Inference Project

Jeremy Olsen

Abstract—A review of the first project for Term 2 of the Robotics Software Engineer Nano-degree. It is an introduction into the implementation of Deep Neural Networks (DNNs) using the Nvidia DIGITS framework. The materials cover techniques on how to capture data, train and deploy models within accepted time and accuracy. A personal inference project is also embarked upon where the student starts gathering and training their model for later deployment and use on prescribed Nvidia TX2 hardware.

1 Introduction

In the first term, we were introduced to a wide variety of skills that a successful software robotics engineer will require in the workforce. The last project of the term focus upon Deep Learning and required us to build our own Tensor Flow network that would perform semantic segmentation. This project is building upon that knowledge with lessons on utilizing the DIGITS work flow by generating data sets to be used for training models by using popular networks (AlexNet, GoogLeNet, etc.) for object detection and image recognition tasks. There is also an optional section in which these models are deployed and ran on the Nvidia Jetson TX2 hardware, which was introduced and configured earlier in the classroom lessons.

The stated goals of using the supplied data portion of the project are as follows:

- Create a model using the provided data set
- Achieve a 75% accuracy rate
- Perform a measured inference time of under 10 ms

In addition, we are to start formulating and gathering data for our own inference idea. I am choosing to utilize the Detect-net project to find traffic cones as the basis to my idea. I am currently involved in competition where a robot will search out and touch traffic cones on a timed course. I included training pictures that are a part of the data set which is used for model training.

2 Background / Formulation

The first lab of the classrooms walked through the steps to properly configure your dataset and model training sessions, and most importantly, using the right model for the job. Choosing the right model requires some attention to the following details:

- Provide the model an input that it expects.
- Provide the end user (application) an output that is useful.

In the provided dataset example, it contained a set 256x256x3 images, all located in their respective classification folders. It was divided up into three classes and contained a combined total of 7,570 training images and 2,524 validation images. Since both AlexNet and GoogleNet accept these input image sizes and are a match for this example without any further image processing necessary.

Fig. 1. Traffic Cone Training Image

The output of this trained model is a list of vectors with values of the best match of the inferred image class. The application then needs to handle and display the best match on the camera output using these results.

The traffic cone detection portion of the project is also training using the DIGITS work flow, but rather using object detection as is basis. The dataset for object detection requires significantly more preparation then the classification example before the model can be trained. These github projects, Jetson-Inference [1] and a forked project called Blackjack [2], have served as the examples on how to setup and configure the model for training. There is also the additional step of annotating images, which is adding labels to the images by specifying the screen coordinates of bounding box(es) around the object being learned. The object recognition training is utilizing a customized version of a GoogleNet model which requires a 640x640x3 input image size. This list of additional parameters changes is what was suggested by the Jetson-Inference project [1] for successful training:

- Training epochs: 100

Fig. 2. Traffic Cone Training Image

- Subtract Mean: none
- Solver Type: Adam
- Base learning rate: 2.5e-05
- Policy: Exponential Decay
- Gamma: 0.99

3 Data Acquisition

The traffic-cone data set size is approximately the same size as the Blackjack example that is being followed. It contains 234 training images, and 37 validation images. I have been using my mobile phone to capture a variety of situations that the robot might encounter. The initial set of images are focused upon capturing series of single cone scenes and then augmenting with multiple cones in these same scenes. The scenes are scattered around our property and are taken from many different perspectives to ensure variable learning scenarios. The phone is capturing images at much higher resolution then the model requires and all the images need to be scaled and cropped before it can be annotated. During researching which annotation tool to use, a free photo editing application, FIJI [3], was used create a macro for the scaling and cropping of the raw images. After annotation is complete, the dataset needs to be split up into two separate directory grouping for training and validation. It's about an a 6:1 training to validation image ratio that is being used in the Blackjack example dataset.

The DIGITS workspace uses KITTI meta-data format for ingesting the detection bounding labels. This further explains the KITTI format and all of the variables. This format was the driving factor in choosing the correct annotation process. The Alps plugin [4] for FIJI was chosen from a list of suggested annotation tools by the classroom.

4 Results

It was possible to achieve the accuracy and inference time goals of the provided data set quickly by following the lab example for setting up the training parameters. The lab example used AlexNet as the training model and suggested 8 training epochs. In further testing, it was observed that by increasing the values of the training epochs nominally, the accuracy would suffer. This is a sign that over-training is occurring. The AlexNet was the only model used for training to save time on the instance for the other part of the project.

The traffic cone portion of the project has been difficult to obtain successful results. The only successes that have been observed is the ability create a dataset in DIGITS and train, deploy the model to the TX2 and run the application without errors. When running a single image test on the newly trained model, there were no bounding boxes observed on the results of the test image which suggests there is a problem with the configuration of the training of the model.

5 Discussion

The observed failures from the implementation of the traffic cone Object Detection example has led to a re-examining of the setup parameters and model configurations that were suggested for this particular training set. The next step in the discovery process is to download and train a model using the dataset that was used for Jetson-Inference and Blackjack projects. The goal is to replicate the other projects positive results which should provide a baseline to measure against, to help find possible discrepancies in either the amount of data used or training parameters chosen. The following reasons are going to be investigated as possible resolutions to the failures:

1) The initial dataset that was obtained is too small
2) The pre-trained model was incorrectly configured
3) Annotated labels are not using the correct KITTI formatting standard

Once the project is running correctly, the hope is that a fast inference time and high accuracy is easily achieved. From initial observations, the goal of having a low inference time should be prioritized over accuracy for the following reasons. A slow inference time, despite high accuracy, could compromise the downstream actions if too slow. The faster the inference, one could reasonably assume that the system could compensate for the lack of accuracy by processing for the detection of the object faster. As more data becomes available and added to the training dataset, this should also help with accuracy issues in the longterm. The inference times are a result of the underlying network architecture and should be addressed early in the development process rather then later.

6 Conclusion / Future work

The obvious future work of this project is to get the detection network fully running. Once that hurdle is completed, the work of applying it to a real world application is the next

steps. There are a few different problem that will need to be addressed:

- Determine when a traffic cone should be the target when multiple cones are detected.
- Are all cones detected actually part of the race or merely false positives?

The gathering of training data will be an on-going process and will require some subtle changes to the training parameters as the dataset grows.

Some other areas of research that should be considered is the idea of training a model for specific conditions and utilizing it when those condition exist. For example, would using a model trained at night vs. one in the day achieve better results depending upon the time of the day? Do cloudy condition, or even wet condition affect the training? Do models perform better when they are specialized or when they use a broad dataset? Lastly do different pre-trained models work better for different conditions? For now, the first step is to get a working model specific to optimal conditions to get a baseline for further testing and development.

REFERENCES

[1] dusty nv, "https://github.com/dusty-nv/jetson-inference."
[2] S4WRXTTCS, "https://github.com/s4wrxttcs/jetson-inference."
[3] Unknown, "http://fiji.sc/."
[4] Unknown, "https://alpslabel.wordpress.com/2017/01/26/alt/."