Machine Learning Engineer Nanodegree

Capstone Project

M P S Vishal Singh Tomar
10 September 2019

I. Definition

Project Name

Detecting diabetic retinopathy: Detect diabetic retinopathy to stop blindness before it's too late.

Project overview

Imagine being able to detect blindness before it happened.

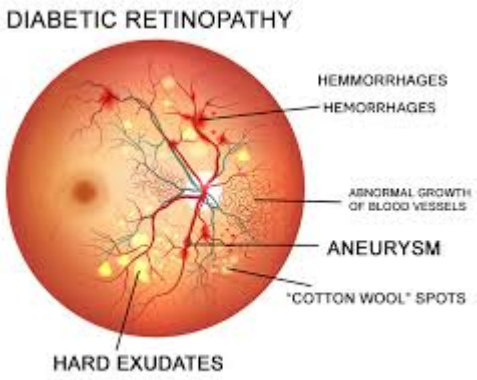
Millions of people suffer from diabetic retinopathy, the leading cause of blindness among working aged adults. This project aim to detect and prevent this disease among people living in rural areas where medical screening is difficult to conduct.

Currently, Technicians travel to these rural areas to capture images and then rely on highly trained doctors to review the images and provide diagnosis. Their goal is to scale their efforts through technology; to gain the ability to automatically screen images for disease and provide information on how severe the condition may be.

We will build a machine learning model to speed up disease detection. Working with thousands of images collected in rural areas to help identify diabetic retinopathy automatically. If successful, we will not only help to prevent lifelong blindness, but these models may be used to detect other sorts of diseases in the future, like glaucoma and macular degeneration.

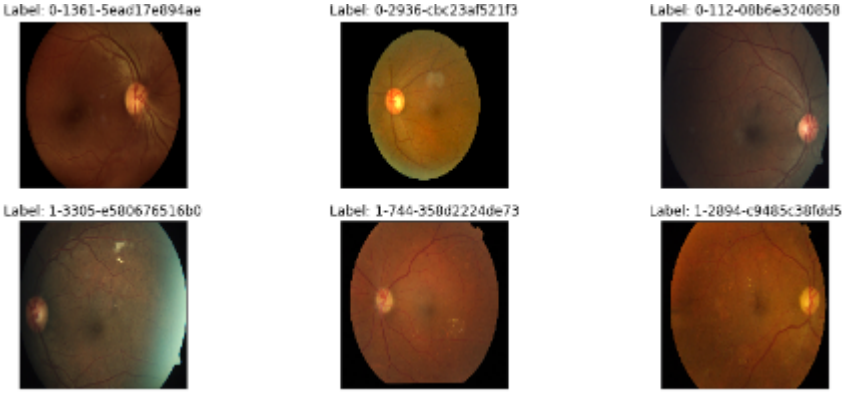
Problem Statement

Diabetic retinopathy (DR) is the fastest growing cause of blindness, with nearly **415 million** diabetic patients at risk worldwide. If caught early, the disease can be treated; if not, it can lead to irreversible blindness.
Lets Look at the following diagram to have a better understanding of the problem.



- There are at least 5 things to spot on.
- These five spots are not easily detectable.
- In fact some images in dataset have very poor lighting which makes it even more harder to spot these spots. These things make training a deep Learning model difficult.

The following figure is a screen shot of dataset images.



It is clear that images have poor lighting and nerves and spots which are to be checked are hardly visible.
We need a way to improve its lighting conditions and also highlight nerves, spots and try to remove other not so relevant features from the images so the our model can efficiently perform.

Metrics

I will use Log-loss function for performance analysis. Logarithmic Loss or log Loss, Works by penalizing the false classifications. It works well for multi-class classification. When working with Log Loss, the classifier must assign probability to each class for all the samples . Suppose, there are N samples belonging to M

classes, then the Log Loss is calculated as below:

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij})$$

Where,

N	*No of Rows in Test set*
M	*No of Fault Delivery Classes*
Y_{ij}	*1 if observation belongs to Class j; else 0*
P_{ij}	*Predicted Probability that observation belong to Class j*

Log Loss has no upper bound and it exists on the range [0, ∞). Log Loss nearer to 0 indicates higher accuracy, whereas if the Log Loss is away from 0 then it indicates lower accuracy.
In general, minimizing Log Loss gives greater accuracy for the classifier.

Log Loss is a Micro -average matrix.

Micro-average is preferable if there is a class imbalance problem.

In our sample data we cannot assume to have a balanced set of images for all the classes hence I prefer using Log Loss function.

II. Analysis

Data Exploration

This data set is taken from kaggle competitions it is publicly available as a part of APTOS 2019 Blindness Detection competition.

Dataset includes both images and csv files.

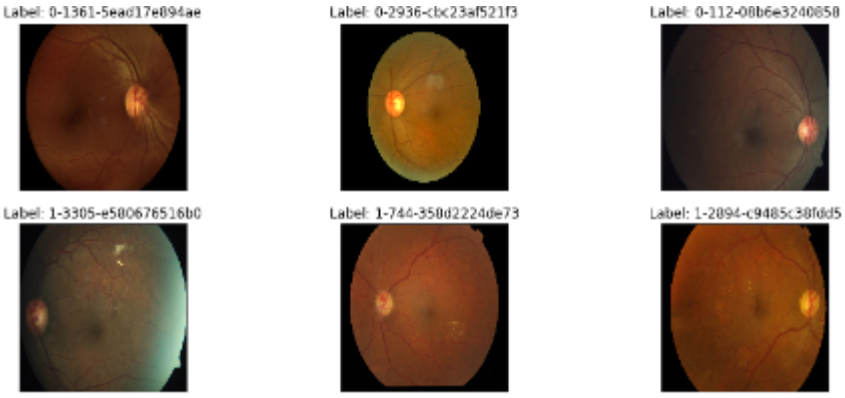
- csv files named as **train.csv** and **test.csv** contains
    1. id_code
    2. diagnosis

- **id_code** is same as the image file name i.e., Each row represents a image filename as **id_code** and what category it belongs to as **diagnosis**.

A clinician has rated each image for the severity of diabetic retinopathy on a scale of 0 to 4:

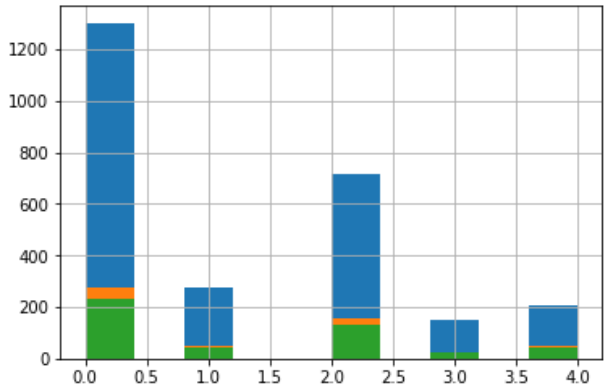
```
0 - No DR

1 - Mild

2 - Moderate

3 - Severe

4 - Proliferative DR
```

1. Below diagram shows image samples in the dataset to be used.



2. These are retinal photographic images.
3. These images indicate damage to eyes by showing blood clotting, dark spots or busted blood vessels.
4. These things are the root causes for blindness. Hence instead of tracking Patient's medical record data which will have many irrelevant parameters we can focus on these symptoms.
5. The dataset we use contains images which are classified into below 5 classes.
6. Number of images per classes are:
    a. Class 0 (No DR): 1805 images (49.2%)
    b. Class 1 (Mild): 370 images (10.1%)
    c. Class 2 (Moderate): 999 images (27.28%)
    d. Class 3 (Severe): 193 images (5.27%)
    e. Class 4 (Proliferative DR): 295 images (8.05%)
7. Hence we can conclude that dataset is largely imbalanced.
8. I will randomly split data into train, test and validation sets with 60%, 20% and 20% respectively.

Exploratory Visualization



The above graph shows the number of sample images in each category (0-4) Train (blue), test (yellow) and valid (green). It is clear that number of sample data is not balanced across the categories. Hence we have to carefully select the evaluation Metrics.

Algorithms and Techniques

- I have used Ben Graham's preprocessing method to improve lightning condition in the images.
- Auto image cropping method to crop out black background around the retina images.
- VGG16 pre trained model to extract bottleneck features of the images.

Benchmark Model

This project is still under research hence no solid papers have been published on benchmarks. Hence i would like to use the following notebook published publicly by KeepLearning as benchmark.

- This is a ResNet50 implementation with loss='categorical_crossentropy'. Applied on image size = 300.

III. Methodology

Data Preprocessing

- First we will load the image.
- Resize the images to 224 x 224 x 3 so that it matches the ImageNet format.
- Then we will use Ben Graham's preprocessing method [2] to improve the lightning condition. As mentioned in the above problem statement image clarity is poor due to dull lighting.
    - We will first try to sharpen image using GaussianBlur function.
    - Then crop the image to remove black space on the side.

Mix-up & Data Generator

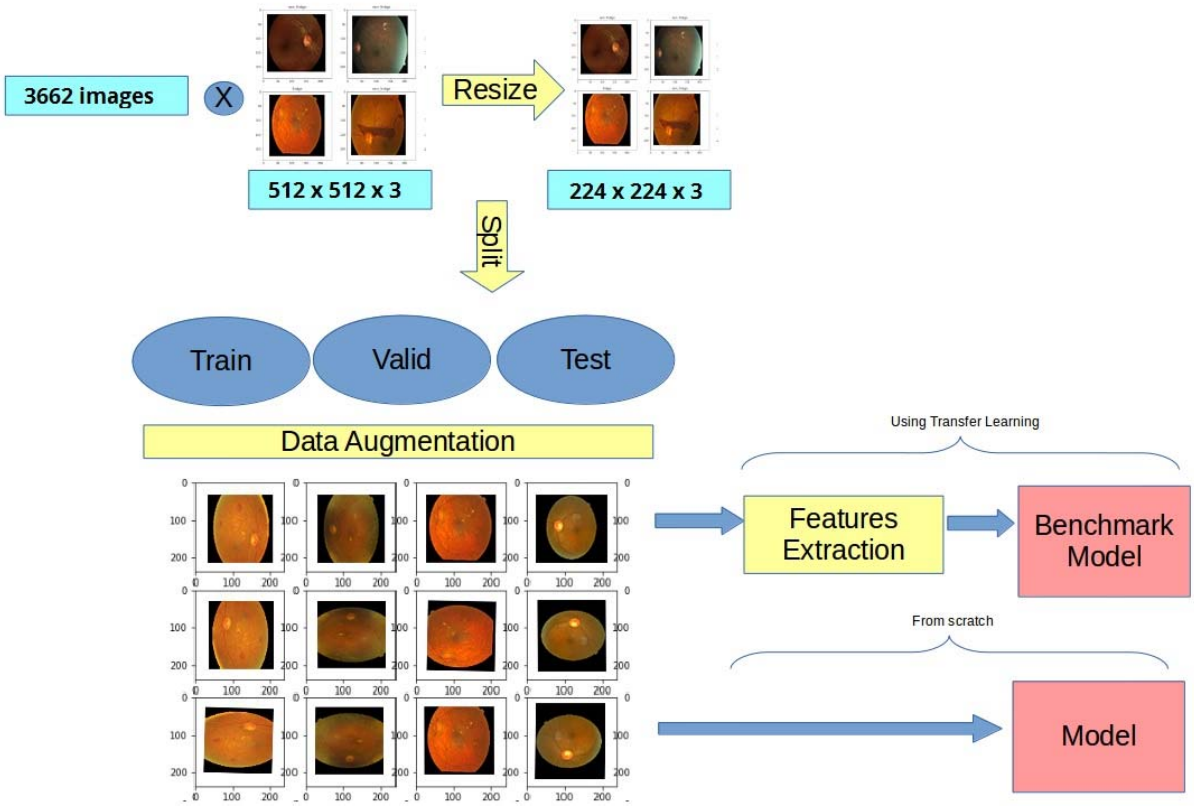
We will create a data generator that will perform random transformation to our datasets (flip vertically/horizontally, rotation, zooming). This will help our model generalize better to the data.

Implementation

I have used vgg16 pre trained model to create Bottleneck features of the given data. Then made a simple model (summary screenshot shared below) and Fed the obtained bottleneck features as input to this model.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
global_average_pooling2d_1 ( (None, 512)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              525312
_________________________________________________________________
dropout_2 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 5125
=================================================================
Total params: 530,437
Trainable params: 530,437
Non-trainable params: 0
_________________________________________________________________
```

Below figure demonstrates the implementation of the model



IV. Results

Model Evaluation and Validation

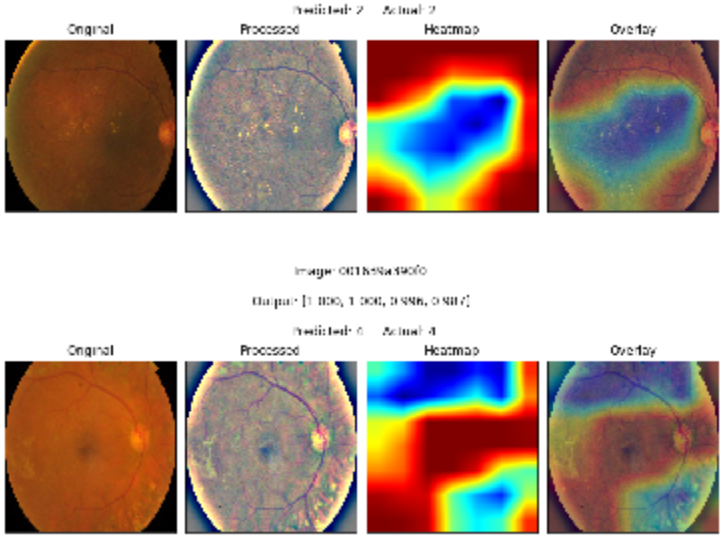
The final result obtained for the model on test data is loss=0.528 and accuracy of 79.81%. I used vgg16 model to create bottleneck features and then used it as input to my model.

Justification

The work published by KeepLearning on kaggle uses Resnet50 pretrained model. It gives an accuracy of 76.63 and loss = 0.697 on the test data it also took about 512 sec for each epoch on kaggle gpu.
I used vgg16 model instated of Resnet50 which improved the learning time a lot. It took around 62 sec for each epoch which is very less compared to resnet50 model. I have done some changes in preprocessing step which significantly improved the result. Obtained model's accuracy is 79.81 and loss= 0.528.

V. Conclusion

Free-Form Visualization

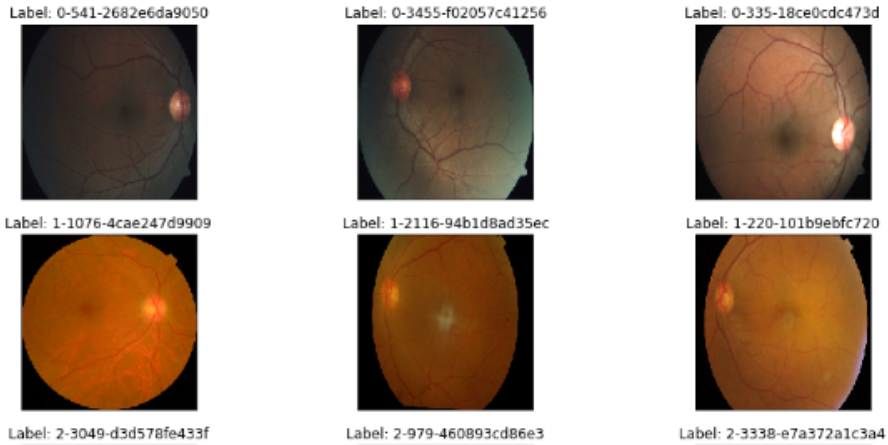


In this image we use heat map and overlay to visualize our predictive model's output.

- We see that in the fig1 there are some white spots in the middle which is reflected in the heat map and overly as blue or cooler area
- In the fig2 there are black spots in the middle which are marked as red area.
- This indicates that model is correctly able to differentiate between different features.

Reflection

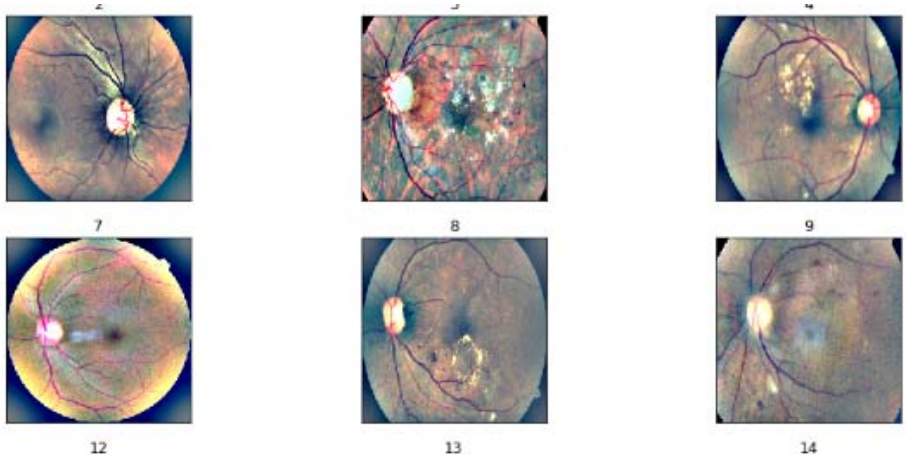
This project is based on image recognition; unlike other image recognition projects it deals an unusual set of images. There were many challenges faced during the pre processing of the images.



As you can see in the above figure

- Images were very dull and visibility was poor.
- Images had black background.
- Features are very hard to recognize.

I learned a lot about image processing through these project techniques such as Ben Graham's preprocessing method describes how to improve visibility by changing image gamma value and also auto cropping black background. Below figure shows the processed images



Improvement

I feel that there is still scope of improvement in the image preprocessing step, As these are retinal images we can try different preprocessing methods to improve the quality.

VI. References

[1] Oxford Applied and Theoretical Machine Learning Group : https://github.com/OATML/bdl-benchmarks/projects

[2] Ben Graham's preprocessing method : https://github.com/btgraham/SparseConvNet/tree/kaggle_Diabetic_Retinopathy_competition

[3] DenseNet: https://github.com/liuzhuang13/DenseNet

[4] Log Loss Matrix : https://datawookie.netlify.com/blog/2015/12/making-sense-of-logarithmic-loss/