Machine Learning Engineer Nanodegree Program

Capstone Proposal

STEM NGUYEN

Blindness Detection (Kaggle Competition)

Domain Background

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. Aravind Eye Hospital in India hopes to detect and prevent this disease among people living in rural areas where medical screening is difficult to conduct. Successful entries in this competition will improve the hospital's ability to identify potential patients. Further, the solutions will be spread to other Ophthalmologists through the 4th Asia Pacific Tele-Ophthalmology Society (APTOS) Symposium.

Currently, Aravind 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.

References

https://www.technology.org/2019/07/15/aptos-2019-blindness-detection/

https://www.kaggle.com/c/aptos2019-blindness-detection

https://towardsdatascience.com/aptos-2019-blindness-detection-520ae2a4acc

Problem Statement

This is a deep learning problem. Inputs are the images and the goal is to predict severity of diabetic retinopathy on a scale of 0 to 4:

0 - No DR

1 - Mild

2 - Moderate

3 - Severe

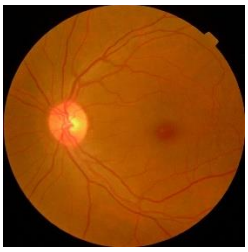
4 - Proliferative DR

Datasets and Inputs

The datasets are provided on Kaggle competition website (https://www.kaggle.com/c/aptos2019-blindness-detection). They are free to download.

Datasets are **3662** color images with png format.

Example:



Dataset files

train.csv - the training labels

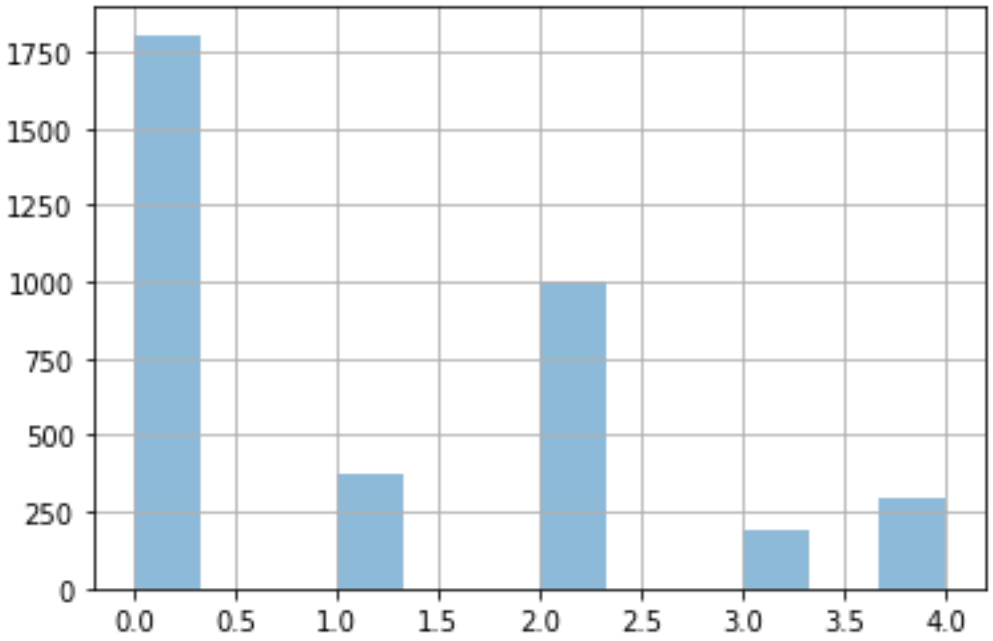
test.csv - the test set (you must predict the diagnosis value for these variables)

sample_submission.csv - a sample submission file in the correct format

train.zip - the training set images

test.zip - the public test set images

Dataset is imbalanced



Solution Statement

I used OpenCV to preprocess the images. OpenCV loads the images and resize the images to 224x224.

After that I used the image-classifiers (https://github.com/qubvel/classification_models) to classify the images.

Benchmark Model

Reference - https://github.com/BloodAxe/Kaggle-2019-Blindness-Detection

Solutions	Accuration	Data
Resnet18	0.9077 ± 0.0045 (Cross validation 4 folds)	3662 images
SEResnext50	0.9213 ± 0.0033 (Cross validation 4 folds)	3662 images
SEResnext101	0.9176 ± 0.0080 (Cross validation 4 folds)	3662 images
reg_resnext50_rms	0.9234 ± 0.0035 (Cross validation 4 folds)	3662 images

Evaluation Metrics

Precision

Let's start with precision, which answers the following question: what proportion of predicted Positives is truly Positive?

Precision = (TP)/(TP+FP)

In the asteroid prediction problem, we never predicted a true positive.

And thus precision=0

Recall

Another very useful measure is recall, which answers a different question: what proportion of actual Positives is correctly classified?

Recall = (TP)/(TP+FN)

In the asteroid prediction problem, we never predicted a true positive.

And thus recall is also equal to 0.

F1 Score

The F1 Score is the 2*((precision*recall)/(precision+recall)). It is also called the F Score or the F Measure. Put another way, the F1 score conveys the balance between the precision and the recall.

Project Design

Data Visualization: Tried multiple predictors, each predictor I separate the dataset based on k-fold Cross-Validation with 80% for training and 20% for testing.

Data Preprocessing: Resize the images to 224x224 for inputs. Change the color from BGR to RGB. Scaling and Normalization operations on data and splitting the data in training, validation and testing sets.

Feature Engineering: Tried multiple predictors and take the best one.

Model Selection: Experiment with various algorithms to find out the best algorithm for this use case. I will try with many models Resnet18, SEResnext50, SEResnext101 and reg_resnext50_rms, after that I will pick the best one with the highest accurate.

Model Tuning: Fine tune the selected algorithm to increase performance without overfitting.

Testing: Test the model on testing dataset.