**An Application to recognize National landmarks using Deep Learning**

Sourav Kanta, Skybits Technologies Prvt. Ltd

**Table of Contents**

* Acknowledgement
* Background and description of problem
* Main Objective
* Approach for solution of problem
* Theoretical background for solution
* Technical details
* Screenshots
* References

**Acknowledgement**

This project would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to Mr. Tirthankar Banerjee for his guidance and constant supervision as well as for providing necessary information regarding the project and also for his support in completing the project.

I would like to express my gratitude towards my college IIEST Shibpur for their kind co-operation and encouragement which helped me in completion of this project.

**Background and Description of Problem**

In today’s date tourism is a big source of income for any given local population as well as the state itself. Therefore attracting more tourists would help the local as well as the state economy flourish. One approach to achieve this would be to keep tourist constantly well informed of the various interesting details about the places they visit. Not only does this increase a tourist’s travel experience it helps them acknowledge and appreciate the history and background of the place they are visiting. In this project we try to propose a technical solution to this very problem.

**Approach to solve problem**

One thing that we can all agree upon is that the first thing each of us does while visiting a place is to take a few pictures of it, be it to capture the insane beauty of the place or as a nice addition to our own timelines. But what if one click of that camera can do more than that? Our solution proposes to fetch the photo captured by the user and run it against a backend that applies various machine learning techniques upon this image and finally predicts the place the user is visiting and comes up with a nice little summary of the description as well as the history of the place. Two birds with one stone, right?

**Theoretical background to application**

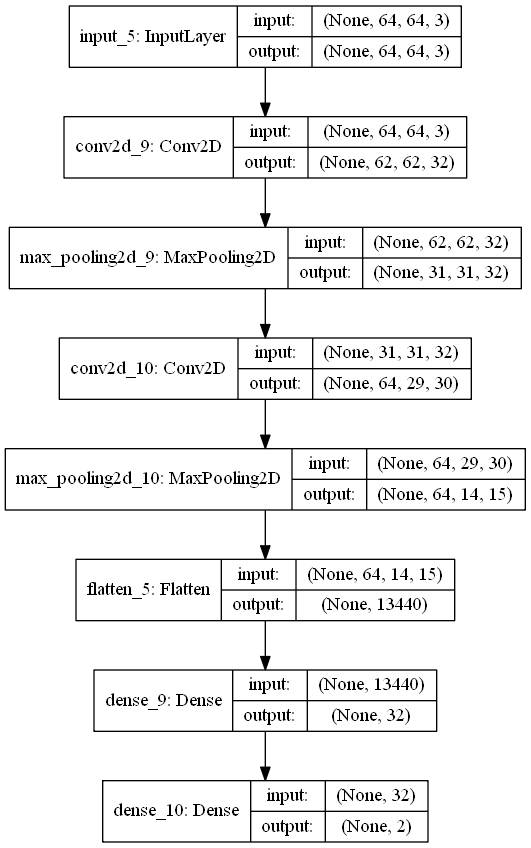
For recognizing a landmark image, i.e. , image classification we are applying deep learning on the image captured by user. **Deep learning**, as the name suggests, is a sub sect of machine learning. Deep Learning mostly involves using deep artificial neural networks (algorithms/computational models loosely inspired by the human brain) to tackle machine learning problems. State-of-the-art image classification solutions today use deep learning.  So, what is a neural network? Here’s an analogy: imagine a neural network as a series of doors one after another and think of yourself as the ‘input’ to the neural network. Every time you open a door, you become a different person (i.e. you change in some way). By the time you open the last door, you have become a very different person. When you exit through the last door, you become the ‘output’ of the neural network. Each door, in this case, represents a **layer**. A neural network, therefore, is a collection of layers that transform the input in some way to produce an output. Each layer in the neural network consists of ‘weights’ and ‘biases’ — these are just numbers that augment the input. The overall idea of a neural network is that it takes in some input (usually a collection of numbers that represent something, e.g. Red-Green-Blue values of pixels in an image), applies some mathematical transformations to the input using the weights and biases in its layers and eventually spits out an output. You can look at the input, output and weights as matrices. The input matrix gets transformed by a series of matrices (i.e. the weight and bias matrices of the layers) and that becomes your output.

A deep neural network is just a neural network with many layers (as you stack layers on top of another, the neural network keeps getting ‘deeper’). How many is many? Well, there’s a [VGG16](https://arxiv.org/pdf/1409.1556v6.pdf) neural network architecture (used for image classification) that consists of 16 layers and then there’s the [ResNet](https://arxiv.org/pdf/1512.03385.pdf" \t "_blank) architecture (also used for image classification) that consists of 152 layers — so, the range is pretty wide. The basic idea of deep learning is using neural networks with multiple layers.

Neural networks consist of layers that consist of weights and biases (which are just collections of numbers). During the training phase, the neural network tries to find the right weights/biases that lead to the most accurate output. It does so using a method called backpropagation. Before a neural network is trained, the weights/biases are initialized, either randomly or from a previously trained model. Either ways, when training happens, the neural network changes those weights and biases based on what it ‘learns’. When we build a neural network, we have to decide on (i.e. choose or design) something called a **cost function**. The cost function is basically just a mathematical function that takes in the output from a neural network (for a given input) and the ground truth data (i.e. the expected output from the neural network for that given input) and calculates how off/bad the result from the neural network was. Using optimization techniques like gradient descent, the computer calculates how to change the weights and biases such that the cost function is minimized. It keeps doing this as it trains on more and more data (get the output from the neural network, calculate cost and back propagate to change weights). Over time, the weights and biases adjust with the data and (hopefully) you end up with a neural network that has high output accuracy. Remember, the practical effectiveness or accuracy of a neural network is largely dependent on the data used to train it; so it’s very important that the proper dataset is built or chosen. Without good data (and a good amount of data) it can be very hard to train an accurate neural network.

**Technical Details**

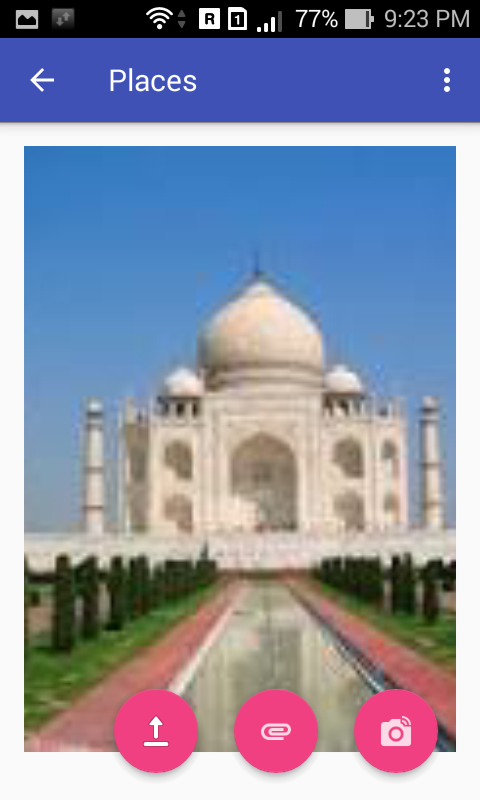
In our project we apply the following model for our neural network.

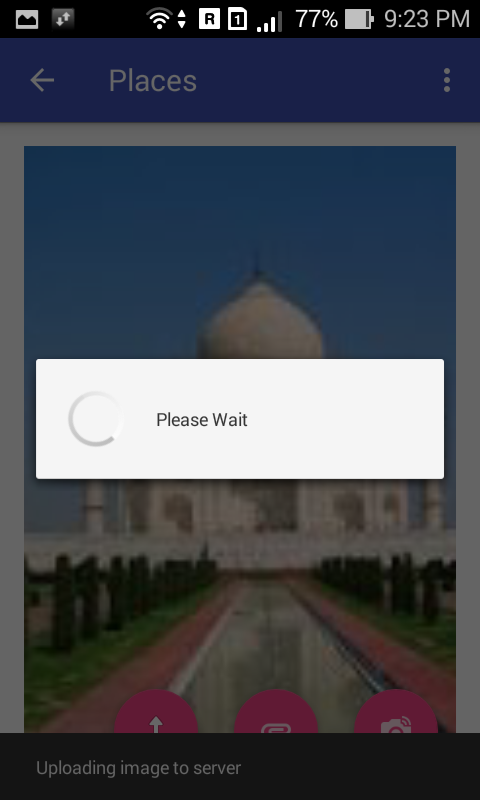
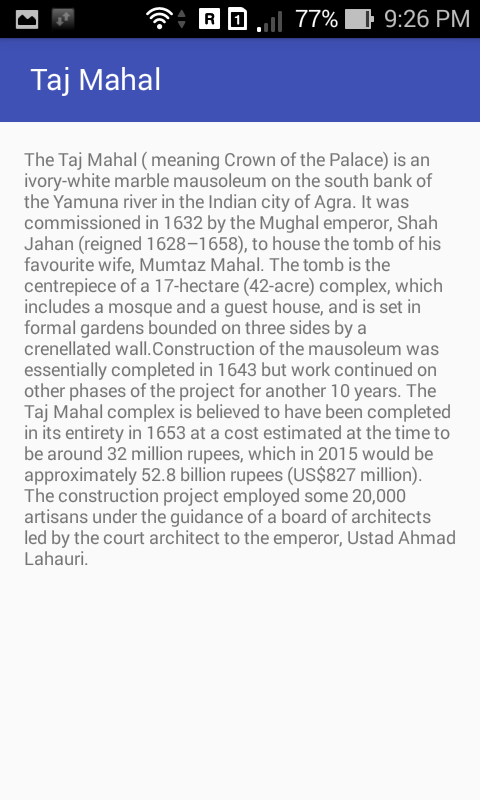


We are using a Django backend that accepts a base64 encoded image and the Keras module in Python to implement our deep learning. Initially the neural network has been trained with about 200 images of each particular landmark and gives an accuracy of 80.3% on a stratified portion of the training dataset. We fetch the photo the user has taken through the camera and convert it into a base64 encoded string and transmit it to the server. The server decodes the image and resizes it to a 64 X 64px image finally feeding it into the above neural network as a numpy array. We set our prediction by selecting the landmark which has the highest confidence value after the predict function call. We return a JSON string from the backend to the mobile application which contains the following fields:

* Status – Indicates success or failure of query
* Description – This field contains a brief summary of the landmark that will be displayed and read out to the user.
* Title – Name of the landmark
* Img – Url for a picture of the image

**Screenshots**

**References**

<https://github.com/sourav-kanta/server>

<https://github.com/sourav-kanta/LandmarkRecogniser>

<https://github.com/sourav-kanta/PlacerecogniserApp/>