# 

# 

# 

# Ocular Disease Classification using Image Recognition

CSCE 4390/5300

Group 3

Mica Haney, Neda Langroodi, Bijesh Patel Vachanni

# Abstract

Ocular disease is an umbrella term that is used to refer to any condition that affects the eye and its ability to function properly. There are several different forms that could take place, from cataracts to diabetic retinopathy and glaucoma. Typically, in order to detect if an individual has one of these diseases, an eye examination needs to take place. This could involve a vision test, dilating the eyes, and getting imaging tests of the entire eye (see Figure 1), which includes the retina, optic nerve, macula, and blood vessels. These images then need to be assessed by a professional to try to target the key issue that is causing eye problems.

This analysis will train and deploy a machine learning model that can quickly classify a digital imaging test to identify the most likely ocular disease. Image classification will be used to determine if an eye image falls under one of the categories: a healthy eye, diabetic retinopathy, glaucoma, cataract, age related macular degeneration, hypertension, pathological myopia, and other diseases. This model will serve as a tool that can be used during digital imaging to give the medical professional a lead of what the eye test looks like based on historical images, which can give confidence to the final diagnosis and treatment plan.

# Workflow

This project is titled Ocular Disease Classification using Image Recognition. Participants include Mica Haney (micahaney@my.unt.edu), Neda Langroodi (nedalangroodi@my.unt.edu), and Bijesh Patel Vachanni (bijeshpatelvachanni@my.unt.edu). Each team member had an equal role in deciding on the project goal, coming up with a model design, testing their model, then choosing the best final design for the project. The following report and project presentations were also a collaborative effort between the team. The team communicated through Discord and kept each other updated on their projects through shared Google Drive and GitHub repository links. Shared Google Docs were also created for collaborative efforts on the project report and presentations.

# Methodology

## Data Specification

The dataset that will be used for this analysis is named the [Ocular Disease Intelligent Recognition](https://www.kaggle.com/andrewmvd/ocular-disease-recognition-odir5k) (ODIR) and has been compiled by Shanggong Medical Technology Co., Ltd. from different hospitals/medical centers in China (Kaggle, 2021). ODIR is a structured ophthalmic database of 5,000 patients with images from left and right eyes as well as diagnostic keywords specified from doctors. The dataset has eight available classifications: Normal (N), Diabetes (D), Glaucoma, (G), Cataract (C), Age-related Macular Degeneration (A), Hypertension (H), Pathological Myopia (M), and other diseases/abnormalities (O). The preprocessed folder that will be used for this analysis consists of 6,392 images available across all classes and each image has a shape of (512, 512, 3). There is also a csv file that is included in the dataset that links each file name with its target classification. All images and the csv are loaded into a Python notebook using cv2 and pandas, respectively. They are stored in separate lists where the index denotes which image matches up with which label. Then, the available number of images per classification is observed using a bar chart, shown in Figure 1. There are far more images of normal and diabetic eyes than other classifications. This will need to be considered during data preprocessing in order to balance the data input into the model for training and validation. At this point, data will be split into training, validation, and test sets; supervised learning will be used to create a convolutional neural network with the image arrays and labels.

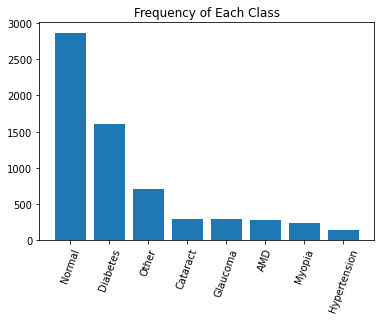
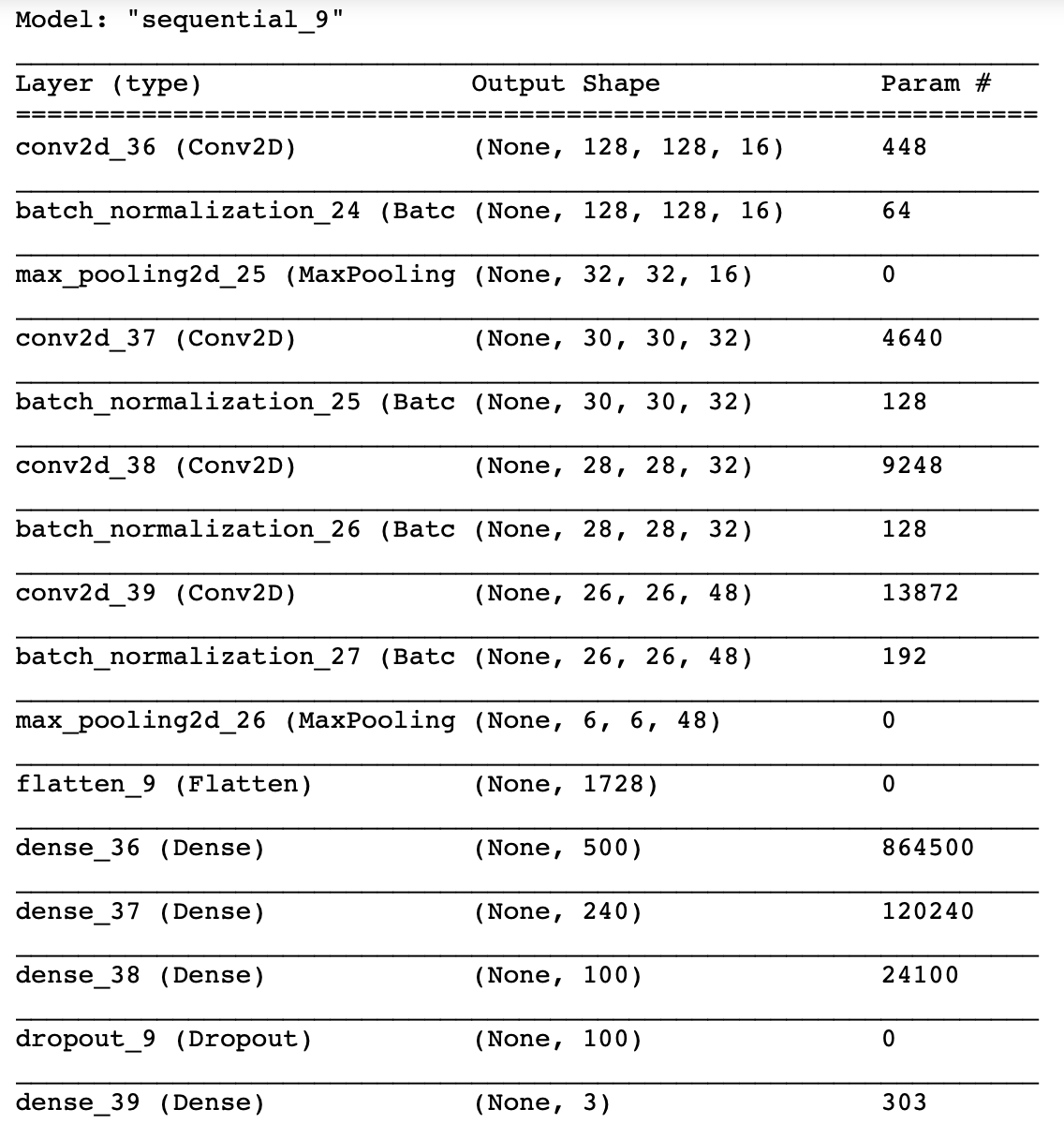


Fig1. Frequency of each classification in full dataset

## Model Design 1

The first model design constructed a convolutional neural network from scratch. First, the images are read in and stored in a list using the cv2 module. The labels are also read in using pandas and mapped to a categorical number in preparation for the CNN. Then, an 80/20 train/test split is performed using scikit-learn’s train\_test\_split() function. The shape of the images are printed out and shown to be (512, 512, 3). This image size is quite large to feed to a CNN and will require much more training time. Therefore, the images will be resized to (128, 128) in preparation for the model. Tensorflow and keras will now be used to create iterable datasets and build/train the CNN. First, a tensorflow dataset will be created for both the train and test images and labels. A batch size of 10 will be used. Then, a sequential keras model will be created. The architecture of the model is shown in Figure 2.



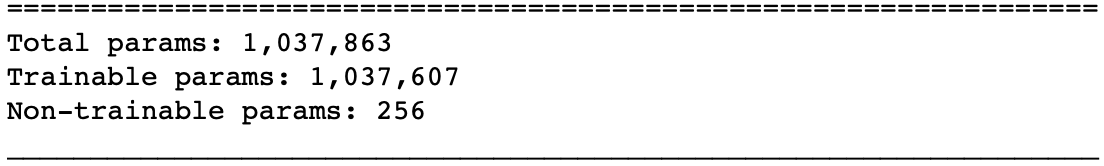


Fig2. CNN Architecture Design 1

This model is compiled using a learning rate of 1e-5 and a sparse categorical cross entropy loss function. It was then run on all of the available classifications in the original dataset for 30 epochs. However, the learning curves of this model were not satisfactory, where the validation accuracy only increased in the first five epochs then stayed stagnant around 50%. The confusion matrix for this model showed that the model was not correctly detecting several of the classes. Therefore, only three classes were chosen to continue training: Normal, Cataracts, and Myopia. This meant that imbalance class treatment needed to be performed since there were far more normal eye images than cataract and myopia images. The normal images were limited to 300 before train/test split since cataracts and myopia had a little less than 300 images per class. The model was then run again for 30 epochs and tweaked with only these three classes. The learning curves, confusion matrix, and classification report of this run are shown in the figures below.

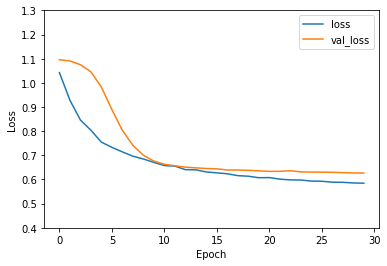
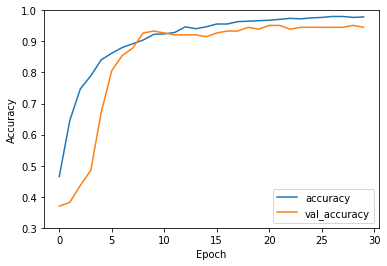


Fig3. Accuracy and Loss Learning Curves

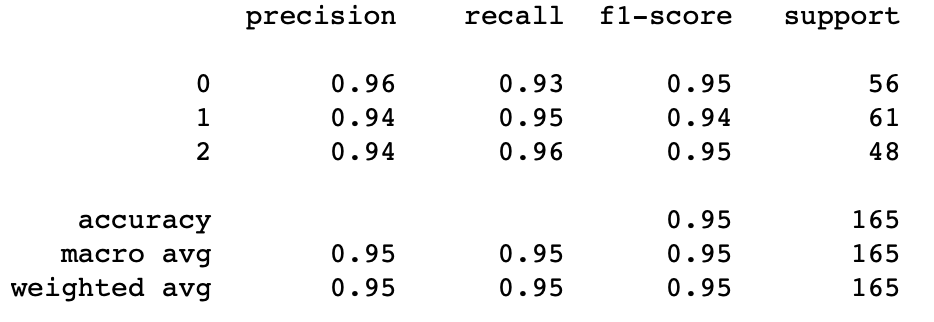
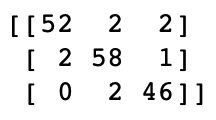


Fig4. Confusion Matrix and Classification Report (Classes: 0=Normal, 1=Cataract, 2=Myopia)

The validation accuracy reached 94.55% while the validation loss ended at 0.6269. The learning curves for this run indicate that the model did learn in conjunction with the training data and had a good cutoff at 30 epochs since learning dramatically slowed down in the last few epochs. The precision and recall for each classification is from 93-96% and the confusion matrix shows that most of the classes were correctly identified as true positives in the test data. This model shows promising results and may continue to be tweaked to improve its metrics across all available classifications.

## Model Design 2

The second model was loaded on pre-trained model VGG19 a convolutional neural network which has 19 layers .The input image size of the network was resized to (224,224) and converted these images, labels into numpy arrays. Data was split into training and testing sets using train\_test\_spilt function in Sklearn model selection. First, the Vgg19 model was loaded using an application interface in keras. Then, a sequential model was created. Later, Flatten layer and sigmoid activation function were added to the model.

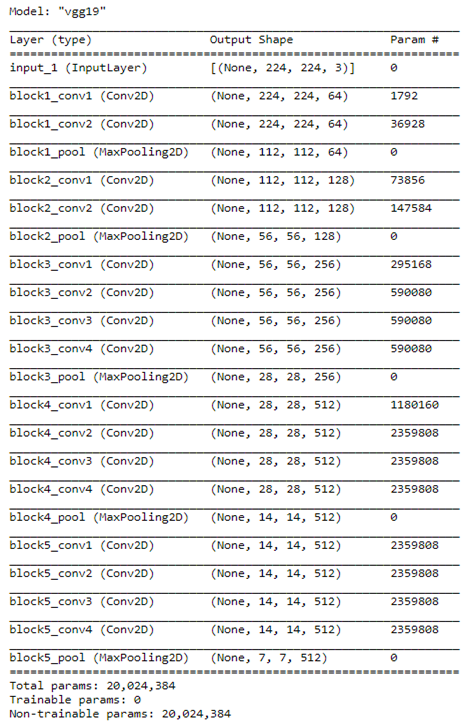
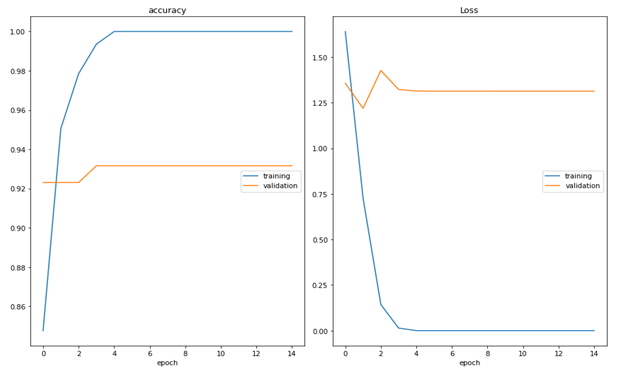
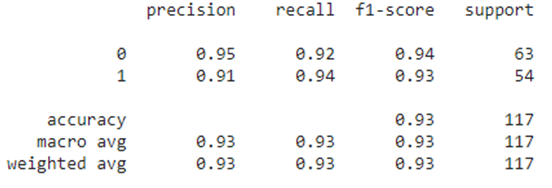
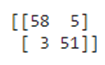


Fig5. VGG19 Architecture Design

The model was compiled using Adam optimizer and loss as binary cross-entropy. Keras model checkpoint was used to save the model weights at frequent intervals by passing the monitor – validation accuracy, save best only - True, mode - max and verbose - 1. To reduce learning rate when there is no improvement in the metric we used ReduceLROnPlateau by passing monitor - validation loss, factor - 0.1, minimum learning rate - 0.00001 and mode - auto. To balance both Normal and Cataract Images we reduced Normal class to 290 images. We trained the model using the fit method by passing batch size - 32, epochs - 15, training data, validation data and callbacks function. The learning curve plots of loss, accuracy on training and validation sets was created using keras PlotLossesKerasTF method.

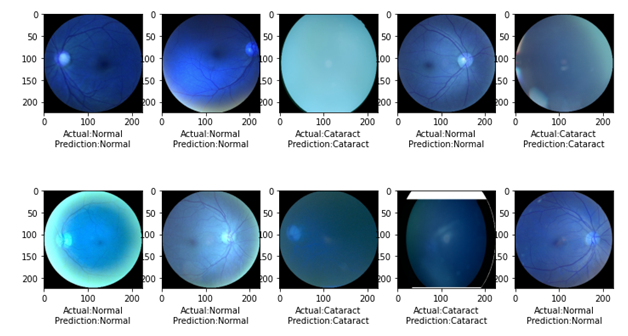


**Fig6. Accuracy and Loss Learning Curves**

 **Fig7.**  **Confusion Matrix and Classification Report (Classes: 0=Normal, 1=Cataract)**

To evaluate the model performance we have built a classification report and confusion matrix on test data using Sklearn metrics. We obtained 93% accuracy using the VGG19 model with epochs – 15. The training accuracy converged reasonably quickly but the validation accuracy did not improve. The confusion matrix shows that both Normal and Cataract classes were predicted correctly on test data. Using the VGG19 pre-trained model we got good results but compared to the first model the accuracy did not improve on test data.

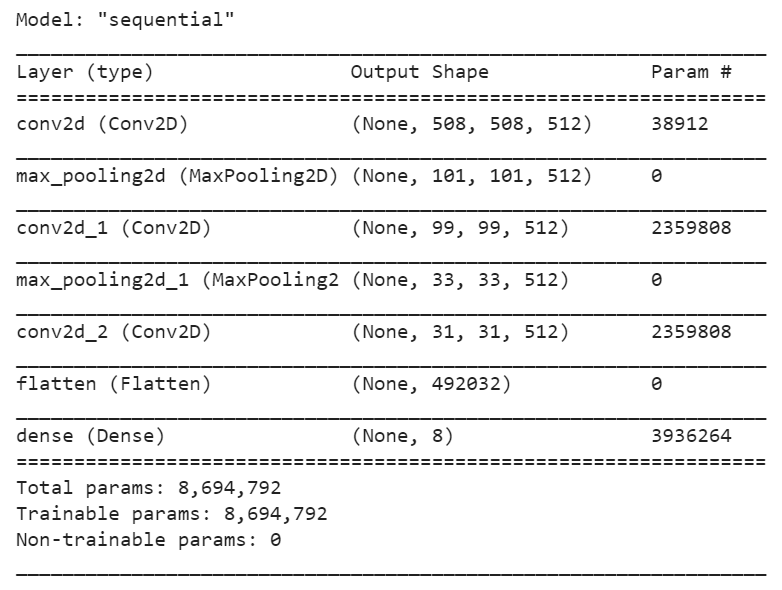
The model results on test data with actual and predicted classes are shown below:



**Fig8.**  **The Output Results with actual and predicted classes**

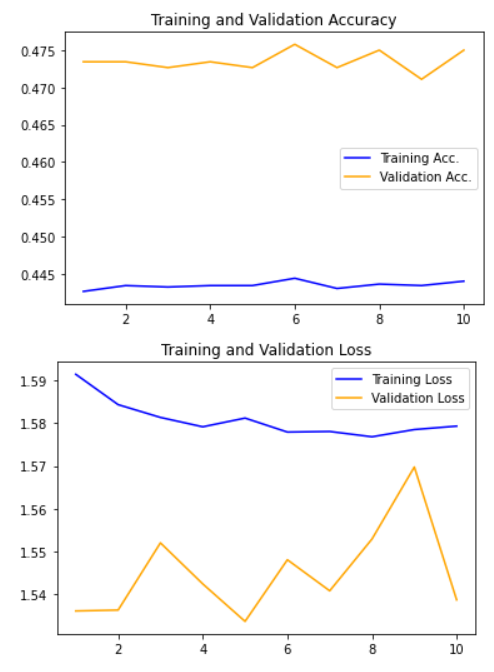
## Model Design 3

The third model was built and trained from scratch, with the layers shown in the figure below. The data was split using pandas sample function, with 80% for training and 20% for validation.



**Fig.9 Scratch Architecture Design**

The model was complied with an Adam optimizer and a categorical crossentropy loss function. The model was trained for 10 epochs, with a batch size of 5 due to memory limitations. Neither training nor validation accuracy increased notably during training. This was disappointing but not unexpected, given that the model was extremely simple and did not utilize transfer learning. No class adjustments were made, the model trained over the natural distribution of the dataset with all labels present.



**Fig.10 Scratch Accuracy and Loss Learning Curves**

Training did not show any notable improvements. The accuracy remained at roughly 0.45 for training accuracy, and 0.47 for validation accuracy over all 10 epochs. The training loss appears to drop slightly, but rising spikes in the validation loss indicate that the loss was in truth worsening, rather than improving. The final loss was 1.54, and the final accuracy was 0.4734. Further statistics are not available

## Final Model

The first design (Model Design 1) was chosen as the final model since it showed a decent learning curve that plateaued towards the end of its training period and reached 95% accuracy on the validation set. Its precision ranged from 94-96% while its recall ranged from 93-96% across target classes. This model produces good results, however, it will still need to be continuously tweaked until a 98-99% accuracy is reached before model deployment. This is due to the fact that it will be a tool used in the healthcare realm. Some images from this final model are shown in Figure 11.

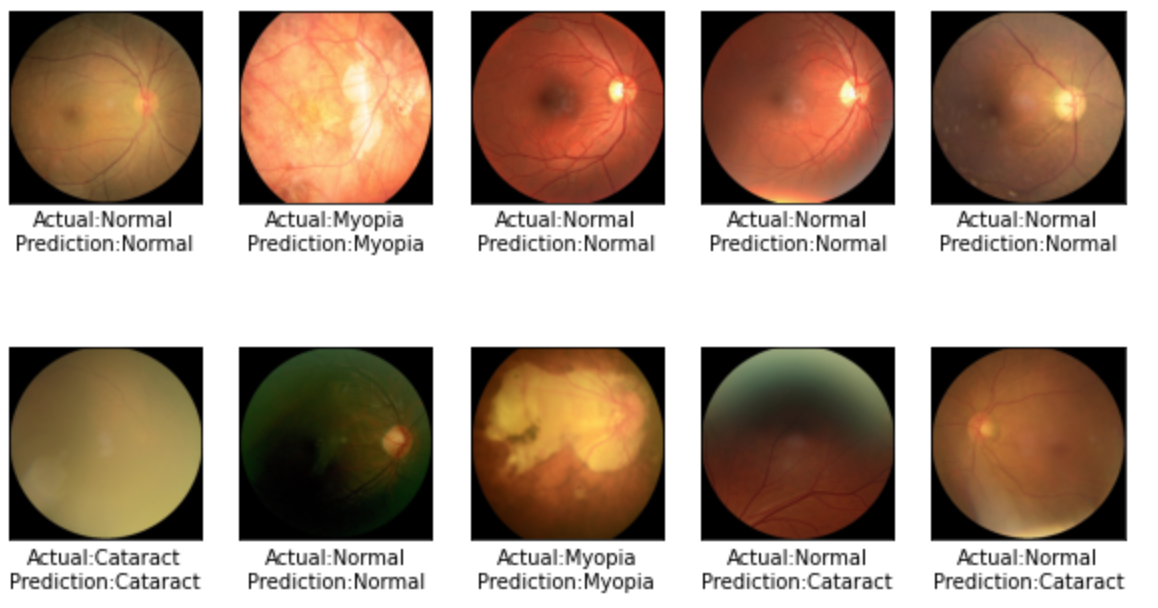


Fig.11 Predicted Images from Final Model

# Project Milestones

The first key milestone in this project was selecting the ocular disease dataset and deciding on the project goal. After this, the team needed to decide how we would use the data to meet the goal. Data pre-processing consisted of loading the images, loading the provided csv file, assigning correct images and labels to the same position in separate lists/arrays, and resizing and standardizing the images (dividing by 255.0 to put pixel values in [0, 1] range). These images then needed to be split into training/validation/test variables in order to properly train and predict using a neural network. The next milestone that was reached from the project plan was building the architecture of the convolutional neural network (CNN) and training it. Each individual in the team created their own CNN in order to determine which approach yielded the best results. There were CNNs that were made from the first layer, as well as others that used transfer learning to build off of a pre-trained model. This was the step that resulted in the most variance as far as design approach. Model architecture as well as target classes were experimented with during this phase. Certain classes that weren’t performing well were removed in order to create a substantial model on those that had the ability to learn given the timeframe in which the model architecture had to be created and trained. Each model was then assessed and metrics were calculated based on reserved test datasets.

The next steps that may be pursued for this project include continuing to tweak the model architecture to allow for better accuracy across the maximum number of ocular disease classifications as possible. This could potentially mean creating an intricate model architecture that could take days to train. Once a model is obtained that results in a reliable 98-99% accuracy range, a web application can be created to allow medical professionals to use the service. Users will be able to access a web application, upload their target images, and receive a disease prediction that is reliable and aids them in making a proper patient diagnosis. Eventually, this model can be developed along with imaging equipment to give on-demand disease prediction seconds after the medical imaging takes place.

# Conclusion

An ocular disease prediction system is one that opthamologists and medical professionals can utilize when determining a patient’s diagnosis based on medical image processing. This initiative focused on using digital image processing to train a neural network to create an ocular disease prediction system. The best model created achieved 95% accuracy, 94-96% precision, and 93-96% recall across target classifications. While these results are good, this model needs to continue to be tweaked to improve these metrics and the reliability of the resulting model. This disease prediction system can then be distributed to medical professionals in order to use in conjunction with imaging tests to give direction and more confidence to the resulting diagnoses.

# References

<https://towardsdatascience.com/ocular-disease-recognition-using-convolutional-neural-networks-c04d63a7a2da>

<https://www.kaggle.com/andrewmvd/ocular-disease-recognition-odir5k>

<https://www.kaggle.com/gpreda/cataract-prediction-using-transfer-learning>

<https://www.tensorflow.org/tutorials/images/cnn>

# 

# Appendix A

## Repository

The GitHub repository for this project is located at the following [location. https://github.com/nlangroodi/ocular\_disease\_recognition.git](https://github.com/nlangroodi/ocular_disease_recognition.git)

The shared Google Drive for this project is located at the following [location. https://drive.google.com/drive/folders/1QAEvBwMNqfGpNA8Km-FyIxKeYXj7bPCr](https://drive.google.com/drive/folders/1QAEvBwMNqfGpNA8Km-FyIxKeYXj7bPCr)

## Code

The code has been compiled and included in subsequent pages.