TATA CONSULTANCY SERVICES
Experience certainty.

TATA

Machine Learning for Agriculture

Dr. Gopi Kandaswamy
TCS Research and Innovation

Contents

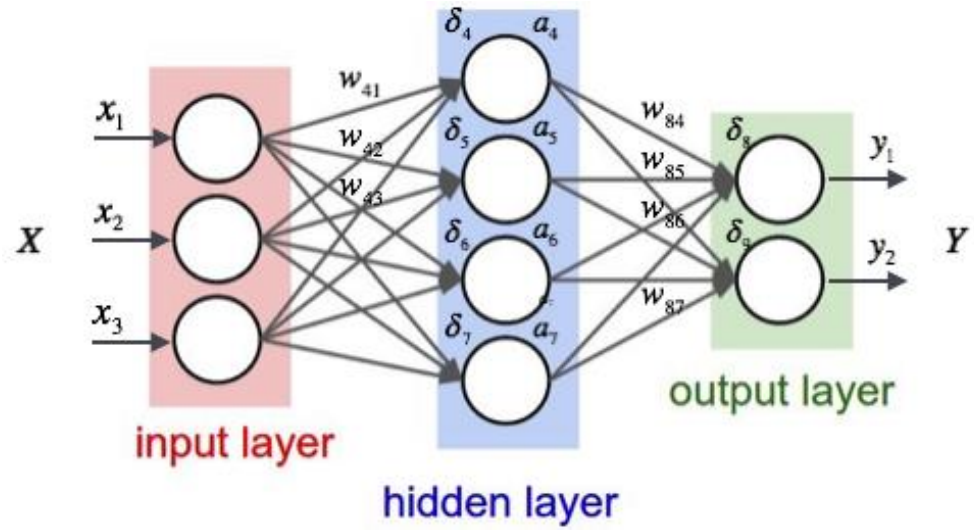
- Introduction to machine learning and deep learning
- Tensorflow as a deep learning tool
  - Installing Tensorflow
  - Annotating images
  - Downloading and configuring DL models
  - Training a DL model
  - Monitoring the training
  - Using the trained DL model for classification

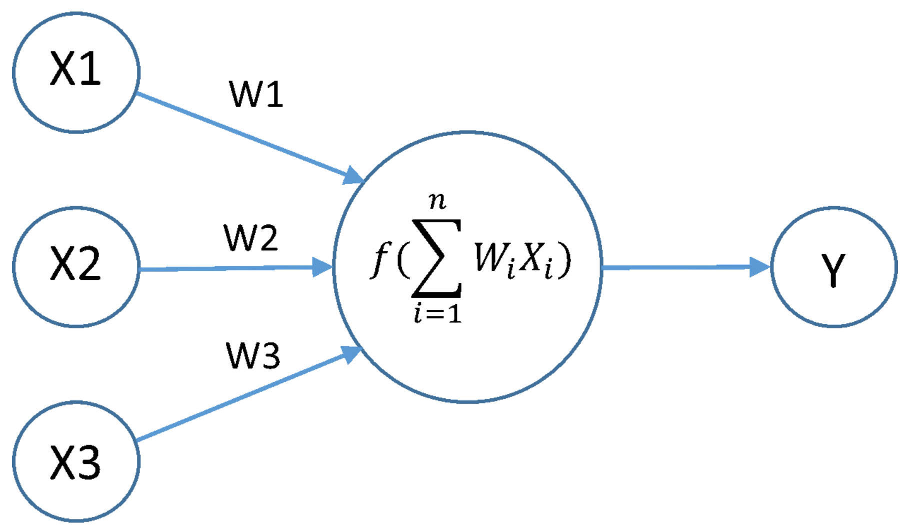
Introduction to Machine Learning

- An application of AI that provides computers with the ability to learn and improve without being explicitly programmed
- Learning begins with data as examples, experiences & rules to look for patterns to make better decisions
- Machine learning algorithms are categorized into
  - Supervised machine learning
  - Unsupervised machine learning
  - Semi-supervises machine learning
  - Reinforcement machine learning
- Can be used to analyze massive quantities of data
- Generally faster and more accurate results
- Requires additional time and resources for proper training

Artificial Neural Networks

- Interconnected group of artificial neurons using a computational model for information processing
- Adaptive system that changes structure based on information flow through it
- It is a non-linear statistical data modelling or decision making tool
- Used to model complex relationships between inputs & outputs & find patterns in data

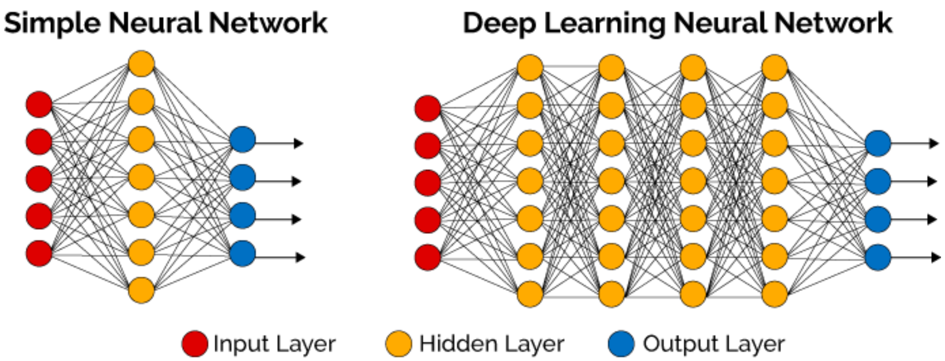




* Images downloaded from multiple websites

Deep Learning

- Dates back to 1965 but became a revolution in 2012
- Uses multiple layers of non-linear processing units for feature extraction and transformation
- Currently used for many applications including speech recognition, image recognition, NLP, drug discovery, bio-informatics, mobile advertising etc.



* Images downloaded from multiple websites

TensorFlow

- TensorFlow is an open source software library for data flow programming used for a wide variety of tasks
- Easily deployed across a variety of platforms including CPUs, GPUs and TPUs
- Originally developed by the Google Brain team
- Used widely by many organizations for a variety of tasks
- Website: https://www.tensorflow.org/

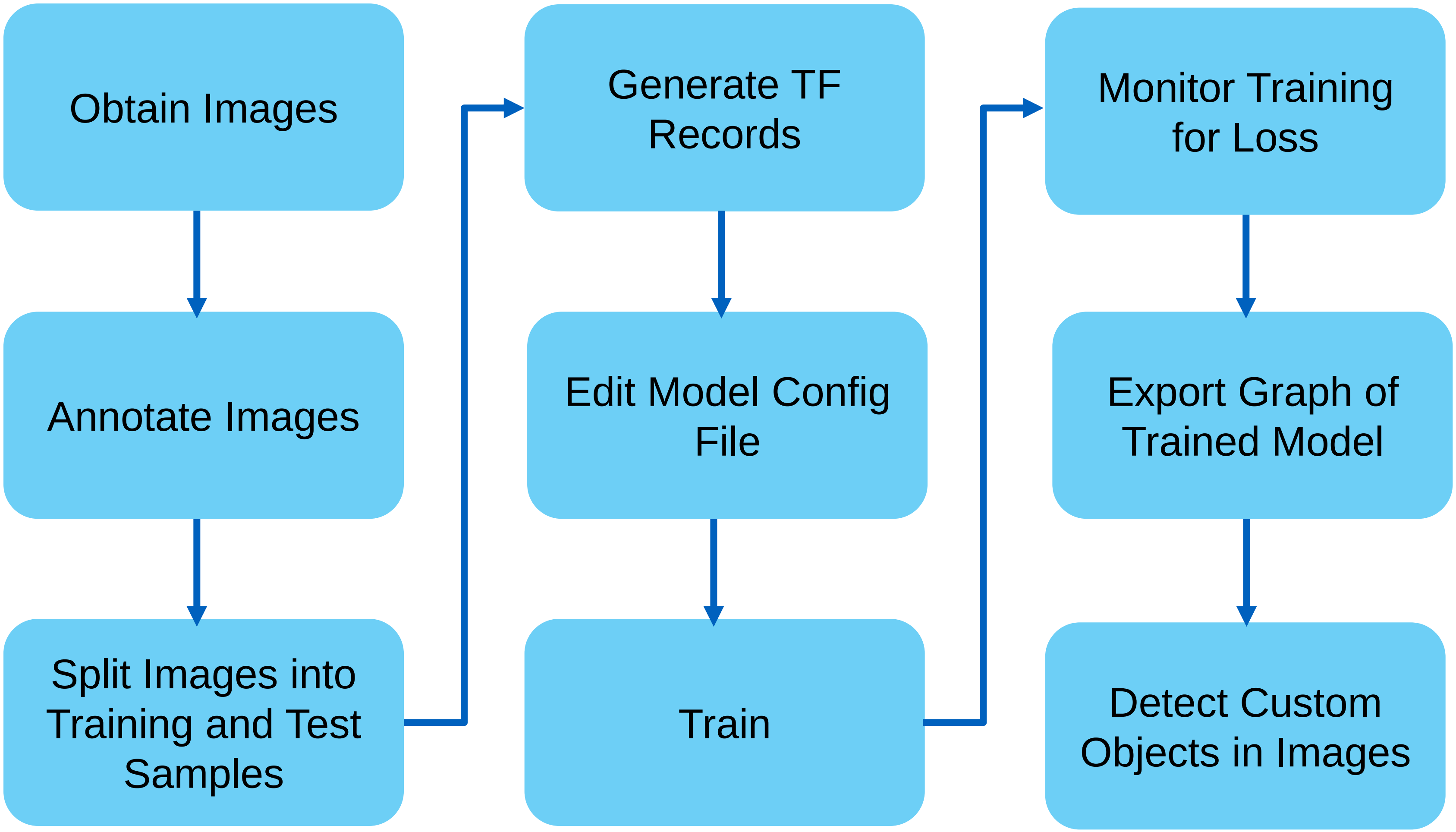
Installing TensorFlow

- Open terminal and follow these steps to install Tensorflow on Linux
- *pip3 install tensorflow   # Python 3.n; CPU support (no GPU support)*
- *pip3 install tensorflow-gpu # Python 3.n; GPU support*
- Validating installation – Invoke python from terminal using *python* command
  - Test with small python program
  - *# Python*
  - *import tensorflow as tf*
  - *hello = tf.constant('Hello, TensorFlow!')*
  - *sess = tf.Session()*
  - *print(sess.run(hello))*
  - Output in terminal should be : ***Hello, TensorFlow!***

source: https://www.tensorflow.org/install/install_linux#installing_with_native_pip

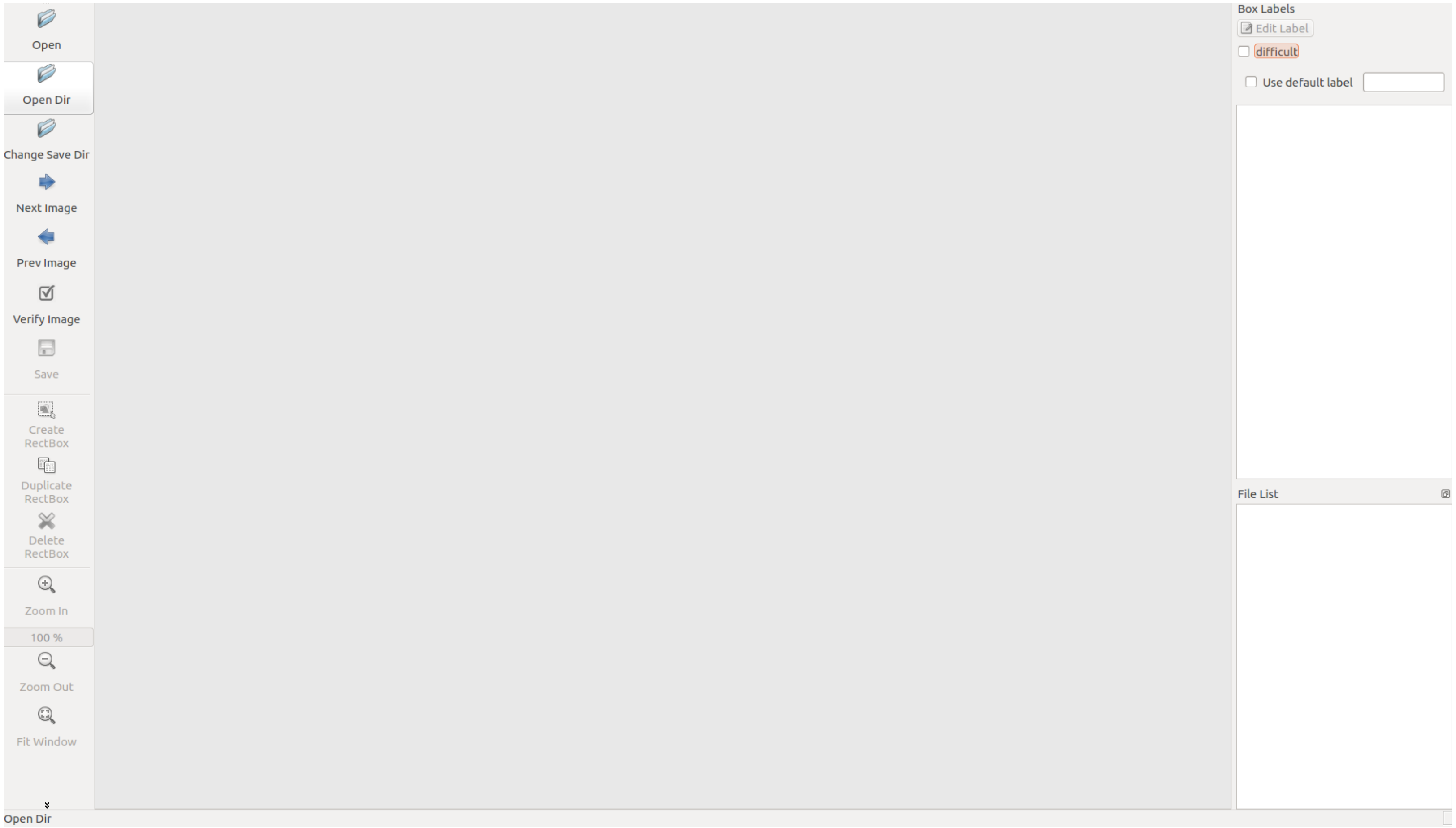
- Download the tensorflow models from github
  - *https://github.com/tensorflow/models*
- Check for python dependencies *pillow, lxml, matplotlib*
- Run following commands from models directory
  - protoc object_detection/protos/*.proto –python_out=.
  - export PYTHONPATH=$PYTHONPATH:`pwd`:`pwd`/slim
- Now we are all set to use different tensorflow models present in research folder

Classifying Objects using TensorFlow

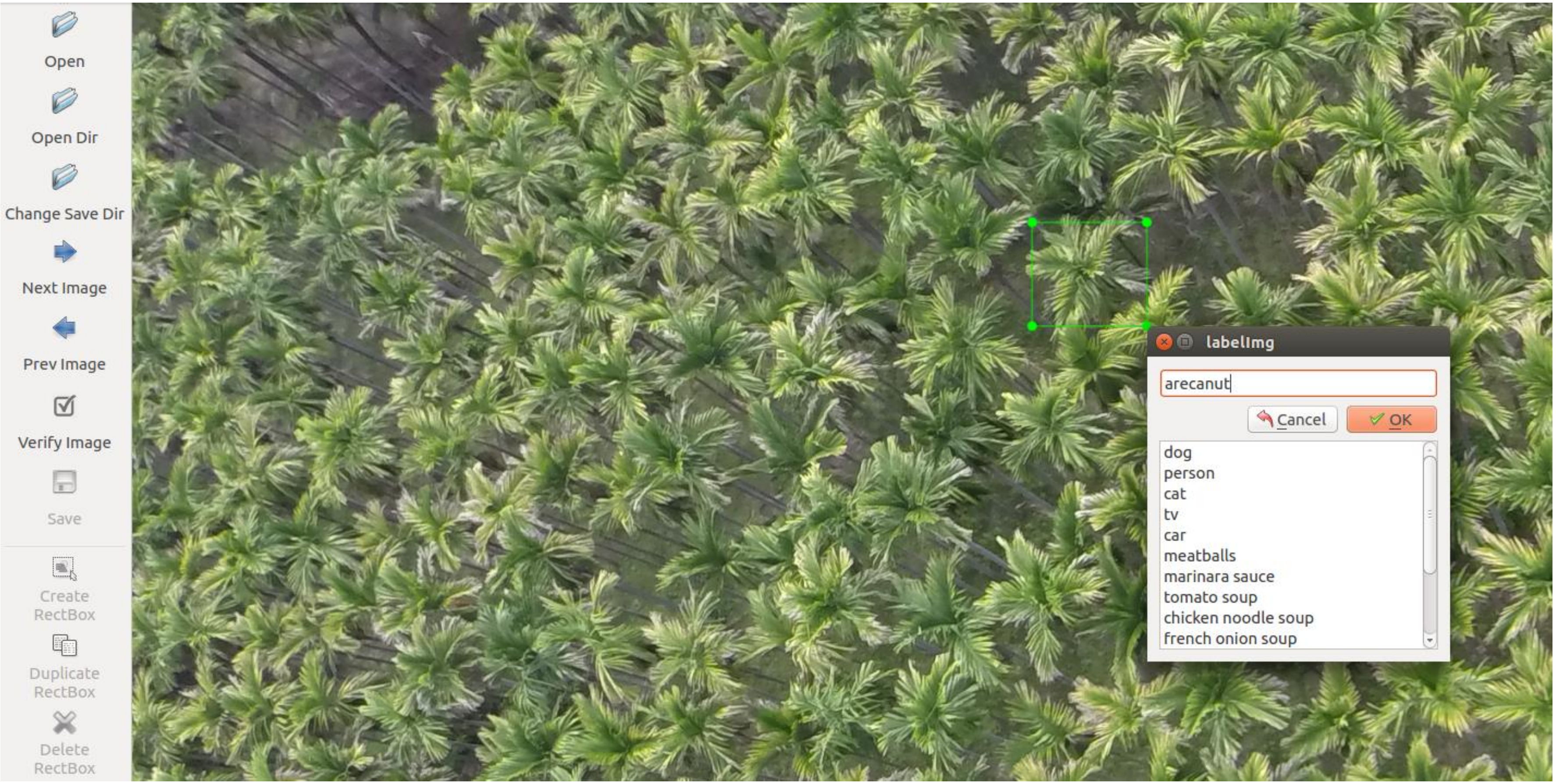


Annotating Images

- Use a tool to annotate images; we use LabelImg
  - python3 labelImg.py



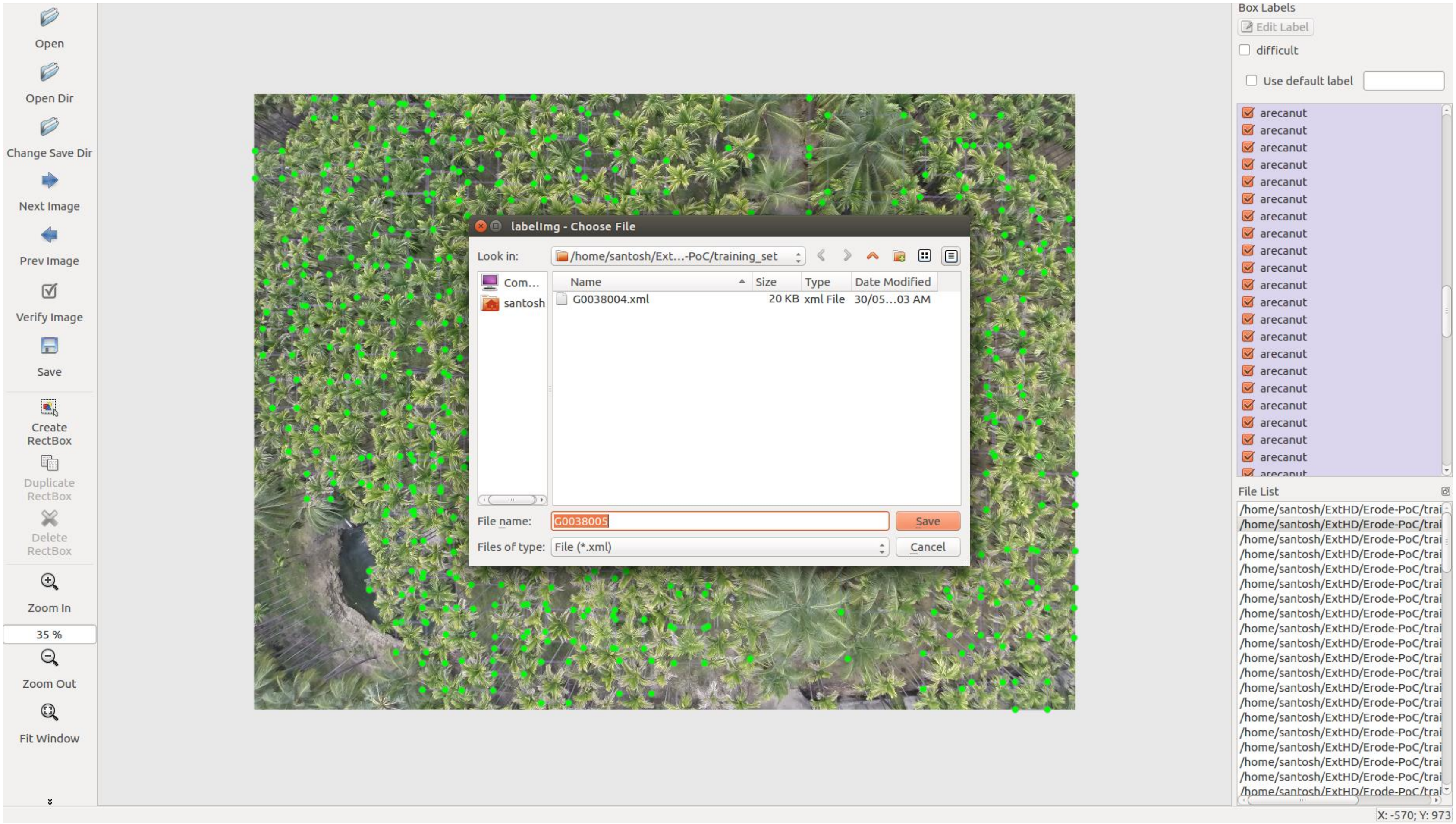
Labelling Arecanut Trees



Labelling Arecanut Trees

Labelling Coconut Trees

Saving Annotated Images



Spliting Images into Train and Test Samples

- Need to have two sets of images; train and test
- Metadata for annotated images will be saved as xml file
- Edit ***xml_to_csv.py*** to convert xml files to a single csv file

```
def main():
    image_path = os.path.join(os.getcwd(), 'ac_test')
    xml_df = xml_to_csv(image_path)
    xml_df.to_csv('ac_test.csv', index=None)
    print('Successfully converted xml to csv.')
```

- Generate CSV files for train and test samples
  - *python xml_to_csv.py*

Generating TF Record

- TFRecord is one of the data types used in tensorflow
- Makes it easy to deal with images in datasets
- Edit *generate_tfrecord.py*

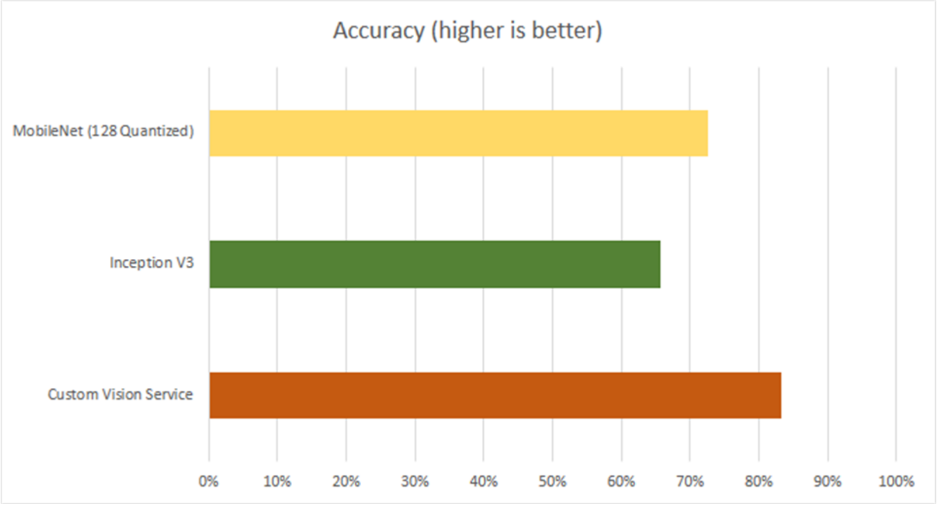
```
# TO-DO replace this with label map
def class_text_to_int(row_label):
    if row_label == 'arecanut':
        return 1
    if row_label == 'coconut':
        return 1
    else:
        None
```

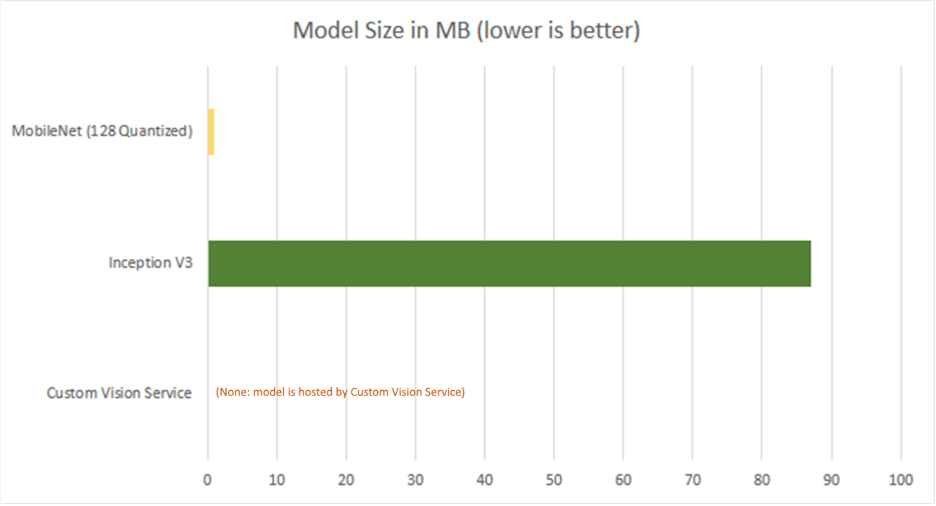
- *python generate_tfrecord.py --csv_input=ac_train --output_path=ac_train.record*
- *python generate_tfrecord.py --csv_input=ac_test --output_path=ac_test.record*

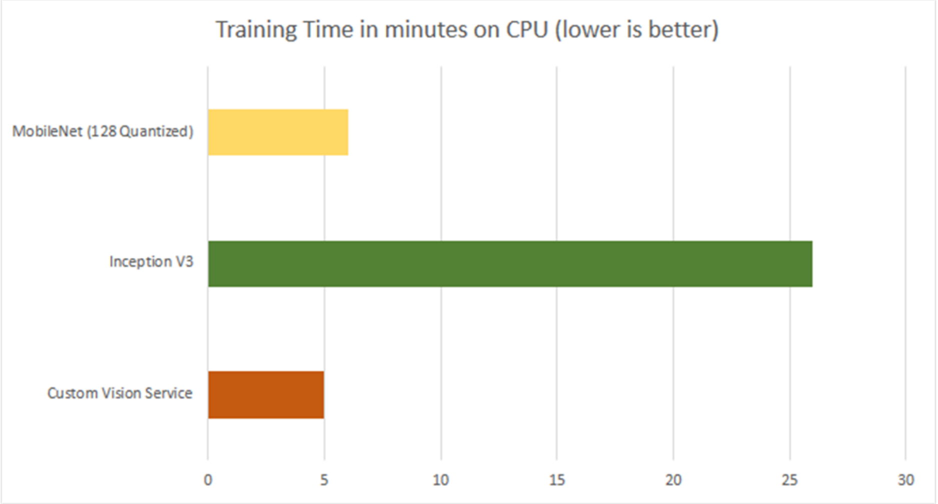
Configuring a DL Model

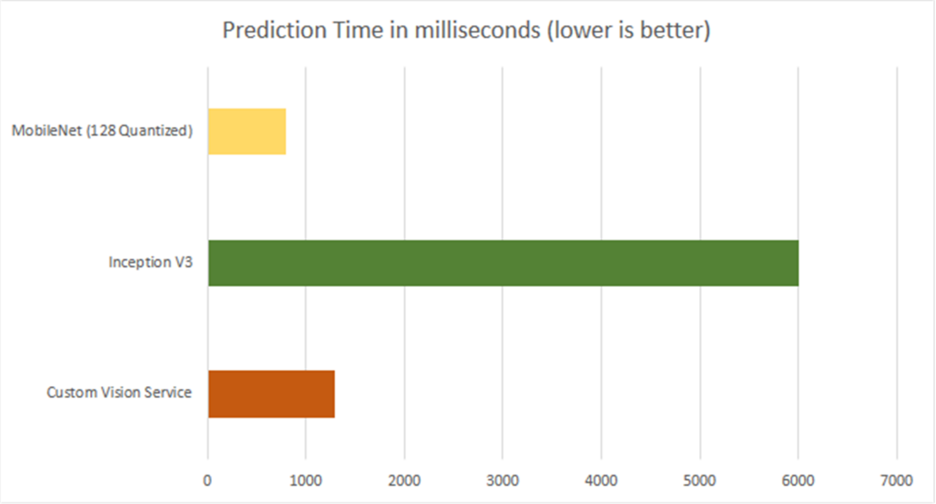
- Download tensorflow mobilenet model from
  - *http://download.tensorflow.org/models/object_detection/ssd_mobilenet_v1_coco_11_06_2017.tar.gz*
- Download mobilenet configuration file from
  - *https://raw.githubusercontent.com/tensorflow/models/master/object_detection/samples/configs/ssd_mobilenet_v1_pets.config*
- Create folder *models/research/data* to store tfrecord files

Choosing a DL Model





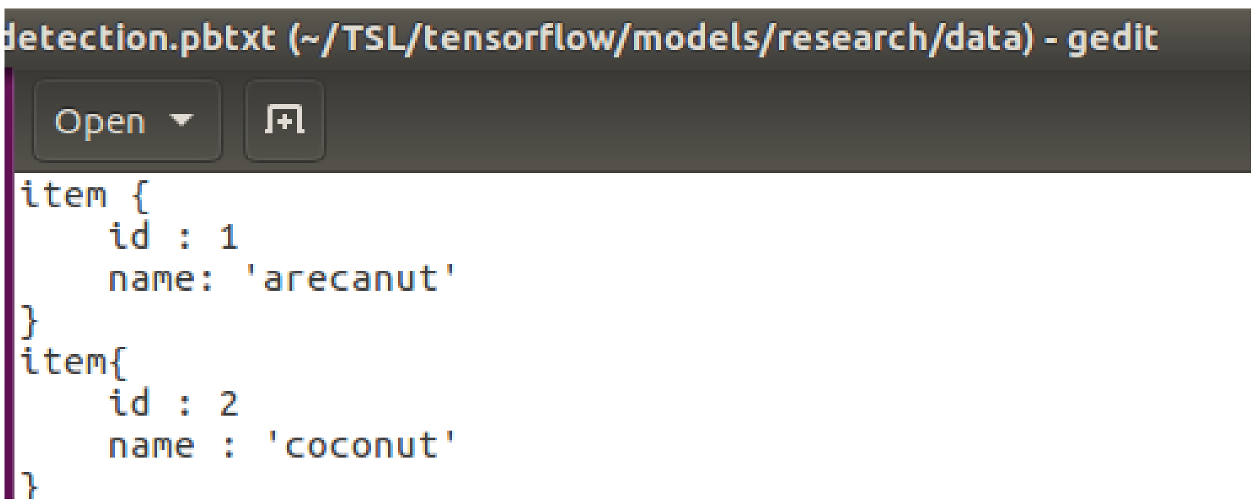




* Images downloaded from multiple websites

Creating Label Map File

- Create a label map file ***object_detection.pbtxt*** inside models/research/data



Editing the Model Configuration File

- Change ***num_classes*** to number of custom objects to be trained; in our case it is 2 (arecanut and coconut)

```
model {
  ssd {
    num_classes: 2
    box_coder {
      faster_rcnn_box_coder {
        y_scale: 10.0
        x_scale: 10.0
        height_scale: 5.0
        width_scale: 5.0
      }
    }
    matcher {
```

Editing the Model Configuration File

- Change ***fine_tune_checkpoint*** to mobilenet’s checkpoint
  - We are retraining existing mobilenet model to recoganize arecanut and coconut images
  - ***fine_tune_checkpoint*** refers to the previous graph that needs to be retrained

```
    }
    momentum_optimizer_value: 0.9
    decay: 0.9
    epsilon: 1.0
  }
}
fine_tune_checkpoint: "ssd_mobilenet_v1_coco_2017_11_17/model.ckpt"
from_detection_checkpoint: true
data_augmentation_options {
  random_horizontal_flip {
  }
```

Editing the Model Configuration File

- Change the ***tf_record_input_reader*** input path to tfrecord location and label_map_path to labelmap location in both ***train_input_reader*** and ***eval_input_reader***.

```
train_input_reader: {
  tf_record_input_reader {
    input_path: "data/ac_train.record"
  }
  label_map_path: "data/object-detection.pbtxt"
}
```

```
eval_input_reader: {
  tf_record_input_reader {
    input_path: "data/ac_test.record"
  }
  label_map_path: "data/object-detection.pbtxt"
  shuffle: false
  num_readers: 1
}
```

- Create dir models/research/training & store this *model.config* file there

Training the Model

- Start the training using the command

python3 train.py --logtostderr –train_dir=training --pipeline_config_path=training/model.config

```
(tensorflow) santosh@Rhino:~/TSL/tensorflow/models/research$ python train.py --l
ogtostderr --train_dir=training/ --pipeline_config_path=training/model.config
WARNING:tensorflow:From /home/santosh/TSL/tensorflow/models/research/trainer.py:
228: 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.create_global_step
INFO:tensorflow:depth of additional conv before box predictor: 0
INFO:tensorflow:depth of additional conv before box predictor: 0
INFO:tensorflow:depth of additional conv before box predictor: 0
INFO:tensorflow:depth of additional conv before box predictor: 0
INFO:tensorflow:depth of additional conv before box predictor: 0
INFO:tensorflow:depth of additional conv before box predictor: 0
```

Monitoring the Training

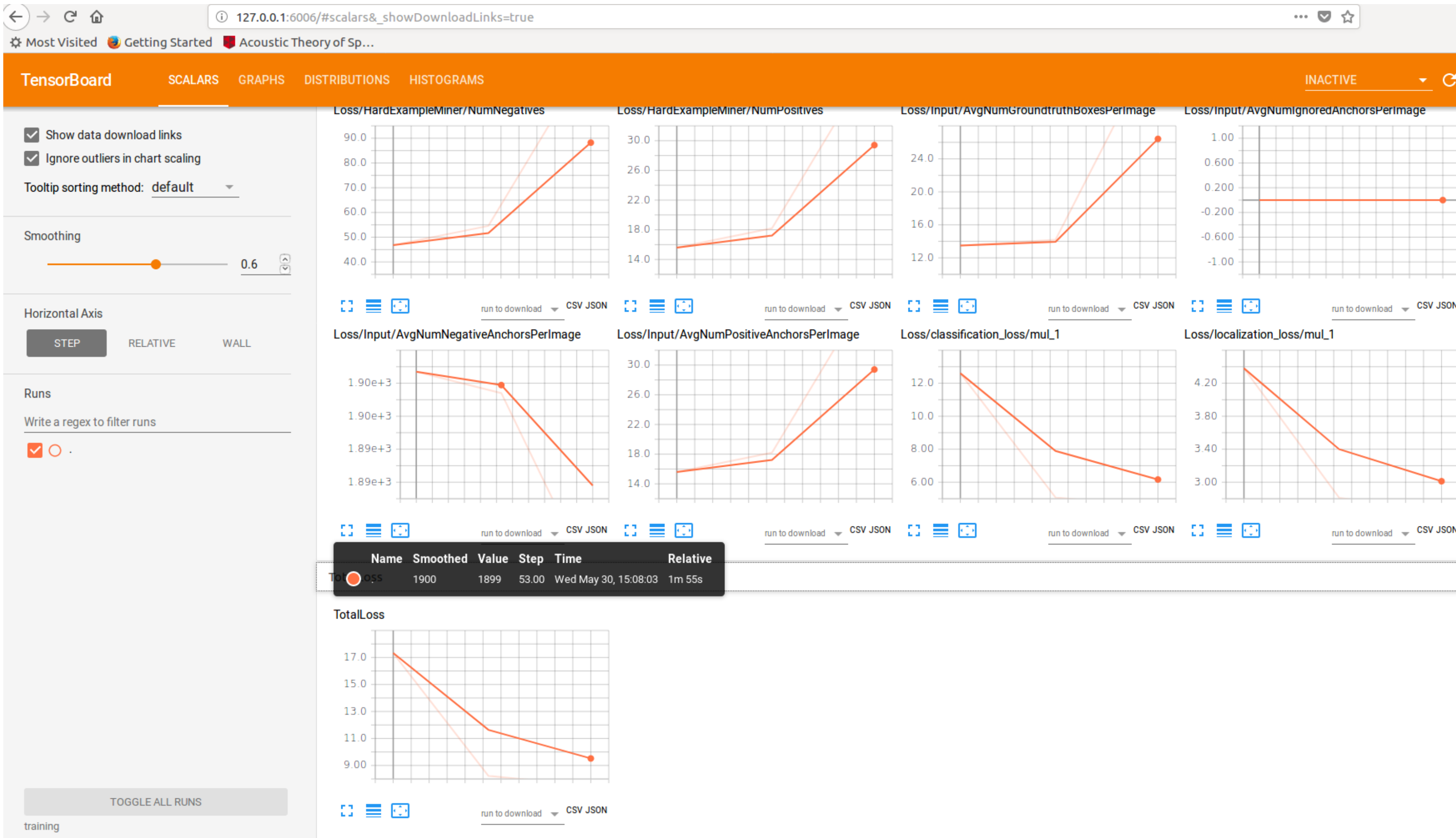
- Launch Tensorboard
  - *tensorboard –logdir='training'*

```
(tensorflow) santosh@Rhino:~/TSL/tensorflow/models/research$ tensorboard --logdi
r='training'
TensorBoard 1.5.1 at http://Rhino:6006 (Press CTRL+C to quit)
W0530 15:10:25.149281 Thread-1 application.py:273] path /paper-ripple/paper-ripp
le.html not found, sending 404
W0530 15:10:25.150618 Thread-1 application.py:273] path /paper-behaviors/paper-b
utton-behavior.html not found, sending 404
W0530 15:10:25.162271 Thread-2 application.py:273] path /paper-behaviors/paper-i
nky-focus-behavior.html not found, sending 404
```

- Launched Tensorboard GUI available at above address

Monitoring the Training

- TensorBoard showing percentage loss

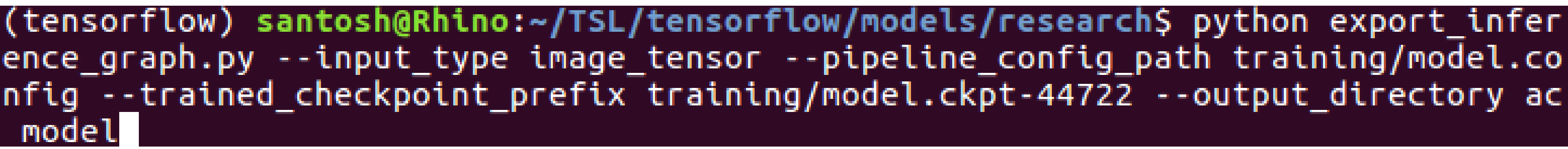


Monitoring the Training

Exporting Trained Model to Graph

- Saved checkpoint can be exported to graph using

python export_inference_graph.py --input_type image_tensor --pipeline_config_path training/model.config --trained_checkpoint_prefix training/model.ckpt-35439 --output_directory ac_model



Running the Model for Classification

- Place the images to be classified in say *models/research/test_images*

```
# What model to download.
MODEL_NAME = 'ac_model_3'
MODEL_FILE = MODEL_NAME + '.tar.gz'
DOWNLOAD_BASE = 'http://download.tensorflow.org/models/object_detection/'

# Path to frozen detection graph. This is the actual model that is used fo
PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'

# List of the strings that is used to add correct label for each box.
PATH_TO_LABELS = os.path.join('data', 'object-detection.pbtxt')

NUM_CLASSES = 2
```

```
# image2.jpg
# If you want to test the code with your images, just add path to the images to the TEST_IMAGE_PATHS.
PATH_TO_TEST_IMAGES_DIR = 'test_images'
TEST_IMAGE_PATHS = [ os.path.join(PATH_TO_TEST_IMAGES_DIR, 'image{}.jpg'.format(i)) for i in range(1, 6) ]
```

- Run custom object detector using command ***python object_detection.py***

Testing DL Model with Images