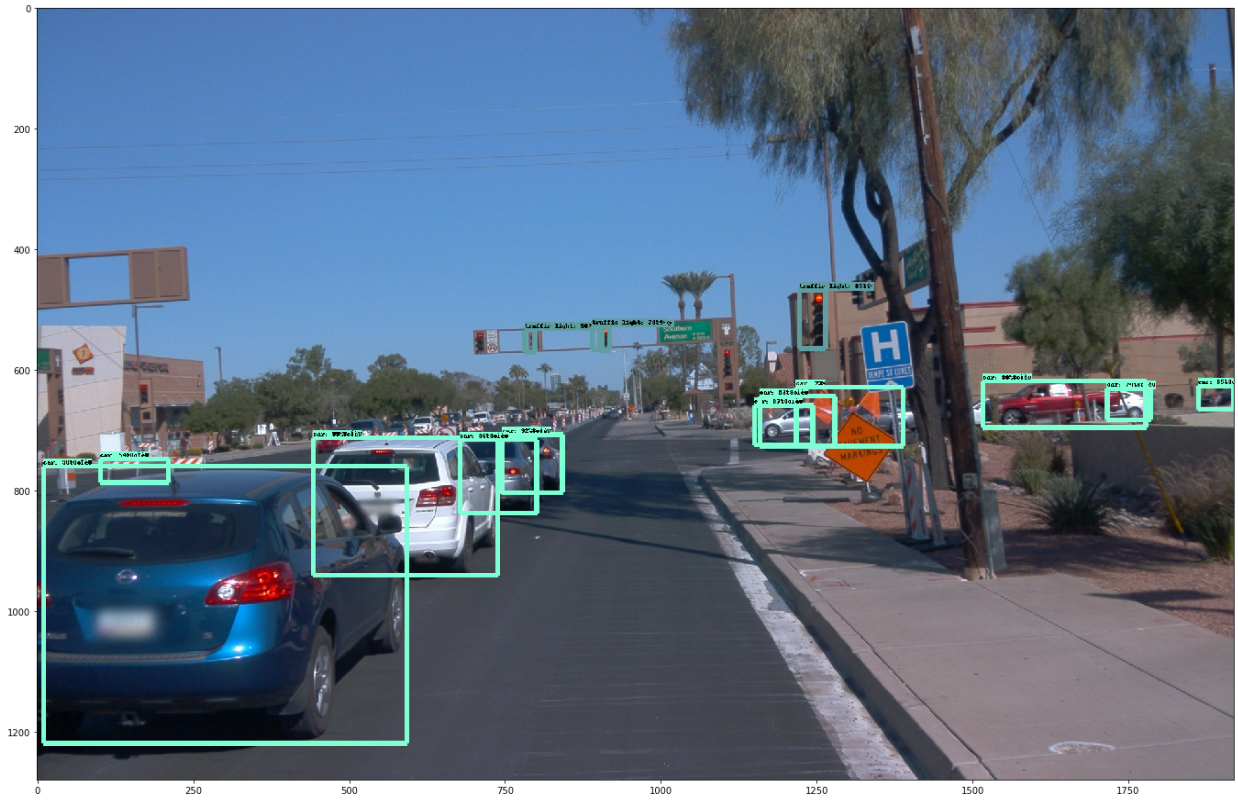
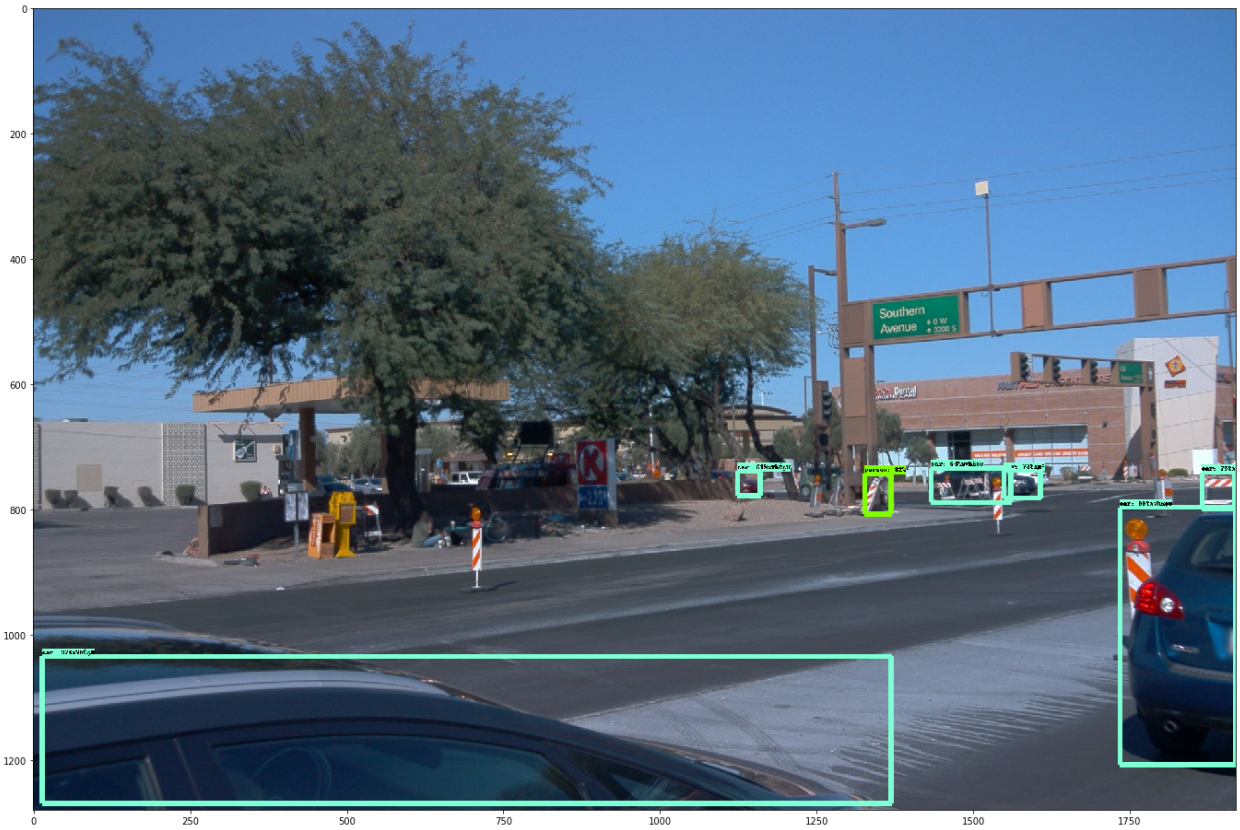
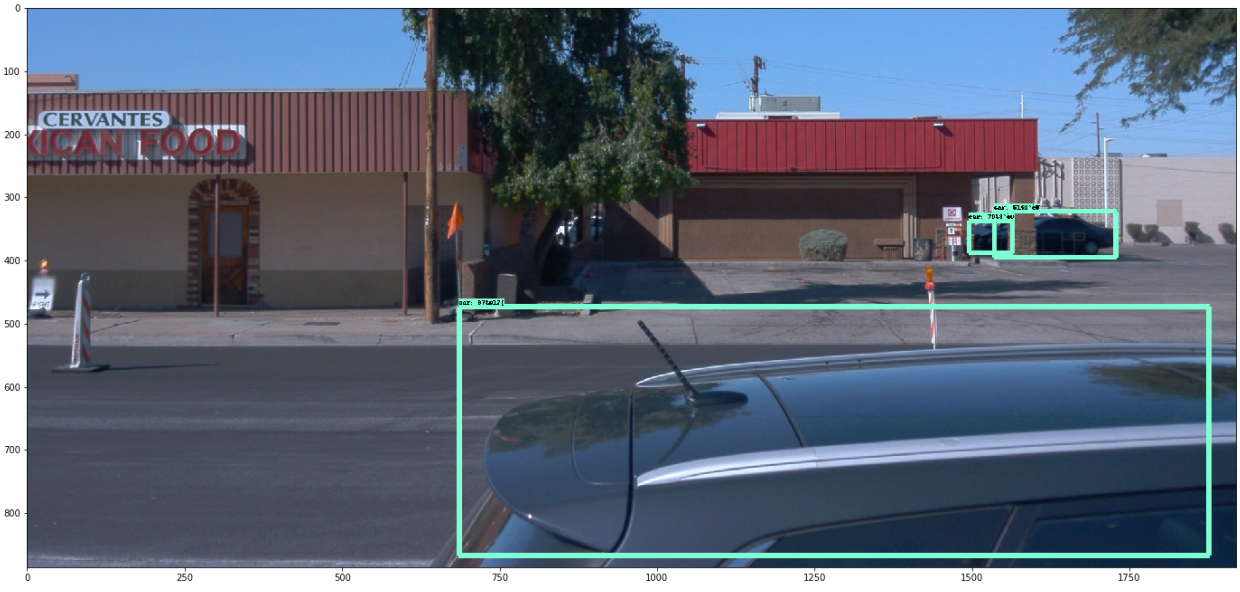
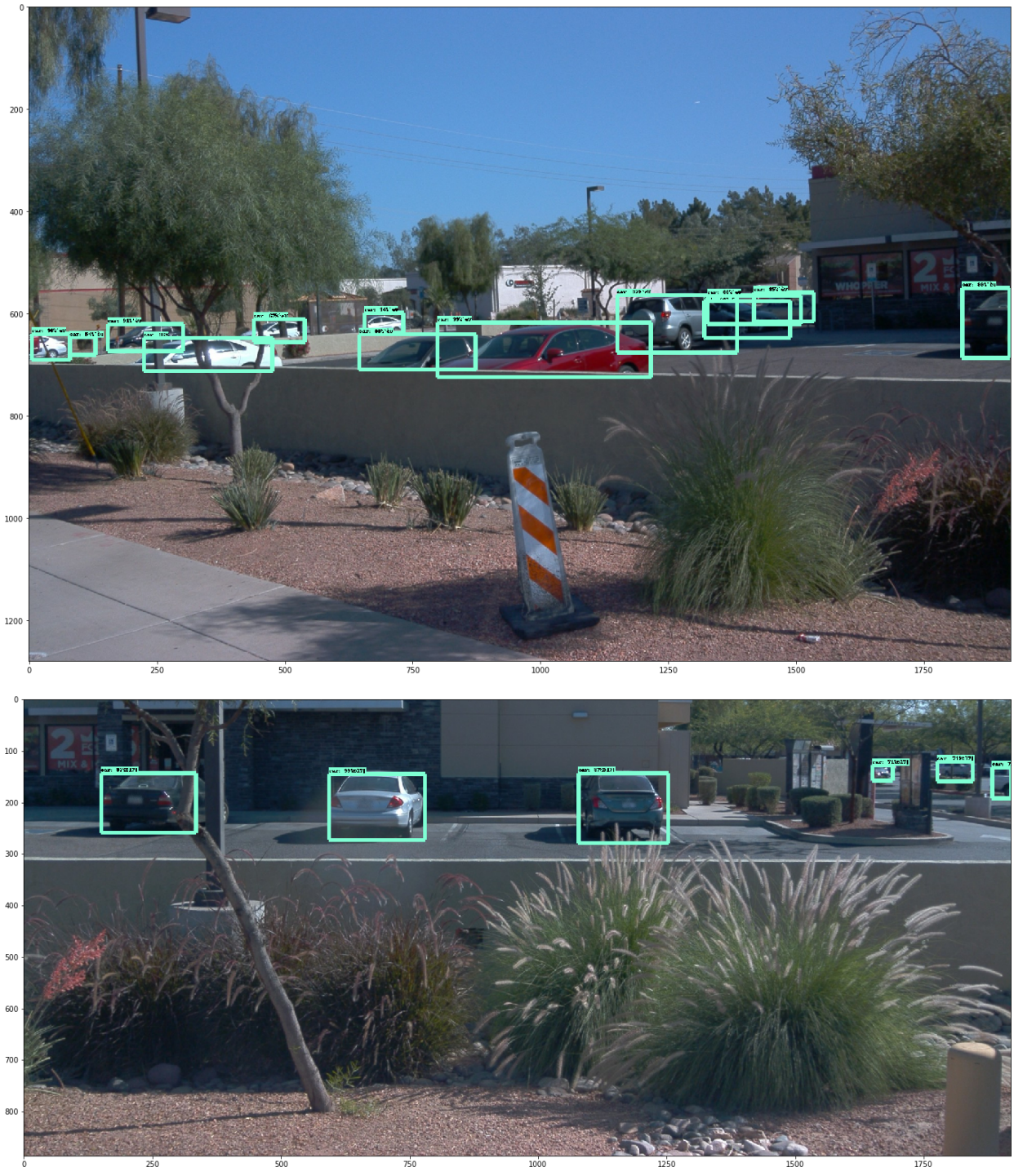
1. The Waymo dataset contains both labeled images and Lidar point cloud data. Our goal was to train a model on Waymo's images, evaluate results, and export the model.
2. The model chosen to do object detection and training with was faster_rcnn_resnet101_coco. The model's folder was downloaded from the Tensorflow Object Detection API Github.
3. This model was already pretrained, so object detection was performed on images from the Waymo dataset to see how well the model performed before any further training.

Untrained model results:




Southern
Avenue


CERVANTES



4. Training the model on new data required train and test TFRecords. The Waymo dataset was already formatted as several TFRecords. Inside each TFRecord were images with bounding boxes and class labels. At first, we attempted to directly use Waymo's own TFRecords to do training. However, they weren't formatted to be used for training.

5. Instead, brand-new TFRecords needed to be made. Two folders were created, train and test. A script was made to extract the original, unlabeled camera images from one of Waymo's TFRecords, that is, the images that do not have bounding boxes. These images were placed in the train and test folders.

   Image script:

```
count = 0

# image width and height added to an array, to use in csv file later
widthList = []
heightList = []

import PIL.Image
from io import BytesIO
import IPython.display
import numpy as np
def showarray(a, idx, wList, hList, cnt, fmt='png'):
    a = np.uint8(a)
    f = BytesIO()

    filename = "image" + str(idx+cnt) + ".png"
    result = PIL.Image.fromarray(a)

    width, height = result.size

    wList.append(width)
    hList.append(height)

for index, image in enumerate(frame.images):
  showarray(tf.image.decode_jpeg(image.image), index, widthList, heightList, count)
```

6. Next, a new train CSV file was created. The image filename, bounding box coordinates, and classes were extracted into this CSV file. The process was repeated for a similar test CSV file.

   CSV script:

```
# create a labels csv file
import csv

with open('train_labels.csv', 'w') as csvfile:
    filewriter = csv.writer(csvfile, delimiter=',',
                            quotechar='|', quoting=csv.QUOTE_NONE) # QUOTE_MINIMAL
    filewriter.writerow(['filename', 'width', 'height', 'class', 'xmin', 'ymin', 'xmax', 'ymax'])
```

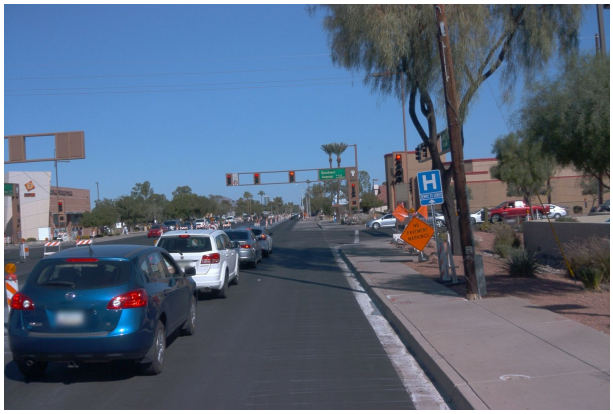
```
    xmin = int(label.box.center_x - 0.5 * label.box.length)
    ymin = int(label.box.center_y - 0.5 * label.box.width)
    xmax = int(label.box.center_x + 0.5 * label.box.length)
    ymax = int(label.box.center_y + 0.5 * label.box.width)

    row_class = ''
    if label.type == 1:
      row_class = "vehicle"
    elif label.type == 2:
      row_class = "pedestrian"
    elif label.type == 3:
      row_class = "sign"
    elif label.type == 4:
      row_class = "cyclist"
    else:
      row_class = "unknown"

    # if idx == 2:
    row = [filename, img_width, img_height, row_class, xmin, ymin, xmax, ymax]
    writer.writerow(row)

csvFile.close()
```

7. One of the resulting images and CSV files:



	A	B	C	D	E	F	G	H
1	filename	width	height	class	xmin	ymin	xmax	ymax
2	image0.pr	1920	1280	vehicle	887.6997	658.4326	900.9631	669.1696
3	image0.pr	1920	1280	vehicle	1143.494	663.1695	1280.549	712.4335
4	image0.pr	1920	1280	vehicle	699.8016	669.1696	733.9075	696.328
5	image0.pr	1920	1280	vehicle	587.3787	677.3803	602.5369	685.5909
6	image0.pr	1920	1280	vehicle	841.5937	662.2221	854.8571	675.4855
7	image0.pr	1920	1280	vehicle	649.2744	667.5907	692.8541	704.2229
8	image0.pr	1920	1280	vehicle	1892.559	675.8013	1920	692.2226
9	image0.pr	1920	1280	pedestriar	231.4777	703.2755	251.057	746.8552
10	image0.pr	1920	1280	vehicle	761.6975	666.3275	787.5927	687.8016
11	image0.pr	1920	1280	vehicle	1510.29	615.1687	1738.61	694.1174
12	image0.pr	1920	1280	vehicle	730.6439	696.2231	855.0674	803.1724
13	image0.pr	1920	1280	vehicle	503.8508	678.6435	562.9045	718.4336
14	image0.pr	1920	1280	vehicle	1695.03	627.4058	1787.243	677.7754
15	image0.pr	1920	1280	vehicle	455.6922	693.17	522.0091	739.2761
16	image0.pr	1920	1280	vehicle	1854.664	617.3792	1919.718	666.0117
17	image0.pr	1920	1280	vehicle	316.4266	701.6965	337.9007	718.1178
18	image0.pr	1920	1280	vehicle	10.73703	754.1185	622.1162	1179.81
19	image0.pr	1920	1280	vehicle	931.2795	658.4326	938.2269	671.696
20	image0.pr	1920	1280	vehicle	638.2217	680.2224	664.1169	695.3806

8. The train and test CSV files were converted into train and test TFRecords.
   The following tutorial was used:
   https://pythonprogramming.net/creating-tfrecord-files-tensorflow-object-detection-api-tutorial/
   And: https://github.com/datitran/raccoon_dataset
   with generate_tfrecord.py

   Command used:

```
python3 generate_tfrecord.py --csv_input=data/train_labels.csv --
output_path=data/train.record --image_dir=images/

python3 generate_tfrecord.py --csv_input=data/test_labels.csv --
output_path=data/test.record --image_dir=images/
```

Now, in your data directory, you should have `train.record` and `test.record`.

9. Inside the detection model's folder a configuration file, pipeline.config, was created. It was based off a sample config file but with paths changed to the train TFRecord, test TFRecord, and mscoco_label_map.pbtxt.

   Pipeline config:

```
train_input_reader: {
  tf_record_input_reader {
    input_path: "/data/cmpe297-02-fa2019/michaelhsieh/train/train.record"
  }
  label_map_path: "/home/010151297/models/research/object_detection/data/mscoco_
label_map.pbtxt"
}

eval_config: {
  metrics_set: "coco_detection_metrics"
  num_examples: 1101
}

eval_input_reader: {
  tf_record_input_reader {
    input_path: "/data/cmpe297-02-fa2019/michaelhsieh/validation/test.record"
  }
  label_map_path: "/home/010151297/models/research/object_detection/data/mscoco_
label_map.pbtxt"
  shuffle: false
  num_readers: 1
}
```

10. Now that there was a train record, test record, and pipeline.config, the model could be trained. The training code was run, specifying the pipeline config path, and after a long time the model finished training.

11. At first, the model was trained on 30 images, 25 train and 5 test. It turned out Waymo's labels didn't match the labels expected by the coco label map, which resulted in the model not displaying any results correctly, so these all needed to be changed to match the coco labels. For example, vehicle became car and pedestrian became person. After these changes, the model was trained on 205 train images and 5 test images.

    Modified script:

```
    xmin = int(label.box.center_x - 0.5 * label.box.length)
    ymin = int(label.box.center_y - 0.5 * label.box.width)
    xmax = int(label.box.center_x + 0.5 * label.box.length)
    ymax = int(label.box.center_y + 0.5 * label.box.width)

    row_class = ''
    if label.type == 1:
      row_class = "car"
    elif label.type == 2:
      row_class = "person"
    elif label.type == 3:
      row_class = "traffic light"
    elif label.type == 4:
      row_class = "bicycle"
    else:
      row_class = "unknown"

    # if idx == 2:
    row = [filename, img_width, img_height, row_class, xmin, ymin, xmax, ymax]
    writer.writerow(row)

csvFile.close()
```

Modified CSV file:

	A	B	C	D	E	F	G	H
1	filename	width	height	class	xmin	ymin	xmax	ymax
2	image0.pr	1920	1280	car	887	658	900	669
3	image0.pr	1920	1280	car	1143	663	1280	712
4	image0.pr	1920	1280	car	699	669	733	696
5	image0.pr	1920	1280	car	587	677	602	685
6	image0.pr	1920	1280	car	841	662	854	675
7	image0.pr	1920	1280	car	649	667	692	704
8	image0.pr	1920	1280	car	1892	675	1920	692
9	image0.pr	1920	1280	person	231	703	251	746
10	image0.pr	1920	1280	car	761	666	787	687
11	image0.pr	1920	1280	car	1510	615	1738	694
12	image0.pr	1920	1280	car	730	696	855	803
13	image0.pr	1920	1280	car	503	678	562	718
14	image0.pr	1920	1280	car	1695	627	1787	677
15	image0.pr	1920	1280	car	455	693	522	739
16	image0.pr	1920	1280	car	1854	617	1919	666
17	image0.pr	1920	1280	car	316	701	337	718
18	image0.pr	1920	1280	car	10	754	622	1179
19	image0.pr	1920	1280	car	931	658	938	671
20	image0.pr	1920	1280	car	638	680	664	695

Screenshot during training:

```
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=0.29s).
Accumulating evaluation results...
DONE (t=0.04s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.214
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.366
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.279
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.253
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.629
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.098
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.205
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.236
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.012
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.306
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.664
INFO:tensorflow:Finished evaluation at 2019-12-09-13:17:36
I1209 13:17:36.386079 140424354654016 evaluation.py:275] Finished evaluation at 2019-12-09-13:17:36
```

Screenshot when training finished:

```
W1209 15:54:53.995266 140424354654016 ag_logging.py:145] Entity <bound method BatchNormalization.call of <tensorflow.python.layers.normalization.BatchNormalization object at 0x7fb5f04a7780>> could not be transformed and will be executed as-is. Please report this to the AutgoGraph team. When filing the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause: converting <bound method BatchNormalization.call of <tensorflow.python.layers.normalization.BatchNormalization object at 0x7fb5f04a7780>>: AssertionError: Bad argument number for Name: 3, expecting 4
WARNING:tensorflow:From /home/010151297/models/research/object_detection/model_lib.py:418: The name tf.saved_model.signature_constants.PREDICT_METHOD_NAME is deprecated. Please use tf.saved_model.PREDICT_METHOD_NAME instead.

W1209 15:54:54.027411 140424354654016 deprecation_wrapper.py:119] From /home/010151297/models/research/object_detection/model_lib.py:418: The name tf.saved_model.signature_constants.PREDICT_METHOD_NAME is deprecated. Please use tf.saved_model.PREDICT_METHOD_NAME instead.

INFO:tensorflow:Done calling model_fn.
I1209 15:54:54.886056 140424354654016 estimator.py:1147] Done calling model_fn.
WARNING:tensorflow:From /home/010151297/venv-3.6.6-gpu/lib/python3.6/site-packages/tensorflow/python/saved_model/signature_def_utils_impl.py:201: build_tensor_info (from tensorflow.python.saved_model.utils_impl) is deprecated and will be removed in a future version.
Instructions for updating:
This function will only be available through the v1 compatibility library as tf.compat.v1.saved_model.utils.build_tensor_info or tf.compat.v1.saved_model.build_tensor_info.
W1209 15:54:54.886338 140424354654016 deprecation.py:323] From /home/010151297/venv-3.6.6-gpu/lib/python3.6/site-packages/tensorflow/python/saved_model/signature_def_utils_impl.py:201: build_tensor_info (from tensorflow.python.saved_model.utils_impl) is deprecated and will be removed in a future version.
Instructions for updating:
This function will only be available through the v1 compatibility library as tf.compat.v1.saved_model.utils.build_tensor_info or tf.compat.v1.saved_model.build_tensor_info.
INFO:tensorflow:Signatures INCLUDED in export for Classify: None
I1209 15:54:54.887215 140424354654016 export_utils.py:170] Signatures INCLUDED in export for Classify: None
INFO:tensorflow:Signatures INCLUDED in export for Regress: None
I1209 15:54:54.887333 140424354654016 export_utils.py:170] Signatures INCLUDED in export for Regress: None
INFO:tensorflow:Signatures INCLUDED in export for Predict: ['tensorflow/serving/predict', 'serving_default']
I1209 15:54:54.887422 140424354654016 export_utils.py:170] Signatures INCLUDED in export for Predict: ['tensorflow/serving/predict', 'serving_default']
INFO:tensorflow:Signatures INCLUDED in export for Train: None
I1209 15:54:54.887502 140424354654016 export_utils.py:170] Signatures INCLUDED in export for Train: None
INFO:tensorflow:Signatures INCLUDED in export for Eval: None
I1209 15:54:54.887579 140424354654016 export_utils.py:170] Signatures INCLUDED in export for Eval: None
INFO:tensorflow:Restoring parameters from /data/cmpe297-02-fa2019/michaelhsieh/modeloutput4/model.ckpt-3000
I1209 15:54:54.891312 140424354654016 saver.py:1280] Restoring parameters from /data/cmpe297-02-fa2019/michaelhsieh/modeloutput4/model.ckpt-3000
INFO:tensorflow:Assets added to graph.
I1209 15:54:55.878391 140424354654016 builder_impl.py:661] Assets added to graph.
INFO:tensorflow:No assets to write.
I1209 15:54:55.878725 140424354654016 builder_impl.py:456] No assets to write.
INFO:tensorflow:SavedModel written to: /data/cmpe297-02-fa2019/michaelhsieh/modeloutput4/export/Servo/temp-b'1575935676'/saved_model.pb
I1209 15:54:57.918945 140424354654016 builder_impl.py:421] SavedModel written to: /data/cmpe297-02-fa2019/michaelhsieh/modeloutput4/export/Servo/temp-b'1575935676'/saved_model.pb
INFO:tensorflow:Loss for final step: 0.15547611.
I1209 15:54:58.472426 140424354654016 estimator.py:368] Loss for final step: 0.15547611.
```

Model output:

```
(venv-3.6.6-gpu) [010151297@coe-hpc1 modeloutput4]$ ls
checkpoint                                        model.ckpt-2333.index              model.ckpt-2705.meta
eval_0                                            model.ckpt-2333.meta               model.ckpt-2889.data-00000-of-00001
events.out.tfevents.1575925604.coe-hpc1.sjsuad.sjsu.edu  model.ckpt-2521.data-00000-of-00001  model.ckpt-2889.index
export                                            model.ckpt-2521.index              model.ckpt-2889.meta
export-inference-model                            model.ckpt-2521.meta               model.ckpt-3000.data-00000-of-00001
graph.pbtxt                                       model.ckpt-2705.data-00000-of-00001  model.ckpt-3000.index
model.ckpt-2333.data-00000-of-00001               model.ckpt-2705.index              model.ckpt-3000.meta
```

12. There was an attempt to export the trained model, and compare the results of object detection using the trained model.