Drivebuddy Hiring Assignment

Table of Content

1. Objective

The goal is to train a model that can text classify whether a given image is of pavbhaji or not.

2. Exploratory data analysis (EDA)

While performing the, following observations were noticed.

a. **Data Size**: In the dataset, there exist a total of 453 labelled images of two classes (Pavbhaji & Non Pavbhaji). These samples are too less to train a deep neural network, so we will use image augmentation technique to increase the train data size. (refer figure - 1).

b. **Imbalance Data**: Out of 453, 270 (59.6%) are Non-Pavbhaji images and 183(40.4%) are Pavbhaji images. It implies dataset is imbalanced. This imbalance is not too high. So, can be left explicitly untreated. (refer figure - 1).

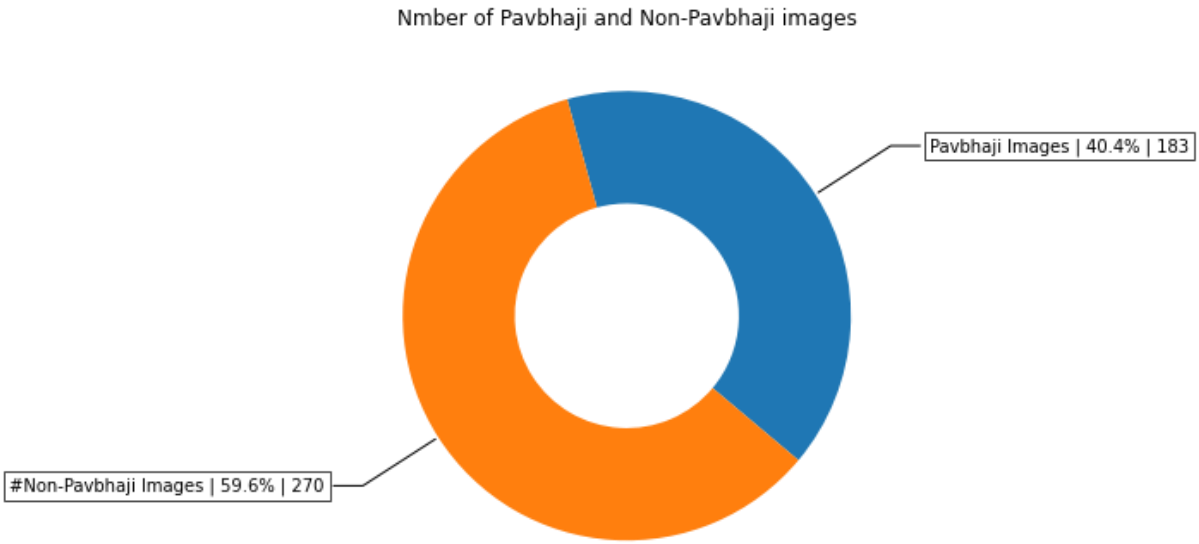

Figure – 1. Class size & ratio

c. **Correct Image Tagging** : Image label seem to be correct for a short sample. Wrong image tagging is not noticed. (refer figure - 2)

d. **Image Size & Aspect Ratio** : The images are of different sizes and aspect ratio, with at least one side of 512 pixels. While doing data preparation, will have to make all images of same height and width (refer figure - 3).

Figure – 2. Image tagging

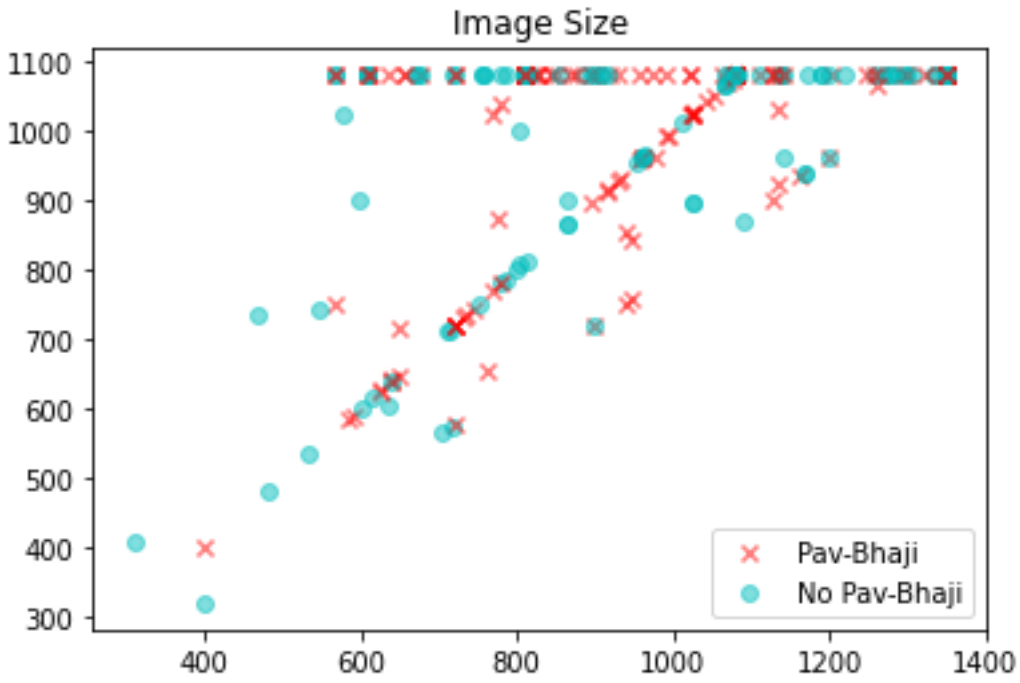

Figure – 3. Image Size

3. Data Preparation.

To prepare the data for modelling, following points are take cared:

a. Image augmentation – To increase the image size and variant image augmentation is used. In this technique, images are rotated, flipped, sheared in different angle.

b. Pixel Normalization – All pixel values are normalized to 0-1.

4. Model Proposal & Architecture.

While preparing the model, a strong constraint to keep in mind is – the data is way too less to train a deep neural model. So a better approach here can be to use Transfer Learning. Here we propose to use the InceptionV3 network weights appended by tree more layers:

InceptionV1 – Conv2D – Dense – Dense – Dense

The final model we implemented after hyperparameter tuning is at figure – 4.

Layer (type)	Output Shape	Param #
inception_v3 (Functional)	(None, 5, 5, 2048)	21802784
conv2d_759 (Conv2D)	(None, 3, 3, 32)	589856
dropout_19 (Dropout)	(None, 3, 3, 32)	0
global_average_pooling2d_7 (	(None, 32)	0
flatten_7 (Flatten)	(None, 32)	0
dense_26 (Dense)	(None, 1024)	33792
dropout_20 (Dropout)	(None, 1024)	0
dense_27 (Dense)	(None, 1024)	1049600
dense_28 (Dense)	(None, 256)	262400
activation_757 (Activation)	(None, 256)	0
dropout_21 (Dropout)	(None, 256)	0
dense_29 (Dense)	(None, 2)	514

Total params: 23,738,946
Trainable params: 7,092,194
Non-trainable params: 16,646,752

Figure-4. Best Model Architecture

5. Hyper-Parameter Tuning

Hyperparameter tuning is a very important and time-consuming step. In this assignment we have used Hyperopt and Hyperas package to do the tuning. Refer: code in figure – 5 to check how it has been implemented. Here we are tuning the below three parameters:

a. Activation Function. ['relu', 'sigmoid']
b. Learning Rate. [10**-3, 10**-2, 10**-1]
c. Optimizer. ['adam', 'rmsprop']
d. Number of hidden layers. Three or Two
e. Kernel Rregularizer. [10**-3, 10**-2, 10**-1]

It took around 6 hours for the model to run. Computation resource being a constraint number of epochs were taken only as 20. The final model architecture we are using is mentioned at Figure-4.

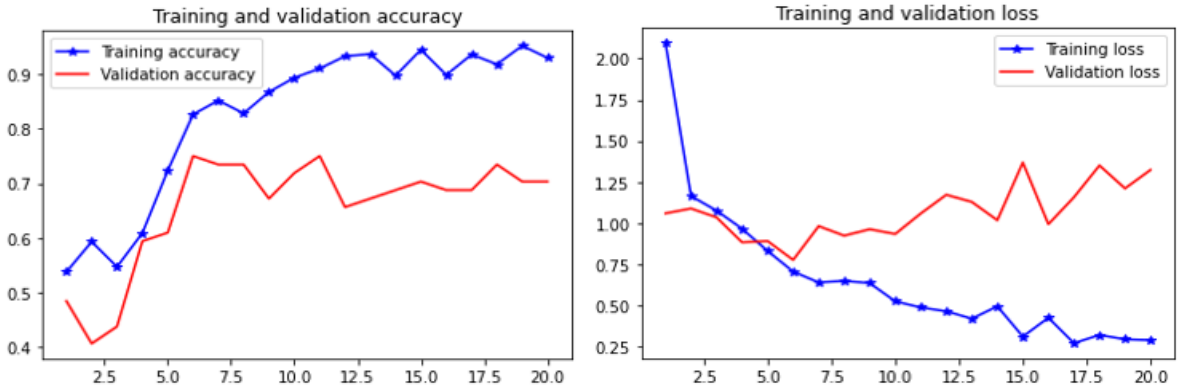

Figure – 6. accuracy & loss – Train & Validation

```
model = Sequential()

# Add the convolutional base model
model.add(inception)
model.add(Conv2D(32, 3, activation={{choice(['relu', 'sigmoid'])}}))# kernel_regularizer=l2(0.01), bias
model.add(Dropout({{uniform(0, 1)}}))
model.add(GlobalAveragePooling2D())
# Add new layers
model.add(Flatten())

model.add(Dense({{choice([512, 1024])}}, activation={{choice(['relu', 'sigmoid'])}}))
model.add(Dropout({{uniform(0, 1)}}))
model.add(Dense({{choice([512, 1024])}}, activation={{choice(['relu', 'sigmoid'])}}))

if {{choice(['two', 'three'])}} == 'three':
    model.add(Dense({{choice([128, 256])}}))
    model.add(Activation({{choice(['relu', 'sigmoid'])}}))
    model.add(Dropout({{uniform(0, 1)}}))

# last layer
model.add(Dense(2, activation='softmax', kernel_regularizer=l2({{choice([10**-3, 10**-2, 10**-1])}})))

adam = keras.optimizers.Adam(lr={{choice([10**-3, 10**-2, 10**-1])}})
rmsprop = keras.optimizers.RMSprop(lr={{choice([10**-3, 10**-2, 10**-1])}})
sgd = keras.optimizers.SGD(lr={{choice([10**-3, 10**-2, 10**-1])}})

choiceval = {{choice(['adam', 'rmsprop'])}}
if choiceval == 'adam':
    optim = adam
else:
    optim = rmsprop

model.compile(loss='binary_crossentropy', optimizer=optim, metrics=['accuracy'])
```

Figure – 5. Hyperparameter Tuning

6. Model Evaluation

a. **Accuracy and loss** - In the best model it was seen that Validation accuracy (refer figure -6) is increasing with the increasing Train accuracy. As the train loss (refer figure -6) is decreasing, unlikely CV loss is increasing. The model would have been even better, if we would have chosen a lower value of patience.

b. **Confusion Matrix** - In the confusion matrix at figure-7, it can be seen that True Negative and True Positive numbers are higher, which is good. False Positives holds 12 samples and False negative hold only 6 samples. Overall accuracy of the model is: 73.5%.

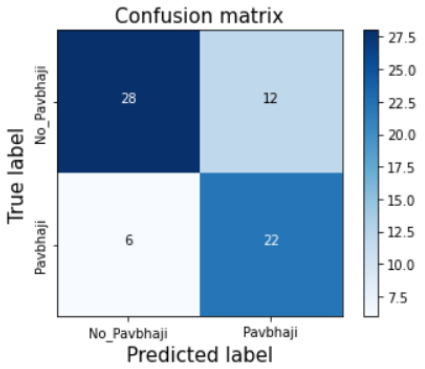

Figure-7 Confusion Matrix

c. **ROC Curve** – By interpreting the ROC curve in Figure-8 we see that the Area Under Curve for both positive and negative samples is 0.81, which is a good indicator of model being performing well.

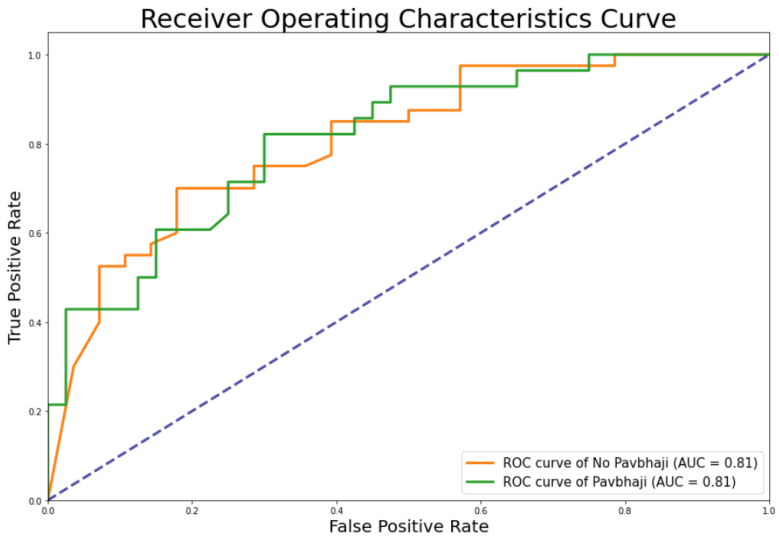

Figure-8 ROC Curve

d. **Evaluating Individual Image**. We tried checking how the model is performing by visualizing random sample from test data (refer figure – 9). Here out of five1 image is predicted wrong.

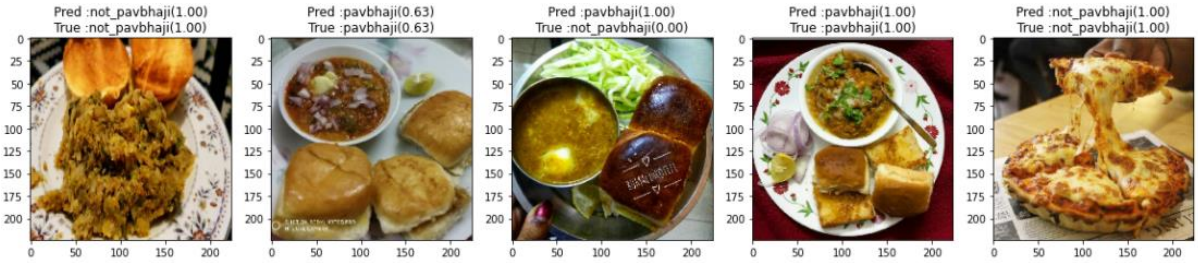

Figure-9 Evaluating Individual Image

e. **Checking images predicted incorrect**.
There can be many reasons for incorrect classification – 1. Incorrect actual label, 2. Image has lot many other objects which model has got confused with, 3. Image is not clear or partially clicked causing model fail. (Refer figure-10 below)
   a. Image at position (2,2) seems to be partially clicked. Image (3,2) contains only one object (only bhaji is there, no Pav), so it's too incomplete.
   b. Image (1,2) and (1,3), seems to contain too many objects along. In our train images, most of the images doesn't contain other object information. So such misclassification might be happening.
   c. Actual labels are correct.

7. Summary

a. Model is performing decently good with accuracy 73.5% and AUC for both positive and negative samples is 0.81.
b. Number of samples is way too less to train a deep neural network.

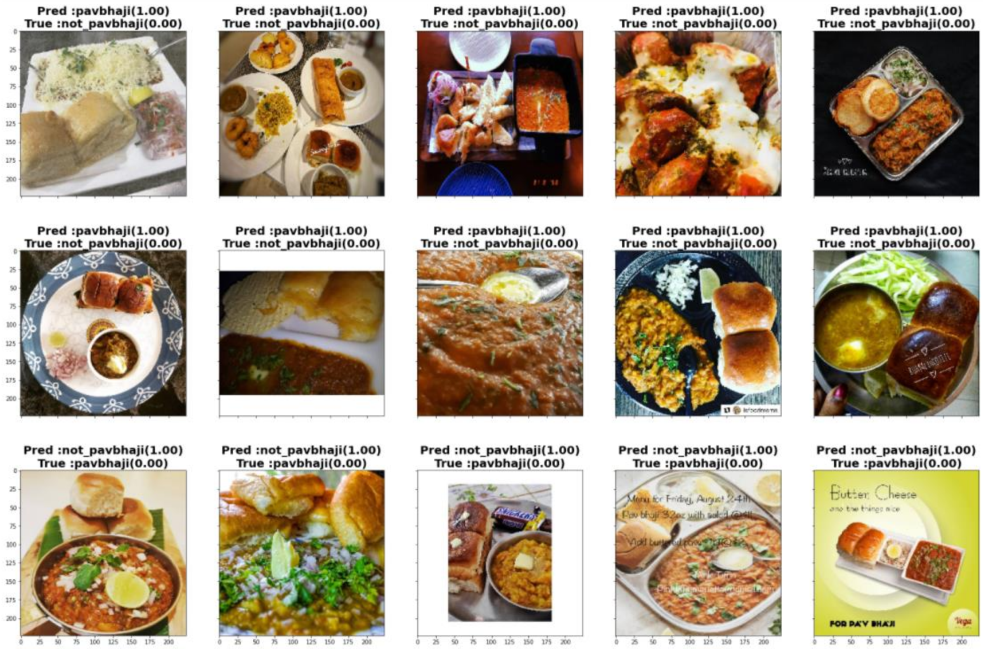

Figure-10. Grid of Incorrect classification