**Sight Sage: Eye Disease Detection Model**

**Submitted for**

**Machine Learning (UCS411)**

**Submitted by:**

**(102103748)              Svea Chawla**

**(102103773)              Swapnil Chhabra**

**BE Third Year (3CO27)**

**Batch: 2021-25**

Submitted to-

Dr. Ashutosh

Assistant Professor



**Computer Science and Engineering Department**

**TIET, Patiala**

**Introduction of Project**

Sight Sage is a new project working to improve visibility. The main purpose is to detect eye diseases by simply uploading retinal images. In a world where technology meets health, Sight Sage offers innovative solutions for fast and easy eye testing.

Our model demonstrates efficiency in a comprehensive dataset comprising 4217 high resolution industry standard images, that specifically target the detection of early stages of three diseases- Glaucoma, Diabetic Retinopathy and Cataract.

The inherent strength of CNN lies in its ability to autonomously learn hierarchical features from image data. This translates to a granular analysis and recognition of patterns/ anomalies within the ocular structures when compared to a healthy eye.

CNNs pick up the finer indicators of the onset or the incipient stages often imperceptible to the untrained eye. By harnessing this very quality of CNNs, our model aims to provide a high standard of precision and efficacy in identifying ocular pathologies, opening up avenues for early interventions and improving disease detection.

**Significance of this Project**

Visual impairment is a major global health concern, affecting over 2.2 billion people worldwide. Among these individuals, approximately 50 million are blind, and 250 million have moderate to severe vision impairment. Cataract, glaucoma, and diabetic retinopathy are the leading causes of vision impairment, accounting for a significant proportion of these cases.

Traditional methods of eye disease detection rely on manual examination by ophthalmologists or optometrists. However, these methods are often time-consuming, resource-intensive, costly and prone to human error. Machine learning offers a promising solution to overcome these limitations and improve the accuracy, efficiency, and accessibility of eye disease detection.

The primary objectives of this project are:

1. **Develop a machine learning model capable of accurately detecting cataract, glaucoma, and diabetic retinopathy from retinal fundus images.**
2. **Evaluate the performance of the model on a diverse dataset of images, including those with varying image quality and resolution.**
3. **Assess the potential impact of the model on improving early detection and timely intervention for eye diseases.**

**About the Dataset**

The dataset for this project consists of a collection of retinal fundus images labelled with the presence or absence of three eye diseases: Cataract, Glaucoma, and Diabetic retinopathy. The dataset was gathered from [Kaggle](https://www.kaggle.com/datasets/gunavenkatdoddi/eye-diseases-classification). It includes images of both healthy and diseased eyes, with a variety of image qualities and resolutions.

The dataset is divided into three main categories: training, testing and validation. The training set is used to train the Machine Learning model, the testing set is used to evaluate its performance on unseen data, and the validation set is used to tune the hyperparameters of the classifier. The dataset is further divided into four subcategories out of which 3 correspond to the three eye diseases being detected: Cataract, Glaucoma, Diabetic Retinopathy and the 4th category is Normal retinal fundus images.

**Dataset Characteristics:**

Number of images: 4217

* Cataract: 1038
* Glaucoma: 1007
* Diabetic Retinopathy: 1098
* Normal: 1074

Image labelling: Multi-Class (Normal vs. Diseased)

**METHOD USED**

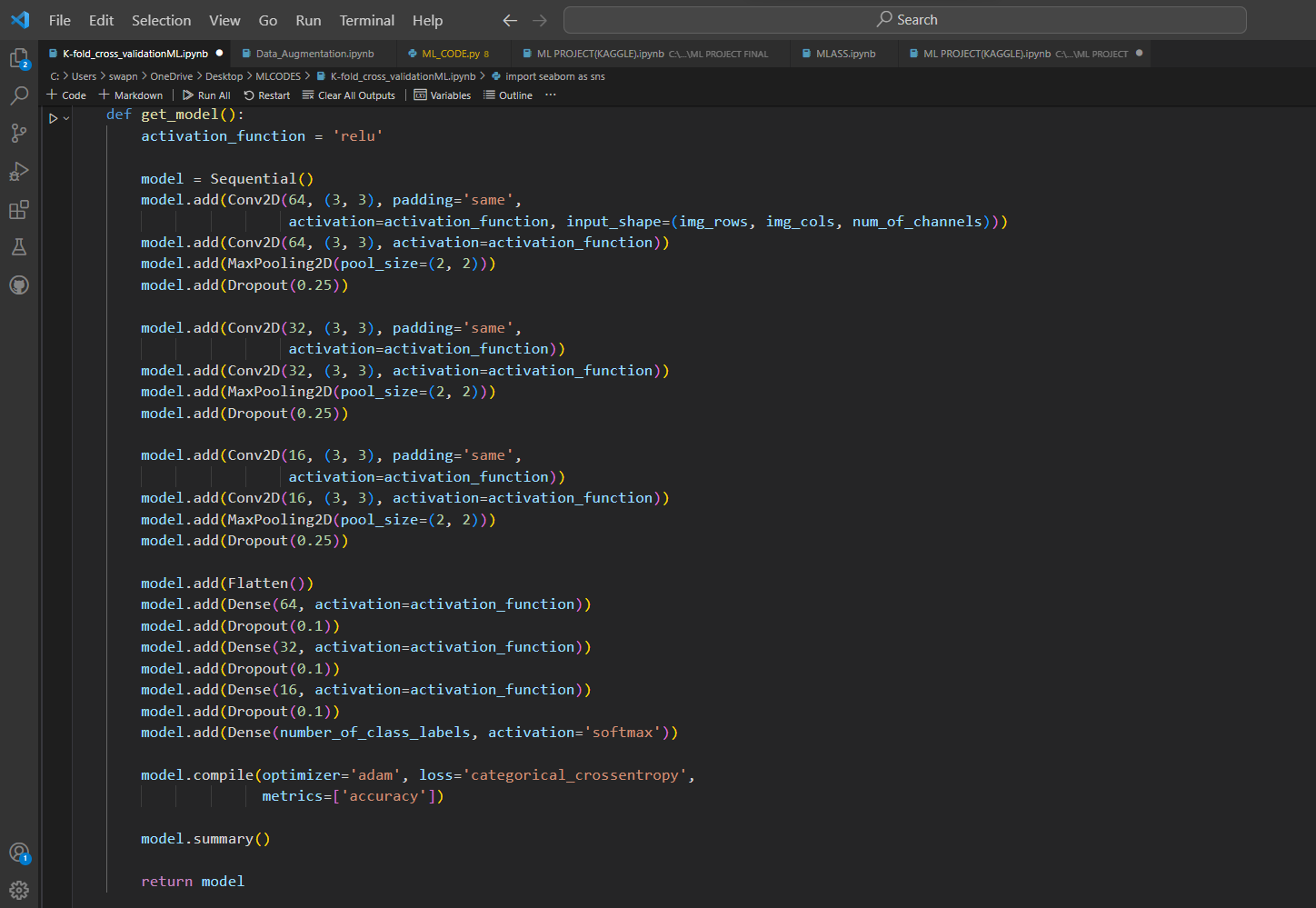
1. **Storing Dataset:** We create 3 sub-folders train, test and validation inside a main dataset folder. The dataset downloaded from Kaggle is stored in the train folder with each image mapped to its class label. The test and validation sub-folders are initially kept empty. In the code we have also designed functions to check that the test and validation folders remain empty before we train the model.
2. **The CNN Model:** Our CNN model has been made from scratch. The model contains 6 MaxPooling2D layers and 4 Dense layers, the MaxPooling2D layer helps to select the maximum element from the region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

The Dense layer is that Convolutional Layer uses fewer parameters by forcing input values to share the parameters. The Dense Layer uses a linear operation meaning every output is formed by the function based on every input.

Activation method used for each layer is ReLU (Rectified Linear Unit) introduces non-linearity to the model. While convolutional layers themselves apply linear transformations to the input data, stacking these linear operations without non-linear activation functions would result in a model that can only learn linear relationships.

The activation method used for last activation layer is softmax.

We chose a batch size of 32 which was suitable according to the size of our dataset. The number of epochs used per fold is 20 and the initial input channel count is 3 as it is RGB.

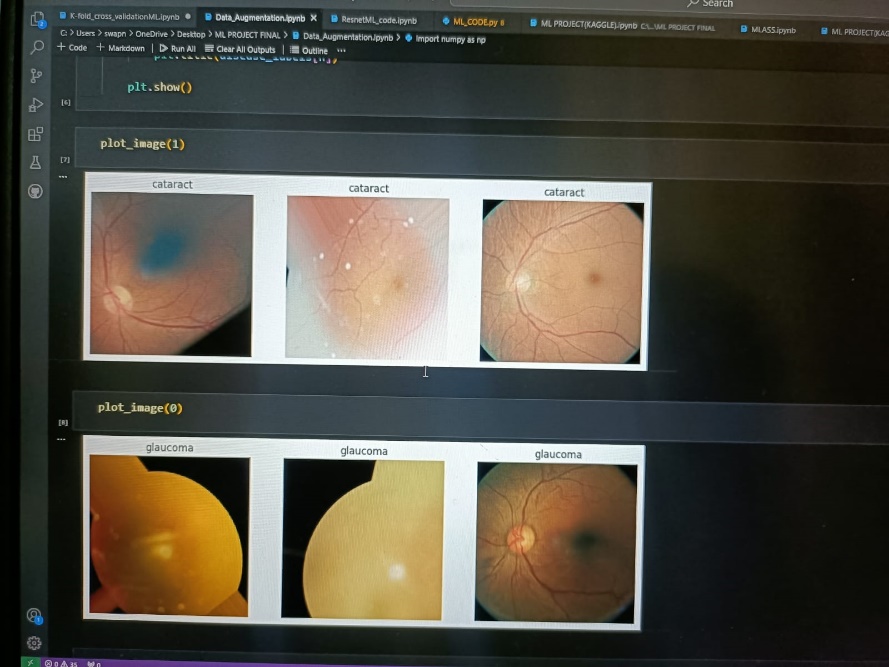


***Figure:*** CNN Structure

1. **The stratified K-fold cross validation:** We havedevelopeda cross-validation loop for training a convolutional neural network (CNN) for image classification using the Keras library. The cross-validation is performed using Stratified K-Fold, where the dataset is split into training and validation sets in each fold (No. of folds=5), ensuring that the class distribution is maintained. Within each fold, image data augmentation is applied to the training set using the ImageDataGenerator from Keras. The ImageDataGenerator augments images artificially without storing them in the original dataset.

The code then trains the CNN model for 20 epochs and collects the training accuracy for each epoch. After training, it evaluates the model's performance on the validation set, computing metrics such as accuracy, precision, and F-score. This process is repeated for each fold, and the training accuracy for each epoch is plotted for visual analysis. Finally, the code processes the test data using the trained model and evaluates its performance on the test set, computing and printing metrics like accuracy, precision, and F-score.

We used K-fold cross validation as it trains and validates the model k times, each time using a different fold as the validation set. This allows the model to be trained and tested on different subsets of the data, maximizing the use of the available data for both training and evaluation. K-fold also maintains equal distribution of data for training and validation which helps to train unbalanced data in a better way.



***Figure:*** Showing dataset images after data augmentation

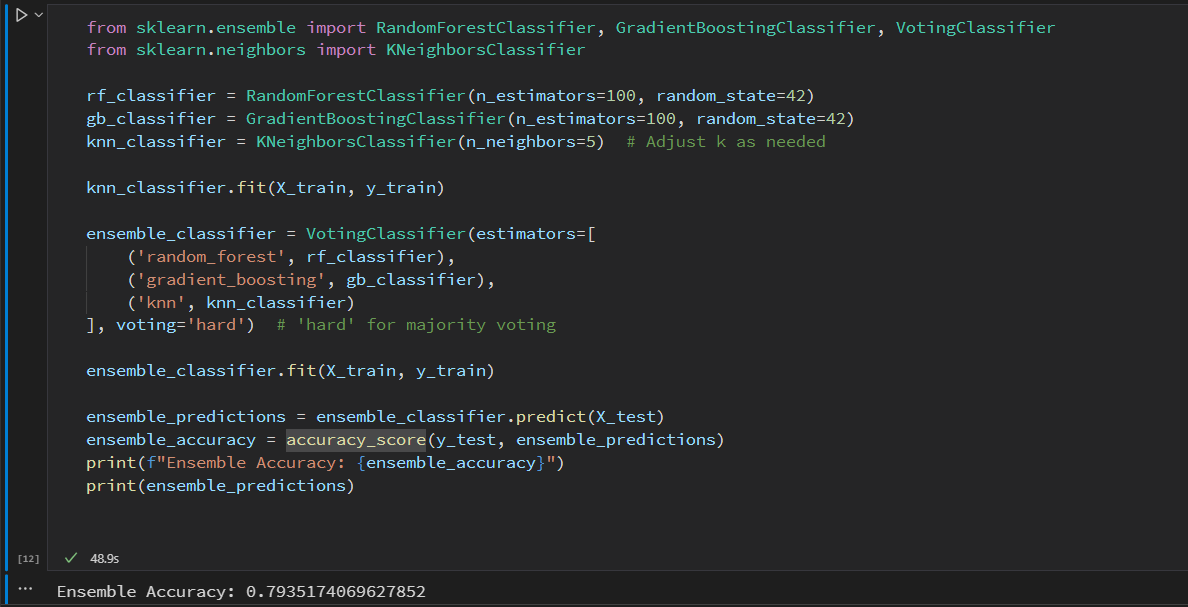
We initially tried to implement data augmentation and storing augmented images but it only resulted in distorting the clarity of the dataset and hence reducing accuracy (65%).

**Comparing our CNN Model with other models on the same dataset**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Model** | **Train Accuracy** | **Validation Accuracy** |
| **1.** | **Ensemble** | **84.73** | **79.35** |
| **2.** | **Densenet** | **91.26** | **88.26** |
| **3.** | **Resnet50** | **95.13** | **86.32** |
| **4.** | **VGG16** | **94.96** | **82.55** |
| **5.** | **Data Augmentation** | **98.33** | **65.94** |
| **6.** | **CNN With K-fold (after 5th fold)** | **83.41** | **82.33** |

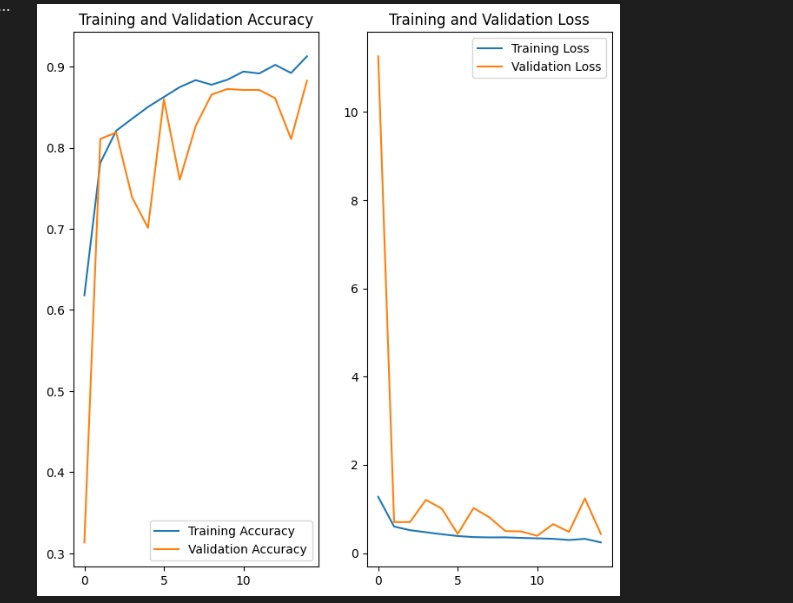
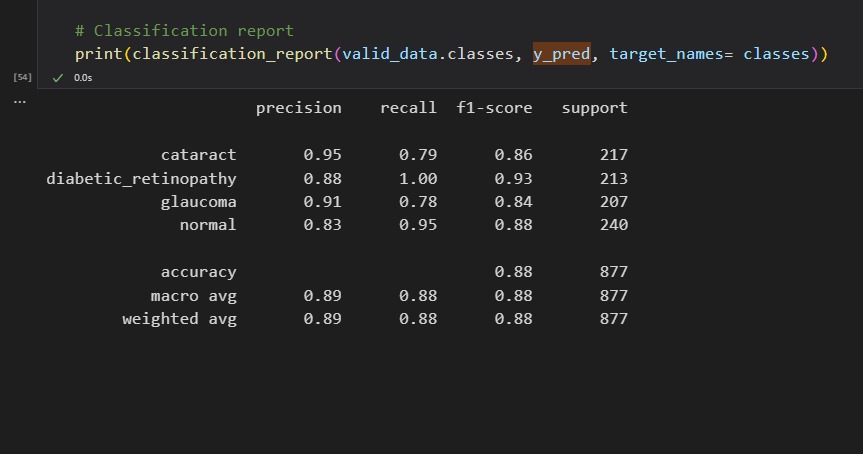
The final test accuracy of our CNN model is: 80.00

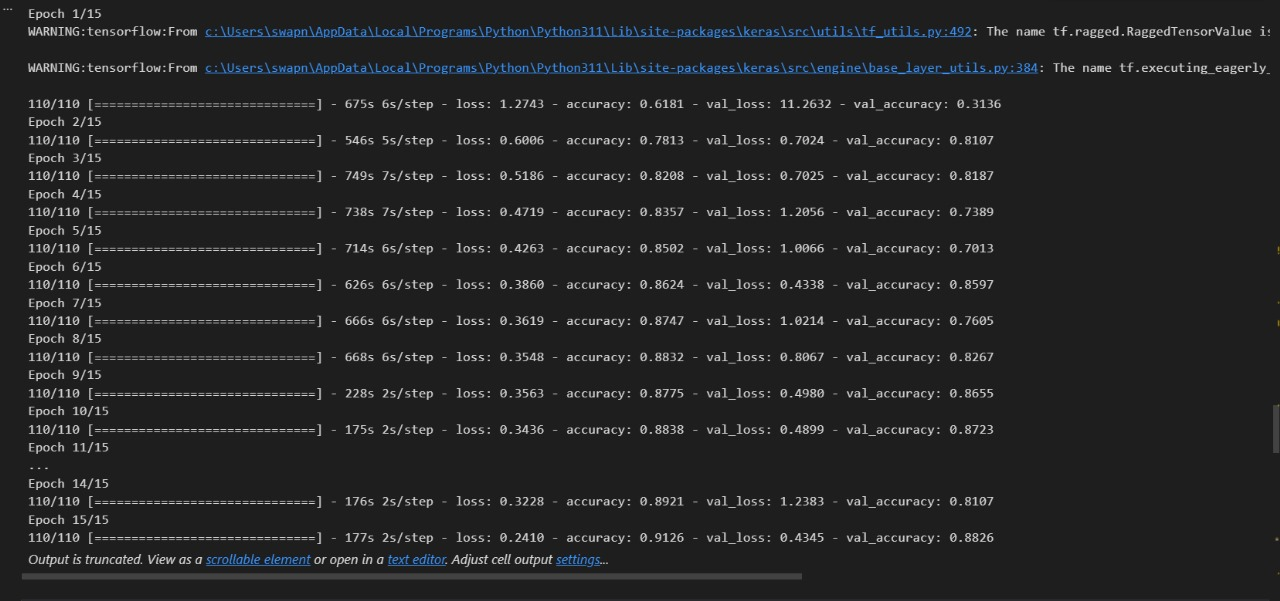
**1. Ensemble Coding:** RandomForestClassifier, GradientBoostingClassifier, KNeighbourClassifier were applied on the dataset and then they were all ensembled using the Voting Classifier. Ensemble learning is a machine learning technique that involves combining the predictions of multiple models to improve overall performance. The idea is that by combining different models, each contributing its own strengths and weaknesses, the ensemble can often achieve better results than any individual model.



***Figure:*** Accuracy using Ensemble coding

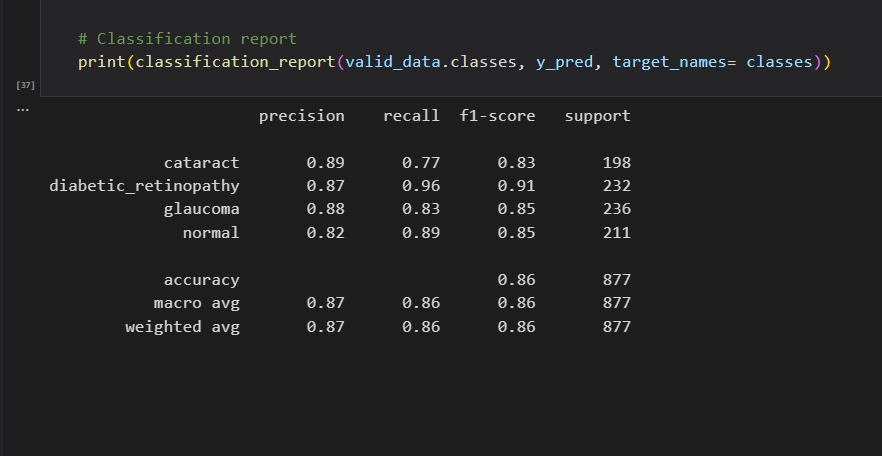
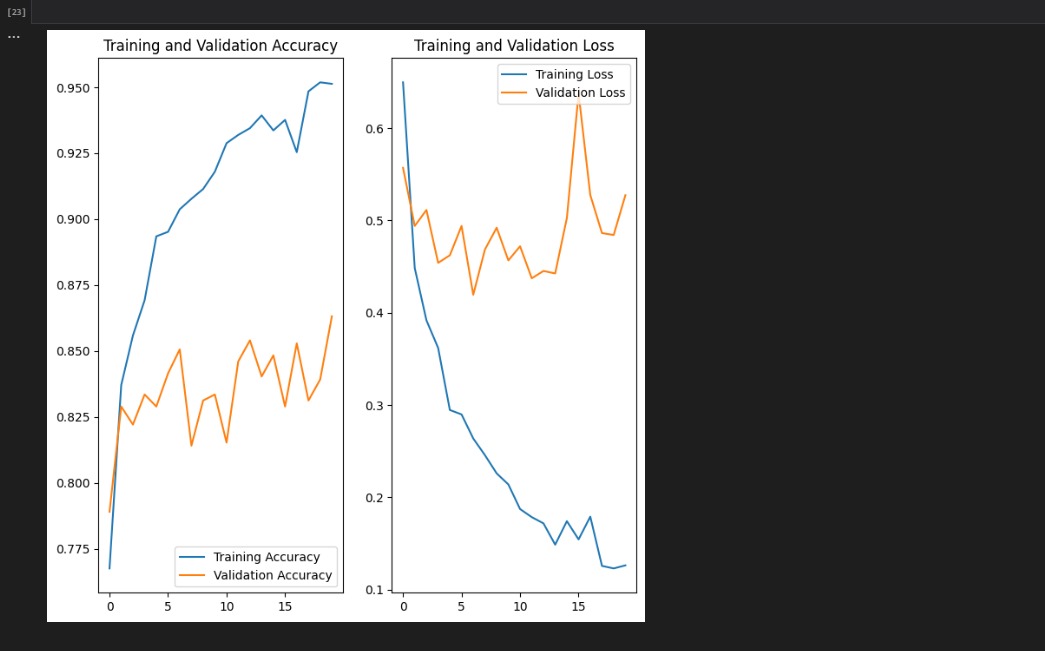
**2. DenseNet:** DenseNet is known for its densely connected structure, where each layer is connected to every other layer in a feedforward fashion. Unlike traditional convolutional neural networks (CNNs) where each layer takes input only from the preceding layer, DenseNet allows for direct connections between any two layers. Number of epochs were 15 and batch size was kept 32.

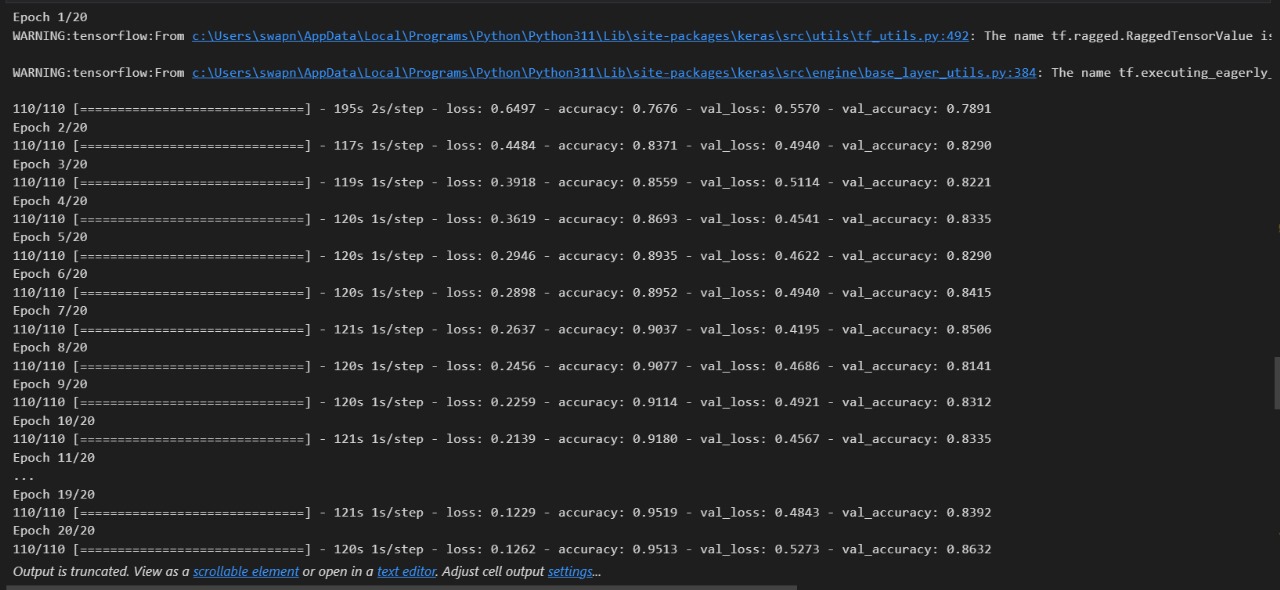




***Figures***: Images showing the classification report, Accuracy plots and final validation accuracy for Densenet

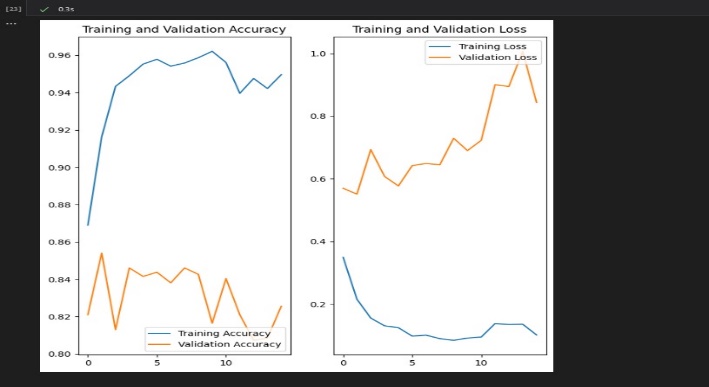
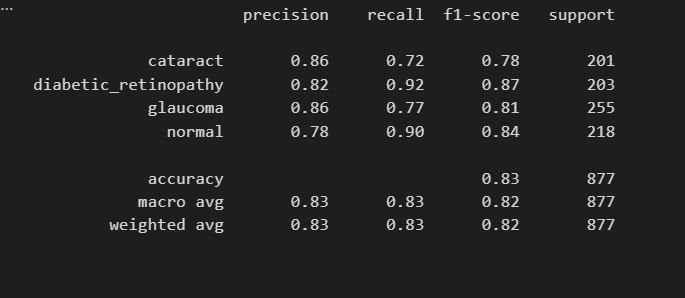
1. **ResNet:** ResNet is the use of residual blocks, which contain shortcut connections (also known as skip connections or identity mappings). These connections skip one or more layers in the network, allowing the model to learn residual functions. The main motivation behind using residual blocks is to address the degradation problem observed in very deep neural networks. We have implemented Resnet50 where 50 defines the depth of the network. Number of epochs 20 and batch size was kept 32.

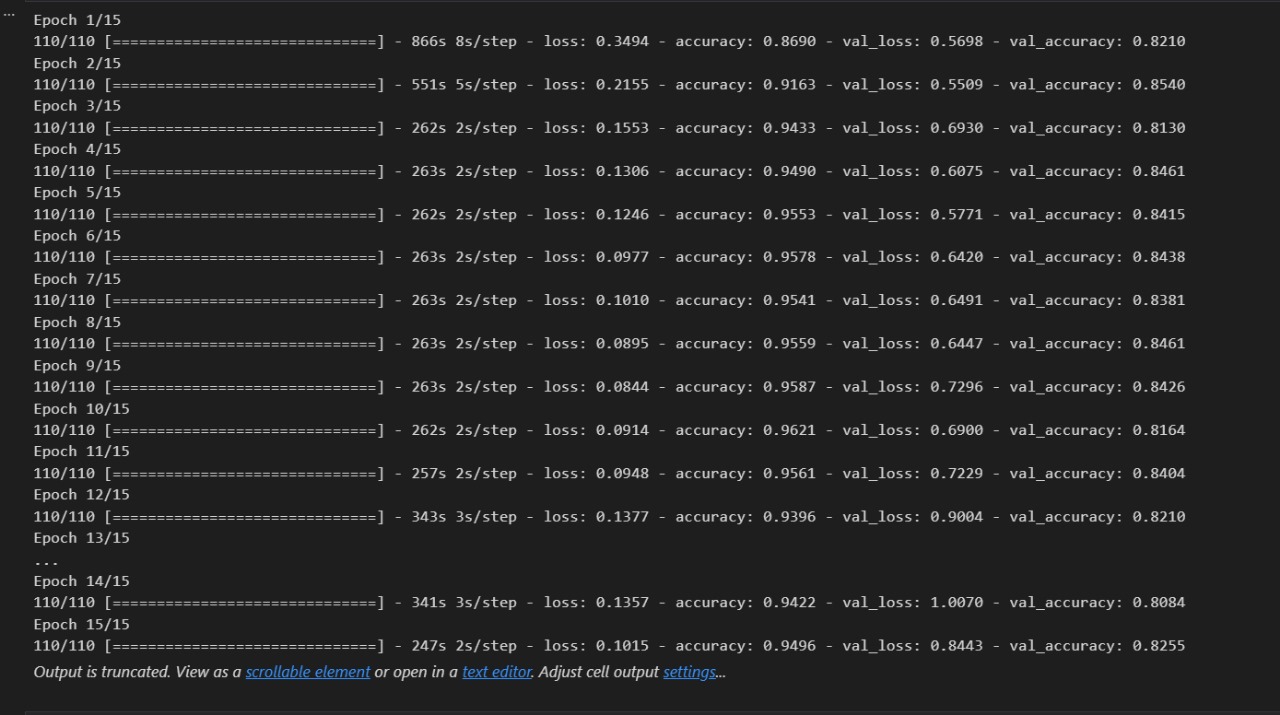
 



***Figures***: Images showing the classification report, Accuracy plots and final validation accuracy for Resnet

1. **VGG16:** VGG16 is like a picture detective that uses a set of rules to investigate each image. It breaks down the picture into small pieces and analyzes them to figure out what's in it. The "16" in VGG16 means it has 16 layers of these detective rules. Each layer looks at the picture in a bit more detail. It's a bit like having 16 layers of magnifying glasses, each helping the computer see more and more clearly. Number of epochs is 15 and batch is kept 32.

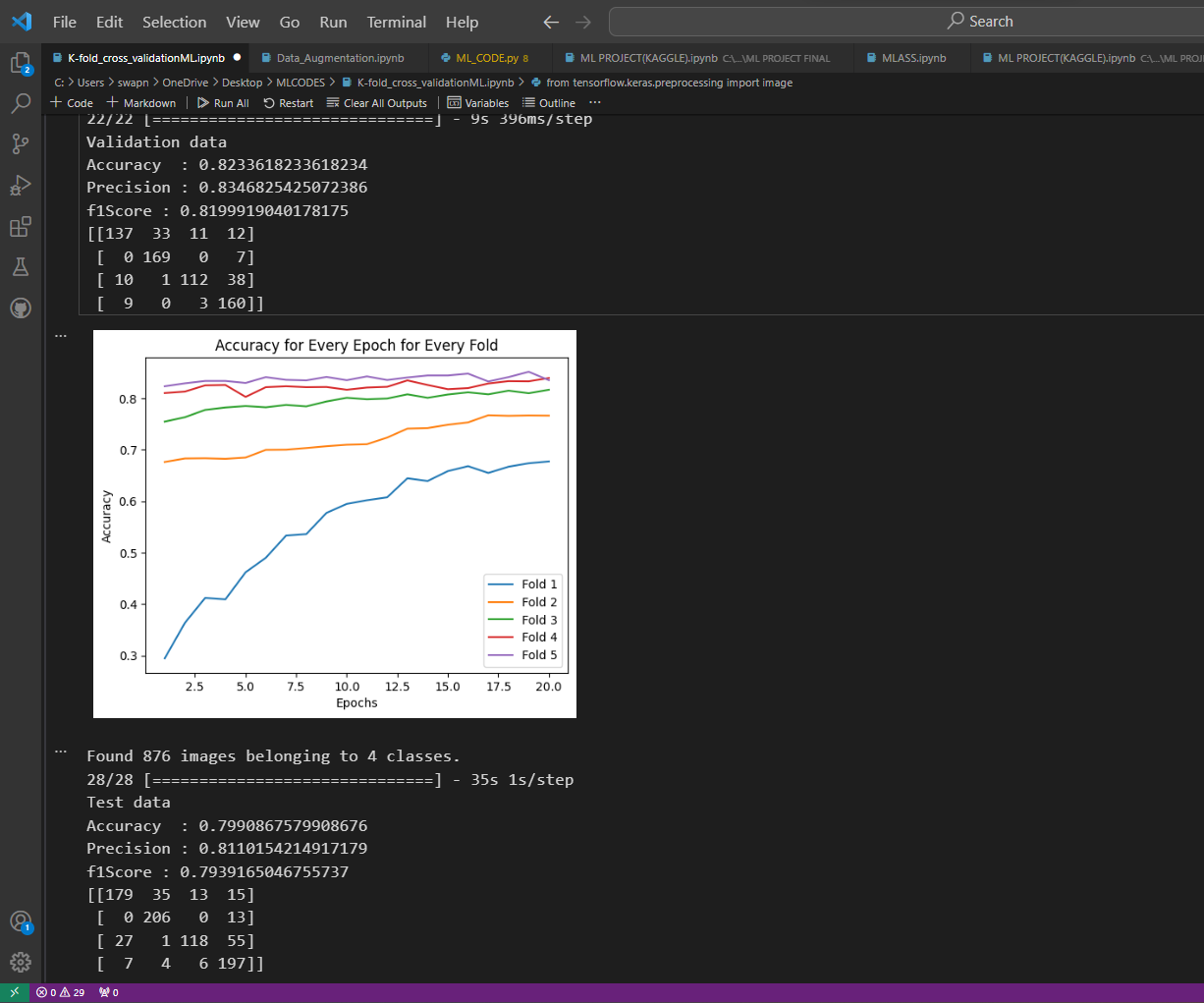




***Figures***: Images showing the classification report, Accuracy plots and final validation accuracy for VGG16

1. **OUR MODEL:** CNN made from scratch and validated using K-fold cross validation





***Figure:*** Showing the results for our own CNN model

**Contributions**

**Svea:** K-fold validation, Other models (Densenet, VGG16), documentation.

**Swapnil:** CNN Model, Plotting, Other models (Ensemble, Resnet50).

**References**

* Abràmoff MD, Garvin MK, Sonka M. Retinal imaging and image analysis. IEEE Rev Biomed Eng. 2010;3:169-208. doi: 10.1109/RBME.2010.2084567. PMID: 22275207; PMCID: PMC3131209.:

Link: [<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3131209/>]

* Article On Retinal OCT Images Using CNN By Tjhr Published on 11 July 2022:

[<https://medium.com/@tjhr123/classification-of-retinal-oct-images-using-cnn-fe70c07b2258>]

* Laith Alzubaidi, Jinglan Zhang, Amjad J. Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, J. Santamaría, Mohammed A. Fadhel, Muthana Al-Amidie & Laith Farhan| Review of deep learning: concepts, CNN architectures, challenges, applications, future directions

Link: [<https://link.springer.com/article/10.1007/s00530-021-00769-7>]

* Convolutional Neural Network Tutorial (CNN) | How CNN Works | Deep Learning Tutorial | Simplilearn:

[Youtube] Link : [<https://www.youtube.com/watch?v=Jy9-aGMB_TE>]