ENEL 645: Final Project Report

Intelligent Ocular Disease Recognition

by

Aditya Porwal

Sparsh Mehta

Submitted to:

Department of Electrical and Computer Engineering

Schulich School of Engineering

University of Calgary

Calgary, Alberta

1 Introduction and Objectives

Sight is the most important sense out of the five human senses. An eye can suffer a number of diseases such as Glaucoma, Age-related Macular Degeneration (AMD), Cataract, etc., depending upon several factors like age, sex or even other diseases like Diabetes, which can lead to partial or permanent blindness [1, 2, 3]. According to the World Health Organization (WHO), globally, 1 billion people have a vision impairment that could have been prevented or has yet to be addressed, this includes Cataract (65.2 million), Glaucoma (6.9 million), Diabetic Retinopathy (3 million) among others [4]. Thus, it is evident that the early detection of Ocular diseases would have a great impact in preventing or at least reducing the severeness of eye damage.

In addition to this, the detection of eye diseases is a challenging task for Ophthalmologists. Even with support systems like computer-aided diagnosis (CADx), it can be costly if not applied to large populations [5]. Hence, a deep-learning based approach where features are automatically extracted, selected and classified without the need of domain-specific expert knowledge has great significance. This not only addresses the cost issue for expensive detection, but also cuts down on the requirement of time and professional trained personnel that might not be available in many cases.

In this project of ocular disease recognition, we have developed our own CNN model from scratch, and 3 other models (VGG19, InceptionV3, Xception) using transfer learning techniques. We compare and contrast their performance on metrics such as- Precision, recall, F1 score etc and finally find the best suitable model giving an acceptable classification accuracy for this problem.

2 Dataset

The dataset used to train our models was generously provided by Shanggong Medical Technology Co.Ltd. [10] in a an international competition on the 'Global Grand Challenge' website organised by Peking University (PKU). The database, named as Ocular disease intelligent recognition (ODIR) database, is now publicly available on the online machine learning community- Kaggle [9]. It contains images of the Fundus of the left and right eyes of 5000 patients. Dataset is having approximately 7000 images with labels for training and 1000 images without labels for testing. Around 5000 pre-processed images are also provided, with the pre-processing techniques unknown.

By observing the pre-processed images and applying few techniques on normal images, we achieved similar looking images by centralizing the Fundus, cropping unnecessary black portions and resizing the images. So, we think these were the processing techniques used by the author. Note: Because of the OpenCv library (which reads an image in the BGR format) the images appear to have bluish tint, we solved this problem by simply converting the images back to the RGB format.

3 Prior literature

This topic is very vast and yet to be explored. There are limited past studies where authors propose disease classification using modern deep-learning algorithms for ocular diseases recognition. For instance, Xiangyu Chen et al. (2015) [6] used a six layer architecture (four convolutional layers and two fully-connected layers with output fed to a soft-max classifier) for the detection of Glaucoma. Their model achieved area under Receiver Operating Characteristic (ROC) curve at 0.887.

Another work authored by Abbas Q., used an integrated approach of combining Convolutional Neural Network (CNN), Deep Belief Network (DBN) and the Softmax deep-learning classifiers [7]. They curated a dataset of 1200 images containing normal Fundi and Glaucoma diseased Fundi, and extracted the Regions of Interest (ROI)- Optic Disc (OD) and Optical Cup (CUP) for deep learning features' classification. They achieved an accuracy of 99% and precision of 84% but again concentrated their classification on one disease.

A study on Cataract detection [8] using Deep Convolutional Neural Network (DCNN) achieved 93.52% accuracy, working with only the G-filter (Green component) of RGB Fundus images, increasing the time-efficiency. They also experimented with the scalability of database effects on accuracy in DCNN classification and established that the accuracy continuously improved with the amount of available samples. As a Kaggle competition[12], a dataset for diabetic retinopathy detection was introduced and the winning solution proposed by Graham et al. used SparseConvNet CNN with min-pooling for their classification of Fundus images. Various image pre-processing and augmentation techniques were applied before providing the images as input to the architecture.

Various authors have developed deep learning models for the ODIR dataset. Islam et al. [13] implemented a CNN architecture which used the left and right eye Fundi individually to input to their model. They also considered the samples having a single label, this reduced the problem complexity and lead them to achieve an accuracy of 88% with AUC value of 80.5%. Another work [14] considered this as a multi-label and multi-class problem and compared various combinations of CNN architectures, and two approaches- considering left and right Fundus individually, and then in concatenated form. Their best model received the AUC and F1 score of around 85% and 85%, respectively.

4 Methodology

4.1 Exploratory Data Analysis

We are using an open source dataset available on the Kaggle platform under the name "Ocular Disease Recognition; Right and left eye Fundus photographs of 5000 patients" [9, 10]. This is a structured ophthalmic database constructed using images of the Fundus of the eye. An eye Fundus is the interior surface of the eye behind the lens that includes the Retina, Optic Disc (OD), Macula, Fovea, and Posterior pole (see Figure 1). All these areas of interest play an important role in detecting diseases in the eye images. The dataset was created with the input from many domain experts who provided diagnostic keywords, and labeled the images with corresponding diseases. In addition to this, information such as the age, gender of the patient is also provided.

Following is a brief introduction to the dataset:

- Train Images: 7000, Test Images : 1000 (no labels)
- Dataset includes images of both, the left and the right Fundus (with no NULL values present)
- Pre-processed Images (provided in the ODIR dataset): 5000 Images
- Eight classes: Normal, Diabetes, Glaucoma, Cataract, AMD, Hypertension, Myopia, and Others.
- Diagnostic Keywords Combinations are provided for each eye (Left and Right), along with the patient ID, age, sex and labels.
- A patient might be suffering from more than one disease. This was verified by plotting the correlation matrix during data exploration. Hence, a multi-label classification problem.

4.2 Dataset Manipulation

The dataset we are using is highly imbalanced in terms of number of image samples per class of disease. For instance, normal Fundi constitute around 1140 images while for Myopia only 174 samples are provided (Figure 2a). In accordance with our final goal for this project, i.e. to classify ocular diseases into 4 categories- normal, Cataract, Glaucoma and Myopia, we extract corresponding images from the main dataset, creating a mini-dataset to feed into our model. This task was achieved using writing a python script, in which we checked all the diagnostic keywords of each and every image and shortlisted the ones which had 'cataract' mentioned in them. After that, all the labels were scanned for the label 'C' (corresponding to Cataract) and if one was found, it was stored in a data frame. This was repeated for the Normal, Glaucoma and Myopia classes as well, and finally concatenating the data frames a sub-dataset was created.

This sub-dataset contains normal eye Fundus images (around 600), and images suffering from cataract (around 600), Glaucoma (around 600), and Myopia (around 450) (Figure 1). According to a published research [15], it is beneficial to use Contrast Limited Adaptive Histogram Equalisation (CLAHE) to bring out minute details of the Fundus in such datasets. We then convert the image to the 'LAB color format' (L- Lightness, A- color component ranging from Green to Magenta, B- color component ranging from Blue to Yellow) [16], and used CLAHE image processing technique on the L channel of the image, which resulted in images shown in Figure 1.

This new dataset with processed images is then divided into the training (60%), validation (30%) and test (10%) sets, i.e. approx. 2040 images for the model development (Train and selection) and 230 images for model testing. We used Min-Max dataset normalization (normalization with mean-standard deviation was also tried but it gave slightly poor results) to change the values of numeric columns in the dataset to use a common scale. Min-max dataset normalization is done using the following formula:

$$X_{norm} = \frac{(X - X_{min})}{(X_{max} - X_{min})} \qquad \text{where X= train, validation and test samples}$$

The labels are represented using the one hot encoding, which is a type of categorical binary representation that maps the categorical data to numbers that machine learning algorithms can work with.

Deep learning models need as much data as possible to learn and then classify images. Considering our case of 4 classes, insufficient data, and imbalanced dataset, we employed Data Augmentation techniques such as rotating (rotation range- 15 degrees), and horizontal flipping of the images. Even after using these techniques, this much amount of data is not sufficient and at some point of time even data augmentation might start feeding redundant images to the model, thus, not letting the model learn new information. We avoided the use of other augmentations techniques to preserve the minutest details in the Fundus images that may affect the overall decision making if tampered. We used the batch size of 32 for this task. Data Augmentation was used on both- train and validation set.

4.3 Models and their Training

To achieve the best possible performance on our dataset, we tried 4 different models (CNN, VGG16, Inception, Xception) with many variations.

4.3.1 Convolutional Neural Network (CNN) model

Key points (Remains same for all the models):

- Initial learning rate: $1e^{-5}$
- Learning rate for Fine tuning: $1e^{-6}$
- Did not import original classification network for any of the models
- Same classification network and training parameters for transfer learning.

Owing to the fact that our dataset is quite small in comparison to the ones needed to train deep learning models, we knew beforehand that training a deep neural network from scratch might not yield satisfactory results. Still, as a learning exercise, after experimenting we developed a model with the following configurations:

- Convolutional layers: 10
  Dense layers: 4 (including the output dense layer)
- Optimizer: Stochastic Gradient Descent (SGD)
- Trainable Parameters: 197,160,836
  Non-Trainable Parameters: 9,728

The model trained for 3 epochs, after which it started to over-fit and was eventually stopped by a call-back. Dropouts techniques are employed to let the model learn redundant paths to the same output. This model had its own limitations (prime issue being such a small dataset). As expected, the best accuracy achieved by CNN after varying the hyper-parameters was 30%, and it classified every image in the normal class. Thus we shifted transfer learning approaches.

Next, we implemented transfer learning on three pre-trained models (VGG16, Inception and Xception) to see if there were any improvements in the results.

4.3.2 VGG19 model

About the VGG19 model (pre-trained on the ImageNet dataset):

- VGG19 has 16 convolution layers, 3 Fully connected layers, with the final fully connected layer having Softmax activation function as the non-linear part.
- The base convolutional layers having pre-trained weights were frozen.
- For our classification network: We used two dense layers. The final dense layer has four neurons (one for each class) and the activation function as Softmax activation.
- For fine-tuning, we unfroze the base convolutional layers and ran the model for additional 20 epochs or the callbacks stopped the model from training. (Whichever is smaller)
- For classification, total trainable parameters were 100,356, while non-trainable were 20,024,384. Whereas for fine-tuning, they were 20,124,740 and 0 respectively.
- The network was trained with call-backs, to stop training the network in case of over-fitting.
- Training parameters:
  1. Loss Function: Categorical Cross-entropy
  2. Optimizer: ADAM

The pre-trained weights helped a lot in adjusting the classification network to our dataset. VGG19 showed significant improvement in the accuracy, loss values. This model clocked out at 77.7% accuracy and a loss of 0.532 after fine tuning.

4.3.3 InceptionV3 model

The third model we worked upon was InceptionV3. Again, using transfer learning, we trained this model on our dataset and noted its performance metrics. Model specifications are:

- InceptionV3: a quite complex model with concepts such as: Factorized convolutions, Smaller convolutions, Asymmetric convolutions, Grid size reduction.
- The base convolutional layers having pre-trained weights were frozen.
- For classification, total trainable parameters were 13,108,484 & non-trainable were 21,802,784. Whereas for fine-tuning they were 34,876,836 and 34,432 respectively.

With the results getting better with each epoch, this model started to over-fit at 12th epoch and was stopped by call-back. The best saved model is then fine-tuned and resulted in the accuracy score of 82.22% and loss of 0.4032 on the test set.

4.3.4 Xception model

Finally, we used transfer learning on the Xception network. Some details for this model are:

- It was first termed as the extreme version of Inception model, hence, Xception [17]
- Based entirely on depth-wise separable convolution layers
- 36 convolutional layers forming the feature extraction base of the network
- The 36 convolutional layers are structured into14 modules, all of which have linear residual connections around them, except for the first and last modules [17]
- The base convolutional layers having pre-trained weights were frozen
- For classification, total trainable parameters were 25,691,396 & non-trainable were 20,861,480. Whereas for fine-tuning, they were 46,498,348 and 54,528 respectively.

Xception performed extremely well on our dataset and gave us decent results with the loss and accuracy values of 0.343 and 85.3%.

5 Performance Metrics and Results

We developed and trained the models using around 2040 images, and now we test them with the test-split dataset consisting approximately 230 samples. The area under receiver operating characteristics (ROC) curve is shown in Figure 3a. The final performance of the models is evaluated on Accuracy, Precision , Recall and F1 score which are defined in terms of true positive (TP), false positive (FP), true negative (FP), and false negative (FN).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \qquad Precision = \frac{TP}{TP+FP}$$

$$Recall = \frac{TP}{TP+FN} \qquad F1score = 2 * \frac{Precision * Recall}{(Precision + Recall)}$$

Precision talks about how precise our model is. That is, out of all the predicted positives, how many are true positives. Achieving a good score on this metric is important for this problem. For instance, while deciding whether or not an eye is normal, predicting an eye suffering from cataract as a normal eye, is a case of false positive. This leads to the ignorance of the disease and ultimately results in partial/permanent blindness. Our final model (Xception) achieved a precision of 83.5%.

Recall on the other hand is calculated using False Negatives and True Positives. Recall is the model metric used to select the best model when there is a high cost associated with False Negative, which is also a perfect metric for our case. For example, a person has a perfectly normal eye but he is diagnosed to have the cataract disease (False Negative). This might make him take unnecessary treatment which is in no condition acceptable. Recall for our model was 97.9%.

F1 Score is needed when we want to seek a balance between Precision and Recall. The same behaviour can be achieved using accuracy, but it can be largely contributed by many True Negatives which in many cases might not make a difference. We got a F1 score of 85.3% (Table 1).

Talking about the results, with time the loss of the network decreased and accuracy increased. Loss on test set turned out to be 0.3425 while the accuracy reached 0.8559 (Figure 4). The deep learning network with the best results in the performance metrics is found to be Xception (refer Table 1).

5.1 Other things we tried

A number of different configurations were tried based on:

- Number of convolutional layers
- Number of kernels and kernel size
- Number of dense layers and neurons in these layers
- Loss function: Binary or Categorical cross-entropy
- Optimizer: Adaptive Momentum (ADAM) or Stochastic Gradient Descent (SGD)

Additionally, classification of all 8 classes (considering multi-label approach- using problem transformation methods; where we transformed problem into a set of binary classification problems) was also tried. Even though the model achieved an accuracy above 80%. The model still was predicting everything as a single class (as depicted in Figure 5). This indicated, that model was learning only redundant data and more data is required to get a satisfactory score on all metrics.

6 Conclusion and Future work

1. Training a deep learning model from scratch without sufficient data is not a just option.
2. Using transfer learning techniques on the three models in study, we compensate this requirement of more data samples.
3. While the pre-trained models can be useful in some cases, a further research is required for conclusive evidence (like why some models fit the dataset better than others?).
4. Among all models, the best model suited for the dataset was Xception.

7 Figures

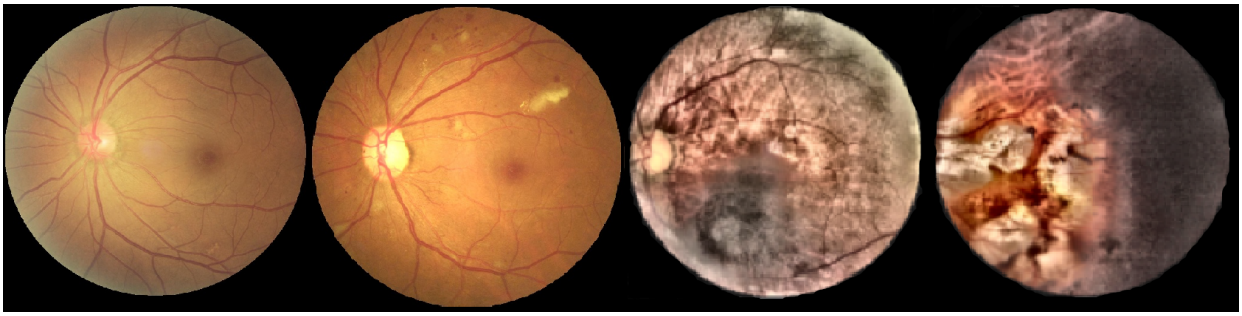
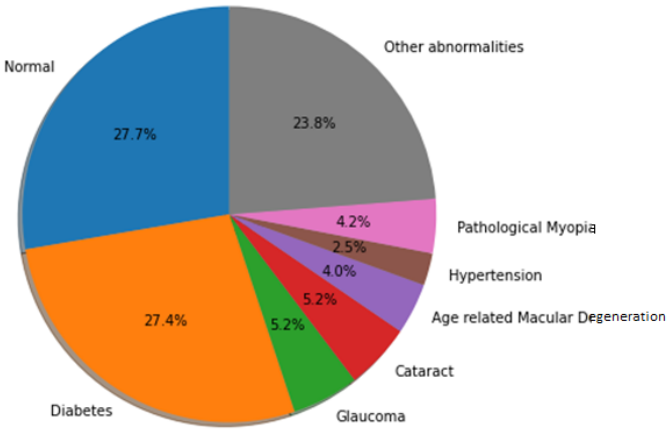
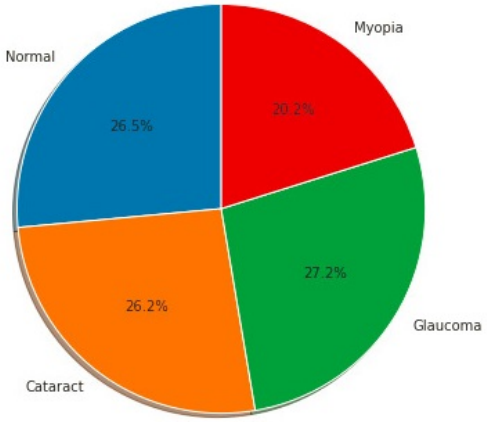


Figure 1: Eye Fundi images- first two are without any pre-processing, while the last two are CLAHE processed

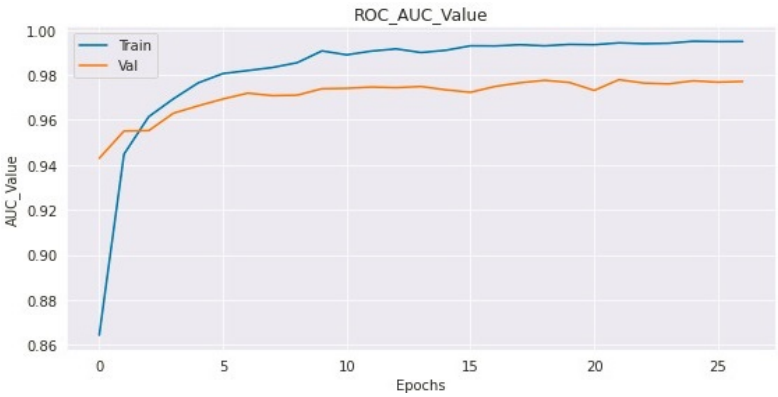


(a) Highly imbalanced ODIR dataset

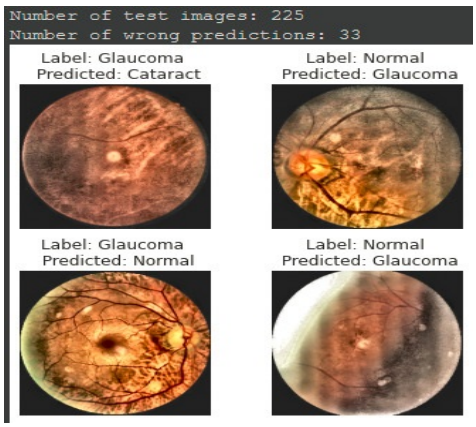


(b) Our sub-dataset with balanced sample distribution

Figure 2: Pie graph to visualize dataset distribution



(a) Behaviour of Area Under Curve (AUC) with number of epochs



(b) Predictions that the model wrongly guessed

Figure 3: Other metrics for performance analysis

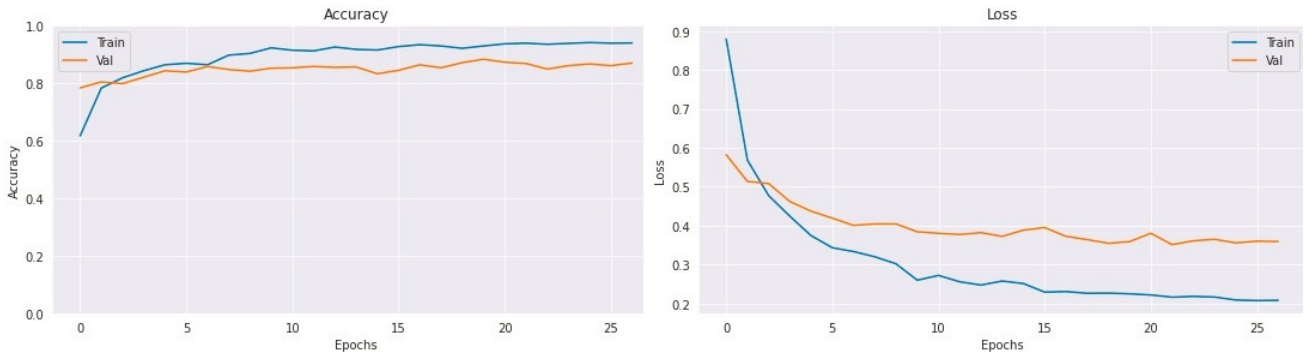


Figure 4: Plot of Loss and Accuracy against number of epochs

```
[29]  1 model.load_weights(model_name)
      2
      3 loss, accuracy_multilabel, f1_score, auc_value = model.evaluate(test_generator, verbose = 0)
      4 print(f'loss: {loss} \tAccuracy: {accuracy_multilabel} \tF1_score: {f1_score} \tAUC_value: {auc_value}')

loss: 0.33974963426589966      Accuracy: 0.8683965802192688    F1_score: 0.3497931659221649    AUC_value: 0.6503129601478577

[48]  1 Y_pred = model.predict(X_test)
      2 right_indexes = np.where(tf.cast(tf.round(Y_pred), tf.int32) == Y_test)[0]
      3 print(f'Y_pred:\n {tf.cast(tf.round(Y_pred[right_indexes][4:6]), tf.int32)}')
      4 print(f'Y_test:\n {Y_test[right_indexes][4:6]}')

Y_pred:
 [[1 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]]
Y_test:
 [[0 1 0 1 0 0 0 0]
 [1 0 0 0 0 0 0 0]]
```

Figure 5: Results of Multi-label classification

	CNN	VGG19	InceptionV3	Xception
Loss	1.389	0.532	0.403	**0.343**
Accuracy	0.302	0.777	0.822	**0.853**
ROC AUC Value	0.0	0.799	0.831	**0.850**
Recall	0.500	0.947	0.968	**0.979**
Precision	0.0	0.742	0.808	**0.835**
F1 Score	0.140	0.773	0.822	**0.853**

Table 1: A comparison between all the models we tried with their respective results on different metrics tested on the test set

References

[1] Yau JW, Rogers SL, Kawasaki R, Lamoureux EL, Kowalski JW, Bek T, Chen SJ, Dekker JM, Fletcher A, Grauslund J, Haffner S, Hamman RF, Ikram MK, Kayama T, Klein BE, Klein R, Krishnaiah S, Mayurasakorn K, O'Hare JP, Orchard TJ, Porta M, Rema M, Roy MS, Sharma T, Shaw J, Taylor H, Tielsch JM, Varma R, Wang JJ, Wang N, West S, Xu L, Yasuda M, Zhang X, Mitchell P, Wong TY; Meta-Analysis for Eye Disease (META-EYE) Study Group. Global prevalence and major risk factors of diabetic retinopathy. Diabetes Care. 2012 Mar;35(3):556-64. doi: 10.2337/dc11-1909. Epub 2012 Feb 1. PMID: 22301125; PMCID: PMC3322721.

[2] Cheung N, Mitchell P, Wong TY. Diabetic retinopathy. Lancet. 2010 Jul 10;376(9735):124-36. doi: 10.1016/S0140-6736(09)62124-3. Epub 2010 Jun 26. PMID: 20580421.

[3] Ting DSW, Cheung CY, Lim G, et al. Development and Validation of a Deep Learning System for Diabetic Retinopathy and Related Eye Diseases Using Retinal Images From Multiethnic Pop-

ulations With Diabetes. JAMA. 2017;318(22):2211–2223. doi:10.1001/jama.2017.18152

[4] Bourne RRA, Flaxman SR, Braithwaite T, Cicinelli MV, Das A, Jonas JB, et al.; Vision Loss Expert Group. Magnitude, temporal trends, and projections of the global prevalence of blindness and distance and near vision impairment: a systematic review and meta-analysis. Lancet Glob Health. 2017 Sep;5(9):e888–97.

[5] Anton A, Fallon M, Cots F, Sebastian MA, Morilla-Grasa A, Mojal S, Castells X. Cost and detection rate of glaucoma screening with imaging devices in a primary care center. Clin Ophthalmol. 2017;11:337-346

[6] Xiangyu Chen, Yanwu Xu, Damon Wing Kee Wong, Tien Yin Wong, & Jiang Liu (2015). Glaucoma detection based on deep convolutional neural network. Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual International Conference, 2015, 715–718. https://doi.org/10.1109/EMBC.2015.7318462

[7] Abbas, Q. (2017). Glaucoma-deep: detection of glaucoma eye disease on retinal Fundus images using deep learning. Int J Adv Comput Sci Appl, 8(6), 41-5.

[8] Linglin Zhang et al., "Automatic cataract detection and grading using Deep Convolutional Neural Network," 2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC), Calabria, 2017, pp. 60-65, doi: 10.1109/ICNSC.2017.8000068.

[9] Larxel. *Ocular Disease Recognition: Right and left eye Fundus photographs of 5000 patients*. kaggle. Retrieved February 17, 2021, from https://www.kaggle.com/andrewmvd/ocular-disease-recognition-odir5k

[10] Shanggong Medical Technology Co. Ltd. *Peking University International Competition on Ocular Disease Intelligent Recognition (ODIR-2019)*. Grand Challenge. Retrieved February 17, 2021, from https://odir2019.grand-challenge.org/introduction/

[11] Agrawal R. *Optimization Algorithms for Deep Learning*. The Medium. Retrieved February 17, 2021, from https://medium.com/analytics-vidhya/optimization-algorithms-for-deep-learning-1f1a2bd4c46b

[12] *Diabetic Retinopathy Detection*. Kaggle. Retrieved February 17, 2021, from https://www.kaggle.com/c/diabetic-retinopathy-detection

[13] M. T. Islam, S. A. Imran, A. Arefeen, M. Hasan and C. Shahnaz, "Source and Camera Independent Ophthalmic Disease Recognition from Fundus Image Using Neural Network," 2019 IEEE International Conference on Signal Processing, Information, Communication and Systems (SPICSCON), Dhaka, Bangladesh, 2019, pp. 59-63, doi: 10.1109/SPICSCON48833.2019.9065162.

[14] N. Gour and P. Khanna, "Multi-class multi-label ophthalmological disease detection using transfer learning based convolutional neural network", Biomed. Signal Process. Control, vol. 66, Dec. 2020.

[15] Sonali, and Sahu, Sima and Singh, Amit and Ghrera, S.P. and Elhoseny, Mohamed. (2018). An approach for de-noising and contrast enhancement of retinal fundus image using CLAHE. Optics and Laser Technology. 110. 10.1016/j.optlastec.2018.06.061.

[16] Gupta, V. *Color spaces in OpenCV (C++ / Python)*. learnopencv. Retrived April 2, 2021 from https://learnopencv.com/color-spaces-in-opencv-cpp-python/

[17] Chollet, Francois. (2017). Xception: Deep Learning with Depthwise Separable Convolutions. 1800-1807. 10.1109/CVPR.2017.195.

Work distribution and Consensus Score

All the work required for this project was completed with equal efforts from both the team members. From exploring the dataset and model specifications, to reading and reviewing prior art, and finally documenting the work over Overleaf, we have constantly worked upon each other's findings. We coordinated our work over WhatsApp (social media platform) and telephonic conversations. Reviewing and understanding previous published papers was achieved by splitting the workload (each member working with half the number of references). This enabled us to develop different perspectives on the methodology that would be used to develop this project, which in turn catalyzed meaningful discussions. The presentation was created over Zoom meeting, with balanced appearance from both members. Please refer Table 2 for the score.

Member	Score
Aditya Porwal	3
Sparsh Mehta	3

Table 2: Consensus Table

Appendix A Important Links

Our git repository can be found here: GitHub
For step by step directions to successfully run the project, follow this link : Code running guide
The saved models can be accessed from here: Google Drive