Machine Learning Engineer Nanodegree

Capstone Project

Aditya Mehta
August 10th, 2019

I. Definition

Project Overview

This capstone project is for Healthcare domain and focussed on detecting blindness beforehand using eye images. This is provided as a competition on Kaggle by Aravind Eye Hospital in India- https://www.kaggle.com/c/aptos2019-blindness-detection/overview.

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

I am motivated to solve this problem statement as it would help in improving the lives of people by enabling proactive actions to prevent blindness.

Here's one academic article with application of Machine Learning for detection of Diabetic Retinopathy in Retinal Fundus Photographs-
https://jamanetwork.com/journals/jama/fullarticle/2588763

Problem Statement

The objective of this competition is to detect diabetic retinopathy using images which Aravind technicians capture from rural areas of India.

Current State- Currently highly trained doctors review these images and provide diagnosis

Desired State- Scale the doctors efforts through technology by automatically screening the images for disease and provide information on how sever the condition may be.

This is a problem to classify status of disease into four classes based on retinal image.

Metrics

Scoring metric is 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). In the event that 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.

II. Analysis

Data Exploration

We are provided with a 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

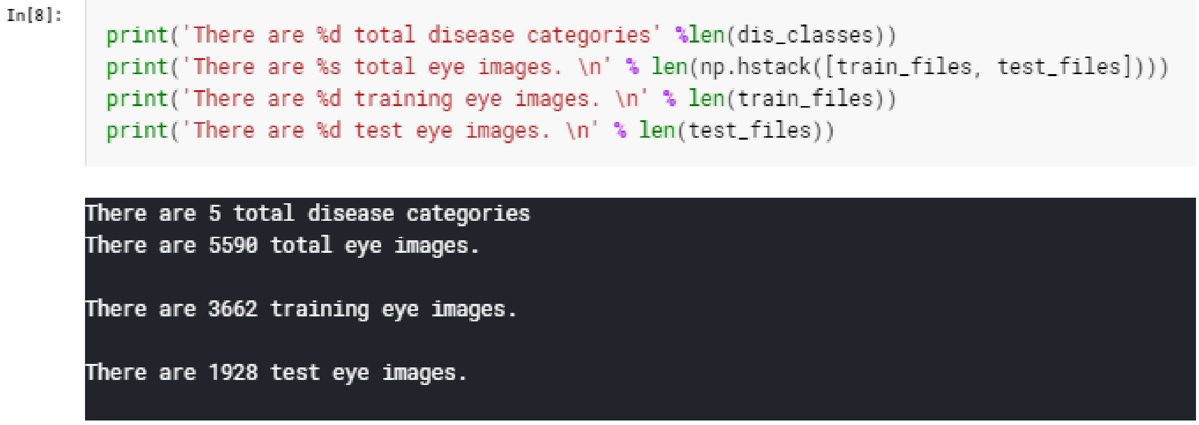
3 - Severe

4 - Proliferative DR

Like any real-world data set, we will encounter noise in both the 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 of time, which will introduce further variation.

How are the images formatted? Are there color layers? Are the image dimensions consistent?

There are 3662 images which have been provided for training the model and 1928 images in test dataset. Of these 3662 images, below is the distribution into the five classes-



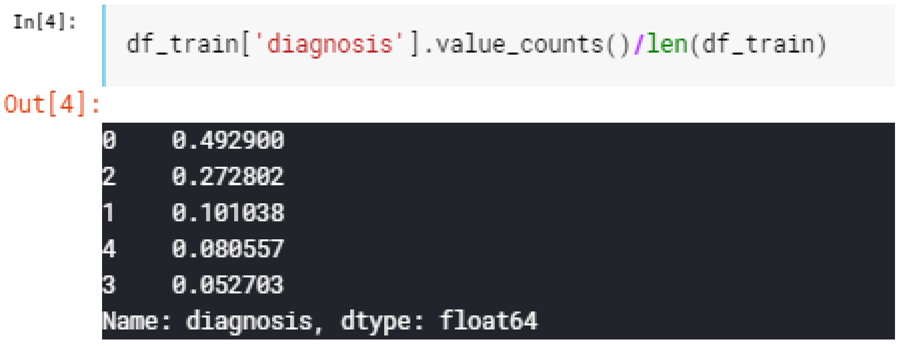
In[8]:

```
print('There are %d total disease categories' %len(dis_classes))
print('There are %s total eye images. \n' % len(np.hstack([train_files, test_files])))
print('There are %d training eye images. \n' % len(train_files))
print('There are %d test eye images. \n' % len(test_files))
```

```
There are 5 total disease categories
There are 5590 total eye images.

There are 3662 training eye images.

There are 1928 test eye images.
```



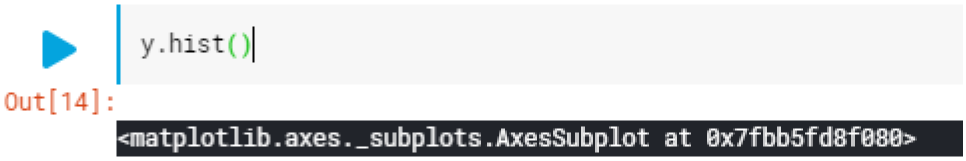
In[4]:

```
df_train['diagnosis'].value_counts()/len(df_train)
```

Out[4]:

```
0    0.492900
2    0.272802
1    0.101038
4    0.080557
3    0.052703
Name: diagnosis, dtype: float64
```

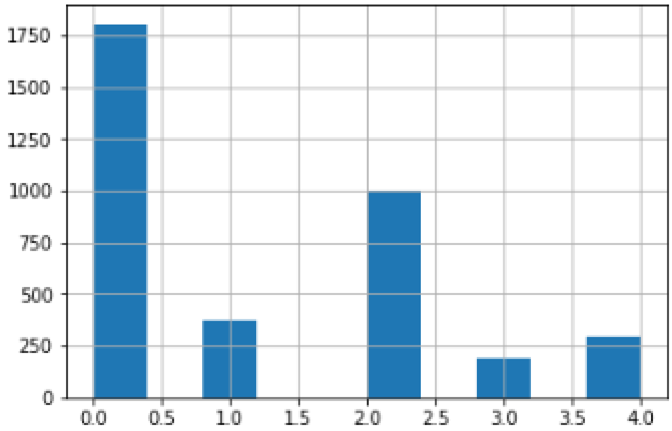
Exploratory Visualization

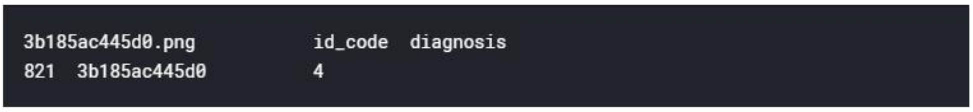


```
y.hist()
```

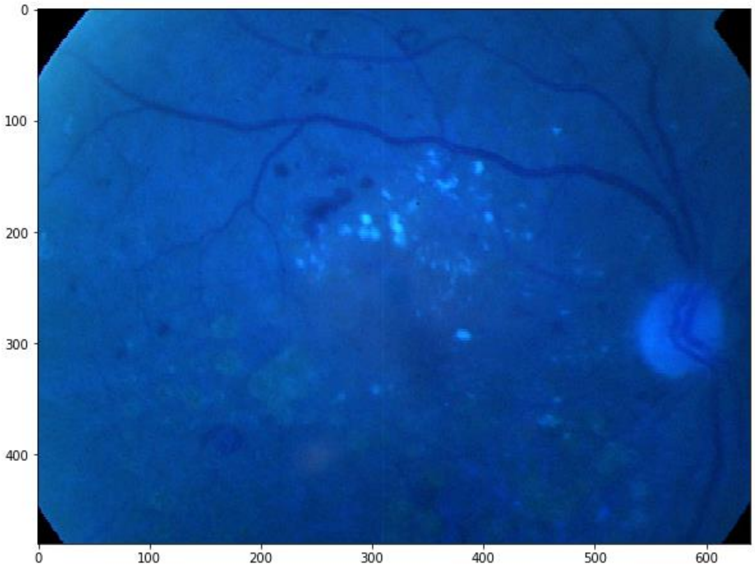
Out[14]:

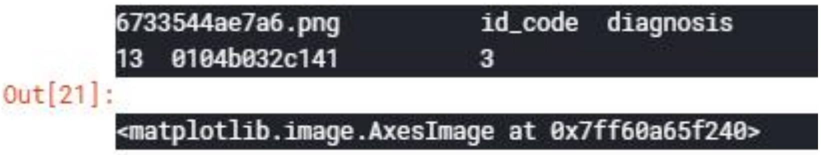
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb5fd8f080>
```





```
3b185ac445d0.png          id_code   diagnosis
821  3b185ac445d0          4
```

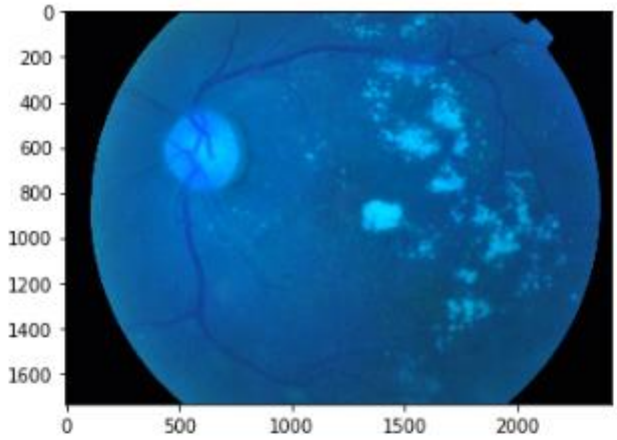


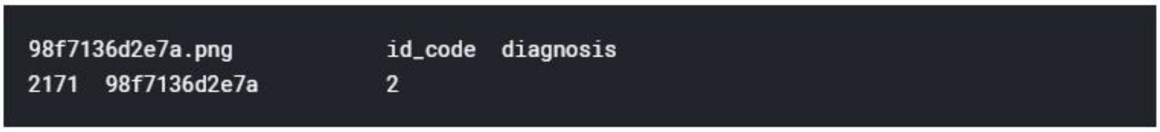


```
6733544ae7a6.png           id_code  diagnosis
13  0104b032c141           3
```

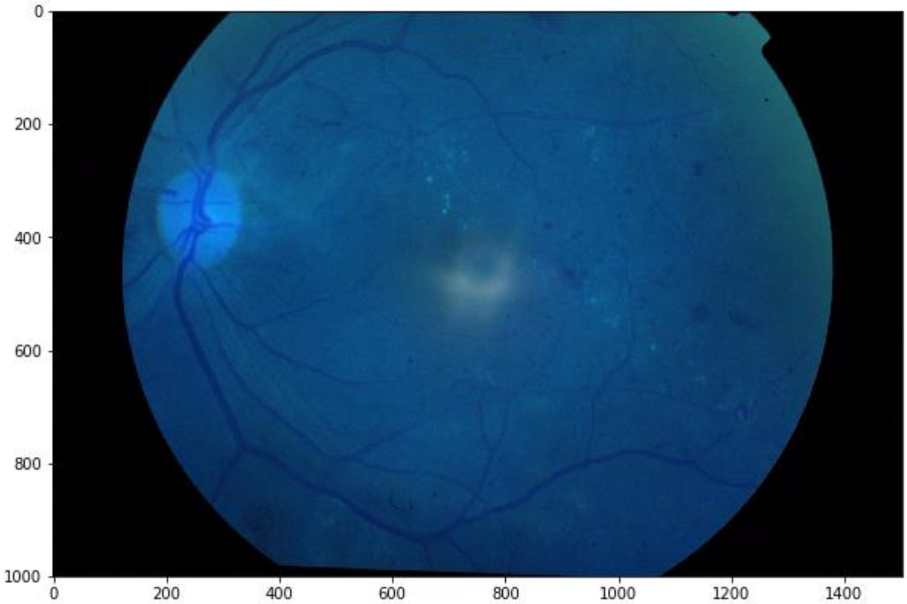
Out[21]:

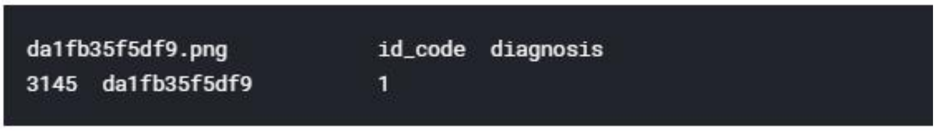
```
<matplotlib.image.AxesImage at 0x7ff60a65f240>
```



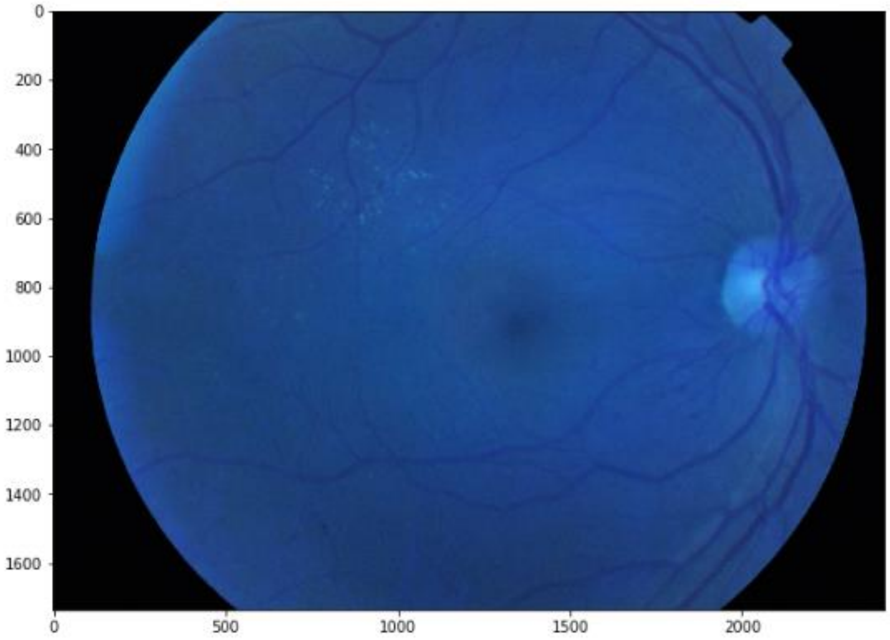


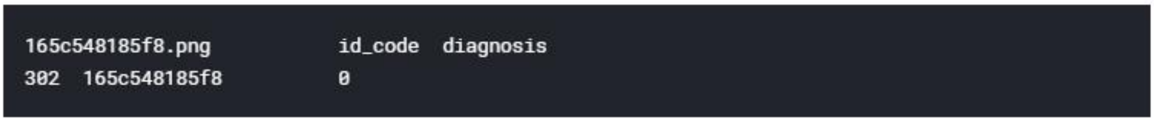
```
98f7136d2e7a.png           id_code  diagnosis
2171  98f7136d2e7a           2
```



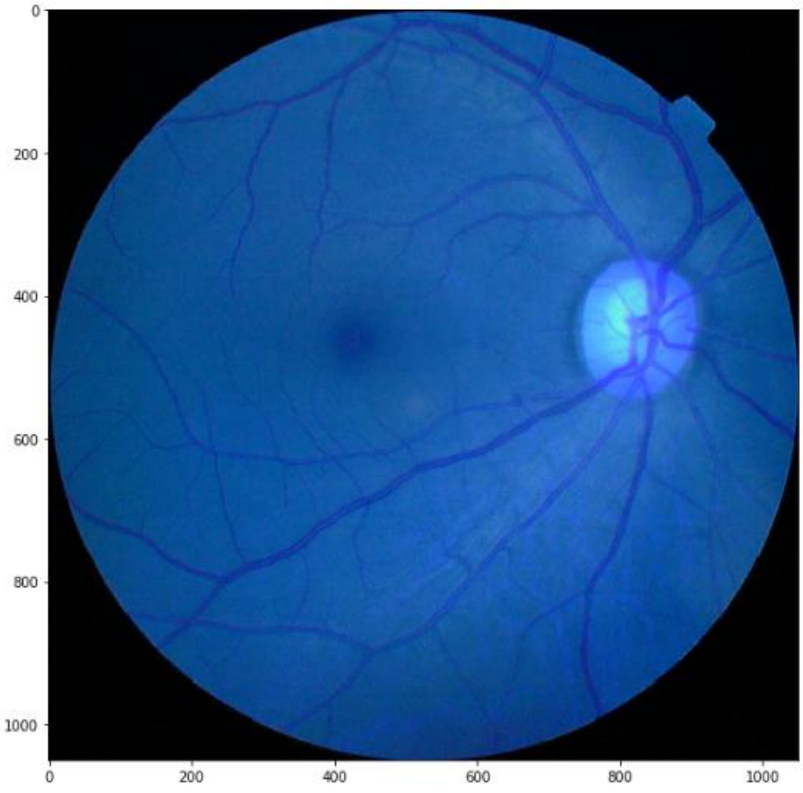


```
da1fb35f5df9.png           id_code  diagnosis
3145  da1fb35f5df9           1
```





```
165c548185f8.png           id_code  diagnosis
302  165c548185f8            0
```



For severity level 4 and 3, the example indicates patches in the retina. Level 2 is slightly lesser patches while there seem to be no patches in Level 0.

Algorithms and Techniques

The base classifiers are Logistic Regression and RandomForest Classifier. The next is Convolutional 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. The algorithm outputs an assigned probability for each class.

The following are some of the parameters which can be tuned to optimize the classifiers:

- Training parameters for RandomForest
    - Max Depth
    - Minimum number of observations for splitting a node
    - Number of processors to use
    - Minimum number of observations in leaf node
- Training parameters CNN
    - Learning rate
    - Number of epochs
    - Batch size (how many images to look at once during a single training step)
    - Optimizer- For eg. Adam, Ada, Adagrad etc
    - Evaluation Metric- categorical crossentropy
    - Learning rate (how fast to learn; this can be dynamic)
- Neural network architecture
    - Number of layers
    - Layer types (Dense, Convolutional , fully-connected, pooling etc)
    - Layer parameters (see links above)
    - Pre-processing parameters (see the Data Pre-processing section)

The Kaggle kernel used for the base_model does not leverage GPU while the CNN_keras has GPU enabled..

Benchmark

The benchmark for me would be a multiclass logistic regression model. I also compared my results with some of the other participants' results to get an understanding of how accurate my model is.

III. Methodology

Data Preprocessing

The preprocessing done in the “Base_Model” notebook leverages cv2 HuMoments and Histogram feature extraction-

- First step is to read all of the images using cv2 in color
- Extract and flatten Hu Moments features
- Calculate histograms by creating 8 bins
- Create global features with the above two arrays
- Apply MinMax scaling on the global features

The preprocessing done in the “CNN_Keras” notebook leverages ImageDataGenerator class from Keras and consists of the following steps:

- First step is to read all of the images using cv2 in color
- Resize all the images to 150 * 150 pixels
- Dividing the images array by 255 to normalize them between 0-1 range
- Images and labels are divided into training and validation sets
- Using ImageDataGenerator with below parameters
    - featurewise_center=True
        - Transforms the images to 0 mean
    - featurewise_std_normalization=True
        - Standardize pixel values across the entire dataset
    - rotation_range=20
        - a range within which to randomly rotate pictures
    - width_shift_range=0.2
        - Range within which to randomly translate pictures vertically or horizontally
    - height_shift_range=0.2
    - horizontal_flip=True
        - Randomly flipping half of the images horizontally to ensure that patches on either side of the eye could impact eye health

Reference- https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

Implementation

The implementation process can be split into two main stages:

1. The Base Model using Multiclass Logistic regression and RandomForest Classifier in Base_Model notebook

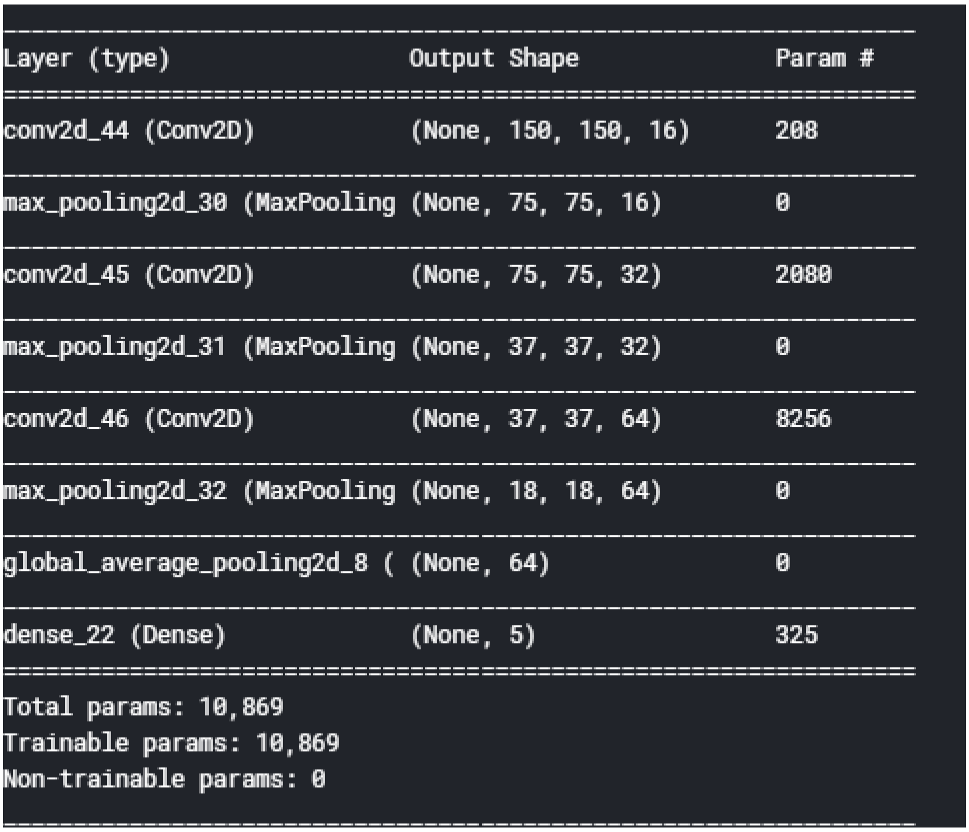
2. The CNN Model using Convolutional Neural Network in CNN_Keras notebook

Below are the steps in the first model-

- Create and extract Hu Moments features for shape matching to determine if the shape of the eye impacts diagnosis
- Extract Histogram features from the images
- Create and scale global features with the previously created features
- Apply Label Encoder on the dependent variable
- Create and Fit Logistic Regression and Random Forest Classifier on the data-
- Visualize the learning curves for the two classifiers
- Apply Hu Moments and Histogram feature extraction and transformation on test dataset
- Make predictions on the test features

Below are the steps in the second model-

- Read and resize training images
- Convert the labels to categorical using keras utils
- Extract and pre-process features using ImageDataGenerator
- Split the data into training and validation sets in 80:20 ratio with a random state for reproducibility
- Create a sequential CNN with below architecture-



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_44 (Conv2D)           (None, 150, 150, 16)      208
_________________________________________________________________
max_pooling2d_30 (MaxPooling (None, 75, 75, 16)        0
_________________________________________________________________
conv2d_45 (Conv2D)           (None, 75, 75, 32)        2080
_________________________________________________________________
max_pooling2d_31 (MaxPooling (None, 37, 37, 32)        0
_________________________________________________________________
conv2d_46 (Conv2D)           (None, 37, 37, 64)        8256
_________________________________________________________________
max_pooling2d_32 (MaxPooling (None, 18, 18, 64)        0
_________________________________________________________________
global_average_pooling2d_8 ( (None, 64)                0
_________________________________________________________________
dense_22 (Dense)             (None, 5)                 325
=================================================================
Total params: 10,869
Trainable params: 10,869
Non-trainable params: 0
_________________________________________________________________
```

-
- The first convolutional layer has an input shape of (150,150,3) corresponding with the input shape of the image features
- Max Pooling layer is added after each of the three convolutional layers to learn hierarchy of high level features and to reduce the dimensionality
- Added a Global Average Pooling layer after the last Max Pooling layer. Similar to Max pooling layers, GAP layers are used to reduce the spatial dimensions

- of a three-dimensional tensor. This is because fully connected layer only accepts row vector.
- Number of nodes in the last fully connected layer were setup as the number of unique diagnosis(5) along with softmax activation function to obtain probabilities of the prediction. Relu activation function was used for all other layers.
- Model is compiled using Adam optimizer with a learning rate of 0.0001 which was finalised after iterations with different values of learning rates
- Loss function used is categorical cross entropy along with accuracy as the metric
- Batch size of 50 was passed
- Model trained in 50 epochs

A checkpointer was added to save the best model to prevent loss of model

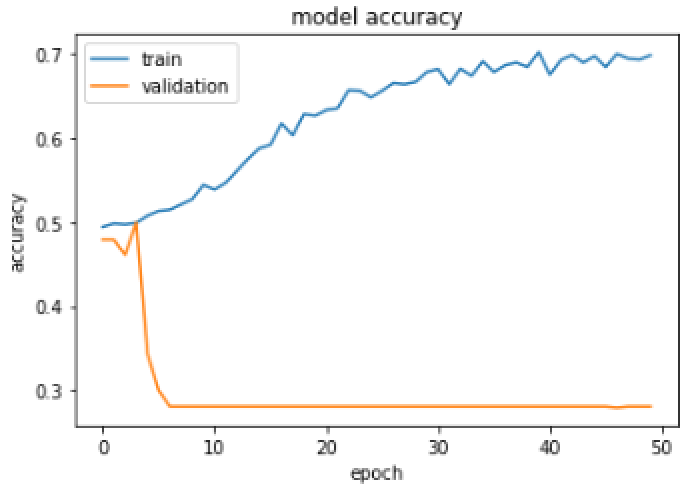
Refinement

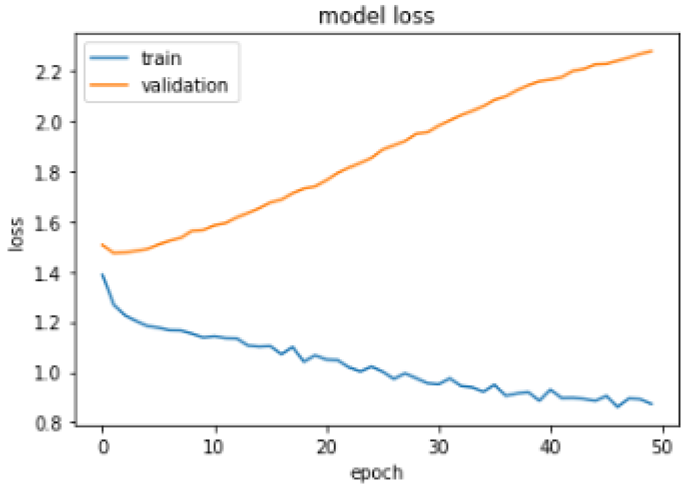
In the first notebook, RandomForestClassifer performed better than logistic regression.

Base_Model (version 2/3) Succeeded 0.040 □
a month ago by Aditya Mehta
From Kernel [Base_Model]

Using this I was able to get a public score of around 0.040. This was my baseline model which I tried to compare with the CNN model built in second workbook.

I used the data pre-processing steps and basic architecture but the validation accuracy does not seem to increase even after multiple trials-





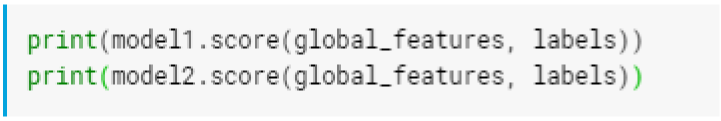
Even though the training accuracy was increasing, the validation accuracy was stagnant.

Multiple steps were tried to refine the model by adding image data generator flow in training, validation and test sets.

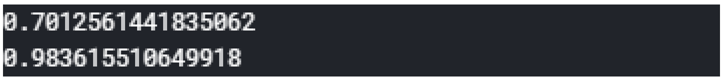
IV. Results

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.

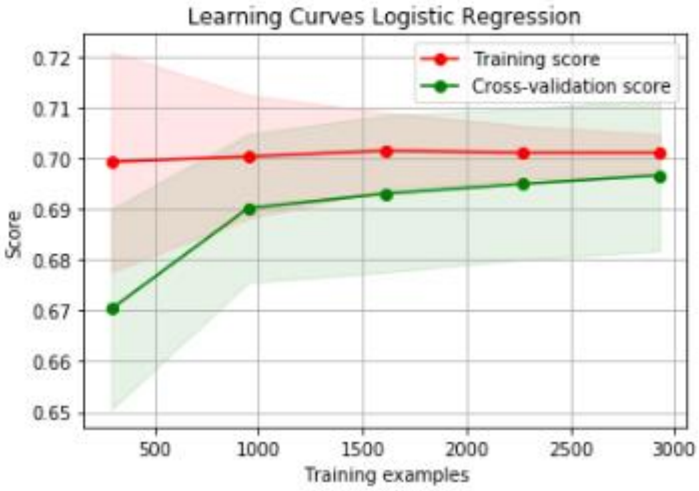


```
print(model1.score(global_features, labels))
print(model2.score(global_features, labels))
```

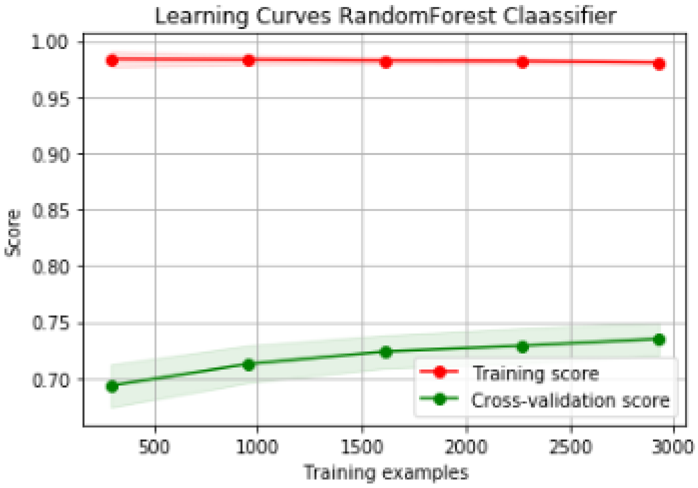


```
0.7012561441835062
0.983615510649918
```

Learning curves for Logistic Regression and Random Forest Classifier-

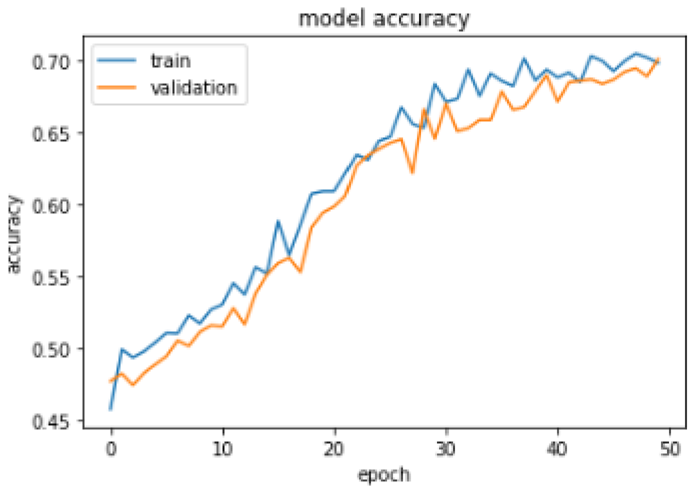


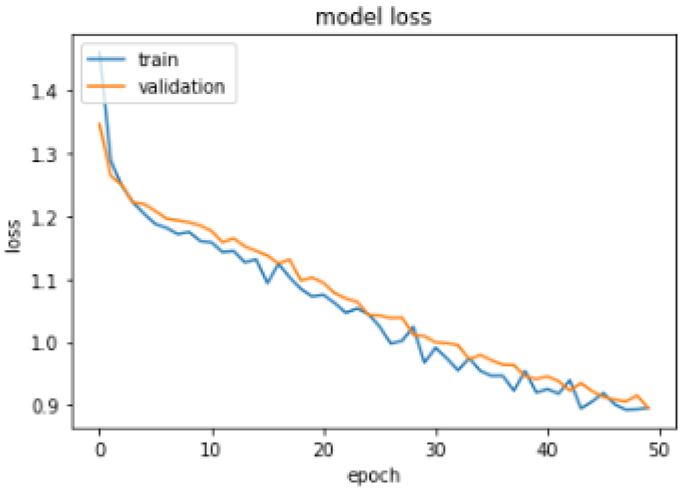
Logistic Regression results tend to converge for training and validation sets as we increase the number of training observations.



Random forest classifier seems to be over-fitting.

After multiple trials of changes in parameters and architecture, validation accuracy started to improve over iterations. Below is the CNN model accuracy on training and validation sets in 50 epochs-





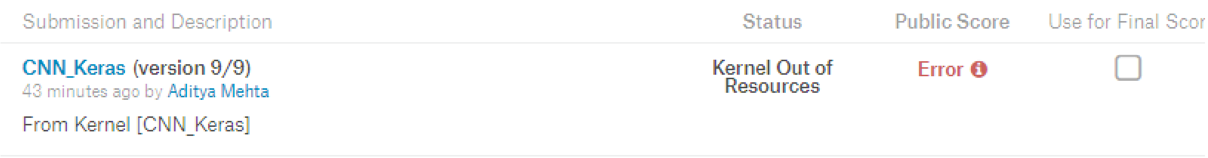
Justification

The CNN model is not better than the baseline randomforest classifier. These models need further enhancements to be able to get used in real world scenarios. Even after multiple enhancements and adjustments, the validation loss is not decreasing below certain threshold. Based on comparison with others' kernels there is a significant scope of improvement of these models.

V. Conclusion

Free-Form Visualization

Since this competition is kernel-only, it's difficult for me to submit the predictions and get scoring on the test-dataset. Here's is one screenshot of the error I get when submitting the current version of CNN_Keras model-



Submission and Description	Status	Public Score	Use for Final Scor
CNN_Keras (version 9/9) 43 minutes ago by Aditya Mehta From Kernel [CNN_Keras]	Kernel Out of Resources	Error ⓘ	□

Reflection

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

1. Kaggle competition of interest was found
2. Exploratory Analysis was done on data
3. A benchmark Logistic Regression Classifier was created
4. RandomForestClassifier was added as the Logistic Regression did not provide reasonable accuracy
    a. This model was submitted as an entry in the compeitions
5. A Convolutional Neural Network was created leveraging ImageDataGenerator features
    a. Due to reasons unknown Public Score for this model was not able to get calculated, so the initial model is better amongst the two at this stage.

Improvement

There is a significant scope of improvement in the two notebooks-

- RandomForestClassifier-
    - Include additional features in addition to Hu Moments and Histogram features
    - Using k-fold cross validation for hyper-parameter tuning and for avoiding overfitting
- CNN- Keras-
    - Tweak ImageDataGenerator arguments for extracting more variety of features
    - Try to get the public score for this model using less kernel resources
    - Try flow_from_dataframe functionality in Keras