Testing Training & Deploying Object Detection Model

Piyush Malhotra
piyushmalhotra28@gmail.com

This proposal puts forward the idea of ***training, testing and deploying an Object Detection Deep Learning Model*** *as a REST API.* The idea is to make an environment which is user-friendly and easily accessible. The proposal is divided into 7 sections discussing various thoughts and methods kept in mind while scripting this project proposal. (1) Domain Background; (2) Problem Statement; (3) Datasets and Inputs; (4) Solution Statement; (5) Benchmark Model; (6) Evaluation Metrics; (7) Project Design.

1. Domain Background

Deep Learning is prevailing in today's world and industry is making new breakthroughs every day. This boom results in new and old approaches to be looked through a new lens. It has opened major doors for us in the name of Intelligent Applications. Looking towards computer vision using Deep Learning, we realize we have made significant improvement and these improvements are the need of the hour for daily users. The new architectures designed are state-of-the art yet many of them don't come in the hands of daily user. This proposal tries to bring one such thing in the hands of every developer who knows or doesn't know how to train and use a Deep Learning model.

It's always a painstaking to task to have authentication as a requirement when using APIs provided by big companies. There has been major rise in applications requiring to use authentication to use the interface and interact with APIs. This leads to more code to write, test and debug.

We have many new architectures which are compute extensive and cannot be ran on small devices like smartphone. We also have issues of multiple platforms (Android, iOS, Windows) resulting in requirement of an application which is platform-independent. To get around this a REST API having access to a server with more than enough compute power is the best way to go (given that, credibility of cross-platform applications performance can be questioned).

2. Problem Statement

Going around the bush in the domain background we can now hit the bush with the exact problem statement. "*Training, Testing and Deploying a Deep Learning Model for Object Detection as a REST API*".

Although the problem statement is clear enough there are still many queries to unmask. What will be the model architecture? What will be the dataset? Will there be any preprocessing? And many more.

All these queries will be answered in the subsequent sections while keeping in mind that there are many hiccups in documenting and real implementation. Thus, multiple parameters can be upgraded in the actual project to make sure we arrive at an efficient and a desired result.

3. Datasets and Inputs

For the purpose of this project I'll be using the Common Objects in Context (COCO) dataset. The dataset contains over 200,000 images and 80 object categories. Techniques like rotation will also be applied to produce more "fake data" if needed.

The input image can be of at max 700 pixel height and 700 pixel width; any image larger than this will be rejected. This is to keep in mind that the image higher than this can cause latency to occur.

A fully functional input pipeline will also be produced to normalize the image and pass it on to the network in an efficient manner.

4. Solution Statement

The solution of this interesting yet subtly complex problem is to use the most battle tested architecture present in the wild – You Only Look Once (YOLOv2), alongside a flask web service and a redis database to maintain the incoming images. A flow graph of this can be found under the Project Design section in figure 1.

5. Benchmark Model

Setting up a benchmark to achieve is the utmost task in any project. In section 4 it is stated that YOLOv2 will be used for Object Detection. Thus the benchmark model for this project will be the model used by the authors of YOLOv2 for COCO dataset detection task.

Mean Average Precision (mAP) of YOLOv2 as given by the author on train-dev is 48.1%.

6. Evaluation Metrics

The evaluation metrics used for this project is Mean Average Precision (mAP). How to calculate mAP?

a. Calculate Intersection over Union (IoU) between the proposed object bounding box A and the set of true object Bounding Box B.

$$IoU(A, B) = \frac{A \cap B}{A \cup B}$$

b. Select A if IoU is greater than threshold.
c. For each class c, one can calculate the
   - True Positive (c): a proposal was made for class c, and there actually was an object of class c.
   - False Positive (c): a proposal was made for class c, but there was no object of class c.
   - Average Precision (c) = $\frac{\#TP(c)}{\#TP(c)+\#FP(c)}$

$$mAP = \frac{1}{|classes|} \sum_{c \in classes} \frac{\#TP(c)}{\#TP(c) + \#FP(c)}$$

7. Project Design

a. Training Phase

This phase consist of two main parts: (i) Defining the input pipeline for training and (ii) Actual Training. It might feel like these two parts are sequential, but these can go in loop. Changing the training phase can change how we setup the input pipeline. Few things that will be permanent part of the input pipeline is augmenting data using techniques like rotation, and other pixel tweaks.

The training phase will take into account mainly 3 metrics (over training and validation data) – loss (mean square error for most part), accuracy and mAP (see section 6). The precise details of training will be documented in the project documentation.

b. Testing Phase

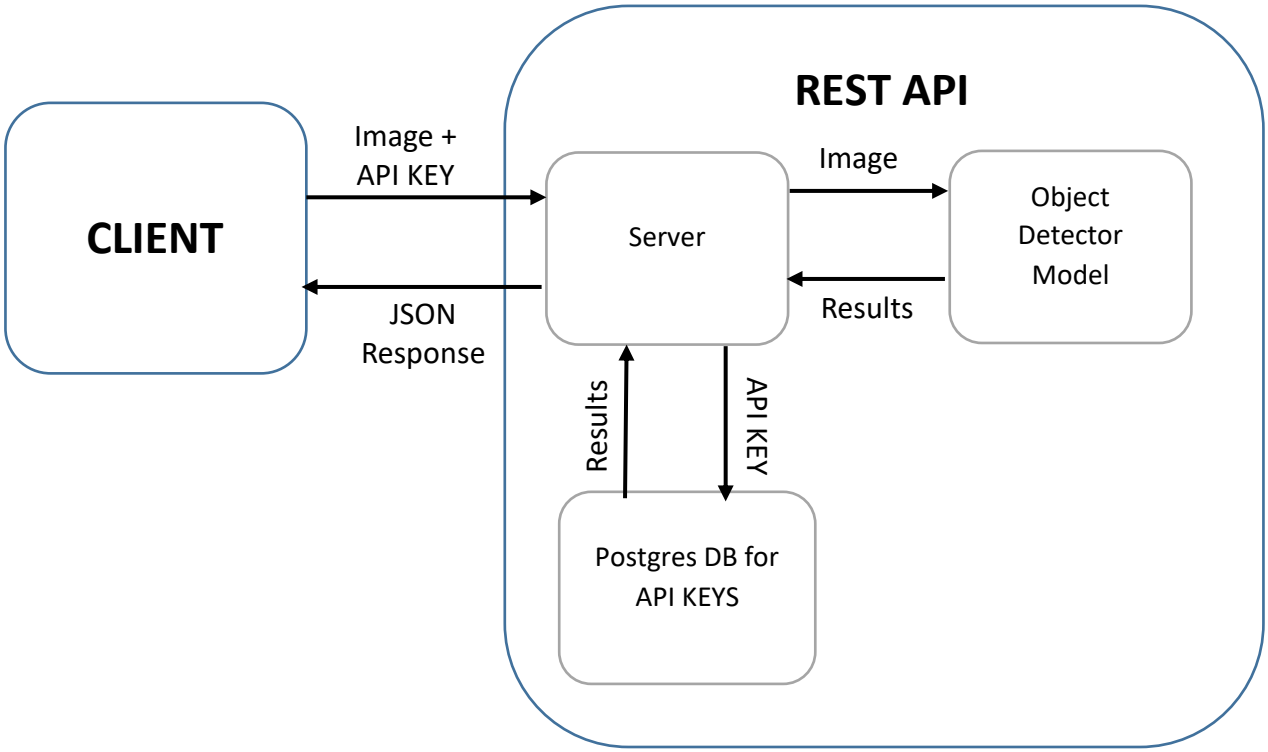
Testing phase will majorly cover parts of dataset which was kept hidden from the model in the training phase. This phase will determine the actual credibility of the model when it is presented out in the wild. This phase will also be divided in two categories (i) Metrics based evaluation on all the dataset kept away from the model during training phase and (ii) Human based evaluation on hand picked data of almost 300 images.

Why step (ii)? We know that the images that will be used in step (i) will be drawn from COCO dataset as are the images in training phase, this means that there is a very high chance that the model can overfit over the COCO dataset itself. Thus, if we just rely on step (i) evaluation, we might just be relying on the model to work on only one kind of dataset rather than on a model that can perform well in general scenario.

c. Deployment Phase

This phase takes the TensorFlow model trained and tested in above two phase and make a servable using TensorFlow serving to deploy it on a server and produce an end point as gRPC, which in turn will act as a REST API.

END RESULT


REST API
CLIENT
Image +
API KEY
JSON
Response
Server
Image
Results
Object
Detector
Model
Results
API KEY
Postgres DB for
API KEYS