Using Supervised Learning to classify White Blood Cell type based on microscopic images

Sasank Viswanadha, Srinivasa Vijay Bhaskar Vemuri

24 April 2019

Abstract

Blood and its components play an important place in human life as they are the best tool of indicator in determining many medical conditions. Identification of cell types from microscopic images is the first step in the diagnosis of several blood related diseases. In particular, the classification of white blood cells plays a crucial role in the diagnosis of several diseases like Hemochromatosis, Hemolytic anemia, Hemophilia. In this paper, we proposed several supervised learning approaches to classify the images into one of the four white blood cell types namely, Eosinophil, Lymphocyte, Monocyte, and Neutrophil. We implemented five different approaches - Multi-class Logistic Regression, Decision Tree Classifier, Multi-class Support Vector Machine, Naive Bayes Classifier and Convoluted Neural Network to classify the cell type based on image input. We compared the accuracy of these approaches in a comparative study of the described methods.

1 Introduction

Blood is a body fluid in humans that delivers necessary substances such as nutrients and oxygen to the cells and transports metabolic waste products away from those same cells. Blood is also important for regulation of the bodys pH, temperature, osmotic pressure, the circulation of nutrients and removal of waste, the distribution of hormones from endocrine glands, and the elimination of excess heat and contains components for blood clotting. Blood is made of of several components, including red blood cells, white blood cells, platelets, and the plasma, which contains coagulation factors and serum.

White blood cells, also called leukocytes (leuko, meaning white), make up approximately one percent by volume of the cells in blood. The role of white blood cells is to identify and target pathogens, such as invading bacteria, viruses, and other foreign organisms. White blood cells are formed continually; some only live for hours or days, but some live for years. They have nuclei and do not contain hemoglobin. The different types of white blood cells are identified by their microscopic appearance after histologic staining, and each has a different specialized function. The two main groups, are the granulocytes, which include the neutrophils, eosinophils, and basophils, and the agranulocytes, which include the monocytes and lymphocytes.

Neutrophil
Eosinophil
Basophil
(a) Granulocytes
Monocyte
Lymphocyte
(b) Agranulocytes

Usually, in hematology, microscopic analysis of peripheral blood smear results is time-consuming and costly process. Moreover, White blood cells often lead to misclassification and misidentification because they are inherently unstable. At the same time, the statistical bias and inconsistencies of the hematologist further slow down the evaluation of the results. For all the above stated reasons, the development and use of computer-based systems instead of traditional methods will greatly contribute to the acceleration of the analysis and produce more accurate results.

This paper focuses on the classification of white blood cell images based on their statistical and geometrical feature information using multiple techniques. Image analysis plays a crucial role in the diagnosis of many diseases in the medical field. In the results of the proposed project, we would like to provide the most suitable machine learning algorithm for the classification of white blood cell images.

2 Technical Approach

2.1 Multi-class Logistic Regression (MLR)

Regression analysis is a statistical method used to determine the relationship between two or more variables with cause-effect relationship and to make estimations or predictions on the subject by using this relationship. Logistic Regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). MLR is a model that is used to predict the probabilities of the different possible outcomes of a categorically distributed dependent variable, given a set of independent variables.

The LR model is a special form of general linear models obtained for the dependent variables with binomial distribution and is expressed as in the below equation.

$$\pi(x) = \frac{\alpha + \sum_{i=1}^{p} \beta_i x_i}{1 + e^{\alpha + \sum_{i=1}^{p}}} \tag{1}$$

Here, $\pi[x]$ represents the probability of an event being examined, α is a dependent variable constant, β_1 β_2, ...,β_p are the independent variables denoting the regression coefficients, x_1, x_2, ..., x_p are the arguments, p is the independent variable number and e is the error term.

The MLR model is the extended version of the LR model with two states, as shown in the below equation.

$$\pi_j(x_i) = \frac{\alpha_i + \sum_{l=1}^{p} \beta_{lj} x_{li}}{1 + \sum_{j=1}^{k-1} e^{\alpha_i + \sum_{l=1}^{p} \beta_{lj} x_{li}}} \tag{2}$$

Here, j_1,j_2,... ,j_k represents the k^{th} category, n represents the level of possible independent levels.

2.2 Naive Bayes Classifier (NBC)

The Nave Bayes Classifier is a simple probabilistic classification method based on Bayes' theorem (Thomas Bayes, 1702-1761). According to the Bayes' theorem, in the case of two random events X and Y occurring consecutively, the probability of the occurrence of the second event in the event of one of these two events can be represented by $P(X \cap Y)$. As with the below equation, the multiplication rule can be written with two different expressions;

$$P(X \cap Y) = \frac{P(X|Y)P(Y)}{P(X)} \tag{3}$$

The Bayes' theorem describes the relationship between an arbitrary X event due to a random process and conditional probabilities and marginal probabilities for another random event, Y.

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \tag{4}$$

The probabilities of the dependent situations likely to occur in any problem are calculated by the Bayes equation given above. In this equation, the $P(Y|X)$ expression represents the probability of the problem input, the probability of the P(Y) statement possible output state, and the $P(Y|X)$ expression represents the probability of Y output states versus the previous X input.

In the Naive Bayes classification technique, it analyzes the relationship between dependent and independent features to create a contingent probability from each relationship. To classify a new instance, an estimate is made by combining the effects of the independent variables on the dependent variable.

2.3 Decision Tree Classifier (DT)

The Decision Tree is a machine learning algorithm that can classify data by continually dividing the data set according to a certain criterion. A decision tree structure consists of roots, nodes, branches and leaves. The bottom part of the tree structure and the upper part of the leaves are called roots. Each feature in the data set represents nodes. The link between the nodes is called the branch. It is very important to decide which node to start partitioning in decision trees. If the appropriate node does not start, the number of nodes and leaves in the tree will be very high.

2.4 Support Vector Machine (SVM)

The Support Vector Machine is a machine learning algorithm based on the principle of structural risk minimization and based on convex optimization. It is mainly designed to solve binary classification problems. The aim here is to obtain a hyper-plane that will optimally separate the classes from each other. In the classification, it is usually represented by class labels such as -1, +1. The data to be classified can be separated linearly (AND/OR problem) or cannot be separated by a single line (XOR problem). Therefore, SVM is divided into two groups as Linear SVM and Nonlinear SVM depending on the data.

As is known, many classification problems in the real world consist of more than two classes. To solve such problems, a multi-class SVM classifier is needed. Multiple classification can be achieved by combining binary classifiers. If it is assumed that the training data consisting of numbers of samples for training of SVM in a linearly separable class classification problem is $\{x_i,\ y_i\}$ where $i = 1, 2, ..., n$, $y_i \in \{-1, +1\}$, $x_i \in R^d$ then the decision function of the optimal separation plane will be -

$$y_i = \begin{cases} w.x_i + b \geq +1 & +1 \\ w.x_i + b \leq +1 & \text{-1} \end{cases}$$

Where R^d represents the d-dimensional space of the input patterns $(x_i), y_i$ presents the labels where the inputs are classified as -1, +1. w represents the normal value of the hyper-plane, b represents the tendency (bias) value. In order to determine the optimal separation plane, the boundaries that are parallel to this correction must be determined. That is, support vectors are required. This procedure is expressed as $w.x_i + b = \pm 1$. As in the classification of medical images, it is not possible to separate the data linearly in many other image processing problems. In this case, it is possible to solve the problem by defining a part of the training data on the other side of the optimal hyper-plane by defining a positive artificial variable (ζ_i) the balance between maximizing the boundary value and minimizing the mis-classification errors can be controlled by identifying an edit parameter indicated by C, which takes positive values. The optimization problem for data that cannot be discriminated linearly by using the regulation parameter and the artificial variable is as in the below equation.

$$\min(\frac{\|w\|^2}{2} + C\sum_{i=1}^{r}\xi_i) \tag{5}$$

In order to solve the optimization problem expressed in Equation 11, the data that cannot be separated linearly in the input space is displayed in a multidimensional space defined as property space (Kavzaolu, T. lkesen, . 2010). Thus, the linear separation of data can be made and the hyperplane between classes can be determined. Nonlinear transformations can be made with the help of a kernel function, which is expressed as $K(X, x_i) = \Phi(X)\Phi(x_i)$ mathematically. As a result, the decision rule for the solution of a two-class problem that cannot be separated linearly using the kernel function can be written as

$$f(X) = sign(\sum_{i=1}^{k} \alpha_i y_i \phi(X)\phi(x_i) + b) \tag{6}$$

2.5 Convolutional neural network (CNN)

A Convolutional Neural Network (CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a CNN is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, CNN have the ability to learn these filters/characteristics.

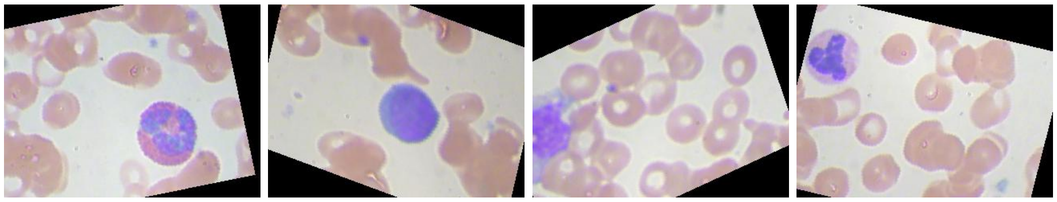
The architecture of a CNN is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

CNN's are able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and re-usability of weights. In other words, the network can be trained to understand the sophistication of the image better.

3 Experimental Results

3.1 Dataset

The dataset contains 12,500 augmented images of blood cells with accompanying cell type labels. There are approximately 3,000 images for each of 4 different cell types grouped into 4 different folders (according to cell type). The cell types are Eosinophil, Lymphocyte, Monocyte, and Neutrophil. This dataset is generated by augmenting the original dataset containing 410 images (pre-augmentation) as well as two additional subtype labels (WBC vs WBC) and also bounding boxes for each cell in each of these 410 images. Shown below are the sample images for each of the 4 classes, Eosinophil, Lymphocyte, Monocyte, and Neutrophil.



3.2 Analysis

We implemented the classical Learning models (MLR,DT,NBC and SVM) using the scikit-learn library with the Python API. All experiments were performed on a 64-bit Ubuntu 14.04 machine. We first down-sampled each of the (240X320) images in the dataset to size (60X80). We then considered the grayscale values of each of these images and reshaped them as a feature vector of shape (4800,). We then used PCA to reduce the number of features from 4800 to several values in the range 200-4000 and trained a Multi-class Logistic Regression model separately on all the features. The target labels used for training were the class values (0-3) corresponding to WBC. Table 1 gives the training and testing accuracy's after the features were reduced using Principal Component Analysis(PCA). Table 2 gives the training and testing accuracy's when the model was trained without reducing the features.

Table 1: Statistical measurements of classification success after reducing features using PCA

Features after PCA	**MLR**		**DT**		**NBC**		**SVM**	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
200	37.12	25.70	100	26.26	45.77	34.81	100	23.23
800	44.73	26.03	100	25.58	51.78	31.17	-	-
1400	52.30	24.86	100	24.81	54.17	28.64	-	-
2000	59.16	25.62	100	25.42	52.94	27.11	100	25.12
2600	66.78	25.34	100	24.85	53.30	25.66	-	-
3200	77.34	24.65	100	25.86	54.53	26.06	-	-
3000	100	25.66	100	25.62	56.37	25.74	100	27.67
4000	100	25.74	100	25.82	56.68	25.74	-	-

Note: All the above measurements are in %

Table 2: Statistical measurements of classification success without using PCA

Data	**MLR**	**DT**	**NBC**	**SVM**
Train (%)	100	100	35.73	100
Test (%)	25.38	27.27	27.35	25.10

3.2.1 Performance Metric

The accuracy of a model is defined as the ratio of the total number of images correctly classified to the total number of images in the datsaset. We used this accuracy as as a metric to compare the performance of the various models trained which allowed us to efficiently compare their performances.

3.2.2 Multi-class Logistic Regression

We trained a Multi-class Logistic Regression classifier with 0.00001 as the regularization parameter using the Limited-memory BFGS optimization technique. We observe that as the number of features increase the training accuracy seems to increase until it reaches 100%. We also observe that the testing accuracy oscillates in the range of 24% -27% with increasing number of features. This could be because the features extracted are not complex enough to allow us to accurately differentiate between each of the 4 classes. This is a direct consequence of the similarity between the the size and shapes of the different WBC cell types.

3.2.3 Multi-class Support Vector Machine

We trained a Support Vector Classification(SVC) model with a linear kernel and 1 as the error term. We observe that as the number of features increase the training accuracy is always 100% irrespective of the number of parameters. This is because SVM is better at classifying linearly separable high dimensional data compared to logistic regression model. We also observe that the testing accuracy oscillates in the range of 23% -27% with increasing number of features. Again, This could be because the features extracted are not complex enough to allow us to accurately differentiate between each of the 4 classes. This is a direct consequence of the similarity between the the size and shapes of the different WBC cell types.

3.2.4 Naive Bayes Classifier

We observe that as the number of features increase the training accuracy seems to increase until it reaches from 45% to 57%. We also observe that the testing accuracy oscillates in the range of 23% -34%. Naive Bayes classifier does not produce good results on this dataset. This could be because the feature vector does not obey the Naive Bayes assumption.

3.2.5 Decision Tree Classifier

We observe that as the number of features increase the training accuracy is always 100% irrespective of the number of parameters. We also observe that the testing accuracy oscillates in the range of 25% -6%. The reason for the low testing accuracy is the same as for the other models.

3.2.6 CNN

Figure 1 illustrates the architecture of the CNN used. It has 2 Convolution layers each with a filter size of (3X3X32), 2 Dense layers, 2 Dropout layers and one Max Pooling layer. The model is trained using the individual images (60X80) of the WBC's and their corresponding one-hot encoded class labels.

Plot 1 illustrates the accuracy of the CNN with respect to the number of epochs. We could observe from the graph that as the number of epochs increases, both the training and validation accuracy increases.

Figure 3 represents the confusion matrix obtained on the dataset of WBC images. As we can see from the figure, Lymphocytes were accurately predicted by the CNN classifier and there was some error in classifying true labels of Eosiophils. Around 32% of the Monocytes are misclassified as Neutrophils. This is because of the structure of Monocytes and Neutrophils are similar and the noise in the microscopic images led to the misclassification.

Figure 4 represents the plot of Model accuracy and Model loss for the dataset. We could observe that the training and testing accuracy increases with the increase in the number of epochs and similarly, the model loss decreases as the number of epochs increases.

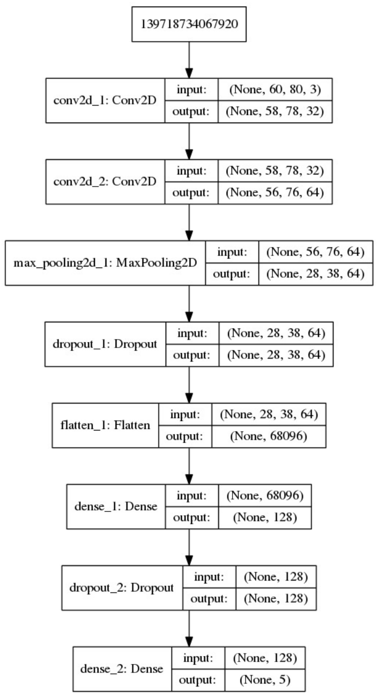


Figure 1: Architecture of CNN

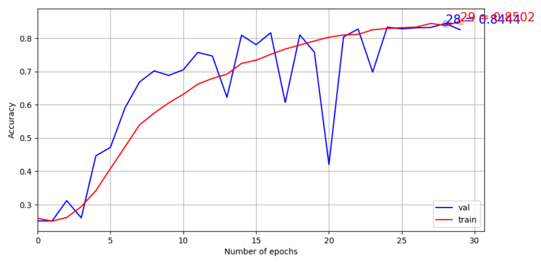


Figure 2: Accuracy of CNN

4 Conclusion

The results obtained from the experiments described in the previous sections suggest that the Convolution Neural Network is the best model to use for the problem of WBC image classification. We achieved a testing accuracy of 82.59% for the CNN model with architecture illustrated in Figure 1. The other classification algorithms fail in giving us promising results. We believe this could be because of the inherent similarity of the grayscale values between images from different classes. This is because the images of different classes are difficult to differentiate when we only consider standard features such as the grayscale values.

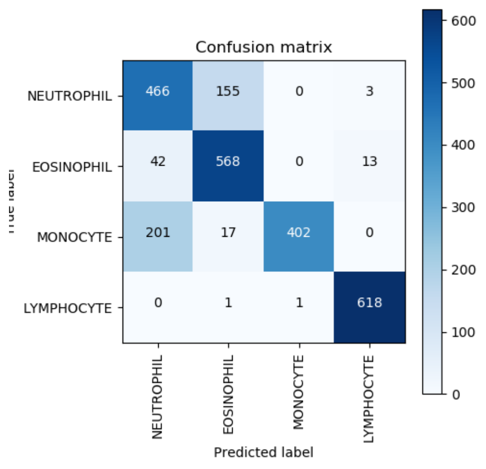


Figure 3: Confusion Matrix

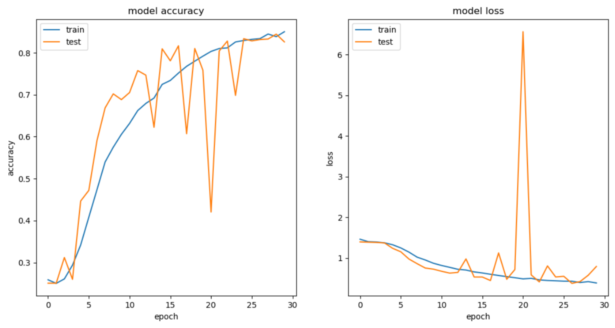


Figure 4: CNN Model accuracy and Model loss

5 Participants Contribution

Sasank Viswanadha was primarily responsible for preprocessing the images, performing PCA on the image dataset. He implemented Decision Tree and SVM classifiers,including training, testing and hyper parameter tuning.

Srinivasa Vijay Bhaskar Vemuri was primarily responsible for setting up the project, literature review and constructing and training generative networks. He implemented Logistic Regression and Naive Bayes classifier, including training, testing and hyper parameter tuning.

Both Sasank and Bhaskar, worked on the CNN classifier, hyper-parameter tuning for the CNN classifier and contributed equally in generating the results, interpreting them and producing the final report and results.

6 References

- Elen, Abdullah, and M. Kamil Turan. ”Classifying White Blood Cells Using Machine Learning Algorithms.” Uluslararas Mhendislik Aratrma ve Gelitirme Dergisi 11.1: 141-152.
- Adjouadi, M., Zong, N. Ayala, M. (2005). Multidimensional Pattern Recognition and Classification of White Blood Cells Using Support Vector Machines. Particle Particle Systems Characterization, 22(2): pp. 107-118.

- Bikhet, S. F., Darwish, A. M., Tolba, H. A. Shaheen, S. I. (2000). Segmentation and classification of white blood cells. IEEE International Conference on Acoustics, Speech, and Signal Processing, stanbul, pp. 2259-2261.
- Joshi, M. D., Karode, A. H. Suralkar, S. R. (2013). White Blood Cells Segmentation and Classification to Detect Acute Leukemia. International Journal of Emerging Trends Technology in Computer Science (IJETTCS), 2(3): pp. 147-151
- Logistic Regression - https://en.wikipedia.org/wiki/Logistic_regression
- Support Vector Machine - https://en.wikipedia.org/wiki/Support-vector_machine
- Convolutional Neural Network https://en.wikipedia.org/wiki/Convolutional_neural_network
- Decision Tree https://en.wikipedia.org/wiki/Decision_tree
- Naive Bayes Classifier https://en.wikipedia.org/wiki/Naive_Bayes_classifier
- scikit-learn - https://scikit-learn.org/stable/documentation.html
- keras - https://keras.io