Final Project Report

White Blood Cell Classification
Image Classification Neural Network – Transfer Learning Based

Edward Sung
2/13/22

Problem Statement

Numerous health related illness and diseases can be diagnosed through a quick and simple blood cell counting called a complete blood count (CBC). White blood cells specifically, play a significant role in body immunity with different types of white blood cells functioning to combat specific types of infections. Cell staining techniques can stain the white blood cell's nucleus and make it easier to identify and differentiate.

- Eosinophils: bi-lobed nucleus that are sausage-shaped; fight against parasites.
- Neutrophils: multi-lobed nucleus; fight against bacterial infections.
- Lymphocytes: single large and round nucleus; fight against viral infections.
- Monocytes: single large and kidney shaped nucleus; clean up dead or damaged cells.

Using image classification techniques and algorithms, how can images of stained and isolated white blood cells be correctly identified? What kinds of problems may arise when trying to classify between 4 different types of white blood cells? How can the image classifier be further used or improved?

Data Set

The data comes from Kaggle and is provided by Paul Mooney. He augments 410 original images from Shenggan github to create a training set and a test set of images that have folders containing each of the 4 types of white blood cells: eosinophil, lymphocyte, monocyte, and neutrophil.

The folders contain images that are in jpeg and have a size of 240 x 320.

Training Set Folder contains:

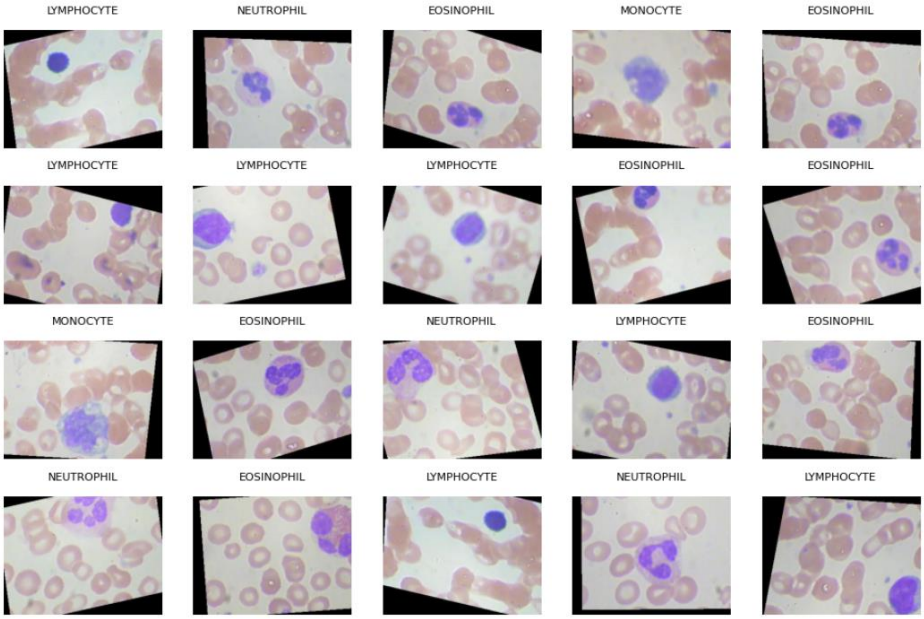
- Eosinophil Folder: 2497 images
- Lymphocyte Folder: 2483 images
- Monocyte Folder: 2478 images
- Neutrophil Folder: 2499 images

Test Set Folder contains:

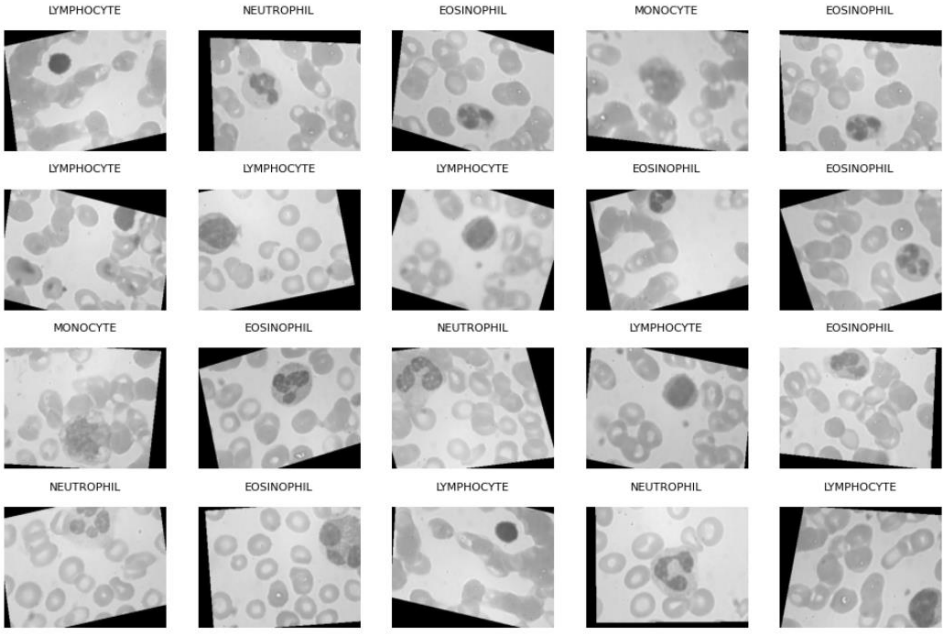
- Eosinophil Folder: 623 images
- Lymphocyte Folder: 620 images
- Monocyte Folder: 620 images
- Neutrophil Folder: 624 images

Data Wrangling and Exploratory Data Analysis

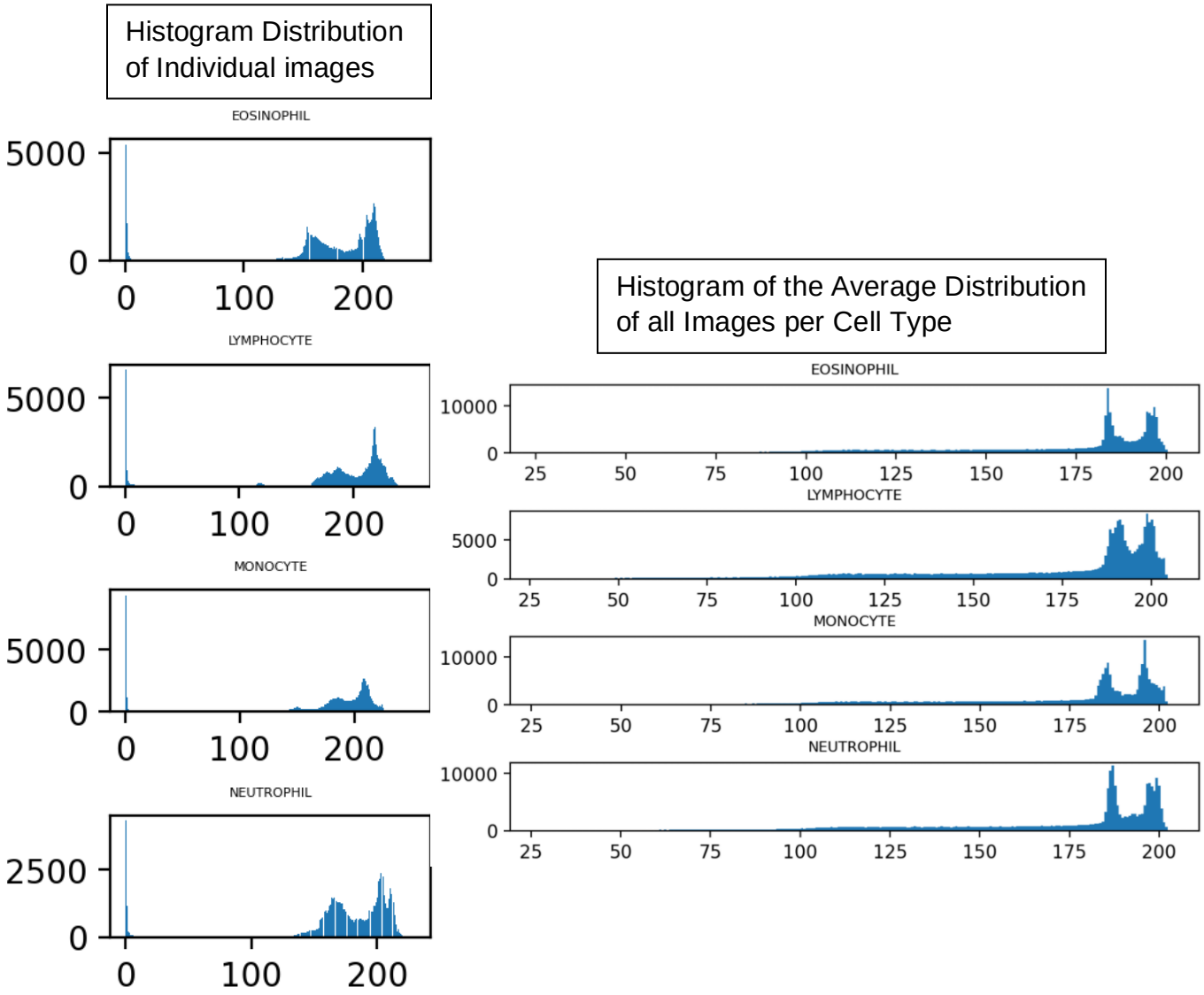
Since the images have been curated and augmented by the author of the Kaggle dataset, there were no transformation needed to be done to the images. They were all set to the same image size of 240 x 320 and the image quality looked clean and clear.



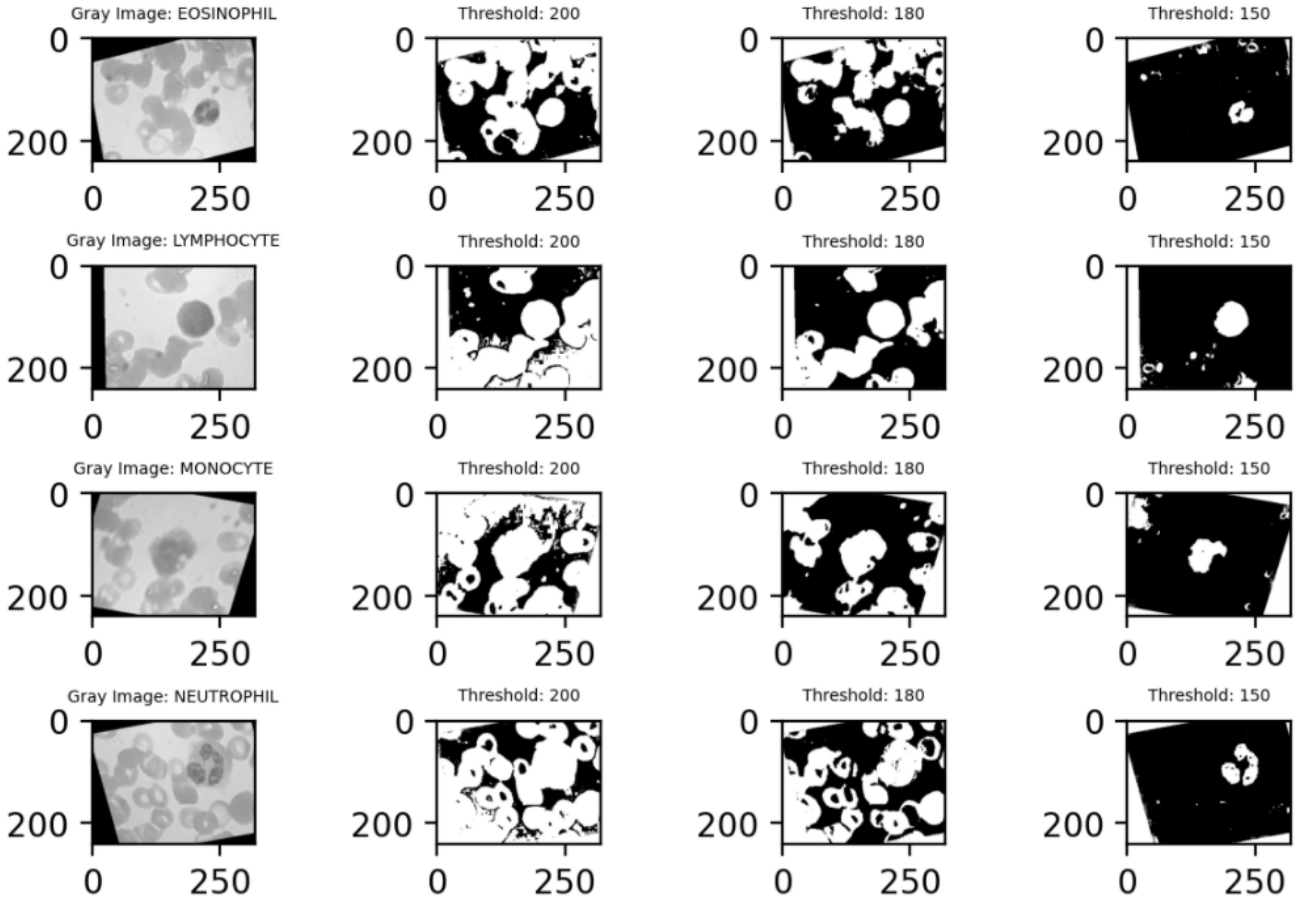
Given that there is only a distinct purple staining of the white blood cells versus the non-target cells, I converted the images to gray scale. This simplifies the image for exploration instead of having 3 different color channels. The while blood cells were still easily identifiable in gray scale.



Histograms of the pixel intensity in gray scale showed that each image had a large bin size at 0 shown in the left graph. This pixel intensity of 0 represents the black parts of the image due to the rotational augmentation of the images. But upon taking the average pixel intensity of all the images per cell type shown in the right graph, this pixel intensity at around 0 was eliminated and is insignificant compared to the rest of the pixel intensities. This also means that it is evenly distributed among all the cell types and does not need to be taken into consideration for the model.



There were also no significant differences between the types of white blood cells in terms of histogram distribution of the pixel intensities. They all showed the same bi-modal peaks between 175 to 200. Using thresholding, these two peaks were identified to be the empty spaces between the cells at 200 and the non-target/stained cells at 180. Reducing the thresholding further showed that the target white blood cells were identified to be at around 150. When set to this limit, the distinct shape of each type of white blood cell were easily identifiable.

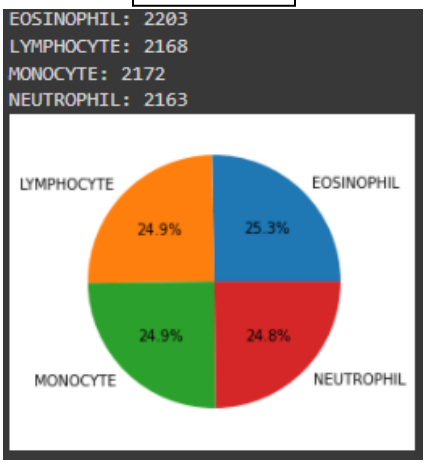


- Eosinophils: bi-lobed nucleus that are sausage-shaped.
- Lymphocytes: single large and round nucleus.
- Monocytes: single large and kidney shaped nucleus.
- Neutrophils: multi-lobed nucleus.

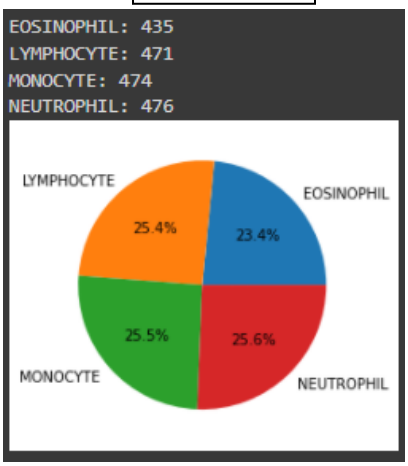
Preprocessing and Model Training

Since the images were augmented from a base of 410 images and details about the augmentation were not clearly defined in the description or summary, I decided to create a combined data set from the training and test sets. This is performed to eliminate any bias or potential augmentation specificity done to either the training set or test set separately, and to ensure that the test set is reflective of the types of images in the training set. From this combined data set, I derived my own training, validation, and test sets with a split of 70%, 15% and 15% respectively.

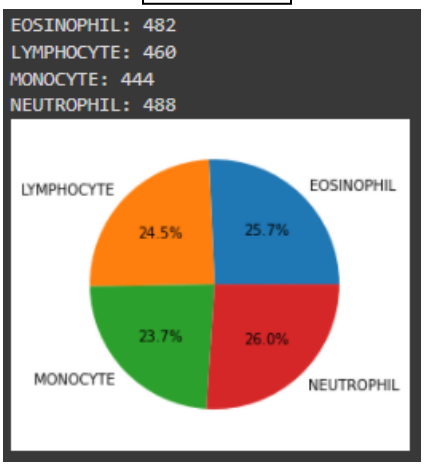
Training



Validation



Testing



I used transfer learning to levearge well-researched and well-built neural networks that have been trained on imagenet to form the base of my own image classification neural network. Specifically, I chosed Xception from the list of available application models provided by Keras for its best accuracy and speed for its size.

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	0.790	0.945	22,910,480	126	109.42	8.06
VGG16	528	0.713	0.901	138,357,544	23	69.50	4.16
VGG19	549	0.713	0.900	143,667,240	26	84.75	4.38
ResNet50	98	0.749	0.921	25,636,712	-	58.20	4.55
ResNet101	171	0.764	0.928	44,707,176	-	89.59	5.19
ResNet152	232	0.766	0.931	60,419,944	-	127.43	6.54
ResNet50V2	98	0.760	0.930	25,613,800	-	45.63	4.42
ResNet101V2	171	0.772	0.938	44,675,560	-	72.73	5.43
ResNet152V2	232	0.780	0.942	60,380,648	-	107.50	6.64
InceptionV3	92	0.779	0.937	23,851,784	159	42.25	6.86
InceptionResNetV2	215	0.803	0.953	55,873,736	572	130.19	10.02
MobileNet	16	0.704	0.895	4,253,864	88	22.60	3.44

I added a preprocessing layer that Keras provides as a method, to take the image array values and convert them to values that are between -1 and 1 for Xception to accept. I applied transfer learning technique and developed my own top layer classifier that takes in the number of nodes from the Xception base model and exponentially decreases to the number of nodes needed for classification. I included dropout layers in between the dense layers to reduce overfitting.

I performed 2 rounds of training for my model with early stopping that tracked the sparse categorical crossentropy loss metric of the validation set to ensure the model does not unnecessarily overfit to the training data without providing significant boost to its classifying abilities.

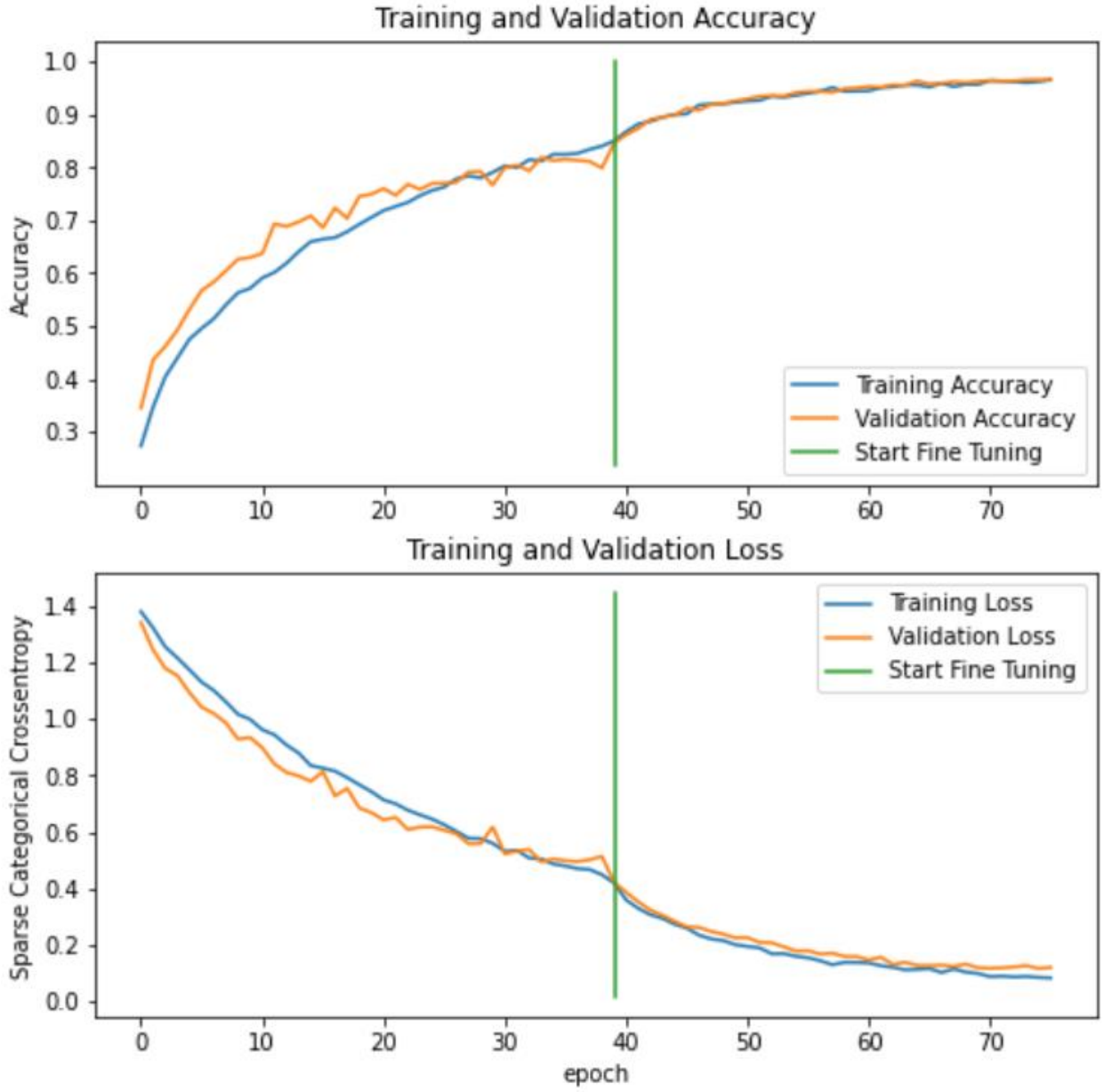
For the first training round, I froze the base model Xception and trained my top layer on the training set.

After the appropriate weights were trained for the top layer classifier, I performed a second round of training that unfroze the base model and used a very low learning rate value. This allowed the Xception model to adjust its weights to be specialized towards my training set cell images rather than the 1.4 million images and 1000 classes it has been trained on through imagenet.

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 240, 320, 3)]	0
tf.math.truediv (TFOpLambda)	(None, 240, 320, 3)	0
tf.math.subtract (TFOpLambda)	(None, 240, 320, 3)	0
xception (Functional)	(None, 8, 10, 2048)	20861480
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
dropout (Dropout)	(None, 2048)	0
dense (Dense)	(None, 1024)	2098176
dropout_1 (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 512)	524800
dropout_2 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 128)	65664
dropout_3 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 64)	8256
dropout_4 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 8)	520
dropout_5 (Dropout)	(None, 8)	0
dense_5 (Dense)	(None, 4)	36

Out of the box model with an untrained top layer yield a 26% accuracy on the test set. The first-round of training of just the top layer classifier yield an 82% accuracy on the test set. With the second round of fine tuning and training of the entire base model and top layer classifier, the final model was able to classify with 96% accuracy, a 15% increase in classifying power.

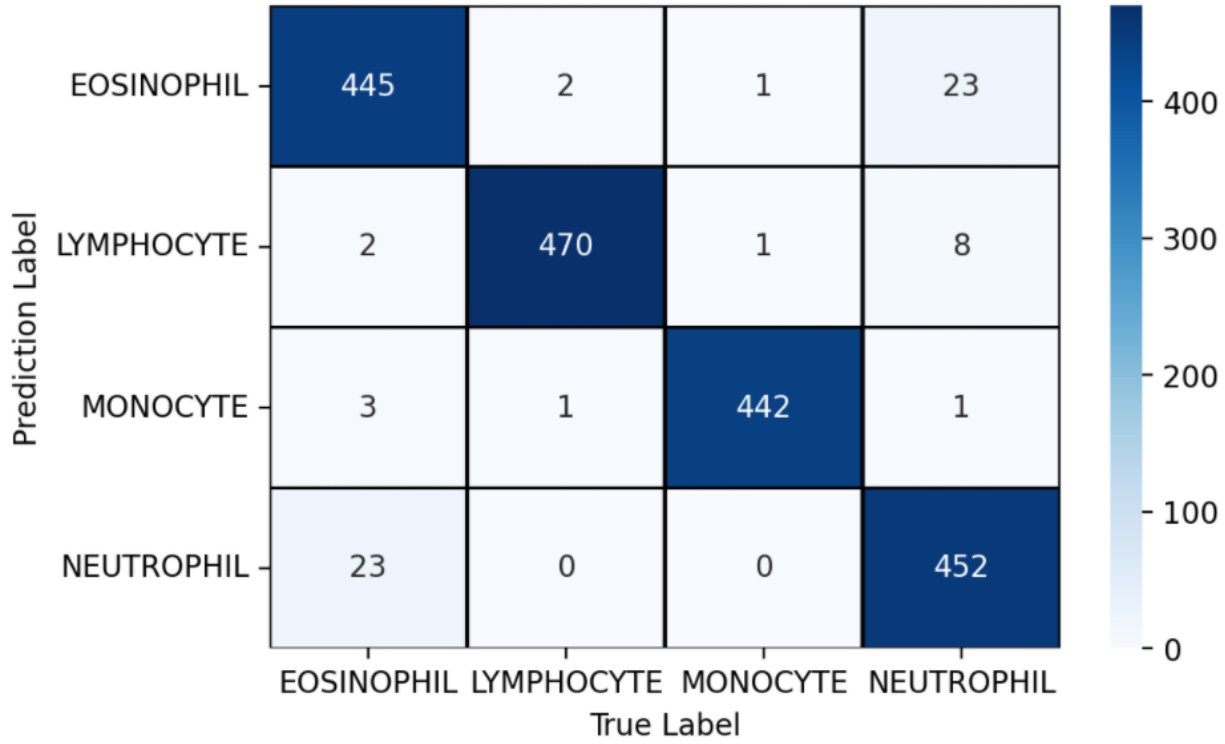
Model	Validation Accuracy	Test Set Accuracy
Untrained Base Model	-	26%
Trained Base Model	82%	82%
Fine Tuned Entire Model	97%	96%



Modeling and Evaluation

The final tuned model classified the test set with a 96% accuracy, incorrectly classifying 65 images out of the 1874 images in the test set. Evaluating the confusion matrix, the model looks to have difficulty classifying between Eosinophil and Neutrophil, mixing up between the two.

	precision	recall	f1-score	support
EOSINOPHIL	0.94	0.94	0.94	471
LYMPHOCYTE	0.99	0.98	0.99	481
MONOCYTE	1.00	0.99	0.99	447
NEUTROPHIL	0.93	0.95	0.94	475
accuracy			0.97	1874
macro avg	0.97	0.97	0.97	1874
weighted avg	0.97	0.97	0.97	1874



Reviewing the images that the model classified wrong, a common trend in most of the pictures were scattered residue of the stain that formed shapes that the model may have accidentally included in its evaluation. This seems the most logical explanation for the cases in differentiating between eosinophil and neutrophil, where the difference is the number of stained nucleus. The other common trend for misclassified images were that these images were actually not as clear and cleaned compared to most of the images. These images had parts of the target cell cut off or the shape and stain of the target cell was not well defined enough. This can be seen in the case between a lymphocyte and a monocyte, where the defining feature of a monocyte is its kidney shape. But if the cell image is not distinct enough for the model to recognize the shape, it will classifying it as a lymphocyte.

Follow-Up and Further Projects

After evaluating the model's performance and seeing the issues that caused problems for the model, a follow-up project could be to create either a cleaner training set or provide additional layers to the classifier with hopes that it will give it more features to use in evaluating the cells. Neural network also requires an immense size of training data to be effective and reliable. I can leverage either the use of a data augmentation generator to create more images or collect more images for the model to train on. The latter is preferred, since it will allow more variety of unique cells rather than augmentation of current images. Also isolating the target cells using image processing and extraction could benefit the classifier, by removing unnecessary noise and allowing the model to focus solely on the target cells.

A different project could be improving the model to be able to count the number of types of white blood cells within a set image or train the model to include a more diverse range of cells rather than just white blood cells. This will extend the model capabilities and be used to diagnosis more than just blood cells, but potentially cancer cells or use in drug discovery.

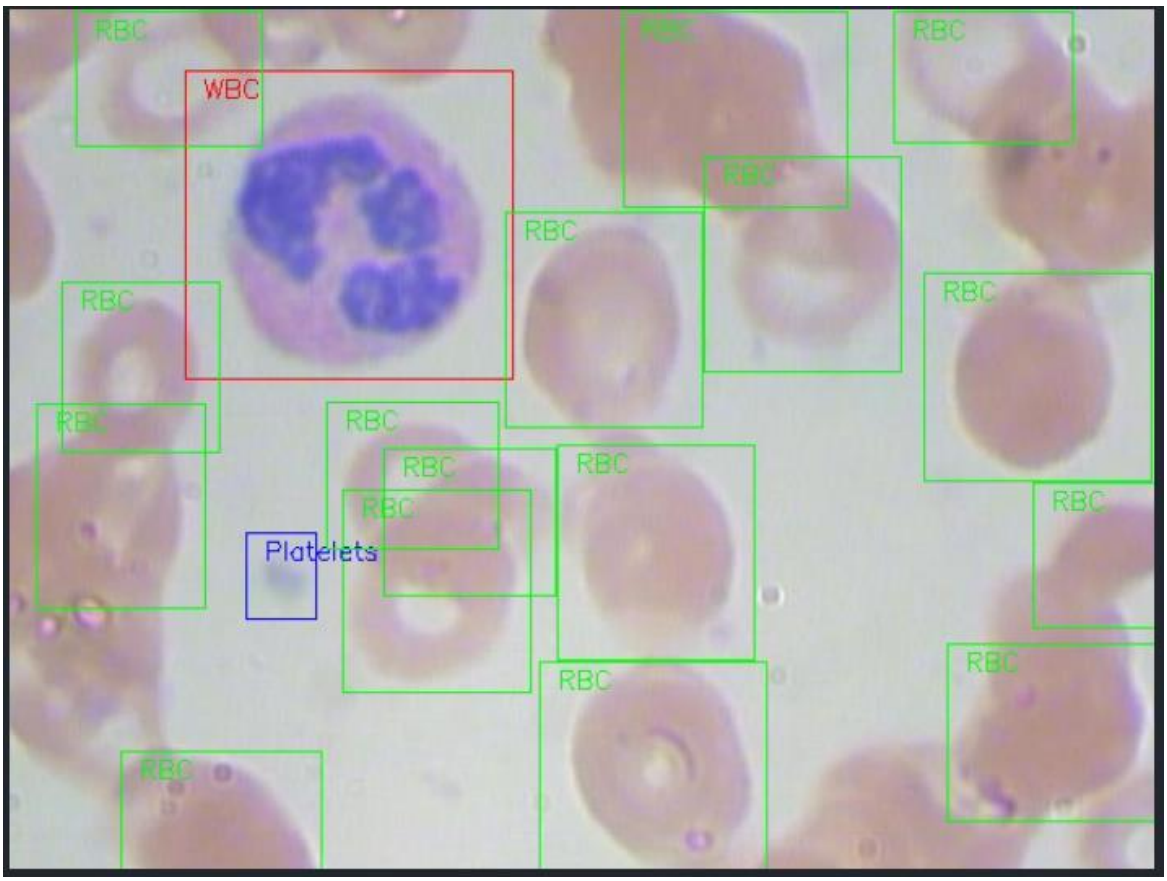


Image from Shenggan Github BCCD Project with label boxes around cells.