

School of Information Studies

IST 707: Applied Machine Learning

Final Project

Ocular Disease Recognition

Karthik Ramanathan 999806582

Shaunak Edakhe 439623054

Trishla Jain 593506380

**Acknowledgments**

We would like to wholeheartedly thank Professor Joshua Introne who has been our guide and mentor throughout this project. He was always willing to make our lives easier by being on the front foot in providing us with resources and hardware needed to carry out the required tasks. Taking up a complex project of this sort would not have been possible without his support and motivation.

We would also like to thank our classmates from the IST 707 class, especially Anand Vamsi and Eric Pfeiffer, for their valuable opinions and suggestions throughout the process.

Big thanks to Syracuse University and the 505 on Walnut for providing us the infrastructure to carry out all project-related tasks peacefully.

Finally, lots of gratitude to the resources available on Google like Kaggle and Colaboratory, and Stackoverflow for providing us with a ‘see’ of knowledge.

## **Introduction**

## **Project Overview**

Ocular diseases i.e., diseases concerning the eye, are a problem affecting people all over the world. With advancements in Image Computing and Artificial Intelligence by reading images and processing them using Deep Learning, the idea that an algorithm can recognize an ocular disease by looking at its image and aid the physician in diagnosis is a useful implementation for the medical industry as a whole.

In this project, we have implemented Deep Learning algorithms to read in images present in the dataset and classify said images based on the disease they are likely diagnosed with.

The project was inspired by [this dataset on Kaggle](https://www.kaggle.com/datasets/andrewmvd/ocular-disease-recognition-odir5k).

## **Problem Statement**

Quoting the dataset introduction from Kaggle:  
“Ocular Disease Intelligent Recognition (ODIR) is a structured ophthalmic database of 5,000 patients with age, color fundus photographs from left and right eyes and doctors' diagnostic keywords from doctors.

This dataset is meant to represent ‘real-life’ set of patient information collected by Shanggong Medical Technology Co., Ltd. from different hospitals/medical centers in China. In these institutions, fundus images are captured by various cameras in the market, such as Canon, Zeiss and Kowa, resulting into varied image resolutions.

Annotations were labeled by trained human readers with quality control management.

They classify patient into eight labels including:

Normal (N),

Diabetes (D),

Glaucoma (G),

Cataract (C),

Age related Macular Degeneration (A),

Hypertension (H),

Pathological Myopia (M),

Other diseases/abnormalities (O)”

Our goal for this project was to find the most accurate model or combination of models that would correctly detect the ocular disease present in the image.   
The following tasks were performed:

1. Download and explore the data that consists of 5000 images, preprocessed images, and patient information spreadsheet.
2. Use a trained model to process given data on and check accuracy.
3. Generate predictions on a new set of images.
4. Repeat process using different models and check which model offers the best prediction.

The final model is expected to be able to correctly predict the ocular disease after seeing the image, as frequently as possible.

## **Analysis**

## **Data Exploration**

Our dataset consists of 3 components:

1. The ODIR-5K (Ocular Disease Intelligent Recognition) folder consists of around 8000 images collected from a database of 5000 patients. The images all have different resolutions but mostly range around the 1920\*1080 pixels resolution, meaning they are present in high-definition format.   
     
   

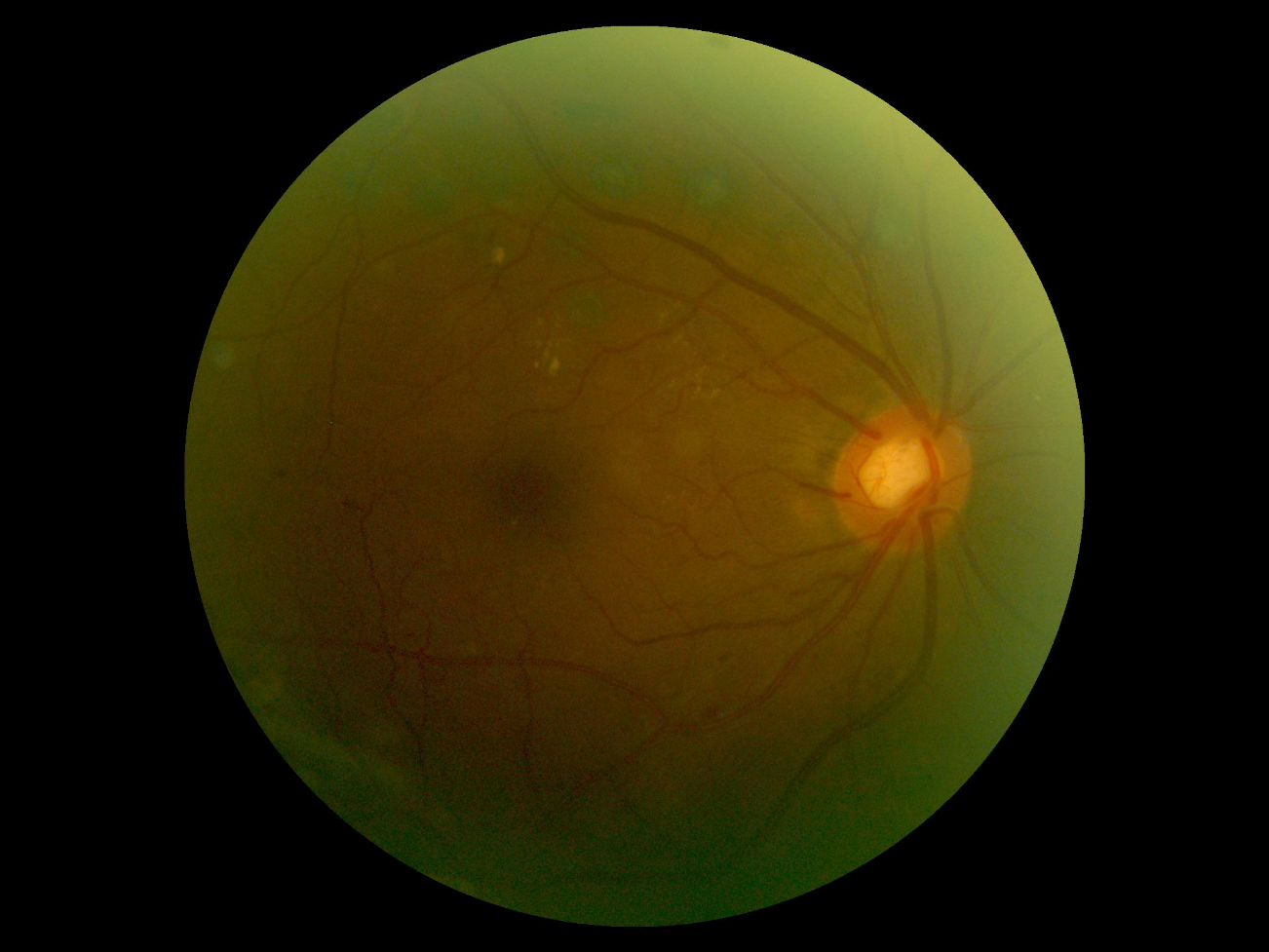
Figure . Sample left fundus  


Figure . Sample right fundus

1. The ‘preprocessed\_images’ folder consists of around 6400 images, all of which have been standardized to a 512\*512 pixels resolution. The small and standardized format makes it easier for the model to process the image data.



Figure . Preprocessed left fundus  
  
  


Figure . Preprocessed right fundus

1. The ‘full\_df.csv’ is spreadsheet that contains patient data consisting of their age, sex, the disease that their eye(s) are detected to have, and notes from the doctors that consist of disease-relevant keywords.  
   The spreadsheet has the following 19 columns:
   * + 1. ID
       2. Patient Age
       3. Patient Sex
       4. Left-Fundus (Contains the name of the left fundus image file)
       5. Right-Fundus (Contains the name of the right fundus image file)
       6. Left-Diagnostic Keywords (Notes from the doctor pertaining to the left eye)
       7. Right-Diagnostic Keywords (Notes from the doctor pertaining to the right eye)
       8. N (Normal)
       9. D (Diabetes)
       10. G (Glaucoma)
       11. C (Cataract)
       12. A (Age related Macular Degeneration)
       13. H (Hypertension)
       14. M (Pathological Myopia)
       15. O (Other diseases/abnormalities)
       16. filepath (location of the eye image in the ODIR-5K directory)
       17. labels (letter denoting the disease that has been diagnosed)
       18. target (disease that has been diagnosed in an array form)
       19. filename (name of the eye image file)

## **Exploratory Visualizations**

The first plot below shows the distribution of patients by sex in our dataset. Understanding the spread of data may help our model find patterns in the diseases for a specific sex.



Figure . Distribution of patients by sex

The graph tells us that the male patients outnumber the female patients by a count of around 500. Calculating percentages for the values gives us a ratio of 53.5% males and 46.5% females, which is a fairly equal distribution.

The second plot shows us the distribution of ocular diseases in our dataset. Understanding how common a certain disease is may help our model find nuances for classification that are only found in the comparatively rarer occurring diseases.

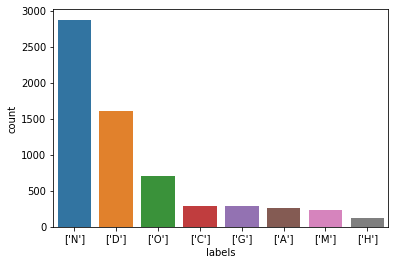


Figure . Distribution of ocular disease

The labels are represented in the following way:  
Normal (N), Diabetes (D), Glaucoma (G), Cataract (C), Age related Macular Degeneration (A), Hypertension (H), Pathological Myopia (M) and Other diseases/abnormalities (O).   
  
As we can see above, a Normal fundus is the most common occurrence in our dataset. Ocular diseases caused due to Diabetes are the second-most common, followed by Other diseases/abnormalities (O). Cataract (C), Age related Macular Degeneration (A), and Glaucoma (G) all have very similar counts in our dataset.

The next plot shows us a distribution of the patients in our dataset by age.

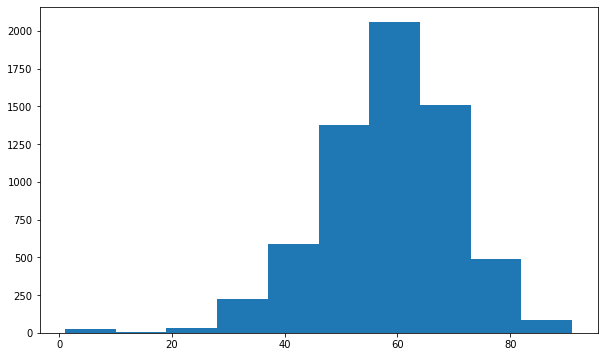


Figure . Distribution of the patients in by age

The above histogram tells that very few people below the age of 30 suffer from ocular diseases. People in the range of 50-75 seem to be the most affected group. This makes sense as natural ageing affects eyesight among others. Also, since these years are generally spent working, stress and other factors such as lack of exercise lead to conditions such as hypertension and diabetes which may have a knock-on effect on the eyes.

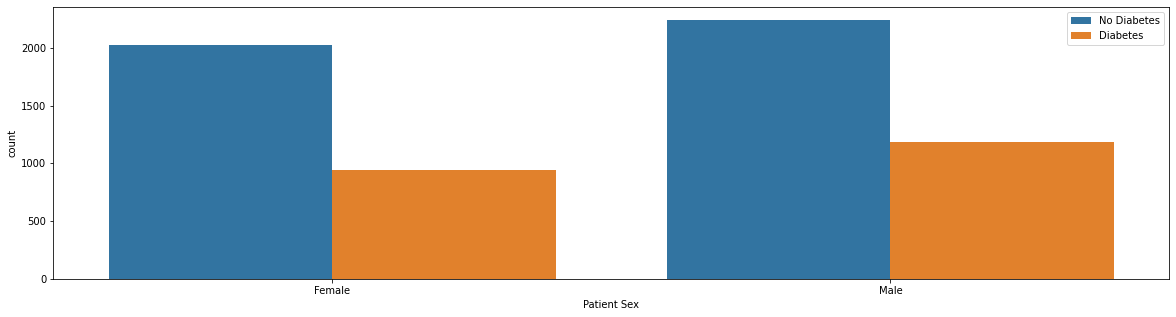
Combining the factors of sex and a particular type of disease (in this case – diabetes), we tried to find if there is any kind of relation present or disparity caused due to sex.   


Figure . Distribution of diabetes grouped by sex

Looking at the graph tells us that there is no marked disparity when grouped by sex. The number of people affected in both sexes is quite proportionate.

## **Algorithms and Techniques**

Utilizing concepts of Deep Learning, we have implemented algorithms such as [Convolutional Neural Network](http://cs231n.github.io/convolutional-networks/)s, [VGG16](https://www.mygreatlearning.com/blog/introduction-to-vgg16/), and [ResNet50](https://keras.io/api/applications/resnet/), which are state-of-the-art algorithms for most image processing tasks, including classification. These algorithms need a large amount of training data as compared to other approaches and are suitable for our application since we have around 6500 images to process and train from.

The following parameters can be tuned to optimize the classifier:

* Neural network architecture
  + Number of layers
  + Layer types ([convolutional](http://cs231n.github.io/convolutional-networks/#conv), [fully-connected](http://cs231n.github.io/convolutional-networks/#fc), or [pooling](http://cs231n.github.io/convolutional-networks/#pool))
  + Layer parameters
* Preprocessing parameters
* Training parameters
  + Training length (number of epochs)
  + Batch size (how many images to look at once during a single training step)
  + Solver type (what algorithm to use for learning)
  + Learning rate (how fast to learn; this can be dynamic)

During training, both the training and the validation sets are loaded into the RAM. After that, random batches are selected to be loaded into the GPU memory for processing. Using a GPU speeds up the training process as the epochs are processed in a parallel manner. The GPU is only utilized in this step, as the predictions generated by the trained algorithm are matched appropriately by the CPU.

# **Methodology**

**Model 1: Simple Sequential Model with multiple layers**

**Pre-Processing**

* The first task was to create separate dataframes for each disease so that the data could be fed easily to the model.
* Based on the diagnostic keywords present for both the right and left eyes, we fetched the file names and moved these particular file names to a dataframe solely for one disease. This dataframe now had filenames for the both the eyes and pertained to one disease, say Glaucoma.
* This process was repeated for all eight diseases and similar dataframes were made.
* Next, we define the X and y matrices, which are further broken down into training, validation, and testing sets.
  + For the X matrix, we read the images and converted them into an array of red, green and blue pixel values
  + For the y matrix, we converted the diseases to specific label values. E.g., Normal is given label 0, Cataract was given 1 etc.
  + We split the X and y matrices into three sub-matrices each: X\_train, X\_val, X\_test, y\_train, y\_val, y\_test.
  + X and y are split into X\_val-y\_val and X\_train-y\_train with 20% random data being in the validation set and rest of the 80% data being in the training set.
  + X\_val and y\_val are further split into X\_test-y\_test and X\_val-y\_val. The validation and test sets in this scenario are split randomly and equally. The validation set is used to track test error during model training, while the test sets are used to assess the model’s error on never-seen-before data.

**Implementing the model**

* We initialized the Sequential model and added the first 2D Convolution layer ([Conv2D](https://keras.io/api/layers/convolution_layers/convolution2d/)) where we defined the input shape as (224, 224, 3), filters (the number of output filters in the convolution) as 64, a kernel size (height and width of the 2D convolution window) of (3,3), padding of ‘same’ type, and ‘[ReLU](https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/)’ (Rectified Linear Unit) as the activation function.
* Next, we added an identical layer as the one above, followed by a [MaxPool2D](https://keras.io/api/layers/pooling_layers/max_pooling2d/) layer having pool size of (2,2) and strides of (2,2). This particular MaxPool2D layer was used multiple times in this model.
* We repeated the addition of 2D Convolution layers, adding 2 layers with specifications identical to that of the ones above, with the exception of the number of filters being raised from 64 to 128. This was followed by the addition of a MaxPool2D layer with identical specifications as from above.
* Three 2D Convolution layers were added again with identical specifications but number of filters being raised from 128 to 256, followed by a MaxPool2D layer.
* Three more 2D Convolution layers were added with 512 filters, followed by a MaxPool2D layer. This combination was inserted in the model twice.
* Finally, we added two Dense layers, each with a batch of 4096 units and ‘[ReLU](https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/)’ (Rectified Linear Unit) as the activation function.
* We added an additional Dense layer to the model with a batch of 8 units and ‘[Softmax](https://deepai.org/machine-learning-glossary-and-terms/softmax-layer)’ activation function.
* We then optimized the model using the [Adam algorithm](https://keras.io/api/optimizers/adam/) with a learning rate of 0.001.
* X\_train and y\_train were the dataframes used to train the model with a batch size of 128 for 20 epochs for the first iteration. For the second iteration, we had a batch size of 64 for 50 epochs. The validation was performed against the X\_val and y\_val dataframes we obtained above.

**Performance Metrics and Inferences**

**Iteration 1:**

* Model Accuracy: **31.72%**
* Model Loss: **1.6980**
* Running a user-defined function, we were able to plot the trends in Accuracy and Loss over epochs. For Accuracy, we see curves that have plateaued in a particular range with the exception of one epoch, where there was a heavy drop in accuracy for both the training and validation dataframes,

For Loss, the Loss values for the Training set have dropped hard in the very first few epochs. For the Validation set, the values are present in a straight line, set at a single magnitude with no change whatsoever.

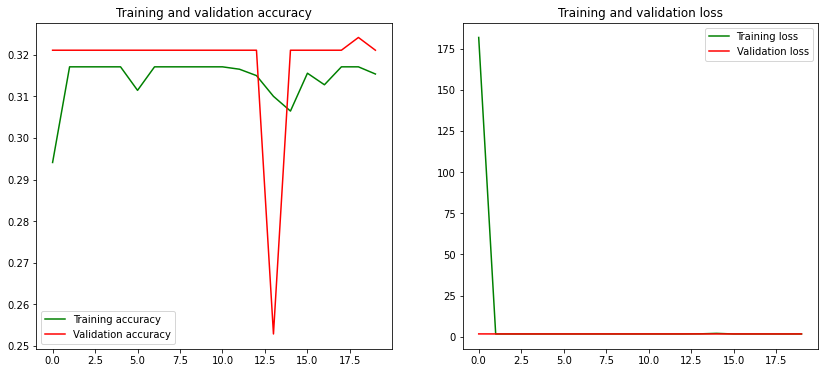
The trendlines lead us to conclude that the model almost overfit.  
 ****

Figure 9. Model 1 - Accuracy and Loss graphs - Iteration 1

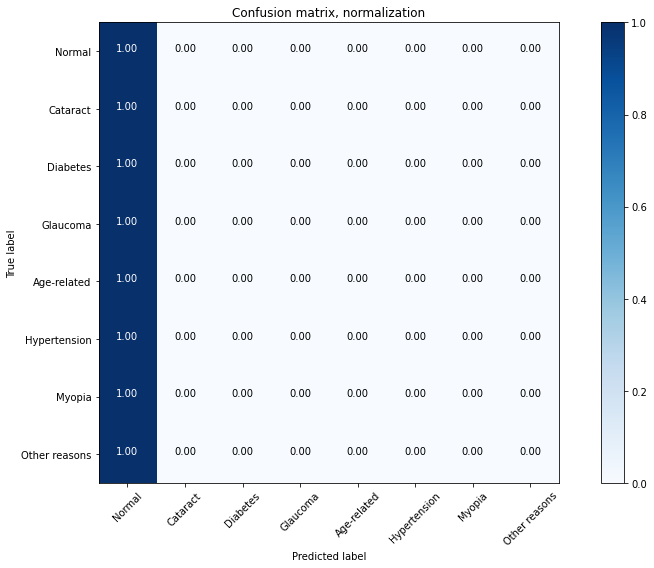
* The confusion matrix is reinforcing what we see with our accuracy metrics - the model is pretty bad at detecting any of the diseases and is unable to even identify Normal fundus images despite the dataset having a vast majority of Normal Fundus images.  
  

Figure 10. Confusion matrix for Model 1 – Iteration 1

* Classification Report:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **Normal** | 0.32 | 1.00 | 0.48 | 414 |
| **Cataract** | 0.00 | 0.00 | 0.00 | 62 |
| **Diabetes** | 0.00 | 0.00 | 0.00 | 327 |
| **Glaucoma** | 0.00 | 0.00 | 0.00 | 64 |
| **Age-related** | 0.00 | 0.00 | 0.00 | 56 |
| **Hypertension** | 0.00 | 0.00 | 0.00 | 45 |
| **Myopia** | 0.00 | 0.00 | 0.00 | 37 |
| **Other reasons** | 0.00 | 0.00 | 0.00 | 300 |
|  |  |  |  |  |
| **accuracy** |  |  | 0.32 | 1305 |
| **macro avg** | 0.04 | 0.12 | 0.06 | 1305 |
| **weighted avg** | 0.10 | 0.32 | 0.15 | 1305 |

* Precision and recall was simply too bad for all the images in this model.

**Iteration 2:**

* Model Accuracy: **31.72%**
* Loss: **1.6963**
* Similar to the results obtained in the first iteration, for Accuracy, we see curves that have plateaued in a particular range with the exception of one epoch, where there was a heavy drop in accuracy for both the training and validation dataframes.

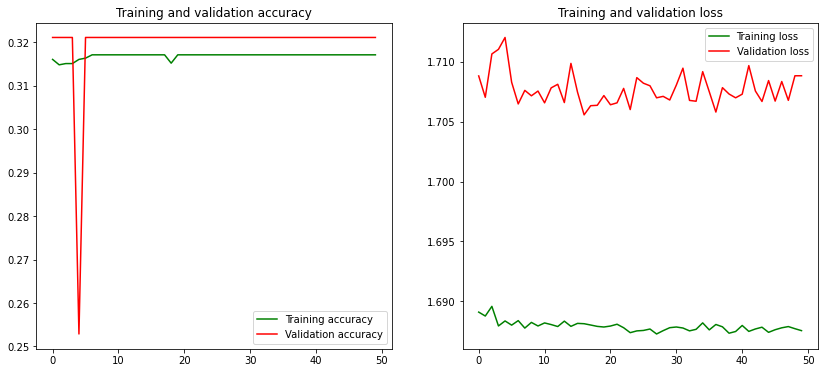
For Loss, the values for both the Training and Validation sets seem to have plateaued in a particular range. There is no visible upward or downward trend in either.  
 ****

Figure 11. Model 1 - Accuracy and Loss graphs – Iteration 2

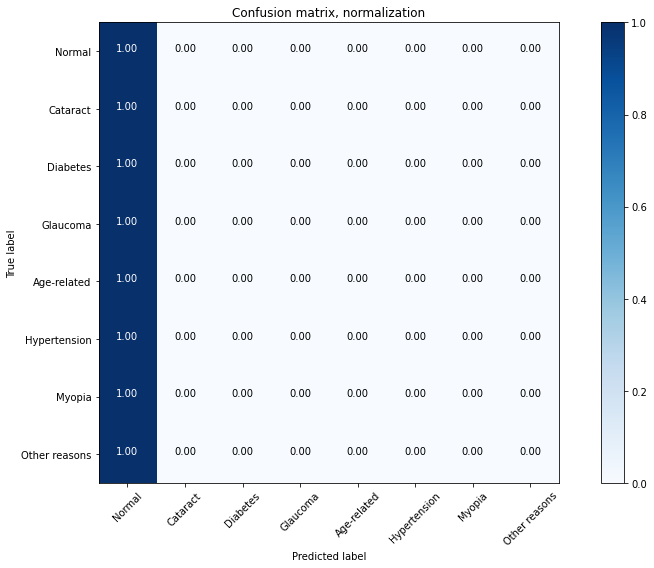
* The confusion matrix adds to the fact that there was no improvement at all in the model.

Figure 12. Confusion matrix for Model 1 – Iteration 2

* Classification Report:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **Normal** | 0.32 | 1.00 | 0.48 | 414 |
| **Cataract** | 0.00 | 0.00 | 0.00 | 62 |
| **Diabetes** | 0.00 | 0.00 | 0.00 | 327 |
| **Glaucoma** | 0.00 | 0.00 | 0.00 | 64 |
| **Age-related** | 0.00 | 0.00 | 0.00 | 56 |
| **Hypertension** | 0.00 | 0.00 | 0.00 | 45 |
| **Myopia** | 0.00 | 0.00 | 0.00 | 37 |
| **Other reasons** | 0.00 | 0.00 | 0.00 | 300 |
|  |  |  |  |  |
| **accuracy** |  |  | 0.32 | 1305 |
| **macro avg** | 0.04 | 0.12 | 0.06 | 1305 |
| **weighted avg** | 0.10 | 0.32 | 0.15 | 1305 |

* We can see that both the precision and recall have not changed for any of the fundus images.

Attempting to make predictions for a random set of 10 images using this model, the model was able to correctly predict and classify 4 of 10 images. A clear downgrade.

**Model 2: Resnet50**

**Pre-Processing**

* All pre-processing steps for this model are identical to that of the previous models, starting from the task where we create separate dataframes for each disease, to splitting the X and y matrices into training, validation, and testing sets.

**Implementing the model**

* We initialized the Sequential model and added a [ResNet50](https://keras.io/api/applications/resnet/) layer with pooling set to ‘max’ and weights set to pre-training on ImageNet.
* Next, we added a Dense layer with a batch of 8 units and ‘[Softmax](https://deepai.org/machine-learning-glossary-and-terms/softmax-layer)’ activation function.
* We then optimized the model using the [LazyAdam](https://www.tensorflow.org/addons/api_docs/python/tfa/optimizers/LazyAdam) algorithm.
* X\_train and y\_train were the dataframes used to train the model with a batch size of 128 for 50 epochs for the first iteration. For the second iteration, we had a batch size of 64 for 100 epochs. The validation was performed against the X\_val and y\_val dataframes we obtained above.

**Performance Metrics and Inferences**

**Iteration 1:**

* Model Accuracy: **73.10%**
* Model Loss: **0.9230**
* Running a user-defined function, we were able to plot the trends in Accuracy and Loss over epochs. For Accuracy, we see that the curves for both Training and Validation sets rise together steeply and later on plateau towards a particular narrow range of values.

For Loss, the values for the Training set are almost flat from the get-go. For the Validation set, there is a steep drop in the values at the very beginning, soon after which they start to flatten out.

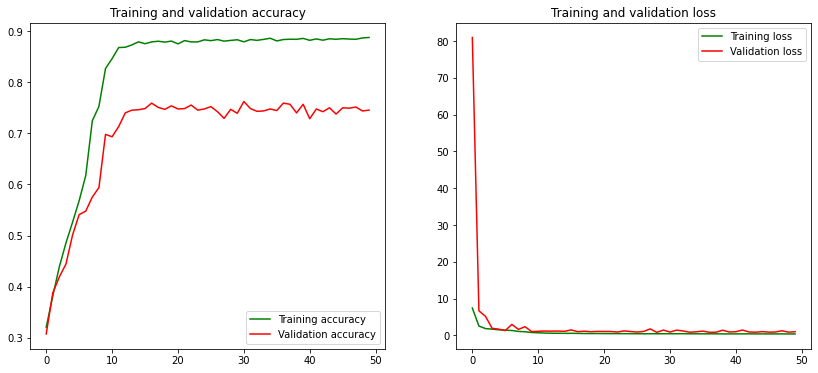
The trendlines lead us to conclude that the model is not overfitting and we get a good reduction in our validation loss.  
 ****

Figure 13. Model 2 - Accuracy and Loss graphs - Iteration 1

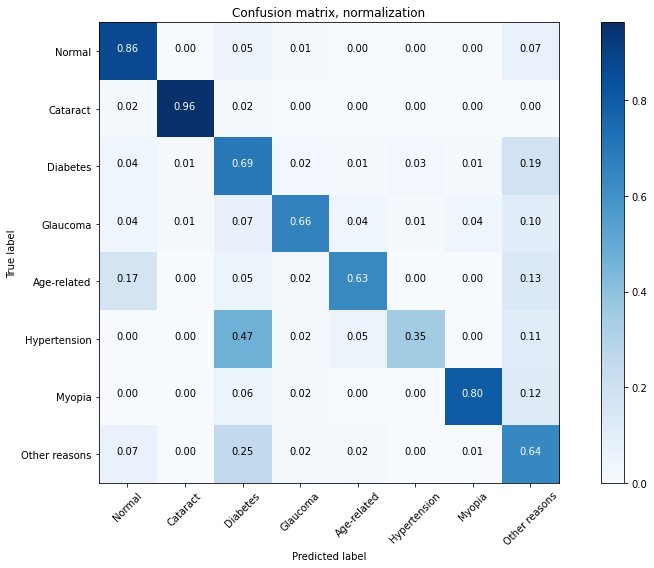
* The confusion matrix tells us that for Normal, Myopia, and Cataract fundus images, the algorithm did a good job identification. Diabetes, Glaucoma, Age related diseases, and diseases caused due to Other reasons got an average performance. Hypertension identification had the worst performance and we can also see that it is correlated to Diabetes. This may have caused our model to get confused in trying to distinguish between the two. These performance results make sense since we had a lot of images fed to the model for Normal, Myopia, and Cataract, whereas Hypertension had the least amount of data (382 images). The poor performance could be attributed to a lack data for the model to learn from.   
  

Figure 14. Confusion matrix for Model 2 – Iteration 1

* Classification Report:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **Normal** | 0.88 | 0.86 | 0.87 | 407 |
| **Cataract** | 0.89 | 0.96 | 0.93 | 53 |
| **Diabetes** | 0.64 | 0.69 | 0.67 | 338 |
| **Glaucoma** | 0.69 | 0.66 | 0.67 | 67 |
| **Age-related** | 0.73 | 0.63 | 0.68 | 60 |
| **Hypertension** | 0.63 | 0.35 | 0.45 | 55 |
| **Myopia** | 0.83 | 0.80 | 0.82 | 50 |
| **Other reasons** | 0.60 | 0.64 | 0.62 | 275 |
|  |  |  |  |  |
| **accuracy** |  |  | 0.73 | 1305 |
| **macro avg** | 0.74 | 0.70 | 0.71 | 1305 |
| **weighted avg** | 0.73 | 0.73 | 0.73 | 1305 |

* Normal, Cataract, Age-related, and Myopia have a good recall and precision as compared to the rest of the diseases, which are having an average recall and precision. We can also see that the recall was particularly bad for Hypertension.

**Iteration 2:**

* Model Accuracy: **58.70%**
* Loss: **1.5931**
* In contrast to the results obtained in the first iteration, for Accuracy, we see a curve that has plateaued in a particular range with the exception of a few epochs, where there was a heavy drop in accuracy for the training set. For the validation set, the accuracy is all over the place and varies wildly throughout the epochs.

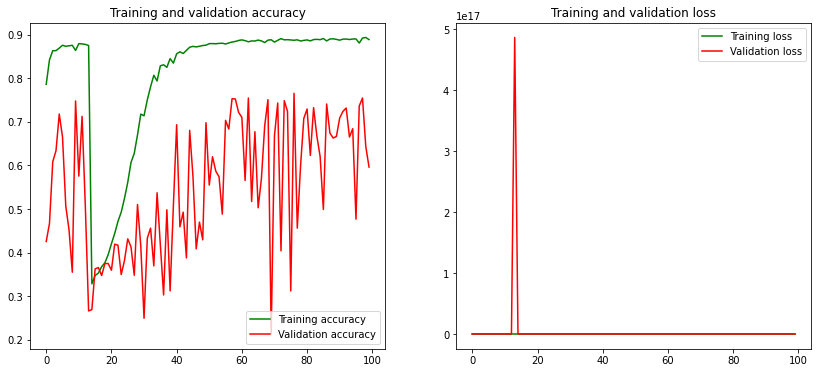
For Loss, the values for both the Training and Validation are completely flat at 0, except for a few epochs for the validation set, where there is a sudden spike.   
Both the accuracy and loss graphs show that we are overfitting and that increasing the batch size may have been a bad idea.  
 ****

Figure 15. Model 2 - Accuracy and Loss graphs – Iteration 2

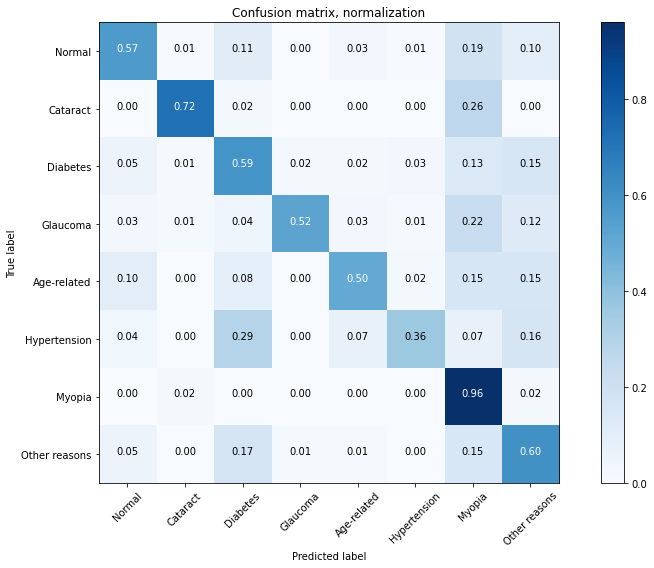
* The confusion matrix supports our observation that the model performance has decreased for all our classes.  
  

Figure 16. Confusion matrix for Model 2 – Iteration 2

* Classification Report:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **Normal** | 0.84 | 0.57 | 0.68 | 407 |
| **Cataract** | 0.83 | 0.72 | 0.77 | 53 |
| **Diabetes** | 0.63 | 0.59 | 0.61 | 338 |
| **Glaucoma** | 0.78 | 0.52 | 0.62 | 67 |
| **Age-related** | 0.54 | 0.50 | 0.52 | 60 |
| **Hypertension** | 0.59 | 0.36 | 0.45 | 55 |
| **Myopia** | 0.19 | 0.96 | 0.32 | 50 |
| **Other reasons** | 0.58 | 0.60 | 0.59 | 275 |
|  |  |  |  |  |
| **accuracy** |  |  | 0.59 | 1305 |
| **macro avg** | 0.62 | 0.60 | 0.57 | 1305 |
| **weighted avg** | 0.68 | 0.59 | 0.61 | 1305 |

* We can see that both precision and recall have decreased for all the fundus images.
* Attempting to make predictions for a random set of 10 images using this model, the model was able to correctly predict and classify only 4 of 10 images.

**Model 3: VGG16 with Normal, Glaucoma, Diabetes and Cataract disease types only**

**Pre-Processing**

* Similar to the above models,the first task was to create separate dataframes for each disease so that this data can be fed easily to the model.
* Based on the diagnostic keywords present for both the right and left eyes, we fetched the file names and moved these particular file names to a dataframe solely for one disease. This dataframe now had filenames for the both the eyes and pertained to one disease, say Glaucoma.
* This process was repeated for all four diseases and similar dataframes were made.
* Next, we define the X and y matrices, which are further broken down into training, validation, and testing sets.
  + For the X matrix, we read the images and converted them into an array of red, green and blue pixel values
  + For the y matrix, we converted the diseases to specific label values. E.g., Normal is given label 0, Cataract was given 1 etc.
  + We split the X and y matrices into three sub-matrices each: X\_train, X\_val, X\_test, y\_train, y\_val, y\_test.
  + X and y are split into X\_val-y\_val and X\_train-y\_train with 20% random data being in the validation set and rest of the 80% data being in the training set.
  + X\_val and y\_val are further split into X\_test-y\_test and X\_val-y\_val. The validation and test sets in this scenario are split randomly and equally. The validation set is used to track test error during model training, while the test sets are used to assess the model’s error on never-seen-before data.

**Implementing the model**

* [VGG16](https://www.mygreatlearning.com/blog/introduction-to-vgg16/) is a pre-trained model that is part of the Tensorflow Keras library of packages.
* We imported the model, set the weights to pre-trained weights from ImageNet, and defined the input shape as (224, 224, 3), which is essentially the size of the arrays obtained on processing the images.
* We added two Dense layers to the model, each with a batch of 128 units and ‘[ReLU](https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/)’ (Rectified Linear Unit) as the activation function.
* The Output layer consisted of batches of 4 units (one for each disease) and ‘[Softmax](https://deepai.org/machine-learning-glossary-and-terms/softmax-layer)’ activation function.
* We then optimized the model using the [Adam algorithm](https://keras.io/api/optimizers/adam/) with a learning rate of 0.00001.
* X\_train and y\_train were the dataframes used to train the model with a batch size of 128 for 20 epochs. The validation was performed against the X\_val and y\_val dataframes.

**Performance Metrics and Inferences**

* Model Accuracy: **89.01%**
* Loss: **0.3611**

**Chart, histogram

Description automatically generated** Figure 17. Model 3 - Accuracy and Loss graphs

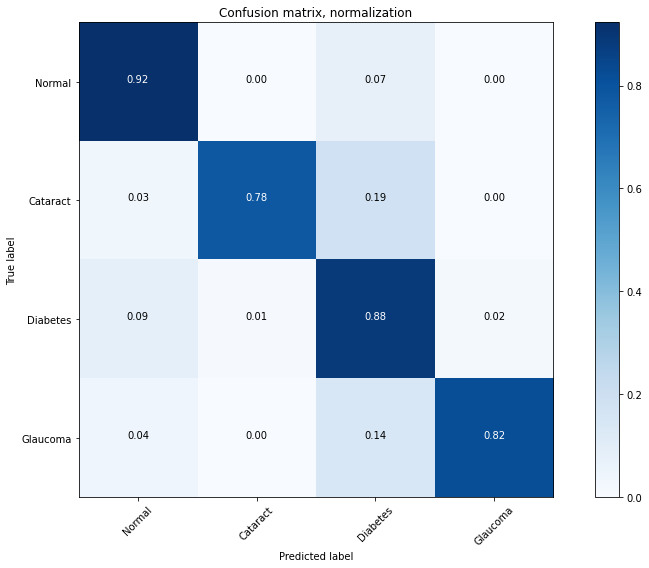
* The confusion matrix tells us that the algorithm did well predicting fundi that were either Normal, Glaucoma or Diabetes, but gave a comparatively lower accuracy when it came to Cataract fundi.  
   ****

Figure 18. Confusion matrix for Model 3

* Classification Report:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **Normal** | 0.91 | 0.92 | 0.92 | 440 |
| **Cataract** | 0.90 | 0.78 | 0.84 | 59 |
| **Diabetes** | 0.86 | 0.88 | 0.87 | 385 |
| **Glaucoma** | 0.88 | 0.82 | 0.85 | 71 |
|  |  |  |  |  |
| **accuracy** |  |  | 0.89 | 955 |
| **macro avg** | 0.89 | 0.85 | 0.87 | 955 |
| **weighted avg** | 0.89 | 0.89 | 0.89 | 955 |

* The recall tells us how well the model performed in detecting positive cases of each fundus out of the total number of truly positive cases. Low recall stands for risk of not detecting all truly positive cases which is fatal when it comes to health data. As we can see above, other than Cataract, every other disease has a good recall. This may be due to the choice of the loss function used, which is 'cross-entropy' in our case. It simply penalises the difference between predictions and ground truth positives but it doesn't take the false negatives into considerations. Hence, it is possible that a high number of false negatives for glaucoma impacted the recall which could be due to wrong labelling of certain images.
* The precision of the model tells us what proportion of the predicted positive cases are actually positive. Again this is very important when it comes to health data, since we wouldn’t want to misinform a patient about them having contracted a disease, as this can cause emotional turmoil to the patient, incur unwanted expense, and treatment to an organ which isn't damaged (thereby damaging a good organ), resulting in legal action against the doctor/medical institute. The precision for all of the diseases falls very close to each other - almost etching towards 0.90, helping us form a claim that we have good precision for this model.
* Trying to make a prediction for a newly uploaded high-resolution image using this model, the model was able to correctly report the image as belonging to the Diabetes type.

**Model 4: VGG16 for all 8 disease types**

**Pre-Processing**

* Similar to the previous model, the first task was to create separate dataframes for each disease so that the data could be fed easily to the model.
* Based on the diagnostic keywords present for both the right and left eyes, we fetched the file names and moved these particular file names to a dataframe solely for one disease. This dataframe now had filenames for the both the eyes and pertained to one disease, say Glaucoma.
* This process was repeated for all eight diseases and similar dataframes were made.
* Next, we define the X and y matrices, which are further broken down into training, validation, and testing sets.
  + For the X matrix, we read the images and converted them into an array of red, green and blue pixel values
  + For the y matrix, we converted the diseases to specific label values. E.g., Normal is given label 0, Cataract was given 1 etc.
  + We split the X and y matrices into three sub-matrices each: X\_train, X\_val, X\_test, y\_train, y\_val, y\_test.
  + X and y are split into X\_val-y\_val and X\_train-y\_train with 20% random data being in the validation set and rest of the 80% data being in the training set.
  + X\_val and y\_val are further split into X\_test-y\_test and X\_val-y\_val. The validation and test sets in this scenario are split randomly and equally. The validation set is used to track test error during model training, while the test sets are used to assess the model’s error on never-seen-before data.

**Implementing the model**

* We imported the [VGG16](https://www.mygreatlearning.com/blog/introduction-to-vgg16/) pre-trained model, set the weights to pre-trained weights from ImageNet, and defined the input shape as (224, 224, 3), which is essentially the size of the arrays obtained on processing the images.
* We added two Dense layers to the model, each with a batch of 128 units and ‘[ReLU](https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/)’ (Rectified Linear Unit) as the activation function.
* The Output layer consisted of batches of 8 units, instead of 4 from the previous one, and ‘[Softmax](https://deepai.org/machine-learning-glossary-and-terms/softmax-layer)’ activation function.
* We then optimized the model using the [Adam algorithm](https://keras.io/api/optimizers/adam/) with a learning rate of 0.00001.
* X\_train and y\_train were the dataframes used to train the model with a batch size of 128 for 20 epochs for the first iteration.

**Performance Metrics and Inferences**

**Iteration 1:**

* Model Accuracy: **65.13%**
* Loss: **1.0082**
* Running a user-defined function, we were able to plot the trends in Accuracy and Loss over epochs. For Accuracy, we see a steadily-rising curve as the number of epochs increases for both the training and validation dataframes.

For Loss, we see a similar trend where the Loss values for both the Training and Validation dataframes are falling over increasing epochs.

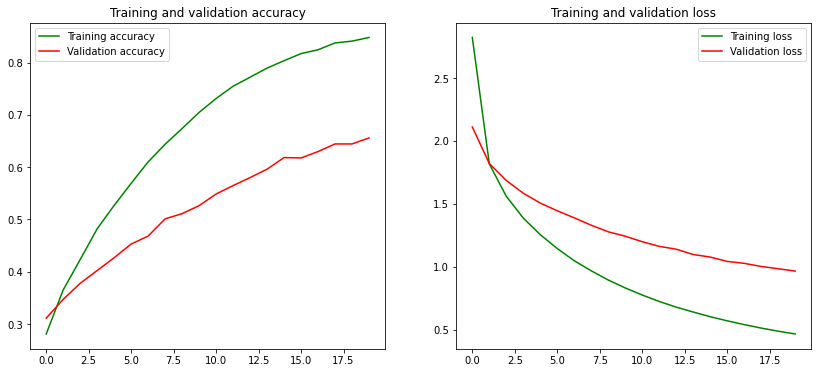
The trendlines lead us to conclude that the model is not overfitting.  
 ****

Figure 19. Model 4 - Accuracy and Loss graphs – Iteration 1

* The confusion matrix tells us that the algorithm did well predicting fundi that were Normal or were suffering from Myopia and Cataract. Diabetes, Glaucoma, Age-related diseases and diseases falling under Other Reasons got an average performance. Hypertension detection had the worst performance. These performance results make sense since we had a lot of images fed to the model for Normal, Myopia, and Cataract, whereas Hypertension had the least amount of data (382 images). The poor performance could be attributed to a lack data for the model to learn from.  
  

Figure 20. Confusion matrix for Model 4 – Iteration 1

* Classification Report:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **Normal** | 0.78 | 0.81 | 0.79 | 393 |
| **Cataract** | 0.91 | 0.79 | 0.85 | 63 |
| **Diabetes** | 0.58 | 0.61 | 0.59 | 344 |
| **Glaucoma** | 0.57 | 0.48 | 0.52 | 56 |
| **Age-related** | 0.66 | 0.45 | 0.54 | 55 |
| **Hypertension** | 0.32 | 0.17 | 0.23 | 46 |
| **Myopia** | 0.78 | 0.77 | 0.77 | 47 |
| **Other reasons** | 0.55 | 0.59 | 0.57 | 301 |
|  |  |  |  |  |
| **accuracy** |  |  | 0.65 | 1305 |
| **macro avg** | 0.64 | 0.58 | 0.61 | 1305 |
| **weighted avg** | 0.65 | 0.65 | 0.65 | 1305 |

* Normal, Cataract, Age-related and Myopia have a good recall and precision as compared to the rest of the diseased fundus which are having an average recall and precision. The poor performance while detecting other diseases seems to be inherited due to the imbalance in our dataset.

**Iteration 2:**

For the second iteration, we had a batch size of 64 for 50 epochs. The validation was performed against the X\_val and y\_val dataframes we obtained above.

* Model Accuracy: **74.41%**
* Loss: **0.8624**
* Looking at the trends in Accuracy and Loss over epochs, for Accuracy, we see a curve that starts at an already high accuracy value and plateaus very quickly as the number of epochs increases for both the training and validation dataframes. A small observation would be that for the validation dataframe, the rise is slightly steeper than that for the training dataframe.

For Loss, we see a similar trend where the Loss values for both the Training and Validation dataframes are falling at the beginning followed by a plateau over increasing epochs.

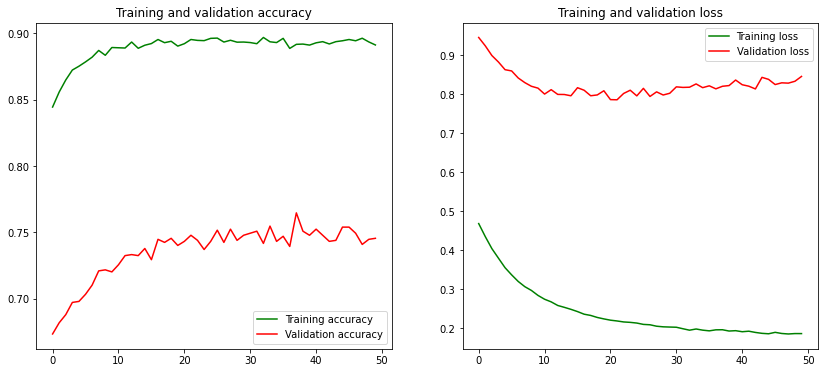
Since our loss and accuracy for both training and validation sets seem to be going in the same direction, we can conclude that the model isn’t overfitting.  
 ****

Figure 21. Model 4 - Accuracy and Loss graphs – Iteration 2

* The confusion matrix has quite similar results as that from the previous iteration, where the nature of the accuracies has remained same (Normal performing well, Hypertension performing poorly) but we notably we have a rise in accuracy for all diseases. This improvement in accuracies, though not great, is quite notable.

Figure 22. Confusion matrix for Model 4 – Iteration 2

* Classification Report:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **Normal** | 0.89 | 0.91 | 0.90 | 393 |
| **Cataract** | 0.95 | 0.89 | 0.92 | 63 |
| **Diabetes** | 0.69 | 0.67 | 0.68 | 344 |
| **Glaucoma** | 0.62 | 0.55 | 0.58 | 56 |
| **Age-related** | 0.66 | 0.64 | 0.65 | 55 |
| **Hypertension** | 0.49 | 0.41 | 0.45 | 46 |
| **Myopia** | 0.77 | 0.79 | 0.78 | 47 |
| **Other reasons** | 0.65 | 0.68 | 0.66 | 301 |
|  |  |  |  |  |
| **accuracy** |  |  | 0.74 | 1305 |
| **macro avg** | 0.71 | 0.69 | 0.70 | 1305 |
| **weighted avg** | 0.74 | 0.74 | 0.74 | 1305 |

* We can see that both the precision and recall have increased for all the images.
* Attempting to make predictions for a random set of 10 images using this model, the model was able to correctly predict and classify 8 of 10 images.

**Model 5: VGG16 with additional layers for all 8 disease types**

**Pre-Processing**

* All pre-processing steps for this model are identical to that of the previous model, starting from the task where we create separate dataframes for each disease, to splitting the X and y matrices into training, validation, and testing sets.

**Implementing the model**

* We imported the [VGG16](https://www.mygreatlearning.com/blog/introduction-to-vgg16/) pre-trained model, set the weights to pre-trained weights from ImageNet, and defined the input shape as (224, 224, 3), which is essentially the size of the arrays obtained on processing the images.
* We added two Dense layers to the model, each with a batch of 64 units and ‘[ReLU](https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/)’ (Rectified Linear Unit) as the activation function, instead of the 128 from the previous Model 2.
* A [Dropout](https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout) layer with a rate of 0.5 was also added. The rate refers to the fraction of the input units to drop.
* The Output layer consisted of batches of 8 units and ‘[Softmax](https://deepai.org/machine-learning-glossary-and-terms/softmax-layer)’ activation function.
* We then optimized the model using the [Adam algorithm](https://keras.io/api/optimizers/adam/) with a learning rate of 0.00001.
* X\_train and y\_train were the dataframes used to train the model with a batch size of 64 for 20 epochs for the first iteration.

**Performance Metrics and Inferences**

**Iteration 1:**

* Model Accuracy: **73.49%**
* Model Loss: **0.8413**
* Running a user-defined function, we were able to plot the trends in Accuracy and Loss over epochs. For Accuracy, we see a steadily-rising curve as the number of epochs increases for both the training and validation dataframes, where the trendlines for the two dataframes overlap quite often.

For Loss, we see a similar trend where the Loss values for both the Training and Validation dataframes are falling over increasing epochs, again overlapping at a few points.

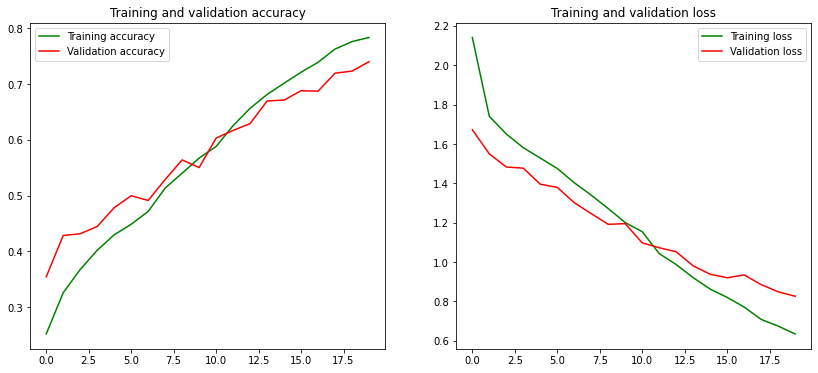
The trendlines lead us to conclude that the model is not overfitting.  
 ****

Figure 23. Model 5 - Accuracy and Loss graphs - Iteration 1

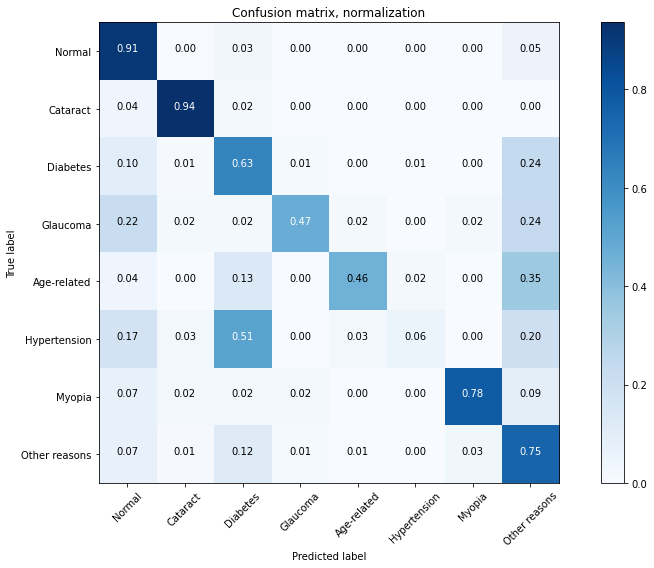
* Following the same pattern as from the previous model, the confusion matrix tells us that the algorithm did well predicting fundi that were Normal or were suffering from Myopia, Cataract, and diseases falling under Other Reasons. Diabetes, Glaucoma, and Age-related diseases got an average performance. Hypertension detection had the worst performance. These performance results make sense since we had a lot of images fed to the model for Normal, Myopia, and Cataract, whereas Hypertension had the least amount of data (382 images). The poor performance could be attributed to a lack data for the model to learn from.   
  

Figure 24. Confusion matrix for Model 5 – Iteration 1

* Classification Report:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **Normal** | 0.82 | 0.91 | 0.86 | 422 |
| **Cataract** | 0.80 | 0.94 | 0.86 | 47 |
| **Diabetes** | 0.76 | 0.63 | 0.69 | 363 |
| **Glaucoma** | 0.72 | 0.47 | 0.57 | 55 |
| **Age-related** | 0.84 | 0.46 | 0.59 | 46 |
| **Hypertension** | 0.40 | 0.06 | 0.10 | 35 |
| **Myopia** | 0.76 | 0.78 | 0.77 | 45 |
| **Other reasons** | 0.59 | 0.75 | 0.66 | 292 |
|  |  |  |  |  |
| **accuracy** |  |  | 0.73 | 1305 |
| **macro avg** | 0.71 | 0.62 | 0.64 | 1305 |
| **weighted avg** | 0.73 | 0.73 | 0.72 | 1305 |

* Normal, Cataract, Age-related, Diabetes, and Myopia have a good recall and precision as compared to the rest of the diseased fundus which are having an average recall and precision. The poor performance while detecting other diseases seems to be inherited due to the imbalance in our dataset.

**Iteration 2:**

For the second iteration, we had a batch size of 64 for 50 epochs. The validation was performed against the X\_val and y\_val dataframes we obtained above.

* Model Accuracy: **76.40%**
* Loss: **0.8286**
* Looking at the trends in Accuracy and Loss over epochs, for Accuracy, we see a curve that starts at an already high accuracy value, rises for a bit, and then starts plateauing as the number of epochs increases for both the training and validation dataframes. A small observation would be that for the validation dataframe, the rise is slightly less steep than that for the training dataframe.

For Loss, we see a trend where the values for the Training set are constantly falling whereas the values for the Validation set seem to fluctuate over a particular range of values over increasing epochs.

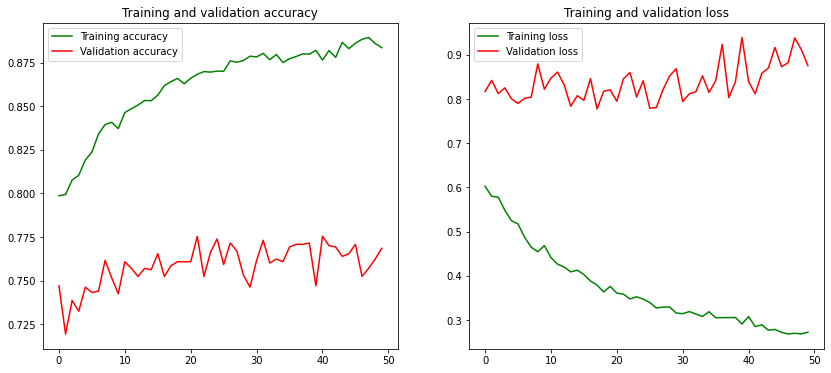
Since our loss and accuracy for both training and validation sets seem to be going roughly in the same direction, we can conclude that the model isn’t overfitting.  
 ****

Figure 25. Model 5 - Accuracy and Loss graphs – Iteration 2

* Following the same pattern as the second iteration from the previous model, the confusion matrix has quite similar results as that from the previous iteration, where the nature of the accuracies has remained same (Normal performing well, Hypertension performing poorly) but we notably we have a rise in accuracy for all diseases. This improvement in accuracies, though not great, is quite notable. The accuracy for Hypertension has a significant jump from 0.06 to 0.31.

Figure 26. Confusion matrix for Model 5 – Iteration 2

* Classification Report:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **Normal** | 0.89 | 0.90 | 0.89 | 422 |
| **Cataract** | 0.82 | 0.96 | 0.88 | 47 |
| **Diabetes** | 0.72 | 0.71 | 0.71 | 363 |
| **Glaucoma** | 0.74 | 0.71 | 0.72 | 55 |
| **Age-related** | 0.78 | 0.61 | 0.68 | 46 |
| **Hypertension** | 0.41 | 0.31 | 0.35 | 35 |
| **Myopia** | 0.76 | 0.76 | 0.76 | 45 |
| **Other reasons** | 0.67 | 0.70 | 0.68 | 292 |
|  |  |  |  |  |
| **accuracy** |  |  | 0.76 | 1305 |
| **macro avg** | 0.72 | 0.71 | 0.71 | 1305 |
| **weighted avg** | 0.76 | 0.76 | 0.76 | 1305 |

* We can see that both the precision and recall have increased for all the fundus images.
* Attempting to make predictions for a random set of 10 images using this model, the model was able to correctly predict and classify 9 of 10 images. This is an improvement over the previous model.

**Model 6: VGG16 with additional layers for all 8 disease types**

**Pre-Processing**

* All pre-processing steps for this model are identical to that of the previous model, starting from the task where we create separate dataframes for each disease, to splitting the X and y matrices into training, validation, and testing sets.

**Implementing the model**

* We imported the [VGG16](https://www.mygreatlearning.com/blog/introduction-to-vgg16/) pre-trained model, set the weights to pre-trained weights from ImageNet, and defined the input shape as (224, 224, 3), which is essentially the size of the arrays obtained on processing the images.
* We added two Dense layers to the model, each with a batch of 64 units and ‘[ReLU](https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/)’ (Rectified Linear Unit) as the activation function, instead of the 128 from the previous Model 2.
* We added an additional Dense layer to the model with a batch of 8, one for each disease.
* A [Dropout](https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout) layer with a rate of 0.5 was also added. The rate refers to the fraction of the input units to drop.
* The Output layer consisted of batches of 8 units and ‘[Softmax](https://deepai.org/machine-learning-glossary-and-terms/softmax-layer)’ activation function.
* We then optimized the model using the [Adam algorithm](https://keras.io/api/optimizers/adam/) with a learning rate of 0.00001.
* X\_train and y\_train were the dataframes used to train the model with a batch size of 64 for 70 epochs for the first iteration.

For Iteration 2, we used the prediction from our first model and along with the patient age and sex to attempt to improve the accuracy by using a Neural Network.

* **INPUT**: predictions from Iteration 1 Model, Patient Age, Patient Sex
* **MODEL**: Neural Network with
  1. Dense layer of 60 neurons and ‘[ReLU](https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/)’ (Rectified Linear Unit) as the activation function;
  2. Dense layer of 15 neurons and ‘[ReLU](https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/)’ (Rectified Linear Unit) as the activation function;
  3. Dropout layer with rate 0.5;
  4. Dense output layer with 8 neurons and ‘[Softmax](https://deepai.org/machine-learning-glossary-and-terms/softmax-layer)’ activation function.
* **TRAINING**: We trained the model for 60 epochs

**Performance Metrics and Inferences**

**Iteration 1:**

* Model Accuracy: **77.7%**
* Model Loss: **0.99**
* Running a user-defined function, we were able to plot the trends in Accuracy and Loss over epochs. For Accuracy, we see a steadily-rising curve as the number of epochs increases for both the training and validation dataframes, where the trendlines for the two dataframes overlap quite often.

For Loss, we see a similar trend where the Loss values for both the Training and Validation dataframes are falling over increasing epochs, again overlapping at a few points.

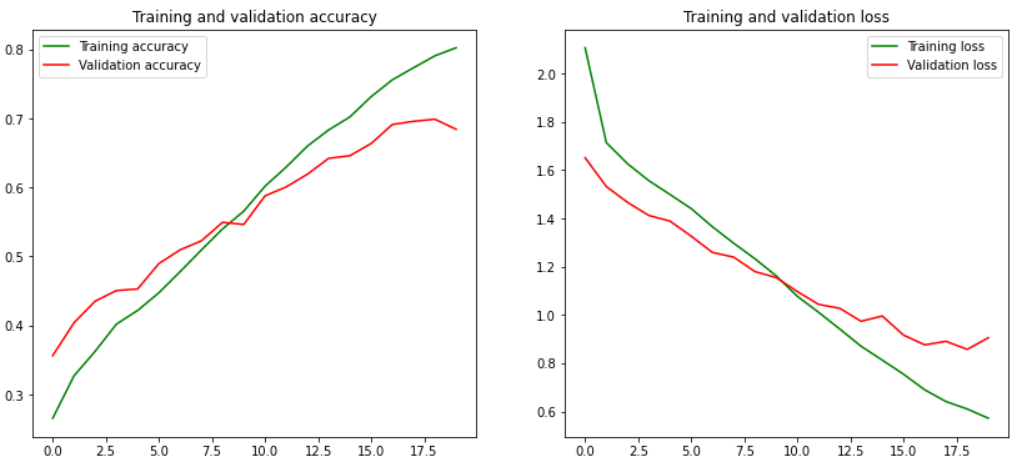
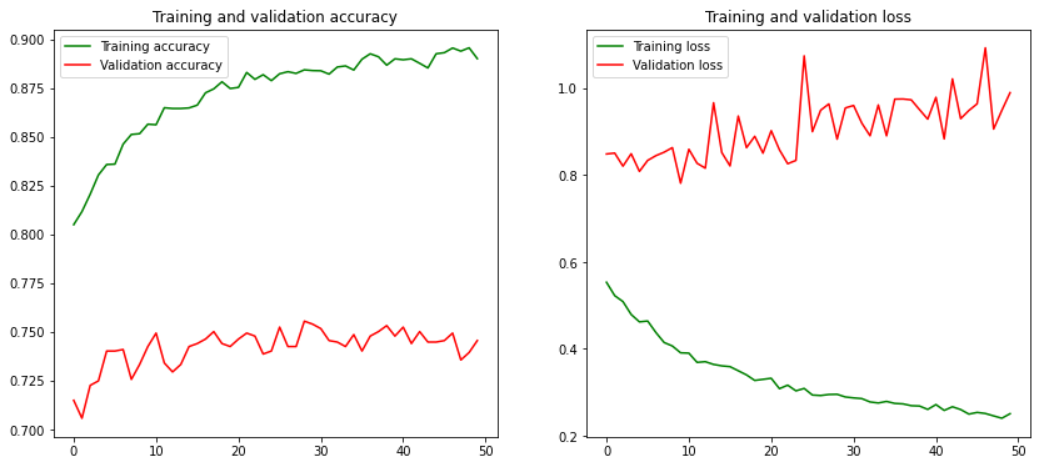
The trendlines lead us to conclude that the model is not overfitting.  
 ****

Figure 27. Model 6 - Accuracy and Loss graphs - Iteration 1

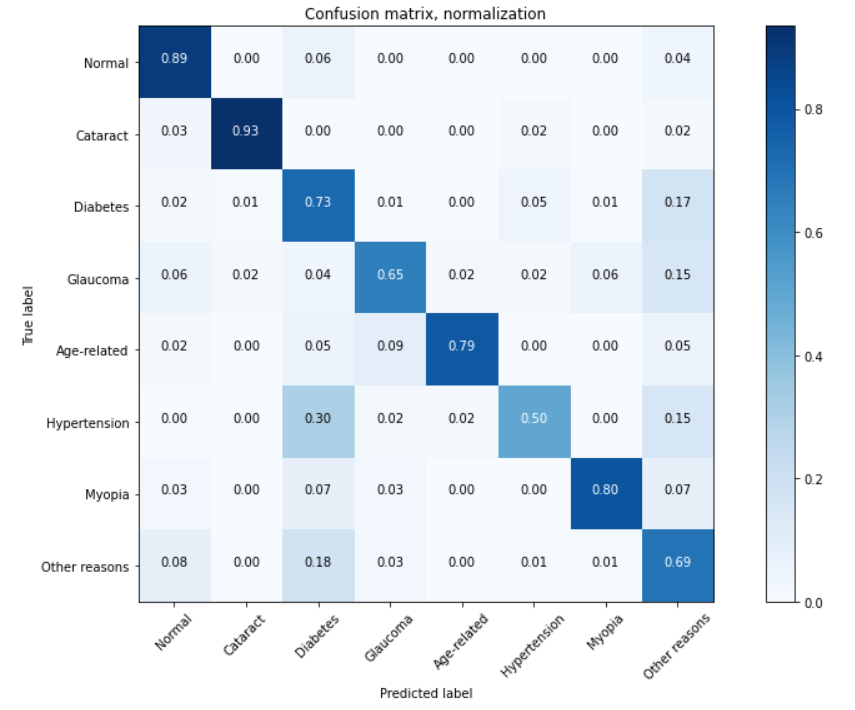
* Following the same pattern as from the previous model, the confusion matrix tells us that the algorithm did well predicting fundi that were Normal or were suffering from Myopia, Cataract, and diseases falling under Other Reasons. Diabetes, Glaucoma, and Age-related diseases got an average performance. Hypertension detection had the worst performance. These performance results make sense since we had a lot of images fed to the model for Normal, Myopia, and Cataract, whereas Hypertension had the least amount of data (382 images). The poor performance could be attributed to a lack data for the model to learn from.   
   

Figure 28. Confusion matrix for Model 6 – Iteration 1

* Classification Report:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **f1-score** | **support** |
| **Normal** | 0.92 | 0.89 | 0.90 | 424 |
| **Cataract** | 0.89 | 0.93 | 0.91 | 61 |
| **Diabetes** | 0.72 | 0.73 | 0.73 | 349 |
| **Glaucoma** | 0.67 | 0.65 | 0.66 | 54 |
| **Age-related** | 0.94 | 0.79 | 0.85 | 56 |
| **Hypertension** | 0.52 | 0.50 | 0.51 | 46 |
| **Myopia** | 0.76 | 0.80 | 0.78 | 40 |
| **Other reasons** | 0.65 | 0.69 | 0.67 | 275 |
|  |  |  |  |  |
| **accuracy** |  |  | 0.78 | 1305 |
| **macro avg** | 0.76 | 0.75 | 0.75 | 1305 |
| **weighted avg** | 0.78 | 0.78 | 0.78 | 1305 |

* Normal, Cataract, Age-related, Diabetes, and Myopia have a good recall and precision as compared to the rest of the diseased fundus which are having an average recall and precision. The poor performance while detecting other diseases seems to be inherited due to the imbalance in our dataset.

**Iteration 2:**

* Model Accuracy: **80.22%**
* Model Loss: **0.6076**
* Running a Neural Network model to boost the performance helped improve the overall accuracy to 80.22% from 77.7%. A slight improvement from the previous model.

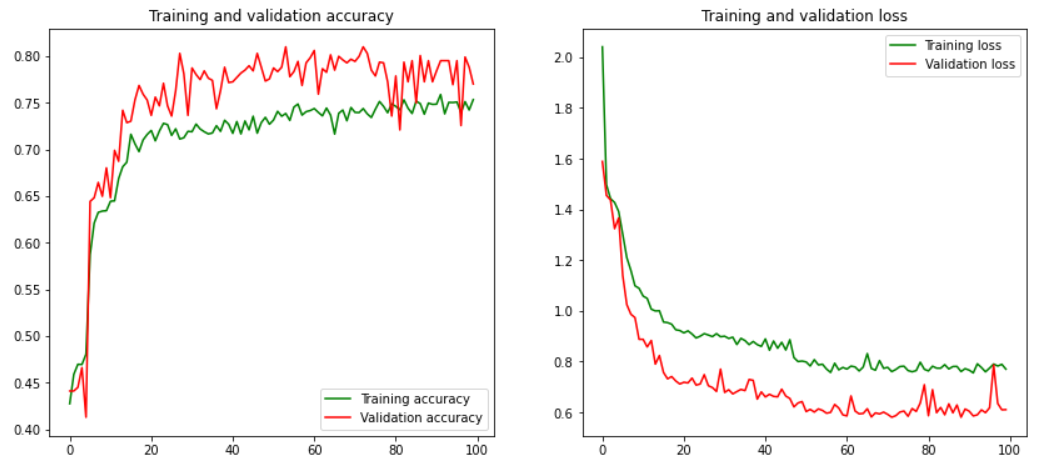


Figure 27. Model 6 - Accuracy and Loss graphs - Iteration 2

# **Results**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Model** | **Accuracy** | **Loss** |
| Model 1 | Simple Sequential Model with multiple layers | Iteration 1:  **31.72%** | **1.6980** |
| Iteration 2:  **31.72%** | **1.6963** |
| Model 2 | Resnet50 | Iteration 1:  **73.10%** | **0.9230** |
| Iteration 2:  **58.70%** | **1.5931** |
| Model 3 | VGG16 with 4 disease types (Normal, Glaucoma, Diabetes, and Cataract) | **89.01%** | **0.3611** |
| Model 4 | VGG16 with all 8 disease types | Iteration 1:  **65.13%** | **1.0082** |
| Iteration 2:  **74.41%** | **0.8624** |
| Model 5 | VGG16 with additional layers for all 8 disease types | Iteration 1:  **73.49%** | **0.8413** |
| Iteration 2:  **76.40%** | **0.8286** |
| Model 6 | VGG16 with additional layers for all 8 disease types followed by boosting with a Neural Network | Iteration 1 (Model A):  **77.7%** | **0.99** |
| Iteration 2 (Model B):  **80.22%** | **0.6076** |

We obtain the best accuracy and lowest loss with model 3 where we passed only 4 out of 8 disease types through VGG16. Unfortunately, we cannot consider this model as our final selection, since it accounts for only half the disease types present. As we turn to the second-best model, Iteration 2 for Model 6 seems to be the best choice as it has a good accuracy score of 80.22% and lowest loss value of 0.6076 when compared to the rest of the models.

The overall pattern was that the models based on the VGG16 and ImageNet algorithm performed well for our data. The model based on Resnet50 had quite decent performance during the first iteration but had a huge drop in accuracy in the second iteration. The Simple Sequential models performed the worst and the model had extremely bad accuracy even in the second iteration.

# **Future Scope**

* Since there was an imbalance present in the data with regards to the disease types, we could try different Data Augmentation techniques such as resampling, rotation of images, changing image brightness.
* Expanding hardware capabilities such as faster GPUs, more CPU memory would enable us to utilize Full HD high resolution images present in the dataset, perform analysis in a single notebook and not cut corners by placing code in multiple notebooks as a workaround.
* More processing power would also mean more variations in the types of models being implemented, complex models being explored, further kinds of activation functions being used, training epochs being longer than those for limited-power notebooks.
* A new model with grayscale images instead of colored images might also be an approach leading to new insights.
* We could collaborate with different opticians to form our own dataset, thereby expanding our already present dataset. This process would also enhance domain knowledge to a great extent through discussions and guidance received from professionals working in the field on a day-to-day basis.

# **References**

* <https://www.kaggle.com/code/tikoboss/data-visualisation-ocular-disease-recognition>
* <https://medium.com/analytics-vidhya/ocular-disease-recognition-using-cnns-4953e4619679>
* <https://github.com/onyekaokonji/OCCULAR-DISEASE-RECOGNITION/blob/main/ODR.ipynb>
* <https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c>
* <https://www.kaggle.com/code/aaya23/finetuning-restnet50-occulardisease-dinia>