ROBOTIC INFERENCE

Abstract:

In this project, there are two main sections that are covered. First section is about analyzing the default data, train the data and evaluate the results and adjust the parameters until the results are as expected. The second part is about collecting a data real time and train the data to get good results.

Introduction:

This Project is about discussing the idea of the Robotic inference and the tradeoffs that may be applied when we move the project to the production (that is for practical application of Robotic Inference). In this Project, we will also discuss the data collection process and the techniques used while collecting. The Idea of Robotics Inference discussed here is training the network to classify three types of objects. 1. Scientific Calculator, 2. Game controller 3. TV Remote.

Background:

The first part of the project is to classify two objects and one background used in capturing the objects. To Classify this, GoogLeNet is used as provided by NVIDIA DIGITS to classify the objects from the data I collected and for the default data.

GoogLeNet Architecture:

The architecture of GoogLeNet is based on the inception module of the deep neural network. The GoogLeNet contains about 9 inception modules and each module consists of Convolution, pooling, Softmax as major things. Basically, it is a network in a network in a network and the winner of ImageNet competition, 2014.

There are 3 default networks available in the NVIDIA DIGITS platform. LeNet, AlexNet and GoogLeNet. Out of these GoogleNet has more layers and is successful in addressing the Efficiency and Practicality that most deep networks face as the challenge. Also, GoogLeNet is 12 times more accurate than its predecessor AlexNet.

As the efficiency and accuracy of GoogLeNet is more than the others, I chose GoogLeNet for the training of data. Also, tried training using AlexNet which did not give me results as expected.

For the training of default of data, default parameters as per the GoogLeNet are used but for the training of collected data, the number of epochs are reduced by half to get better results.

Data Acquisition:

As mentioned in the Introduction, three types of objects as shown in the figures 1,2,3 below are used.

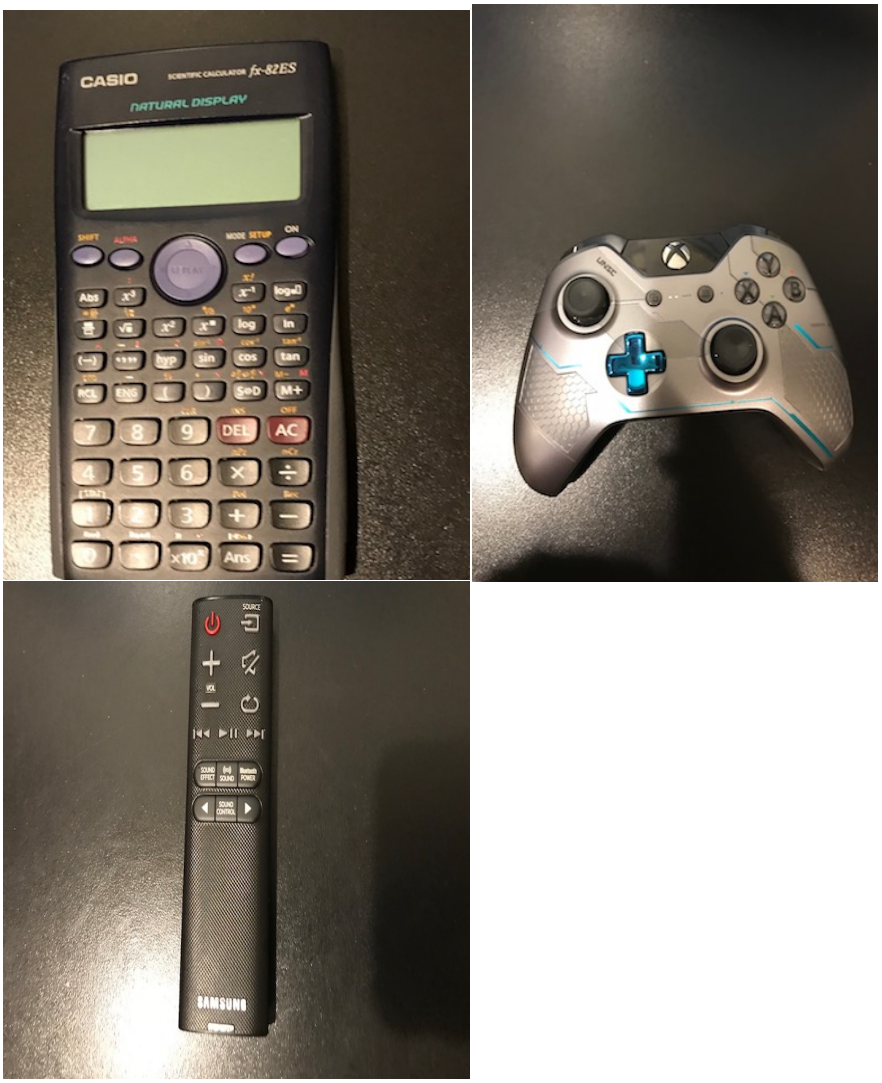


Fig: 1,2,3 from left to right(clockwise) 1. Scientific Calculator 2. Gaming Controller 3. TV Remote

These three objects are taken and captured the images with light and without light and on the same background. Various techniques like capturing them in different angles, different zoom in angles, close-ups and in the upside-down directions are used.

The total number of images collected are about 350 in number. Each object is captured by phone camera have the size of around 2MB. So, resizing them to 640X480 using a feature in mac, got the each file size of 40-50KB which saved time in uploading the images.

When tried to resize the images, the files automatically changed to .jpeg format. So, to convert all of them to .png format, a python script is used as shown in the figure 4 below.

Python Code: (taken from stack overflow and modified as per my use).

```
import os,sys
folder = 'Folder path'
for filename in os.listdir(folder):
        infilename = os.path.join(folder,filename)
        if not os.path.isfile(infilename): continue
        oldbase = os.path.splitext(filename)
        newname = infilename.replace('.jpeg', '.png')
        output = os.rename(infilename, newname)
```

Figure 4: Python Script

Results:

For the Default data, the required results of accuracy more than 75% and inference time of less than 10ms is achieved and it is as shown in the figure 5 below.

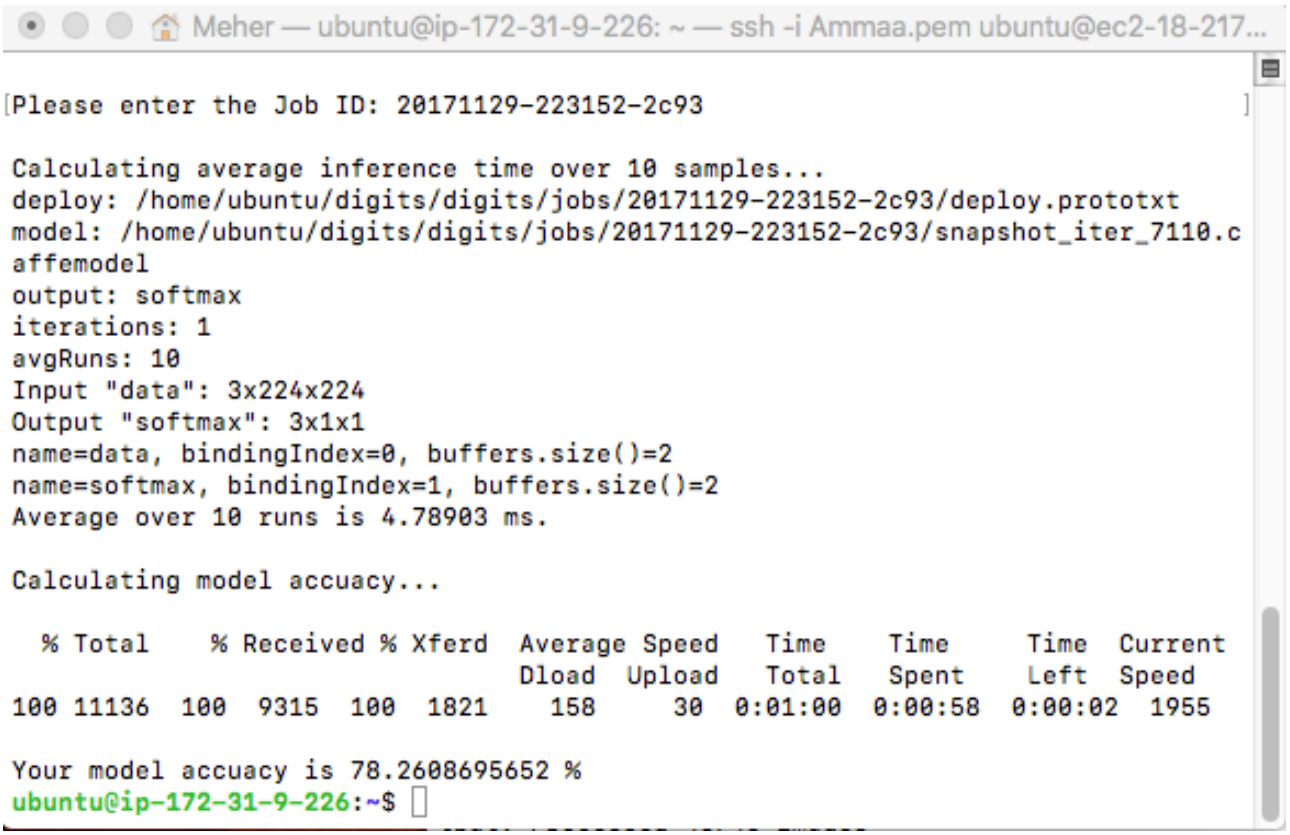


Figure 5: Results showing the inference time and accuracy for the default data.

The Graphical data for the results is shown in the figure 6 below.

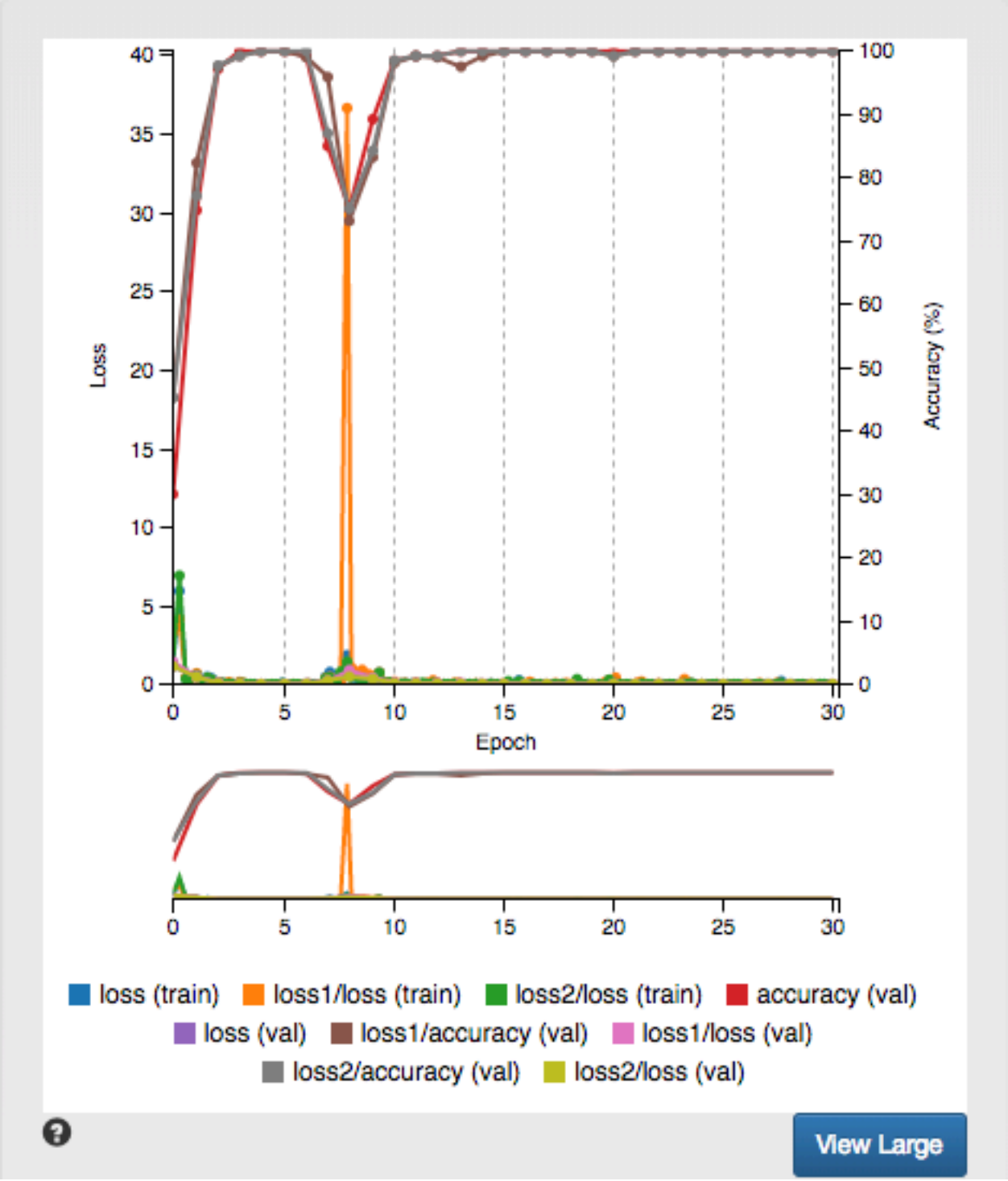


Figure 6. Screenshot showing accuracy and loss function values for default data.

The graph showing the accuracy and loss function values for the collected data is shown in the figure below.

Fig 7: Accuracy curve and learning curve across all epochs.

For the collected data, the network can detect the object with around 83% accuracy. A different color controller image is used and to test the accuracy of the classification which is shown in the figures 8,9 below.

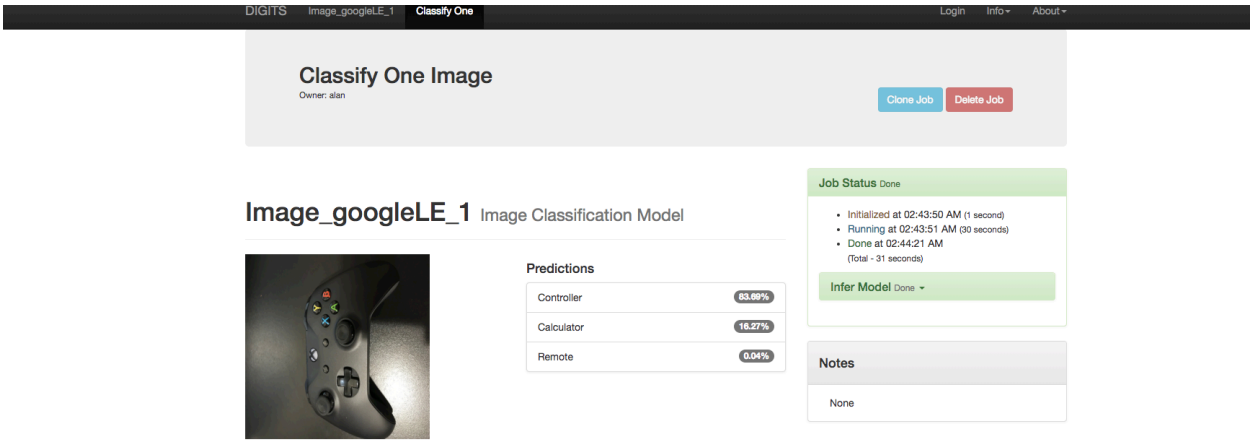


Figure 8: Image showing the results for test data not used in training the network.

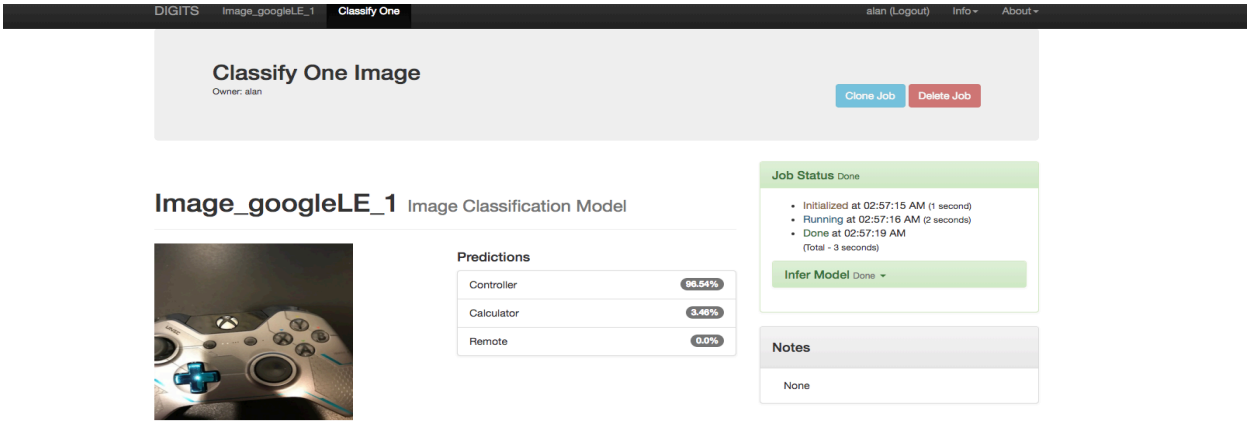


Figure 9: Image showing the results for the data used in training the network.

Even though a grey color controller image is used for training purposes more than 80% accuracy is achieved for the controller image of different color which is not used for training.

There are some things that are followed to get the good results for training.

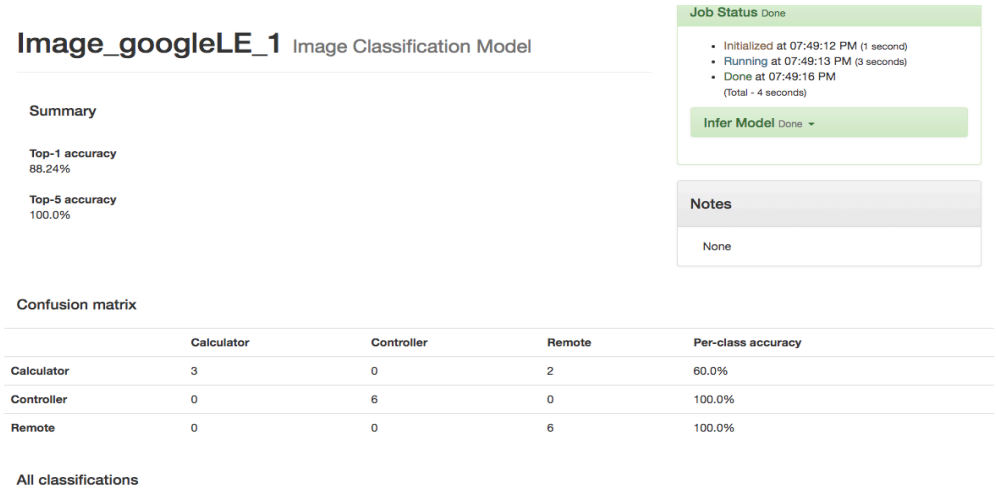
1. Plain backgrounds
2. No other objects in the background for training images.
3. If a design background is used for an object, then it should not be used for any other image in the training set.

Discussion:

The data collected for the Robotics Inference project is images of two different types of controllers and a calculator. So, if this project is applied in real world, the application would be mostly in AR industry where the object in real time is classified and detected on a mobile device. We all know accuracy is as important as the inference, but thanks to all the technology and the GPU, TPU's inference seems achievable these days.

I feel accuracy is more important than inference for the project as a network can be trained beforehand with lots of images and can be deployed using the current technology to meet the inference needs.

Since many years, the objects that are trained looks somewhat similar in size and shape and there is not much revolution and this works for now. So, if we feed more images then the required accuracy can be achieved. The confusion matrix for the objects can be shown in the below figure 10.



Image_googleLE_1 Image Classification Model

Summary

Top-1 accuracy
88.24%

Top-5 accuracy
100.0%

Job Status Done

- Initialized at 07:49:12 PM (1 second)
- Running at 07:49:13 PM (3 seconds)
- Done at 07:49:16 PM
  (Total - 4 seconds)

Infer Model Done

Notes

None

Confusion matrix

	Calculator	Controller	Remote	Per-class accuracy
Calculator	3	0	2	60.0%
Controller	0	6	0	100.0%
Remote	0	0	6	100.0%

All classifications

Figure 10: Showing the confusion matrix of images.

The above figure 10 shows the number of times the objects are classified as the ground truth. If we can see the objects controller and remote, they are classified correctly. But when it comes to the calculator it is misclassified more as remote (twice).

Futher improvements can be made to the project by feeding the more number of images to get the required accuracy.

Future Work:

The future improvements for this project can be adding controllers and remotes of different brands and training the network to detect the objects real time in an AR app which can be shaped out to a commercial product.

I want to work on this idea furthermore with Jetson TX2 as I can see the scope of this application.