

To Detect Ploughed or Unploughed driveway using Google
TensorFlow's Object detection API

By

Abhilash Reddy Mandadi

An Internship Report submitted to
University at Albany
in partial fulfillment of the requirements
for the degree of
MASTER OF SCIENCE
in
Computer Science and Engineering

Prof. Siwei Lyu

05/04/2018

AUTHORIZATION FOR REPRODUCTION
OF WORK

I grant permission for the reproduction of this work in its entirety, without further authorization from me, on the condition that the person or agency requesting reproduction, absorb the cost and provide proper acknowledgment of authorship.

Date__05/04/2018_____

Abhilash R Mandadi,
79 Homestead Ave, Apt.2,
Albany.NewYork,12203

ABSTRACT

In this report I am about to discuss about the work I accomplished as a Computer Vision Intern at Centre for Technology in Government(CTG) in Albany. As an academic intern for its Student Technology Innovation lab Experience(STILE) program it is about Object detection using Google's Tensorflow machine learning framework. The project that we are working on is 'Fighting Urban Blight' project and the whole idea about the project is that we collected large amounts of data in the form of buildings' images in the capital region including Troy, Schenectady, Clifton park and Albany wherein the model would predict building's irregularities. So, if any state agency wants to see the updated information about the buildings they can access our website instead of going to the location physically. In this project, I am dealing with the back-end including setting up environment, data pre-processing, model configuration, training and testing model and presenting the results to our team.

As I am not fully aware of using Google's Object detection API, it took a quite a bit of time to understand the concepts involving this project. In this internship my task was to use a machine learning model to detect whether a driveway is ploughed or unploughed. I tried to train my own dataset with two classes and collected images and trained the model using these images on my workstation and while training the dataset I encountered few issues in my system which I would be describing in later sections. Finally, I was able to perform all the specified tasks assigned to me and was able to successfully train the model on the GPU.

TABLE OF CONTENTS

1	Introduction.....	6
2	Object Detection.....	7
2.1	Implementing Google’s Object Detection API.....	7
2.2	Preprocessing Data.....	9
2.2.1	Identifying Images.....	9
2.2.2	Label Images for Training.....	9
2.2.3	Generate TFRecords.....	10
2.3	Model Configuration.....	10
2.4	Training and Exporting the Model.....	10
2.5	Running the Model.....	12
2.6	Running the Model on GPU.....	12
2.7	Present Results to Team.....	16
3	Challenges.....	18
4	Future Work.....	19
5	Conclusion.....	19
6	References.....	19
7	Mentor Acknowledgement.....	21

TABLE OF FIGURES

Figure 1(a)- Model detecting a dog with 94% accuracy.....	8
Figure 1(b)- Model detecting person and kite with decent accuracy.....	8
Figure 2- LabelImg annotation tool used to label images.....	10
Figure 3- Creating TFRecords.....	10
Figure 4- Exporting the model to a tensorflow graph.....	12
Figure 5- Loss dropping down to almost less than 1 after 11000 steps during training.....	13
Figure 6(a)- CPU and Memory usage while training the model.....	14
Figure 6(b)- CPU and Memory usage while not training the model.....	14
Figure 7(a)- GPU description before training the model.....	15
Figure 7(b)- GPU description while training the model.....	15
Figure 8- Result when trained the model locally with fewer images.....	16
Figure 9- Result showing a Ploughed image while training on GPU with larger dataset.....	17
Figure 10- Result showing an Unploughed image while training on GPU with larger dataset..	17

1 INTRODUCTION

In this section I will give a brief overview of the work I have done in this internship. I started my internship with setting up of my system at my office. This was the first time I was working on a fully Linux based system (Ubuntu 16.04 Lts) so it was a different experience all together shifting from Windows to Ubuntu 16.04. It took me a couple of weeks to finally get started with the system as I went into trouble where I also need to reinstall entire operating system. When I started off with the system I was able to learn few basic command used in Linux like 'ls', 'rm', 'mv', 'clear' etc. as using Linux is pretty much like using command prompt in windows but we almost and always use Linux's 'terminal' to perform all the tasks.

After getting acquainted with the system I got the opportunity to explore the work done previously at CTG. They collected huge amounts of data of building's images in Albany, Clifton park and Troy and pre-processed some datasets and has some pre-defined classes. I also got the opportunity in exploring the tools that they have been using to manage team work including GitLab server, GPU and raspberry pi video sensor. CTG has its own Git sever where only the employees have the access to the server while the work done by the team members can be pushed to a repository on the server. The GPU being an NVIDIA GeForce is custom built at CTG and video sensor is built with raspberry pi μ C and GPS of 10HZ frequency and along with a camera. The video sensor is generally attached to a moving car to collect the images with the camera capturing the images with a new image every 200ms. The Overall work flow of the project is that collecting images, staging process, looking up for GPS values, running a detection model and storing into the Cassandra database.

Apart from the regular day to day tasks at CTG, the main task that I handled is running a detection model using tensorflow. The reason we continued to work with tensorflow framework is that Google has made good advancements in fields of machine learning and artificial intelligence with google having regular updates to its own object detection API. While one of my team member dealt with web interface, I was taking care of the back-end with running the machine learning model. My main goal/task assigned to me is to use machine learning to build a model that can detect a driveway that has not been cleared of its snow in a set of test images that contain plowed and unplowed driveways. The detailed description of this task is explained in the following sections

2 OBJECT DETECTION

Object detection is a technique where a machine learning model takes the input of pre-processed data and classifies the components of images into classes specified by the user and gives output along with the accuracy of object's detection. Few examples of object detection include face detection, pedestrian detection along with its applications in self driving cars, image retrieval and video surveillance.

2.1 IMPLEMENTING GOOGLE'S OBJECT DETECTION API

Before we start building any object detection model using tensorflow we must be able to implement the tensorflow object detection API provided by google. To start off we need to setup the environment by installing python. While I installed python 3.6.5 as python 3.x versions are the latest versions in use and are frequently updated whereas python 2.7 is standard with not much updates. After setting up python environment, one can install anaconda which makes installation of packages/libraries easier. One can even setup a virtual environment with anaconda so that all the work done is saved to the environment without affecting other files in the system. Though it caused some problems which I will discuss later in challenges section. So, if one must install anaconda it is downloaded based on the operating system and as well as the python you are using.

Then, we continued with installation of tensorflow depending on our system type and python version. The detailed description of installation is given in the tensorflow's open source website^[1]. Following we will install some important libraries as mentioned in object detection API^[2] including Jupyter, pillow, matplotlib, Cython, cocoapi, protobuf etc. After successful installation of all the libraries we can start building the model.

Before dealing with the specific task that was given to me at my workplace, I implemented default object detection model given in the API. The output is shown in the below figure (Fig. 1).



Fig. 1(a) Model detecting a dog with 94% accuracy.

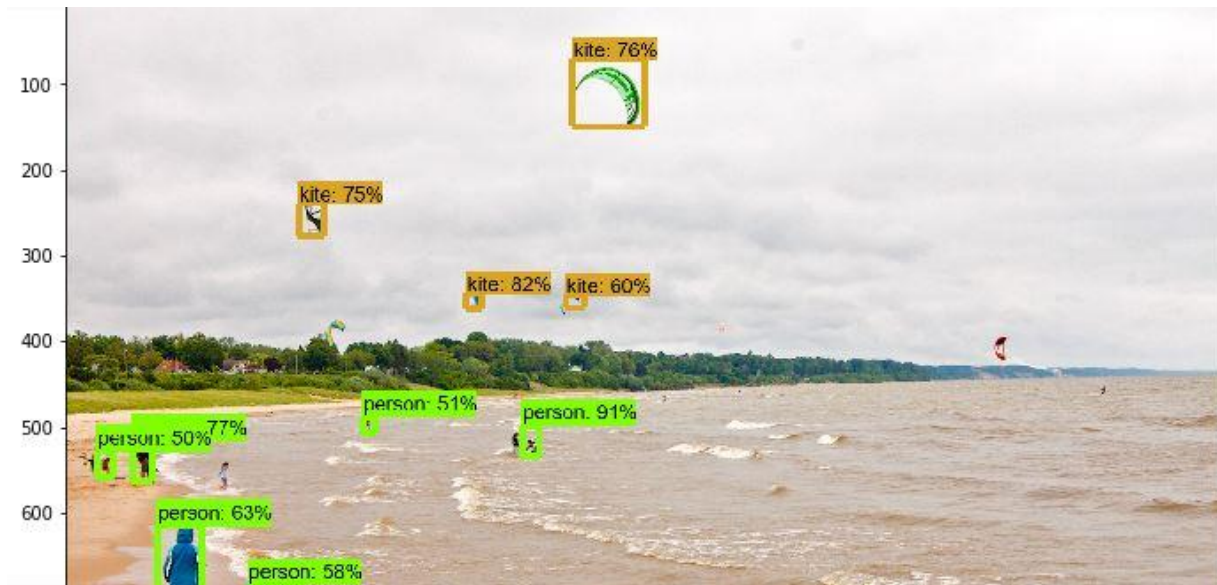


Fig. 1(b) Model detecting person and kite with decent accuracy.

At my work I was given the task to build a machine learning model to detect a driveway whether it is plowed or unplowed. To do this we collected thousands of snow images in capital region and trained with the model. The following sections describe the process in detail.

2.2 PREPROCESSING DATA

2.2.1 IDENTIFYING IMAGES

Though we were able to collect thousands of images, a lot of images in the datasets are not exactly the images we would like to use for training the model. So, I need to manually identify the images suitable for training. I also need to take care of the ratio of number of ‘plowed’ images to that of ‘unplowed’ images as we cannot train the model with either all images belonging to ‘plowed’ class or all of them belonging to ‘unplowed’ class. So, I maintained a ratio of about 55 percent of plowed ones and 45 percent of unplowed ones. Totally, I collected about 100 images from the available datasets.

2.2.2 LABEL IMAGES FOR TRAINING

After identifying and collecting images, we need to label the images in the required format. The annotation tool I used for this is LabelImg v1 which is a simple image annotation tool available online^[3]. The labelled images will be saved as xml files in PASCAL VOC format and for every image in jpeg format there will be a corresponding xml file. So, we got about 200 files with about 180 files in training folder and 20 in test folder. We also need to have ‘label map’ for training which describes the classes that are to be used while training the model. So, I configured my own label map with the name ‘snow_map’ which can be found in my github repository^[4].

After having train, test data and label map, we need to convert all the xml files to csv format using python script^[5]. So, finally we will have two csv files for both training and testing which would be the input to generate TFRecords^[6] files.



Fig. 2 LabelImg annotation tool used to label images.

2.2.3 GENERATE TFRECORDS

After labeling the images and having their xml files, we can generate TFRecords using a python script which can be in my repository^[7] |||||. Though, the python script is available in the object detection API^[8], I need to configure it with the label map and set the path with the current working directory. The API also provided detailed information on prepare our own dataset^[9] which was also helpful. At this stage I took only 10 images to train the model before I could run on the GPU with a bigger dataset.

```
C:\Users\Abhilash Reddy M\tensorflow\models\research\object_detection>python tfrecordGeneration.py --csv_input=data/test_labels.csv --output_path=test.record
Successfully created the TFRecords: C:\Users\Abhilash Reddy M\tensorflow\models\research\object_detection\test.record
C:\Users\Abhilash Reddy M\tensorflow\models\research\object_detection>
```

Fig. 3 Creating TFRecords

2.3 MODEL CONFIGURATION

We used the default SSD mobilenet v1 model which was used by default in tensorflow object detection tutorial. After selecting the model SSD mobilenet v1 model, we need to start configuring the model. This step is known as configuring object detection pipeline^[10]. In this step, I need to configure the ‘config’ file corresponding to the model chosen. While configuring the config file, we have five parts which we need take care off-

- a) Model configuration
- b) Parameters
- c) Metrics
- d) Training Input
- e) Evaluation Input

- a) In model configuration, we define the type of model, number of classes and the meta-architecture of the model. I edited the number of classes to 2 and pretty much left others to default^[11].
- b) These are the parameters which you can set for training. These include batch-size, learning rate etc. These too I left as default as my initial priority was to run the model successfully and later train a bigger dataset.
- c) The metrics we used here are PASCAL VOC metrics.
- d) In training input we set the path to the input TFRecord(train.record) and set path to label map(snow_map.pbtxt)
- e) Even in this, we set the path to input TFRecord(test.record) and to label map (snow_map.pbtxt).

2.4 TRAINING AND EXPORTING THE MODEL

To train the I have executed the python file, ‘train.py’^[12] in the terminal window using the command from research/object_detection folder “python train.py --logtostderr --train_dir=models/model/train/ --pipeline_config_path=models/model/ssd_mobilenet_v1_coco.config”. This command will

generate files in the training directory. At this point I noticed that my system being slowed down and I couldn't perform any other task except training.

After training we need to export the model to a tensorflow graph proto which is done by executing the following command “python export_inference_graph.py --input_type image_tensor --pipeline_config_path .\models\model\ssd_mobilenet_v1_coco.config --trained_checkpoint_prefix .\models\model\train\model.ckpt-0 --output_directory output_inference_graph.pb”. This would create a directory named output_inference_graph.pb in the object_detection folder.^[13]

```
C:\Users\Abhilash Reddy M\tensorflow\models\research\object_detection>python export_inference_graph.py --input_type image_tensor --pipeline_config_path .\models\model\ssd_m
obilenet_v1_coco.config --trained_checkpoint_prefix .\models\model\train\model.ckpt-0 --output_directory output_inference_graph.pb
WARNING:tensorflow:From C:\Users\Abhilash Reddy M\AppData\Local\Programs\Python\Python36\lib\site-packages\tensorflow\contrib\learn\python\learn\datasets\base.py:198: retry
(from tensorflow.contrib.learn.python.learn.datasets.base) is deprecated and will be removed in a future version.
Instructions for updating:
Use the retry module or similar alternatives.
WARNING:tensorflow:From C:\Users\Abhilash Reddy M\tensorflow\models\research\object_detection\exporter.py:351: get_or_create_global_step (from tensorflow.contrib.framework.
python.ops.variables) is deprecated and will be removed in a future version.
Instructions for updating:
Please switch to tf.train.get_or_create_global_step
2018-04-25 16:12:32.534209: I T:\src\github\tensorflow\tensorflow\core\platform\cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not
compiled to use: AVX2
Converted 199 variables to const ops.
```

Fig. 4 Exporting the model to a tensorflow graph.

2.5 RUNNING THE MODEL

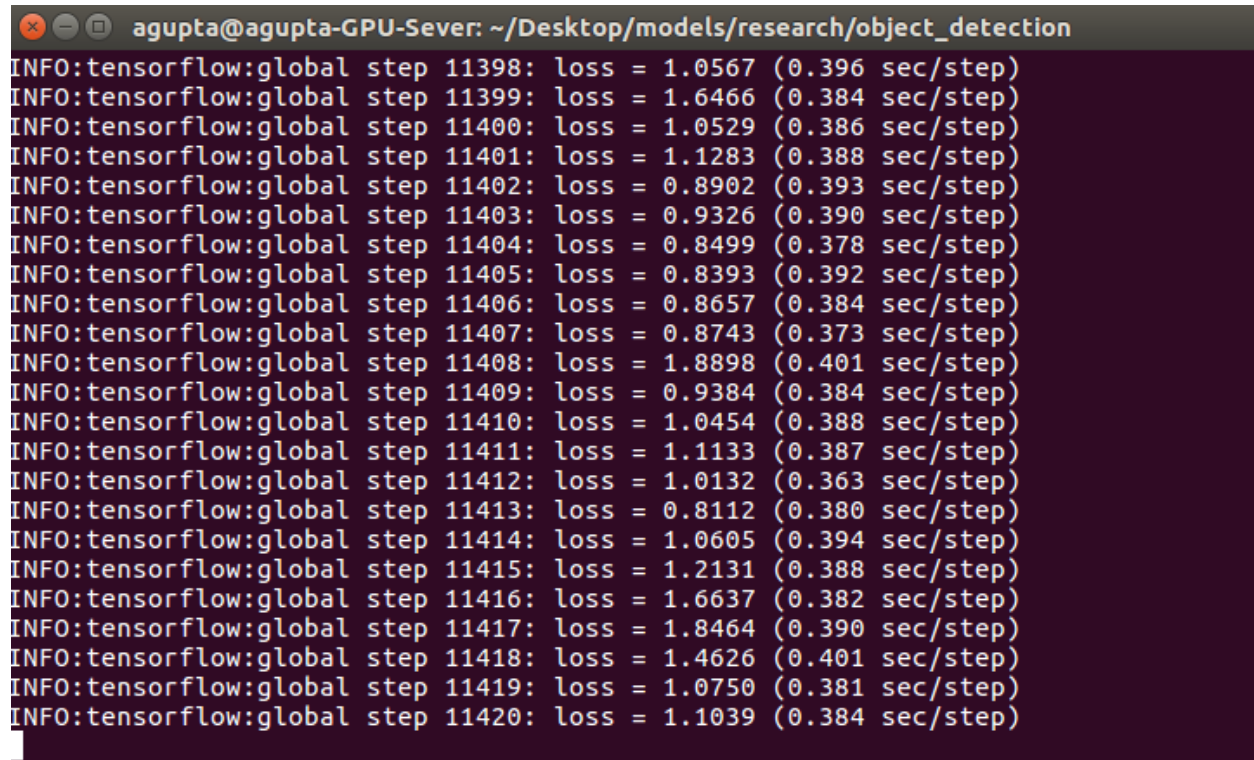
We can start running the model on the Jupyter notebook by using the command “jupyter note” in the terminal from models/research/ directory. Then, we need to set the path to PATH_TO_CKPT, PATH_TO_LABELS, TEST_IMAGE_PATHS variables to their appropriate folders. After saving the notebook, we can start running the notebook to get the desired results^[14].

2.6 RUNNING THE MODEL ON GPU

After successfully running the model on my system, I tried to train the model on the GPU with a larger dataset of about 100 images. As this system is already configured, I just need to start with the conversion of all the xml's to CSV file format. Then, I generated TFRecords and downloaded the model and made the required changes in the config. Then, I trained the model and

exported the tensorflow graph as done earlier. The detailed training process is described in the pdf file that I pushed to my repository^[15].

Initially at the start of training, the loss was between 3 and 4 and after about 11000 steps loss came down to almost less than 1 as shown in figure below.



```
agupta@agupta-GPU-Sever: ~/Desktop/models/research/object_detection
INFO:tensorflow:global step 11398: loss = 1.0567 (0.396 sec/step)
INFO:tensorflow:global step 11399: loss = 1.6466 (0.384 sec/step)
INFO:tensorflow:global step 11400: loss = 1.0529 (0.386 sec/step)
INFO:tensorflow:global step 11401: loss = 1.1283 (0.388 sec/step)
INFO:tensorflow:global step 11402: loss = 0.8902 (0.393 sec/step)
INFO:tensorflow:global step 11403: loss = 0.9326 (0.390 sec/step)
INFO:tensorflow:global step 11404: loss = 0.8499 (0.378 sec/step)
INFO:tensorflow:global step 11405: loss = 0.8393 (0.392 sec/step)
INFO:tensorflow:global step 11406: loss = 0.8657 (0.384 sec/step)
INFO:tensorflow:global step 11407: loss = 0.8743 (0.373 sec/step)
INFO:tensorflow:global step 11408: loss = 1.8898 (0.401 sec/step)
INFO:tensorflow:global step 11409: loss = 0.9384 (0.384 sec/step)
INFO:tensorflow:global step 11410: loss = 1.0454 (0.388 sec/step)
INFO:tensorflow:global step 11411: loss = 1.1133 (0.387 sec/step)
INFO:tensorflow:global step 11412: loss = 1.0132 (0.363 sec/step)
INFO:tensorflow:global step 11413: loss = 0.8112 (0.380 sec/step)
INFO:tensorflow:global step 11414: loss = 1.0605 (0.394 sec/step)
INFO:tensorflow:global step 11415: loss = 1.2131 (0.388 sec/step)
INFO:tensorflow:global step 11416: loss = 1.6637 (0.382 sec/step)
INFO:tensorflow:global step 11417: loss = 1.8464 (0.390 sec/step)
INFO:tensorflow:global step 11418: loss = 1.4626 (0.401 sec/step)
INFO:tensorflow:global step 11419: loss = 1.0750 (0.381 sec/step)
INFO:tensorflow:global step 11420: loss = 1.1039 (0.384 sec/step)
```

Fig. 5 Loss dropping down to almost less than 1 after 11000 steps during training.

While training the model on GPU, I was also able to learn few things about the processing power of GPU and its memory usage. The GPU that we are working on has Intel i5-7600K @ 3.80GHz x 4 quad core processor with TITAN Xp/PCIe/SSE2 graphics with the internal memory(RAM) as good as 50GB and a total disk space of 1TB. While training I was able to observe that almost 222.0% of CPU processing power is used along with 11% of its memory.

agupta@agupta-GPU-Sever: ~/Desktop/models/research/object_detection

top - 14:58:53 up 2:22, 1 user, load average: 4.69, 4.22, 4.10
Tasks: 212 total, 3 running, 209 sleeping, 0 stopped, 0 zombie
%Cpu(s): 61.8 us, 9.5 sy, 0.0 ni, 28.5 id, 0.0 wa, 0.0 hi, 0.2 si, 0.0 st
KiB Mem : 49388248 total, 38942252 free, 6991432 used, 3454564 buff/cache
KiB Swap: 16708604 total, 16708604 free, 0 used. 41542056 avail Mem

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
5079	agupta	20	0	35.856g	5.624g	518512	S	222.0	11.9	167:23.65	python
960	root	20	0	343204	88736	66528	S	39.3	0.2	13:48.33	Xorg
10600	agupta	20	0	729748	41732	33860	S	9.3	0.1	3:31.49	psensor
2089	agupta	20	0	1700636	260680	82320	R	6.0	0.5	4:10.04	compiz
1008	root	-51	0	0	0	0	S	1.3	0.0	2:06.69	irq/131-nv+
5044	agupta	20	0	663092	38064	28352	S	0.7	0.1	0:23.60	gnome-term+
32113	agupta	20	0	427928	21872	18512	S	0.7	0.0	0:00.19	gnome-scre+
8	root	20	0	0	0	0	S	0.3	0.0	0:03.80	rcu_sched
1721	agupta	20	0	524120	28340	22200	S	0.3	0.1	0:06.89	bamfdaemon
6153	agupta	20	0	13960	2596	2240	S	0.3	0.0	0:12.41	watch
19241	root	20	0	0	0	0	S	0.3	0.0	0:00.71	kworker/3:2
1	root	20	0	185132	5752	3976	S	0.0	0.0	0:01.17	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
4	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/0:+
6	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	mm_percpu_+
7	root	20	0	0	0	0	S	0.0	0.0	0:00.07	ksoftirqd/0
9	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_bh

Fig. 6(a) CPU and Memory usage while training the model.

agupta@agupta-GPU-Sever: ~/Desktop/models/research/object_detection

top - 15:05:35 up 2:29, 1 user, load average: 2.30, 3.94, 4.09
Tasks: 203 total, 2 running, 201 sleeping, 0 stopped, 0 zombie
%Cpu(s): 17.9 us, 10.5 sy, 0.0 ni, 70.8 id, 0.0 wa, 0.0 hi, 0.7 si, 0.0 st
KiB Mem : 49388248 total, 44606872 free, 1463376 used, 3318000 buff/cache
KiB Swap: 16708604 total, 16708604 free, 0 used. 47208364 avail Mem

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
10600	agupta	20	0	729748	41732	33860	S	33.9	0.1	4:20.53	psensor
960	root	20	0	343376	88836	66556	S	28.9	0.2	15:45.71	Xorg
2089	agupta	20	0	1683412	261896	82320	R	20.6	0.5	4:39.25	compiz
1008	root	-51	0	0	0	0	S	3.0	0.0	2:13.80	irq/131-nv+
10467	agupta	20	0	427980	21944	18532	S	2.3	0.0	0:00.14	gnome-scre+
6153	agupta	20	0	13960	2596	2240	S	1.7	0.0	0:14.30	watch
1721	agupta	20	0	524120	28340	22200	S	0.7	0.1	0:07.99	bamfdaemon
5044	agupta	20	0	663776	38740	28816	S	0.7	0.1	0:26.30	gnome-term+
1	root	20	0	185132	5752	3976	S	0.0	0.0	0:01.17	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
4	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/0:+
6	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	mm_percpu_+
7	root	20	0	0	0	0	S	0.0	0.0	0:00.08	ksoftirqd/0
8	root	20	0	0	0	0	S	0.0	0.0	0:04.21	rcu_sched
9	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_bh
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.03	migration/0
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.01	watchdog/0

Fig. 6(b) CPU and Memory usage while not training the model.


```

agupta@agupta-GPU-Sever: ~/Desktop/models/research/object_detection
Every 0.1s: nvidia-smi
Mon Apr 30 15:06:27 2018

+-----+
| NVIDIA-SMI 384.111                Driver Version: 384.111          |
+-----+-----+
| GPU  Name      Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
+-----+-----+
|  0  TITAN Xp    Off         | 00000000:01:00.0 On  |      N/A      Default |
|29%   41C    P8     18W / 250W | 369MiB / 12186MiB |    25%          |
+-----+-----+

+-----+
| Processes:                         GPU Memory Usage |
+-----+-----+
| GPU       PID    Type    Process name                  |  Usage |
+-----+-----+
|    0      960     G   /usr/lib/xorg/Xorg              | 222MiB |
|    0     2089     G   compiz                        | 145MiB |
+-----+

```

Fig. 7(a) GPU description before training the model.

```

agupta@agupta-GPU-Sever: ~/Desktop/models/research/object_detection
Every 0.1s: nvidia-smi
Mon Apr 30 14:57:47 2018

+-----+
| NVIDIA-SMI 384.111                Driver Version: 384.111          |
+-----+-----+
| GPU  Name      Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
+-----+-----+
|  0  TITAN Xp    Off         | 00000000:01:00.0 On  |      N/A      Default |
|41%   66C    P2     89W / 250W | 11764MiB / 12186MiB |    35%          |
+-----+-----+

+-----+
| Processes:                         GPU Memory Usage |
+-----+-----+
| GPU       PID    Type    Process name                  |  Usage |
+-----+-----+
|    0      960     G   /usr/lib/xorg/Xorg              | 238MiB |
|    0     2089     G   compiz                        | 145MiB |
|    0     5079     C   python                        | 11377MiB |
+-----+

```

Fig. 7(b) GPU description while training the model.

2.7 PRESENT RESULTS TO TEAM

The initial results when I trained the model locally were not so good as shown in the figure below.

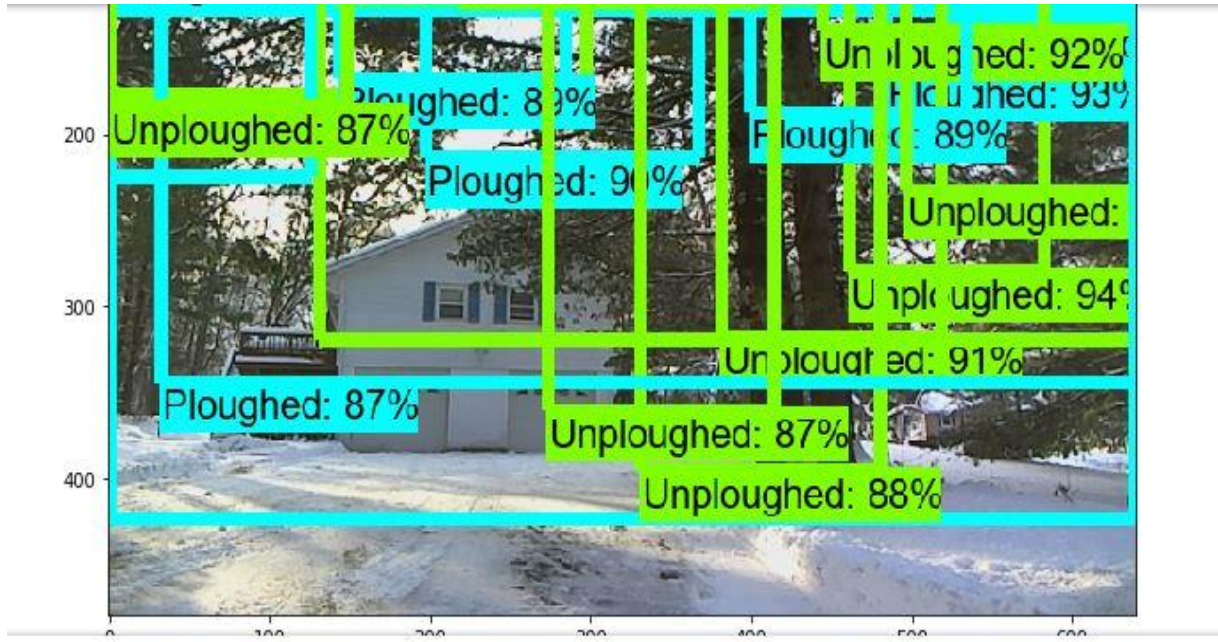


Fig. 8 Result when trained the model locally with fewer images.

But training locally helped me know how to configure and train the model. Finally, I got good results on the GPU while using all the 100 labelled images to train the model. The results on GPU are decent enough with model detecting the driveway whether it is ploughed or unploughed. As shown in the below figures Fig. 9 and Fig. 10, model in certain cases was able to detect a ploughed driveway with almost 99% accuracy and same with the unploughed driveway^[16].

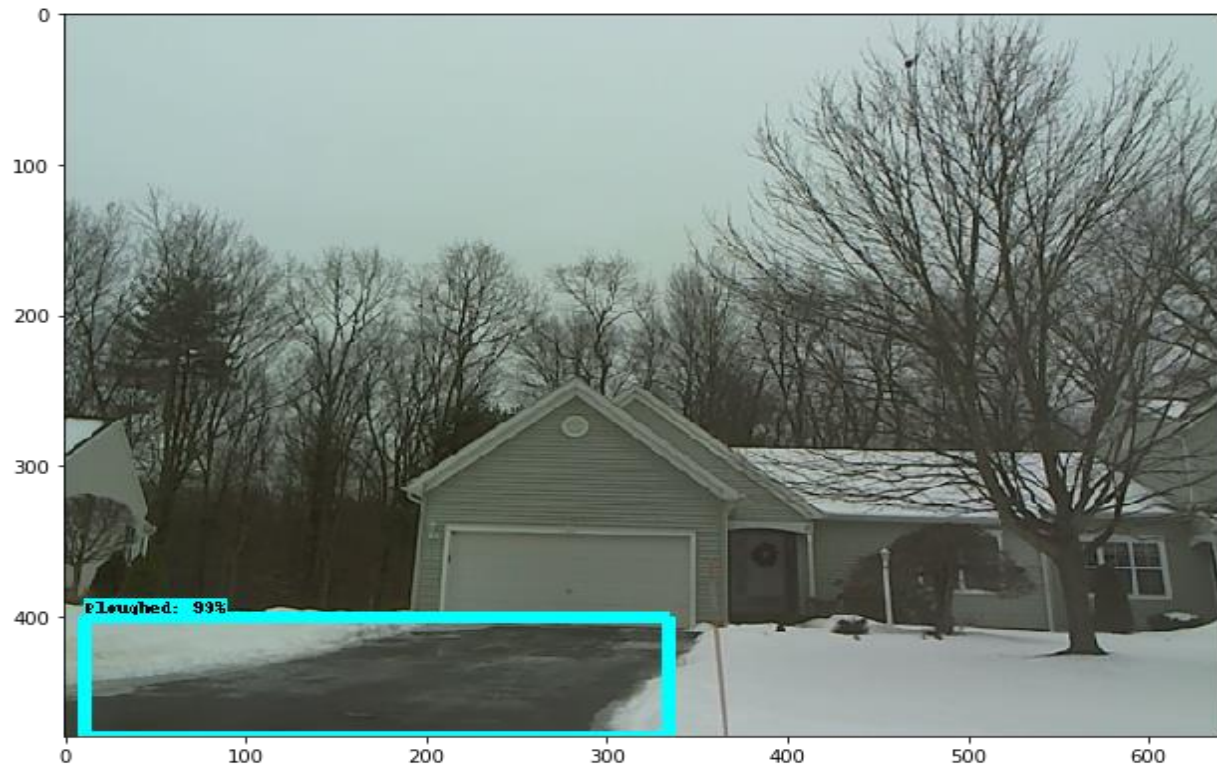


Fig. 9 Result showing a Ploughed image while training on GPU with larger dataset.

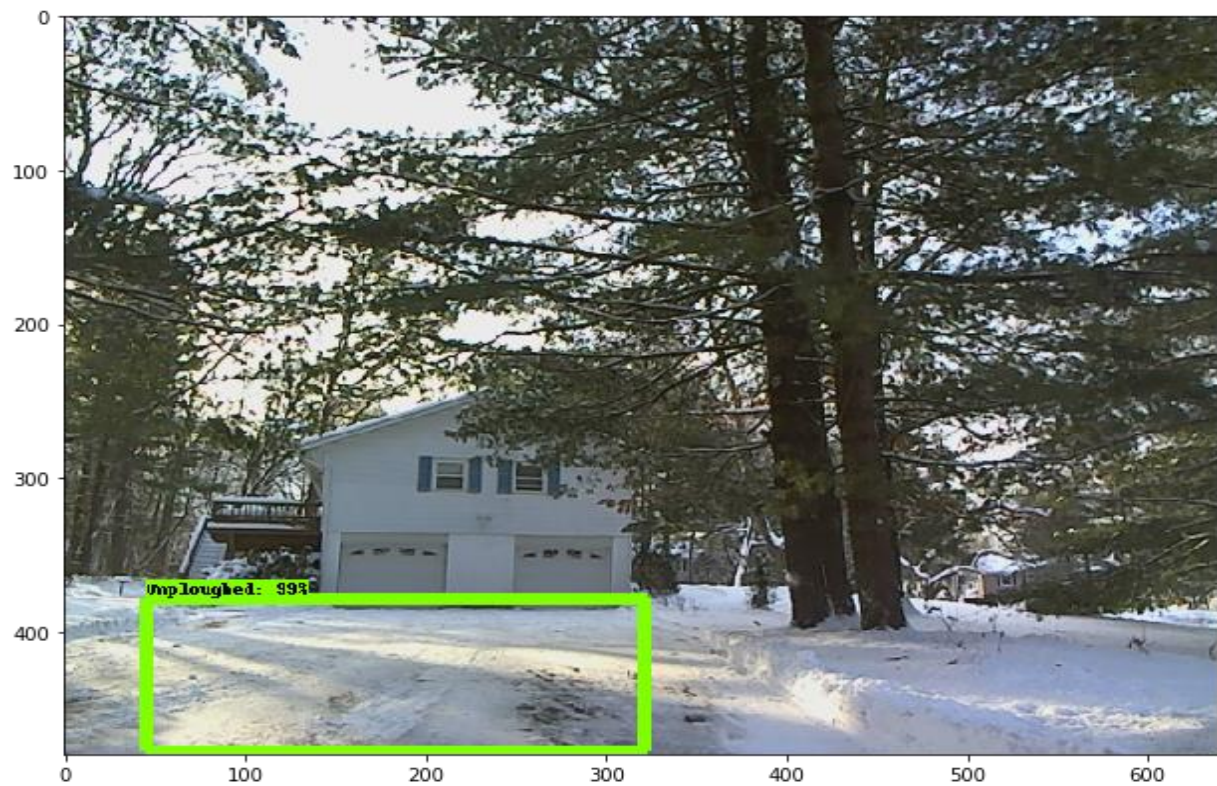


Fig. 10 Result showing an Unploughed image while training on GPU with larger dataset.

2 CHALLENGES

I have faced a lot of challenges while doing this internship. Some of them are very much time consuming that I also need to redo the entire work from scratch. Though there were a lot of issues I faced, I will explain few major ones.

- At the start of my internship, I have to use a different workstation which is linux based OS, ubuntu 16.04 LTS. Though I have used linux in a virtual box before, this is the first time I used a fully linux based system. While installation of libraries according to API^[17], it lead to system crash. So, I need to reinstall entire OS and configure API again.
- Though I had my own workstation to work on at CTG, I also used my own system to learn and complete certain tasks. As my system is windows-based OS, I installed Ubuntu 16.04 along with windows and continued working. Though initially I didn't face any issue, there was a time where there is a conflict between libraries which again lead to crash of entire system. This resulted in a huge loss as I didn't have any backup of my data. After this I tried to use only a single OS which is windows on my system and linux at CTG.
- I also faced some issues while installing certain libraries like PyQt5 and SIP. Installing these libraries is a tedious task and I relied on the articles in GitHub and stackoverflow to solve these issues.
- I also faced problems while conversion of xml to csv files. Then I had to do some python debugging in setting the path to training images.
- While doing this internship I was also able to create virtual environments in the system. At one point I have two virtual environments with python 3.6 working on it while it was python 2.7 installed on the main system. These multiple environments with different python versions also caused problems.
- After configuring the object detection API, configuring and training the model, I wondered when I got an error "ModuleNotFoundError: No module named pycocotools.mask". This error would arise only if I didn't follow the previous steps correctly but doing a good research in one of the blogs in Microsoft^[18] finally helped me resolve the issue.

3 FUTURE WORK

In future we would like to use different models^[19] including RCNN, Resnet, SSD inception to train on these datasets and compare the results. In fact, our Director at CTG suggested to design an automation system to able to use effortlessly where one can choose the model of his choice without worrying about the configuration of the entire object detection model.

4 CONCLUSION

I think it's a great opportunity for students to work on newer technologies like machine learning, deep learning which is might be the future advancements in science. I am privileged to work with the Derek Werthmuller, Director of Technology Innovation and services to be my mentor at Centre for Technology in Government (CTG) and always tried to motivate me to complete my task and without his guidance and support it would not be possible to successfully complete my task in time. I would also like to thank Prof. Siwei Lyu for being my advisor for class CSI698.

6 REFERENCES

[1] <https://www.tensorflow.org/install/>

[2]https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/installation.md

[3] <https://github.com/tzutalin/labelImg>

- [4] https://github.com/abhi-1111/CSI698/blob/master/research/object_detection/data/snow_map.pbtxt
- [5] https://github.com/abhi-1111/CSI698-GPU/blob/master/xml_to_csv.py
- [6] https://www.tensorflow.org/api_guides/python/python_io#tfrecords_format_details
- [8] https://github.com/tensorflow/models/tree/master/research/object_detection/dataset_tools
- [9] https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/using_your_own_dataset.md
- [10] https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/configuring_jobs.md
- [11] https://github.com/abhi-1111/CSI698/blob/master/research/object_detection/models/model/ssd_mobilenet_v1_coco.config
- [12] https://github.com/abhi-1111/CSI698/blob/master/research/object_detection/train.py
- [13] https://github.com/abhi-1111/CSI698/tree/master/research/object_detection/output_inference_graph.pb
- [14] https://github.com/abhi-1111/CSI698/blob/master/research/object_detection/object_detection_tutorial-Copy1.ipynb
- [15] https://github.com/abhi-1111/CSI698-GPU/blob/master/GPU_Training.pdf

[16] https://github.com/abhi-1111/CSI698-GPU/blob/master/research/object_detection/object_detection_tutorial-Copy1.ipynb


[17] https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/installation.md

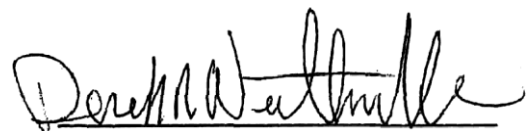
[18] <https://blogs.msdn.microsoft.com/pythonengineering/2016/04/11/unable-to-find-vcvarsall-bat/>

[19] https://github.com/abhi-1111/CSI698/tree/master/research/object_detection/samples/configs

7 MENTOR ACKNOWLEDGEMENT

I hereby acknowledge that I have reviewed the Internship report submitted by Mr. Abhilash Reddy Mandadi and found it to be satisfactory.


Student


Mentor