Classification of white blood cells from microscopic images using CNN

J.Setu Sai Sowmya Kumari[1], K.ChandraSekharaChari[2], M.Leelavathi[3], K.Pavan Kumar[4], Mrs.N.Rajeswari[5]

[1,2,3,4]Student, [5]Associate Professor
CSE Department & GEC Gudlavalleru, Krishna

Abstract: **White Blood Cells also known as leukocytes plays 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 human body and find the root cause of diseases. Currently there are a large amount of research going on in this field. Considering a huge potential in the significance of classification of WBCs, a deep learning technique named Convolution Neural Networks (CNN) will be used which can classify the images of WBCs into its subtypes namely, Neutrophil, Eosinophil, Lymphocyte and Monocyte. The results of various experiments executed on the Blood Cell Classification and Detection (BCCD) dataset using CNN are reported in this project.**

Keywords**: Convolutional Neural Network (CNN), Eosinophil, Lymphocyte, Monocyte, Neutrophil, ReLu, Softmax, Flatten, Max Pooling.**

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 immune system and platelets that trigger blood clotting in damaged tissues. White Blood Cells makes 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 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 of 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.
There are major five categories of WBC in 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% approximately), Lymphocytes (30% approximately) and Neutrophils (62% approximately).

II.RELATED WORK

Several researchers have previously proposed features to differentiate white cells. Ingram and Preston, have reported a technique for cell identification based upon highly recursive image transformations, known as Golay pattern transforms. These transforms yield information about such properties of the cell nucleus as area, length of perimeter, and fine structure. While it appears that their technique is of acceptable accuracy, it is comparatively slow in performing a differential count. Young, has also described a technique using a sequential procedure based on erythrocyte color to locate the white blood cells in the field of view. Statistical analysis on a set of 74 training sets yielded a four-dimensional feature vector. Monici have worked with a set of white cells that their fluorescence properties had been studied in suspension and on single cells at microscopy. Lymphocytes, monocytes, neutrophils and eosinophils were distinguished according to the intensity and spectral shape of the auto-fluorescence emission in the visible range from 440 to 580 nm. It appears that their technique fails to distinguish the basophil type.

III.METHODOLOGY

For developing the system certain methodologies are used. The methodology used in this project is image classification using CNN.

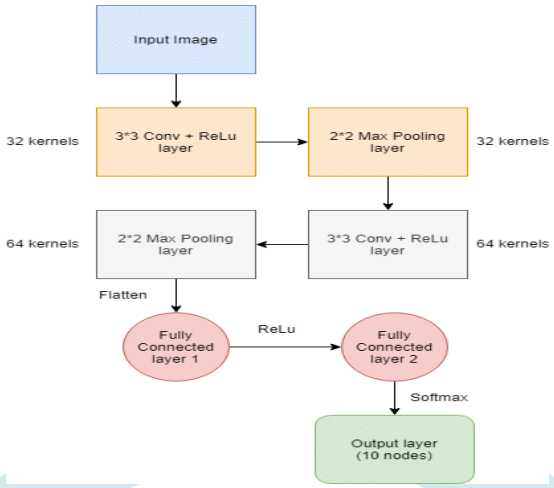


Fig 3.1 Architecture of CNN

The convolutional neural network (CNN) is developed by using the following steps:

3.1. Data Collection:

The size of the training dataset will be crucial because one of the characteristics of deep learning is its ability to perform well when trained with large data sets thus thousands of images are required to train CNN effectively. The worker's images are collected through browsing. The image files collected are in JPG or PNG image format.

3.2. Data Preprocessing:

3.2.1. Standardization:

The images converted to the same image format i.e. JPG and were transformed to the same size i.e. 128px by 128px.

3.2. Folder Structure:

Approximately 80% of the images are for the training dataset and the remaining 20% is for the testing dataset. The images collected were put into the respective dataset folder and subfolders based on the images to store in it. The subfolder name defined the label that would be applied to each image.

3.3. Training the CNN model :

The CNN model is created based on the requirement. Train the model by assigning the training dataset.

3.3.1 Importing the packages:

Keras is a framework for building deep neural networks using Python. To avoid complexity, it is designed to build a deep neural network with a few lines of code. Keras provides an easy front-end layer to build a deep neural network utilizing Tensor Flow or Theano at the back-end. It enables fast experimentation with deep neural networks.
First, we create our model using the "Sequential" model from Keras. This model is a linear stack of layers, meaning that we will create our model layer-by-layer.

3.3.2Adding Layers:

The sequential model with the convolution layer, max-pooling layer, flattening layer, dense layer, hidden layer, and drop out layer is built.

a) Convolution Layer:

The first convolution layer is the input layer. The first parameter is the number of convolution filters to use in the layer, which is set to "32". This is also the number of neurons or nodes that will be in this layer. The second parameter is the filter's size or the receptive field. Imagine we are creating a window of the size (3, 3), or width of three, and a height of three, that our convolution layer is restricted to looking through at any given time.
The third parameter is the input shape. This parameter only needs to be specified in the first convolution layer. After the first layer, the rest can be handled by our model. The input shape is simply the shape of the images that will be fed to CNN. The shape of our input images will be (128, 128, 3) (width, height, depth). The fourth parameter we will set is the activation function. Our nonlinear activation function is ReLu or rectified linear unit. The ReLu function is f(x) = max (0, x). Therefore, all negatives are converted to zeros while all positives remain the same. ReLu is one of the most popular activation functions because it reduces the vanishing gradient issue and is computationally cheaper to compute. This does not mean that the ReLu function is perfect, but it will get the job done for most applications.

b) Max Pooling Layer:

The max-pooling layers will only have one parameter for this model. The parameter is the pool size or the factor to downscale the input's spatial dimensions. The pool size will be set to (2, 2), which will downscale by half each time.

c) Flatten:
Flattening is required to convert multi-dimensional data into usable data for the fully connected layers so that for the fully connected layers to work, we need to convert the convolution layer's output to a 1D vector. Our convolution layers will be using 2D data (images). before it is fed into the classifier, this will have to be reshaped, or flattened, to one dimension.

d) Dense - ReLu:
Dense layers are the fully connected layers in which means every neuron is connected to all the neurons in previous layers. We will be using 128 nodes. This also means that the fully connected layer has an output size of 128. For this fully connected layer, the ReLu activation function will be used.

e) Dense - Softmax:
Our final fully connected layer will use the softmax function. Our problem involves two classes: Helmet worn and Helmet not worn. This is a categorical classification problem where softmax can be used to return a probability over the classes for each image.

Compile Keras Model:

Now that the model is defined, we can compile it. Compiling the model uses efficient numerical libraries under the covers (the so-called backend) such as Theano or TensorFlow.Training a network means finding the best set of weights to map inputs to outputs in our dataset.
We must specify the loss function to use to evaluate a set of weights, the optimizer is used to search through different weights for the network and any optional metrics we would like to collect and report during training.
In this case, we will use cross-entropy as the loss argument. This loss is for categorical classification problems and is defined in Keras as "categorical_crossentropy".
We will define the optimizer as the efficient stochastic gradient descent algorithm "adam". This is a popular version of gradient descent because it automatically tunes itself and gives good results in a wide range of problems.
Finally, because it is a classification problem, we will collect and report the classification accuracy, defined via the metrics argument.

3.4: Training the model:

We will be using the ImageDataGenerator() class from Keras for our data augmentation which helps us to expand our training dataset.
The first step is to rescale our data because most images have RGB values ranging from 0-255 and these values are too high for most models to handle, but we can condense each RGB value to a value between 0-1 by multiplying these values by 1/255. This is much easier to process.
After that we have shear_ range which will randomly apply shear transformations or shear mapping to the data. The value "0.2" is the intensity.
Zoom_range is also set to "0.2". All these for randomly zooming in on the images. A horizontal flip is set to "True" because we want to randomly flip half of the images in our dataset.
This is where we rescale our test set. The test set does not contain all of the same transformations applied to the training data. The training data only can be manipulated to avoid overfitting. The test set must be the original images to accurately predict the security helmet.
Now we will take the path of our test, train, and validation folders and generate batches of augmented data using flow_from_directory() from Keras.

- The directory is the first argument to pull from.
- The dimensions of the images or the target size after they are resized is the second argument.
- The third argument is class_mode. class_mode is set to categorical.
- The next step is to train the model that will be done using the fit_generator() method which is from Keras where this will train the model on batches of data that are generated from the training set.
- The steps per epoch is the first argument where the steps per epoch will tell the model the total number of batches of samples to produce from the generator before concluding that specific epoch.
- The number of epochs or training iterations is the second argument. The Keras documentation states that an epoch is defined as iteration over the entire data provided, as defined by steps_per_epoch.
- The validation data is the third argument which the model will use where the model will not be trained on the validation data, but this will help measure the loss at the end of every epoch.

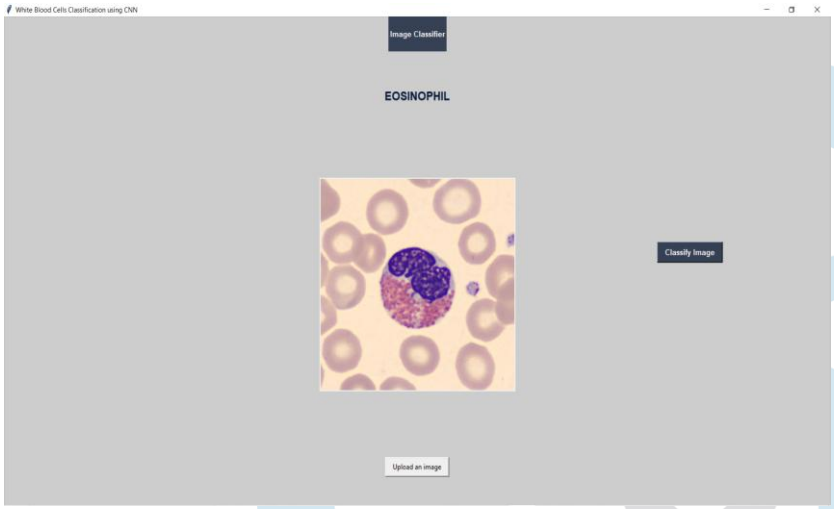
The final argument is the validation steps. our validation data is coming from a generator, so the number of batches of samples to produce from the generator must be set, similar to the steps per epoch

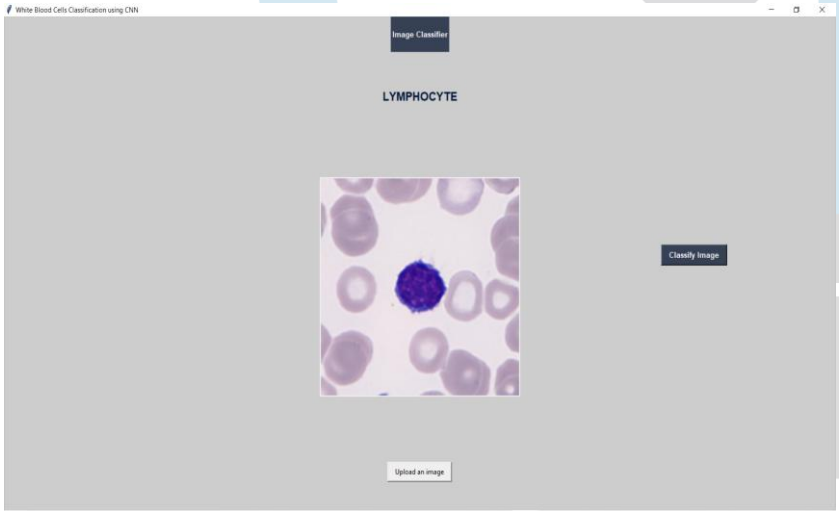
IV.RESULTS & DISCUSSION

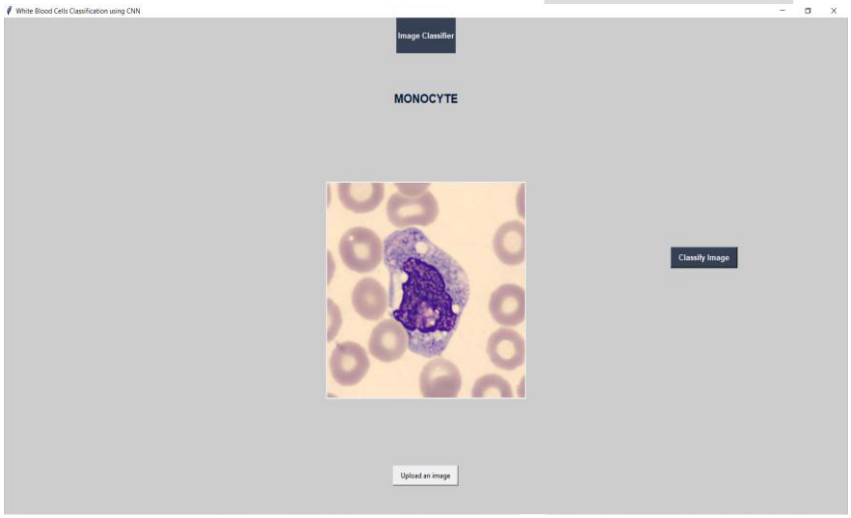
The output is seen through the user interface which consists of the button to upload the image it shows the output by considering the input as uploaded image that is if we give the image then it predicts the respective output of an image as

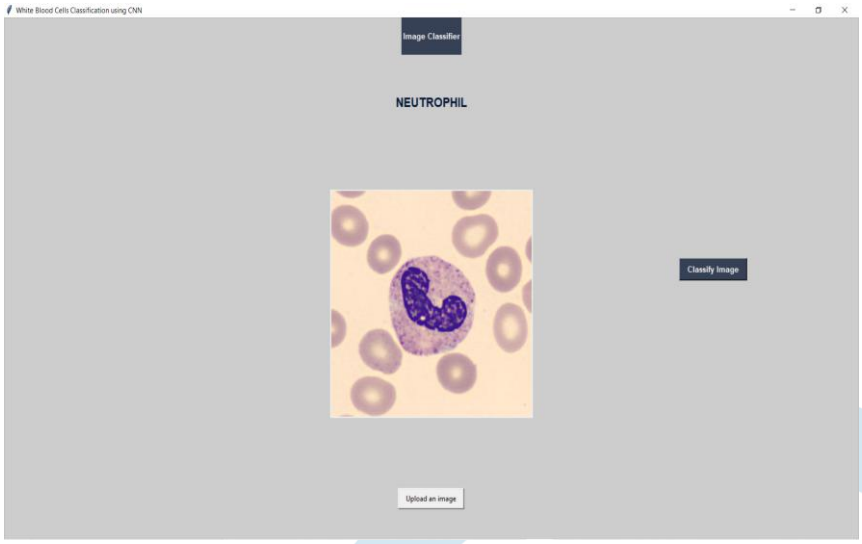
- Eosinophil
- Lymphocyte
- Monocyte
- Neutrophil.

Result of the some of the instant images is as follows









Discussions:

CNN is a very powerful algorithm which is widely used for image classification and object detection. The hierarchical structure and powerful feature extraction capabilities from an image makes CNN a very robust algorithm for various image and object recognition tasks.

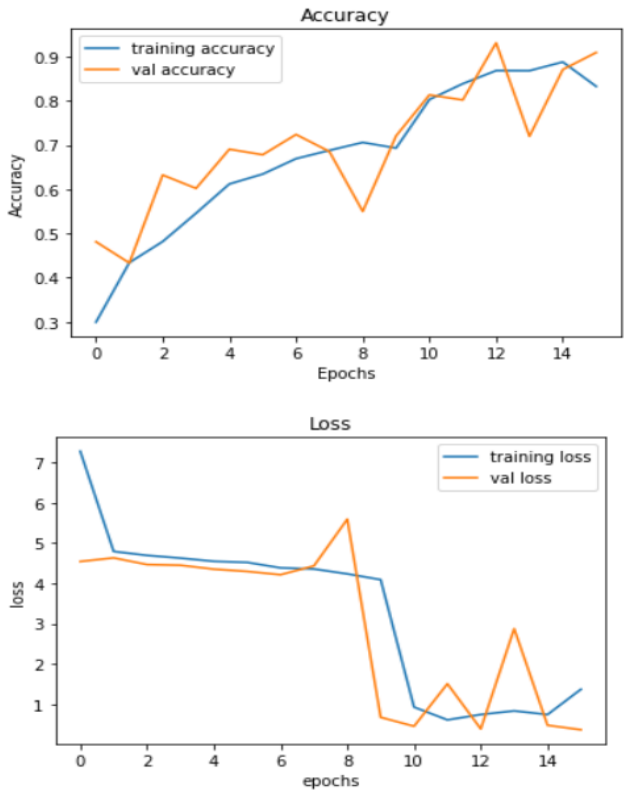
The depth-wise separable convolutional neural network model created in this study is especially suited for mobile devices since it have low latency, uses less computational power while at the same time maintaining high accuracy. Finally, the developed system included a security helmet map feature. The hierarchical structure and powerful feature extraction capabilities from an image makes CNN a very robust algorithm for various image and object recognition tasks.The main advantage of CNN compared to its predecessors is that it automatically detects the important features without any human supervision. CNN gives the best accuracy for image classification among all the algorithms.

Running the example, you can see that the verbose output on each epoch shows the loss and accuracy on both the training dataset and the validation dataset.

Accuracy:

```
Epoch 12/16
120/120 [==============================] - 60s 502ms/step - loss: 0.7008 - accuracy: 0.8142 - val_loss: 1.5090 - val_accuracy:
0.8021
Epoch 13/16
120/120 [==============================] - 63s 529ms/step - loss: 0.7918 - accuracy: 0.8574 - val_loss: 0.3882 - val_accuracy:
0.9312
Epoch 14/16
120/120 [==============================] - 62s 513ms/step - loss: 0.7301 - accuracy: 0.8790 - val_loss: 2.8767 - val_accuracy:
0.7198
Epoch 15/16
120/120 [==============================] - 62s 515ms/step - loss: 1.2998 - accuracy: 0.8567 - val_loss: 0.4786 - val_accuracy:
0.8708
Epoch 16/16
120/120 [==============================] - 59s 495ms/step - loss: 1.6195 - accuracy: 0.8210 - val_loss: 0.3682 - val_accuracy:
0.9094
```

Curve between accuracy & epochs and loss & epochs :



V.CONCLUSION

This paper helps the hematologist to classify White Blood Cells into their subtypes with the help of microscopic images of cell using Convolutional Neural Network techniques. This classification helps to distinguish the cells and check what type of disease a patient is suffering from. The results obtained from this experiment helps identify images in a robust way as compared to the orthodox lab methods. The good level of accuracy above 90 for the test set. Hence, when the model is trained with high computational abilities present, a perfect model can be trained and can be applied in the medical analysis and applications dealing with the number of white blood cells and sub types of white blood cells.

VI.FUTURE WORK

In this study it is assumed that the number of samples in each individual class is identical and we have a balanced database whereas in practice typical proportions of the cell types are not the same in blood smear slides (e.g., neutrophil (40- 75%) vs basophil granulocytes (0.5%)). In such cases, a Breiman Random Forest (BRF) classifier may be potentially useful. The BRF algorithm can deal with imbalanced data, can handle more variables (features) than observations (large attributes, small sample), is robust for data sets containing noisy samples, and has a good predictive ability without overfitting the data. In the processing of low quality blood images we expect improved performance by combining the CNN and BRF classifiers. We are also considering other approaches such as using multidimensional pattern classifiers or structural pattern recognition methods.

REFERENCES

[1] Dertat, A.: Applied Deep Learning - Part 4: Convolutional Neural Networks, Medium, 13 November 2017. https://towardsdatascience.com/applied deep-learning-part-4-convolutional-neural-networks-584bc134c1e2.
[2] Prabhu: Understanding of Convolutional Neural Network (CNN) - Deep Learning, Medium, 21 November 2019. https://medium.com/@RaghavPrabhu / understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148.
[3] Dernoncourt, F.: What is batch size in neural network? Cross Validated, 01 June 1965. https://stats.stackexchange.com/questions/153531 / what-is-batchsize in-neural-network.
[4] Daniel: What's is the difference between train, validation and test set, in neural networks? Stack Overflow, 01 July 1960. https://stackoverflow.com/questions/2976452/whats-is-the-difference-between-train-validation-and-test-set-in-neural network.
[5] LNCS Home Page. http://www.springer.com/lncs Accessed 4 Oct 2017.
[6] Blood Cell Images. https://www.kaggle.com/paultimothymooney/blood-cells . Accessed 21 Apr 2018.
[7] Kampbell, N.A.: Biology. Benjamin Cummings, San Francisco (n.d.) AQ5.
[8].NCI Dictionary of Cancer Terms (n.d.) https://www.cancer.gov/publications/dictionaries /cancer-terms/.
[9] Macawile, M.J., Quinones, V.V., Ballado, A., Cruz, J.D., Caya, M.V.: White blood cell classification and counting using convolutional neural network. In: 2018 3rd International Conference on Control and Robotics Engineering (ICCRE) (2018) Author Proof12 I. Singh et al.
[10] Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., Walter, P.: Molecular Biology of the Cell, p. 1367. Garland Science, New York (2002) AQ4.