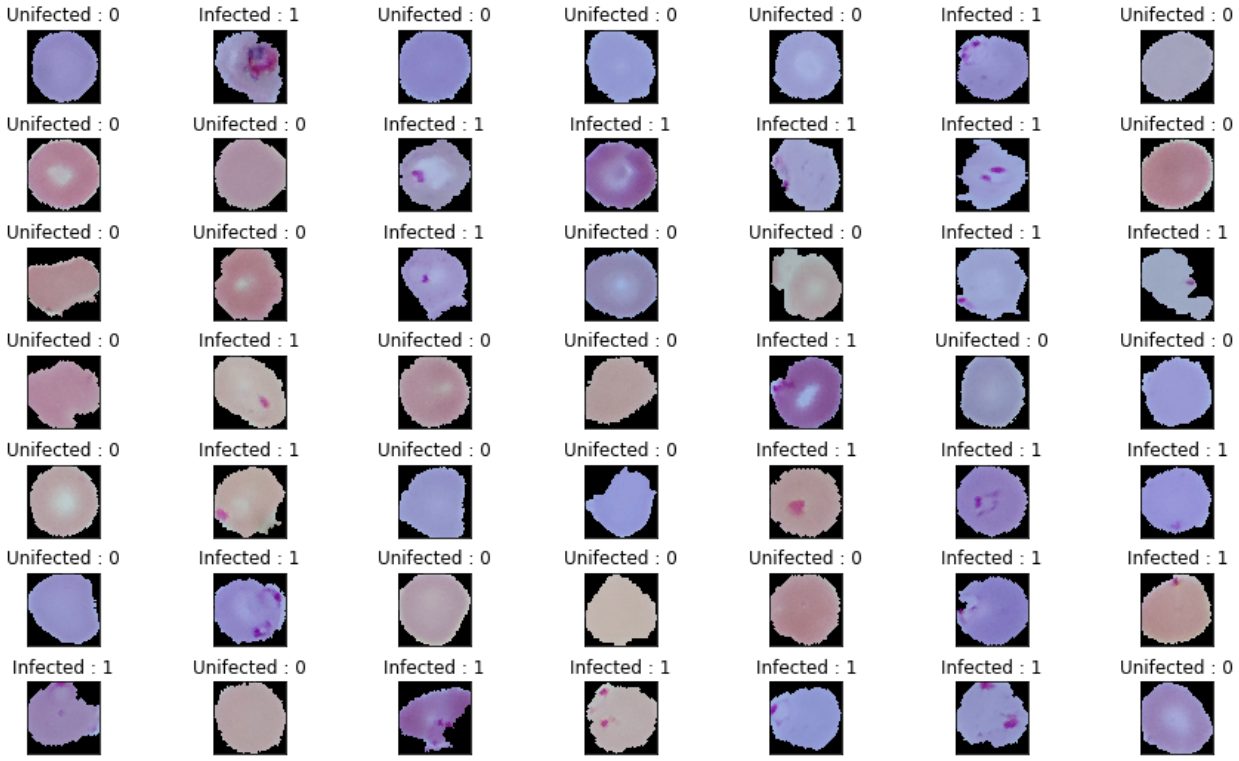
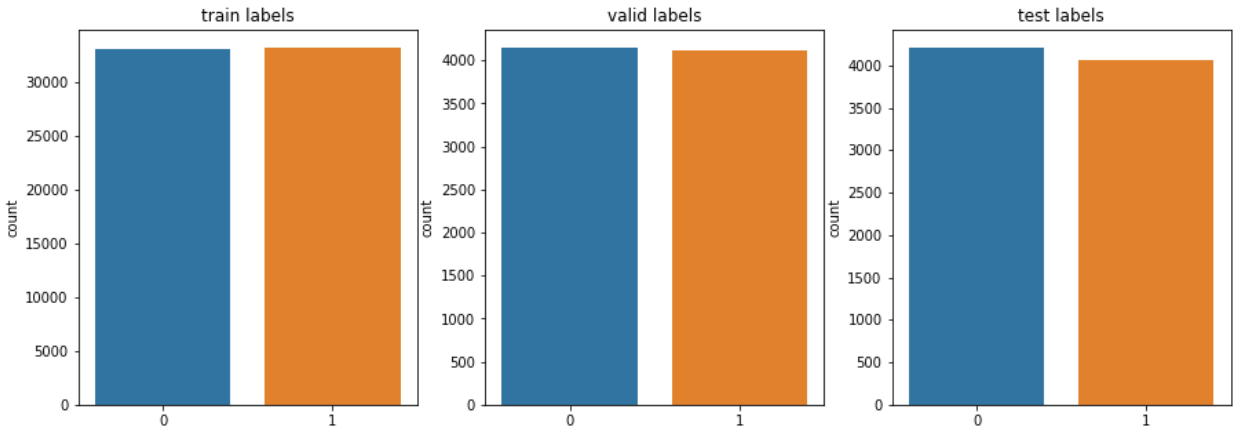
Project Report

Ruilong Zhuang

As mentioned in the proposal, the dataset has a total of 27,558 cell images of size 142 × 163 pixels. Since there is no specific orientation for images, we rotate the images to expand the dataset so we can have more instances. Then, we sample a few pictures of both parasitized and uninfected cells to have a visualization of these two categories. The pictures are shown below.



Then, we use functions train_test_split from the tensorflow library to split the dataset into training set, validation set and test set. The ratio of (Train: Validation: Test) = (8:1:1). The bar graph below shows that the distribution of instances of each categories in different set is uniform after random shuffle, so we do not need to sample data.



After inspecting the images, we consider that the image is too big to train a neural network, so we shrink the image to 50x50 pixels without losing important information. The images are captured in RGB mode, so there are 3 channels that we need to take into consideration. We use OpenCV library to shrink and read the images into an array of shape (50, 50, 3). The input set has 82764 instances, so the final input has a shape of (82674, 50, 50, 3) and label is an array of shape (82674,). The format of the data needs to be reset and reconfirmed, so during shuffling, we cast the value of images to tf.float32 and cast the labels to tf.int32. In order to have a better performance, we normalize the value of input of each channel of pixel by dividing 255 from it. The labels are casted into 0s and 1s corresponding to uninfected and parasitized. Then, the input data is ready to train.

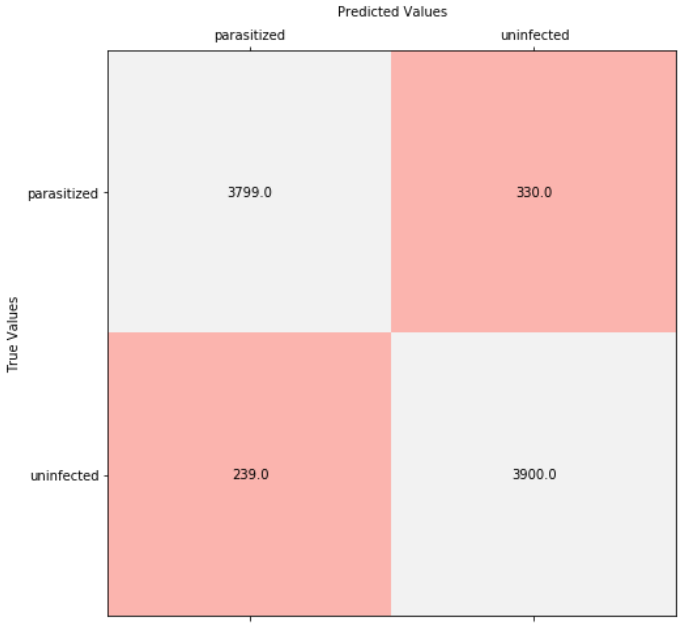
I implement three neural network models in total. The first one mimics the MNIST model from the book which consists of 2 convolutional layers, 1 pooling layers and 2 dense layers. Because the first one does not utilize the validation set and it may overfit the training set, the second contains regularization by early stopping method. The second model has the same structure of neural layers with early stopping methods implemented. The third model uses Keras package to construct a neural network model. The warning message tells us to use Keras instead and I am curious about how to use Keras with an early stopping method. The

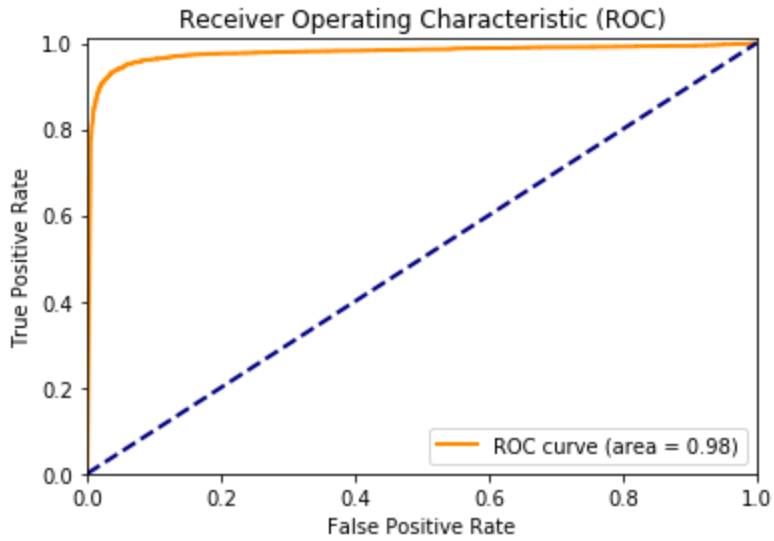
output for the third model has shape of (,1) compared to the former two have shape of (,2) because it is a binary classifier. The third model uses sigmoid activation, so the output p is the probability of an uninfected cell and 1-p is the probability of a parasitized cell. All the models are using Adam optimizer to minimize the cross entropy loss. The accuracy is calculated by counting the hits over the total number of predictions.

The first model is trained twenty epoches and uses a batch size of 50. This model consists of 2 convolutional layers, 1 pooling layer and 2 dense layers. The last five batches accuracy and test set accuracy is presented below.

```
15 Last batch accuracy: 1.0 Test accuracy: 0.9473875
16 Last batch accuracy: 1.0 Test accuracy: 0.9489598
17 Last batch accuracy: 1.0 Test accuracy: 0.95065314
18 Last batch accuracy: 1.0 Test accuracy: 0.9481132
19 Last batch accuracy: 1.0 Test accuracy: 0.94400096
```

The output is a softmax output that has two columns representing the probabilities of two categories. Then, we use the argmax function from numpy library to get the predictions.
We can see that the final model has 94.4% accuracy on the test set, which can be considered as a good model. Then, looking into the confusion matrix, we can calculate precision rate and recall rate. (Precision = 0.9408122833085686, Recall = 0.9200775006054734).



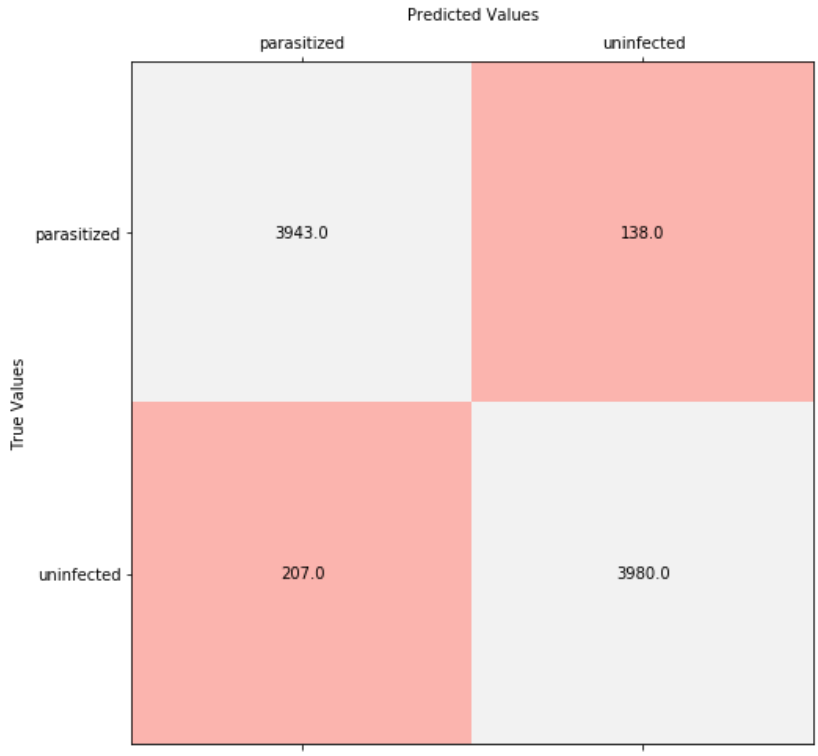


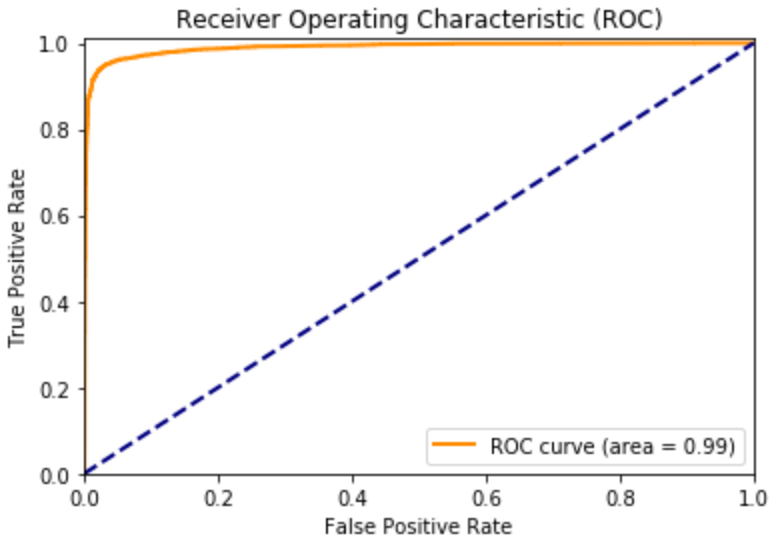
There is a potential problem of overfitting because this model does not have any regularization and we can see that the test accuracy is decreasing.

The second model has the same structure of layers as the first model, and the regularization by early stopping method is added into the model. The training will stop if the evaluations on the validation set do not improve in the last 10 epochs. Therefore, we can see that the training for the second model stops at 13 epochs while the first model goes through the full 20 epochs.

```
Epoch 9, last batch accuracy: 100.0000%, valid. accuracy: 95.6091%, valid. best loss: 0.133095
Epoch 10, last batch accuracy: 96.0000%, valid. accuracy: 95.1010%, valid. best loss: 0.133095
Epoch 11, last batch accuracy: 94.0000%, valid. accuracy: 95.2704%, valid. best loss: 0.133095
Epoch 12, last batch accuracy: 98.0000%, valid. accuracy: 95.1857%, valid. best loss: 0.133095
Early stopping!
Final accuracy on test set: 0.9582729
```

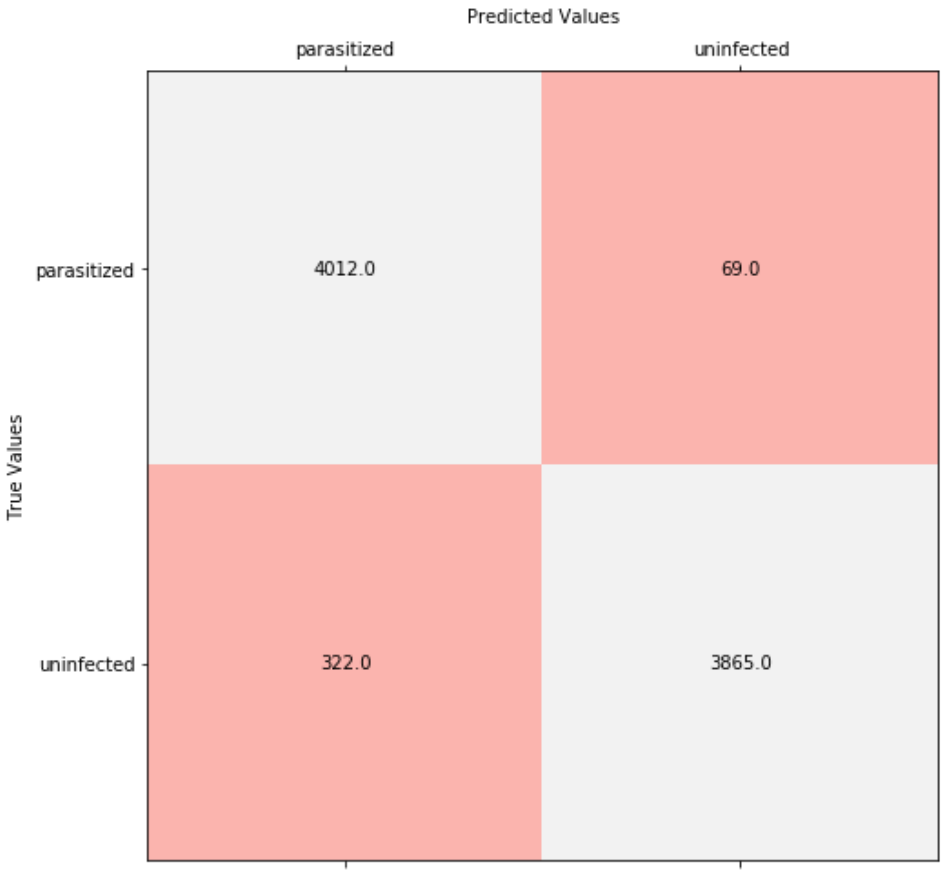
This model attenuates the effect of overfitting and it performs better on evaluating the test set. The accuracy score for this mode is 0.95827 which is better than the first model. We can calculate precision rate and recall rate which are (Precision = 0.9501204819277108, Recall = 0.9661847586375888). We are glad to see the higher recall rate because we want to model to be more sensitive when it is detecting if the cells are infected. The confusion matrix and ROC curve are presented below.

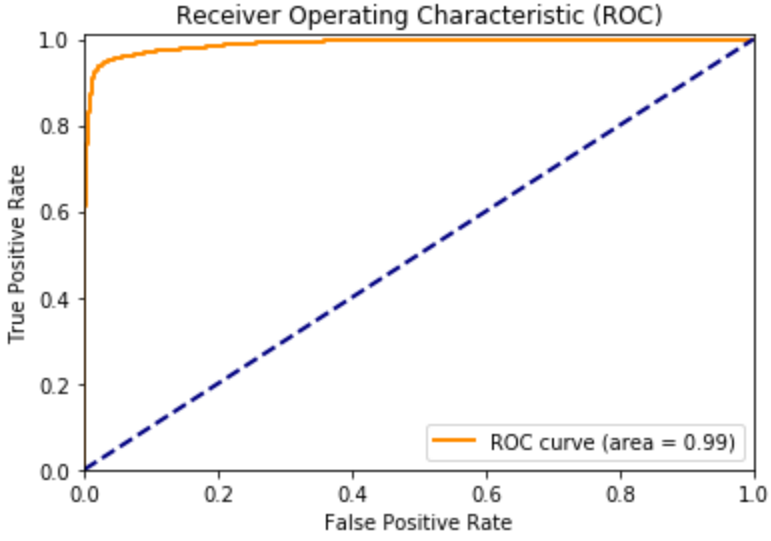




The second model can be considered as a reasonable great model for this task.

The third model is using Keras library which is for my own interest, because I am interested in the different interfaces between Keras and Tensorflow. After studying a little bit about Keras, I find that Keras is easier to set up a clear structure of layers for neural networks, so this model consists of convolutional layers, dense layers, pooling layers and layers for batch normalization. A lot of methods are encapsulated in the functions such as batch shuffle, cost function, feeding data, evaluation and prediction. The output of this model is 1 dimensional using sigmoid activation, so it indicates the probability of being an uninfected cell. By rounding this output, we can turn the probability into an exact prediction. The model also has regularization by early stopping method. The accuracy score of this model is 0.9527. We can calculate precision rate and recall rate which are (Precision = 0.925703737886479, Recall = 0.9830923793187945). We are glad to see the higher recall rate because we want the model to be more sensitive when it is detecting if the cells are infected. The confusion matrix and ROC curve are presented below.





Overall, the second model works the best and has more even performances on both precision and recall rate.