<u>AML Assignment 1 : Project Report - Sristi Bafna</u>

A. Preliminary details about the dataset:

The given dataset contained 2 separate sets of images:

1. Training and Test Images - 7000 training and 1000 test images of different dimensions
2. Preprocessed images - 6392 images

It also contained 2 csv files with details about the dataset:

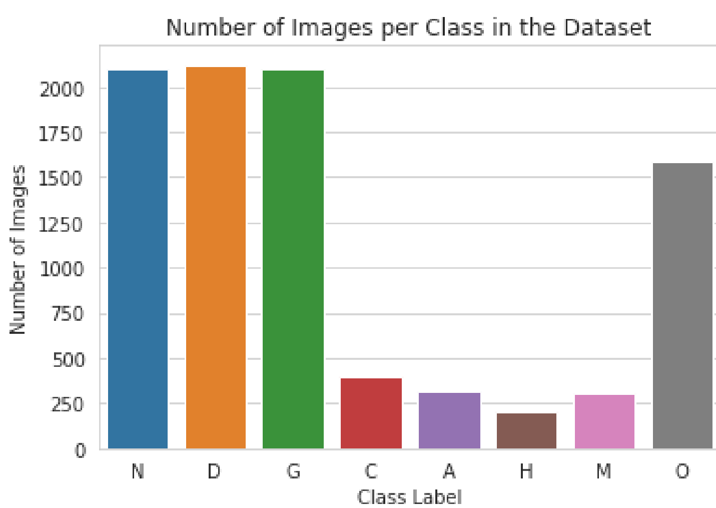
1. full_df.csv - contained the labels and targets of all the images, data about the patient (features such as age, gender, patient ID, diagnostic keywords) and names of all retinal pathology image files as well as their paths in the dataset.
2. data.xlsx - similar to full_df, did not contain file names and file paths.

B. Dataset Exploration:

Classes:

1. Normal (N)
2. Diabetes (D)
3. Glaucoma (G)
4. Cataract (C)
5. AMD (A)
6. Hypertension (H)
7. Myopia (M)
8. Other diseases/abnormalities (O)

The class distribution was as follows:



1. As we can see, the dataset is highly imbalanced. Most images are classified for normal, diabetes or glaucoma while classes like cataract, AMD and hypertension barely have any occurrences.
2. As is visible, the dataset has multiple labels and hence targets for certain images due to the presence of more than one condition.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	ID	Patient Age	Patient Sex	Left-Fundus	Right-Fundus	Left-Diagnostic Keywords	Right-Diagnostic Keywords	N	D	G	C	A	H	M	O
78	76	70	Male	76_left.jpg	76_right.jpg	retinochoroidal coloboma	retinochoroidal coloboma	0	0	0	0	0	0	0	1
79	77	67	Male	77_left.jpg	77_right.jpg	roliferative retinopathy，myelina	myelinated nerve fibers	0	1	0	0	0	0	0	1
80	78	46	Male	78_left.jpg	78_right.jpg	chorioretinal atrophy	normal fundus	0	0	0	0	0	0	0	1
81	79	72	Female	79_left.jpg	79_right.jpg	epiretinal membrane	ild nonproliferative retinopatl	0	1	0	0	0	0	0	1
82	80	50	Female	80_left.jpg	80_right.jpg	lens dust，normal fundus	s dust，myelinated nerve fib	0	0	0	0	0	0	0	1
83	81	66	Male	81_left.jpg	81_right.jpg	:e non proliferative retinopathy，	non proliferative retinopathy	0	1	0	1	0	0	0	0
84	82	33	Male	82_left.jpg	82_right.jpg	normal fundus	myelinated nerve fibers	0	0	0	0	0	0	0	1
85	83	61	Female	83_left.jpg	83_right.jpg	normal fundus	macular epiretinal membrane	0	0	0	0	0	0	0	1
86	84	51	Female	84_left.jpg	84_right.jpg	normal fundus	normal fundus	1	0	0	0	0	0	0	0
87	85	67	Female	85_left.jpg	85_right.jpg	normal fundus	drusen，atrophic change	0	0	0	0	0	0	0	1
88	86	56	Female	86_left.jpg	86_right.jpg	mild nonproliferative retinopathy	pathological myopia	0	1	0	0	0	0	1	0
89	87	41	Female	87_left.jpg	87_right.jpg	derate non proliferative retinopa	ild nonproliferative retinopatl	0	1	0	0	0	0	0	0
90	88	71	Female	88_left.jpg	88_right.jpg	normal fundus	macular epiretinal membrane	0	0	0	0	0	0	0	1
91	89	60	Female	89_left.jpg	89_right.jpg	retinitis pigmentosa	roliferative retinopathy，reti	0	1	0	0	0	0	0	1
92	90	53	Female	90_left.jpg	90_right.jpg	t，moderate non proliferative re	on proliferative retinopathy，	0	1	0	0	0	0	0	1
93	91	28	Male	91_left.jpg	91_right.jpg	normal fundus	myelinated nerve fibers	0	0	0	0	0	0	0	1
94	92	53	Male	92_left.jpg	92_right.jpg	normal fundus	retinitis pigmentosa	0	0	0	0	0	0	0	1

3. Another thing we notice is that there are 2 images per patient. This is because every eye has been uploaded as a separate image considering diagnoses of both eyes can be separate.

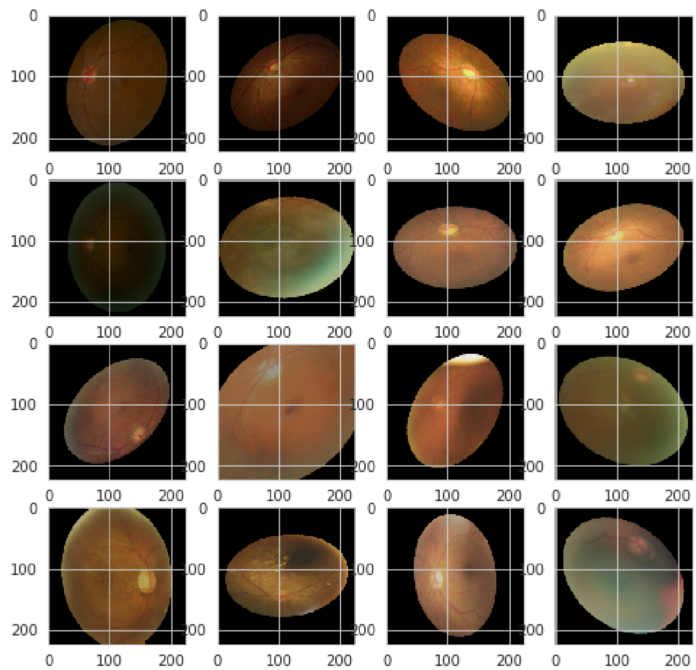
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	ID	Patient	Patient	Left-Fu	Right-F	Left-Dia	Right-D	N	D	G	C	A	H	M	O	filepath	labels	target	filenam
2	0	69	Female	0_left.jpg	0_right.jpg	cataract	normal fu	0	0	0	1	0	0	0	0	../input/o	['N']	[1, 0, 0, 0,	0_right.jpg
3	0	69	Female	0_left.jpg	0_right.jpg	cataract	normal fu	0	0	0	1	0	0	0	0	../input/o	['C']	[0, 0, 0, 1,	0_left.jpg
4	1	57	Male	1_left.jpg	1_right.jpg	normal fu	normal fu	1	0	0	0	0	0	0	0	../input/o	['N']	[1, 0, 0, 0,	1_right.jpg
5	1	57	Male	1_left.jpg	1_right.jpg	normal fu	normal fu	1	0	0	0	0	0	0	0	../input/o	['N']	[1, 0, 0, 0,	1_left.jpg
6	10	70	Male	10_left.jpg	10_right.jp	epiretinal	normal fu	0	0	0	0	0	0	0	1	../input/o	['N']	[1, 0, 0, 0,	10_right.jpg
7	10	70	Male	10_left.jpg	10_right.jp	epiretinal	normal fu	0	0	0	0	0	0	0	1	../input/o	['O']	[0, 0, 0, 0,	10_left.jpg
8	100	59	Male	100_left.jp	100_right.	macular e	normal fu	0	0	0	0	0	0	0	1	../input/o	['N']	[1, 0, 0, 0,	100_right.jpg

4. Images also have varying sizes. Dimensions are around 2048x1536 or 2976x2976 pixels for all images.

Since the file paths in the full_df.csv file are for the Training Images file, we use only that set of images for our model. We split the 6392 images into training, test and validation sets.

C. Data Preprocessing and Augmentation

1. We use the ImageDataGenerator class in TensorFlow for the data loading and augmentation. This class generates batches of augmented images according to our specified batch size and loads it into the deep learning model. We perform augmentations like rescaling, rotation, shearing, altering brightness range, flipping (horizontal and vertical). This is done to normalise the input data to a certain extent which helps bring all features to a similar scale. Data normalisation also helps reduce overfitting.



2. We only rescale the test dataset since we only perform augmentation on the training dataset to generate additional training examples which help the model's ability to generalise the data. Performing augmentations on the testing dataset would change the distribution of the data and give incorrect evaluations of the model on our dataset considering the test set is unseen.
3. I tried 2 different experiments with dataset augmentation to see the performance of the dataset:
    a. Smaller image size - Although for 3 experiments I worked with 224x224x3 as the image dimensions (according to the ImageNet standards), I tried reducing the size to 150x150x3 for one experiment to see the model's performance on the same.
    b. Working with binary classification - Since the task at hand is multiclass classification for 8 labels, I tried rearranging the dataset into just 2 classes:
        i. Normal and Others signifying the diagnosis was either pathologically normal or not.
        ii. Normal and Cataract signifying the diagnosis was either pathologically normal or the eye had cataract.

    The dataset for the latter was a reduced one and I ran it on a benchmark CNN (VGG19) to see its performance as the model's accuracy rate for that should be considerably higher than the other experiments given the augmentations performed. This was also to understand the kind of accuracy levels I would be working with so I could evaluate my model reasonably.

D. Our network

I experimented with 2 different kinds of convolutional neural networks:

1. **<u>CNN 1:</u>**
    a. Block 1:
        i. 2 sub-blocks. Each sub-block contains a Convolutional layer followed by Batch Normalisation and then ReLU Activation
        ii. Max pooling
        iii. Dropout
    b. Block 2:
        i. 2 sub-blocks. Each sub-block contains a Convolutional layer followed by Batch Normalisation and then ReLU Activation
        ii. Max pooling
        iii. Dropout
    c. Flattening
    d. Dense layer with Sigmoid activation

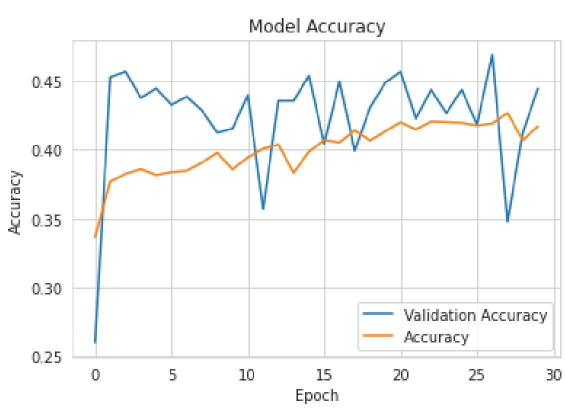
- The convolutional layer is followed by batch normalisation because the convolutional layer extracts features from the input image, and the batch normalisation layer normalises the activations of the convolutional layer. This is done to help reduce the internal covariate shift and improve the convergence of the network.
- ReLU Activation is done after the batch normalisation to introduce non-linearity into the network and also help create decision boundaries to separate our 8 different classes of inputs. Specifically, ReLU is used because it has a sparsity property that helps reduce the number of parameters in the network.
- Max pooling is done after it because it helps reduce the spatial size of the output volume, enabling more efficient learning by the network of the complex features and decision boundaries from the previous layers.
- Dropout layer is added to reduce overfitting so that the network learns to generalise better.
- This block is repeated to allow our CNN to learn increasingly complex and abstract features of the input data.
- Flattening converts the output from both blocks into a one-dimensional feature vector which can then be fed into a fully connected layer that learns to perform classification or regression on the input based on the extracted features by the CNN.
- Dense layer is used to map the final one-dimensional feature vector to classes and assign each a probability score between 0 and 1, representing the likelihood of each image belonging to a particular class.

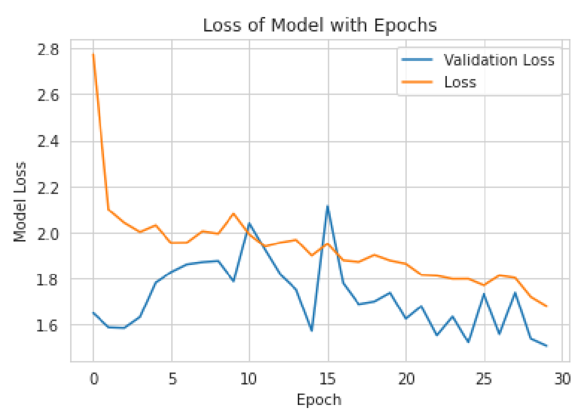
2. **CNN 2:**
    a. Block 1:
        i. 2 sub-blocks. Each sub-block contains a Convolutional layer followed by Batch Normalisation and then ReLU Activation
        ii. Convolutional layer
        iii. Dropout
    b. Block 2:
        i. 2 sub-blocks. Each sub-block contains a Convolutional layer followed by Batch Normalisation and then ReLU Activation
        ii. Convolutional layer
        iii. Dropout
    c. Flattening
    d. Dense layer with Sigmoid activation

The only difference between this CNN and the previous one is the removal of max pooling. Having 4 additional convolutional layers after ReLU activation helps increase the network's representational power while also facilitating it to learn more complex and hierarchical features from the input data.

E. Results

1. Base experiment : **CNN 1, Size - 224x224x3, 8-class Classification**



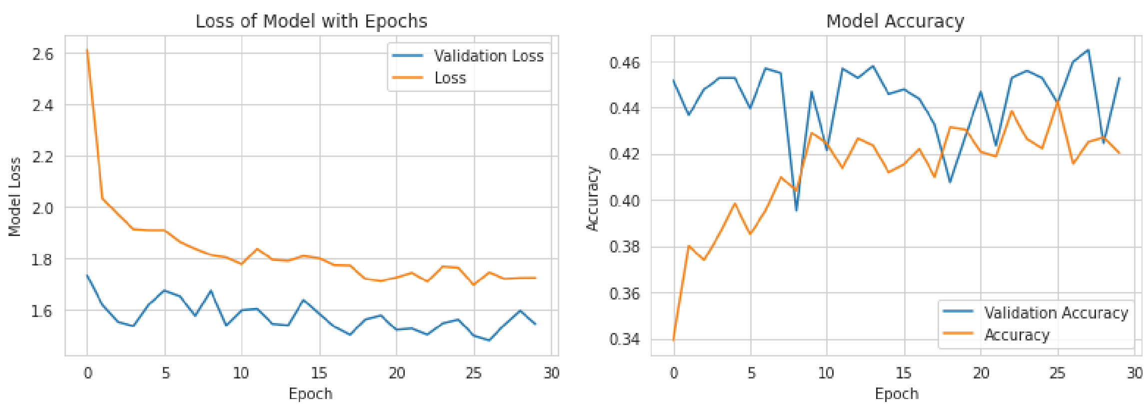


Even though the loss decreases over time and the accuracy increases with epochs, we can see that the model accuracy and loss fluctuates a lot. The final accuracy was 42% and final validation accuracy was 45%. This could be due to the following reasons:

- Learning rate value : I would like to experiment with different learning rates to find the optimal value for the model.

- Model Architecture : Possibly adding more blocks to ensure better learning of the features because it is possible that the model architecture is too simple for a dataset of 6000+ images and 8 classes which means the decision boundaries are also not too strong.
- Training data : The data at hand could be noisy implying that there is underlying variance and instability in the values given to the model itself which is why the learning is poor.

2. **CNN 1, Size - 150x150x3, 8-class Classification**



We can see that the model loss consistently decreases with epochs but the model accuracy, despite increasing with epochs, fluctuates a lot. The final accuracy was 42% and final validation accuracy was 45%. This can be because of these reasons:

- Inconsistency in data (similar to Experiment 1)
- This implies the model is probably overfitting too much and not generalising to the dataset. This can be rectified by the following methods:
  - More dropout layers
  - More data augmentation. Since the model's statistics are quite similar to the previous experiment, it implies that the reduction in image size did not impact the model's performance, indicating that other avenues of data augmentation need to be looked into.
  - Simplifying model architecture

It was not possible to generate a confusion matrix due to the limits of colab after having used it for 5 hours per experiment but looking into which classes had poor decision boundaries would also help realise the drawbacks of the dataset.

3. **CNN 1, Size - 224x224x3, Binary Classification**

It was not possible to finish this experiment because of the poor WiFi quality on campus. Additionally, Google Colab's limits on GPU usage (despite me creating 2 new Gmail accounts for the same) owing to the large size of the dataset and the number of epochs, every experiment took 5+ hours to run. The final model accuracy was 56% and validation accuracy was 54% which was higher compared to the previous experiments with 8 classes implying reducing the number of classes probably better defined the model's decision boundaries.

```
Epoch 1/30
127/127 [==============================] - 750s 6s/step - loss: 1.0690 - accuracy: 0.5138 - precision_1: 0.5122 - recall_1: 0.5111 - val_loss: 0.7117 - val_accuracy: 0.4536 - val_precision_1: 0.4540 - val_recall_1: 0.4526
Epoch 2/30
127/127 [==============================] - 348s 3s/step - loss: 0.8792 - accuracy: 0.5172 - precision_1: 0.5155 - recall_1: 0.5155 - val_loss: 0.7192 - val_accuracy: 0.4617 - val_precision_1: 0.4596 - val_recall_1: 0.4587
Epoch 3/30
127/127 [==============================] - 391s 3s/step - loss: 0.8469 - accuracy: 0.5261 - precision_1: 0.5197 - recall_1: 0.5219 - val_loss: 0.6962 - val_accuracy: 0.5353 - val_precision_1: 0.5288 - val_recall_1: 0.5454
Epoch 4/30
127/127 [==============================] - 346s 3s/step - loss: 0.8305 - accuracy: 0.5323 - precision_1: 0.5200 - recall_1: 0.5264 - val_loss: 0.6991 - val_accuracy: 0.5605 - val_precision_1: 0.5577 - val_recall_1: 0.5746
Epoch 5/30
127/127 [==============================] - 340s 3s/step - loss: 0.8141 - accuracy: 0.5271 - precision_1: 0.5239 - recall_1: 0.5271 - val_loss: 0.7504 - val_accuracy: 0.5282 - val_precision_1: 0.5287 - val_recall_1: 0.5292
Epoch 6/30
127/127 [==============================] - 348s 3s/step - loss: 0.8290 - accuracy: 0.5200 - precision_1: 0.5200 - recall_1: 0.5158 - val_loss: 0.7357 - val_accuracy: 0.5292 - val_precision_1: 0.5310 - val_recall_1: 0.5605
Epoch 7/30
127/127 [==============================] - 393s 3s/step - loss: 0.8184 - accuracy: 0.5256 - precision_1: 0.5238 - recall_1: 0.5268 - val_loss: 0.7565 - val_accuracy: 0.5504 - val_precision_1: 0.5540 - val_recall_1: 0.5736
Epoch 8/30
127/127 [==============================] - 343s 3s/step - loss: 0.8231 - accuracy: 0.5246 - precision_1: 0.5214 - recall_1: 0.5111 - val_loss: 0.7380 - val_accuracy: 0.5423 - val_precision_1: 0.5392 - val_recall_1: 0.5615
Epoch 9/30
127/127 [==============================] - 338s 3s/step - loss: 0.7982 - accuracy: 0.5502 - precision_1: 0.5358 - recall_1: 0.5441 - val_loss: 0.8173 - val_accuracy: 0.4990 - val_precision_1: 0.4950 - val_recall_1: 0.4970
Epoch 10/30
127/127 [==============================] - 345s 3s/step - loss: 0.8125 - accuracy: 0.5246 - precision_1: 0.5186 - recall_1: 0.5103 - val_loss: 0.7170 - val_accuracy: 0.5665 - val_precision_1: 0.5600 - val_recall_1: 0.5554
Epoch 11/30
127/127 [==============================] - 342s 3s/step - loss: 0.7911 - accuracy: 0.5411 - precision_1: 0.5370 - recall_1: 0.5291 - val_loss: 0.7361 - val_accuracy: 0.5333 - val_precision_1: 0.5268 - val_recall_1: 0.6038
Epoch 12/30
127/127 [==============================] - 342s 3s/step - loss: 0.8210 - accuracy: 0.5234 - precision_1: 0.5246 - recall_1: 0.5172 - val_loss: 0.7434 - val_accuracy: 0.5766 - val_precision_1: 0.5776 - val_recall_1: 0.6079
Epoch 13/30
127/127 [==============================] - 389s 3s/step - loss: 0.8365 - accuracy: 0.5229 - precision_1: 0.5218 - recall_1: 0.5195 - val_loss: 0.7097 - val_accuracy: 0.5786 - val_precision_1: 0.5868 - val_recall_1: 0.5010
Epoch 14/30
127/127 [==============================] - 342s 3s/step - loss: 0.7756 - accuracy: 0.5419 - precision_1: 0.5323 - recall_1: 0.5254 - val_loss: 0.7179 - val_accuracy: 0.5534 - val_precision_1: 0.5481 - val_recall_1: 0.5978
Epoch 15/30
127/127 [==============================] - 358s 3s/step - loss: 0.7884 - accuracy: 0.5313 - precision_1: 0.5326 - recall_1: 0.5352 - val_loss: 0.6818 - val_accuracy: 0.5796 - val_precision_1: 0.5662 - val_recall_1: 0.6895
Epoch 16/30
127/127 [==============================] - 348s 3s/step - loss: 0.7807 - accuracy: 0.5557 - precision_1: 0.5551 - recall_1: 0.5559 - val_loss: 0.7433 - val_accuracy: 0.5030 - val_precision_1: 0.5127 - val_recall_1: 0.3649
Epoch 17/30
127/127 [==============================] - 347s 3s/step - loss: 0.8156 - accuracy: 0.5320 - precision_1: 0.5239 - recall_1: 0.5182 - val_loss: 0.8417 - val_accuracy: 0.4627 - val_precision_1: 0.4639 - val_recall_1: 0.4657
Epoch 18/30
127/127 [==============================] - 341s 3s/step - loss: 0.8474 - accuracy: 0.5559 - precision_1: 0.5521 - recall_1: 0.5584 - val_loss: 0.8497 - val_accuracy: 0.5837 - val_precision_1: 0.5945 - val_recall_1: 0.5484
Epoch 19/30
127/127 [==============================] - 389s 3s/step - loss: 0.8282 - accuracy: 0.5369 - precision_1: 0.5354 - recall_1: 0.5384 - val_loss: 0.7207 - val_accuracy: 0.5554 - val_precision_1: 0.5501 - val_recall_1: 0.5645
Epoch 20/30
127/127 [==============================] - 339s 3s/step - loss: 0.7852 - accuracy: 0.5537 - precision_1: 0.5515 - recall_1: 0.5443 - val_loss: 0.7774 - val_accuracy: 0.5484 - val_precision_1: 0.5483 - val_recall_1: 0.5494
Epoch 21/30
127/127 [==============================] - 338s 3s/step - loss: 0.8212 - accuracy: 0.5335 - precision_1: 0.5333 - recall_1: 0.5259 - val_loss: 0.7320 - val_accuracy: 0.5917 - val_precision_1: 0.5857 - val_recall_1: 0.5514
Epoch 22/30
127/127 [==============================] - 394s 3s/step - loss: 0.7874 - accuracy: 0.5520 - precision_1: 0.5491 - recall_1: 0.5507 - val_loss: 0.7047 - val_accuracy: 0.5383 - val_precision_1: 0.5377 - val_recall_1: 0.5242
Epoch 23/30
127/127 [==============================] - 388s 3s/step - loss: 0.7877 - accuracy: 0.5411 - precision_1: 0.5429 - recall_1: 0.5480 - val_loss: 0.7390 - val_accuracy: 0.5524 - val_precision_1: 0.5576 - val_recall_1: 0.5565
Epoch 24/30
127/127 [==============================] - 337s 3s/step - loss: 0.7794 - accuracy: 0.5552 - precision_1: 0.5535 - recall_1: 0.5527 - val_loss: 0.7632 - val_accuracy: 0.5575 - val_precision_1: 0.5582 - val_recall_1: 0.6139
Epoch 25/30
127/127 [==============================] - 335s 3s/step - loss: 0.8332 - accuracy: 0.5404 - precision_1: 0.5401 - recall_1: 0.5446 - val_loss: 0.8023 - val_accuracy: 0.4718 - val_precision_1: 0.4739 - val_recall_1: 0.4486
Epoch 26/30
127/127 [==============================] - 384s 3s/step - loss: 0.8151 - accuracy: 0.5313 - precision_1: 0.5299 - recall_1: 0.5303 - val_loss: 0.7119 - val_accuracy: 0.5000 - val_precision_1: 0.5010 - val_recall_1: 0.4899
Epoch 27/30
127/127 [==============================] - 386s 3s/step - loss: 0.8100 - accuracy: 0.5362 - precision_1: 0.5342 - recall_1: 0.5333 - val_loss: 0.7033 - val_accuracy: 0.5494 - val_precision_1: 0.5233 - val_recall_1: 0.6683
Epoch 28/30
 85/127 [==================>...........] - ETA: 1:17 - loss: 0.7871 - accuracy: 0.5578 - precision_1: 0.5484 - recall_1: 0.5464
```

4. **CNN 2, Size - 224x224x3, 8-class Classification**

It was not possible to finish this experiment either because of reasons stated above as the runtime got disconnected only when 4 epochs were remaining. However, we see that by removing max pooling and having more convolutional layers to encourage better feature learning, the model shows lower final accuracy (34%) and validation accuracy (28%). This shows that max pooling layers are good for our dataset and perhaps the additional convolutional layers could be added in addition to max pooling. Considering our task at hand is multiclass classification (8 classes), max pooling is extremely crucial for reducing output volume as well as better feature representation. It also aids in better decision boundaries as it is evident that in its absence the decision boundaries were poorer leading to worse model accuracy and validation accuracy.

```
Epoch 1/30
127/127 [==============================] - 339s 2s/step - loss: 6.6150 - accuracy: 0.3223 - precision_2: 0.2015 - recall_2: 0.7967 - val_loss: 1.7607 - val_accuracy: 0.4466 - val_precision_2: 0.2148 - val_recall_2: 0.8589
Epoch 2/30
127/127 [==============================] - 329s 3s/step - loss: 3.0656 - accuracy: 0.3642 - precision_2: 0.2215 - recall_2: 0.8109 - val_loss: 1.7567 - val_accuracy: 0.4546 - val_precision_2: 0.2540 - val_recall_2: 0.8417
Epoch 3/30
127/127 [==============================] - 328s 3s/step - loss: 2.7480 - accuracy: 0.3775 - precision_2: 0.2166 - recall_2: 0.8267 - val_loss: 2.1356 - val_accuracy: 0.1673 - val_precision_2: 0.2446 - val_recall_2: 0.8276
Epoch 4/30
127/127 [==============================] - 338s 3s/step - loss: 3.0127 - accuracy: 0.3818 - precision_2: 0.2182 - recall_2: 0.8061 - val_loss: 2.2245 - val_accuracy: 0.4365 - val_precision_2: 0.1848 - val_recall_2: 0.8972
Epoch 5/30
127/127 [==============================] - 341s 3s/step - loss: 3.7281 - accuracy: 0.3594 - precision_2: 0.2138 - recall_2: 0.8002 - val_loss: 3.3103 - val_accuracy: 0.3770 - val_precision_2: 0.2340 - val_recall_2: 0.7994
Epoch 6/30
127/127 [==============================] - 337s 3s/step - loss: 5.5143 - accuracy: 0.3376 - precision_2: 0.2141 - recall_2: 0.7786 - val_loss: 6.2348 - val_accuracy: 0.1250 - val_precision_2: 0.1619 - val_recall_2: 0.8407
Epoch 7/30
127/127 [==============================] - 357s 3s/step - loss: 7.9932 - accuracy: 0.3042 - precision_2: 0.2097 - recall_2: 0.7531 - val_loss: 7.1733 - val_accuracy: 0.4234 - val_precision_2: 0.2674 - val_recall_2: 0.8125
Epoch 8/30
127/127 [==============================] - 352s 3s/step - loss: 6.8481 - accuracy: 0.3116 - precision_2: 0.2144 - recall_2: 0.7610 - val_loss: 4.3051 - val_accuracy: 0.3740 - val_precision_2: 0.1953 - val_recall_2: 0.8387
Epoch 9/30
127/127 [==============================] - 345s 3s/step - loss: 7.9920 - accuracy: 0.3084 - precision_2: 0.2156 - recall_2: 0.7702 - val_loss: 10.6729 - val_accuracy: 0.2490 - val_precision_2: 0.2144 - val_recall_2: 0.8488
Epoch 10/30
127/127 [==============================] - 352s 3s/step - loss: 8.1893 - accuracy: 0.3168 - precision_2: 0.2123 - recall_2: 0.7799 - val_loss: 10.4499 - val_accuracy: 0.2107 - val_precision_2: 0.1401 - val_recall_2: 0.6190
Epoch 11/30
127/127 [==============================] - 348s 3s/step - loss: 7.8085 - accuracy: 0.3245 - precision_2: 0.2127 - recall_2: 0.7794 - val_loss: 5.2263 - val_accuracy: 0.2228 - val_precision_2: 0.1896 - val_recall_2: 0.8286
Epoch 12/30
127/127 [==============================] - 391s 3s/step - loss: 8.3223 - accuracy: 0.3086 - precision_2: 0.2114 - recall_2: 0.7729 - val_loss: 11.0375 - val_accuracy: 0.4103 - val_precision_2: 0.2699 - val_recall_2: 0.7974
Epoch 13/30
127/127 [==============================] - 349s 3s/step - loss: 9.0062 - accuracy: 0.3205 - precision_2: 0.2145 - recall_2: 0.7667 - val_loss: 7.3459 - val_accuracy: 0.3639 - val_precision_2: 0.2267 - val_recall_2: 0.7833
Epoch 14/30
127/127 [==============================] - 325s 2s/step - loss: 10.1282 - accuracy: 0.2990 - precision_2: 0.2121 - recall_2: 0.7878 - val_loss: 12.7188 - val_accuracy: 0.4244 - val_precision_2: 0.2913 - val_recall_2: 0.7550
Epoch 15/30
127/127 [==============================] - 330s 3s/step - loss: 10.7713 - accuracy: 0.2945 - precision_2: 0.2036 - recall_2: 0.7558 - val_loss: 12.7674 - val_accuracy: 0.2520 - val_precision_2: 0.2408 - val_recall_2: 0.7974
Epoch 16/30
127/127 [==============================] - 386s 3s/step - loss: 7.7788 - accuracy: 0.3104 - precision_2: 0.2226 - recall_2: 0.7695 - val_loss: 8.2569 - val_accuracy: 0.3256 - val_precision_2: 0.2699 - val_recall_2: 0.7460
Epoch 17/30
127/127 [==============================] - 338s 3s/step - loss: 9.0372 - accuracy: 0.3240 - precision_2: 0.2119 - recall_2: 0.7573 - val_loss: 17.1194 - val_accuracy: 0.2308 - val_precision_2: 0.2055 - val_recall_2: 0.8377
Epoch 18/30
127/127 [==============================] - 332s 3s/step - loss: 11.4803 - accuracy: 0.3012 - precision_2: 0.2040 - recall_2: 0.7762 - val_loss: 10.0645 - val_accuracy: 0.1260 - val_precision_2: 0.1420 - val_recall_2: 0.5685
Epoch 19/30
127/127 [==============================] - 330s 3s/step - loss: 8.5798 - accuracy: 0.3205 - precision_2: 0.2094 - recall_2: 0.7467 - val_loss: 13.9465 - val_accuracy: 0.2661 - val_precision_2: 0.1842 - val_recall_2: 0.8770
Epoch 20/30
127/127 [==============================] - 320s 2s/step - loss: 7.6142 - accuracy: 0.3265 - precision_2: 0.2098 - recall_2: 0.7876 - val_loss: 5.3370 - val_accuracy: 0.2661 - val_precision_2: 0.2124 - val_recall_2: 0.8196
Epoch 21/30
127/127 [==============================] - 323s 2s/step - loss: 6.2328 - accuracy: 0.3161 - precision_2: 0.2115 - recall_2: 0.7729 - val_loss: 6.1165 - val_accuracy: 0.3790 - val_precision_2: 0.2863 - val_recall_2: 0.7399
Epoch 22/30
127/127 [==============================] - 326s 3s/step - loss: 3.1884 - accuracy: 0.3453 - precision_2: 0.2267 - recall_2: 0.7467 - val_loss: 2.3411 - val_accuracy: 0.1573 - val_precision_2: 0.1694 - val_recall_2: 0.7470
Epoch 23/30
127/127 [==============================] - 322s 2s/step - loss: 1.8056 - accuracy: 0.3924 - precision_2: 0.2434 - recall_2: 0.8004 - val_loss: 1.6480 - val_accuracy: 0.3982 - val_precision_2: 0.2597 - val_recall_2: 0.8397
Epoch 24/30
127/127 [==============================] - 321s 2s/step - loss: 1.6837 - accuracy: 0.4070 - precision_2: 0.2612 - recall_2: 0.7903 - val_loss: 1.6311 - val_accuracy: 0.4143 - val_precision_2: 0.2765 - val_recall_2: 0.8004
Epoch 25/30
127/127 [==============================] - 320s 2s/step - loss: 5.0309 - accuracy: 0.3498 - precision_2: 0.2238 - recall_2: 0.7908 - val_loss: 6.8365 - val_accuracy: 0.2812 - val_precision_2: 0.2676 - val_recall_2: 0.8185
Epoch 26/30
 48/127 [==========>...................] - ETA: 2:12 - loss: 3.2181 - accuracy: 0.3420 - precision_2: 0.1787 - recall_2: 0.8944
```

F. Discussion and Action Points

With regards to the benchmarking experiment (VGG19 on a reduced version of the dataset by performing classification for either Cataract or Normal diagnosis), the accuracy achieved was 79%. This implies that we need a more complex model architecture, more data augmentation and a lesser complex dataset to help with firmer decision boundaries for the classes. Future action points and experiments would involve exploring avenues mentioned as the possible causes for the drawbacks in the 4 experiments that were performed.