Assessment Report

The Assessment is performed on the provide database of cell images classified into 2 categories labeling

- Parasitized cells
- Uninfected cells

Data Processing and Analysis:

The data is processed to analyze image size with respect to each other as convolutional neural networks(CNNs) perform purely on different image sizes.

This problem is looked into and resolved by resizing the 3 channel (RGB) Image into 128 by 128 pixel for model1 and 60 by 60 pixel for model2.

Data Splitting

The data was split into 3 folds consists of

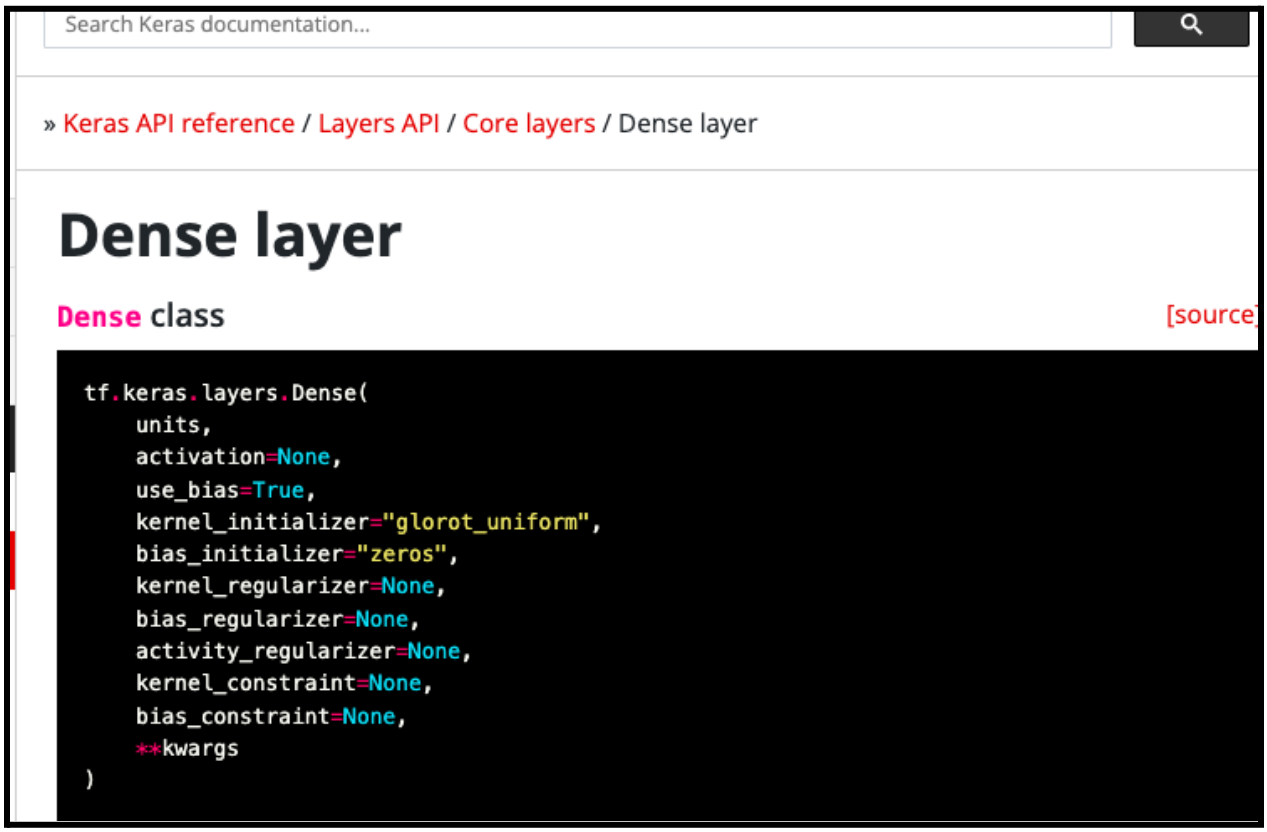
- 75% train set - used for training the data
- 15% valid set - unbiased evaluation of model fit on training data for tuning hyperparameters.
- 10% test set - used to test the trained data

Model Designing and Evaluation and Errors(or mistakes in testing data)

Model1 - Evaluated and Designed on the specifications from Assessment Documentation.

- Model 1 was set to run on 25 epochs but the model stopped at 7th epoch using the EarlyStopping function with patience equal 2. (EarlyStopping - It was a set up to monitor loss metric, assuming the goal of the training is to minimize loss.) — this function helps in stopping model training when it stops learning, which decreases model's time complexity.

- Xavier Weight(or glorot_uniform) is popular and aims to reduce the vanishing and exploding gradient problem.
  According to Keras documentation, the dense layers have a default Xavier weight function for kernel initialization.



- The model 1 evaluated with the accuracy of 69.30%, which lead 49.3% accuracy for F1 score on test data and lead to whole predictions from the test data, a total of 1396 images as mistakes out of 2750(uninfected + parasitized)

```
errors = []

for i in range(len(x_test1)):
    if y_test1[i] != y_pred1[i]:
        errors.append(i)

print("Number of mistakes: {}".format(len(errors)))

plt.figure(1 , figsize = (15 , 9))
n = 0
for i in range(49):
    n += 1
    # r = np.random.randint( 0  , X_test.shape[0] , 1)
    plt.subplot(7 , 7 , n)
    plt.subplots_adjust(hspace = 0.5 , wspace = 0.5)
    plt.imshow(x_test1[errors[i]])
    plt.title('true: {}; pred {}:'.format(y_test1[errors[i]] , y_pred1[errors[i]]))
    plt.xticks([]) , plt.yticks([])
```

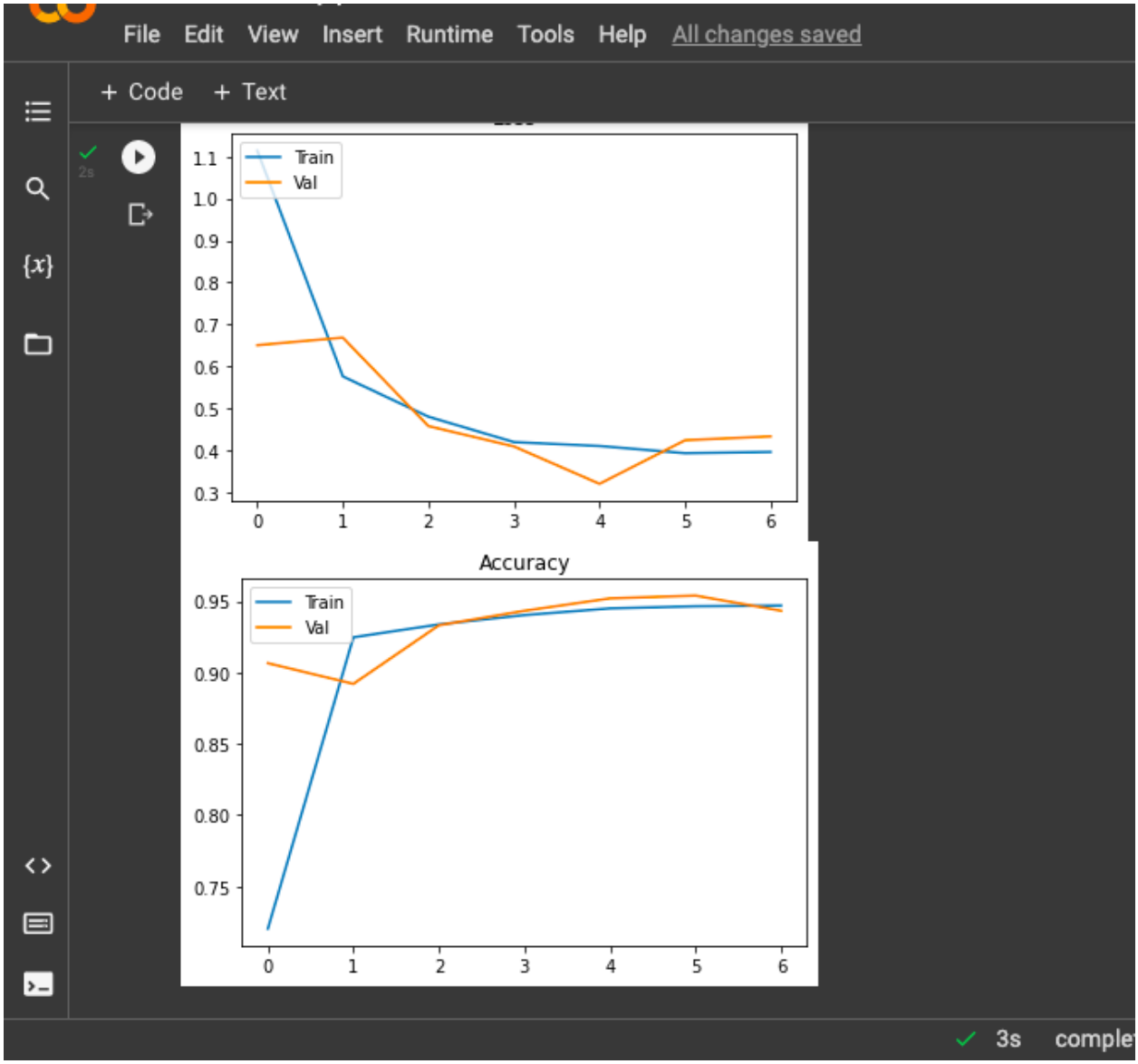
```
Number of mistakes: 1396
```

- F1, precision and recall calculated as classification metrics

```
[40] from sklearn.metrics import confusion_matrix , classification_report , accuracy_score

     y_pred1 = model1.predict(x_test1)
     y_pred1 = [1 if i>=0.5 else 0 for i in y_pred1]

     print('{} \n{} \n{}'.format(confusion_matrix(y_test1, y_pred1) ,
                                 classification_report(y_test1 , y_pred1) ,
                                 accuracy_score(y_test1 , y_pred1)))
```

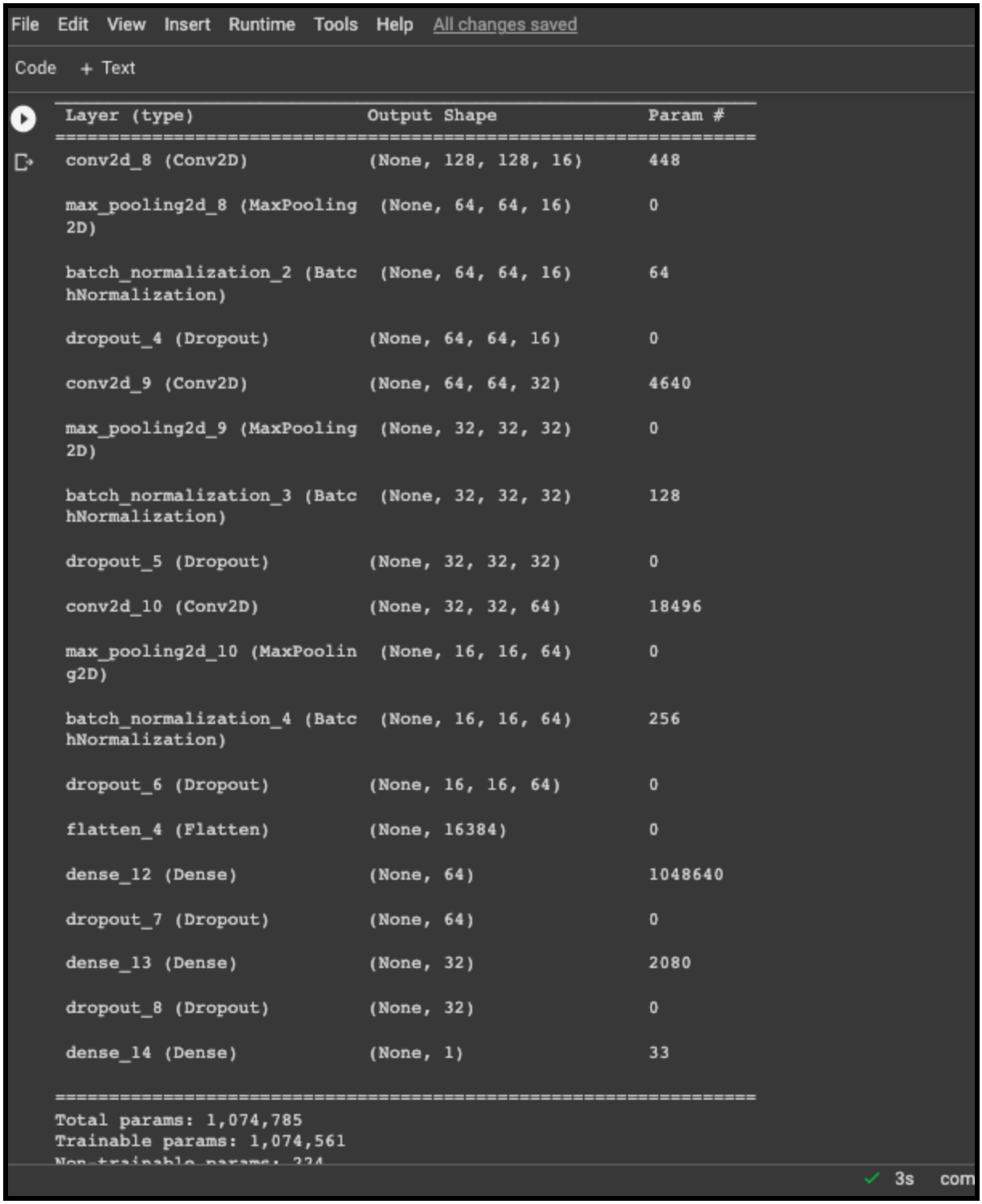
```
87/87 [==============================] - 35s 400ms/step
[[   0 1396]
 [   0 1360]]
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00      1396
         1.0       0.49      1.00      0.66      1360

    accuracy                           0.49      2756
   macro avg       0.25      0.50      0.33      2756
weighted avg       0.24      0.49      0.33      2756

0.4934687953555878
```

- Loss and Accuracy for train and valid set is evaluated for model1 and demonstrated visually.



Model1 - New approach to improve model performance

- Resized using 60*60 pixel image.
- Model redesigned using 3 CNN layers(kernel = (2,2)) followed by Maxpooling2D layer, BatchNormalization Layer and Dropout layer of 20 probability.
  CNNs layers are accompanied by 3 Dense layers (1 layer with L2 regularization) followed by a dropout layer of 30% probability.
  Eventually decreasing model parameters, resulting in decrease of model space and time complexity.



- Model2 Evaluated with the accuracy of 93.18% with improved Classification metrics.

```
[47] from sklearn.metrics import confusion_matrix , classification_report , accuracy_score

     y_pred2 = model2.predict(x_test1)
     y_pred2 = [1 if i>=0.5 else 0 for i in y_pred2]

     print('{} \n{} \n{}'.format(confusion_matrix(y_test1, y_pred2) ,
                                 classification_report(y_test1 , y_pred2) ,
                                 accuracy_score(y_test1 , y_pred2)))
```

```
87/87 [==============================] - 13s 148ms/step
[[1366   30]
 [ 158 1202]]
              precision    recall  f1-score   support

         0.0       0.90      0.98      0.94      1396
         1.0       0.98      0.88      0.93      1360

    accuracy                           0.93      2756
   macro avg       0.94      0.93      0.93      2756
weighted avg       0.94      0.93      0.93      2756

0.9317851959361393
```

- The model mistake on the testing dataset is a total of 188 images out of 2750(uninfected + parasitized) which is far less than the earlier model.

```
+ Code  + Text

plt.figure(1 , figsize = (15 , 9))
n = 0
for i in range(49):
    n += 1
    # r = np.random.randint( 0  , X_test.shape[0] , 1)
    plt.subplot(7 , 7 , n)
    plt.subplots_adjust(hspace = 0.5 , wspace = 0.5)
    plt.imshow(x_test1[errors[i]])
    plt.title('true: {}; pred {}:'.format(y_test1[errors[i]] , y_pred2[errors[i]]))
    plt.xticks([]) , plt.yticks([])
```

```
Number of mistakes: 188
```