# Final Report

## Abstract

Leukemia is responsible for the deaths of 157 people every single day. Acute Myeloid can be in specific when white Blood Cells are trapped in the bone marrow and do not make their way into the blood. The condition can be visualized in blood smears where a ratio of WBC to RBC can be observed and calculated. An incorrect ratio can help diagnose the condition. Using a resNet 50 architecture as a pretrained model that is trained of thousands of images, a detector can be input for specific object identification. Using the FasterRCNN that is trained on a BCCD dataset the ML model can detect and annotate an input image with bounding boxes and their classes. Taking the detected bounding boxes and their classes and getting a ratio of WBC and RBC. Given this ratio an if statement is used to output the likelihood of AML being present.

## Introduction

Our project is based upon the disease acute myeloid leukemia or AML, which is represented by a lack of white blood cells within the patients blood stream. This is because the disease traps white blood cells within bone marrow limiting the amount that is in the blood. Our project is a machine learning program that is able to analyze a given blood smear, detect individual white and red blood cells within the smear, give a count of each, and then compare the ratio of red to white blood cells that it has detected with what is considered a healthy ratio. After comparing the determined ratio with the healthy ratio the program is then able to determine the patient's likelihood that they have AML. This will be useful to doctors because around 28% of cancer cases involve misdiagnosis which can be solved by using a more accurate program to perform the diagnosis. Hopefully this will also increase the ability to detect the leukemia earlier, giving the patient a better chance at beating the disease.

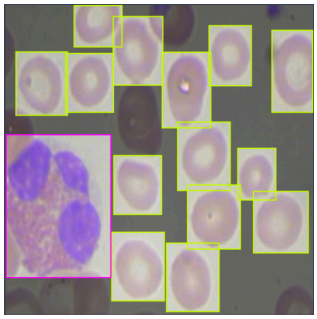
After researching multiple different neural networks such as R-CNN, Faster and Fast R-CNNs, YOLO and others, we determined that Faster R-CNN, even though it might be more of a hassle to train than for example a YOLO program, would be best suited for our cause due to it’s minimal computing resources requirement as well as it’s ability to return accurate results. After determining which neural network to use, we had to figure out the best way to use object detection models to identify the individual cells within the blood smear data we provide. After comparing and contrasting 3 different models being ResNet, AlexNet, and VGG16, we determined that ResNet, even though it requires more computing time and is more complex, would be best for our needs. ResNet is one of many standard object detection models that is typically used to classify entire images at once. For our needs, we needed it to detect individual objects within an image. Using the Faster R-CNN network we were able to modify ResNet to look for individual objects within an image using bounding boxes, labels, and classes which is exactly what we needed.

The last anticipated bump in our plans was the fact that to train a machine learning algorithm, it requires annotated data sets to learn from before applying that knowledge to new test images. When doing machine learning in MatLab, MatLab prefers that you create the pre-annotated data sets within MatLab, however this would take a extremely long time to perform for each image needed to properly train the model. Luckily for us we were able to find a pre-annotated data set of 400+ blood smears online, and were able to download and incorporate them into our program. After some slight modifications including changing the file type to uint8 files as well as resizing the images from their previous 416 x 416 pixel dimensions to 224 x 224 pixels which is the size preferred by ResNet, we were able to train our modified version of a pre-trained Faster R-CNN model to detect each blood cell and determine their class. Typically we would have liked to use a more computationally intensive model, with greater epoch iterations, and a different initial learning rate, however simply due to the resources we had available to us as well as the time constraint set on the project, we were limited to a set learning rate as well as only a single epoch.

## Results and Discussion

Unfortunately, because of the following detect function where where unable to get an output for bounding boxes however the output would look similar to the second image below:

```
% Detect objects in the resized image
[bbox, labels, scores] = detect(detectors, tester, 'SelectStrongest', true, 'MinibatchSize', 1);
```



1) Detect function(Select Strongest was used to have at least any output if the detect function had a sigmoid out put less than .5) 2) Bounding boxes on red blood cells and White Blood Cells

The problem manifests in the detect function, which, despite correct input image conversion and accurate detector operation, returns an empty array with the correct bounding box size but lacks information. The potential causes include concerns with the CUDA-enabled GPU, ensuring compatibility and proper initialization; scrutinizing the input image processing for accuracy in format, size, and color channels; and reviewing detector configurations, class labels, and associated parameters. Out of these problems the GPU not running parallel processing is the most likely problem and can be addressed in the future.

For showing the image a basic for loop was used to run through the data and output a ratio, the code of which is displayed below:

```
% Visualize the results
detectedImg = insertObjectAnnotation(testImage, 'Rectangle', bbox, labels);
imshow(detectedImg);
```

```
%% Step 11: Results
% Get Ratio of WBC and RBC

% Extract the first two labels
firstTwoLabels = labels(2:3);

% Count the number of times each label appears
uniqueLabels = unique(firstTwoLabels);
counts = zeros(length(uniqueLabels), 1);

for i = 1:length(firstTwoLabels)
    for j = 1:length(uniqueLabels)
        if strcmp(firstTwoLabels(i), uniqueLabels(j))
            counts(j) = counts(j) + 1;
        end
    end
end

% Calculate the ratio between the first two labels
ratio = counts(1) / counts(3);

% Print the results
if ratio < 6 %Values minimized for available data
    disp("Chance of AML: LOW");
elseif ratio < 15
    disp("Chance of AML: MEDIUM");
else
    disp("Chance of AML: HIGH");
end
```

## Conclusion

After finishing our project, we came to the conclusion that there were a few things that we could've done to make the overall difficulty of the project easier. Just to start off, with the lack of time and the problem with importing pre-annotated data, MatLab just is not the ideal program for performing this sort of task. It would have been much easier and simpler to just use Python with its various open source libraries. Besides the language used, ideally if we decided to stick with MatLab, annotating the data within MatLab, given enough time, would have made the task of providing annotated data to train the model on significantly easier. We would have been able to skip the whole steps dealing with converting the JPG files into uin8 files for MatLab to read which ended up being many lines of code. The last thing we should have done before committing to this project was checking to make sure that our GPU would be able to process the detect function, which bottlenecked our success by giving us enough problems to prevent us being able to get an actual output, despite the training of the code and everything else looking like it was prepared to work. Some things we think we could improve on in the future with a second iteration of the project would not only be annotating the training data in MatLab but also utilizing better test data because the only data we had available to work with were very small windows of a blood smear that wouldn't properly depict an accurate ratio of red to white blood cells. If we were able to find a group of larger blood smears, we would be able to get much more accurate results because the training data would represent the ratio that we are looking for better. Some changes such as these listed would have been possible for this iteration as well, however given the time frame that we had to produce the final product as well as the fact that

we wanted to make the program work within MatLab, they just weren't reasonable expectations. In the future however with these modifications our program could be very successful and help doctors diagnose AML more accurately than ever before.

## References


**AML Mortality Information References**

(National Institutes of Health, 2020)

(Brown and Barron, 2019)

(Annals of Hematology, 2007)

(National Cancer Institute, 2023)

(Journal of Hematology and Oncology, Ming Yi, Anping Li, Linghui Zhou, Qian Chu, Yongping Song, Kongming Wu, 2017)

(NYU Langone Health, 2023)

(National Center for Technology Information, 2022)

**AML Code References**

(*List Folder Contents - MATLAB Dir*, n.d.)

(*Build Full File Name From Parts - MATLAB Fullfile*, n.d.)

(*Determine if Input Is Cell Array - MATLAB Iscell*, n.d.)

(*Convert Strings to Double Precision Values - MATLAB Str2double*, n.d.)

(*Unique Values in Array - MATLAB Unique*, n.d.)

(*Compare Strings - MATLAB Strcmp*, n.d.)

(*Table Array With Named Variables That Can Contain Different Types - MATLAB*, n.d.)

(*Datastore for Bounding Box Label Data - MATLAB*, n.d.)

(*Datastore for Image Data - MATLAB*, n.d.)

(*ResNet-50 Convolutional Neural Network - MATLAB Resnet50*, n.d.-b)

(*Object Detection Using Faster R-CNN Deep Learning - MATLAB & Simulink*, n.d.)

(*Check Existence of Variable, Script, Function, Folder, or Class - MATLAB Exist*, n.d.)

(*Make New Folder - MATLAB Mkdir*, n.d.)

(*Options for Training Deep Learning Neural Network - MATLAB trainingOptions*, n.d.)

(*Train a Faster R-CNN Deep Learning Object Detector - MATLAB trainFasterRCNNObjectDetector*, n.d.)

(*Resize Image - MATLAB Imresize*, n.d.)

(*Annotate Truecolor or Grayscale Image or Video - MATLAB insertObjectAnnotation*, n.d.)

(*Detect Objects Using ACF Object Detector - MATLAB Detect*, n.d.)

(*Find Matching Features - MATLAB matchFeatures*, n.d.-b)

(*Write a Map Function - MATLAB & Simulink*, n.d.)