# Image Classification of White Blood Cells Using Convolutional Neural Network

Juliet Cagampang[1,2] and Ligaya Leah Figueroa[1,3]

[1] University of the Philippines, Diliman, QC 1101, Philippines,
[2] jpcagampang@up.edu.ph
[3] llfigueroa@up.edu.ph

**Abstract.** Classification of white blood cells (WBCs) is a crucial process in medical diagnosis and research. Automated image classification of white blood cells using machine learning techniques provides faster and more accurate results compared to manual procedures. Convolutional Neural Networks (CNNs) are deep neural systems widely used in the medical classification tasks since they are excellent in feature extraction. This paper used the small Inception or MiniGoogleNet, a simple CNN, to classify five types of white blood cells, namely, basophils, eosinophils, lymphocytes, monocytes, and neutrophils. The quality of the dataset used to train, validate, and test the CNN classifier highly affects its performance. Hence, this paper presents a dataset preprocessing system that involves image processing to improve the quality of images and data augmentation using Albumentations to solve the problem of the highly imbalanced dataset. The model was trained from scratch using the Pytorch library and achieved an accuracy of 97.65%, recall of 94.12%, precision of 94.17%, and F1 score of 94.12%.

**Keywords:** White Blood Cells, Albumentations, Convolutional Neural Networks, Small Inception

## 1. INTRODUCTION

White blood cells (WBCs), also known as leukocytes, play an important role in the body's immune system, protecting against infections and diseases. They are produced in the bone marrow and travel throughout the bloodstream and lymphatic system, constantly monitoring the body for pathogens and abnormal cells. There are different types of white blood cells having unique characteristics and functions, but they work together to defend the body against infections and maintain overall health. Five types of WBCs are basophils, eosinophils, lymphocytes, monocytes, and neutrophils whose sample images are shown in Figure 1. The basophils are responsible for allergic responses and the immune reaction to parasitic infections. They release histamine and other chemicals that fight inflammation and attract other immune cells to the infection site. Eosinophils are essential in fighting parasitic infections and regulating allergic reactions. They release toxic substances to kill parasites and control inflammation. The lymphocytes are helpful in the adaptive immune system, producing antibodies and memory cells to combat some pathogens. Monocytes are


J. Caro et al. (eds.), Proceedings of the Workshop on Computation: Theory and Practice (WCTP 2024), Atlantis Highlights in Computer Sciences 23,
https://doi.org/10.2991/978-94-6463-684-0_20

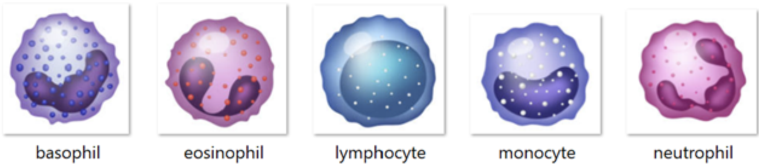



**Fig. 1.** Sample images of the 5 types of white blood cells.

large white blood cells that defend the body against infections, clearing dead cells, and regulating inflammation. They engulf and destroy pathogens, cellular debris, and dead cells. Neutrophils are the most abundant type of white blood cells. They are the first to respond to infections. They also engulf and eliminate bacteria, fungi, and other foreign particles, releasing toxic substances to kill pathogens.

## 2. RELATED LITERATURE

There has been a number of works on WBCs image classification utilizing several machine learning techniques, one of which incorporates nature-inspired optimization methods to enhance the accuracy and efficiency of the classification process [1]. It utilizes Enhanced Naive Bayes Ant Colony Optimization (ENBACO) for efficiently classifying and identifying blood cell images. The most crucial part of their work was the collection of digital microscopic pictures of the stained blood cells, the segmentation, the extraction of geometrical characteristics for each segment, and the classification. The experimental findings using ENBACO were compared with the researchers' other current approaches, and found the ENBACO method better obtaining 99.6% accuracy, 93% precision, 97% recall, and 98% F1 score. One study recommended an automated system for counting white blood cell differentials from bone marrow smear images utilizing a dual-stage convolutional neural network [2]. The authors utilized 2,174 patch images for both training and testing purposes. The dual-stage CNN managed to categorize images into 10 classes spanning the myeloid and erythroid maturation series, achieving an accuracy of 97.06% accuracy, 97.13% precision, 97.06% recall, and 97.1% F1 score. Their method was capable of successfully classifying raw images without the need for single-cell segmentation or manual feature extraction. Another study presented an innovative approach that involved extensive preprocessing combined with data augmentation techniques to create a more significant feature set for improved outcomes [3]. This approach integrates both traditional deep learning and advanced transfer learning models, including ResNet, VGG16, MobileNet, and InceptionV3. Using ResNet results in 74.58% accuracy, 80.75% precision, 73.5% recall, and 73.5% F1 score. Using VGG16 results in 96.09% accuracy, 96.25% precision, 96% recall, and 96% F1 score. Using MobileNet results in 78.47% accuracy, 78.25% precision, 78% recall, and 77.5% F1 score. Using InceptionV3 results in 97.2% accuracy, 97% precision, 97% recall, and 97.25% F1 score. Using their proposed customized CNN results in 99.86% accuracy, 98.75% precision, 98.75% recall, and 98.75% F1 score.

# 3. DATA AND METHODOLOGY

The WBC dataset obtained from Kaggle [4] is highly imbalanced containing 301 images of basophils, 1,066 images of eosinophils, 3,461 images of lymphocytes, 795 images of monocytes, and 8,891 images of neutrophils. After investigating the original dataset, it was found that some of the images are not of good quality, so they were discarded. Hence, the total number of original images used for each type is 301 for basophils, 1,064 for eosinophils, 3,440 for lymphocytes, 795 for monocytes, and 7,448 images for neutrophils. To solve this problem, data augmentation is necessary. There are several data augmentation techniques available but one particular technique that stands out is Albumentations available in Python.

Albumentations is a computer vision tool that enhances the performance of deep convolutional neural networks. It provides a simple but powerful interface for image classification, segmentation, and detection tasks. It is widely used in industries, deep learning research, machine learning competitions, and open source projects [5]. It has more than 60 diverse transformations including rotations, translations, scaling, shearing, brightness, contrast, and blurring. Applying these transformations to the dataset will enhance the robustness and generalization capability of deep learning models. Albumentations was found to have more image transformation operations that are the fastest compared to other tools [6]. Fast processing is important especially when working with large datasets. It was also thoroughly tested to ensure the accuracy and reliability of its operations in order to preserve image quality and avoid potential errors [7]. A remarkable advantage of Albumentations is its capability to combine multiple transform operations into a single pipeline. It allows applying a series of transformations to an original image, resulting in having more varied images. Hence, more unique images will be generated for classes with few available images.

GoogleNet or Inception V1 was first proposed in the paper of Szegedy et al. [8] at Google Inc. This CNN architecture emerged as the winner of the image classification challenge at ILSVRC 2014. It achieved a significant decrease in error rate compared to earlier winners such as AlexNet (winner of ILSVRC 2012), ZF-Net (winner of ILSVRC 2013), and VGG (the runner-up in 2014). It uses a combination of 1x1 and 3x3 convolution pathways in the middle of the architecture and global average pooling. This was used in various image classification problems and was found to be very efficient with high accuracy. Small Inception or MiniGoogleNet, a mini-version of GoogleNet was adapted for the CIFAR10 dataset in the paper of Zhang et al. [9]. It achieved the highest test accuracy compared to small AlexNet and standard multi-layer perceptrons (MLPs). The number of trainable parameters of the MiniGoogleNet is fewer compared with GoogleNet, hence, it has a significantly low computational cost. For these reasons, MiniGoogleNet was used in this study. The last fully connected part of the network was revised to have 5 outputs only instead of 10 in the original architecture. This is because the WBC dataset has 5 classes only.

# 4. RESULTS AND ANALYSIS

Figure 2 shows sample results of applying 13 transformations to the original white blood cell images where on the first column are five samples of original white blood cell images, one for each class.

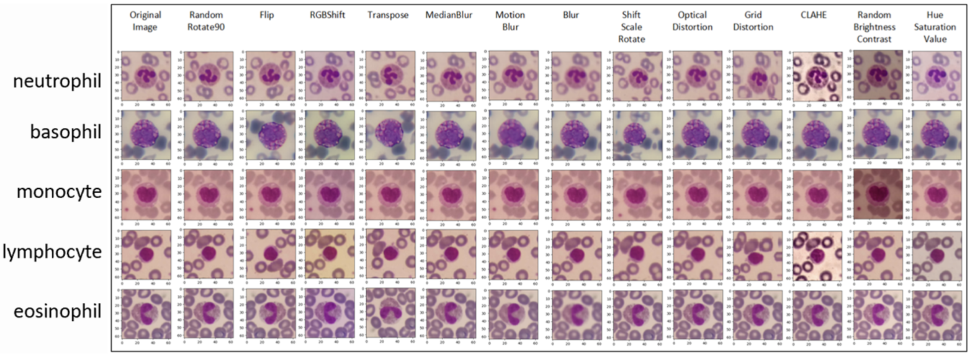



**Fig. 2.** The 13 Albumentations transformations applied to the WBC images for data augmentation.

## 4.1. Data Augmentation

The number of original images in each class is split into 80% and 20% for training dataset and testing dataset respectively. Table 1 shows the summary of the number

**Table 1.** Summary of the WBC dataset after performing data augmentation.

| Classes | Training Dataset | | | Testing Dataset | | |
|---|---|---|---|---|---|---|
| | **Number of Original Images** (from Kaggle) | **New images added using data augmentation** (Albumentations) | **Training (80%)** | **Number of Original Images** (from Kaggle) | **New images added using data augmentation** (Albumentations) | **Testing (20%)** |
| Basophil | 241 | 5,759 | 6,000 | 60 | 1,440 | 1,500 |
| Eosinophil | 851 | 5,149 | 6,000 | 213 | 1,287 | 1,500 |
| Lymphocyte | 2,752 | 3,248 | 6,000 | 688 | 812 | 1,500 |
| Monocyte | 636 | 5,364 | 6,000 | 159 | 1,341 | 1,500 |
| Neutrophil | 5,958 | 42 | 6,000 | 1,490 | 10 | 1,500 |

of original images placed in the training and testing datasets for each white blood cell class and the number of new images added for each class in the training and testing datasets through data augmentation using Albumentations. The original images for each class are first split into training and testing datasets before performing data augmentation in each dataset separately to make sure that there will be no similar images in the training and testing datasets. After data augmentation, a total of 30,000 images were obtained for the training dataset, where each class has 6,000 images and a total of 7,500 images were obtained for the testing dataset, where each class has 1,500 images.

### 4.2. Data Preprocessing

The dataset original cell images have different sizes and rectangular in shape, hence, all images were transformed into square shape by padding and applied centercrop transformation of 75% to 93% depending on the image to obtain the region of interest (ROI) after which they are resized to $64 \times 64 \times 3$. Next, each image is passed to a pipeline of 10 out of 13 Albumentations transformations, namely, RandomRotate90, Flip, RGBShift, Transpose, MedianBlur, MotionBlur, Blur, ShiftScaleRotate, OpticalDistortion,. GridDistortion, CLAHE, RandomBrightnessContrast, HueSaturationValue, to obtain numerous unique images for each of the 5 classes. The training and testing datasets consist of image-label pairs. Each pair is a list of an image and its true label. The labels are 0 for class basophils, 1 for class eosinophils, 2 for class lymphocytes, 3 for class monocytes, and 4 for class neutrophils. The final WBC training and testing dataset files are stored as a PT (Place-Text format) file (.pt file extension) [10] in Google Drive for fast data access, both for saving and loading data files. PT file is a file format used in Pytorch library which can be easily imported to a variety of machine learning applications such as natural language processing or image identification.

The image preprocessing steps to improve the quality of the training and testing datasets are as follows:

1. Transform original cell images to $240 \times 240$ resolution and convert from jpg format to png format for better image quality.
2. Transform images into square shape by padding, since they have different sizes and are rectangular in shape.
3. Use centercrop transformation of 75% to 93% depending on the image to obtain the region of interest (ROI).
4. Resize to $64 \times 64 \times 3$ to fit the model's input layer.
5. Each image is passed to a pipeline of 10 out of 13 Albumentations transformations shown in Figure 2 to generate more unique images for data augmentation.
6. Image enhancement and normalization using Pytorch functions.

### 4.3. Training Parameters

The training parameters used are a learning rate of 0.01 with StepLR to decay the learning rate every 10 epochs, the number of epochs is 113, batch size is 64, dropout is 0, cross-entropy loss as a criterion, and SGD as the optimizer with momentum=0.9. The Pytorch scripts in the Jupyter notebook were run on Google Colab using T4

GPU and ACER laptop with Intel i5 processor and NVIDIA Geforce RTX 3050 GPU.

### 4.4. Evaluation Metrics

The evaluation metrics used to assess the performance of the model are accuracy, precision, recall, and F1 score. Accuracy is the proportion of correct predictions, both true positives and true negatives, out of the total predictions. Precision, also called positive predictive value, is the proportion of true positive predictions out of all positive predictions. Recall is the proportion of actual positives that are correctly identified. F1 score is the harmonic mean of precision and recall, balancing the two. The formulas for these metrics are given in equations 1 to 4 where FP (false positives), TP (true positives), FN (false negatives), and TN (true negatives) were obtained from the confusion matrix of the model's performance result.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{1}$$

$$Precision = \frac{TP}{TP+FP} \tag{2}$$

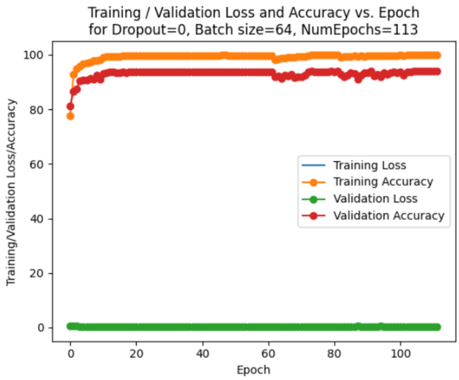
$$Recall = \frac{TP}{TP+FN} \tag{3}$$

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

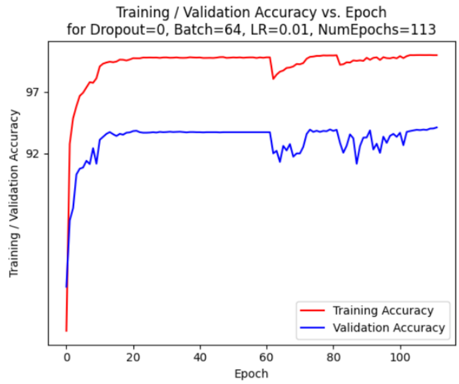
The plot of the training results with validation are shown in Figure 3a. The training accuracy is 99.99, training loss is 0.000, validation accuracy is 94.12, and validation loss is 0.323. Figure 3b shows the plot of the training and validation accuracy while Figure 3c shows the plot of the training and validation loss. The confusion matrix is shown in Figure 3d and the TP, FP, TN, and FN for each class are shown in Table 2. Table 3 presents the model's performance evaluation on the 5

**Table 2.** TP, FN, FP, TN of the 5 classes where B-basophil, E-eosinophil, L-lymphocyte, M-monocyte, and N-neutrophil.

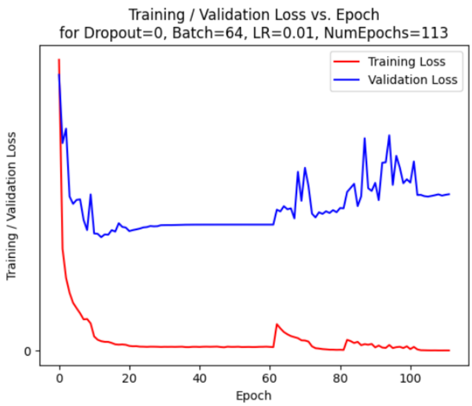
| Class | TP | FN | FP | TN |
|---|---|---|---|---|
| B=0 | 1497 | 3 | 41 | 5959 |
| E=1 | 1395 | 105 | 80 | 5920 |
| L=2 | 1374 | 126 | 109 | 5891 |
| M=3 | 1396 | 104 | 171 | 5829 |
| N=4 | 1397 | 103 | 40 | 5960 |



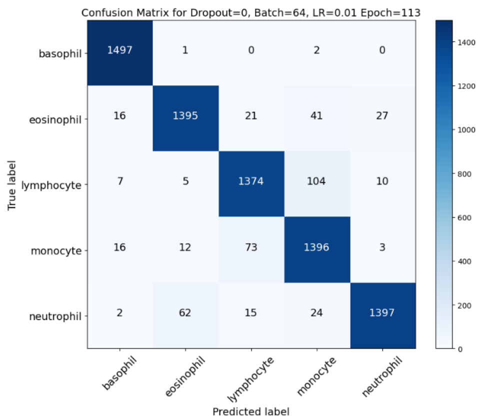

(a) Training results of the model. Plots of the training accuracy and loss, validation accuracy and loss. The training accuracy is 99.99 training loss is 0.000, validation accuracy is 94.12, validation loss is 0.323 at Epoch 113.




**(b)** Plot of the training and validation accuracy.




**(c)** Plot of the training and validation loss.




**(d)** Confusion matrix of the model.

**Fig. 3.** Performance Metrics Figures

**Table 3.** Performance evaluation of the model.

| Class | Performance Evaluation Metrics | | | |
|---|---|---|---|---|
| | Accuracy | Recall (Sensitivity) | Precision | F1 Score |
| Basophil | 99.41% | 99.80% | 97.33% | 98.55% |
| Eosinophil | 97.53% | 93.00% | 94.58% | 93.78% |
| Lymphocyte | 96.87% | 91.60% | 92.65% | 92.12% |
| Monocyte | 96.33% | 93.07% | 89.09% | 91.03% |
| Neutrophil | 98.09% | 93.1% | 97.22% | 95.13% |
| **Overall** | **97.65%** | **94.12%** | **94.17%** | **94.12%** |

classes using the metrics accuracy, recall, precision, and F1 score. From Table 3, the overall testing accuracy of the model is 97.65%, recall is 94.12%, precision is 94.17%, and F1 score is 94.12%. Finally, Table 4 shows the comparison of the performance and methods used in this study with other works. From Table 4 we can see that the

**Table 4.** Performance comparison with other works.

| WBC Image Classification | Accuracy | Recall | Precision | F1 Score | CNN classifier | Image Preprocessing |
|---|---|---|---|---|---|---|
| Kumod Kumar Gupta et. al [1] | **99.60%** | 93.00% | 97.00% | 98.00% | Enhanced Naive Bayes Ant Colony Optimization (ENACO) | Image preprocessing |
| Jin Woo Choi et. al [2] | 97.06% | 97.06% | 97.13% | 97.10% | Dual-stage VGGNet | No segmentation, no feature extraction |
| Oumaima Saidani et. al [3] | | | | | | |
| Proposed customized Deep CNN | **99.86%** | **98.75%** | **98.75%** | **98.75%** | Proposed customized Deep CNN | Extensive preprocessing & data augmentation |
| ResNet50 | 74.58% | 80.75% | 73.50% | 73.50% | ResNet50 | |
| VGG16 | 96.09% | 96.25% | 96.00% | 96.00% | VGG16 | |
| MobileNet | 78.47% | 78.25% | 78.00% | 77.50% | MobileNet | |
| InceptionV3 (GoogleNet) | 97.20% | 97.00% | 97.00% | 97.25% | InceptionV3 (GoogleNet) | |
| **Proposed Method** | **97.65%** | 94.12% | 94.17% | 94.12% | MiniGoogleNet | Preprocessing & data augmentation using Albumentations |

model used in this study ranks third among 8 WBC classifiers in terms of accuracy.

## 5. CONCLUSION

This study used the small Inception model or MiniGoogleNet, to classify five types of white blood cells, namely, basophils, eosinophils, lymphocytes, monocytes, and neutrophils. The dataset obtained from Kaggle is highly imbalanced which would result in a biased model and inaccurate predictions. Hence, data augmentation was performed using Albumentations to generate more unique images using 10 out of 13 transformation operations. This resulted in a balanced dataset, having equal number of images for each class type. In addition, the images in the dataset were enhanced by image processing to improve the model's performance. The model achieved an accuracy of 97.65%, a recall of 94.12%, a precision of 94.17%, and an F1 score of 94.12%, ranking third among 8 classifiers, 7 of which are from other works shown in Table 4. The high performance of the model proves that the approach used in this study, in dataset preparation involving image processing, data augmentation using Albumentations technique, and choosing a simple but powerful CNN, the small Inception, actually works. Furthermore, it is recommended to increase the number of original images for classes with few original images, increase the number of image samples for each class in the training dataset, and explore other CNN models.

## References

1. Gupta, K. K., Kamalraj, R., Gupta, R., Yadav, S., Blood Cell Image Classification Using the Machine Learning Methods With Nature Inspired Optimization. https://www.ijisae.org/index.php/IJISAE/article/view/3019/1598.
2. Choi, J. W., Ku, Y., Yoo, B. W., Kim, J.A., Lee, D. S., Chai, Y. J., Kong, H. J., Kim, H. C., White blood cell differential count of maturation stages in bone marrow smear using dual-stage convolutional neural networks. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0189259.
3. Saidani, O., Umer, M., Alturki, N., Alshardan, A., Kiran, M., Alsubai, S., Kim, T. H., White blood cells classification using multi-fold pre-processing and optimized CNN model. https://www.nature.com/articles/s41598-024-52880-0.
4. White Blood Cell Dataset, https://www.kaggle.com/datasets/masoudnickparvar/white-blood-cells-dataset.

5. Albumentations, https://albumentations.ai/.
6. Buslaev, A., Parinov, A., Khvedchenya, E., Iglovikov, V., Kalinin, A., "Albumentations: fast and flexible image augmentations". https://arxiv.org/abs/1809.06839.
7. Top 5 Image Augmentation Tools for Computer Vision: A Guide, https://blog.roboflow.com/data-augmentation-tools/.
8. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., Going deeper with convolutions. Google Inc. https://arxiv.org/abs/1409.4842.
9. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O., Understanding Deep Learning Requires Rethinking Generalization. https://arxiv.org/abs/1611.03530.
10. PT File, https://fileinfo.com/extension/pt.