Automated Hematological Analysis: A Machine Learning Approach for Blood Cell Classification

Adam Finch, Pranay Reddy Gundala, Tyler Rowe

***Abstract*—This research introduces a novel machine learning model for the analysis of blood samples through image processing. The primary objective is to detect and classify individual cells within blood sample images, classifying the cells as either platelets, white blood cells (WBC), and red blood cells (RBC). The secondary objective further classifies WBC into specific subtypes: Eosinophils, Lymphocytes, Monoctyes and Neutrophils. The methodology implements convolutional neural networks (CNNs) for object detection and classification. We achieve 98.3 % accuracy for detection of WBCs, and 87.6 % accuracy in WBC subtype classification, showcasing the effectiveness of our proposed model in advancing automated hematological analysis.**

Keywords: convolutional neural networks (CNNs), cell classification, object detection

I. Introduction

HEMATOLOGICAL analysis is a critical part of diagnosing patients for a variety of medical issues such as early disease detection for diseases like cancer, assessing organ functions, and tracking disease progression. The more specific classification of white blood cells into four types: Monocytes, Lymphocytes, Neutrophils, and Eosinophils, has significant importance in the identification of infections, hematological disorders, and also monitoring treatment or disease progression. Diverging from medical diagnoses, the classification of cells and specific white blood cell types also play an important role in research and drug development.

Blood-based diseases impact hundreds of thousands of lives each year. Within the United States alone, it is projected that blood-based cancers will claim the lives of 57,380 people in 2023 [13]. Medical researchers are consistently searching for methods to improve the testing, diagnosis, and treatment of these devastating conditions. A frequently employed diagnostic tool for blood-based diseases is the Complete Blood Count (CBC) Test, which measures the levels of each specific type of cell within a blood sample [15]. This situation presents a compelling application for Machine Learning: training a model to accurately identify and classify blood cell types from an image of the patient's blood work. This application is the focus of our research project.

A. Complete Blood Counts in Practice

A CBC test evaluates both the total number as well as the characteristics of cell components in the blood. The tests are used not only for diagnosing hematological conditions, but also as a benchmark for general health. A variety of metrics are calculated during a CBC test.

1) White Blood Cell Counts: White blood cell counts are one of the most important features used in interpreting the results of a CBC. White blood cell counts are an important part of the immune system and each type of WBC have different functions within the body. An increase in WBC count has been shown to be associated with disease, inflammation, problems with bone marrow production, and can serve as a predictor of cardiovascular risk [19]. Total WBC counts as well as the counts of the observed platelets and RBCs are the primary goal of our model.

Furthermore, additional counts of the specific subtypes of WBCs can reveal more information and valuable insights into the health of a patient. Studies have shown that those with an increased amount of neutrophil and lymphocytes in their blood sample have an increased risk of cardiovascular disease such as hyperglucemia, hypertension, dyslipidemia, central obesity, hypertriglyceridemia, low HDL-cholesterol and high fasting glucose [2].

These counts are conducted by spreading a drop of blood over a glass slide, air drying, and then staining the blood with a Romanofsky stain to highlight the differences in blood cells [3].

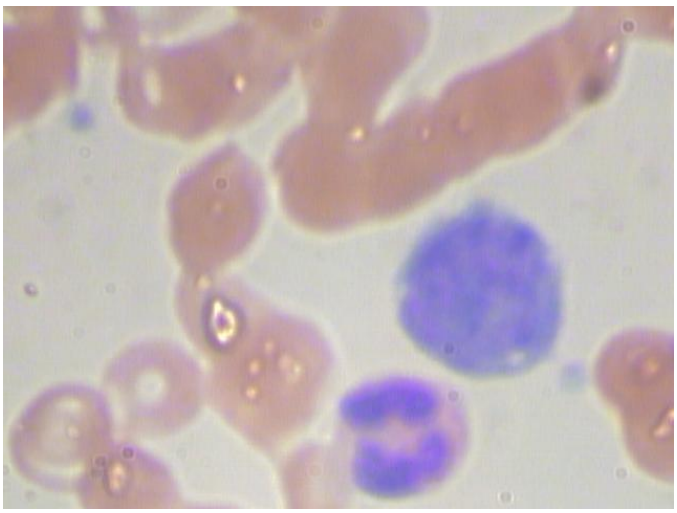

Fig. 1. Microscopic Image of Stained Blood Sample

2) Neutrophil-to-Lymphocyte Ratio: Neutrophils are the most common type of WBC and account for 50-70% of the WBCs in the circulatory system [17]. Neutrophils are immune cells that respond to infection while lymphocytes take part in immunity. The ratio can be used to show the balance between inflammation and immunity and can be a predictive measure

in cancer detection [23].

Classifying the different types of WBCs is the secondary goal of our model.

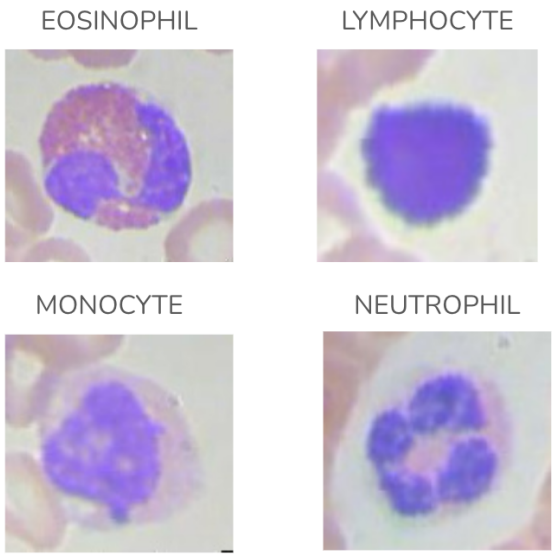

Fig. 2. Sample Images of Each WBC Subtype

B. Challenges and Risks Associated with Hematological Analysis

Traditionally, CBC tests were performed manually. The manual examination process starts after a blood sample is collected. After a blood sample is collected, it first sits until inspection or is sent to a laboratory for counting. Manually inspecting blood samples is not only time consuming, but susceptible to mistakes in classifications. Mistakes can occur with manual CBC tests for reasons such as improper labeling during classification, either incorrect classification or human error when labeling, or pre-analytical pitfalls in sample quality, such as excessive time from blood collection to analysis, which can result in decrease platelet count [10].

Recent technological advancements have created several automated methods for performance of CBC tests. The most common of these methods is hematology analyzer machines. Hematology analyzer machines use electrical impedence measurements to count different blood cell types and determine their sizes [6]. While these machines help to eliminate human error from CBC tests, a medical grade hematology analyzer machine can cost over $100,000 [6]. As a result, there is a large demand for cheaper and more efficient methods to count and classify white blood cells [12]. The need for a cost efficient and accurate automated solution for blood sample diagnosis is clear.

C. Approach

We introduce a machine learning approach to utilize convolutional neural networks (CNNs) to analyze blood sample images to achieve object detection and classification of platelets, red blood cells, and white blood cells. After the classification of the different cell types, our model then classifies the white blood cells into their four subtypes.

The dataset we are utilizing for our primary goal of cell classification is the BCCD Dataset, which includes blood sample images comprising of three cell types: platelets, WBC, and RBC [22]. We then use a reworked BCCD dataset for our secondary goal of classifying WBCs, which includes the same blood samples along with labels classifying each WBC into their correct subtype [16].

In this study, our fundamental question centers on automating the classification of blood cells types, including white bloods cells (WBCs), red blood cells (RBCs), and platelets, within stained blood sample images. Our approach involves bounding box object detection as well as the blood cell classification, achieved with Convolutional Neural Networks.

Our methodology begins with object detection, a critical part that isolates individual cells in an image. We used advanced image processing techniques and CNN-based algorithms. This detection step is a neccessity as it defines what regions our model must focus on and classify, discerning distinct cells within a complex and crowded stained blood sample.

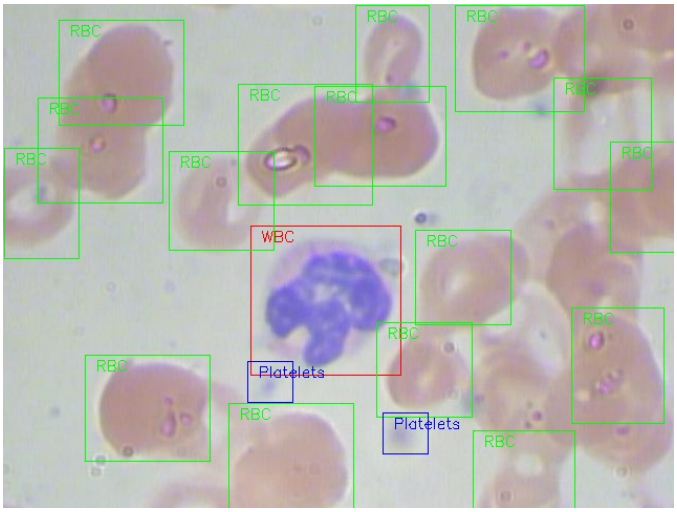

Fig. 3. Sample Labeled Image from BCCD Dataset

Following successful cell detection, our model transitions into the classification phase. Each segmented cell is analyzed to determine its cell type, either RBC, WBC, or platelet. The trained CNN, having recognized the distinct features and patterns such as size, shape, and color, of the outer parts of the cell as well as the size, shape, and color of the inner parts of the cell.

We then extend our classification model to address the specific subtypes of white blood cells. Using the modified version of the Kaggle BCCD Dataset, which includes classifications of the desired WBC types, our model is trained to discern the differences of the WBCs [16].

This approach, integrating object detection and classification, is a notable contribution to the field of automated hematological analysis. By using CNNs for both goals, through training on the original and modified dataset, our study advances

automated hematological analysis by not only classifying a single cell, but detecting multiple cells in stained blood sample and classifying them by their distinguished features. This study has major contributions to the application of automated hematological analysis, enabling future programs to be used to observe an entire blood sample and return insights about multiple characteristics about the blood sample to then be explained by a medical professional where diagnoses and conclusions can be made. The application of machine learning methods have the potential to transform the healthcare industry by reducing costs and increasing efficiency of diagnosing procedures.

D. Related Work

Significant research has already been conducted on automated methods for blood cell classification. Lou et. al. developed a system for red blood cell counting through image segmentation. They chose to apply Spectral Angle Mapping methods and were able to achieve a counting accuracy of 98% [14]. Similarly, Zhao et. al. proposed an automated system for white blood cell detection and classification, integrating a color-based detection algorithm and a granularity feature with Support Vector Machine (SVM) for initial classification [14]. They then utilized convolutional neural networks to automatically extract high-level features for the remaining white blood cell types. Finally, Habibzedah et al. wrote about important preprocessing steps to improve bounding box performance; namely color distortion and image flipping mirroring [7].

Convolutional neural networks are a commonly used approach to classify cells. Kutle et al., in their study in 2020, proposed a method consisting of Regional Based Convolutional Neural Networks and Transfer Learning [11]. Their experimentation utilized CNN architecture of ALexNet, VGG16, GoogLeNet, and ResNet50. Their best performance was with ResNet50 with transfer learning, achieving an accuracy of 100% in determining WBC cells and 99.5% accuracy classifying Lymphocyte, 98.4% for Monocyte, 96.2% for Eosinophil, and Neutrophil with 95% accuracy.

Neelam et al. proposed a method utilizing clustering with K-means and expectation maximization algorithms to automate the count of each class of WBCs [20]. Features used include shape, color, and texture of the cytoplasm and nucleus of each cell. They also utilized neural networks and support vector machines for classification, resulting in an 80% accuracy in classifying WBCs.

Despite the plethora of research already completed in this field, our project endeavors to extend beyond existing studies. Rather than image segmentation, as used by Lou et al., we will attempt to use a bounding box method to isolate the white blood cells. The choice to utilize bounding boxes is due to the lower quality of some of the images in our dataset, as well as the fact that bounding boxes often have less computing needs than semantic segmentation models [12]. Bounding box-based models are also often faster and more memory-efficient, making them better suited for real-world deployment on low-resource devices [12]. This makes more sense for our desired application of real-time diagnosis in the medical field. We also aim to address potential challenges and limitations in the existing methodologies. In particular, we will attempt to address the problems of interference from surrounding cells, generalization to varied clinical settings, and the computational cost of high-level feature extraction [12].

II. Methods

A. Preprocessing

Preprocessing of images is one of the most important steps towards building an object detection model. Shahriar and Li found that image quality plays a crucial role in increasing object recognition or classification rates. They observed that object detection models with properly preprocessed training images significantly outperform similar models using unprocessed noisy images in terms of recognition or classification [21]. As such, we took great care to adequately address this step within our methodology.

One of the first preprocessing steps we took involved the implementation of Adaptive Histogram Equalization (AHE). A fundamental tenant of object detection models is edge detection. Contrast preprocessing serves to accentuate edges, making them more distinct, as adjacent pixels to the edge are emphasized.

AHE is implemented by first dividing an image into small regions. The size of these regions is a parameter within the algorithm. Within each region, a histogram of pixel intensities is generated; this provides an overview of the distribution of intensity values. Naturally, the most frequent pixel values correspond to the peaks in the histogram chart. The cumulative distribution function (CDF) is calculated based on the normalized histogram bin counts within each region. The algorithm then spreads out and distributes the dominant pixel values based on the CDF, aiming to balance the overall intensity distribution. Mathematically, the AHE algorithm can be represented as follows [18]:

1. Generate a histogram of pixel intensities and use the normalized bin counts to compute the CDF.

$$CDF_{x,y}(i) = \sum_{j=0}^{i} P(x, y, i)$$

The CDF corresponds to a specific pixel at the (x, y) position for a given intensity level i. It quantifies the likelihood that the intensity of a pixel at the (x, y) location will not exceed the value i.

2. Use a transformation function to spread out the pixel intensities across the entire available range. This ensures a more uniform distribution of intensities. In this equation, L represents the maximum intensity level within the image.

$$T(x, y, i) = \frac{CDF_{x,y}(i) - CDF_{\text{min}}}{1 - CDF_{\text{min}}} \times L$$

Final Equalized Image: original pixel value $\leftarrow T(x, y, i)$

AHE has the effect of improving contrast within an image by making dark regions even darker, and lighter regions even

lighter. In colored images, AHE often leads to even more vivid color representations of these images as well.

Along with contrast adjustment through AHE, we also employ image augmentation. Image augmentation involves applying various transformations to the training images to artificially increase the diversity of the dataset. This helps the model generalize better to unseen variations and conditions during inference. It also increases the size of our training data and ensures that our model is well equipped to handle different angles, rotations, and viewpoints of blood sample images.

Specifically, we chose to augment the BCCD dataset by flipping images horizontally and vertically, as well as randomly rotating images between -15° and +15°. An example of how AHE and augmentation techniques modified images in our dataset is shown below in Figure 4.

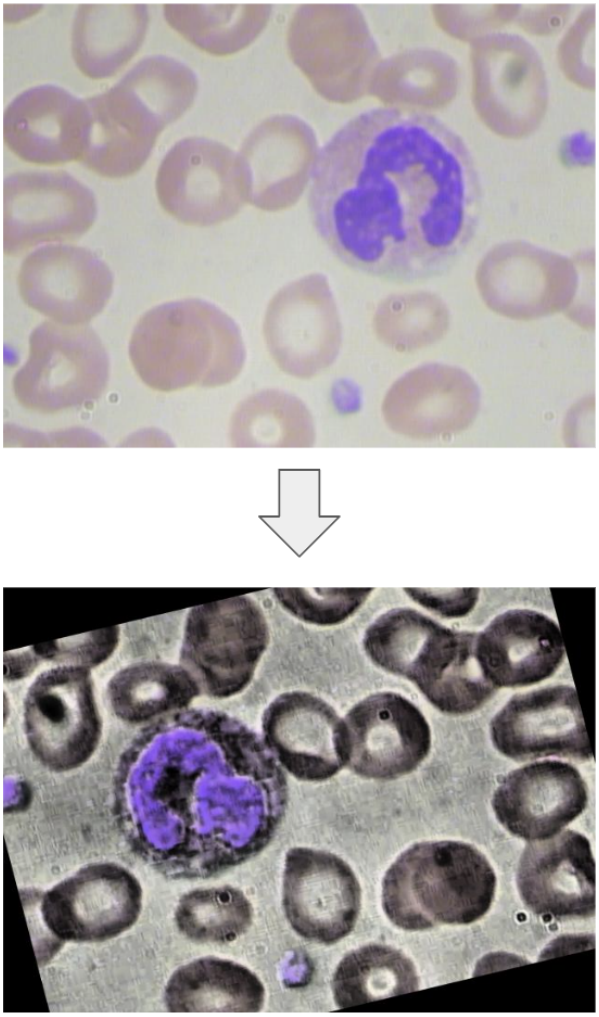

Fig. 4. Sample Image Before and After AHE and Augmentation

B. Bounding Box Object Detection Model

The methodology for our Bounding Box Object Detection Model makes use of Convolutional Neural Networks, specifically the YOLOv8 framework. YOLOv8 is the 8th version of Ultralytic's "You Only Look Once" Model for real-time object detection and image classification [9].

There are several reasons why we chose to use YOLOv8 to train our model. One of the most important factors is that once trained, a YOLOv8 model can provide almost instantaneous predictions for new images. Within the field of medical imaging, anything that can help to streamline the process of diagnosing and treating patients is invaluable.

Additionally, YOLOv8 can handle multiple classes simultaneously, which allows the model to pick out white blood cells even when surrounded by red blood cells and platelets. Detecting multiple objects at once is an important part of blood cell classification problems where there are multiple areas of interest within a single image. With YOLOv8, we can train a model to provide detailed bounding boxes for each of these areas, allowing for a complete count of blood cell objects.

Finally, YOLOv8 has been pretrained on a large dataset with a wide variety of object classes [9]. This means that it has exceptional versatility and adaptability; it can be fine-tuned based on a new set of images such as the BCCD dataset. This pretraining also means that YOLOv8 requires less data to train than models that are built from scratch. All of these advantages made the YOLOv8 framework a strong choice for the object detection portion of our project.

1) Model Architecture: At this time, there has not been a published academic paper providing a detailed explanation of YOLOv8. However, due to the project's open source nature, there is still ample documentation regarding the model's structure. Ultralytics, the team behind the model, is in the process of writing a paper for publication and provide frequent updates in their official GitHub repository [9].

The YOLOv8 model operates as a deep Convolutional Neural Network, passing through multiple stages in order to detect and classify objects. The input layer receives images and divides them into smaller regions; each region is responsible for detecting objects within its domain.

The model backbone for YOLOv8 contains 53 convolutional layers for feature extraction. The backbone for YOLO models is based on a modified version of CSPDarknet53, a model which employs Cross Stage Partial Connections (CSP). This process involves splitting the feature map into segments and merging them through a cross stage hierarchy process [4].

The neck network is responsible for fusing together high-level and low-level feature maps obtained through the backbone network. This has the effect of reducing the feature space and preparing the model to make bounding box predictions. After the neck network processes the features, they are passed into the head layers of the model .

The head of the model consists of fully connected layers which are responsible for predicting bounding boxes and class probabilities for each object detected in the image. The output of the model consists of bounding boxes, each with a distinct confidence score as well as a class label. Though complex, the YOLOv8 framework provides an excellent option for object detection problems such as classifying blood cells.

C. WBC Subtype Classification Model

Along with training an object detection model using YOLOv8, we also wanted to train and implement another

model to classify WBC subtypes in order to accomplish our secondary objective.

We experimented with several models to test which might be suitable for this problem. The model selection process and experimentation in this area is described in great detail in Section III of this report. However, in the Methodology section we wish to focus on the best-performing model: ResNet-50.

ResNet-50 is a deep convolutional neural network model developed by Microsoft Research [8]. It was trained on more than one million images from the ImageNet database [8]. Similar to the YOLOv8 model, the pretrained nature of ResNet-50 allows it to excel in image classification even without an extremely large amount of blood sample training images.

Convolutional Neural Networks were our preferred method for solving the WBC subtype classification problem due to Eosinophil, Lymphocyte, Monocyte, and Neutrophil cells being very similar to each other in appearance. This visual similarity made it necessary for a model to learn complex features from the input images in order to have classification success.

1) Model Architecture: He et. al. introduced the ResNet-50 model in their paper "Deep Residual Learning for Image Recognition" [8]. The key innovation of ResNet-50 is based on the idea of a residual learning framework, rather than a standard feed-forward neural network. Residual connections – or skipping connections – address the problem of vanishing gradients by allowing the model to learn residual mappings which capture the difference between the input and the desired output. Residual connections allow for the model to skip over layers in both forward and backward passes, promoting a more efficient method for training deep neural networks.

The "50" in ResNet-50 refers to the number of layers in the model. Specifically, there are 48 convolutional layers, 1 max pool layer, and 1 average pool layer. The convolutional layers are integrated with batch normalization and a ReLU activation function. These layers are responsible for feature extraction.

Following the convolutional layers are convolutional and identity blocks. In these blocks, residual functions are learned to map the input to its label. Finally, in the fully connected layers of ResNet-50, class probabilities are calculated and the final classification is made based off of a softmax activation function. Overall, ResNet-50 served as an attractive model framework for our secondary objective of classifying WBC subtypes into their four respective categories.

III. Experiments

Extensive experimentation was necessary in order to achieve robust performance for object detection and classification of blood sample images. This experimental process involved model selection, model training, and fine tuning of hyperparameters. Despite computational and time constraint limitations associated with deep learning models, our approach strikes a balance between computational efficiency and model accuracy.

A. Model Training: Objective 1

As mentioned previously, our first objective was to train a model to predict bounding box coordinates for WBC, RBC, and Platelets within a blood sample image. The python libraries ultralytics and roboflow were both extremely useful in this process. Through these libraries, we were able to take a custom labeled dataset of blood sample images and train a model based upon the YOLO v8 framework.

Roboflow offers seamless conversion for images and labels into the proper formatting needed to train YOLO models. Therefore, our first step to train our object detection model was to use Roboflow to separate our images into training, validation, and testing sets. Roboflow also facilitated in generating the YAML files necessary for the trianing of our YOLO model. Finally, once the images were preprocessed and prepared, the YOLO package within the Ultralytics library allowed us to begin the training process.

1) Hyperparameter Tuning: The YOLO v8 model contains a multitude of hyperparameters. However, training a model with any given set of hyperparameters took about 10 hours. This presented a unique challenge, as it was unrealistic to complete a full grid search to find the optimal hyperparameters for our object detection model. However, having an understanding of what each hyperparameter does allowed us to make educated decisions for what values to use for each hyperparameter based on the context of our specific problem and objectives. Table 1 summarizes the hyperparameters for the highest performing model that we trained.

TABLE I
Object Detection Model Hyperparameters

Parameter	Value
Learning Rate	0.001
Confidence Threshold	0.25
Non-Maximum Suppression Threshold	0.45
Intersection Over Union Threshold	0.5
Epochs	25

The learning rate determines the size of steps taken during the model optimization process. The choice of 0.001 attempts to strike a balance between convergence speed and not risking overshooting the optimal solution. While increasing the learning rate may allow for faster convergence, it also increases the risk of divergence or getting stuck at a local minimum.

The confidence threshold filters predictions based on the confidence score assigned by the model. Each predicted bounding box is assigned a confidence score, and predictions are only considered to be valid if the confidence score exceeds the confidence threshold. It was important for us to consider the tradeoff between recall and precision when setting this parameter in our model. A higher confidence threshold increases precision, reducing the number of false positives. However, this comes at the expense of recall, as more false negatives are likely to occur. Ultimately, a confidence threshold of 0.25 allowed us to maintain balance in relation to this tradeoff.

The non-maximum suppression threshold attempts to remove redundant bounding boxes from the final model predictions. It is undesirable for the model to bound the same object more than once, particularly in the context of counting objects like a CBC test. If the percentage of overlap between

two predicted bounding boxes exceeds the non-maximum suppression threshold, the box with the lower confidence score is discarded. The tradeoff with this hyperparameter lies in duplicate prevention versus ensuring valid predictions are not removed due to their close proximity with other objects. This is a real challenge with blood sample images, particularly with RBC detection. RBC are the most frequent blood cell type, and often are found in close proximity to one another, making overlap between bounding boxes plausible. We attempted to maintain balance with this tradeoff by setting the non-maximum suppression threshold at 0.45 for our model.

In the context of Neural Networks, an epoch refers to a complete forward and backward pass of all the training examples. Too few epochs lead to the model underfitting because it is not complex enough, and too many epochs lead to overfitting for the opposite reason. More epochs also take longer for the model to finish training, so there is a tradeoff with time as well. As with all hyperparameters, finding a balance was important to achieve the most success with our model.

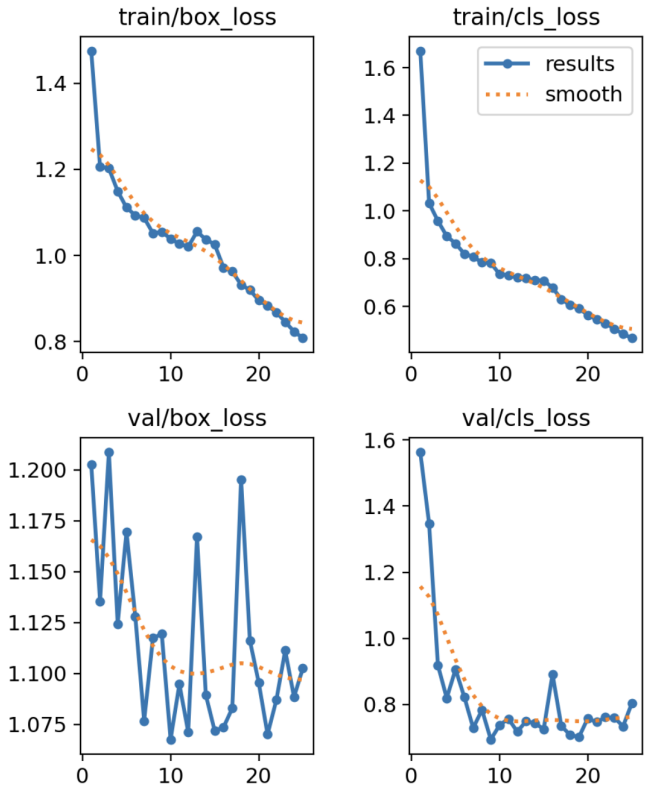

Fig. 5. Change in Box Loss and Classification Loss over 25 Epochs

B. Model Training: Objective 2

Our secondary objective was to train a model to accurately classify WBC subtypes: Eosinophil, Lymphocyte, Monocyte, and Neutrophil cells. Pre-trained models and transfer learning have significantly advanced the field of image classification. These models are initially trained on large and diverse datasets, such as ImageNet, to learn generic features and patterns from a wide range of images.

The knowledge gained during this initial training phase allows these models to develop a robust understanding of visual features, textures, and object hierarchies. Transfer learning then enables the pretrained models to be fine-tuned on specific tasks or datasets with limited labeled examples. This approach is particularly advantageous in scenarios where collecting a large amount of task-specific data is challenging. By leveraging the learned features from the general dataset, pretrained models demonstrate improved performance and faster convergence during the fine-tuning process. This makes them invaluable in various applications, ranging from computer vision tasks to real-world image classification problems, where balancing model accuracy and computational efficiency is crucial.

1) Pre-Trained Model Experimentation: Due to the relatively small size of our training data, we wanted to experiment with pre-trained convolutional neural network (CNN) architectures such as MobileNet, VGG (Visual Geometry Group), and ResNet (Residual Network).

MobileNet is a lightweight deep learning model designed for mobile and edge devices. It features depth-wise separable convolutions, where standard convolutions are split into depth-wise and point-wise convolutions. This architecture reduces computational complexity while maintaining accuracy, making it suitable for resource-constrained environments (SOURCE: MobileNet).

Residual Networks (ResNet) are known for introducing residual learning, addressing the vanishing gradient problem in deep neural networks. It utilizes residual blocks, allowing the model to skip connections and learn residual functions. This architecture facilitates the training of very deep networks, promoting better feature extraction and representation.

The Visual Geometry Group (VGG) architecture is characterized by its simplicity and uniform structure. It consists of repeated blocks of convolutional layers with small 3x3 filters, ReLU activations, and max pooling. This straightforward design facilitates understanding and implementation, making it a popular choice for image classification tasks.

We also used early stopping while training these models. Early stopping is a crucial technique in the training of machine learning models, including neural networks, to prevent overfitting and optimize generalization performance. This method involves monitoring the model's performance on a validation dataset during the training process. The training is halted once the performance on the validation set ceases to improve or starts degrading, indicating that the model has likely reached an optimal point in its learning process.

Early stopping serves as a regularization mechanism, preventing the model from memorizing the training data excessively and ensuring that it generalizes well to unseen data. By terminating the training process early, unnecessary computational resources are conserved, and the risk of overfitting, where the model performs well on the training data but poorly on new data, is mitigated. The choice of the optimal stopping point requires a careful balance, considering both training efficiency and the preservation of model generalization capabilities.

2) Traditional Machine Learning Models: After training and evaluating the pre-trained models we also tried typical machine learning algorithms for classification such as K-Nearest Neighbors (KNN), Random Forest, Support Vector Machine (SVM), and XGBoost.

XGBoost is an ensemble learning algorithm that excels in structured data tasks. It builds a series of decision trees sequentially, correcting errors of the combined ensemble through gradient boosting. Known for its efficiency and scalability, XGBoost is widely used in regression, classification, and ranking problems.

Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the aggregated result. It mitigates overfitting by training each tree on a random subset of features and data. This approach enhances robustness, making Random Forest a popular choice for various machine learning applications.

Support Vector Machine (SVM) is a supervised learning algorithm used for classification and regression tasks. It seeks to find the optimal hyperplane that separates different classes in the feature space. SVM is effective in high-dimensional spaces and can handle linear and non-linear relationships using kernel functions, making it versatile in various domains.

K-Nearest Neighbors (KNN) is a simple and versatile algorithm used for classification and regression. It predicts the class or value of a data point based on the majority class or average of its k nearest neighbors in the feature space. KNN is non-parametric and lazy, making it suitable for small to medium-sized datasets with no assumptions about data distribution.

3) Hyperparameter Tuning: Similar to the bounding box model, training many different Image Classification CNN models takes considerable computational power and time. As such, we were unable to perform an extensive grid search; however, we did experiment with different hyperparameters. The following hyperparameters were included in most of the pre-trained models we experimented with:

TABLE II
WBC SUBTYPE CLASSIFICATION MODEL HYPERPARAMETERS

Parameter	**Value**
input_shape	(224, 224, 3)
include_top	False
Weights	'imagenet'
model_trainable	False

The input_shape parameter specifies the shape of the input data that the model expects. In this case, the input is a 3 dimensional tensor with dimensions 224x224x3, where 3 represents the RGB color channels.

The include_top argument indicates whether to include the top (output) layers of the model. When set to False, it means we are using the MobileNetV2 as a feature extractor, excluding the final fully connected layers typically used for classification.

The weights parameter specifies the source of initial weights for the model. In this case, 'imagenet' means we are using the pre-trained weights on the ImageNet dataset. These pre-trained weights provide a starting point for the model, leveraging knowledge gained from training on a large and diverse dataset.

Finally, for all the pretrained models, we set the model_trainable parameter to 'False'. This effectively froze the weights of the pre-trained model, meaning that the parameters (weights and biases) of the layers in the model were not updated or fine-tuned during the training process of the overall model. By freezing the weights, we were able to more effectively preserve the pre-trained knowledge and prevent it from being overwritten during the initial training steps on the new task.

For MobileNet, we also applied average pooling following the final convolutional layer. Setting the pooling parameter to average causes the model to compute the average value of each feature map, and allows the model to obtain a global average pooling layer before the final classification layer.

In total we created five custom models using MobileNet, VGG, and ResNet:

1) In the first model, we used MobileNet as the model base with the above specified parameters and added a flatten layer on top of that. This layer is used to flatten the output from the previous layer into a one-dimensional array, and is often employed before fully connected layers. Then, we added a dense layer with 128 neurons with a ReLu activation function. We also made use of batch normalization and dropout layers after the dense layer. Batch Normalization is applied to normalize the activations of the previous layer, helping to stabilize and accelerate the training process. Dropout is a regularization technique that randomly sets a fraction of input units to zero during training, which helps prevent overfitting by reducing reliance on specific neurons. In this case, we set the dropout parameter to 0.5, so 50% of the neurons were dropped out during training. Finally, we added output layer with 4 neurons and a softmax activation function to classify the 4 types of white blood cells. The softmax activation function is used to convert the network's raw output into probabilities, indicating the likelihood of each class.

2) In the second model, we used VGG as the model base and again added a flatten layer on top of that. Then, we added 3 dense layers with 32, 16 and, 8 neurons respectively and a ReLu activation function. Again, we used batch normalization before every dense layer and dropout after every dense layer. To finish, we added an output layer with 4 neurons and a softmax activation function.

3) In the third model, we used ResNet as the model base and added GlobalAveragePooling2D instead of a flatten layer. The GlobalAveragePooling2D layer is a down-sampling operation that calculates the average value for each feature map across all spatial positions. It's commonly used as a replacement for the flattening operation before the fully connected layers in convolutional neural networks. This time, we did not add any extra dense layers in the model. Adding extra layers to an already deep model like ResNet is computationally very expensive and can lead to overfitting.

4) The fourth model is the same as third, except this time we used the original image size (240, 320, 3) as input shape. Also, we added one Convolutional Layer (Conv2D) before GlobalAveragePooling2D layer.

5) The fifth model is same as the fourth but without the additional one Convolutional Layer (Conv2D).

To compile all the models, we used Adam optimizer, a categorical crossentropy loss function evaluated performance with the accuracy metric. Adam optimizer is a popular optimization algorithm known for its adaptive learning rates and efficiency. The loss function is a measure of how well the model is performing during training. For multi-class classification problems, a categorical cross-entropy loss function is a common choice. It calculates the cross-entropy loss between the predicted probabilities and the true class labels.

The most successful model trained during this experimentation process was model 5: the ResNet-50 model using the original input image size without an additional Conv2D layer. Figure 6 shows a visual representation of the training loss and accuracy for this model across 8 epochs, after which early stopping prevented the model from being trained further due to a drop off in validation accuracy. Similar representations for all of the CNN model types trained can be found in Appendix A.

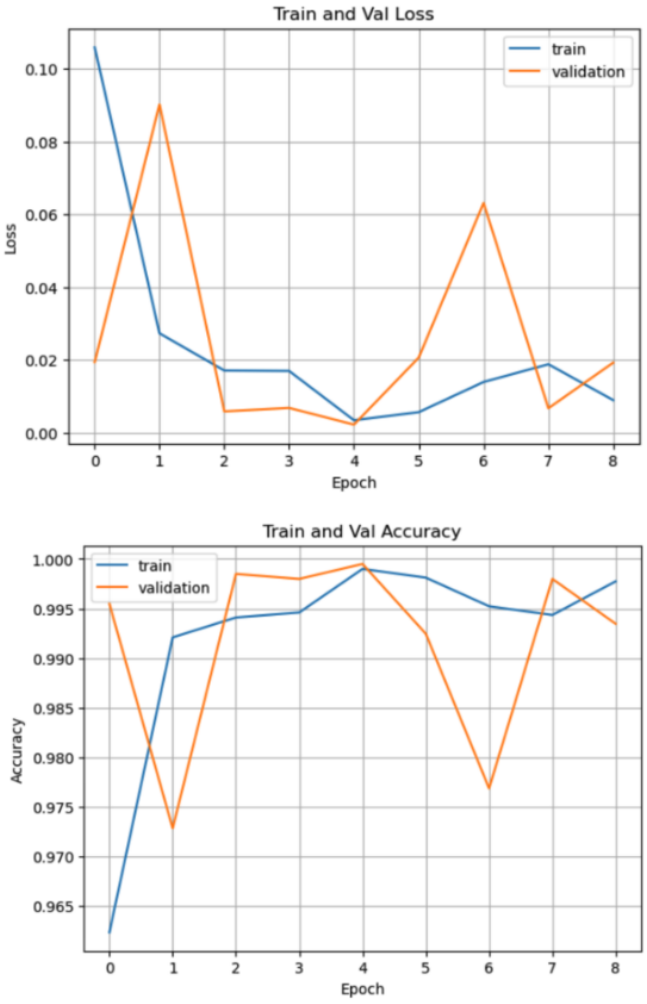

Fig. 6. ResNet-50 Model Loss and Accuracy over 13 Epochs

IV. Results

A. Data

Our primary objective is to train a model proficient in the detection and classification of three primary blood cell types: White Blood Cells, Red Blood Cells, and Platelets. To achieve this, we leverage the publicly available BCCD Dataset, which includes 410 blood sample images. The images are labeled with bounding box coordinates denoting where each WBC, RBC, and Platelet cell is located [22].

The dataset focuses on WBC in particular, ensuring that at least one WBC is present in each blood sample image. In a typical human blood sample, the average ratio of RBC to WBC is approximately 600:1 [24]. In the BCCD dataset, however, the ratio between RBC and WBC is approximately 10:1. This imbalance is due to the blood sample images focusing on areas where all three cell types are present.

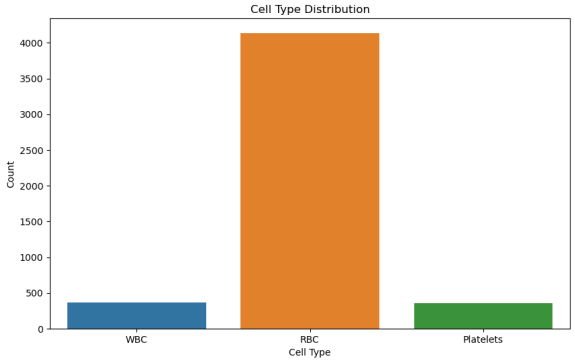

Fig. 7. Distribution of Cells in the BCCD dataset

Because the labels in the BCCD dataset are given in an XML metadata file, an early data exploration and preprocessing step was to generate a one-hot encoded CSV file which effectively maps each bounding box to the image it belongs to. Having all of the image names and labels in one place was essential to our exploratory data analysis. Additionally, it faciliatated the preparation of images to be fed into the input layer for a Convolutional Neural Network.

Aside from our primary goal of detecting WBC, RBC, and Platelets within blood sample images, we also aim to classify WBC into four different subtypes: Eosinophil, Lymphocyte, Monocyte, and Neutrophil. This type of WBC classification is essential for completing a Blood Differential Test, a valuable tool in helping to diagnose blood-based diseases.

To accomplish this secondary objective, we utilize a reworked BCCD dataset, which labels similar blood sample images based on the subtype of WBC present within the image [16]. The majority of images found in this dataset only contain one WBC, and all images with multiple WBCs present are always of the same subtype. Therefore, only one label is needed per image.

The reworked dataset is unbalanced towards Eosinophil cells. Figure 8 shows the distribution of WBC subtypes among the images within the dataset, as well as the frequency of images with more than one WBC. This unbalance between

subtypes is a concern we attempted to address by employing stratified sampling techniques for our training, validation, and test sets. Additionally, we performed similar preprocessing and image augmentation steps as described above for the first dataset to artificially generate more training data and ensure the model was trained using ample examples from each class.

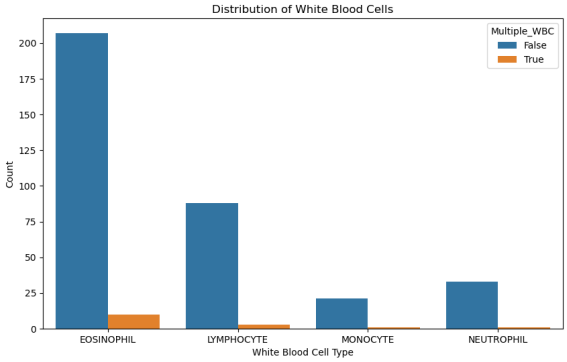

Fig. 8. Distribution of White Blood Cells in the BCCD dataset

B. Bounding Box Object Detection Model Results

After training the YOLOv8 model, we used it to make predictions on the validation and testing sets. Overall, the model achieved a Mean Average Precision (mAP) of 0.93 across all classes. The model performed particularly well on the class of WBC, with a mAP of 0.983. The Precision-Recall Curve is shown below in Figure 9.

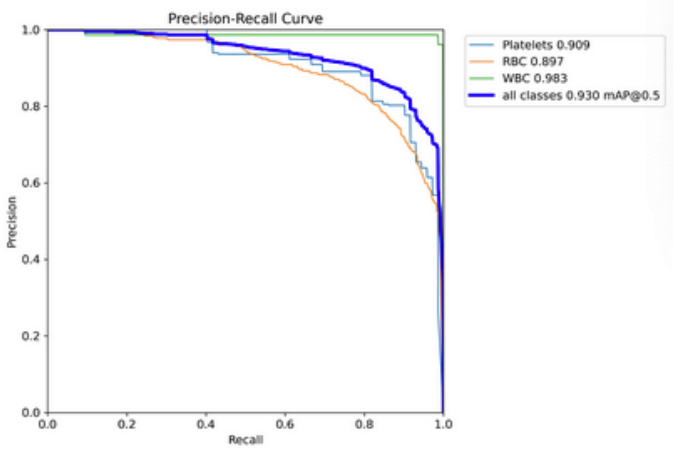

Fig. 9. Mean Average Precision (mAP) across all classes

WBC achieving the highest mAP is unsurprising due to the visual distinctiveness within blood sample images. RBC posed a difficult challenge for the model because oftentimes, RBC would blend together in the background of images. This phenomena is shown in Figure 10, which depicts a visual representation of our model's performance.

Additionally, the confusion matrix shows that the model did not frequently mistake one class for another. Precision was

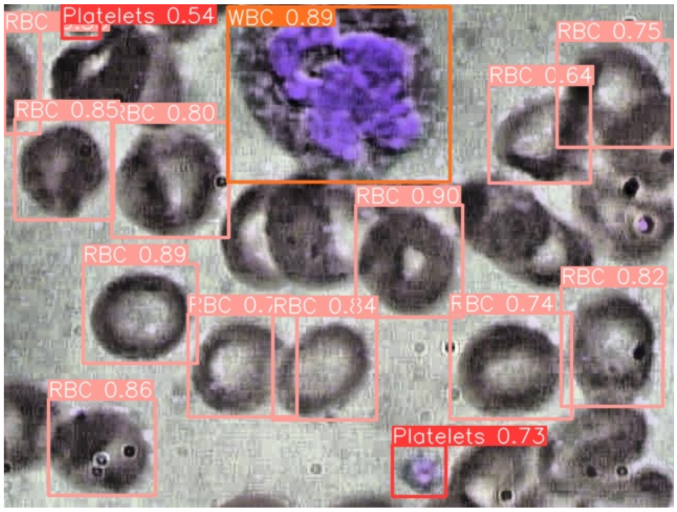

Fig. 10. Model Predictions on Test Set Image

high across all three classes, meaning that there were very few false positives. However, for RBC in particular, there were cases where the model classified something as "background" when it should have detected an object.

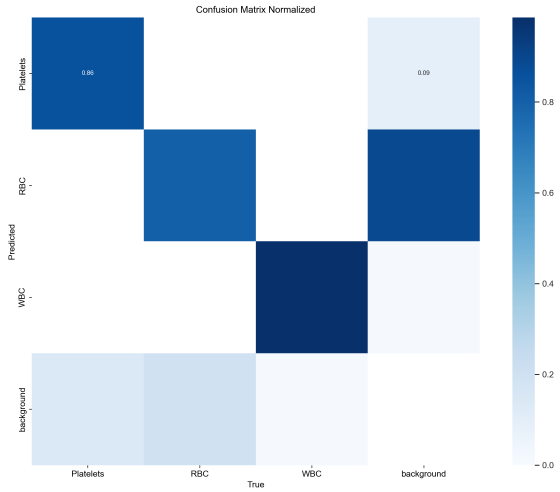

Fig. 11. Normalized Confusion Matrix

Lowering the confidence threshold and non-maximum suppressing thresholds would likely help detect more of the background objects that the model missed. However, lowering these thresholds would also negatively impact precision.

Overall, the model's performance shows the impact that automated hematological analysis could have on Complete Blood Count and Blood Differential Tests. Increasing the amount of training data given to the model would also aid in improving model performance even more.

C. WBC Subtype Classification Model Results

For our secondary goal of WBC Subtype classification, we trained a model based on ResNet-50. This model performed slightly worse than the bounding box object detection model

due to the more difficult task of classifying very visually similar WBC subtypes into four different classes. The model achieved a testing accuaracy of 87.62 % on the test set.

TABLE III
WBC SUBTYPE CLASSIFICATION RESULTS

Class	Precision	Recall	F1 Score
Eosinophil	0.85	0.84	0.84
Lymphocyte	1.00	1.00	1.00
Monocyte	0.99	0.93	0.85
Neutrophil	0.74	0.93	0.82

Table II shows the precision, recall, and F1 Scores for each of the four WBC subtype classes. The model performed very well for Lymphocyte and Monocyte cell types, and less favorably for the Eosinophil and Neutrophil cell types. There are several reasons for this. First, the Lymphocyte WBC cell type is the most visually distinguishable subtype, differing in appearance from the other three types due to its dense color distribution as shown in Figure 2. Secondly, the data was not as strong for Neutrophil cell types; oftentimes the blood sample images for Neutrophil cells were blurry or lower quality. Seeking out more Neutrophil cells with a higher image quality would be beneficial for improving the performance of the WBC subtype classification model.

1) Juxtaposition to Alternative Models: The ResNet-50 model vastly outperformed other modeling types. WBC Subtype Classification is a difficult problem, and baseline models using traditional machine learning algorithms did not perform well. Pre-trained models offered a significant advantage due to their previous training on extremely large image sets such as ImageNet. This previous training meant that these models were already capable of high-level feature extraction such as edge detection.

TABLE IV
WBC SUBTYPE CLASSIFICATION MODEL COMPARISON

Model	Test Set Accuracy
ResNet-50	87.6 %
VGG	41.8 %
KNN	38.0 %
MobileNet	32.4 %
XGBoost	31.0 %
SVM	31.0 %
Random Forest	22.0 %

2) Parameter Settings and Ablation Study: Due to time constraints relating to model training (each ResNet-50 model took over 8 hours to train), it was not feasible to perform a complete ablation study. However, we did perform a partial ablation study by removing different parts of the model and comparing performance.

One way in which the different ResNet models we trained varied was by the size of the input images. We achieved a test set accuracy of 84.1 % when using a modified input image size of (244, 244, 3), compared to an accuracy of 87.6 % when using the original image sizes of (240, 320, 3). Model performance improved when using the original image sizes rather than resizing them.

This suggests that for automated WBC subtype classification, it may be helpful to use a larger image size to allow the model to capture more nuanced and detailed features from the input images. However, using the original image sizes also led to longer training times. Therefore, resizing images may be necessary in the case of computational limitations or extremely large training sets.

Another aspect of the model we tested was the addition of an additional convolutional layer prior to to Global Average Pooling layer in the ResNet model architecture. This did not improve performance, instead causing test set accuracy to decrease to 81.4 %. This may indicate that the additional layer introduced unecessary complexities that led to overfitting rather than improving model performance.

Overall, the highest performing model from our partial ablation study utilized the original input image sizes and did not add an additional convolutional layer prior to the Global Average Pooling layer.

V. CONCLUSION

After building and evaluating the convolutional neural network model for object detection and cell classification, the results show the effectiveness and ability of automated models for hematological analysis. The model is effective in detecting and classifying cells in blood sample images as either WBC, RBC, or platelet, with 93% Mean Average Precision (mAP) and 98.3% mAP on the focus class of WBC. Furthermore, the model is highly accurate in classifying the WBC subtypes within the blood sample images, achieving 87.62% accuracy. These outcomes exhibit the potential of automated hematological analysis through CNN models as an effective technique for blood cell classification.

The results of our model further enforce the claim that automated hematological analysis through CNN models is an effective technique for the classification of blood cells. The model has the ability to be applied in practical settings with high effectiveness and the impact of the use of cell classification models can be significant within healthcare and hematological research.

A. Limitations

While our study marks a strong step forward in blood sample analysis, it has also allowed for a deeper understanding of the challenges and limitations within this field.

One of the key limitations for a model in this problem space is the data quality of the training and test data. The data is one of the most critical parts, as it is the baseline that teaches the model what to look for to classify these cells. Any discrepancies in the data, such as mislabeled cells, can introduce fatal errors to the model, leading to inaccurate classifications, which can massively skew conclusions drawn by either doctors providing medical diagnoses or scientists drawing conclusions in their studies based on the results

provided by the model. Additionally, the size of the training set greatly impacts model performance.

Another limitation that will affect the application of this model is the cost function associated with classifying cells as platelets, RBCs, WBCs, or the more specific WBC types. The model returning false positives or false negatives can have major affects on the diagnoses on patients or the conclusions drawn in the laboratory. A false positive or false negative can result in patients undergoing treatments or stopping treatments prematurely for a variety of medical issues.

B. Improvements

Future improvements to improve the usability, sharpen the performance, and widen the areas of appropriate application are needed. An improvement that must be put in place is a solution to the challenge listed above of false positives or false negatives. One improvement that can address this is adding a penalty when misclassifying as a false positive or false negative. Also, refining a cost function that properly reflects the significance of false positives and false negatives would help to optimize model performance for application towards specific medical testing procedures.

C. Applications

Advancements in automated blood cell classifications would have an overwhelmingly positive affect in the medical world. Precise machine learning models can lead to more accurate classifications of cells, without the time consuming examination by medical professionals. This would result in more time for medical professionals to focus on proper diagnoses and treatments for their patients as well as being able to see and treat more patients.

Overall, the furthering of the integration of machine learning in the medical field would have a massive impact in healthcare, patient outcomes, and the field of hematology.

REFERENCES

[1] 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), 107–118. https://doi.org/10.1002/ppsc.200400888

[2] Babio N, Ibarrola-Jurado N, Bulló M, Martínez-González MÁ, Wärnberg J, Salaverría I, Ortega-Calvo M, Estruch R, Serra-Majem L, Covas MI, Sorli JV, Salas-Salvadó J; PREDIMED Study Investigators. White blood cell counts as risk markers of developing metabolic syndrome and its components in the PREDIMED study. PLoS One. 2013;8(3):e58354. doi: 10.1371/journal.pone.0058354. Epub 2013 Mar 19. PMID: 23526980; PMCID: PMC3602299.

[3] Blumenreich MS. The White Blood Cell and Differential Count. In: Walker HK, Hall WD, Hurst JW, editors. Clinical Methods: The History, Physical, and Laboratory Examinations. 3rd edition. Boston: Butterworths; 1990. Chapter 153. Available from: https://www.ncbi.nlm.nih.gov/books/NBK261/

[4] Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. CoRR, abs/2004.10934. https://arxiv.org/abs/2004.10934

[5] Deshpande, N. M., Gite, S., & Aluvalu, R. (2021). A review of microscopic analysis of blood cells for disease detection with AI perspective. *PeerJ. Computer Science*, 7, e460. https://doi.org/10.7717/peerj-cs.460

[6] Dondelinger, Robert M. (2009). Hematology analyzers. *Biomed Instrumentation & Technology.* 2009 Jul; 43(4), 300-304. doi: 10.2345/0899-8205-43.4.300.

[7] Habibzadeh, M., Jannesari, M., Rezaei, Z., Baharvand, H., & Totonchi, M. (2018). Automatic White Blood Cell Classification Using Pre-Trained Deep Learning models: ResNet and Inception. *Tenth International Conference on Machine Vision (ICMV 2017)*, 13 April 2018. https://doi.org/10.1117/12.2311282

[8] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. CoRR, abs/1512.03385. Retrieved from http://arxiv.org/abs/1512.03385

[9] Jocher, G., Chaurasia, A., & Qiu, J. (2023). Ultralytics YOLOv8 (Version 8.0.0) [Software]. AGPL-3.0. https://github.com/ultralytics/ultralytics

[10] Kubiak, A., Ziółkowska, E., & Korycka-Wołowiec, A. (2022). The diagnostic pitfalls and challenges associated with basic hematological tests. *Acta Haematologica Polonica*, 53(2), 104–111.

[11] Kutlu, H., Avci, E., & Özyurt, F. White blood cells detection and classification based on regional convolutional neural networks, *Medical Hypotheses*, Volume 135, 2020, 109472, ISSN 0306-9877, https://doi.org/10.1016/j.mehy.2019.109472.

[12] LabMedica International (2022). Machine Learning Radically Reduces Workload of Cell Counting for Disease Diagnosis. Retrieved on October 10, 2023 from https://www.labmedica.com/hematology/articles/294793039/machine-learning-radically-reduces-workload-of-cell-counting-for-disease-diagnosis.html

[13] Leukemia and Lymphoma Society (2023). Blood Cancer Statistics. Retrieved on October 6, 2023 from https://www.lls.org/facts-and-statistics/facts-and-statistics-overview

[14] Lou, J., Zhou, M., Li, Q., Yuan, C., & Liu, H. (2016). An automatic red blood cell counting method based on spectral images. *2016 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Datong, China, 2016, 1391-1396*. https://doi.org/10.1109/CISP-BMEI.2016.7852934

[15] Medline Plus (2023). Blood differential testing. Bethesda (MD): National Library of Medicine (US). Retrieved on October 7, 2023 from https://medlineplus.gov/heartattack.html

[16] Mooney, P. (2018). Blood Cell Images. Version 6. Retrieved on September 20, https://www.kaggle.com/datasets/paultimothymooney/blood-cells/data

[17] Paul AM, Mhatre SD, Cekanaviciute E, Schreurs AS, Tahimic CGT, Globus RK, Anand S, Crucian BE, Bhattacharya S. Neutrophil-to-Lymphocyte Ratio: A Biomarker to Monitor the Immune Status of Astronauts. Front Immunol. 2020 Nov 2;11:564950. doi: 10.3389/fimmu.2020.564950. PMID: 33224136; PMCID: PMC7667275.

[18] Pizer, S.M., Amburn, E.P., Austin, J. D., Cromartie, R., Geselowitz, A., Greer, T., Romeney, B., Zimmerman, J.B., & Zuiderveld, K. (1987). Adaptive Histogram Equalization and Its Variations. *Computer Vision, Graphics, and Image Processing* 39, 355-368. https://www.cs.unc.edu/Research/Image/MIDAG/pubs/papers/Adaptive%20Histogram%20Equalization%20and%20Its%20Variations.pdf

[19] Seo, In-Ho, and Yong-Jae Lee. "Usefulness of Complete Blood Count (CBC) to Assess Cardiovascular and Metabolic Diseases in Clinical Settings: A Comprehensive Literature Review." Biomedicines vol. 10,11 2697. 25 Oct. 2022, doi:10.3390/biomedicines10112697

[20] Sinha, N., & Ramakrishnan V., "Automation of differential blood count," TENCON 2003. Conference on Convergent Technologies for Asia-Pacific Region, Bangalore, India, 2003, pp. 547-551 Vol.2, doi: 10.1109/TENCON.2003.1273221.

[21] Shahriar, M.T., & Li, H. "A Study of Image Pre-processing for Faster Object Recognition." arXiv preprint arXiv:2011.06928 (2020). https://arxiv.org/abs/2011.06928

[22] Shenggan. (2017). BCCD Dataset [Data set]. GitHub. https://github.com/Shenggan/BCCD_Dataset

[23] Song M, Graubard BI, Rabkin CS, Engels EA. Neutrophil-to-lymphocyte ratio and mortality in the United States general population. Sci Rep. 2021 Jan 11;11(1):464. doi: 10.1038/s41598-020-79431-7.

[24] "Whole Blood Staining Protocol for Flow Cytometry Analysis." Thermo Fisher Scientific - US, www.thermofisher.com/us/en/home/life-science/cell-analysis/cell-analysis-learning-center/immunology-at-work/immunology-protocols/whole-blood-staining-protocol-flow-cytometry.html. Accessed 8 Dec. 2023.

APPENDIX

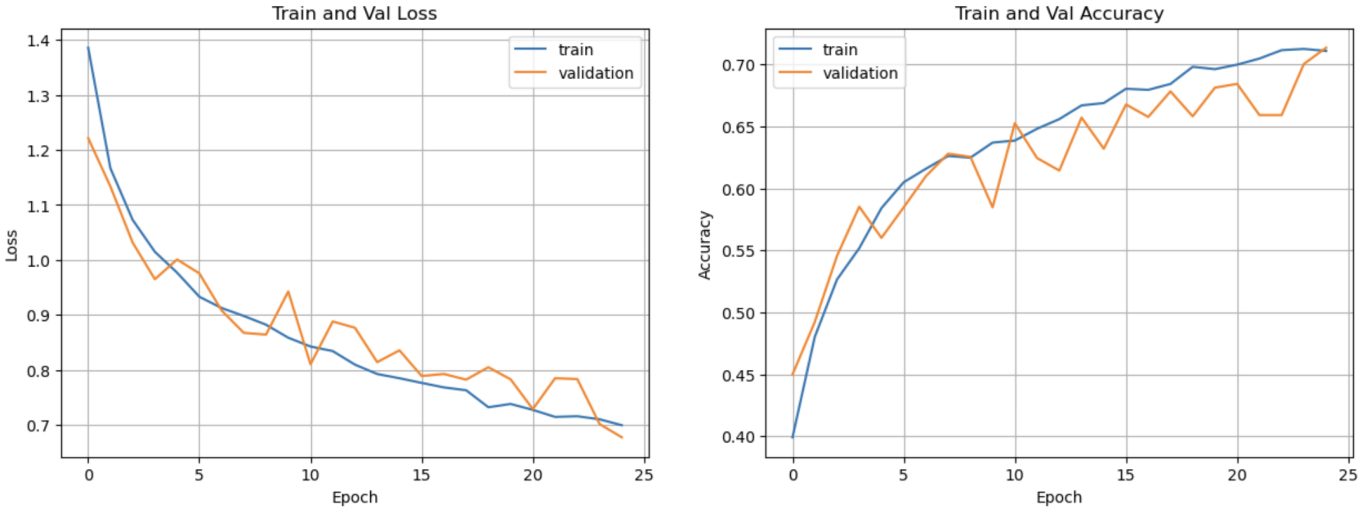

Fig. 12. MobileNet Loss and Accuracy Changes over 25 Epochs

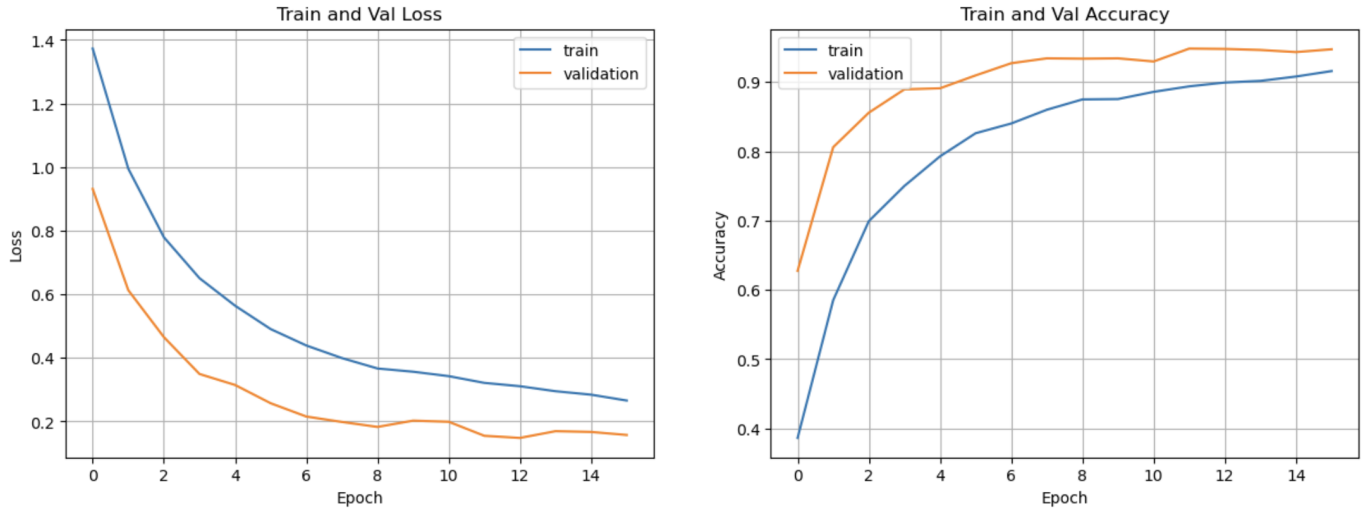

Fig. 13. VGG Loss and Accuracy Changes over 25 Epochs