Diabetic Retinopathy Detection

A P Vaideeswaran

IIT Hyderabad

9 December 2023

Domain/Area of the Project

Computer Vision in Healthcare

Motivation/Relevance of the Project

Diabetic retinopathy (DR) is the leading cause of blindness in the working-age population. This project aims to detect and classify DR into four stages if the disease exists. This might help identify the condition and estimate the severity even before visiting a doctor. This is especially important as the disease barely presents any symptoms in the early stages. Early medical attention can prevent loss of eyesight or slow down the disease. Doctors can also use this model to keep track of the patient's condition without even a visit. DR detection is time-consuming and requires a lot of expertise and infrastructure. Therefore, this project would help doctors and patients alike, giving the health sector time to improve the infrastructure required to cater to the needs of an increasing diabetic population, especially in rural areas.

Description of The Dataset

The images have been labeled in the following format.

Patient 10:Left Eye -10_left

Patient 10:Right Eye -10_right

The upright fundus provides the anatomical retinal view, whereas the inverted fundus provides the retinal view through a microscopic condensing lens. Different cameras provide different views.

Upright right eye fundus

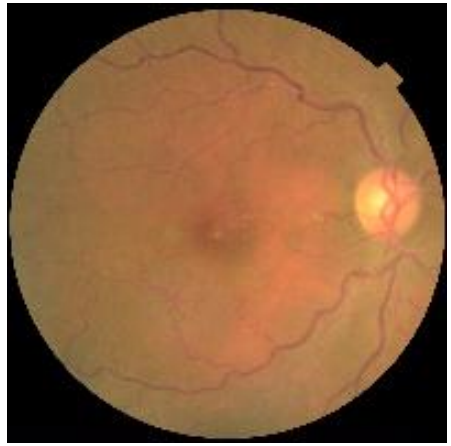
Inverted right eye fundus

Description of The Dataset

The following images correspond to the five stages of the disease.

No DR

Mild DR



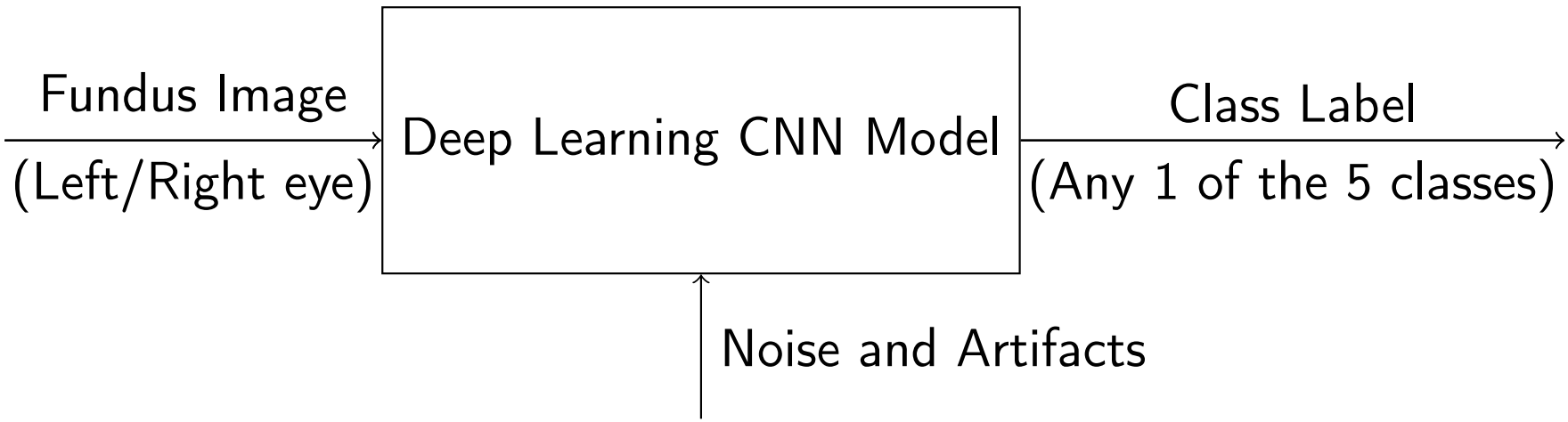
Moderate DR

Severe DR

Proliferative DR

Link to the dataset: `https://www.kaggle.com/competitions/diabetic-retinopathy-detection/data`

Specific Objectives/Goals of the Project



The model should be resilient to noise, artifacts, lack of focus, and inappropriate exposure.

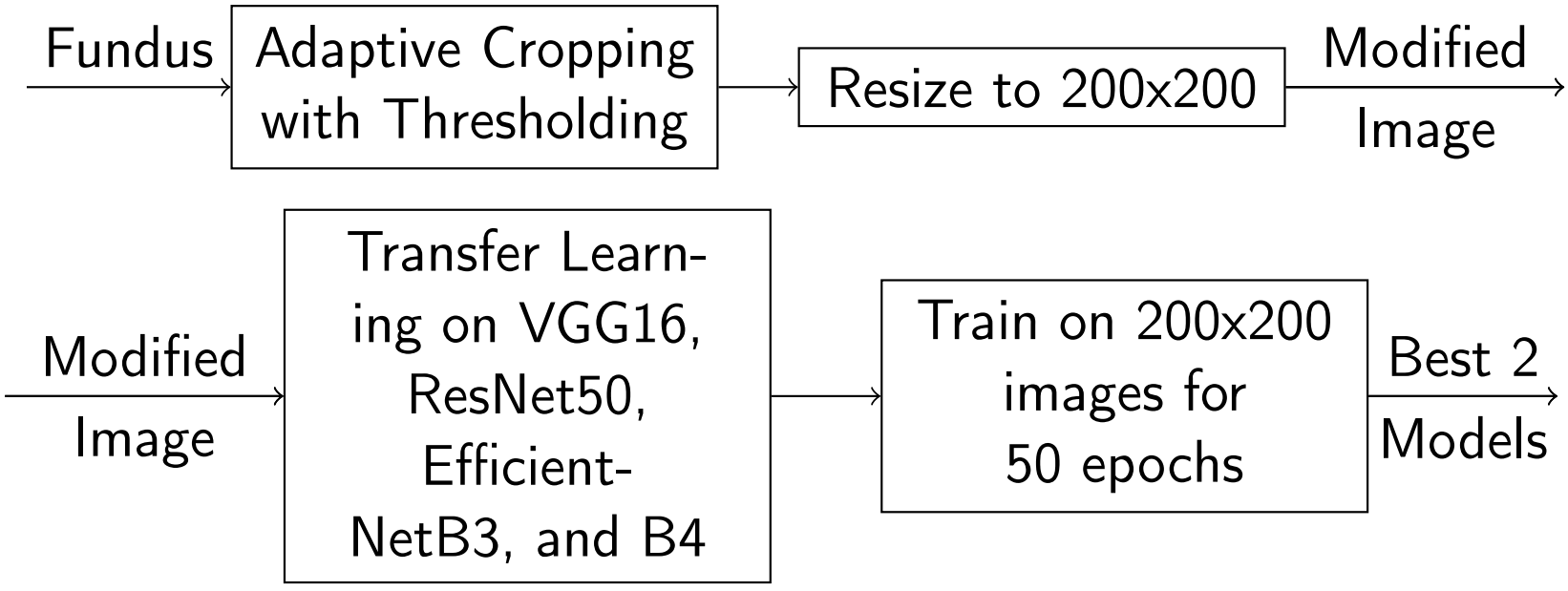
Challenges anticipated

- Effects of noise and artifacts, lack of focus, and inappropriate exposures.
- The dataset is huge, with several high-resolution (above 5000x5000) images with different aspect ratios.
- Pre-processing involving downsampling must be done such that key indicators like microaneurysms are NOT lost.
- We have a notion of distance between classes. Predicting a class 1 image as class 2 or class 4 is incorrect, with the latter being more incorrect. The quadratic kappa metric helps to quantify this.
- There is a severe class imbalance in the dataset with very few class 4 images compared to class 0.

How do you expect to meet those challenges?

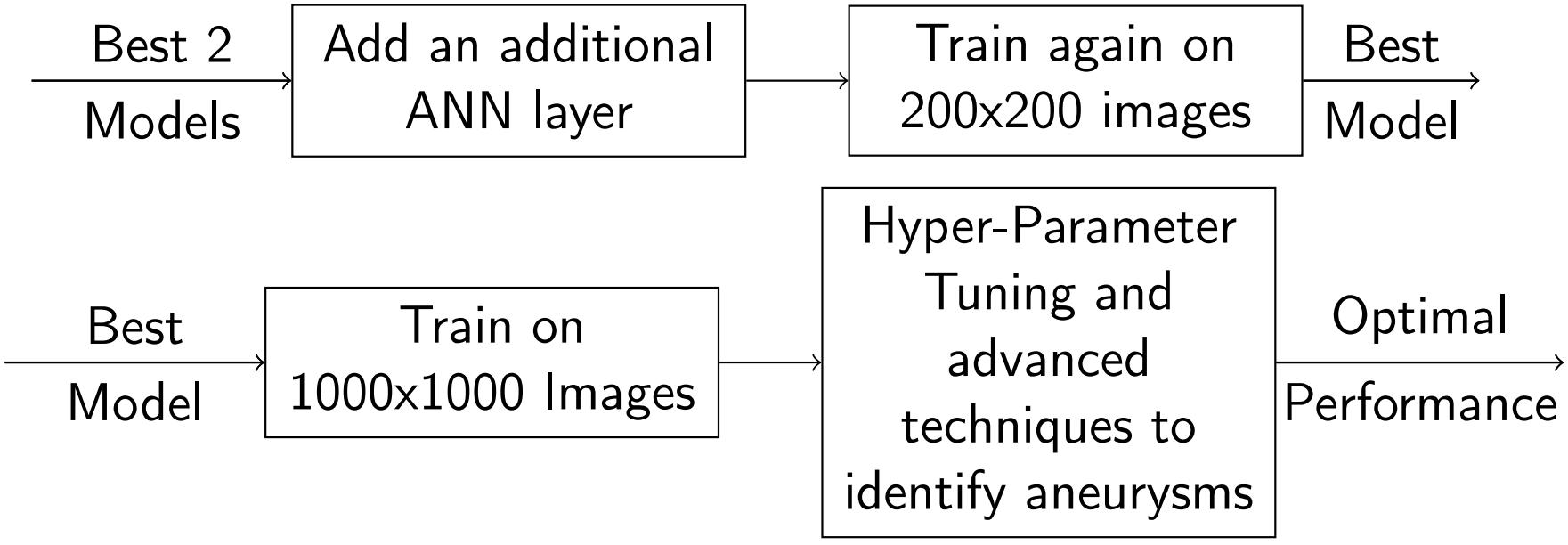
- While pre-processing, methods like histogram stretching and Gaussian blurring might improve image quality.
- FC-DenseNet can be used to identify the microaneurysms. The cross-entropy loss function of DenseNet can be replaced with the focal loss to weigh the aneurysms more than the background itself.
- A loss function that considers the distance between the classes has to be chosen.
- To deal with class imbalance (especially for stage 4), data augmentation, self-supervised learning, and focal loss can be employed.
- Multiple cores should be employed to deal with huge datasets efficiently. The following article was used for some initial insights on doing pre-processing:
  `https://www.nature.com/articles/s41598-021-04750-2`

Initial Approach to the Project



- **Adaptive cropping with thresholding:** If a column has more than 95 % (threshold) black pixels, that column can be removed.
- **Transfer Learning:** Add 16 nodes before the output layer, check initial performance and increase nodes if required to 32 or even 64 without training all the previous layers.

Initial Approach to the Project



- Only the best model shall be trained on the 1000x1000 images, which is very costly. Spending lots of time on a model without guaranteed performance would be wasteful.
- Hyper-parameter tuning includes increasing the number of epochs or increasing the width and depth of the ANN.

Dataset Description

Set	Class 0	Class 1	Class 2	Class 3	Class 4	Total
Train	5255	525	1121	172	127	7200
Val	411	39	107	24	19	600
Test	890	70	177	35	28	1800

Model Description and Training Paradigm

The Model Descriptions can be found in the following Links:

- VGG16
- ResNet50
- EfficientNetB3
- EfficientNetB4

- The Paradigm is clearly that of supervised learning. We have used a 12:1:3 split for the train: val: test split.
- Throughout, we shall use the categorical cross entropy as the loss function. It takes the form:

$$\text{Categorical-Cross Entropy} = -\log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right)$$ where s_p is the CNN score for the positive class.

Activation Functions

We have used the ReLU, ELU, and SoftMax Activation functions. Here are the expressions:

- ReLU Function: f(x) = max(0,x)
- Softmax Function for a given class s_i: $f(s_i) = \left(\frac{e^{sp}}{\sum_j^C e^{s_j}} \right)$

  Where s_j are the scores inferred for each class in C.
- ELU(x) = x if x is non-negative and ELU(x) = $\alpha(e^x - 1)$ for negative values of x.

Evaluation Metrics

We have used the following metrics to judge model performance:

- **MAE (Mean Absolute Error):** $\frac{\Sigma|y_i - \hat{y}_i|}{N}$ where y is the true label, $\hat{y}$ is the predicted label, and N is the size of the sample set.
- **Quadratic Weighted Kappa Metric:** 1- $\frac{\Sigma w_{i,j} O_{i,j}}{\Sigma w_{i,j} E_{i,j}}$ where $O_{i,j}$ corresponds to the number of i (actual) that received a predicted value j, $w_{i,j} = \frac{(i-j)^2}{(N-1)^2}$ An NxN matrix E, is calculated assuming that there is no correlation between values.
- **Accuracy:** $\frac{TP+TN}{TP+TN+FP+FN}$ where TP = True-Positive, TN = True-Negative, FP = False-Positive, and FN = False-Negative.
- **Precision:** $\frac{TP}{TP+FP}$
- **Recall:** $\frac{TP}{TP+FN}$
- **F1-Score:** $\frac{NetTP}{NetTP+0.5(FP+FN)}$

Note: Here, micro-averaging is done to make up for the class imbalance. We do NOT compute the above metrics class-wise but rather compute the aggregated values across all the classes.

Default Hyper-parameter Values

All the coding for this project has been done on TensorFlow. While performing hyper-parameter tuning, only some of the hyper-parameters were changed, leaving the others to their default values. Assume **default values** unless mentioned.

Hyper-Parameters TensorFlow	
Learning-Rate	0.001
$beta_1$	0.9
$beta_2$	0.999
epsilon	1e-7
amsgrad	False
weight_decay	None
clipnorm	None
clipvalue	None
global_clipnorm	None
use_ema	False
ema_momentum	0.99
ema_overwrite_frequency	None

Updates

The performance of all the models on the Test Images was evaluated before transfer learning. The following results were obtained:

Models - Before Training: MAE Scores			
VGG16	ResNet50	EfficientNetB3	EfficientNetB4
2.4172	1.7261	2.8894	2.4217

Transfer Learning Details

Transfer Learning Details	
Nodes in the Penultimate Layer	16
Activation Function	ReLU
Number of Output Nodes	5
Activation Function	SoftMax
Epochs	10
Loss Function	Categorical Cross Entropy
Optimizer	Adam
Batch Sisze	32
Learning Rate	0.001

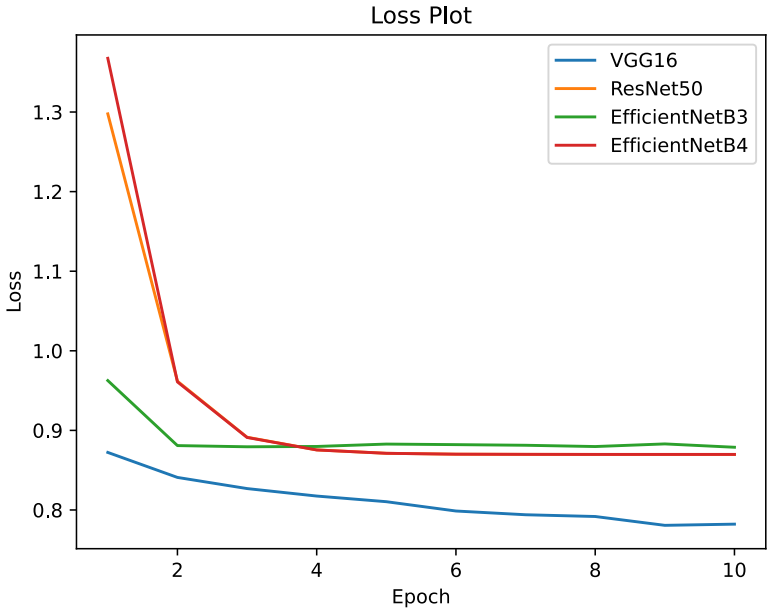
Weights of **all the models mentioned above** were based on classifying the ImageNet Dataset.

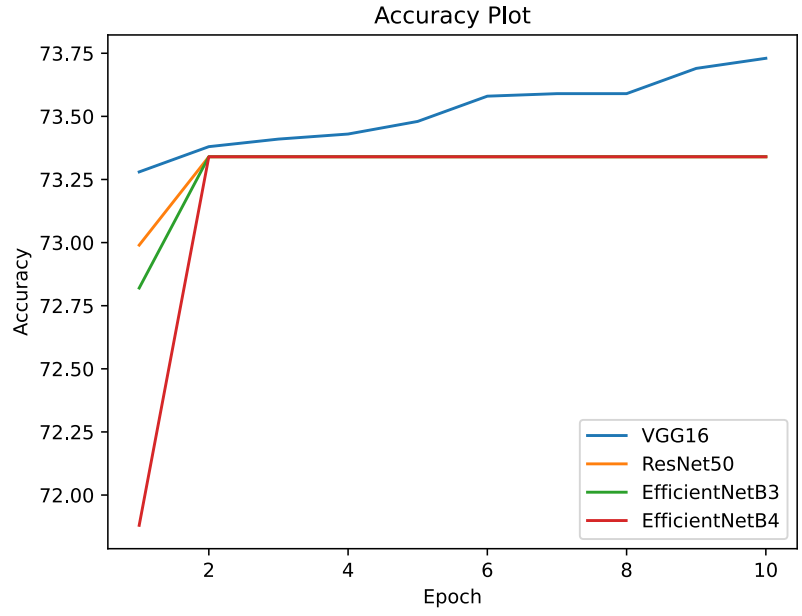
Comments

- More nodes in the penultimate layer weren't explored for VGG16, ResNet50, and EfficientNetB3 owing to their poorer test performance improvement relative to EfficientNetB4 (provided in the Results Slide).
- **Note:** The train accuracies aren't changing significantly for all the models. This could possibly be a limitation of Transfer Learning - only so much can be learned by training the last set of layers.
- **Note:** EfficientNetB4 might still have a good test performance despite having comparable accuracies to other models, as it might provide a "less" incorrect answer.

Loss and Accuracy Plots

The number of epochs wasn't increased from 10 to 50 as the accuracy and loss weren't significantly improving for the models.





Results

The performance of all the models on the Test Images was evaluated after transfer learning. The following results were obtained:

Models - After Transfer Learning: MAE Scores			
VGG16	ResNet50	EfficientNetB3	EfficientNetB4
1.0606	3.4211	1.6133	0.5789

Hence, the following change in MAE scores was observed - **EfficientNetB4** has improved the most - **Golden Bullet Model**:

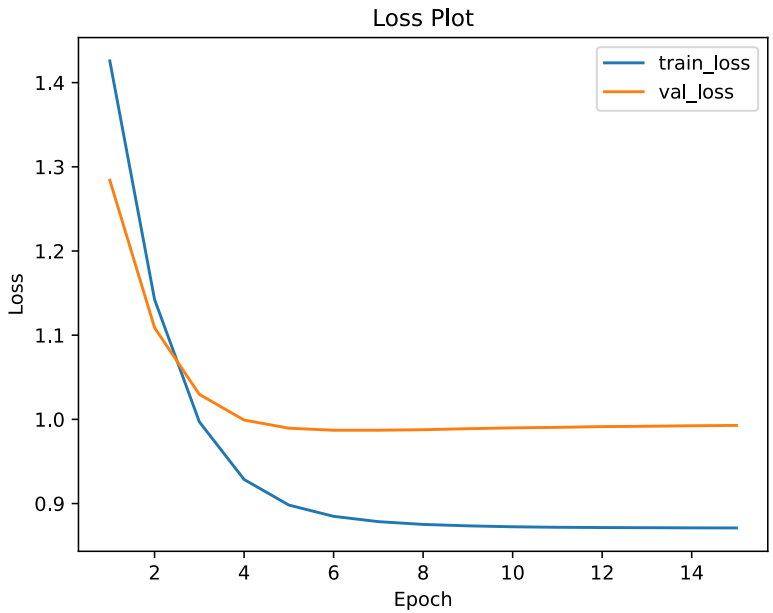
VGG16	ResNet50	EfficientNetB3	EfficientNetB4
1.3567	-1.695	1.2761	1.8428

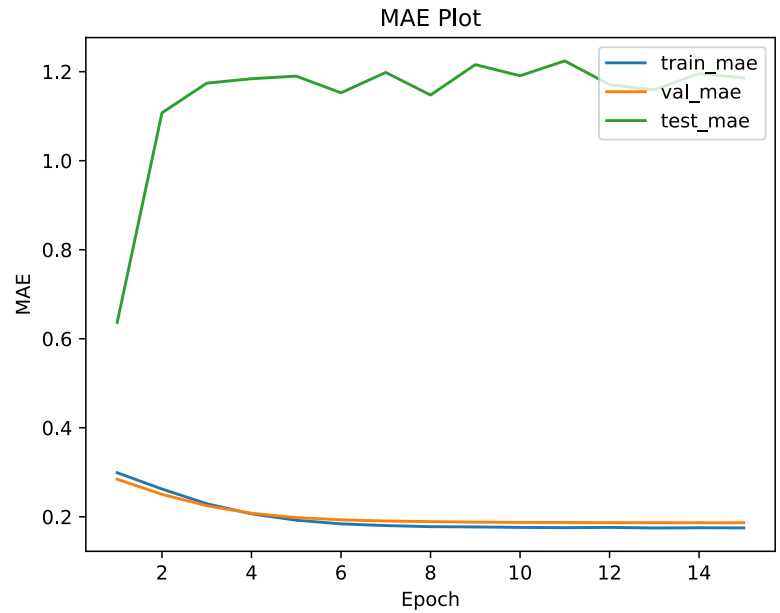
EfficientNetB4 - How to Improve Further?

- To make the model more suitable for this task, train all the model layers to learn the nuances of this dataset rather than **only** increasing the depth and width of the ANN.
- Since we are confident about the learning capacity of EfficienetNetB4, we can train the entire Network on 500x500 Images. These images would also provide the necessary resolution to classify the rarer and serious cases of the disease.

Experiment-1 with EfficientNetB4

The model was trained on 200x200 images for 15 epochs, keeping the learning rate at 0.0005 and batch size = 8. The number of nodes in the penultimate layer = 32.

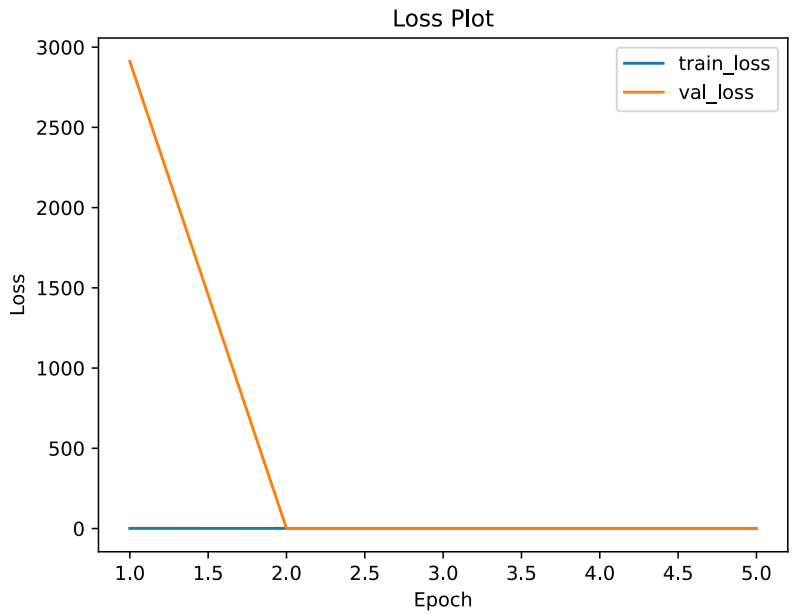


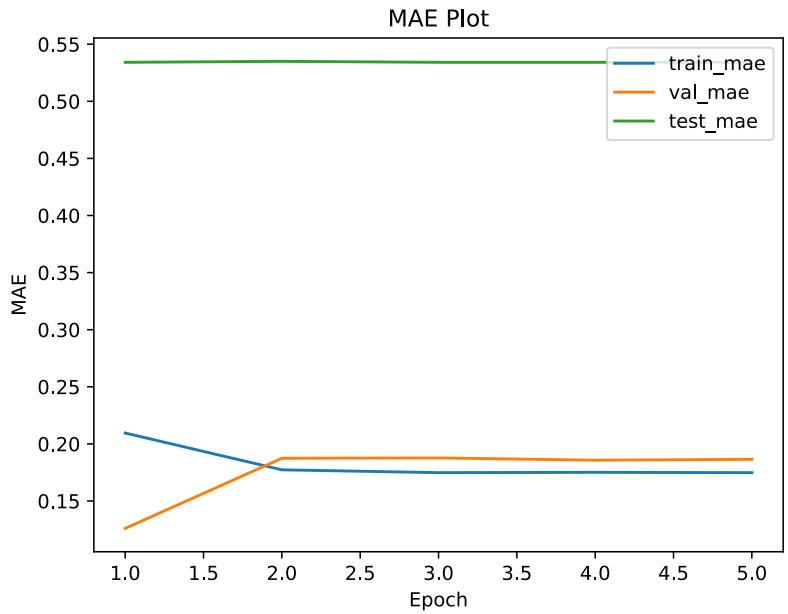


The test performance is best after the 1st epoch, with the prediction on average being around 0.64 classes apart from the true prediction.

Experiment-2

After the first epoch, the model weights were taken and trained for 5 epochs, keeping the learning rate at 0.005 and batch size = 16. Increasing the learning rate might help faster convergence while increasing the batch size prevents the model from learning noise.

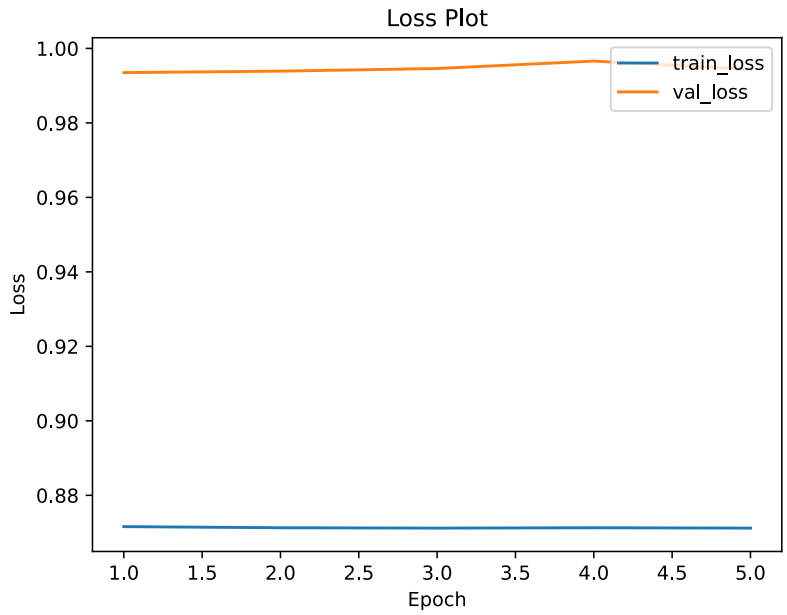


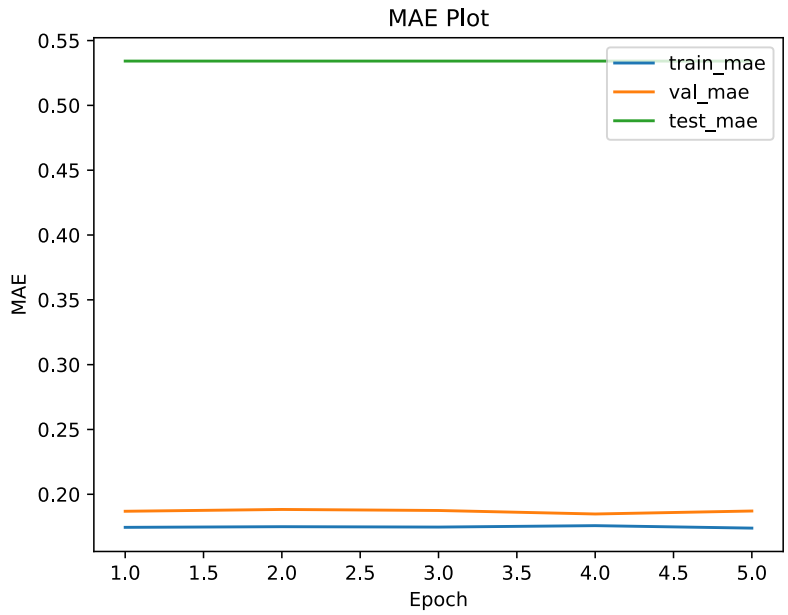


The test performance is better, with the prediction on average being around 0.53 classes apart from the true prediction. However, across epochs, this value does NOT improve.

Experiment-3

After the first epoch, the model weights were taken and trained for 5 epochs, keeping the learning rate at 0.005 and batch size = 32.





The test performance is the same as Experiment 2, with the prediction on average being around 0.53 classes apart from the true prediction. However, across epochs, this value does NOT improve.

Change In Evaluation Methodology

- MAE is a metric that is used **ONLY** for regression problems and **NOT** for classification problems.
- In this case, since we have a notion of distance between classes, it provides a rough idea of how off the prediction is.
- To get a clearer picture regarding how the model is classifying, we use accuracy, precision, recall, F1-Score, and confusion matrix.
- We use micro-averaging to compute the precision, recall, and F1-Score owing to the severe class imbalance.

New Results

- Accuracy = Precision = Recall = F1_Score = 0.7416666666666667 with the model classifying all the 1200 test images under class 0.
- **We conclude that the severe data imbalance in the training set is causing the model to predict all the images under class 0.**
- We also conclude that to say the model is doing any meaningful classification, the accuracy must be $>$ 0.7416666666666667.

Class Weighting

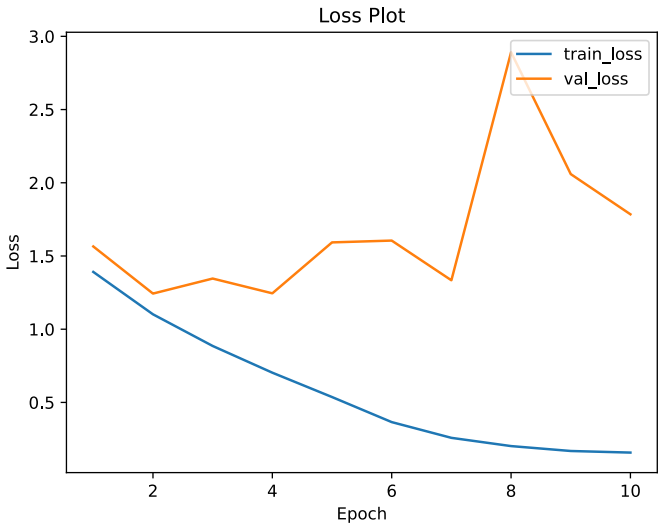
- The contributions of each class were weighed to mitigate the effects of class imbalance. All the parameters were kept identical to Experiment 3.
- However, this method proved significantly worse as the validation accuracies were close to 4%.

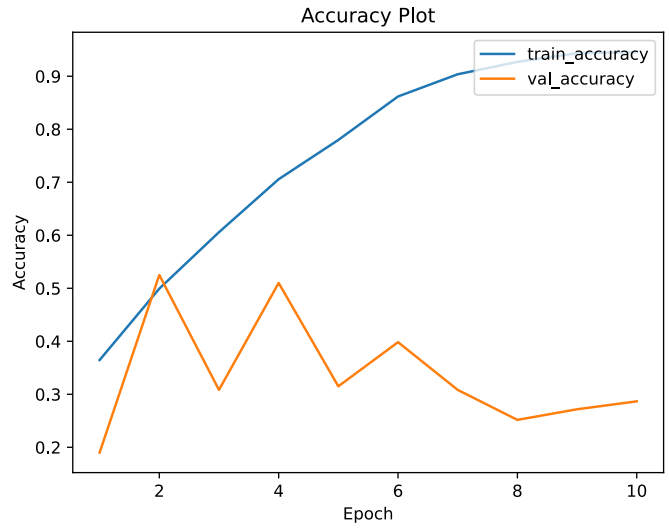
Rotational Augmentation

- To remove the class imbalance, a new augmented dataset was formed, comprising 1000 images each for classes 0,1 and 2, and 688 and 508 images for Classes 3 and 4, respectively.
- Augmentation involved rotating some of the images of the minority classes were rotated by angles of 90, 180, and 270 degrees.
- This method is also highly practicable as the fundus images taken in real life also vary in orientations. Moreover, CNNs are NOT rotational invariant.
- Augmentations involving cropping or changing contrast might be counter-productive as the information regarding the aneurysms might be lost.
- The validation and test sets were left untouched.

Rotational Augmentation: Experiment-1

The model was trained on 200x200 images for 10 epochs, keeping the learning rate at 0.0001 and batch size = 16. The number of nodes in the penultimate layer = 64. The weight decay was set to 0.005.





The test performance was only checked on the model with the best validation accuracy. This is the model weights obtained at the end of the second epoch.

Rotational Augmentation: Experiment-1

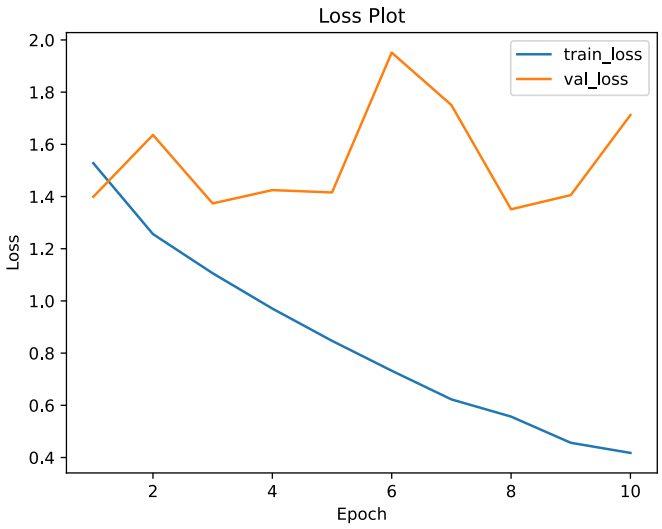
- It was observed that the Accuracy = Precision = F1_Score = Recall = 74%.
- Confusion Matrix:

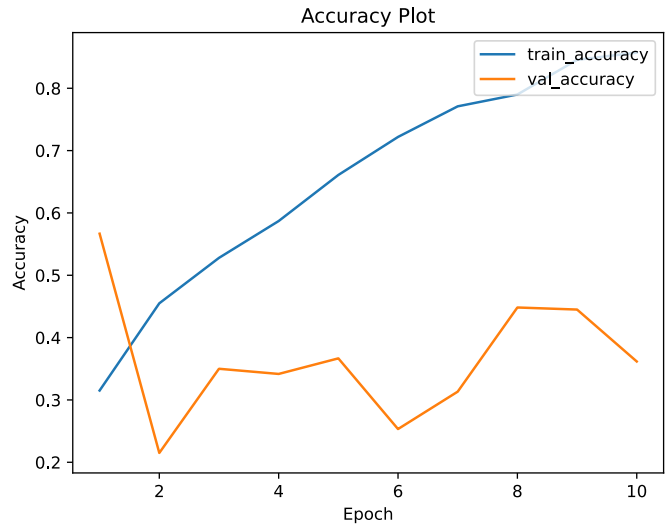
888	0	1	1	0
70	0	0	0	0
176	1	0	0	0
35	0	0	0	0
28	0	0	0	0

- The models at the earlier epochs have better validation accuracies simply by predicting a large number of zeros and not by actually learning the patterns.
- The models obtained at later epochs have much lower validation accuracies and very high training accuracies, as they might be over-fitting.

Rotational Augmentation: Experiment-2

All the hyper-parameters were exactly kept the same. Only the learning rate was reduced to 1e-05.





The test performance was only checked on the model with the best validation accuracy. This is the model weights obtained at the end of the first epoch.

Rotational Augmentation: Experiment-2

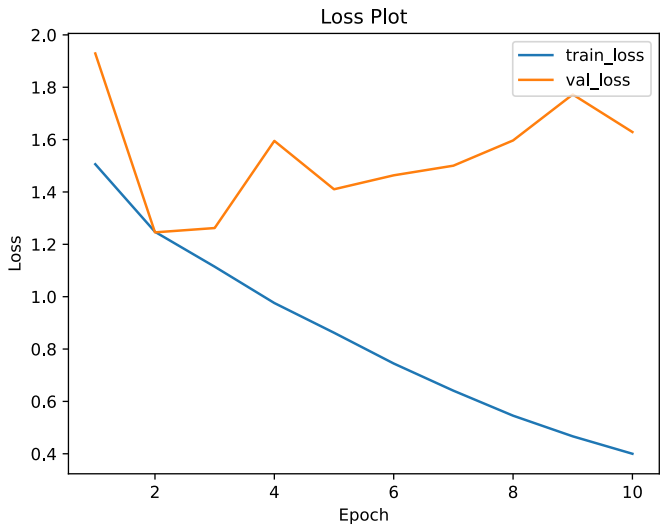
- It was observed that the Accuracy = Precision = F1_Score = Recall = 24.67%.
- Confusion Matrix:

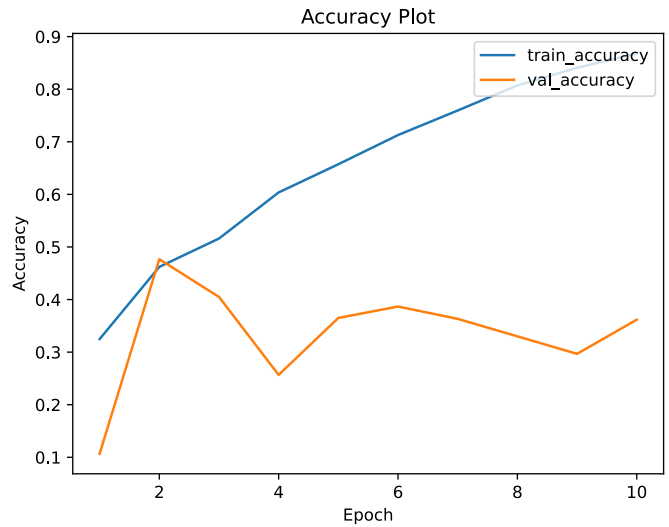
177	109	436	94	74
15	11	35	6	3
33	17	102	14	11
10	3	15	4	3
9	4	12	1	2

- The models at the earlier epochs have better validation accuracies simply by predicting a large number of zeros and twos (the two dominant classes) and not by actually learning the patterns.
- The models obtained at later epochs have much lower validation accuracies and very high training accuracies, as they might be over-fitting.

Rotational Augmentation: Experiment-3

All the hyper-parameters were exactly kept the same. Only the weight decay was increased to 0.05. This should help reduce over-fitting.





The test performance was only checked on the model with the best validation accuracy. This is the model weights obtained at the end of the second epoch.

Rotational Augmentation: Experiment-3

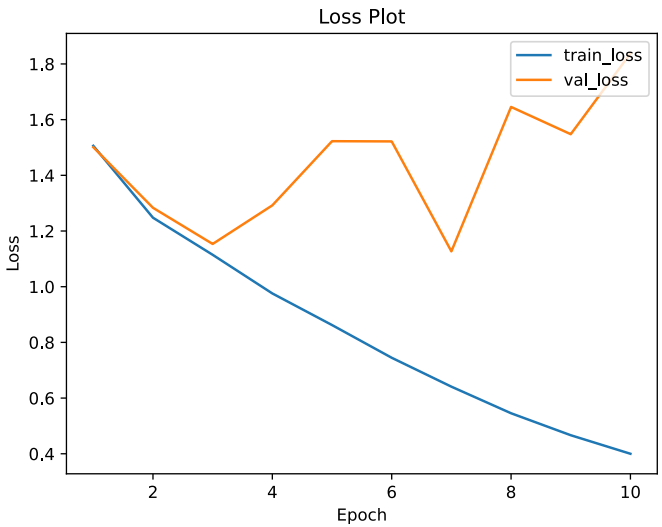
- It was observed that the Accuracy = Precision = F1_Score = Recall = 7.25%.
- Confusion Matrix:

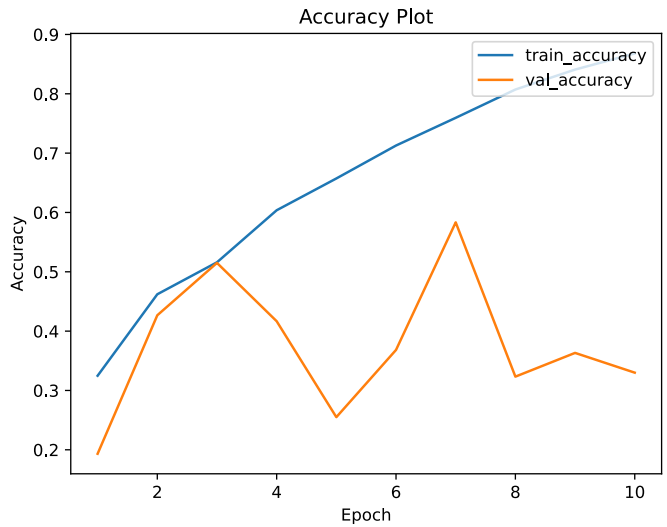
22	658	2	178	30
3	55	0	11	1
5	120	2	46	4
1	26	0	7	1
1	19	0	7	1

- Increasing the regularization parameter has caused the model to predict more of the non-dominant classes. Naturally, the test accuracies have fallen rapidly as most of the images present in the test set belong to the dominant classes.
- The models obtained at later epochs have much lower validation accuracies and very high training accuracies, as they might be over-fitting.

Rotational Augmentation: Experiment-4

The number of nodes in the penultimate layer was altered to 48. The weight decay was set at 0.5. The training was done for 10 epochs.





The test performances were checked on the models with the validation accuracies that are local maximas. This corresponds to the ends of the first, second, third, and seventh epochs.

Rotational Augmentation: Experiment-4

- **1st Epoch:** It was observed that the Accuracy = Precision = F1_Score = Recall = 16.92%.
- Confusion Matrix:

57	73	681	69	10
5	7	52	6	0
14	15	135	11	2
2	2	27	3	1
1	2	20	4	1

- **2nd Epoch:** It was observed that the Accuracy = Precision = F1_Score = Recall = 12.75%.
- Confusion Matrix:

3	7	690	188	2
0	1	56	13	0
0	1	143	33	0
1	0	28	6	0
0	1	22	5	0

Rotational Augmentation: Experiment-4

- **3rd Epoch:** It was observed that the Accuracy = Precision = F1_Score = Recall = 10.17%.
- Confusion Matrix:

9	25	488	363	5
0	1	41	27	1
1	3	100	72	1
0	3	20	12	0
0	1	16	11	0

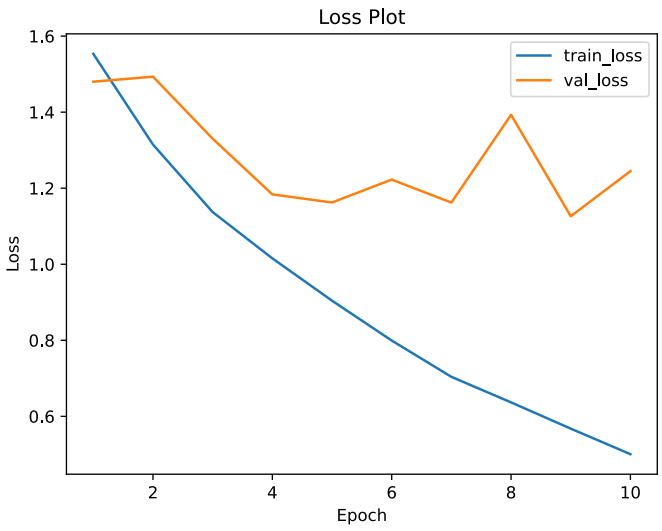
- **7th Epoch:** It was observed that the Accuracy = Precision = F1_Score = Recall = 12.42%.
- Confusion Matrix:

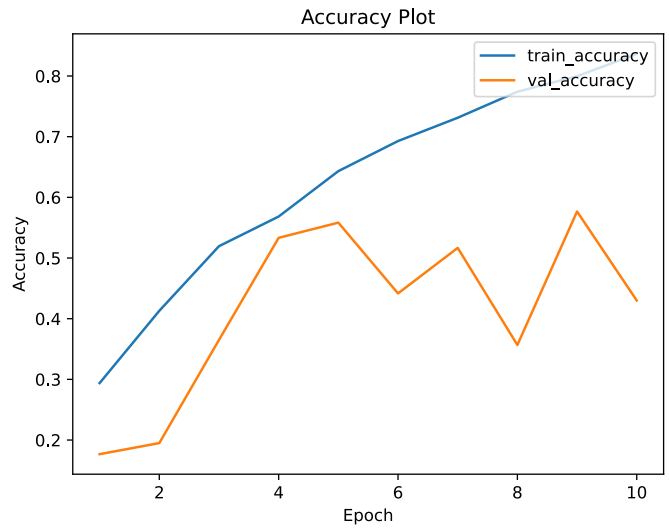
0	46	730	105	9
0	2	56	12	0
2	7	140	27	1
0	1	27	7	0
0	3	21	4	0

- The performance has marginally increased in comparison to Experiment 3. However, it is still very bad because the model predicts more non-dominant class labels.
- Epochs haven't been increased owing to over-fitting.

Rotational Augmentation: Experiment-5

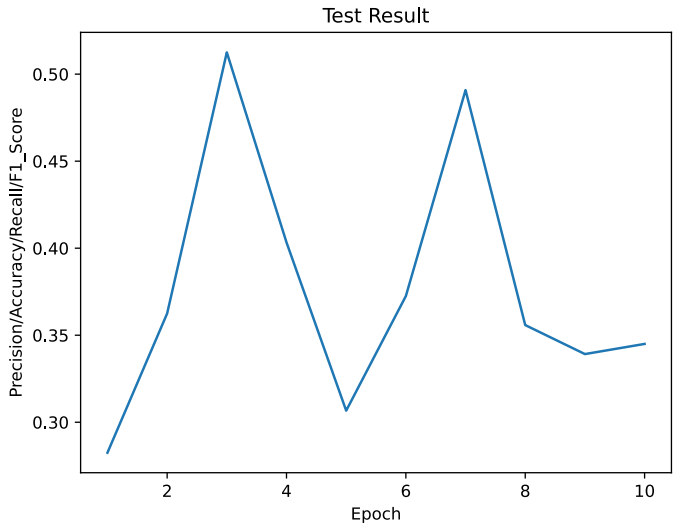
The number of nodes in the penultimate layer was altered to 32. The weight decay was set at 5. The batch size was set at 16, keeping the learning rate at 1e-5. The training was done for 10 epochs.





The test performances were checked on all the models. The results are mentioned in the next slide.

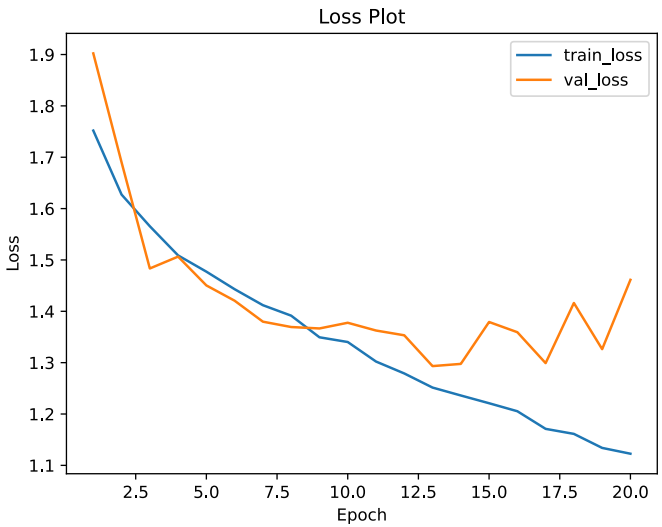
Rotational Augmentation: Experiment-5

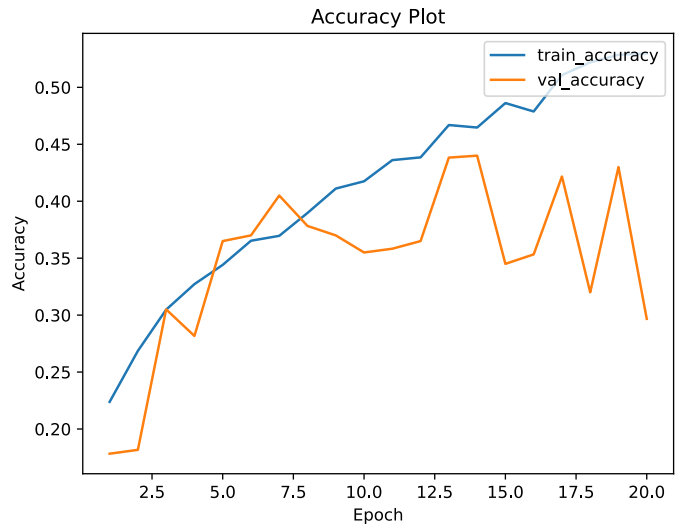


- It was observed that all those models with higher test accuracy predicted most of the images as belonging to class 0. Therefore, the higher accuracies result from the distribution of the classes rather than the model being able to successfully predict based on learning the underlying patterns.
- Models with lower accuracies (usually at higher epochs) predict most images as belonging to the non-dominant classes.

Rotational Augmentation: Experiment-6

The number of nodes in the penultimate layer was altered to 32. The weight decay was set at 5. The batch size was set at 16, keeping the learning rate at 1e-6. The training was done for 20 epochs.





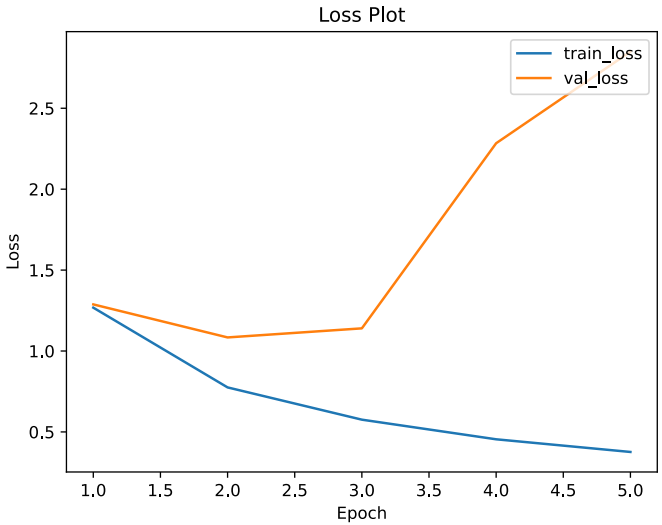
The test performances were checked on all the models. However, accuracy was never above 11.4%. The best test performance was obtained at the end of the third epoch. Observe that the number of epochs has been increased after reducing the learning rate.

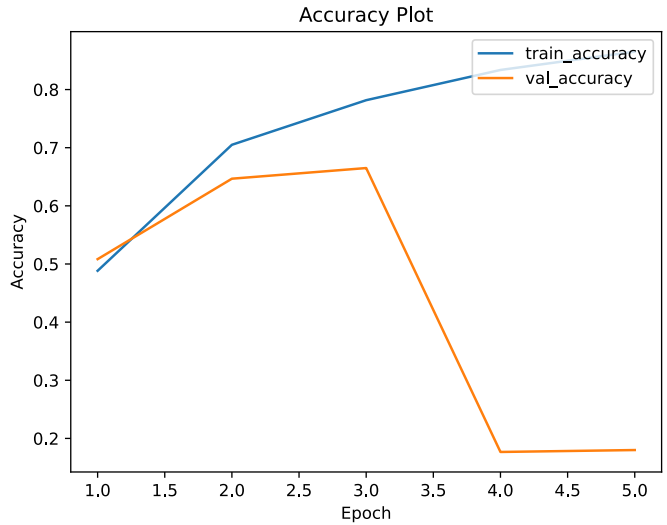
Use of Synthetic Sampling Techniques - SMOTE and ADASYSN

- So far, we used rotational augmentation where the size of the dataset was brought down from 7200 to 4196. The results were NOT satisfactory.
- Instead of reducing the number of samples, we now try to increase it using SMOTE and ADASYN.
- Both of them work by increasing the number of samples synthetically of the minority classes.
- The number of images in the train set was enhanced to around 5*5255 = 26275.

SMOTE

While SMOTEing, the number of epochs was set to 5, batch size = 16, Learning Rate = 1e-5, and weight_decay = 5. The number of nodes in the penultimate layer was set to 32.

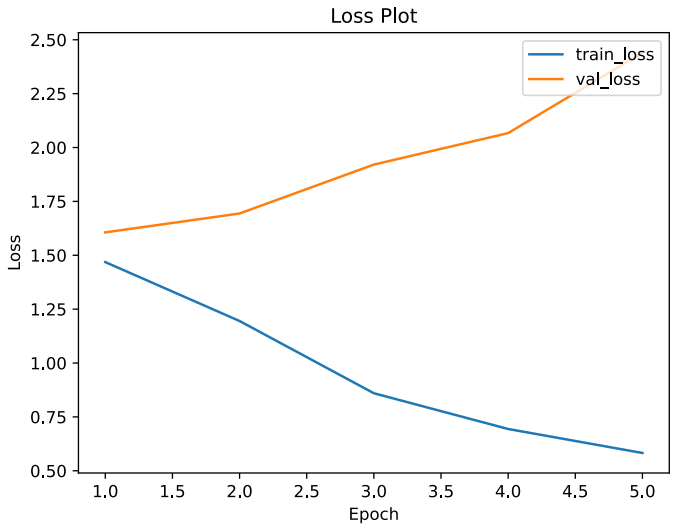


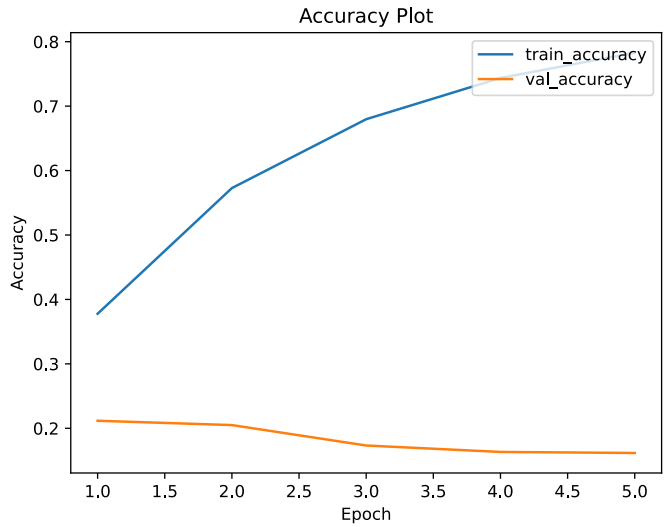


The test performances were checked on all the models. However, accuracy was never above 4.3%. All the epochs had an accuracy between 3-4.3%.

ADASYN

The number of epochs was set to 5, batch size = 16, Learning Rate = 5e-6, and weight_decay = 5. The number of nodes in the penultimate layer was set to 32.





The test performances were checked on all the models. However, accuracy was never above 18.16% (obtained at the end of the first epoch). All the epochs had an accuracy between 9.17-18.16%.
In case of both ADASYN and SMOTE, the model predicts most of the images as belonging to the non-dominant classes.

Inferences

- It appears that the model learns the dominant class when there is a significant class imbalance and the non-dominant class when some form of augmentation is done.
- This might be because the non-dominant classes are easier to learn.
- Therefore, the dataset has to be optimized so that the number of images belonging to the non-dominant classes is neither too high nor too low.
- To enhance the model's performance, FC DenseNet can NOT be used due to the unavailability of segmentation maps. These maps are required to find the aneurysms.
- The Kappa metric is expected to be on the lower side for the given dataset owing to the severe class imbalance and hence wasn't used extensively.
- While testing, the Precision = F1_Score = Recall = Accuracy. This is occurring because we are using micro-averaging.

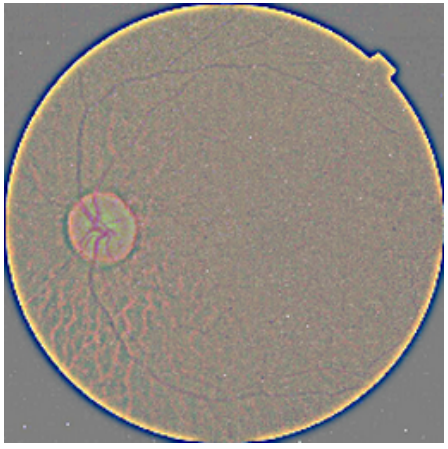
New Dataset

Given all the disadvantages of the previous dataset, we explore a new dataset.
The total number of images = 2750. 2200 Images were used for training and 275 each for testing and validation. Each image was 224x224x3.

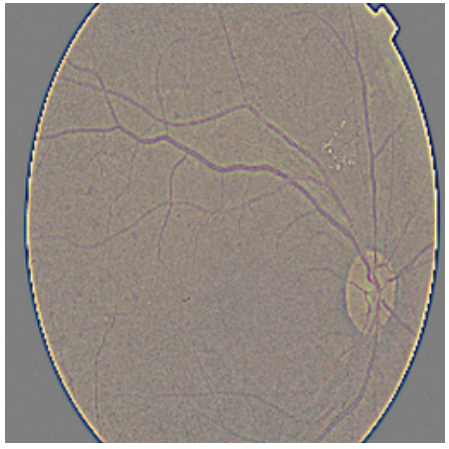
Class 0	1000
Class 1	370
Class 2	900
Class 3	190
Class 4	290

Representative Images

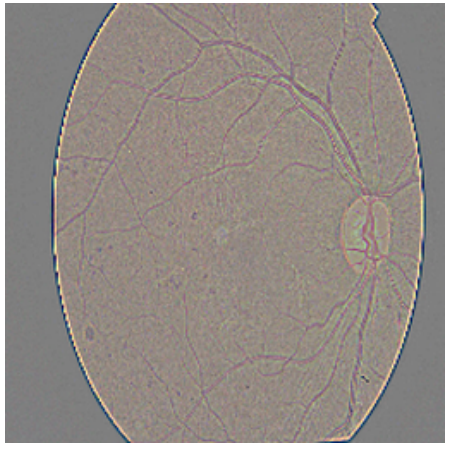
The following images correspond to the five stages of the disease.



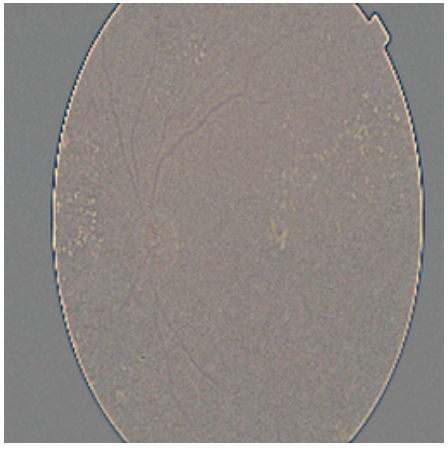
No DR



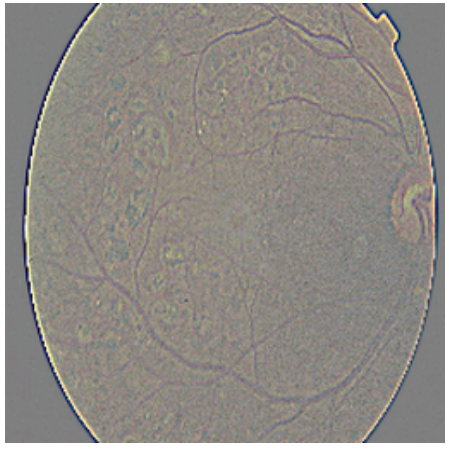
Mild DR



Moderate DR



Severe DR



Proliferative DR

Final Pre-Processing

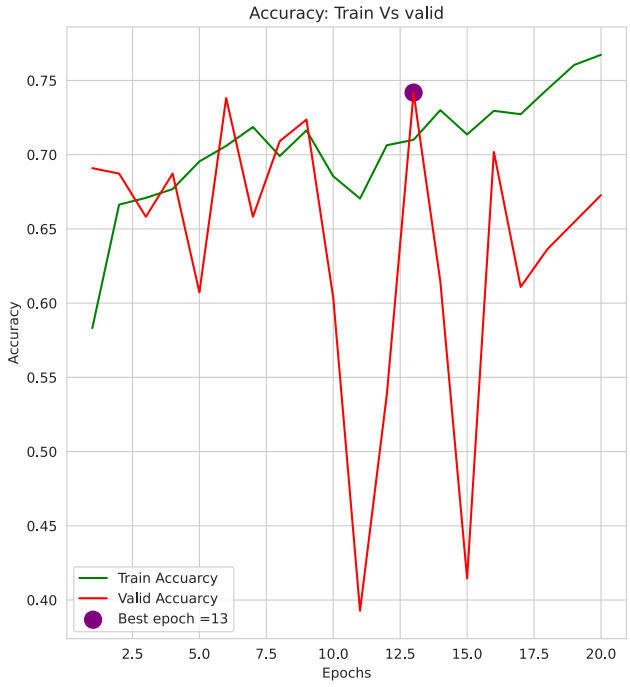
- ZCA Whitening was performed to decorrelate the pixels.
- The rotation range was set to 30 degrees, implying that the input images might be rotated by angles of at most 30 degrees either clockwise or counterclockwise.
- The newly created pixel values after the above transformations are interpolated based on the nearest existing pixel. Resizing was not required as all the images were of the same size

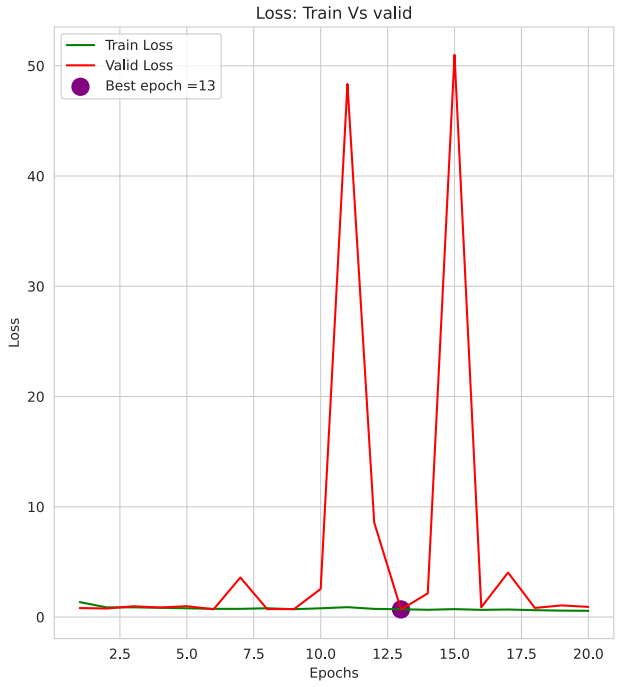
Final Experiment

- Over and above the base model layers of EfficientNetB4, we add a dense layer with 64 neurons using the ReLU Activation function. The output layer had 5 neurons and used the SoftMax Activation Function.
- The model was then trained for 20 epochs.
- The Learning Rate was set at 0.001
- The optimizer used was Adam.
- Batch Size was set to 20.

Final Results - Training and Validation

Given the various specifications in the previous slide, the following results were obtained:

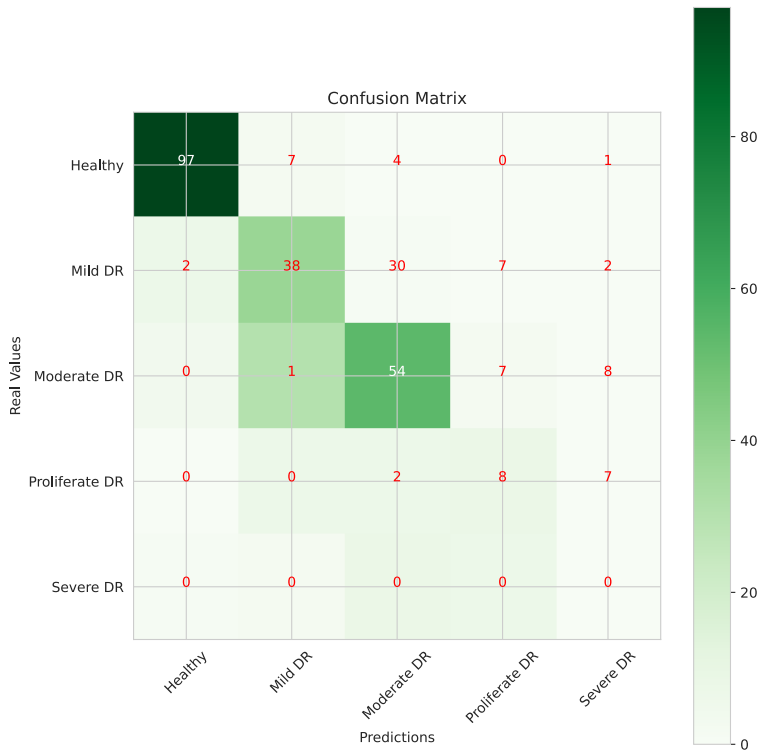




The model obtained at the end of the 20th epoch was tested. The test results have been provided in the next slide.

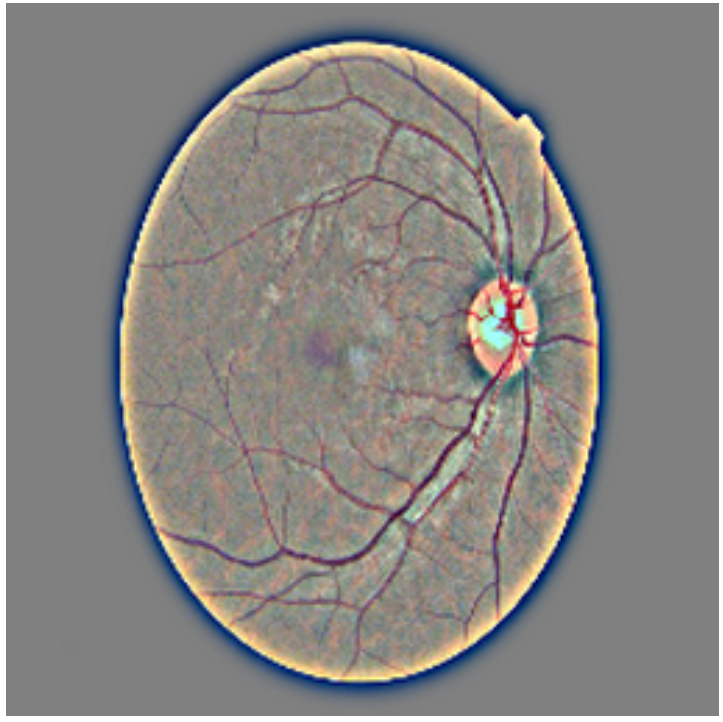
Final Results - Test: Quantitative

The Accuracy = Precision = Recall = F1_Score = 0.7163636363636363
Quadratic Weighted Kappa Metric = 0.7591537406352221



The model performs well in most of the classes except the class: Severe DR, which comprises the least number of samples.

Final Results - Test: Qualitative



Predicted Label = 0 True Label = 0

Conclusion and Future Work

- The training accuracy follows an increasing trend across epochs.
- Since the val accuracy peaked at the end of the 13th epoch, the number of epochs wasn't increased beyond 20 to prevent overfitting.
- Increasing the depth and width of the newly introduced ANN is unlikely to improve substantially as the base model itself is very deep.

Future Improvements

- The performance can be enhanced by oversampling the non-dominant classes keeping in mind the learnability across various classes to prevent over-fitting.
- Datasets comprising of higher-resolution images can be explored, which might help the model detect more intricate patterns.
- The model can be used in conjunction with segmentation models to gain additional insights regarding the microaneurysms, thereby enhancing performance.

Overall, we have met the initial goal at an intermediate level. Given more resources, better datasets, and more models to experiment with, this project can be turned into a product with a good practical value capable of helping society at large.

References

The following references were used:

- Keras Applications
- Adam Optimizer
- Predictive Analysis of Diabetic Retinopathy with Transfer Learning
- Automatic Diabetic Retinopathy Classification with EfficientNet
- Reference Kaggle Notebook

Thank You

I thank Dr. Soumya Jana for providing me with an opportunity to work on an exciting and challenging project.

I would also like to thank Dr. Sumohana Channappayya and Mr. Ambarish Parthasarathy for providing me with GPU access, without which it would have been impossible to train the various models and evaluate their performances on time.

All the codes done so far can be found on the following GitHub Repository:
`https://github.com/Vaideeswaran21/PRML_Project`