease in an individual. The classification can also assist with the identification of diseases like infections, allergies, anemia, leukemia, cancer, Acquired Immune Deficiency Syndrome (AIDS), etc. that are caused due to anomalies in the immune system. This classification will assist the hematologist distinguish the type of White Blood Cells present in the human body and find the root cause of diseases. Currently there is a large amount of research going on in this field. Considering a huge potential in the significance of classification of WBCs, we will be using a deep learning technique Convolution Neural Networks (CNN) which can classify the images of WBCs into its subtypes namely, Neutrophil, Eosinophil, Lymphocyte and Monocyte. In this work, A report on the results of various experiments executed on the Blood Cell Classification and Detection (BCCD) dataset using CNN was given.

### III. DESIGN METHODOLOGY

The system architecture and the steps involved in construction of a model for blood cell subtypes classification using deep learning algorithms is described as in the below figure. As shown in above figure 3 first image data set need to be read from storage/database, different image sizes are not accepted for model training, So, all images need to be transferred to some fixed size it is done by image preprocessing, in image preprocessing involves reshaping the image, crop, rotation, normalization, etc., can be done.

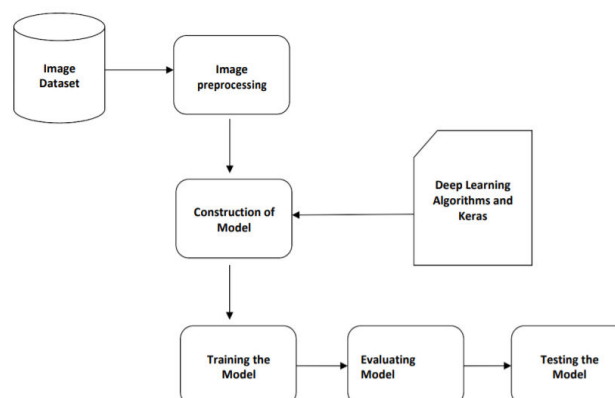


Fig. 3 System Architecture

Models are built using deep learning algorithms which are available in keras module. Using keras one can build our own models as well as by use transfer learning one can use predefined models. In the preprocessing step itself one should divide the image data set into training and testing data sets. Next, using preprocessed training image data, one trains the model. After training the model, evaluation of the model is done using a testing image data set. After the evaluation of the model, accuracy indicates how well the model can predict. For prediction images provided to the model are preprocessed and prediction is given by using predict function which returns classification report or an array indicating the class it belongs to.

#### Deep Learning Algorithms:

**Working of Convolution Neural Networks:** In neural networks, convolutional neural networks (ConvNets or CNNs) is one of the main categories to do image recognition, images classification. Object detections, recognition faces etc., are some of the areas where CNNs are widely used.

CNN image classifications take an input image, process it and classify it under certain categories (E.g., Dog, Cat, Tiger, Lion). Computers see an input image as an array of pixels, and it depends on the image resolution. Based on the image resolution, it will see  $h \times w \times d$  ( $h$  = Height,  $w$  = Width,  $d$  = Dimension). E.g., an image of a  $6 \times 6 \times 3$  array of matrix of rgb (3 refers to RGB values) and an image of  $4 \times 4 \times 1$  array of matrix of grayscale image.

Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernels), pooling, fully connected layers (FC) and apply SoftMax function to classify an object with probabilistic values between 0 and 1. The below figure is a complete flow of CNN to process an input image and

classifies the objects based on values.

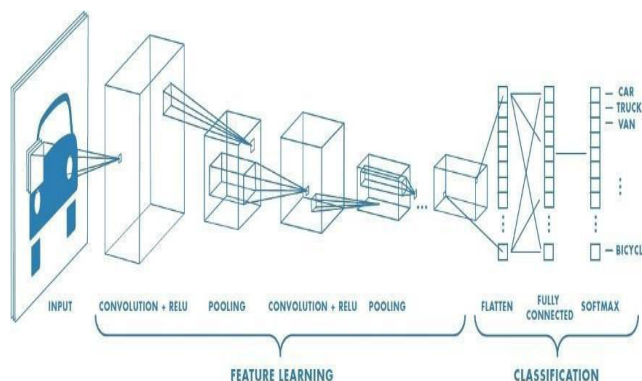


Fig. 4 Neural network with many convolutional layers

**Convolution Layer:** Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.

- An image matrix (volume) of dimension  $(h \times w \times d)$
- A filter  $(f_h \times f_w \times d)$
- Outputs a volume dimension  $(h - f_h + 1) \times (w - f_w + 1) \times 1$

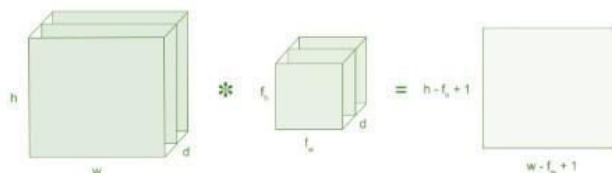


Fig. 5 Image matrix multiplies kernel or filter matrix

Consider a 5 x 5 whose image pixel values are 0, 1 and filter matrix 3 x 3 as shown in below

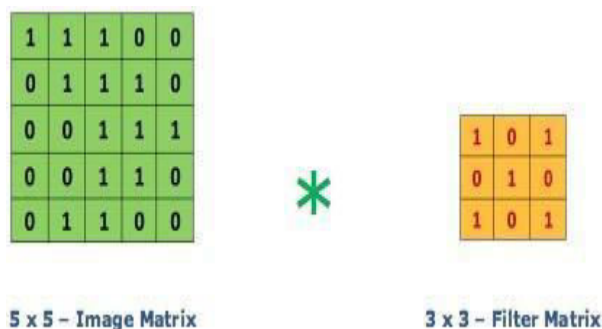


Fig. 6 Image matrix multiplies kernel or filter matrix

Then the convolution of 5 x 5 image matrix multiplied with 3 x 3 filter matrix which is called “Feature Map” as output shown in below

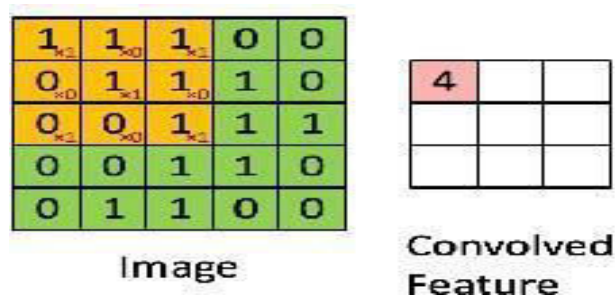


Fig. 73 x 3 Output matrix

Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters. The below example shows various convolution images after applying different types of filters (Kernels).

**Strides:** Stride is the number of pixels shifted over the input matrix. When the stride is 1 then move the filters to 1 pixel at a time. When the stride is 2 then move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a stride of 2.

**Padding:** Sometimes the filter does not perfectly fit the input image. There are two options: Pad the picture with zeros (zero-padding) so that it fits and drop the part of the image where the filter did not fit. This is called valid padding which keeps only valid parts of the image.

**Pooling Layer:** Pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling is also called sub sampling or down sampling which reduces the dimensionality of each map but retains important information.

**Fully Connected Layer:** The layer is called the FC layer, flattened our matrix into a vector and fed it into a fully connected layer like a neural network.








Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Fig. 8Some common filters

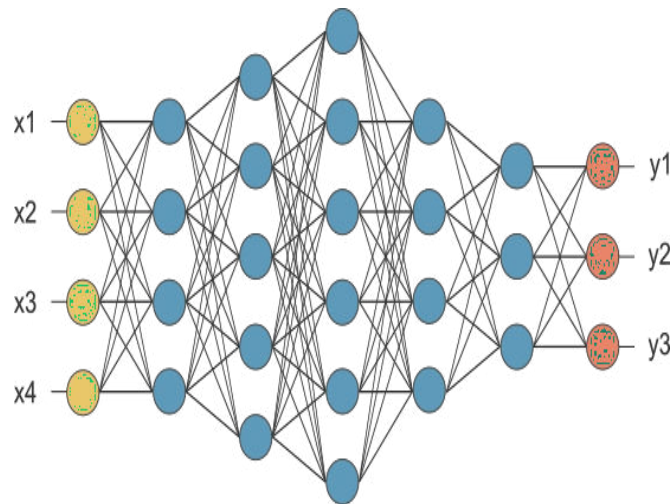


Fig. 9 After pooling layer, flattened as FC layer

**Efficient Net Model:** EfficientNet, first introduced in tan and le, 2019 is among the most efficient models (i.e. requiring least FLOPS for inference) that reaches state-of-the-art accuracy on both imagenet and common image classification transfer learning tasks. The smallest base model is like MnasNet, which reached near-SOTA with a significantly smaller model. By introducing a heuristic way to scale the model, EfficientNet provides a family of models (B0 to B7) that represents a good combination of efficiency and accuracy on a variety of scales. Such scaling heuristics (compound-scaling, details see Tan and Le, 2019) allows the efficiency-oriented base model (B0) to surpass models at every scale, while avoiding extensive grid-search of hyper parameters where various augmentation schemes and semi-supervised learning approaches are applied to further improve the imagenet performance of the models. These extensions of the model can be used by updating weights without changing model architecture.

To use efficientnetB0 for classifying 1000 classes of images from imagenet, run: from tensorflow.keras.applications import EfficientNetB0 model = EfficientNetB0(weights='imagenet')

This model takes input images of shape (224, 224, 3), and the input data should range [0, 255]. Normalization is included as part of the model.

Because training efficientnet on imagenet takes a tremendous number of resources and several techniques that are not a part of the model architecture itself. Hence the keras implementation by default loads pretrained weights obtained via training with auto augment.

**SoftMax Function:** SoftMax converts a real vector to a vector of categorical probabilities. The elements of the output vector are in range (0, 1) and sum to 1. Each vector is handled independently. The axis argument sets which axis of the input the function is applied along. SoftMax is often used as the activation for the last layer of a classification network because the result could be interpreted as a probability distribution.

The SoftMax of each vector  $x$  is computed as  $\exp(x) / \text{tf.reduce\_sum}(\exp(x))$ .

#### Module Description:

**NumPy:** NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with python. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. As it is used to divide the given image from a construction site into  $n$  dimensional objects such that it can be easily compared with the dataset.

**Pandas:** Pandas is a python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. Pandas allows us to analyze big data and make conclusions based on statistical theories. Pandas can clean messy data sets and make them readable and relevant.

**Scikit-learn:** Scikit-learn is probably the most useful library for machine learning in python. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction.



**TensorFlow:** TensorFlow is an open-source software library. TensorFlow was originally developed by researchers and engineers working on the google brain team within Google's machine intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains.

TensorFlow is basically a software library for numerical computation using data flow graphs where:

1. Nodes in the graph represent mathematical operations.
2. Edges in the graph represent the multidimensional data arrays (called tensors) communicated between them.

**Keras:** Keras is a deep learning API written in python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.

TensorFlow 2 is an end-to-end, open-source machine learning platform. You can think of it as an infrastructure layer for differentiable programming. It combines four key abilities:

1. Efficiently executing low-level tensor operations on CPU, GPU, or TPU.
2. Computing the gradient of arbitrary differentiable expressions.
3. Scaling computation to many devices (e.g., the Summit supercomputer at Oak Ridge National Lab, which spans 27,000 GPUs).
4. Exporting programs ("graphs") to external runtimes such as servers, browsers, mobile and embedded devices.

Keras is the high-level API of TensorFlow 2: an approachable, highly productive interface for solving machine learning problems, with a focus on modern deep learning. It provides essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity. Keras empowers engineers and researchers to take full advantage of the scalability and cross-platform capabilities of TensorFlow 2: you can run keras on tpu or on large clusters of GPUs, and you can export your keras models to run in the browser or on a mobile device.

#### IV. EXPERIMENTAL RESULTS

##### Training

Training the model is a process of making a model to learn from provided data/data set and using it for further predictions. Before training all images are to be preprocessed, preprocessing involves reshaping the image, crop, all other data augmentation steps etc.,

Training the model with enough amount and proper data set is very essential. Before training the CNN, model images are to be converted into some fixed size, so that model can be well fit.

Image data set is divided into two parts

1. Training data set
2. Testing data set.

The training data set is used for training the model, testing data set is used for evaluation/testing purpose.

Training images look like.

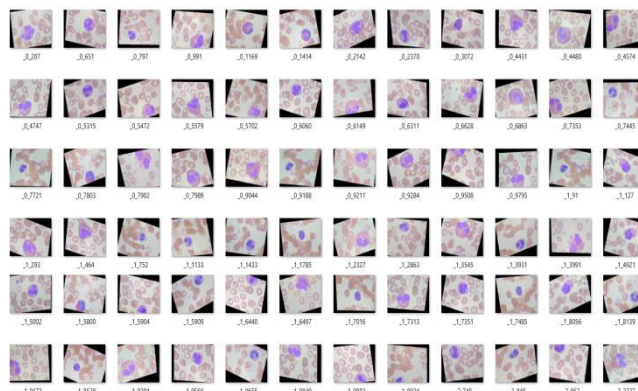


Fig. 10 Training Images

The System has built using EfficientNet model and it is trained on training image data set using fit function as follows

```
history = model2.fit(generator.flow(X_train,
                                   y_trainHot,
                                   batch_size=32),
                    epochs=100,
                    steps_per_epoch=len(X_train)/32,
                    callbacks=[es_callback, reduce_lr],
                    validation_data=(X_test, y_testHot))

pd.DataFrame(history.history)[['accuracy', 'val_accuracy']].plot()
pd.DataFrame(history.history)[['loss', 'val_loss']].plot()
plt.show()
```

Epoch 1/100  
 311/311 [=====] - 76s 200ms/step - loss: 1.4031 - accuracy: 0.3056 - val\_loss: 1.4312 - val\_accuracy: 0.2598  
 Epoch 2/100  
 311/311 [=====] - 62s 199ms/step - loss: 1.2830 - accuracy: 0.4024 - val\_loss: 1.2026 - val\_accuracy: 0.3812  
 Epoch 3/100  
 311/311 [=====] - 62s 200ms/step - loss: 1.1840 - accuracy: 0.4944 - val\_loss: 1.2098 - val\_accuracy: 0.4656  
 Epoch 4/100  
 311/311 [=====] - 62s 199ms/step - loss: 1.0733 - accuracy: 0.5826 - val\_loss: 1.1078 - val\_accuracy: 0.5444  
 Epoch 5/100  
 311/311 [=====] - 62s 200ms/step - loss: 0.9588 - accuracy: 0.6556 - val\_loss: 1.0031 - val\_accuracy: 0.6072  
 Epoch 6/100  
 311/311 [=====] - 62s 199ms/step - loss: 0.8489 - accuracy: 0.7144 - val\_loss: 0.9024 - val\_accuracy: 0.6671  
 Epoch 7/100  
 311/311 [=====] - 62s 200ms/step - loss: 0.7413 - accuracy: 0.7625 - val\_loss: 0.8128 - val\_accuracy: 0.7141

Fig. 11 Training the model

## Results

The below random images are taken from the Internet and few images are taken from training and testing image data set and are fed against the two models that are trained in prior.

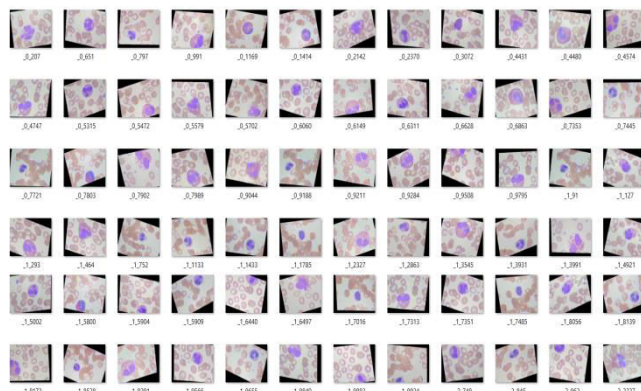


Fig. 12 Images that are used for Prediction

Prediction is done by using a prediction function. Images are read using imread OpenCV function, then images are preprocessed which includes conversion image from BGR to RGB and then images are resized then preprocessed images are converted into array and then normalized by dividing the image by 255 then using predict function the model gives regression result then from that whichever the class has value greater than 0.5 then that particular class is assigned.

Figure 8 and Figure 9 shows the code that is used for reading the new images and preprocessing them then normalization and then sent for prediction function.

After prediction is done by model results are as below

```
def get_data(folder):
    """
    Load the data and labels from the given folder.
    """
    X = []
    y = []
    for wbc_type in os.listdir(folder):
        if not wbc_type.startswith('.'):
            if wbc_type in ['NEUTROPHIL']:
                label = 0
            elif wbc_type in ['EOSINOPHIL']:
                label = 1
            elif wbc_type in ['MONOCYTE']:
                label = 2
            else:
                label = 3
            for image_filename in tqdm(os.listdir(folder + wbc_type)):
                img_file = cv2.imread(folder + wbc_type + '/' + image_filename)
                if img_file is not None:
                    #img_file = scipy.misc.imresize(arr=img_file, size=(60, 80, 3))
                    img_file = cv2.cvtColor(img_file, cv2.COLOR_BGR2RGB)
                    img_file=cv2.resize(img_file,(120,120))
                    img_arr = np.asarray(img_file)
                    X.append(img_arr)
                    y.append(label)
    X = np.asarray(X)
    y = np.asarray(y)
    return X,y

class_names=['NEUTROPHIL','EOSINOPHIL','MONOCYTE','LYMPHOCYTE']

X_pred, y_pred = get_data('/content/drive/MyDrive/Major Project Dataset/dataset2-master/dataset2-master/images/TEST_SIMPLE/')
y_pred_hot = to_categorical(y_pred, num_classes = 4)

res_pred=model2.predict(X_pred)

def display_examples(class_names, images, labels):
    fig = plt.figure(figsize = (20,20))
    fig.suptitle("Results of Test Images Using the Model", fontsize=16)
    for i in range(70):
        plt.subplot(14,5,i+1)
        plt.xticks([])
        plt.yticks([])
        plt.grid(False)
        plt.imshow(images[i], cmap=plt.cm.binary)
        plt.xlabel(class_names[labels[i]])
    plt.show()

display_examples(class_names, X_pred[0:70], test_pred)

def display_examples(class_names, images, labels):
    fig = plt.figure(figsize = (20,20))
    fig.suptitle("Results of Test Images Using the Model", fontsize=16)
    for i in range(70):
        plt.subplot(14,5,i+1)
        plt.xticks([])
        plt.yticks([])
        plt.grid(False)
        plt.imshow(images[i], cmap=plt.cm.binary)
        plt.xlabel(class_names[labels[i]])
    plt.show()

display_examples(class_names, X_pred[0:70], test_pred)
```

Fig. 13 Implementation of Model predictions

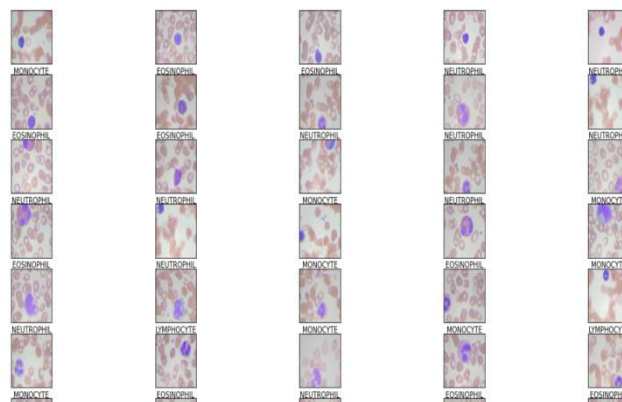


Fig. 14Result of Model

## V. CONCLUSION

In a healthy adult, the WBCs make approximately 1% of the total blood volume. Due to this low proportion of the WBCs in blood, the recognition of the WBCs becomes a challenge which further toughens the job of classifying the subtypes of WBCs.

The change in number of any subtype of classification means that there is a problem in the body and the body is responding to a type of pathogen. The further prognosis of the disease can be determined by the disease type and can help in prescription of treatment for that specific ailment. The traditional method of classifying the WBC types includes studying the blood smeared slides under light and electron microscopes. The blood cells are stained prior to studying them under a microscope. For perfect identification, pathologists need to look for the shape of the nucleus and compare the size relative to RBCs. Since this is a manual process, it is prone to error and is time consuming. The biggest drawback with the manual classification is the aspect of human error associated with mechanical scanning of glass slide and the tradeoff between the image resolution and microscopic field of view (FOV). To overcome these disadvantages, there is a need for an automated method to classify the white blood cells using images of stained blood cells.

Using this model/system a person with blood slide images can know the subtype class. Getting tested by traditional methods is time consuming and costly but using this model by just having blood smeared images that person can know subtype class in a short period of time. This model is fully automated with an end-to-end structure without the need for manual feature extraction. This system can be used in remote places if a web based/API is built.

The future scope of this work can be like providing some user interface/web API using which images can be provided to the model and getting instant results. This model can be used for other disease classification as well which involves similar characteristics. This system can be used for other classification with some modification, and it is beneficial that this model gives you some idea of approach.

## REFERENCES

- [1] Khamael AL-Dulaimi<sup>1</sup>, Jasmine Banks, Vinod Chandran, Inmaculada Tomeo-Reyes and Kien Nguyen, Classification of White Blood Cell Types from Microscope Images: Techniques and Challenges, Research Gate (2018).
- [2] Mr. Dixit Kolhatkar, Ms. Nisha Wankhede, Detection and Counting of Blood Cells using Image Segmentation, 2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (WCFTR'16) - WCFTR
- [3] Prayag Tiwari, Jia Qian, Qiuchi Li, Benyou Wang, Deepak Gupta, Ashish Khanna, Joel J.P.C. Rodrigues, C. Victor Hugo, Detection of subtype blood cells using deep learning, ResearchGate (2018).
- [4] Ishpreet Singh, Narinder Pal Singh, Harnoor Singh, Saharsh Bawankar(B), and Alioune Ngom, Blood Cell Types Classification Using CNN, ResearchGate.
- [5] Prashant Pandey, Prathosh AP, Vinay Kyatham, Deepak Mishra and Tathagato Rai Dastidar, Target-Independent Domain Adaptation for WBC Classification using Generative Latent Search.
- [6] Merl James Macawile, Vonn Vincent Quiñones, Alejandro Ballado JrJennifer Dela Cruz, Meo Vincent Caya, White Blood Cell Classification and Counting Using Convolutional Neural Networks.





- [7] Seyed Hamid Reza tofighi<sup>1</sup>, Kosar Khaksari<sup>1</sup>, and Hamid Soltanian-Zadeh, Automatic Recognition of Five Types of White Blood Cells in Peripheral Blood.

#### **BIOGRAPHY**



**D S Bhavani<sup>1</sup>**, B.Tech(CSE), M.Tech(CS), pursuing her Ph.D. in Koneru Lakshmaiah Education Foundation(KLH Deemed to be University) at Hyderabad. She has teaching experience of 16 years. Currently working as Assistant Professor in the Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Hyderabad. She has various research papers published in the International Journals of repute. Her research area consists of Network Security, Internet of Things, Digital Forensics, Machine Learning, AI etc.

#### **Correspondence Address:**

Mahatma Gandhi Institute of Technology,  
Chaitanya Bharathi Post, Gandipet, RangaReddy District, Hyderabad-500075



**INNO SPACE**  
SJIF Scientific Journal Impact Factor  
**Impact Factor: 8.118**



**ISSN** INTERNATIONAL  
STANDARD  
SERIAL  
NUMBER  
**INDIA**



# INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

 **9940 572 462**  **6381 907 438**  **ijirset@gmail.com**



[www.ijirset.com](http://www.ijirset.com)

Scan to save the contact details