Capstone Proposal

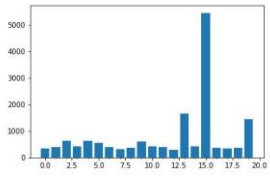
Sagarnil Das

10th August, 2017

Domain Background: Computer vision, being a scientific discipline is concerned with the theory and technology for building artificial systems that can decode and obtain information from images or multidimensional data. From the application standpoint, it seeks to automate tasks that human visual system can perform. Computer vision domain is mainly implemented via deep learning and neural networks. The sub domains in which computer vision is being implemented are video tracking, object recognition, motion estimation, restoration etc (http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks). My main motivation of implementing computer vision as the founding block of this capstone project is building a program which can be very useful to blind people.

Problem Statement: Vision loss is a problem that affects people all around the world. As computers are getting better at understanding images due to advances in computer vision, the concept of a virtual assistant for the blind that could read text, identify/spot objects, or even describe a whole scene in natural language is becoming increasingly realistic. The goal is to create an application, which can detect various objects in real-time and then use an artificially generated voice to utter the name of that object with a certain percentage of threshold accuracy.

Datasets and Inputs: For this project, I will use the PASCAL VOC data to train the classifier, which consists of 20 classes of images. The train/validation data has 11,530 images containing 27,450 ROI annotated objects (http://host.robots.ox.ac.uk/pascal/VOC/). The training data provided consists of a set of images; each image has an annotation file giving a bounding box and object class label for each object in one of the twenty classes present in the image. The dimensions of the images are 500 x 375 or 500 x 333 and these are all color images with 3 channels/layers each. Although the number of images is same as the number of annotation files, within each annotation files, multiple objects can be present. So the label classes are not distributed evenly. Note that multiple objects from multiple classes may be present in the same image. This might cause overfitting as the distribution is right tailed. So based on the results, I may have to chop down some of the classes.



I will use transfer learning on the Mobilenets convolutional neural network (https://arxiv.org/pdf/1704.04861.pdf) and then use the above dataset to train the classifier even more. Finally I have an additional plan of taking my own sets of images, manually labeling and preprocessing them and then train the network with these sets of images. I will implement this plan depending on how well the model gets trained.

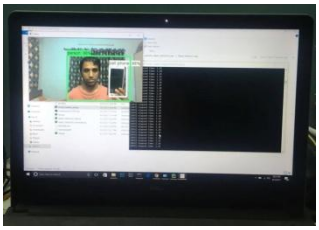
Solution Statement: The steps for implementing the application are as follows. First, I will download and preprocess the PASCAL VOC training data. In the preprocessing stage, in the first pass, I will pass the images as is without any resizing or normalization. Then based on the results, I will most probably scale the image sizes down so that it can run on mobile devices with a faster speed. Then I will configure the object detection pipeline. After that I will train my classifier using the Mobilenets neural network using the Tensorflow object detection API (https://github.com/tensorflow/models/tree/master/object_detection). I will also implement the artificially generated voice in my code using the pyttsx module in python so that it gets correctly mapped to the labels predicted by the classifier. Then once the model is successfully trained and can correctly identify a variety of objects with a good accuracy, I will implement that classifier in an android application where the user will have to track things with the phone's camera and the classifier will attempt to predict the object in the frame correctly.

Benchmark model: To create an initial benchmark for the classifier, I used the Resnet50 architecture using Keras with Tensorflow as the backend to classify various classes of images from the PASCAL VOC module. It achieved an accuracy of 80%.

Evaluation Metrics: Accuracy is a very common metric for both binary and multiclass classifiers. Here it is a multiclass classification problem as we are dealing with many classes of images. The accuracy takes into account both true positives and true negatives with equal weights. Another evaluation metrics is the validation and training loss. The lower the loss is the better. Loss is not in percentage as opposed to accuracy and it is a summation of the errors made for each example in training or validation sets. In the case of neural networks the loss is usually negative log-likelihood in case of classification problem. For accuracy, the formula is:

$$accuracy = \frac{true\ positives + true\ negatives}{dataset\ size}$$

Project Design: The first step towards this project was to test the Mobilenet architecture which was trained with COCO data of 20 classes of images with the tensorflow object detector API. Here I modified the code and implemented OpenCV so that the classifier can also detect the objects from a webcam. I implemented the multiprocessing library of tensorflow to get a faster result with increased frames per second.



So as we can see, the basic philosophy behind the algorithm of this object detector is drawing a bounding box around the subjects and then making a prediction. For training my own dataset, I have downloaded the Pascal VOC dataset. But tensorflow API uses the TFRecord file format. So I had to convert my data to TFRecord. To prepare the input file for the API I needed to consider two things. Firstly, I need an RGB image which is encoded as jpeg or png and secondly I need a list of bounding boxes (xmin, ymin, xmax, ymax) for the image and the class of the object in the bounding box. The files also come with annotations which are in form of a XML file. Now my next target was to create these annotations to a csv file and then split the csv file into two separate TfRecord files which will be fed to the neural network. I wrote a script to convert the files from xml to csv and then the train test split was done and converted into two separate tfrecord files. I also took the visualization_utils.py file which has the labels detected and integrated pyttsx module in this file. I wrote 90 customized voice generated messages for the 90 classes it is being trained on.

For the training, we needed:

1) An object detection pipeline: Tensorflow provides sample configuration files. I took the ssd_mobilenet_v1_pet.config and configured it to ssd_mobilenet_v1_pascal.config for Pascal data. I needed to configure the num classes and describe all the classes I have in my training and testing data. Also I had to change the parameter path_to_be_configured to the appropriate locations in the file system. (After the initial test run, which is still happening in my computer for last 2 days as I am using a CPU, I want to implement additional data augmentation to get a better accuracy as we have only 20 classes of images. Also for my hand labeled images plan, I would most likely need data augmentation as the number of samples in that case will not be large). I also used a pre trained model checkpoint as it would take an even longer time if I wanted to train it from scratch. I used the ssd_mobilenet_v1_coco checkpoint.
2) The dataset and the label maps: We will need the tfrecord files which we just created to be fed into the network and also the label maps which comes in a protobuf text format (pbtxt). Here all the classes are labeled in the following format:

   ```
   item {
     id: 1
     name: 'aeroplane'
   }
   ```

   Based on the initial results, I will re-train the images with varying sizes and also the file size to see which combination gives me the best balance between accuracy and speed. Then I will use that model. For the image size, I am planning to use these dimensions: '224', '192', '128', '64', '32'. Now I finally start the training with 20000 steps. I have also started the evaluation process parallel to the training process so that I can observe both the changes in log loss and accuracy. So far it is painstakingly slow, but here I cannot do anything but to be patient. Maybe I will observe if my model is following the correct trend for 5-6 days and then stop the process and do the whole thing in cloud with a GPU. Here is some initial visualization from tensorboard. After all the training are done, I will use image augmentation if the result set is sparse and also might reduce the number of hidden layers or introduce an image subset to the neural network to fine tune and optimize it.