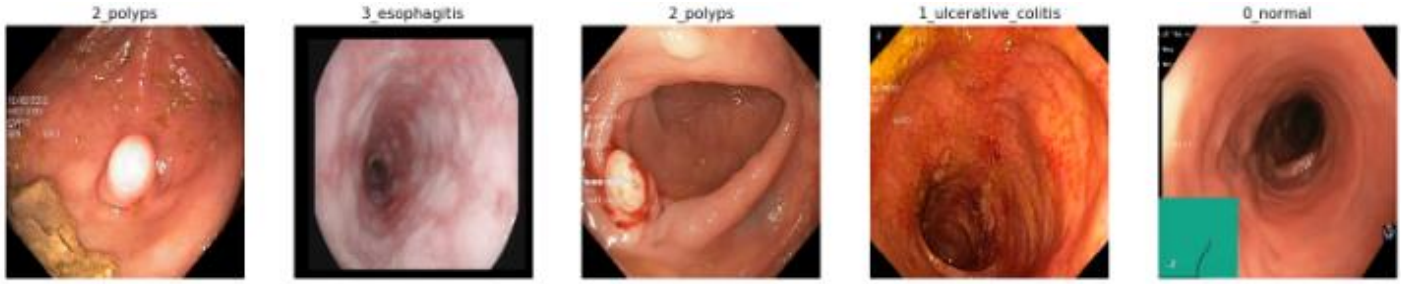
Colon Disease Image Classification Model



Background

A dataset was obtained from Kaggle containing images of colons broken down by test, train, and validation sets. Each set contains 4 classes: normal, ulcerative colitis, polyps, and esophagitis. To a trained eye characterizing these diseases visually may be no problem, especially for physicians overseeing tests and procedures. The issue with classification of diseased colons, is a classification one.

An article by the CDC states that an estimated 15 million colonoscopies were performed in 2012, but only half of people aged 50-75 had been up to date on their screening. New technology has become available to make this process less invasive, and an initiative has been launched by the CDC to increase screenings of this population to 80 percent. The new technology available to patients is a camera that they can swallow that takes images as it passes through the digestive track. With an increase in colonoscopies performed and this new tech, the workload of those reviewing these images is bound to increase.

A deep learning model which performs image classification for the colon diseases stated in this data set could be applied to patient images to flag them as diseased for review. This project is constrained to the diseases contained within the dataset; however, future projects should focus on diversifying diseased images to widen the application of this model.

Data Wrangling and Exploratory Data Analysis

Images from each disease class file folder were loaded into Jupyter Notebook and evaluated. Since the difference in class was mostly distinguishable by the naked eye, it is expected to have a model that performs fairly well (>90%). Each folder was investigated to see quantity per disease class. Figures 1-3 show that each set has an equal distribution of classes. No generation of new images is need, as the data is not sparse or uneven. The training set has 4 classes with 800 images each totaling 3200 images. The validation set has 500 images in each class totaling 2000 images. The test set has 200 images in each class totaling 800 images.

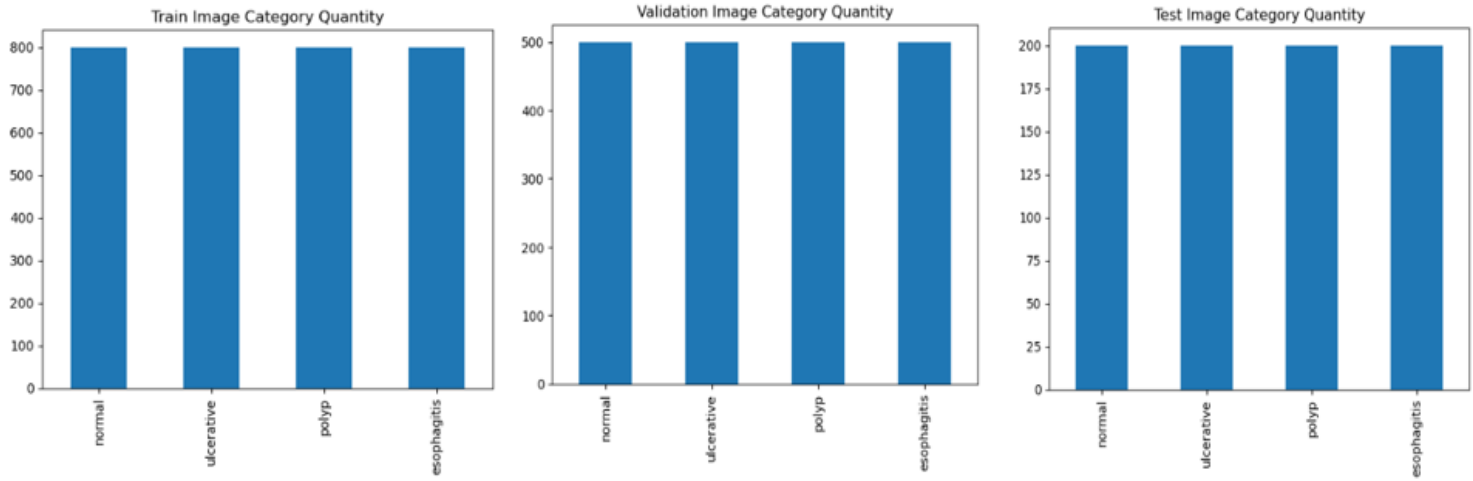


Figure 1 Train, Validation, and Test Class Quantities

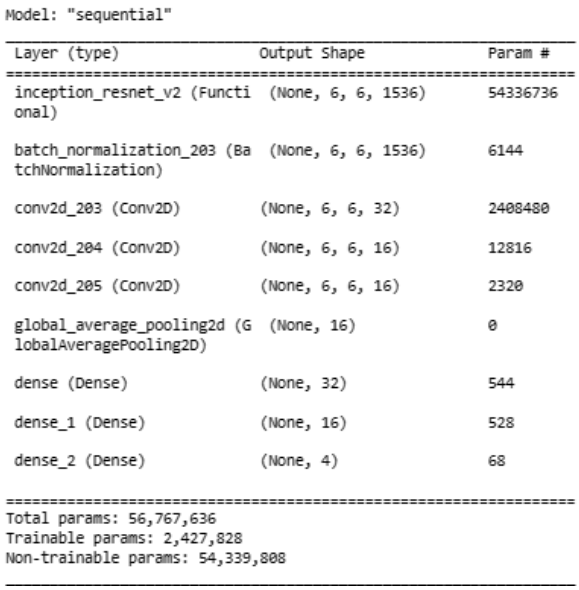
Keras Tensorflow was used to preprocess the data. ImageDataGenerator allowed for each image set to be prepared to be run by the model. Checks on the variables made showed that no data was lost in the process, and images were able to be viewed from the generated format. The image size for each set was (256,256,3).

Modelling

Transfer learning was used as the base model. The base model used was InceptionResNetV2, provided Cornell University (arXiv:1602.07261). This application states that it has been able to achieve competitive performance at low computational processing cost. The team who built this

model, also published that they had seen improvement when used with residual connections. This factor is ideal in a project where fine tuning is needed.

Residual layers added to the base model included BatchNormalization, this layer normalizes images by making each mean and standard deviation the same. This is needed in the model to reduce interference of diverse image qualities. The following three layers were Conv2D, in order to generate element multiplication by sliding over image arrays. An attempt to implement Maxpooling2D was made, but model accuracy decreased with the addition of this layer. A globalaveragepooling2D layer implemented in the model in order to condense features by taking mean of image height and width. The final three layers in the model were dense layers, with the last dense layer having an output equal to four for the four classes we want to classify. The model summary can be seen in figure 2.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_resnet_v2 (Functi  (None, 6, 6, 1536)       54336736
 onal)

 batch_normalization_203 (Ba  (None, 6, 6, 1536)       6144
 tchNormalization)

 conv2d_203 (Conv2D)         (None, 6, 6, 32)          2408480

 conv2d_204 (Conv2D)         (None, 6, 6, 16)          12816

 conv2d_205 (Conv2D)         (None, 6, 6, 16)          2320

 global_average_pooling2d (G  (None, 16)               0
 lobalAveragePooling2D)

 dense (Dense)               (None, 32)                544

 dense_1 (Dense)             (None, 16)                528

 dense_2 (Dense)             (None, 4)                 68

=================================================================
Total params: 56,767,636
Trainable params: 2,427,828
Non-trainable params: 54,339,808
_________________________________________________________________
```

Figure 2 Model Summary (Layer, Output Shape, Param)

The model was compiled and fit on the training data. Epochs were set to 15, Adam was used as the optimizer with a learning rate of 0.001, and Categorical Cross Entropy was used to estimate loss. A call back function was specified with patient of three and set to restore best weights of the model. This call back function stopped our model fitting at an epoch of 7, as shown in figure 3.

```
Epoch 1/15
100/100 [==============================] - 968s 10s/step - loss: 0.2822 - accuracy: 0.8934 - val_loss: 0.2271 - val_accurac
y: 0.9215
Epoch 2/15
100/100 [==============================] - 958s 10s/step - loss: 0.0813 - accuracy: 0.9756 - val_loss: 0.1459 - val_accurac
y: 0.9450
Epoch 3/15
100/100 [==============================] - 924s 9s/step - loss: 0.0623 - accuracy: 0.9809 - val_loss: 0.1515 - val_accurac
y: 0.9485
Epoch 4/15
100/100 [==============================] - 945s 9s/step - loss: 0.0211 - accuracy: 0.9934 - val_loss: 0.1428 - val_accurac
y: 0.9510
Epoch 5/15
100/100 [==============================] - 928s 9s/step - loss: 0.0227 - accuracy: 0.9928 - val_loss: 0.2189 - val_accurac
y: 0.9385
Epoch 6/15
100/100 [==============================] - 940s 9s/step - loss: 0.0196 - accuracy: 0.9944 - val_loss: 0.3808 - val_accurac
y: 0.9255
Epoch 7/15
100/100 [==============================] - 931s 9s/step - loss: 0.0484 - accuracy: 0.9887 - val_loss: 0.2643 - val_accurac
y: 0.9240
```

Figure 3 Model Epochs

Model Evaluation

Figure 4 shows training and validation accuracy at each epoch. Note that the X axis starts at 0 instead of 1 along with the epoch. Epoch 4 is the best fit model when considering both training and validation sets with accuracies of 99 and 95 percent respectively.

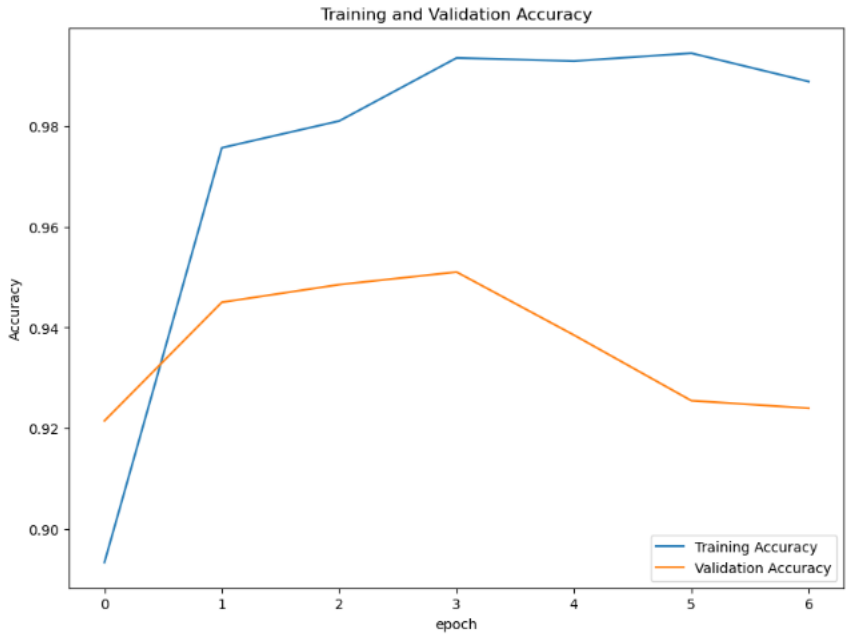


Figure 4 Model Accuracy on Training and Validation Sets

Figure 5 shows model loss on the training and validation sets. Note the X axis scaling is a zero start on the first epoch. Epoch 4 has the lowest loss for both training and validation sets with a loss of 0.02 and 0.14 respectively.

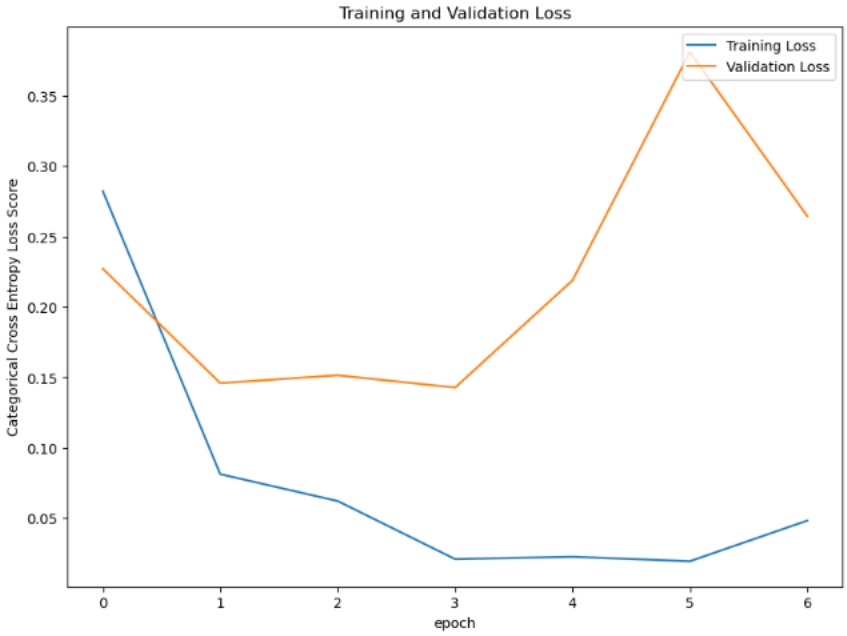


Figure 5 Model Loss on Training and Validation Sets

Figure 6 shows the confusion matrix for the true train set labels versus class labels predicted by the model. Each category class for the train set has over 99 percent accuracy.

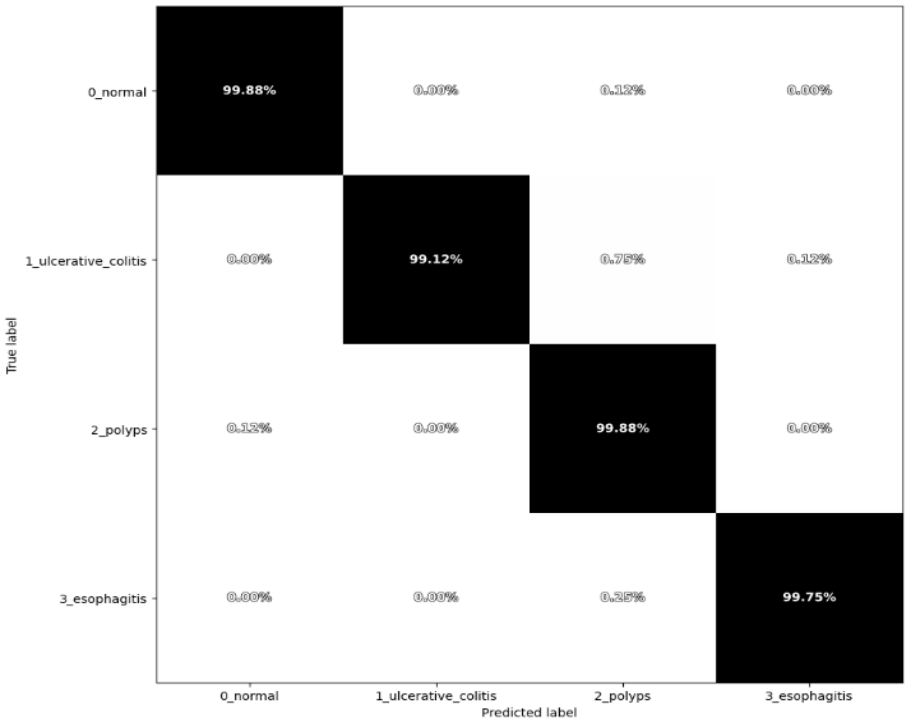


Figure 6 Train Set Model Confusion Matrix

Figure 7 shows the confusion matrix for the true validation set labels versus class labels predicted by the model. Normal and esophagitis class predictions were above 99 percent, with polyp class prediction at 97 percent. The ulcerative colitis class had only 84 percent accuracy, but it should be noted that the other predictions made by the model were for disease classes, no predictions for ulcerative colitis were made under the normal class. This shows that the model is efficient at predicting between normal and diseased colon images.

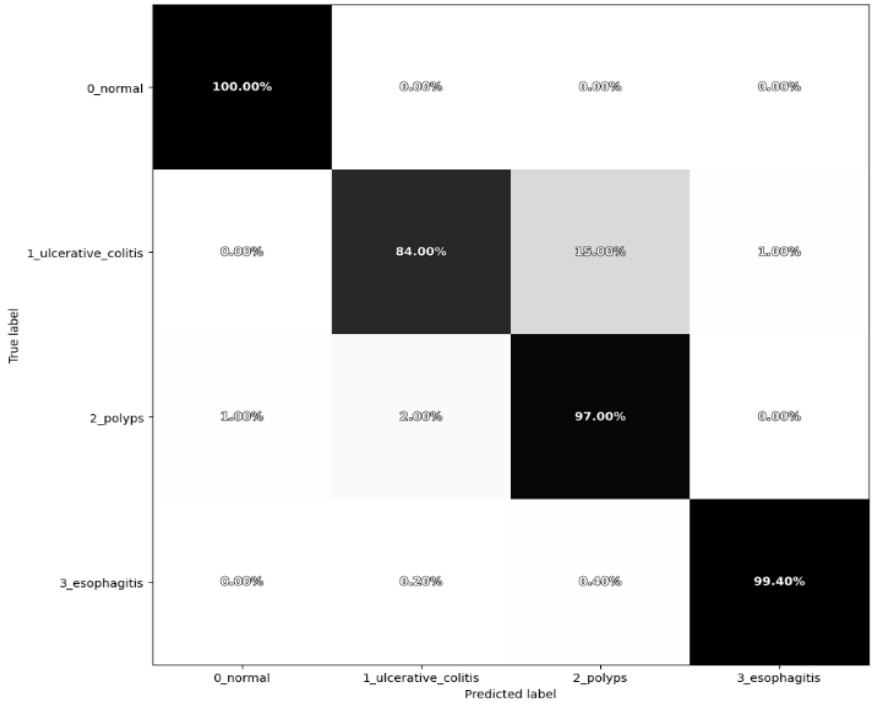


Figure 7 Validation Set Model Confusion Matrix

Figure 8 shows the confusion matrix for the true test set labels versus class labels predicted by the model. Normal class colons had 100 percent accuracy. Esophagitis and polyp class prediction accuracy was at 98.5 and 96 percent, respectively. The ulcerative colitis class had only 86 percent accuracy, like the validation set, the other predictions made by the model were for diseased classes.

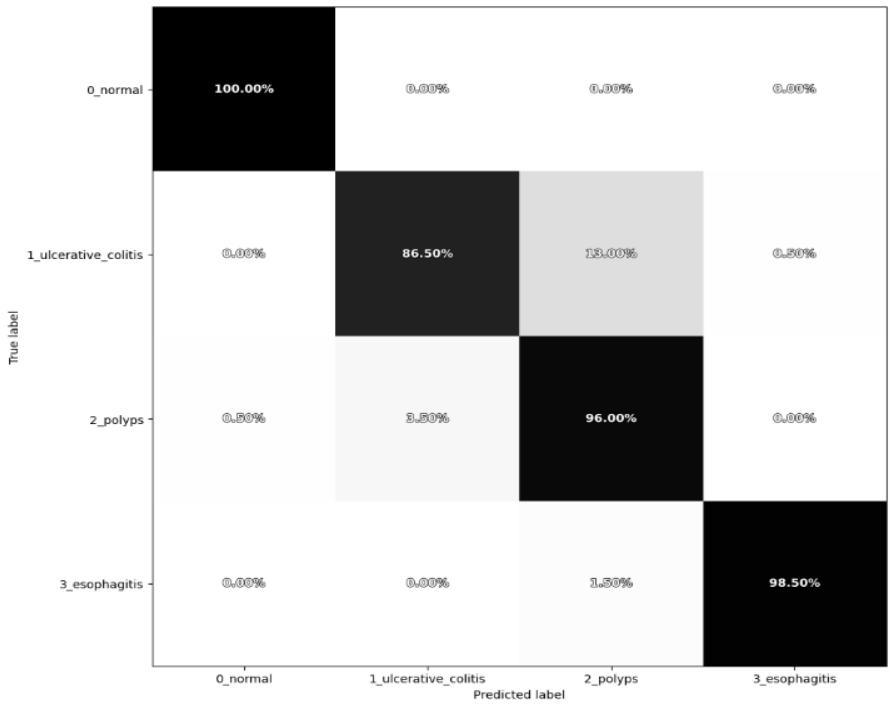


Figure 8 Test Set Model Confusion Matrix

Conclusion

Overall, model performance as an image classifier had exceptional performance in detecting diseased versus normal colon images. Both the validation and test set had 100 percent accuracy for normal class category, and the training set was not far behind at 99.88 percent. Each diseased colon image class had moderate to high predictive rates, and no misclassifications were made as normal colon class.

Other applications of this model would be to train for different disease classes and test predictive accuracy on those classes. As discussed in the background, with new swallow pill endoscopies this model could be used in mass image prediction to manage the uptick in patients receiving this test. The model could reduce workload by flagging cases for review that do not fall under the normal class, since false negative rates are next to none with the model.

Future applications of image classifiers like this one evolves as medicine evolves. For diseases not known to the naked eye, but confirmed by biopsy, models like this could be trained overtime to see differences that the human eye is unable to detect.

Resources

https://www.kaggle.com/datasets/francismon/curated-colon-dataset-for-deep-learning

https://www.mayoclinic.org/tests-procedures/capsule-endoscopy/about/pac-20393366

https://www.cdc.gov/media/releases/2016/p0622-colorectal-cancer-screening.html#print

https://www.tensorflow.org/api_docs/python/tf/keras/layers

https://keras.io/api/applications/inceptionresnetv2/

https://arxiv.org/abs/1602.07261