Diabetic Retinopathy Prediction

FinTech Project

Nov. 30th, 2019

Maria Jesus Perez

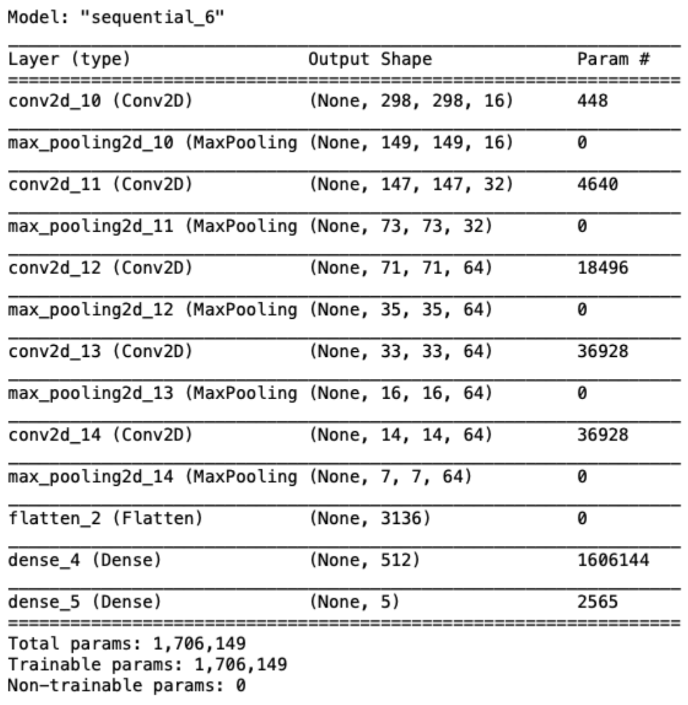
Table of Contents

Introduction

The fastest growing cause of blindness is diabetic retinopathy or DB. This occurs when high blood sugar causes damage to blood vessels in the retina causing vision problems. There are four stages of DR, including mild non-proliferative when small areas of the blood vessels swell, moderate non-proliferative when entire blood vessels swell, severe non-proliferative when several blood vessels blocked, and proliferative when new blood vessels are formed and retinal detachment is possible. Interestingly, early detection can reduce the risk of blindness by 95%. However, the lack of doctors and inconsistency on eye exams causes difficulty of detection. This project focuses on leveraging machine learning techniques to improve accuracy in eye exam detection and the overall impact of DR in society.

Convolutional Neural Network

The first machine learning technique explored was a convolutional neural network (CNN). This type of deep learning algorithm takes images as inputs and assigns importance to various aspects of the image and is able to differentiate them. In this case, we receive a dataset of 1500 retina images and assign 70% as training data and 30% as test data. We started by building a small CNN with an input, five hidden layers, and an output layer. According to CNN structure, each layer has 3 sections including a linear convolution, a non-linear activation function, and a dimensionality reduction or pooling function. A summary of the first model is shown in figure 1 below.



```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_10 (Conv2D)           (None, 298, 298, 16)      448
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 149, 149, 16)      0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 147, 147, 32)      4640
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 73, 73, 32)        0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 71, 71, 64)        18496
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 35, 35, 64)        0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 33, 33, 64)        36928
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 14, 14, 64)        36928
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 7, 7, 64)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 3136)              0
_________________________________________________________________
dense_4 (Dense)              (None, 512)               1606144
_________________________________________________________________
dense_5 (Dense)              (None, 5)                 2565
=================================================================
Total params: 1,706,149
Trainable params: 1,706,149
Non-trainable params: 0
```

Figure 1. Summary of CNN Model 1

This first model achieved a test set accuracy of 0.7166 and the training set accuracy and loss shown in figure 2 below.

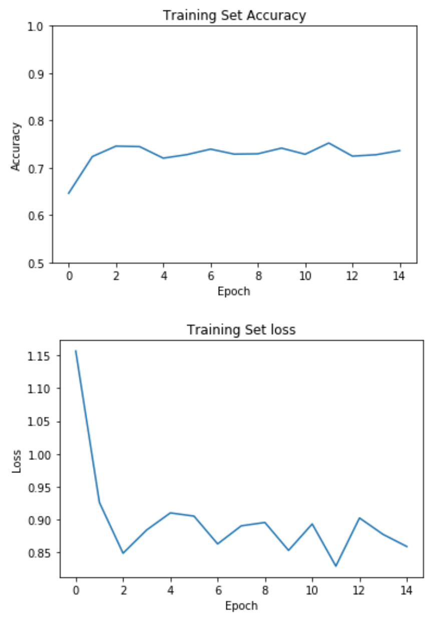


Figure 2. CNN Model 1 Performance

Seeing the relatively low accuracy, I decided to increase the training epochs from 15 to 30 and the steps per epoch from 8 to 15. This second model achieved a test set accuracy of 0.7133 and a training set accuracy shown in figure 3 below.

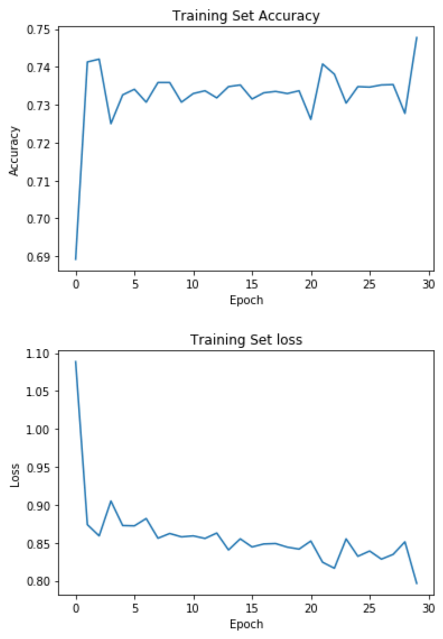
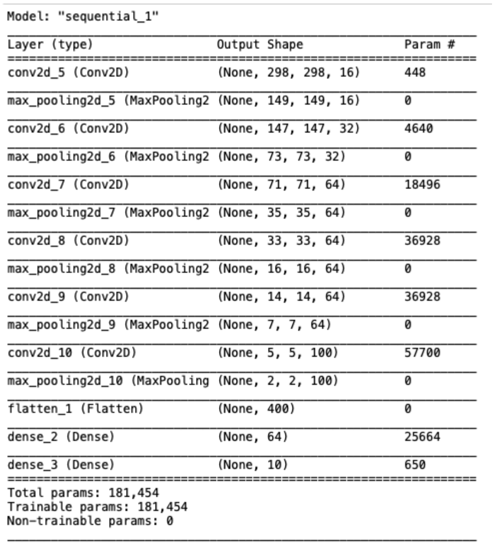


Figure 3. CNN Model 2 Performance

The test set accuracy was actually lower than the first model. The lower accuracy and increased computation time do not justify the increase in epochs and we revert back to the original number of epochs and steps. For the next model we also add an additional layer. A summary of the third CNN model is found below.



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 298, 298, 16)      448
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 149, 149, 16)      0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 147, 147, 32)      4640
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 73, 73, 32)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 71, 71, 64)        18496
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 35, 35, 64)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 33, 33, 64)        36928
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 14, 14, 64)        36928
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 5, 5, 100)         57700
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 2, 2, 100)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 400)               0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                25664
_________________________________________________________________
dense_3 (Dense)              (None, 10)                650
=================================================================
Total params: 181,454
Trainable params: 181,454
Non-trainable params: 0
_________________________________________________________________
```

Figure 4. CNN Model 3 Summary

Adding an additional layer proved to add no predictive value to the model. The test set accuracy was the same as in model 1, 0.7166. The training set performance of model 3 is shown in figure 5 below.

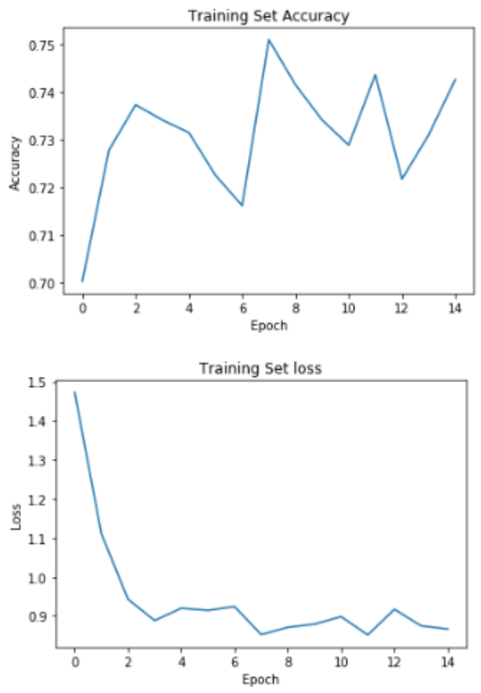
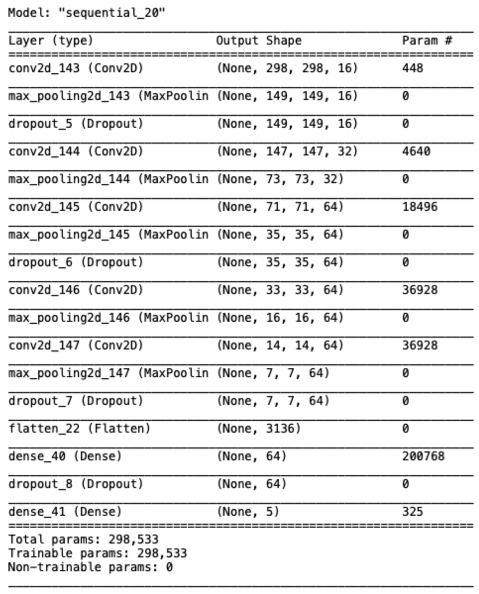


Figure 5. CNN Model 3 Performance

In order to increase the accuracy of the model and prevent overfitting we choose to add dropout functions throughout the network. These functions do not take into consideration a percentage of randomly selected neurons during training. A summary of model 4 is shown in figure 6 below.



```
Model: "sequential_20"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_143 (Conv2D)          (None, 298, 298, 16)      448       
_________________________________________________________________
max_pooling2d_143 (MaxPoolin (None, 149, 149, 16)      0         
_________________________________________________________________
dropout_5 (Dropout)          (None, 149, 149, 16)      0         
_________________________________________________________________
conv2d_144 (Conv2D)          (None, 147, 147, 32)      4640      
_________________________________________________________________
max_pooling2d_144 (MaxPoolin (None, 73, 73, 32)        0         
_________________________________________________________________
conv2d_145 (Conv2D)          (None, 71, 71, 64)        18496     
_________________________________________________________________
max_pooling2d_145 (MaxPoolin (None, 35, 35, 64)        0         
_________________________________________________________________
dropout_6 (Dropout)          (None, 35, 35, 64)        0         
_________________________________________________________________
conv2d_146 (Conv2D)          (None, 33, 33, 64)        36928     
_________________________________________________________________
max_pooling2d_146 (MaxPoolin (None, 16, 16, 64)        0         
_________________________________________________________________
conv2d_147 (Conv2D)          (None, 14, 14, 64)        36928     
_________________________________________________________________
max_pooling2d_147 (MaxPoolin (None, 7, 7, 64)          0         
_________________________________________________________________
dropout_7 (Dropout)          (None, 7, 7, 64)          0         
_________________________________________________________________
flatten_22 (Flatten)         (None, 3136)              0         
_________________________________________________________________
dense_40 (Dense)             (None, 64)                200768    
_________________________________________________________________
dropout_8 (Dropout)          (None, 64)                0         
_________________________________________________________________
dense_41 (Dense)             (None, 5)                 325       
=================================================================
Total params: 298,533
Trainable params: 298,533
Non-trainable params: 0
_________________________________________________________________
```

Figure 6. CNN Model 4 Summary

Although we added 3 dropout functions that take into consideration only 80% of the neurons in their respective layer, the test set accuracy was not significantly improved. Interestingly, the test set accuracy remained exactly the same at 0.7166. The training set performance of model 4 is shown in figure 7 below.

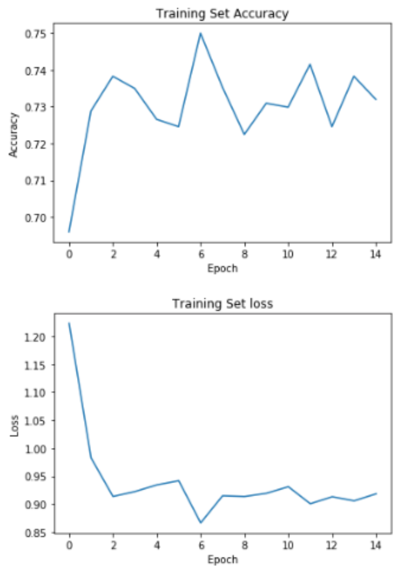
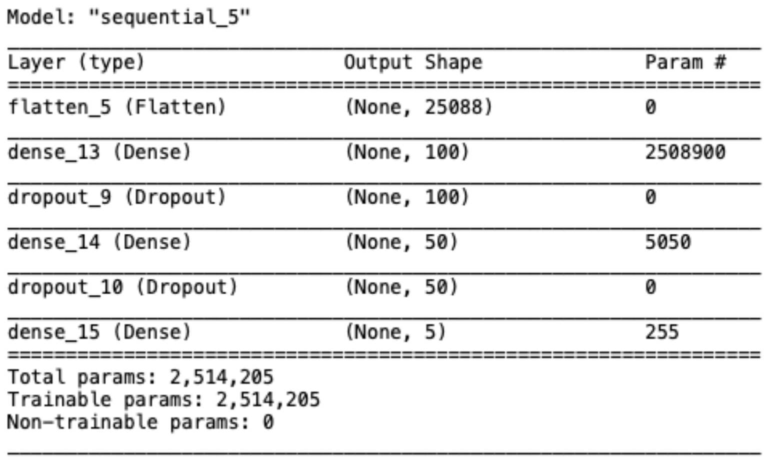


Figure 7. CNN Model 4 Performance

In order to further improve the accuracy of the model and in pursuit of the 98% accuracy goal, we decide to utilize transfer learning. We import the VGG16 model that comes with 11 pre-made convolutional layers. We notice an increased computational speed and therefore increase the epochs to 100. A summary of the transfer learning based model is shown in figure 8 below.



```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_5 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_13 (Dense)             (None, 100)               2508900
_________________________________________________________________
dropout_9 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_14 (Dense)             (None, 50)                5050
_________________________________________________________________
dropout_10 (Dropout)         (None, 50)                0
_________________________________________________________________
dense_15 (Dense)             (None, 5)                 255
=================================================================
Total params: 2,514,205
Trainable params: 2,514,205
Non-trainable params: 0
_________________________________________________________________
```

Figure 8. CNN Model 5 Summary

The performance of the model on the training set model increased significantly over the epochs. At 100 epochs the training set accuracy reached more than 0.9. Figure 9 shows that a high training set accuracy but this came at the cost of test set accuracy. The final accuracy was only 0.66 which is lower than the previous models and shows an overfitting problem.

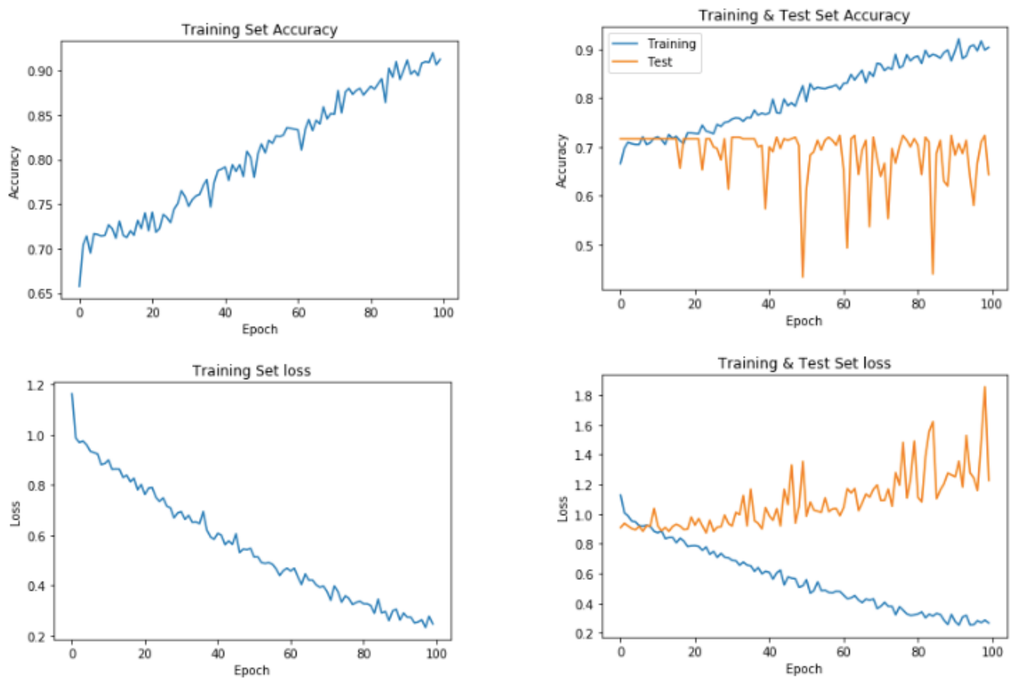


Figure 9. CNN Model 5 Performance

To help us better visualize the results of the model we built the confusion matrix shown in figure 10 below. The model is performing well with images that do not have the disease but not so much with the others. This is probably because the number of images in the training set without the disease was much higher than those that did not.

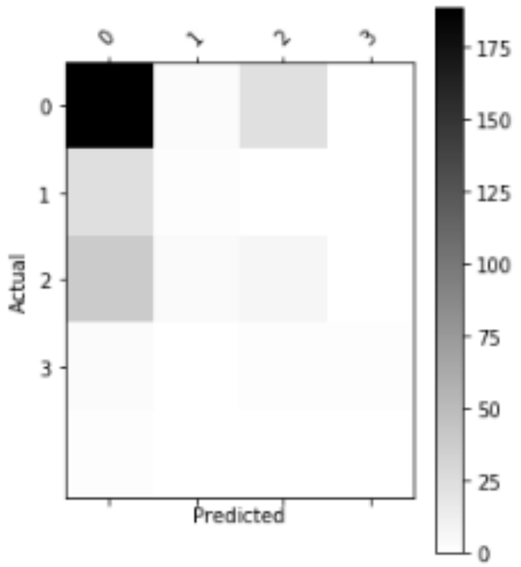
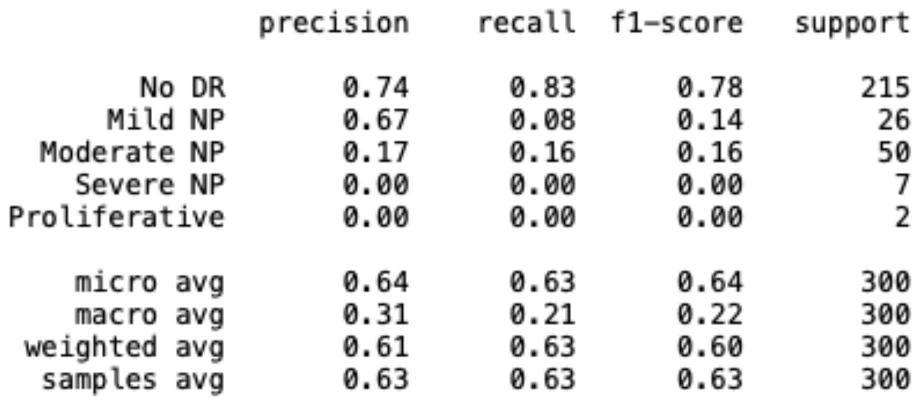


Figure 10. CNN Model 5 Confusion Matrix

We can also look at the classification metrics to examine the model's performance in more detail. Figure 11 below shows the precision, recall, and f1-scores for each category of the model. Supporting the findings from the confusion matrix, we see that the precision and recall scores are significantly higher for the no disease category than any other category. In order to improve the model and combat the overfitting problem I believe it is necessary to add additional images that are better balanced throughout the five categories.



```
               precision    recall  f1-score   support

        No DR       0.74      0.83      0.78       215
      Mild NP       0.67      0.08      0.14        26
  Moderate NP       0.17      0.16      0.16        50
    Severe NP       0.00      0.00      0.00         7
Proliferative       0.00      0.00      0.00         2

    micro avg       0.64      0.63      0.64       300
    macro avg       0.31      0.21      0.22       300
 weighted avg       0.61      0.63      0.60       300
  samples avg       0.63      0.63      0.63       300
```

Figure 11. CNN Model 5 Classification Metrics

Support Vector Machine

The second machine learning technique explored was a support vector machine (SVM). This type of algorithm also takes images as inputs and assigns importance to various aspects of the image to differentiate them. The difference is that a CNN is non-linear whereas a SVM is a linear classifier. One of the main differences between a CNN and SVM is the level of image pre-processing required. Whereas the neural network allows us to feed a data-generator with images, a support vector machine requires us to transform images into vectors that the algorithm can understand and process. Since the total data set consists of 1500 images then the computation time was very elevated. We then built a feature matrix with all the images converted into vectors of 270,000 components and scaled it before feeding it into a SVM model. We utilized the default parameters for training, including a random state of 42, a regularization term of 1, and an auto-generated gamma. The complete information on the model is shown in figure 12 below.

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='linear', max_iter=-1, probability=True, random_state=42,
  shrinking=True, tol=0.001, verbose=False)
```

Figure 12. SVM Model Summary

We decided against cross validation for the selection of the parameters because of the computational intensiveness. In fact, the model with the default parameters took 3 hours and 49 minutes to give predictions for the 300 images in the test set. Therefore, we conclude not doing cross validation was the best decision. Nevertheless, the accuracy of the model does not support this conclusion. The model has an accuracy of only 0.633. A confusion matrix of the predicted vs actual results is shown in figure 13 below. In similarity to the CNN, the model is effective only in images without the disease.

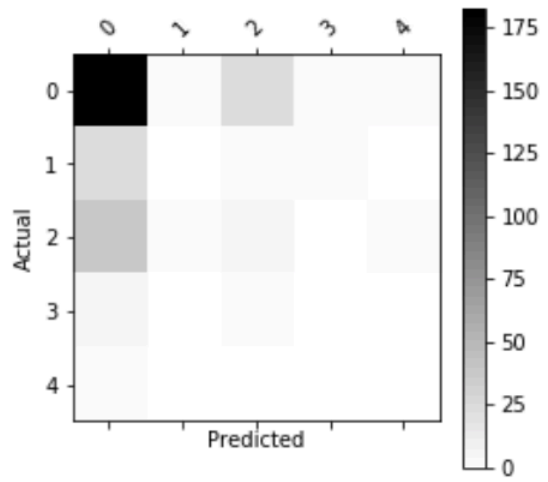
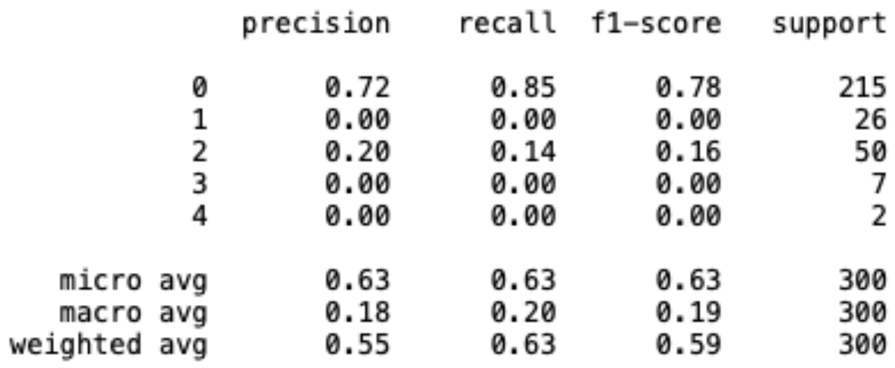


Figure 13. SVM Model Confusion Matrix

We can obtain a more detailed outline of the model’s performance using the classification metrics. Supporting the conclusion from the confusion matrix, the model has a relatively good performance with images that do not have the disease. However, this performance is not imitated in images with the disease especially as the disease gets more severe.



```
              precision    recall  f1-score   support

           0       0.72      0.85      0.78       215
           1       0.00      0.00      0.00        26
           2       0.20      0.14      0.16        50
           3       0.00      0.00      0.00         7
           4       0.00      0.00      0.00         2

   micro avg       0.63      0.63      0.63       300
   macro avg       0.18      0.20      0.19       300
weighted avg       0.55      0.63      0.59       300
```

Figure 14. SVM Model Classification Metrics

Conclusion

The purpose of this report was to develop a machine learning model to classify images of eyes according to the degree of diabetic retinopathy or DB. The vision problems caused by this disease are due to high blood sugar causing damage to blood vessels in the retina. Therefore, there are five stages of DB that we attempted to classify including: no disease, mild non-proliferative, moderate non-proliferative, severe non-proliferative, and proliferative. The first model was a convolutional neural network. After building our own network and utilizing transfer learning the best accuracy on the test set was of 0.7166. The second model was a support vector machine which achieved a smaller accuracy of 0.633. The performance of these models does not reach the goal of 98% accuracy but we think it would be difficult to achieve that with this data set. In future projects, we suggest utilizing a data set with more images for each of the stages of the disease. Otherwise, building binary classification models instead of multi-class classification would probably yield better results.

References

Mamun, Iftekher. “A Simple CNN: Multi Image Classifier.” *Medium*, Towards Data Science, 8 Apr. 2019, towardsdatascience.com/a-simple-cnn-multi-image-classifier-31c463324fa.

Brownlee, Jason. “Display Deep Learning Model Training History in Keras.” *Machine Learning Mastery*, 3 Oct. 2019, machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/.

Morris, Sharon. “Image Classification Using SVM.” *RPubs*, 2018, rpubs.com/Sharon_1684/454441.