Definition

Project Overview

We all are familiar with Diabetes mellitus (DM), commonly known as diabetes, is a group of metabolic disorders characterized by high blood sugar levels over a prolonged period. According to IDF(International Diabetes Federation) 425 Million adults have diabetes in which 1 in 2 remain undiagnosed. One of the many causes of it is Diabetic retinopathy eye Disease.
Diabetic retinopathy affects blood vessels in the light-sensitive tissue called the retina that lines the back of the eye. It is the most common cause of vision loss among people with diabetes and the leading cause of vision impairment and blindness among working-age adults. It doesn't have any early symptoms. As of now, Retina photography is a way to detect the stage of Blindness. Automating it with ML will help a lot in the health domain. Computers are getting better at understanding images due to advances in computer vision. We can now able to detect and diagnose diseases quite well. With this, we can able to gain the ability to automatically detect disease and provide information on how severe the condition may be.

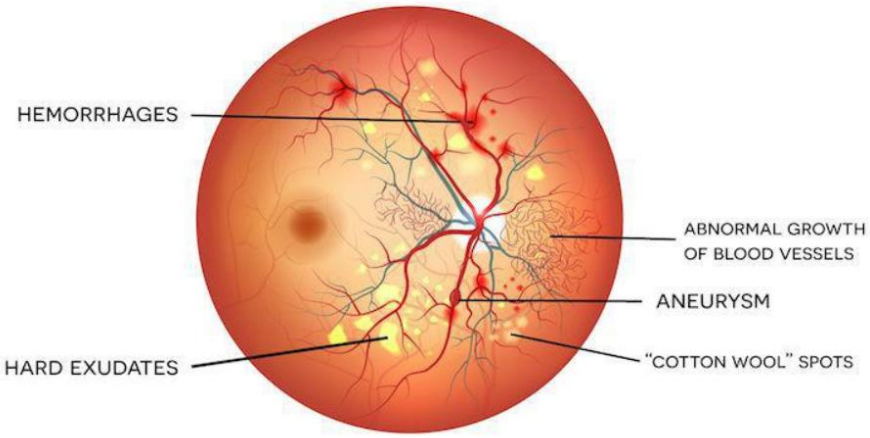

Fig. 1 Diabetic retinopathy can result in many serious issues affecting the blood vessels that nourish the retina**.**

In this project, I worked on Deep learning CNN algorithm adept at processing images and training the model on a large set of retina images to determine the severity level of the patient. The project was a Kaggle Competition.

Problem Statement

The goal is to create a classifier to classify the retina images as per there severity level. The tasks involved are the following:

1. Visualize the dataset and understand the problem for which solution to be searched.
2. Preprocess the data to enhance its features.
3. Data Augmentation and image Generation.
4. Train a classifier that can classify image based upon its severity
5. Predict the test images using the classifier.

The final classifier is expected to be useful to diagnosis the person retina condition and help to take the next step for a cure in the health care field.

Metrics

Predictions are scored based on the Quadratic weighted kappa, which measures the agreement between two ratings. This metric typically varies from 0 (random agreement between raters) to 1 (complete agreement between raters). If there is less agreement between the raters than expected by chance, this metric may go below 0. The quadratic weighted kappa is calculated between the scores assigned by the human rater and the predicted scores.
Images have five possible ratings, 0,1,2,3,4. Each image is characterized by a tuple *(e,e)*, which corresponds to its scores by *Rater A* (human) and *Rater B* (predicted). The quadratic weighted kappa is calculated as follows. First, an N x N histogram matrix *O* is constructed, such that *O* corresponds to the number of images that received a rating *i* by *A* and a rating *j* by *B*. An *N-by-N* matrix of weights, *w*, is calculated based on the difference between raters' scores:
An *N-by-N* histogram matrix of expected ratings, *E*, is calculated, assuming that there is no correlation between rating scores. This is calculated as the outer product between each rater's histogram vector of ratings, normalized such that *E* and *O* have the same sum.

The Formula for weighted Kappa is:

$$\kappa = 1 - \frac{\sum_{i=1}^{k} \sum_{j=1}^{k} w_{ij} x_{ij}}{\sum_{i=1}^{k} \sum_{j=1}^{k} w_{ij} m_{ij}}$$

Analysis

Data Exploration

The large set of retina images taken using fundus photography under a variety of imaging conditions.
A clinician has rated each image for the severity of diabetic retinopathy on a scale of 0 to 4:

0 - No DR
1 - Mild
2 - Moderate
3 - Severe
4 - Proliferative DR

As it is a part of the kaggle Competition, a similar competition was held in 2015 with a similar set of the dataset. Here I am going to use previous competition dataset also to train my Model.

It contains two sets of data:

- 2015 Blindness Detection Dataset:
  - trainlabel15.csv - the training labels with columns ‘image’ and ‘level’.
  - train15.zip - the training set contains 35126 images of the retina.
- 2019 Blindness Detection dataset
  - train.csv - the training labels with columns id_code and diagnosis.
  - test.csv - the test set for each id_code a predicted diagnosis to be mapped.
  - train.zip - the training set contains 3662 images of the retina.
  - test.zip - the test set contains 1928 images of the retina.

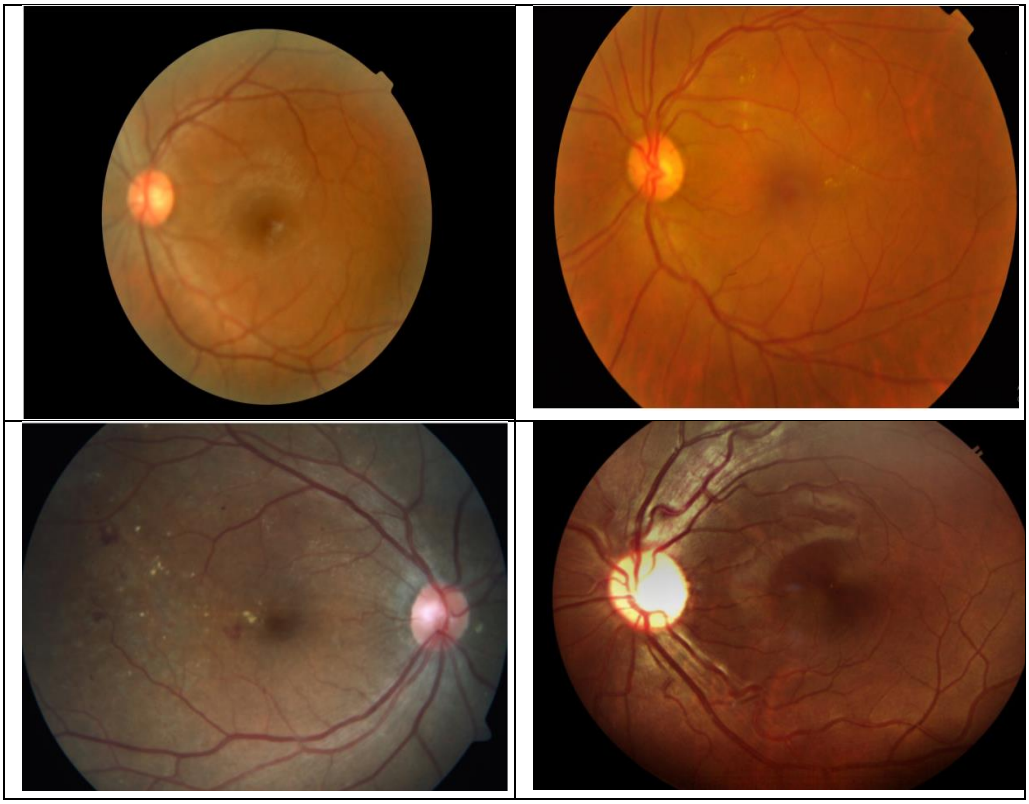

Fig. 2 Sample images of Retina.

Like any real-world data set, it also encounters noise in both images and labels. Images may contain artifacts, be out of focus, underexposed, or overexposed. The images were gathered from multiple clinics using a variety of cameras over an extended period, which will introduce further variation like pixel size of images varies retina image placement in images are not uniform.

Exploratory Visualization

The plot below shows how the class are distributed among the image samples. This helps predict how balanced the classes will be after the images are segmented.

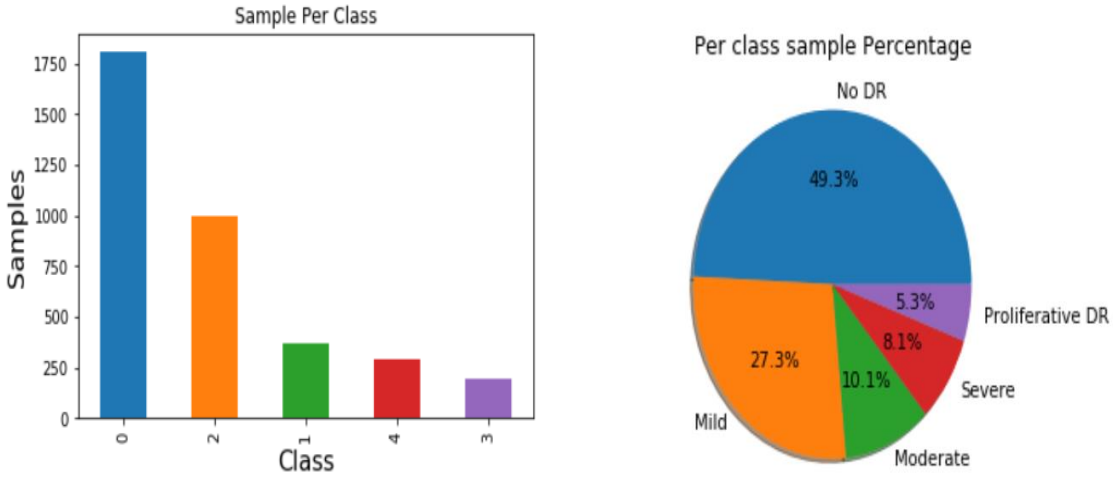

Fig. 3 (a) Plot shows the Sample of images per Class **(b)** Pie chart visualize the distribution of images in each class.

It can be seen that:

- There is a data imbalance of sample images among the classes.
- Sample images of Proliferative DR, Severe and Moderate are less.
- No DR- 1805, Mild- 999, Moderate - 370 , Severe- 295 , Proliferative DR- 193

Algorithms and Techniques

The classifier is a Convolution Neural Network, which is the state-of-the-art algorithm for most image processing tasks, including classification. It needs a large amount of training data compared to other approaches; unfortunately, the Retina images are small in number which can be overcome by using Data Augmentation technique. The preprocessed input data is provided to the pre-trained CNN network using transfer learning. The algorithm outputs an assigned probability for each class; this can be used to reduce the number of false positives using Test Time Augmentation.

The following parameters can be tuned to optimize the classifier:

- Data Verification
  - Labels encoding.
  - Pre-Processing Data.
- Data Augmentation.
  - Generate batches of tensor image data with real-time data augmentation. The data will be looped over (in batches).
- Neural network architecture.
  - Pre-Trained model Resnet-50
  - Pre-Train all the layers with 2015 dataset.
  - Fine-Tune the model with 2019 dataset.
  - The number of layers.

- Layer types (convolutional , fully-connected , Batch Normalization, Dropout, Activation Function, or pooling)
- Layer parameters (see links above)

❖ Training Parameter.
- Training length(Number of epochs)
- Batch size (how many images to look at once during a single training step)
- Learning rate (how fast to learn; this can be dynamic)
- Model Optimizer.
- Model Callbacks(Learning Rate Schedule, ReduceLROnPlateau , EarlyStopping).

❖ Test Time Augmentation.
- Random augmented test images are used to prediction and average the predictions of each corresponding image and take that as our final guess.

Benchmark

To create an initial benchmark for my project. I created a CNN model with Pre-trained Resnet-50 model. The model was trained and tested with the raw dataset.

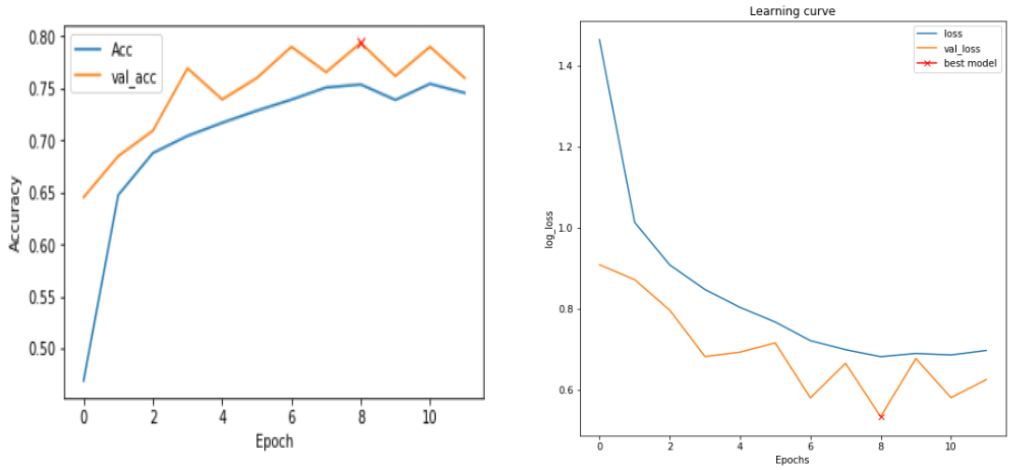

Fig. 4 (a) Accuracy Vs Epoch curve & **(b)** log loss Vs Epoch(Learning curve)

After training the model with the data it yields the below accuracy and loss over validation data set. Then the model is used to predict the testing set of data. Testing images are passed in the Classifier which in turns predicts the class for the test set.

Accuracy	**79.55%**
Loss	**0.54**
evaluation Metrics	**0.63**

Benchmark for the proposed model:
- Accuracy of the model should be greater than 79.55%
- Validation loss of the model should be less than 0.54
- score as per the evaluation metrics should be greater than 0.63

Methodology

Data Preprocessing

1.1 Original Input.

First, let's have a glance of original inputs. Each row depicts each severity level. We can see two problems which make the severity difficult to spot on. First, some images are very dark [pic(2,2) and pic(3,3)] and sometimes different color illumination is confusing [pic (4,4)]. Second, we can get the uninformative dark areas for some pictures [pic(1,2), pic(1,4)]. This is important when we reduce the picture size, as informative areas become too small. So it is intuitive to crop the uninformative areas out in the second case. As well as dimensionality of images is Non-uniform which should be resized in a standard Input shape.

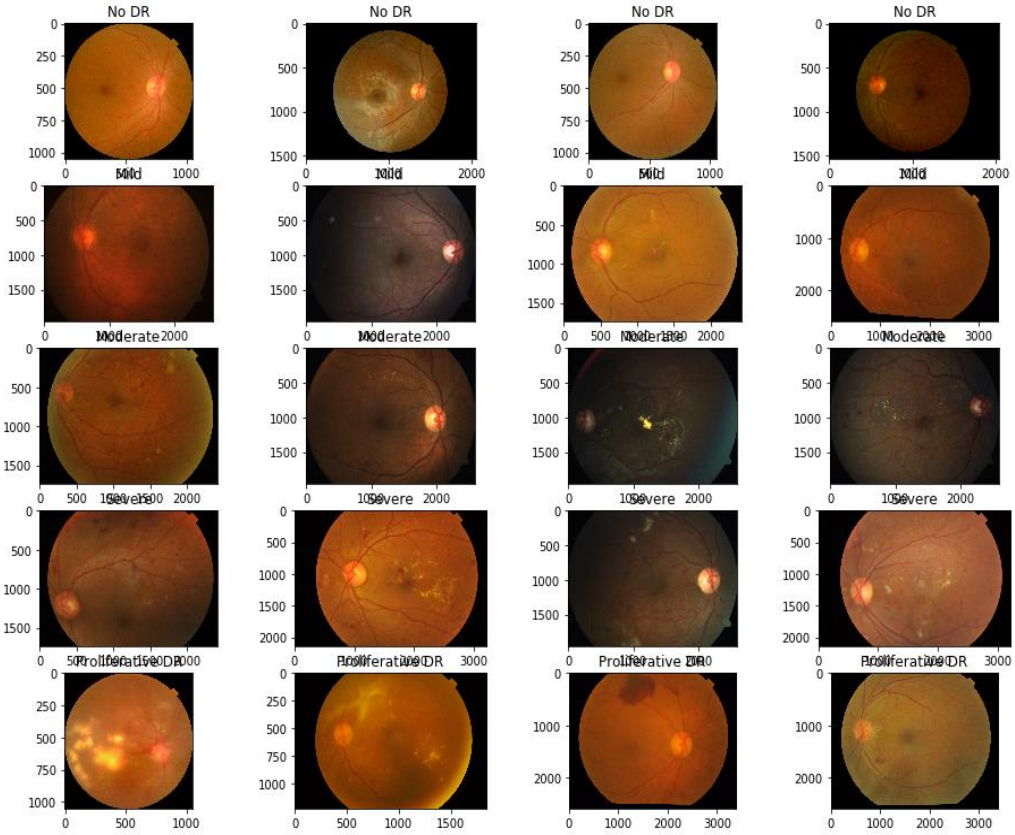

Fig. 5 Sample images of each level of severity

1.2 Data Pre-processing:

- Read in the image:
- Apply Mask & auto-cropping:
  To crop out the uninformative black areas which are evident on the pic(1,2), pic(1,4) and pic(4,2), we can try auto-cropping.
- Resize the image to the desired size.
  Resize the image into a standard resolution for input.

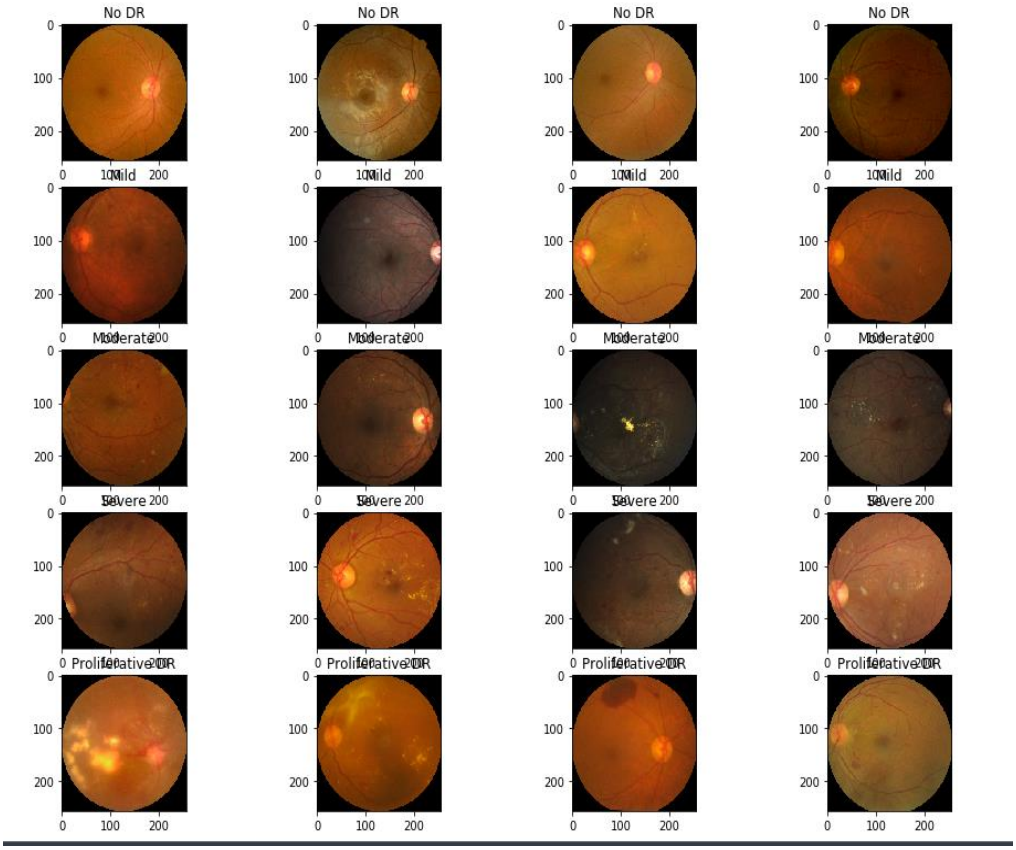

Fig. 6 Sample images of each level of severity after preprocessing

Note: As the Computational cost for training images with high resolution will be high to avoid that I am using training my dataset with 224,256 or 299 images resolution.

Implementation

The implementation process can be split into two main stages:

- Data Augmentation.
- The classifier training stage.

Data Augmentation was carried out using ImageDataGenrater class of Keras:

- ImageDataGenrater Function was defined with below Arguments.
  - Validation split: Float. Fraction of images reserved for validation.

- horizontal flip: Boolean. Randomly flip inputs horizontally.
- vertical flip: Boolean. Randomly flip inputs vertically.
- shear range: Float. Shear Intensity (Shear angle in a counter-clockwise direction in degrees).
- zoom range: Float or [lower, upper]. The Range for random zoom. If a float, [lower, upper] = [1-zoom_range, 1+zoom_range].

During the first stage, the classifier was trained on the preprocessed training data. This was done in a Jupiter notebook Model Training and Evaluation section and can be further divided into the following steps:

1. Load both the training images into memory, preprocessing them as described in the previous section.
2. Implement Data Augmentation:
    a. ImageDataGenrater: Split the image data into training/validation data and augment the training data set.
    b. Train/validate generator: Generating images after image augmentation for training and validation.
3. Define the network architecture and training parameters mentioned above in Algorithm and technique section.
4. Define the loss function, accuracy
5. Train the network, logging the validation/training loss and the validation accuracy
6. Plot the logged values
7. Save and freeze the trained network and load the best weights for prediction.
8. Implement ImageDataGenrater for testing Set of image.
9. Define TTA(Test Time Augmentation) for predicting image data and select the image with maximum probability.
10. Save the predicted label with its respective image Id_code in a CSV file for getting the score based on Evaluation metrics as per kaggle competition.
11. If the score is not high enough, return to step 3.

Refinement:

As mentioned in the Benchmark section, the CNN model architecture trained from Scratch in a CNN with-out Pre-Processing the achieved accuracy, around 77% with 0.55 Validation loss is performed average as determined by the Evaluation metrics.
To get the initial result, the data set was trained in this architecture was ported to a Pre-trained ResNet-50 Model using Transfer learning with added new Neural network layers to train the model for Retina image prediction. The result was an **accuracy of around 84%.** This was improved by using the following techniques:

❖ Dynamic learning rate: whenever the loss function stopped decreasing, a learning rate drop was added.

❖ Weight decay: when over fitting was detected (the training and validation losses diverged too much), the weight decay rate was increased

❖ Adding dropout to a layer: dropout randomly drops weights in the layer it's applied to during training and scales the weights so that the network keeps working during inference.

❖ Pre-training the classifier with a similar type of dataset and fine-tune it with the original dataset to boost the model performance.

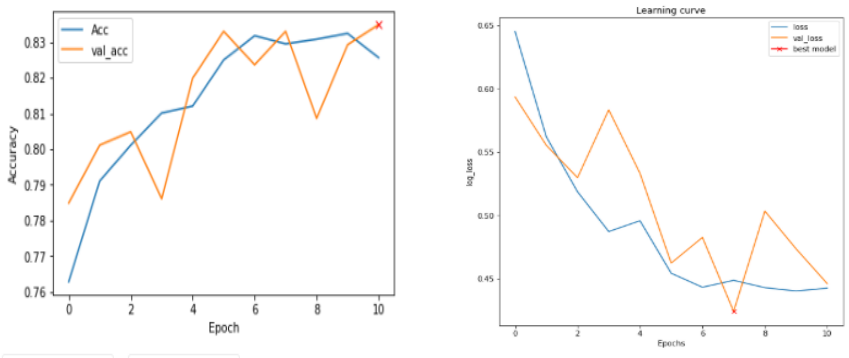

Fig. 7 (a) Accuracy Vs Epoch curve & **(b)** log loss Vs Epoch(Learning curve)

Accuracy	**84%**
Loss	**0.42**
evaluation Metrics	**0.71**

The final Keras model was derived by training iteratively, adjusting the parameters (e.g. learning rate, weight decay ratios) based on plots like the one below. The final model has an **evaluation Metrics of 0.71.**

Results

Model Evaluation and Validation

- During development, a validation set was used to evaluate the model.
- The final architecture and hyperparameters were chosen because they performed the best among the tried combinations.
- Pre-trained Resnet-50 model is mounted on top of the layer.
- A Global average pooling is used with dropouts.
- Two fully connected dense layer(1024 & 512) with dropout is used.
- Batch normalization is used to increase the stability of a neural network.
- Finally, the output layer is connected with 5 nodes (to determine the 5 classifications of the image) with 'softmax' activation.
- The model is designed with callbacks with the following features:
    - EarlyStopping: Stops training once the model performance stops improving on a hold out validation dataset.
    - ReduceLROnPlateau: Reducing the Learning Rate if the result is not improving.
    - Checkpoint: Save the best model weight after every epoch.
    - csv_logger: Callback that streams epoch results to a CSV file to analyse the model performance.

Below Analysis was conducted to validate the robustness of the model.
Training dataset was split into training(80%), validation(10%) and testing(10%) dataset.

- Training/validate data in various image Resolution of image (i.e. 224, 300, 512).
- It is used to select the decent Image resolution to train data with the efficient computational cost for good results.
- Trained Classifier is used to predict and test the model efficiency using test split.

Finally, the model with robust performance is selected with Proper image Resolution which will be having decent Computational Cost.

Image Resolution	Accuracy	loss	Evaluation metrics score	Computational Cost
224*224	79%	0.55	0.85	Less
300*300	81%	0.49	0.88	Less
512*512	82.54%	0.48	0.87	High

Justification

The below statistics signify the model performance on the benchmark established. As we can see the model can classify the images into various classification but the details which it is missing causes false-negative and false-positive diagnosis. a false positive is an error in data reporting in which a test result improperly indicates presence of a condition, such as a disease (the result is positive), when in reality it is not present, while a false negative is an error in which a test result improperly indicates no presence of a condition (the result is negative), when in reality it is present. This is a critical issue while diagnosis due to which model sensitivity and specificity is our primary concern.

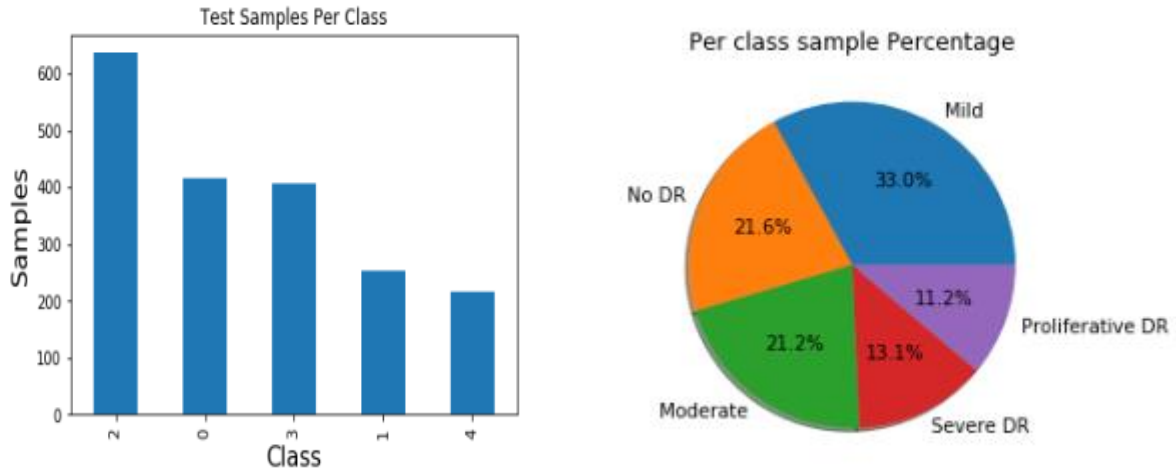

Fig. 8 (a) Plot shows the prediction of images per Class for Bench establishment **(b)** Pie chart visualize the distribution of predicted images in each class for Bench establishment.

Whereas, In the final solution, image classification can capture the small details and able to classify images better than benchmark establishment with better sensitivity and specificity. As the model is trained with an unbalanced dataset which provides fewer details on the severe classifications.

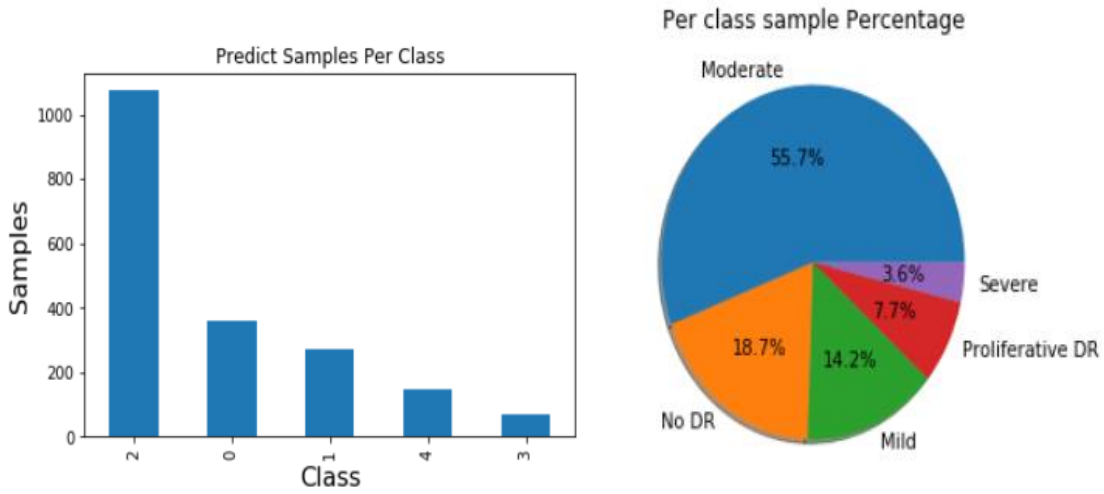

Fig. 9 (a) Plot shows the prediction of images per Class for the final solution **(b)** Pie chart visualize the distribution of predicted images in each class for the final solution.

In summary, the model is still needed more research to improve its performance to use it for real-time Diabetic retinopathy diagnosis (see the Improvement section).

Conclusion

Free-Form Visualization

Below is the result of images which is the result generated from the model where each row depicts each level of severity of the retina.

Following are the Observation we can get from the results:

- As the severity of the retina increases yellow patches are visible more often.
- Retina images with Proliferative DR are having small blood clots on the retina.
- Images from Sever group and Proliferative DR are showing quite good visibility of the "Hemorrhages" and Hard Exudates" mentioned in fig. 1.

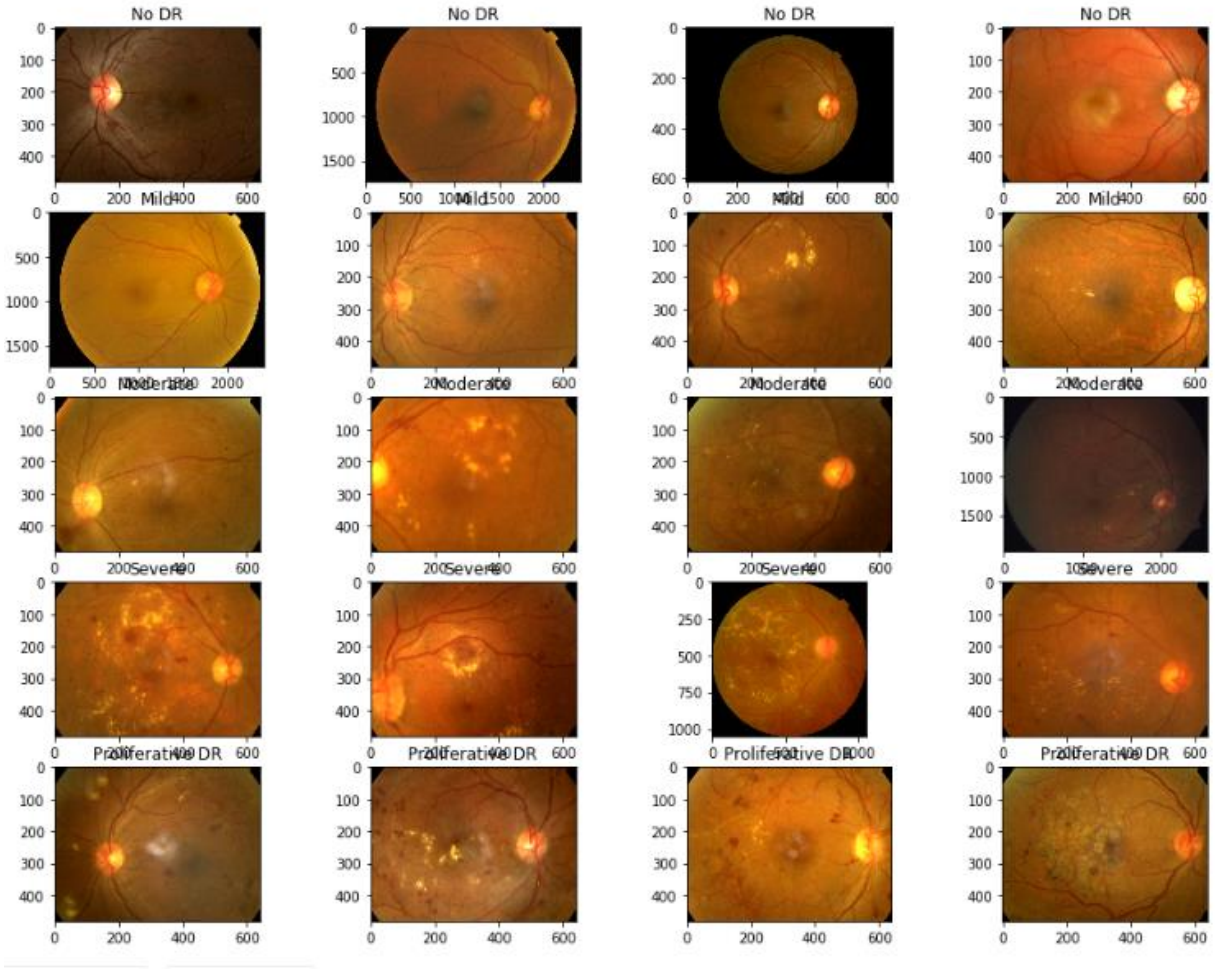

Fig. 6 Sample images of each level of severity predicted by the classifier

Reflection

The process used for the project can be summarized using the following steps:

1. An initial problem and relevant, public dataset were found.
2. The baseline of algorithms and Techniques were decided to solve the problem.
3. The benchmark is created for the Classifier model.
4. The data was pre-processed and visualized to understand it in a better way.
5. The classifier was trained using the data (multiple times, until a good set of parameters, were found)
6. The classifier is also trained with a different resolution of input images multiple times until a good input image resolution is decided to cut computational cost with better efficiency.
7. The classifier is used to predict the testing set of images and tested on the evaluation metrics to generate the score.

I found Step 4,5 and 6 most difficult as I have to try different combination to understand which kind of preprocessed data will work on the classifier. As the various preprocessed data were performing

differently in the same classifier. So finding the efficient connection between the preprocessed data input in the classifier with good image resolution was a difficult task for me.

As of the most interesting aspects of the project, I am very encouraged to use deep learning and understand its application in the health care field and what all we can accomplish with this soon for the betterment of the world. I am looking forward to continuing work on it to create an efficient model for the social cause.

The final solution performed very well for me as a newbie to deep learning but right now it needs a lot of improvements to use it as a real-world application to detect blindness causes.

Improvement

Yes, there is always a scope for improvement in everything, similarly, in this project the improvement on which work to be done are:

- Understanding the data in more detail to preprocessed it using image processing to highlight the details which we want to classify.
- More data to train will surely increase the accuracy of the classifier to understand the features better.
- The classifier can be trained on other model weights like Efficient Net weight using transfer learning to get better results.
- Classifier if trained with good GPU with high resolution can able to detect more details of the features.