DETECTION OF BLINDNESS SEVERITY IN DIABETIC RETINOPATHY

Sandesh Reddy Pendyala

Z1859176

Sneha Ravi Chandran

Z1856678

Priyanjani Chandra

Z1864520

Sindhusha Devi Parimi

Z1855951

OBJECTIVE:

The main objective of this project is to build a machine learning model that can classify images on the severity of Diabetic Retinopathy of the patient's eyes. The model should be able to work fast on the input images and have enough accuracy to be able to be used by a doctor reliably.

ARCHITECTURE:

Overview: We used CNN, MobileNetV2, RestNet V2 50, ResNet V2 101 models along with transfer learning to compare the accuracy and perform prediction of the test data on the one model. The prediction is done on ResNet V2 101.

The below are the class labels

0- No DR

1- Mild

2- Moderate

3- Severe

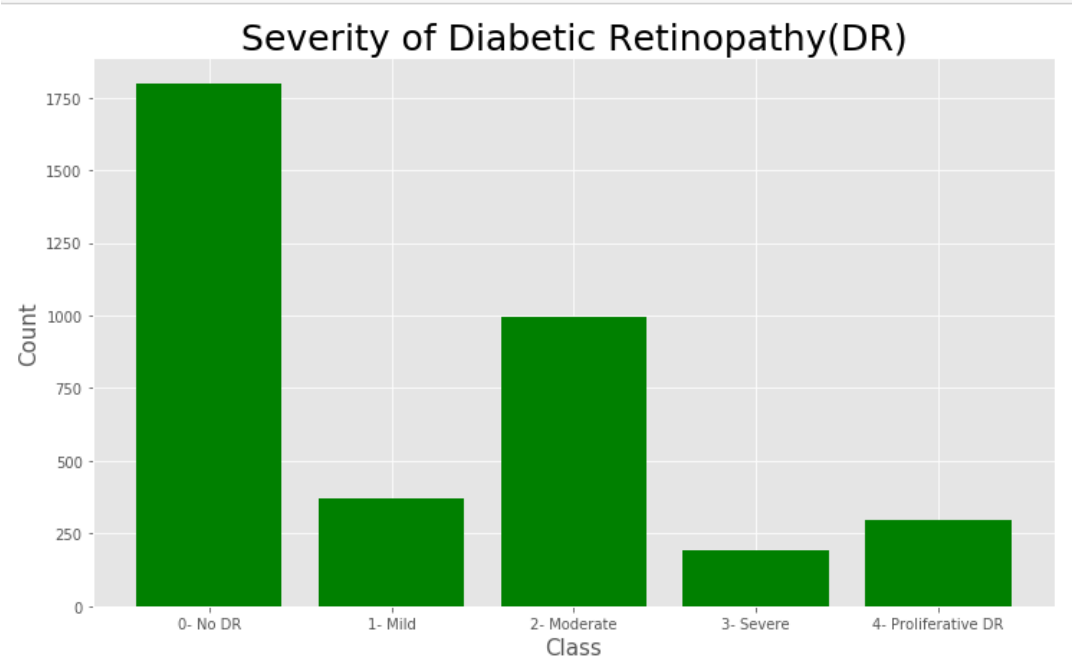
4- Proliferative DR

Steps: Using pandas, we read the initial training CSV into a dataframe and grouped the images into its corresponding class labels (diagnosis) and split them into separate CSVs, one for each class. Based on the images grouped in each of the class CSVs we parsed from the original folder path where training images were present and stored it to the new path, having the directory name same as the class name.

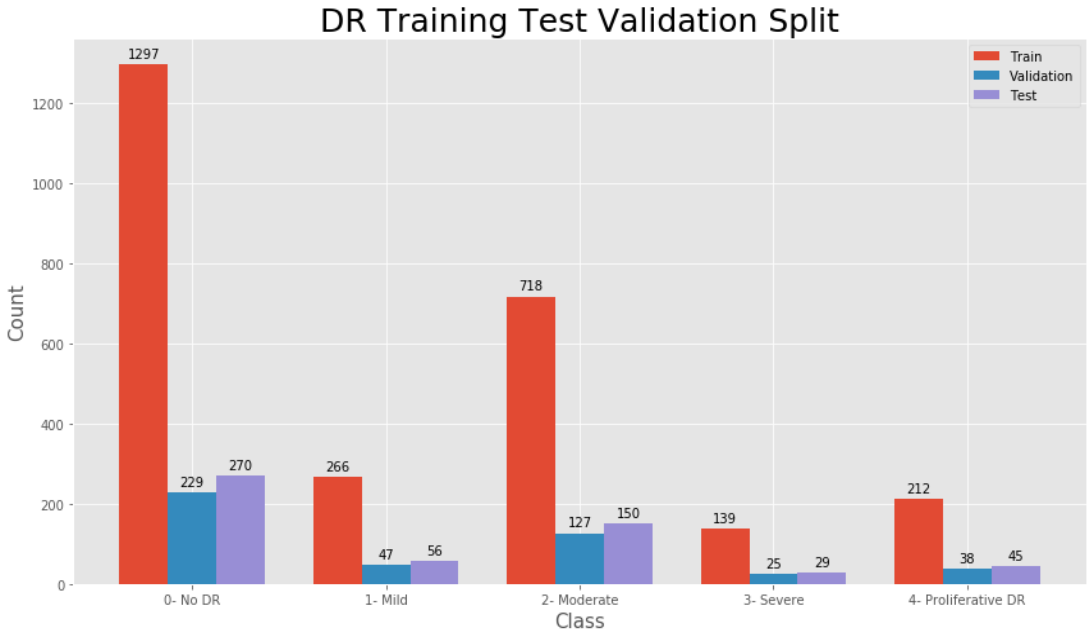
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1805 entries, 0 to 1804
Data columns (total 5 columns):
0    1805 non-null object
1    370 non-null object
2    999 non-null object
3    193 non-null object
4    295 non-null object
dtypes: object(5)
memory usage: 70.6+ KB
```

We begin with a raw data that has variable image size and severe class imbalance. The number of images of class 0 which corresponds with the patient having perfect sight is equal to the number of images of the rest of the classes combined. We had to also split this training image data set where the folder only had images without any class metadata. The class data was on a different csv file with which we had to combine and split into 5 different folders which only had those images that were corresponding with their class. We then split this data into training, validation and test data sets.

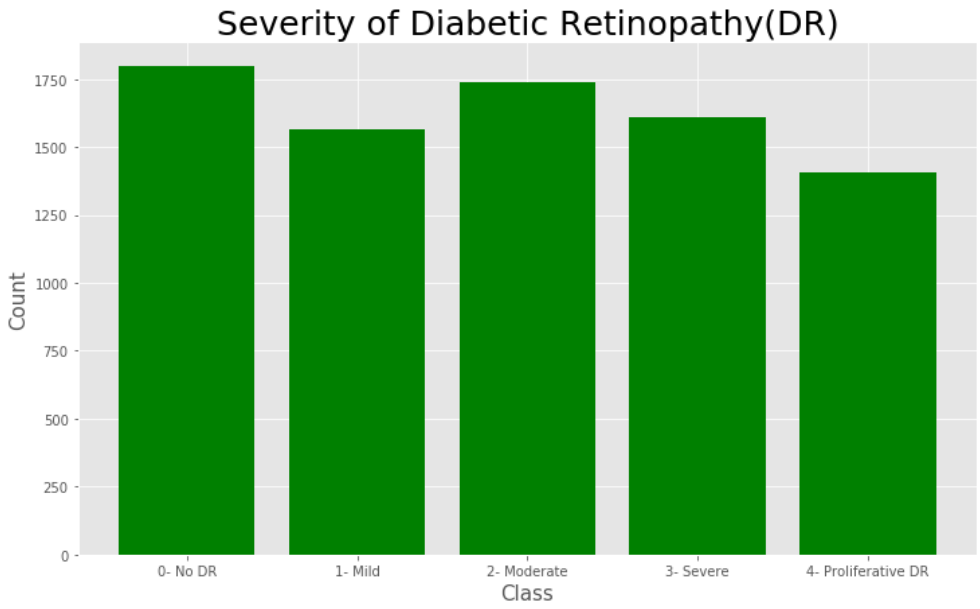
Visualization of the original class labels after splitting into its respective classes



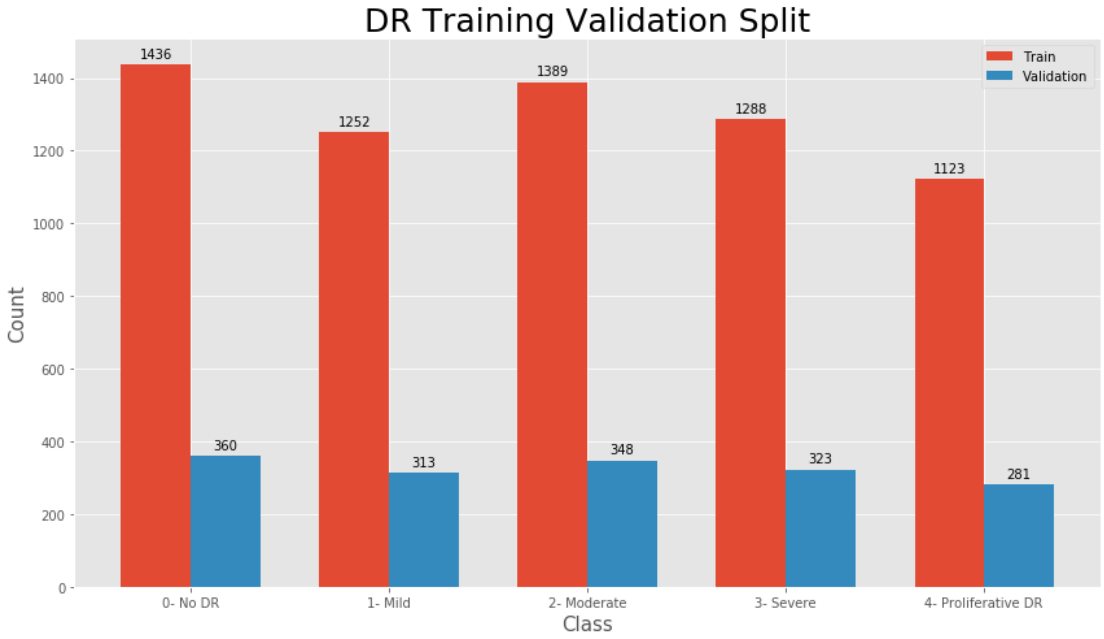
Visualization after splitting into train, test and validation



Due to class imbalance we oversampled minority classes from the dataset and visualized for the same



Visualization of the Training and validation oversampled dataset



Once the data has been cleaned and ready for the model to be parsed, we now had to experiment with the hyper parameters of the Convolutional neural network. Initial runs revealed that the model was learning the characteristics of the images quite well. Augmenting the data using conventional methods like horizontal flip, rotation had resulted in measurable increase in accuracy.

But the CNN model we built was not accurate enough to be used in real life classifications. Any number of modifications to the hyper parameters like changing the number of nodes in layers, number of layers, number of epochs did not see any worthwhile increase in accuracy. So, we now moved onto established ImageNet models that are far more effective at classifying multiclass image datasets.

Using a model like MobileNet V2 that is generally used for binary classification and modifying it work for multiclass datasets and setting appropriate loss function resulted in only a small increase in accuracy over the CNN we had built earlier. We used adam as its optimizer with categorical_crossentropy. The training loss and validation loss also diverged only after 3 epochs which meant that this model was overfitting for the data, we had without any way to fix it.

We now moved onto to ResNet models, both of which were reputed to be better at multiclass classification although at a higher computational cost. The model we chose was ResNet v2 50 with which we had problems in configuring it for a workable optimizer and loss function. Loss functions like sparse_categorical_crossentropy were using too much disk space which eventually resulted in a memory leak crashing our computer while we used categorical_crossentropy for the model. Only a Euclidean function was working with this model like Poisson or mean_squared_error which took far more time.

The final and the most accurate model we have chosen as our submission is ResNet v2 101 which results in a significant increase in accuracy over the previous models we had considered. By enabling sigmoid as activation function, adam as its optimizer and binary_crossentropy to

account for the multiclass dataset we have finally trained a model is accurate enough to be used in field work.

TECHNICAL DISCUSSION:

From the beginning, we split up the work to ensure that we all work in tandem. With the initial data preprocessing being done by Sandesh. He had to process the data into a directory system which makes it compatible to run with the various models we had chosen to build. He also built and trained the CNN which had several hyperparameters that required fine tuning. Sneha was responsible for visualizing the results of the models using matplotlib for the original and over sampled dataset after counting the number of images under each class. She also visualized after splitting into training, test and validation along with implementation of MobileNet V2 model on the augmented and oversampled dataset. Priya had to write the code that took the file system and split it up further into training, test and validation data sets. She also ran the ResNet v50 and solved the problems that arose when we had to train it. Sindhusha was responsible for augmenting the data before we ran it for all the models including CNN. She implemented oversampling to address the severe imbalance in classes in the data. This resulted in a marginal decrease in overall accuracy of the ImageNet models. From this we inferred that our image dataset was inclined to corrupt the models by overfitting features that arose from augmentation which are not present in the original dataset.

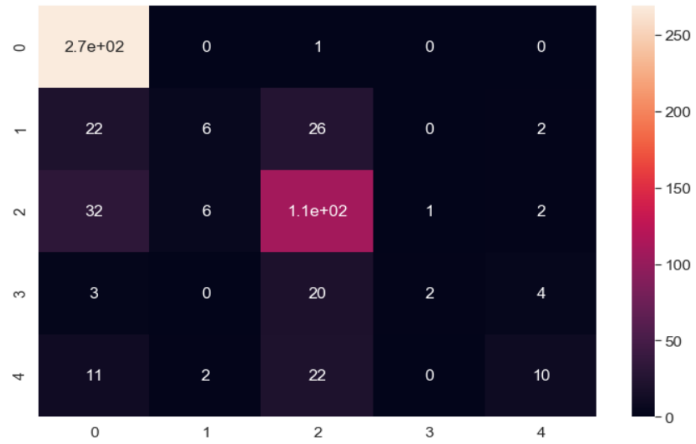
OBSERVED RESULTS:

From the above models that we had built for detecting Diabetic Retinopathy it becomes readily apparent that ResNet v101 is significantly more accurate than its closest competitor, ResNet v50. ResNet v101 achieves a maximum validation accuracy of 91% which is nearly 28% improvement over the validation accuracy of ResNet v50. These models work quite well on multiclass classifications and by enabling softmax, it is possible to work on datasets that contain upwards of 20 classes as well.

TEST RESULTS:

As ResNet v2 101 has the best training and validation accuracy, we implemented this model on our unseen test data which has 550 images. Once the model returned the predicted class labels, we compared them against the actual class labels to calculate the accuracy of the model and show the confusion matrix. The accuracy of the model on test data is 72% with a precision of 68.3% and a recall of 72% with all the class 0 being predicted correctly.

Visualization of confusion matrix:



CONCLUSION:

By working on this project, we have learnt some of the cardinal principles of doing machine learning first hand. Cleaning the data and formatting it such that it can be parsed by the models we would like to train, selecting models that are suitable for the task at hand, selecting the hyper parameters that give optimal results are some of the challenges we faced. Some models although are accurate require unrealistic resources to work, for example, we could not find the point of divergence for training and validation accuracy ResNet v50 and had to give up after processing 10 epochs since our computers could not handle anymore load. With the computational power at our disposal we built a model based on ResNet v101 that is highly accurate in real testing scenarios as well.

REFERENCES:

1. https://www.tensorflow.org/tutorials/images/classification
2. https://stackoverflow.com/questions/14270391/python-matplotlib-multiple-bars
3. https://stackoverflow.com/questions/45292833/using-split-function-on-a-pandas-dataframe
4. https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4
5. https://tfhub.dev/google/imagenet/resnet_v2_50/classification/4
6. https://tfhub.dev/google/imagenet/resnet_v2_101/feature_vector/4
7. https://www.kaggle.com/c/landmark-recognition-challenge/discussion/57716