

Machine Learning Engineer Nanodegree

Capstone Project

Marwan Morsy

September 11st, 2018

I. Definition

Project Overview

Computer vision field helps the computers in gaining high-level understanding from digital images or videos. It does tasks that the human visual system can do. object recognition has gained interest by engineers and scientists in artificial intelligence and computer vision.

In this project I chose to solve the Landmark Recognition problem by using Convolutional Neural Networks and transfer learning which outperforms many traditional machine learning approaches computer vision and pattern recognition.

I used this dataset [Modified Dataset](#). This project is inspired by

[Google Landmark Recognition Challenge on Kaggle](#)

Problem Statement

Given a photo to a specific landmark or monument in some place the program will predict the name of this landmark directly from image pixels. The image classification is done by using CNN and transfer learning to achieve this. The strategy for solving the problem is the following :

1. Use the mentioned dataset and do Image preprocessing
2. Use transfer learning from the pretrained models to get the bottleneck features and train on them.
3. Build the benchmark model which will use a basic CNN architecture only
4. Compare the accuracy between solution model and the benchmark model

Metrics

This problem is a multi classification problem, Accuracy will be used as the evaluation metric between the solution model and the benchmark model because the used dataset has on imbalance. Accuracy is defined by

$$\text{Accuracy} = \text{number of correctly identified instances} \div \text{all instances}$$

II. Analysis

Data Exploration

The dataset was originally in this link [landmark3d](#) of 25 classes with over 55,000 images,

It had some problems :

- The dataset was a bunch of lists that have links to download the images from.
- The dataset was quite large and slow to download and upload it on google drive.
- The data was divided into training and validation sets only.

A Smaller version of this dataset of 10 classes which are :

(Casa Loma - Eiffel Tower - Pantheon - Westminster Abbey - Castel Sant'angelo - Independence Hall - Reichstag - Charlottenburg Palace - Milan Cathedral - Stephansdom) with total number of images 19558 images will be used.

There is no imbalance in the dataset.

The images don't have the same dimensions size (height, width) but all of them are quite big images of size approximately 700px* 1000px or 1000px* 700px so image resizing will be used to solve this difference in dimensions size.

This dataset will be included in this link [Modified Dataset](#) which has three directories for training, validation and testing, each one has 10 folders (classes) each class has the images belong to.

Exploratory Visualization

The plots below show how the classes is distributed in the training, validation and testing data. In which the x-axis is the number of images in each class and y-axis is the class name.

These plots show that there is no imbalance in all the dataset because every class in all the dataset have close number of images with the other.



Figure (1)

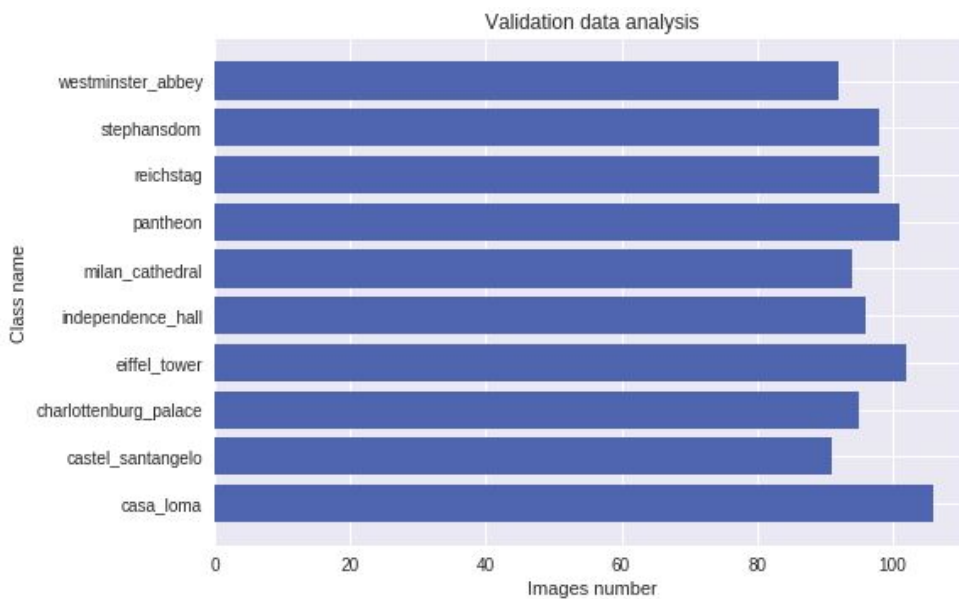


Figure (2)

Algorithms and Techniques

The classification model will be built using CNN (Convolutional Neural Network) which give good results in image class. Transfer learning technique will be used using two pretrained models VGG16 and InceptionV3 built in keras library to save time and get better accuracy.

In each model the bottleneck features will be extracted and use them as an input to train the fully connected deep neural network to detect which landmark that image belongs to.

Benchmark

I will build the benchmark model using a basic CNN architecture, train it on the same used dataset and compare its accuracy in training, validation and testing with the accuracy of the solution model.

The following graph shows the performance of benchmark model in training and validation which gives 0.7588 as training accuracy score and 0.825 as validation accuracy score.

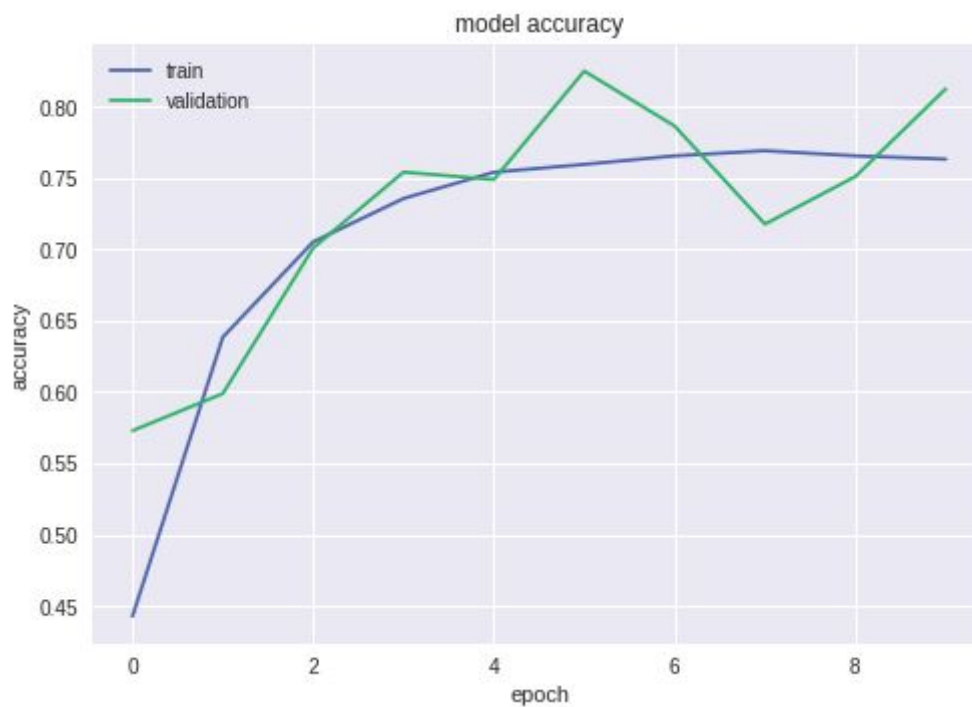


Figure (3)

The testing accuracy score of this benchmark model is *approximately* 0.8033

III. Methodology

Data Preprocessing

1. I divided the dataset into training (16099 images), validation (973 images) and testing (2486 images) sets.
2. The size of the images need to be scaled to a fixed size of 224x224 and the pixels value is divided by 255 to be in range from 0 to 1

Implementation

1. Load VGG16 and InceptionV3 models with average pooling as last layer and use them to extract the bottleneck features from dataset images. Then save these features that will be used for training step.
2. Define a fully connected model with input size equals to the number of features that load from the bottleneck features files saved from the previous step. For the next layer has a size of 256 nodes and using relu activation. I used Dropout with probability 0.3 to reduce overfitting.
The last layer have a size equals to total classes of the dataset which is 10.
3. Compile the models with the chosen optimizer. If the score doesn't work well after training, I will change other optimizer and train again.
4. Fit training and validation bottleneck features to the model in 50 epochs and use checkpointer to save the best weight during training process. Early stopping of 8 epochs patience, then terminate the training process.
5. I made three main functions to make it easier in training and testing of multiple pretrained models with ability to change the optimizer which are :
 1. def save_bottleneck_features(model, name):
 2. def train_top_model(name,optimizer):
 3. def test_top_model(model,name,labels_num):
which name is the model name "VGG16", "Inception"
labels_num is the number of testing images

Refinement

I did refinement in optimizer type and its parameters in the two pretrained models

VGG16 model :

- Before refinement : SGD (Stochastic gradient descent optimizer) is used with learning rate = 0.01 , clipnorm = 1 and momentum = 0.7 :
It got 0.9239 validation accuracy score , 0.8509 training accuracy score and 0.8914 testing accuracy score.
The following graph describes its performance :

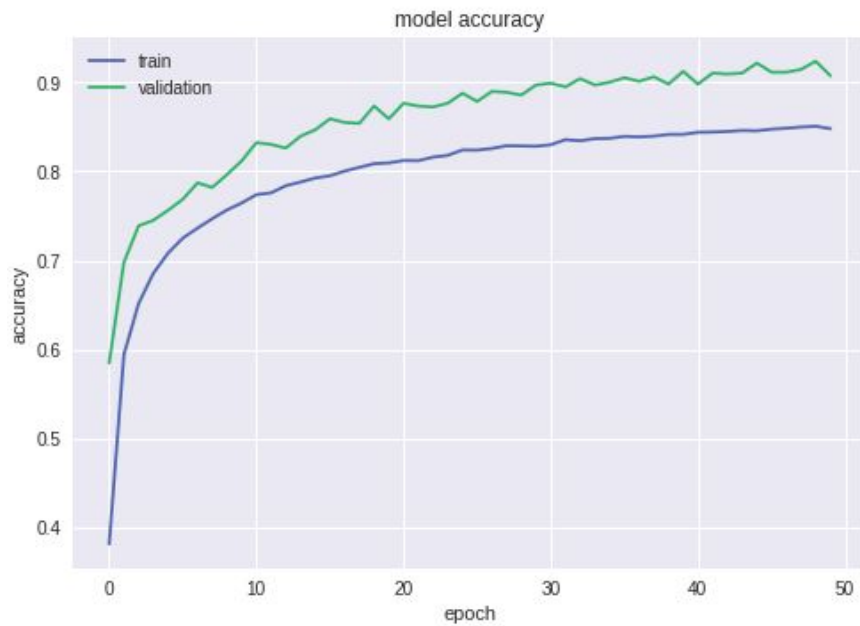


Figure (4)

- After refinement : I used RMSprop optimizer with default parameters values
It got 0.9579 validation accuracy score , 0.8874 training accuracy score and 0.9304 testing accuracy score.

The following graph describes its performance :

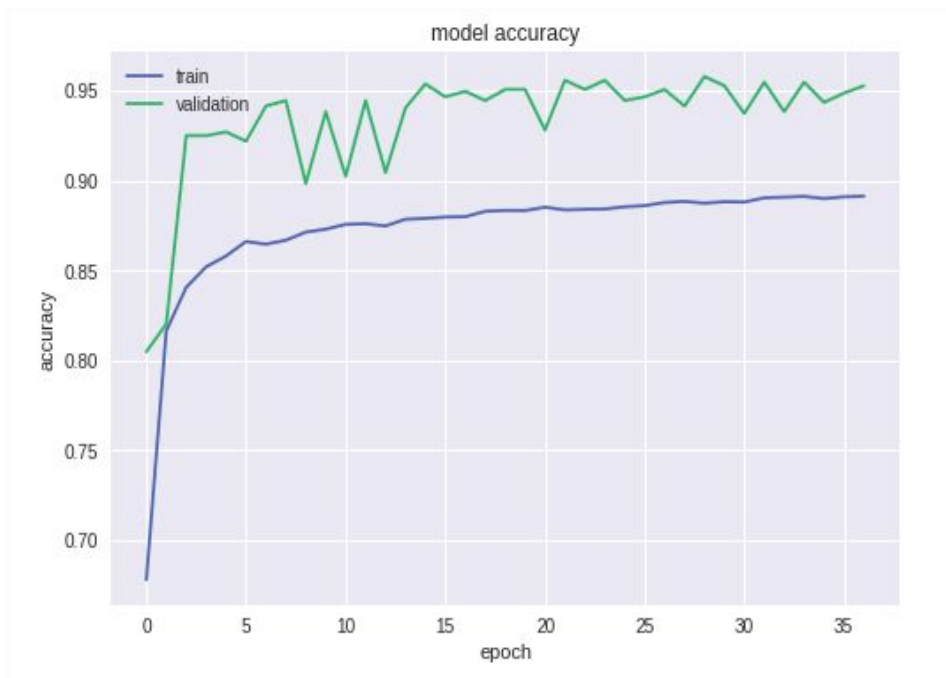


Figure (5)

InceptionV3 model :

- Before refinement : SGD (Stochastic gradient descent optimizer) is used with learning rate = 0.01 , clipnorm = 1 and momentum = 0.7 :

It got 0.9435 validation accuracy score , 0.8919 training accuracy score and 0.9393 testing accuracy score.

The following graph describes its performance :

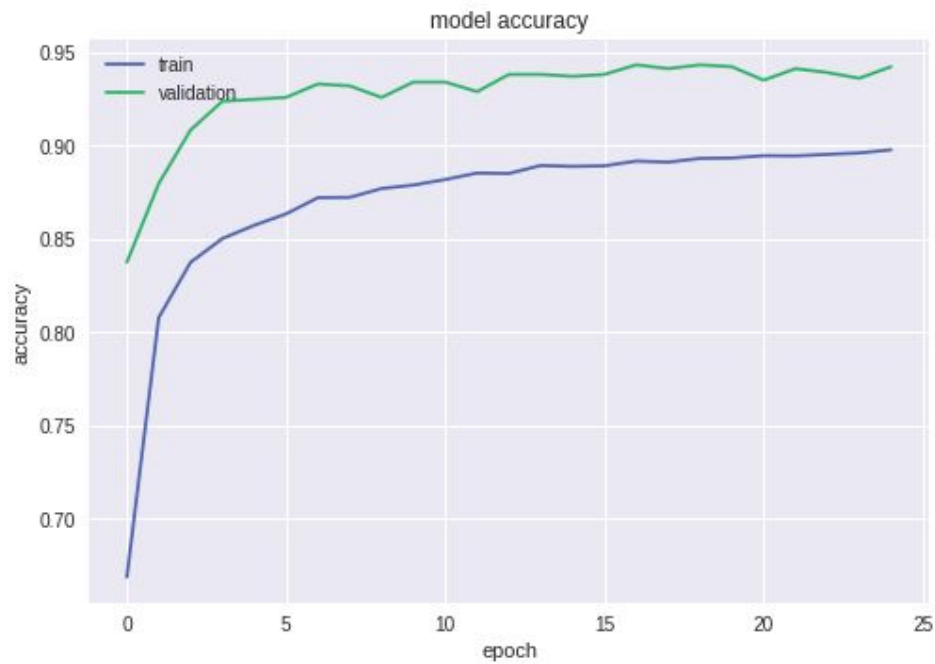


Figure (6)

- After refinement (solution model): I used RMSprop optimizer with default parameters values

It got 0.9538 validation accuracy score , 0.8897 training accuracy score and 0.9473 testing accuracy score.

The following graph describes its performance :

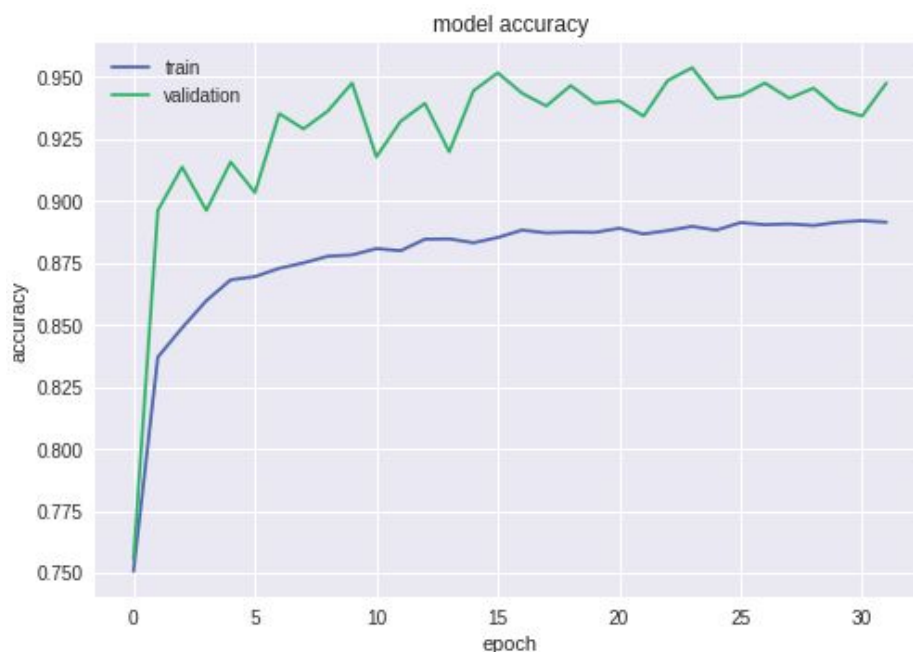


Figure (7)

IV. Results

Model Evaluation and Validation

The final model will be the refined InceptionV3 model which performed well on the testing set.

Sensitivity analysis

To validate the robustness of this final model I changed the size of all images in the dataset from 224 x 224 pixels to 150 x 150 pixels.

I computed the bottleneck features for the new modification in dataset and started training and testing on them.

It got 0.89106 validation accuracy score , 0.8738 training accuracy score and 0.8793 testing accuracy score which is not very far from 0.9473 (Solution model testing score).

The following graph describes its performance :

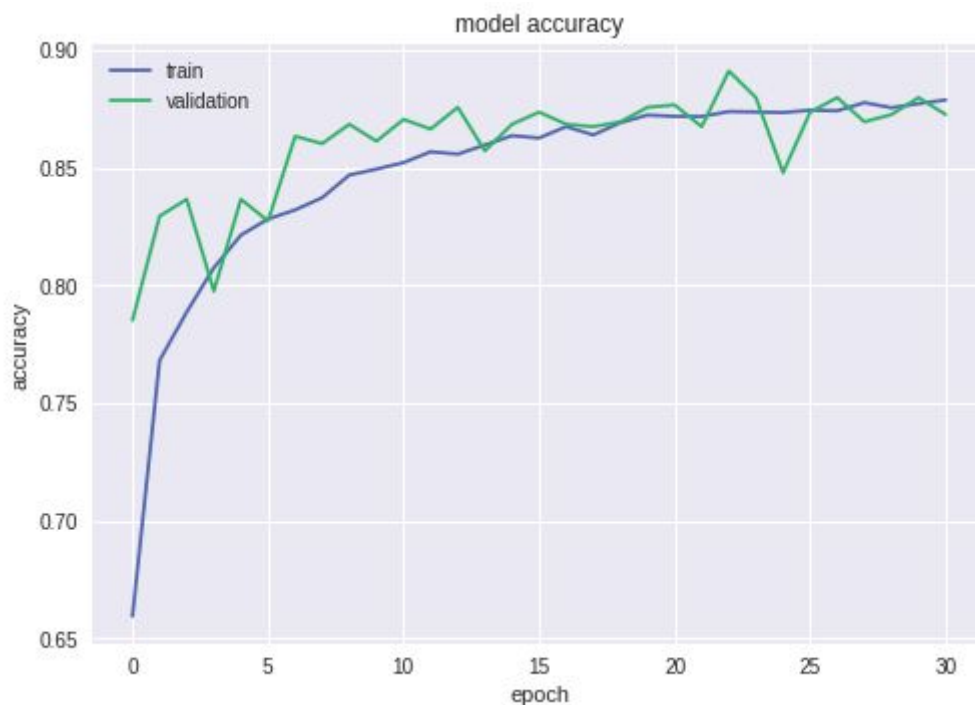


Figure (8)

Justification

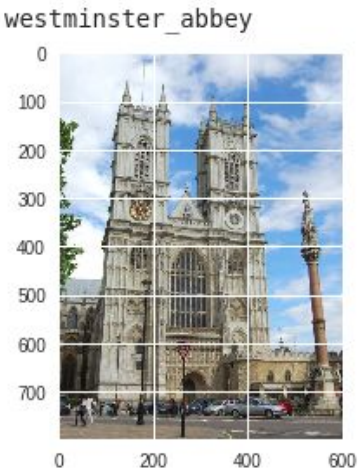

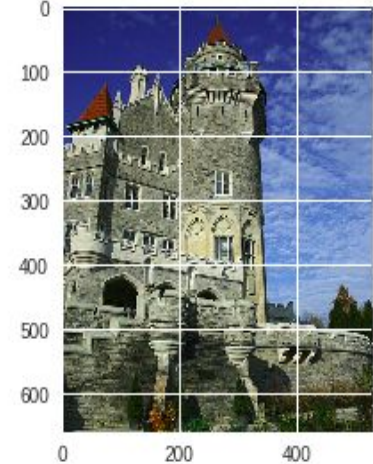

The solution model outperforms the benchmark model in accuracy of unseen data with **95%** because the benchmark model accuracy is **80%**.

This is clear in the Figure (3) and Figure (7) the difference of training and validation accuracy score.

V. Conclusion

Free-Form Visualization

I used some images from the web and tried to get the solution classifier predict them.

Test image and its prediction	Training images that looks like it
<p>westminster_abbey</p> 	<p>Westminster_Abbey</p> 
<p>casa_loma</p> 	<p>Casa_Loma</p> 

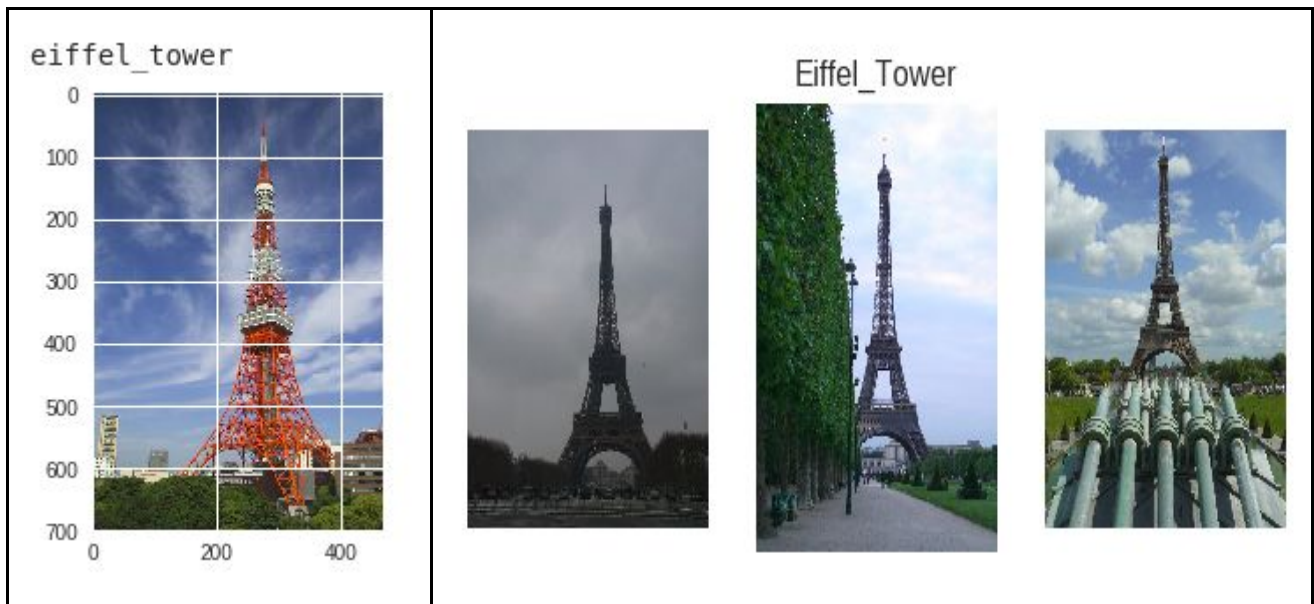


Figure (9)

Figure (9) shows that the solution classifier has done a good job in classifying the unseen tested images.

In the last row of the figure the tested image is a tower in Tokyo that is similar to the eiffel tower in France. The classifier is able to detect the similarity and predict it as eiffel tower.

Reflection

The entire end-to-end problem solution:

1. Choose the problem and a valid dataset
2. Reduce the chosen dataset and divide it among training, validation and testing.
3. Do image preprocessing on the chosen dataset.
4. Build the benchmark model.
5. Extract the bottleneck features from the pretrained models VGG16 and InceptionV3.
6. Train both models, refine them and pick the model with the best performance.

The interesting aspects that the performance of the model that used InceptionV3 bottleneck features is slightly better than the model that used VGG16 bottleneck features. I thought there will be much larger difference between them in accuracy of testing data but there was not.

Difficult aspects in the project was preparing the dataset to be ready, build a strong benchmark model to compare it with the solution model and refining the final model to get best results.

Improvement

The accuracy of my final model is **95%** which is quite good. My suggestions to improve my model is to try

1. Fine tuning pretrained models might give better results than transfer learning but it will take more time as it trains part of the pretrained model whose training takes long time.
2. Transfer learning with more pretrained models and compare results.
3. Increasing the number of images of each class by data augmentation or choosing bigger dataset.

References and useful links

1. [Building powerful image classification models](#)
2. [Save and Load Your Keras Deep Learning Models](#)
3. [Landmark Recognition Using Machine Learning](#)
4. [Tutorial on using Keras flow_from_directory and generators](#)