Deep Convolutional Neural Networks (CNN):

CNN is extensively used for image classification because it uses neighboring pixel information to downsample the image effectively and then perform predictions, which resulted in high accuracies [1,2,3]. Additionally, they work using neural networks which are scalable for large datasets. It comprises of a complex feed forward neural network involving convolutions, pooling and classification. The term convolution refers to calculation of similarities between two functions when one function passes (or convolutes) over other. Pooling is used to reduce the number of parameters when the image is too large, followed by classification based on training and improvement by backpropagations. An image according to the computer is perceived as a collection of numbers or pixels organized in three dimensions viz. width, height and depth. Thus, matrix multiplications are the core operations of CNN. The functioning of CNN can be divided into two parts, the feature extraction and classification.

Feature Extraction: Convolution is performed by sliding a filter (feature vector) over input data to produce a feature map. This is obtained by performing matrix multiplication between the two at every location to extract different parts of the image and sum up the result onto a feature map. This operation is performed multiple times using different values of filters to obtain multiple feature maps. This is the output of the convolution layer. The output in the real world is non-negative and non-linear thus, an activation function is applied on it. In our case we use Rectifier Activation function to introduce non-linearity. In order to prevent the feature map from shrinking we perform padding which is done by appending zeros surrounding the input. It is common to add a pooling layer between the convolution layers to reduce the number of computations in the network by reducing the dimensionality. Various types of pooling like, max pooling (taking the maximum of neighboring pixels after convoluting), average pooling (taking the average of neighboring pixels after convoluting) and sum pooling (considering all neighboring pixel values).

Classification: This part is made up of fully connected layers which accept data in 1 dimension. Thus, we convert our 3D data to 1D by applying the flatten layer to the output of feature extraction. The neurons in this layer have connections to activations of the previous layers. The fully connected layer is followed by the dense layer which form the hidden layer of neurons where backpropagation (using the principle of gradient descent) is applied, followed by the softmax layer which provides a binary output of whether the input image belongs to a particular class or not.

Model Representation:

After testing multiple values for hyperparameters, we obtained the best accuracies with 32 filter, each filter size of 3 X 3, ReLU activation function for imposing non-linearity. A dropout of 25% is also applied to handle overfitting. We also pad the feature maps so that the input size is equal to output size to avoid feature map shrinking. The dense is composed of 128 neurons. In order to obtain high accuracies, we consider the orientations of cell location in each image. For this, we use the ImageDataGenerator in python which generates a batch of tensor images with real-time data augmentation. We apply rotations, height and width shifting, normalizations, and flipping on original image to create a set of images containing the white blood cell type with different orientations. This will ensure that our model will take care of same cells

positioned in different angles are classified correctly. The sequence of layers in our model is as shown in figure 1. We run the model for 30 epochs, i.e. we update the weights 30 times by performing backpropagation to obtain an accuracy of 83%.

Results: As seen in figure 2, with increasing number of epochs, the model obtains close results

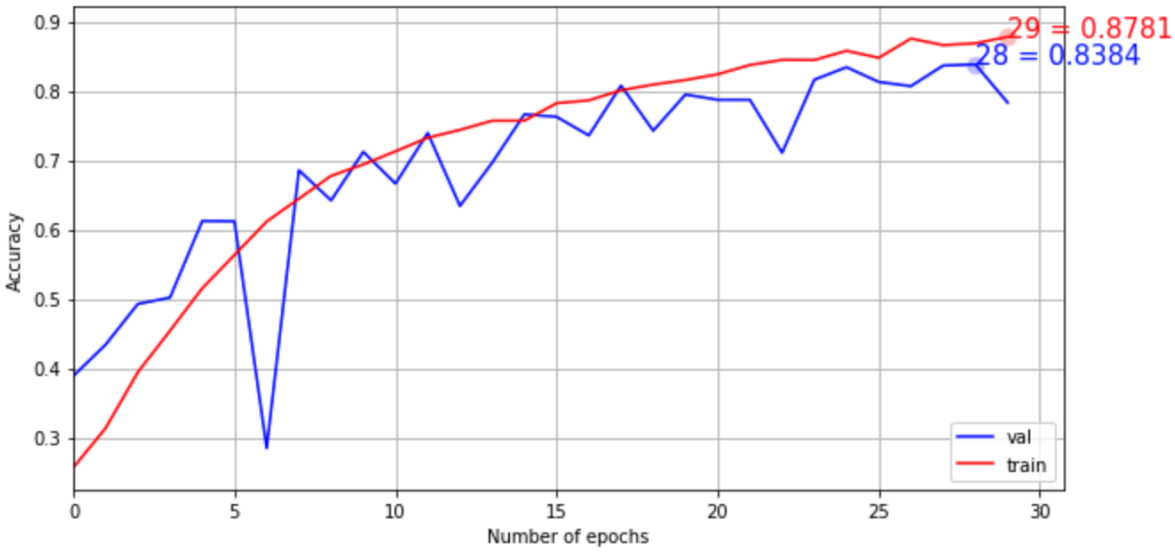

Figure 2: change in accuracy with increasing epochs

	Precision	*Recall*	*F1-score*	*support*
NEUTROPHIL	0.57	0.88	0.69	624
EOSINOPHIL	0.96	0.53	0.68	623
MONOCYTE	0.84	0.81	0.83	620

LYMPHOCTYE	0.97	0.92	0.94	620
AVG / TOTAL	0.83	0.78	0.78	2487

As shown in table 1, we obtain