Machine Learning Engineer Nanodegree

Capstone Project

Winfred Selwyn Ooh

March 27th 2018

Project Overview

Diabetic Retinopathy screening with a robust machine learning modelon datasets of retinal fundus photographs.

Human clinicians identify Diabetic Retinopathy (**DR**) with the presence of lesions associated with vascular abnormalities. If left untreated, the disease leads to the patient’s blindness. With an increasing segment of the population suffering from diabetes worldwide, an effective method for screening and early diagnosis is required to avoid burdening available medical infrastructure.

A large dataset of colour retinal fundus photographs is provided by **EyePACS** for this project.

Problem Statement

With the steady increase of individuals suffering from diabetes (in 2008, 180 million people worldwide, estimated to go up to 360 million by 2030 - World Health Organization), detecting DR places a burden on existing medical infrastructure.

Detecting DR manually is a time-consuming process requiring a trained clinician to examine and evaluate digital colour fundus photographs of the retina.

Treatment is usually delayed as patients suffering from deteriorating eyesight and DR are unaware they have diabetes.

With deeep learning models, the proccess of dianosis can be largely automated.

The clinician should be able to feed photos into the models and get immediate feedback on the patient’s DR level of severity, ranging on a scale of 0 – 4.

Metrics

The trained neural network generates a continuous number between 0 and 1 for referable diabetic retinopathy and other diabetic retinopathy classifications, corresponding to the probability of that condition being present in the image.

The second operating point corresponded to a sensitivity of 97% for detecting referable diabetic retinopathy because a high sensitivity is a prerequisite in a potential screening tool.

Initially, during the proposal phase of the project, the proposed metrics were a **Kappa Score** and accuracy.

The metrics used for project execution will be an **F1 Score** and **accuracy.** A team from **Google Research** released a paper on DR Detection with Tensorflow and measured the accuracy of the **InceptionV3** model based on an F1 score.

Links:

- Video: https://www.youtube.com/watch?v=oOeZ7IgEN4o
- Journal : https://jamanetwork.com/journals/jama/fullarticle/2588763
- Blog Post : https://research.googleblog.com/2016/11/deep-learning-for-detection-of-diabetic.html

A benchmark vanilla model with 4 layers will be used to provide a benchmark F1 and accuracy score.

The final model will built on **InceptionV3**, which is an approach similar to that taken by the Google Research team.

Data Exploration

(*please visit Exploratory Data Analysis.ipyn*b)

A dataset of 32,796 images have been provided by EyePAC on Kaggle for training, divided into the following classes :

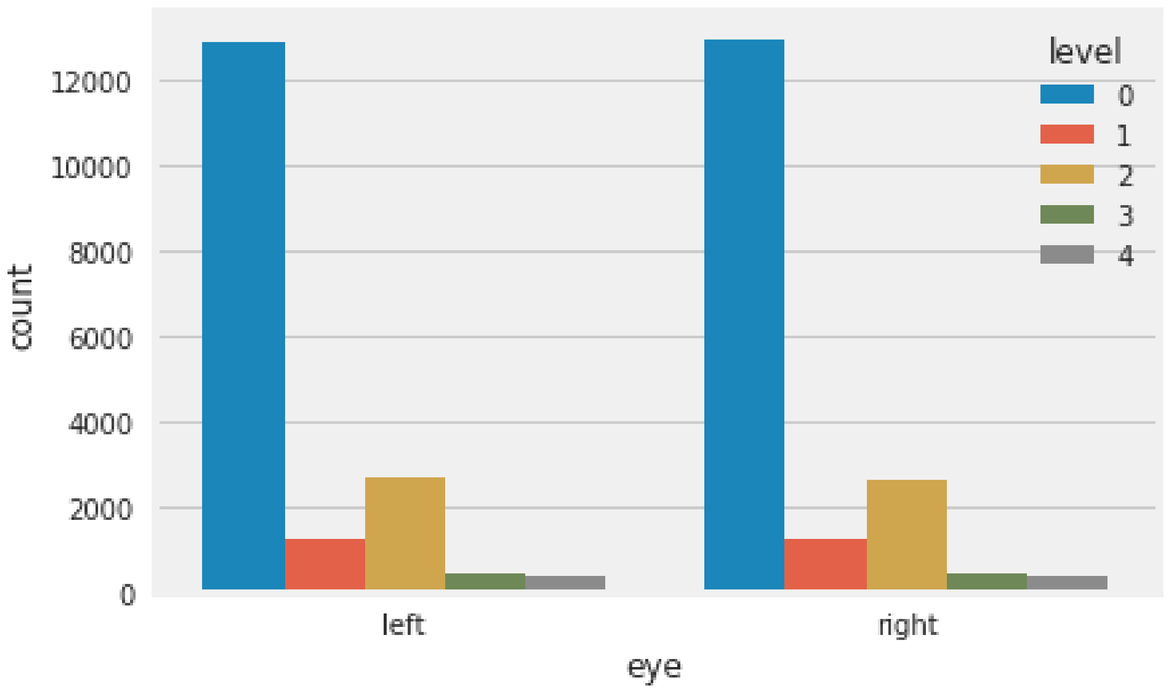
- Class 0, no diabetic retinopathy: 21978
- Class 1, mild diabetic retinopathy: 2078
- Class 2, moderate diabetic retinopathy: 4568
- Class 3, severe diabetic retinopathy: 770
- Class 4, proliferative diabetic retinopathy: 606

Frequency:

- Class 0: 0.734783
- Class 1: 0.150658
- Class 2: 0.069550
- Class 3: 0.024853
- Class 4: 0.020156

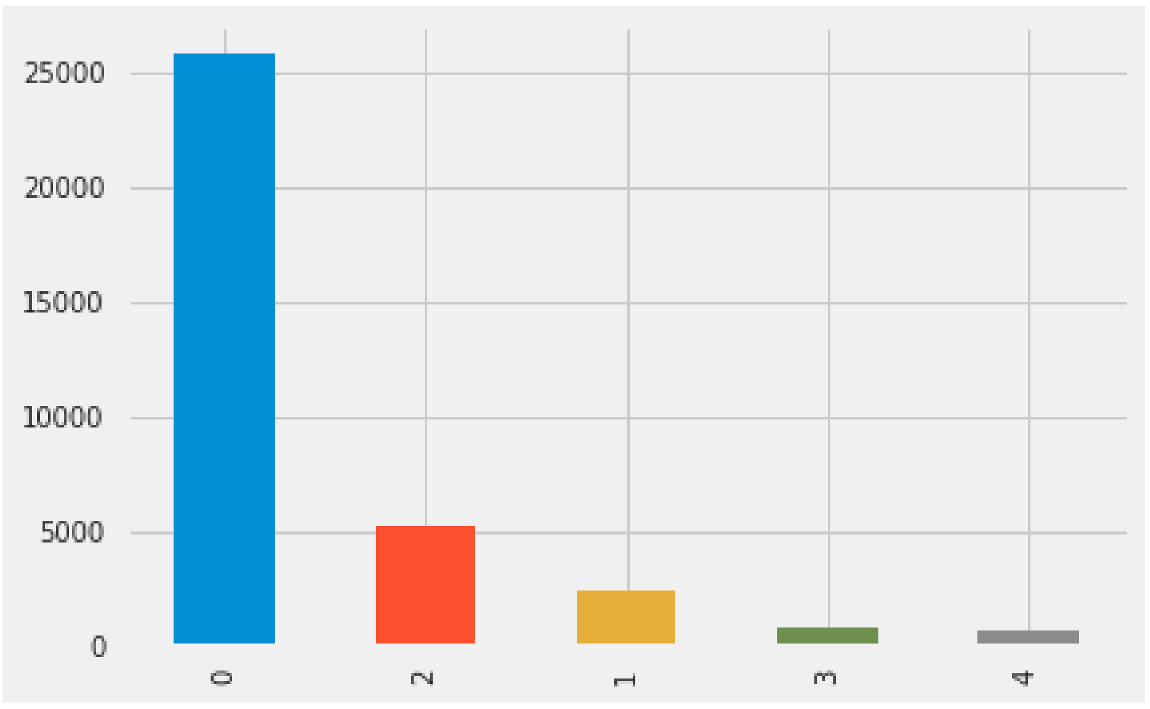
(*continued overleaf)*

The ratio of left:right eye images is **1:1**.



Exploratory Visualization
(*please visit Exploratory Data Analysis.ipyn*b)

The image count by class is as follows:



The image ratios will lead to imbalance if the dataset is used without balancing class weights. The majority of images in the training set are classified as 0 (79.5%).

Balancing is done by fitting the model with the following code:

- class_weight={0: 1, 1: 10.6, 2: 4.6, 3: 30, 4: 37}

The mean of colours by channel for sample images are (R:94, G:71, B:55).

Some points to note:

- Individual-level data including age and sex were not available for the dataset,

- Unique patient codes (deidentified) were unavailable for the dataset,

Algorithms and Techniques

This problem is a multiclass problem which will be solved with a convnet, a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery. (source: https://en.wikipedia.org/wiki/Convolutional_neural_network)

The network will be trained on 30, 000 training images and validated with 5120 validation images.

Keras with a Tensorflow backend will be used for the benchmark model (*see Benchmark model – CNN.ipynb*) and the transfer learning model (*see InceptionV3.ipynb*).

Benchmark

The final InceptionV3 model will be trained and use to classify **53,576** images. The final output can be found in the submission folder:

- submission.csv

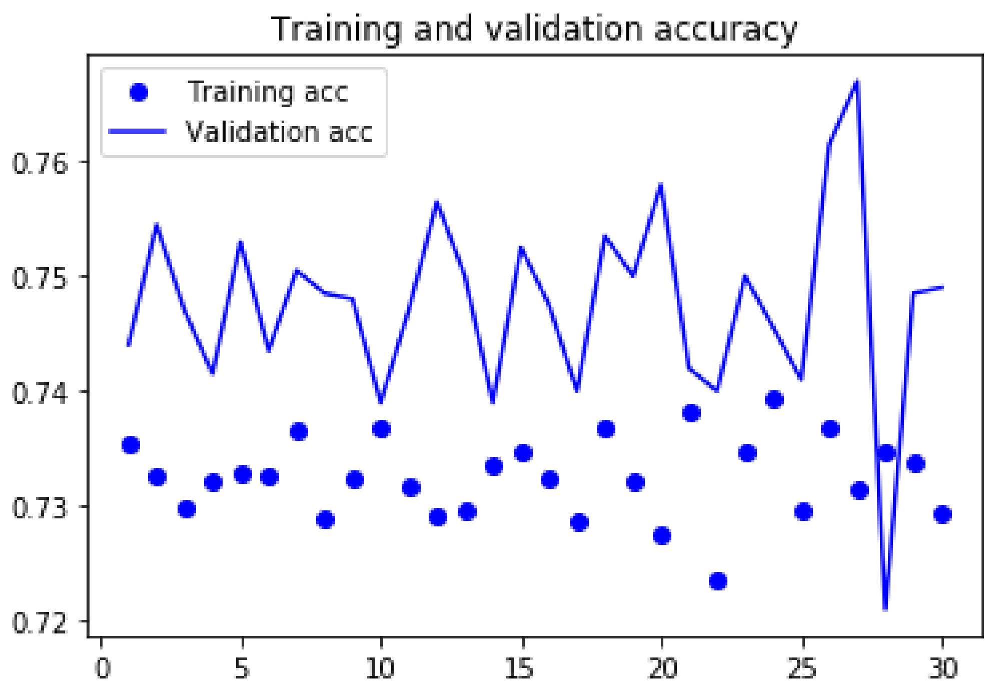
The best scores from the benchmark model is:

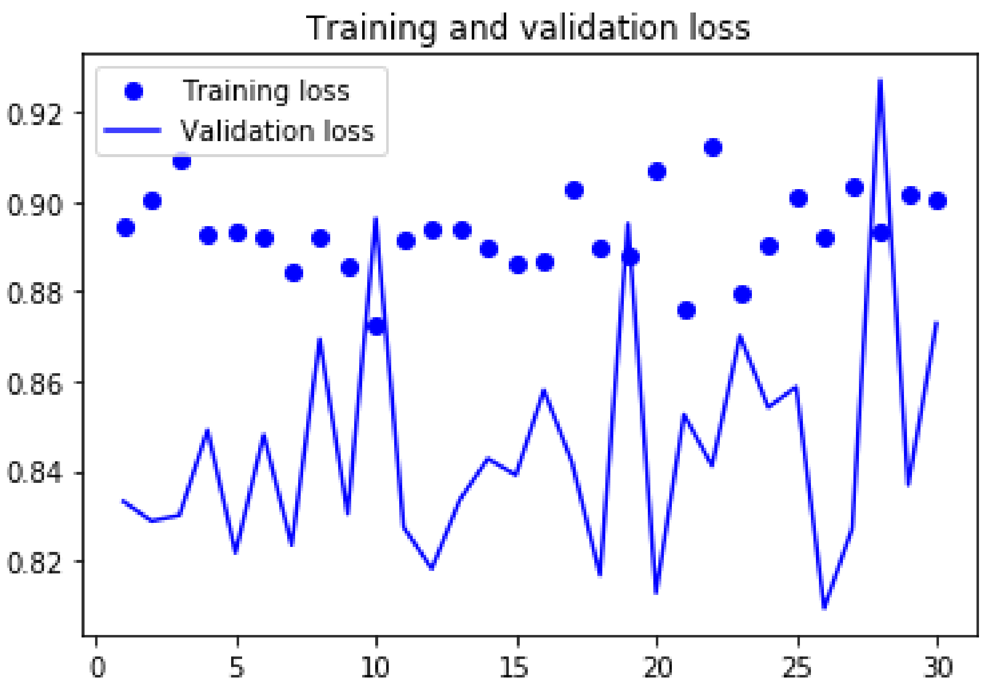
TRAINING

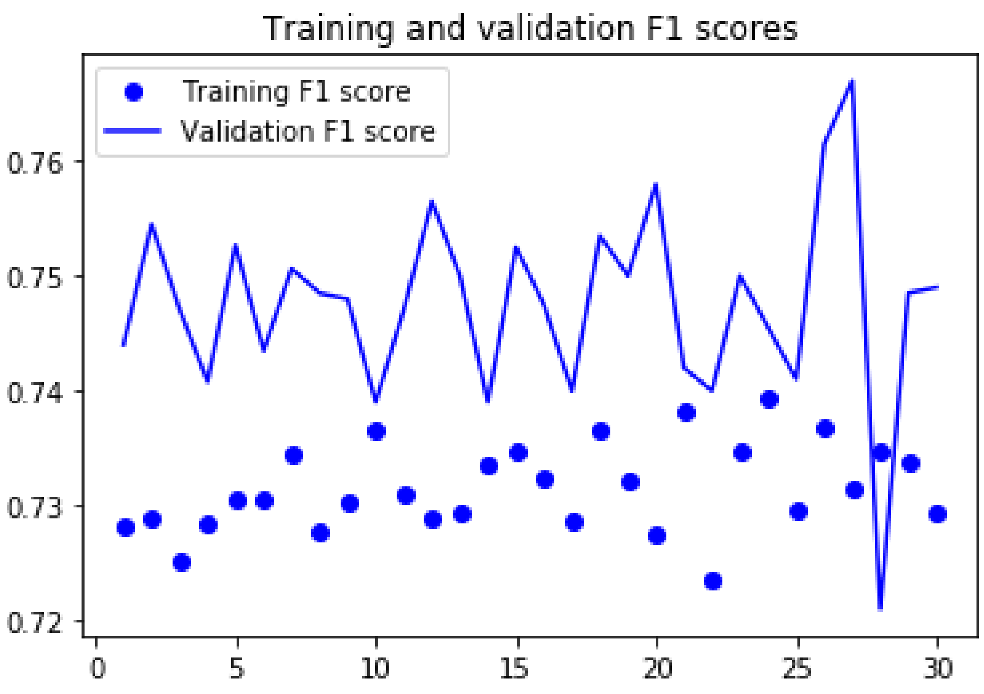
- accuracy: 0.7394
- loss: 0.8724
- **F1: 0.7394**

VALIDATION

- accuracy: 0.7670
- loss: 0.8094
- **F1: 0.7670**




Training and validation loss
Training loss
Validation loss
0.92
0.90
0.88
0.86
0.84
0.82
0
5
10
15
20
25
30


Training and validation F1 scores
Training F1 score
Validation F1 score
0.76
0.75
0.74
0.73
0.72
0
5
10
15
20
25
30

<u>**Data Pre-Processing**</u>

Images in the dataset:

- vary in size and resolution
- have black borders
- are potentially noisy and with artifacts

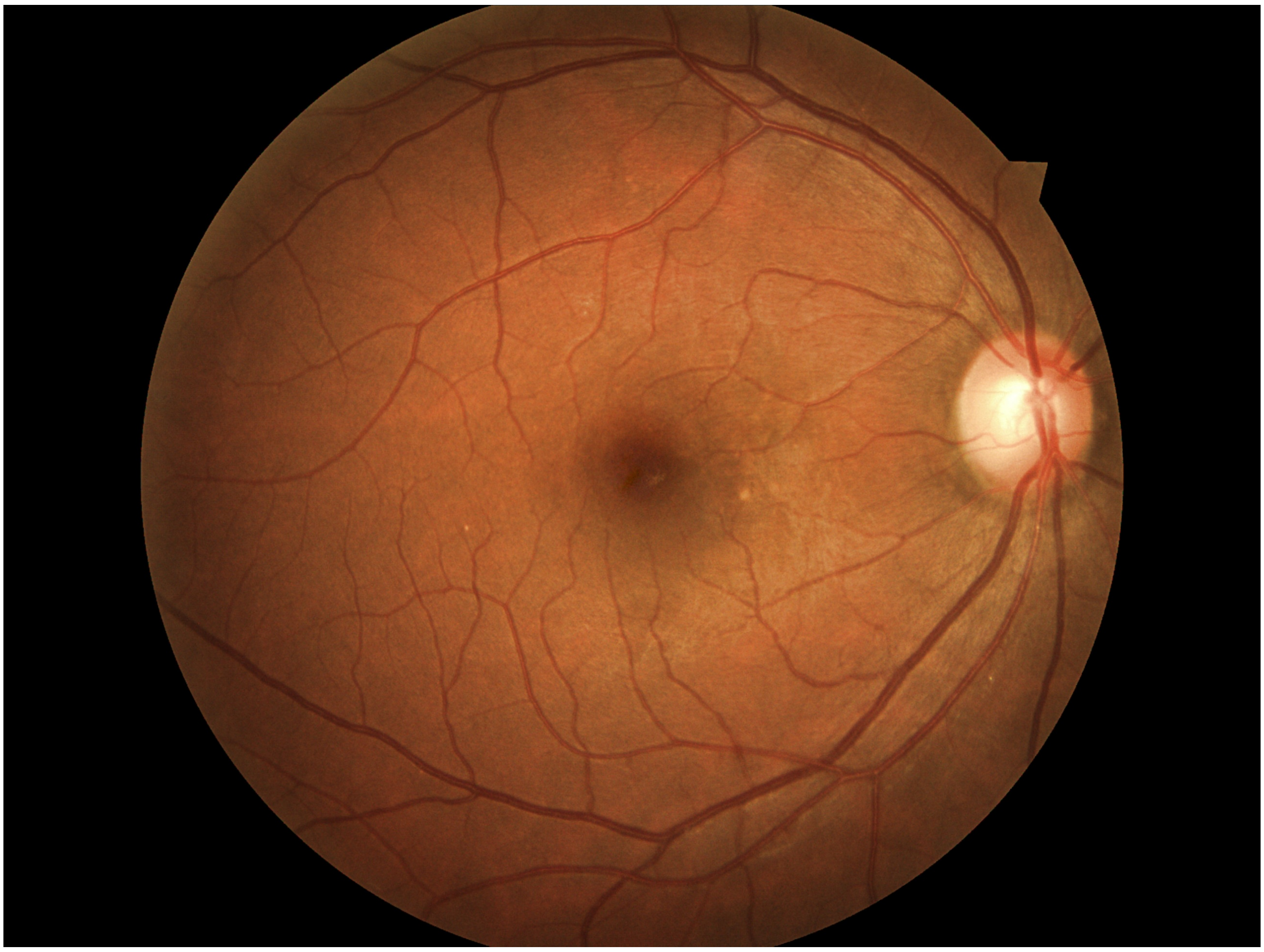


Image processing will be done with a modified script from Tensorflow Model - Research/Slim library:

- https://github.com/tensorflow/models/tree/master/research/slim/datasets

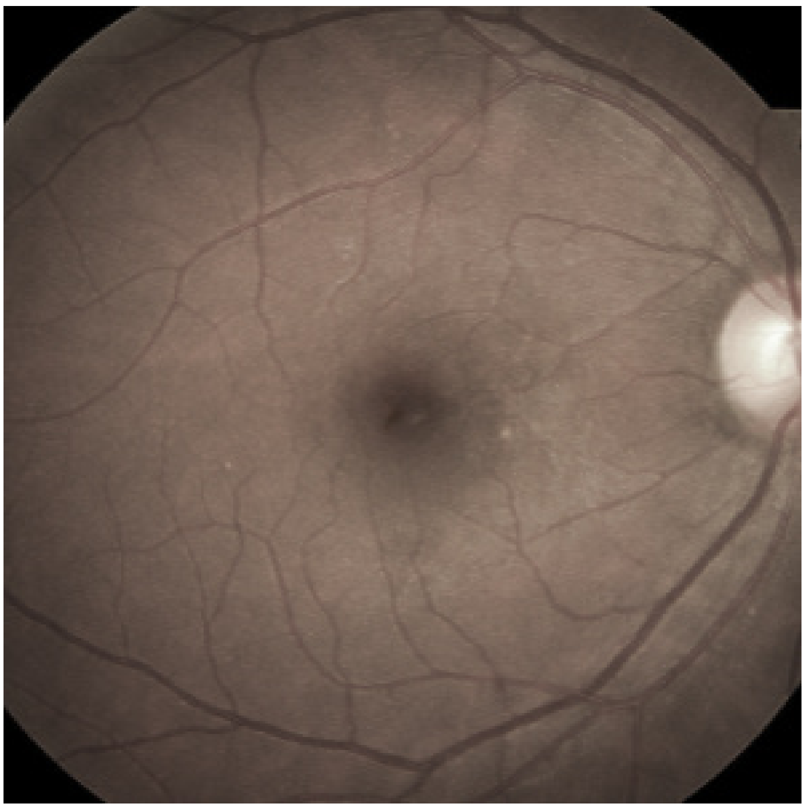
All necessary scripts are provided with the submission folder:

- data_utils.py
- download_and_convert.py
- dataset_utils.py
- download_and_convert_diabetic.py

Images will undergo transformations in the following order:

1. denoising (Median filter)
2. sharpening (Gaussian filter)
3. cropping
4. resize to **299 x 299**

Below is an example of a fundus image after pre-processing.



Implementation

Benchmark model

The initial solution is a CNN model with the following architecture:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 148, 148, 128)     3584
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 146, 146, 128)     147584
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 73, 73, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 71, 71, 64)        73792
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 35, 35, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 33, 33, 64)        36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 16384)             0
_________________________________________________________________
dropout_1 (Dropout)          (None, 16384)             0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               8389120
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_4 (Dense)              (None, 5)                 645
=================================================================
Total params: 8,815,877.0
Trainable params: 8,815,877.0
Non-trainable params: 0.0
_________________________________________________________________
```

While it’s accuracy score on the validations set is 76%, it classified every image as ***Class 0, No Diabetic Retinopathy.***

The results from the first model will be discounted as it was unable to classify any of the images few into the model.

InceptionV3 model

The InceptionV3 model has the following architecture:

```
Layer (type)                 Output Shape              Param #
=================================================================
inception_v3 (Model)         (None, 2048)              21802784
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 10245
=================================================================
Total params: 21,813,029.0
Trainable params: 21,778,597.0
Non-trainable params: 34,432.0
_________________________________________________________________
```

Model Evaluation and Validation

The InceptionV3 model was chosen after reading the paper published by the Google Research team. The model produced prediction for all of the images in the testing set.

The best scores from the InceptionV3 model is:

TRAINING

- accuracy: 0.4215
- loss: 4.4361
- **F1: 0.4048**

VALIDATION

- accuracy: 0.7661
- loss: 1.1813
- **F1: 0.7634**

However, as this problem's solution is in the domain of medial diagnosis, I wouldn't trust the model without extensive testing and an accuracy of at least 90% and above.

Justification

Results were definitely stronger than those produced by the benchmark model.

Possible solutions for increasing the accuracy of the model as well as practical solutions in real-world usage.

<u>Conclusion and improvements</u>

<u>Reflection</u>

After exploratory data analysis and model benchmarks, I've spent a few days trying to get results with VGG19 and ResNet50.

VGG19 and ResNet50 were not producing consistent F1 scores or better accuracy than the benchmark model.

After additional research and uncovering a paper on a similar project(https://jamanetwork.com/journals/jama/fullarticle/2588763) with a Tensorflow backend by the Google Research team, I settled on InceptionV3.

<u>Improvements</u>

1. Use an optimal image resolution – testing with various image resolutions and using the resolution that produces the best results with InceptionV3.

2. Use all the available image information - the pre-processing stage of this project is key to getting good results. Perhaps certain features in fundus images could be emphasized for easier detection.

3. Use QWK as a loss function - for multi-class classification the standardized loss function to use is the logarithmic loss [19]. In [9] is shown that for ordinal regression problems, where not only a multi-class classification is taking place but also there is possible to establish a sorting of the classes based on some hidden underlying causes, QWK-loss can also be used with better results. (source: https://arxiv.org/pdf/1712.08107.pdf)

4. Use a efficient number of features – the model should be run with better resources so a bigger network can be used.