OBJECT DETECTION USING TENSORFLOW
PROJECT DRAFT

SUBMITTED BY

Menita Koonani, 001883043
Renu Hadke, 001873839

ADVANCES IN DATA SCIENCES AND ARCHITECTURE
INFO 7390

NORTHEASTERN UNIVERSITY
BOSTON MA 02115

ABSTRACT:

Object detection is a computer technology connected to computer vision and image processing that deals with detecting instances of objects of a certain class in digital images and videos. It has applications in many areas of computer vision such as face detection, people counting, vehicle detection, anomaly detection and for classifying images online. Our objective is to identify and classify objects spotted on images and real-time video using TensorFlow and to determine the accuracy of each identification. We are considering two Models namely, SSD with MobileNet, SSD Inception V2 model and Faster RCNN Inception model to compare the accuracy and size of the models.

INTRODUCTION:

We are using a COCO Dataset that has been pre-trained using Convolution Neural Network to detect various objects in a common set of 90 classes.

Dataset Specifications:

COCO Dataset is a large-scale object detection, segmentation and captioning dataset. It is downloaded from https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md and it has 200K labelled images categorized into 90 classes.

Convolutional Neural Networks (CNN):

It is a class of deep, feed-forward artificial neural networks that has been applied to analyzing visual imagery.[1] These are a special class of Multilayer perceptron which are well suited for pattern classification. It is specifically designed to recognize 2D shapes with a high level of invariance, skewing and scaling. They are made up of neurons that have learnable weights and biases. Each neuron receives some input, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. A simple ConvNet is a sequence of layers, and every layer of a ConvNet transforms one volume of activations to another through a differentiable function.

[1] "Convolutional neural network - Wikipedia." https://en.wikipedia.org/wiki/Convolutional_neural_network. Accessed 21 Apr. 2018.

There are three main types of layers to build ConvNet architectures: Convolutional Layer, Pooling Layer, and Fully-Connected Layer

Layers in CNN: [2]

- INPUT layer will hold the raw pixel values of the image, of width 32, height 32, and with three color channels Red, Green, Blue.
- CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume.
- RELU layer will apply an elementwise activation function, such as the max (0, x) thresholding at zero.
- POOL layer will perform a down sampling operation along the spatial dimensions (width, height)
- FC (i.e. fully-connected) layer will compute the class scores

Various models using ConvNets:

SSD with MobileNet:

Out of the many detection models, we chose to work with the combination of Single Shot Detectors(SSDs) and MobileNets architecture as they are fast, efficient and do not require huge computational capability to fulfill the Object Detection task. The SSD approach is based on a feed-forward convolutional neural network which produces a fixed-size collection of bounding boxes and scores for the object class instances present in those boxes. The main difference between a “traditional” CNN’s and the MobileNet architecture is instead of a single 3x3 convolution layer followed by batch norm and ReLU, MobileNets split the convolution into a 3x3 depthwise conv and a 1x1 pointwise conv.

SSD Inception V2 Model:

Given an input image and a set of ground truth labels, SSD does the following:

- It passes the image through a series of convolutional layers, providing several sets of feature maps at different scales.

[2] "Convolutional Neural Networks - CS231n Convolutional Neural ...." http://cs231n.github.io/convolutional-networks/. Accessed 21 Apr. 2018.

- For each location in each of these feature maps, a 3x3 convolutional filter is used to evaluate a small set of default bounding boxes.
- For each box, it simultaneously predicts the bounding box offset and the probabilities of each class.
- During training, it matches the ground truth box with these predicted boxes based on IoU(Intersection over Union). The best predicted box is labeled a “positive” along with all the other boxes having an IoU with the truth greater than 0.5.

Faster RCNN Inception Model:

The main insight of Faster R-CNN was to replace the slow selective search algorithm that was used in the R-CNN (Region-based Convolutional Neural Network), with a fast neural net.[3]

The R-CNN basically works in three simple steps:

1. Scans the input image for possible objects using a Selective Search algorithm thus generating ~2000 region proposals.
2. Runs a convolutional neural net (CNN) on top of each of these region proposals
3. Takes the output of each CNN and feed it into (a) an SVM(Support Vector Machine) to classify the region and (b) a linear regressor to tighten the bounding box of the object, if such an object exists.

Faster R-CNN is similar to the original R-CNN but is improved on its detection speed through two augmentations:

- It performs feature extraction over the image even before proposing regions, thus running only one CNN over the entire image instead of running 2000 CNN’s across 2000 overlapping regions
- It replaces the SVM with a softmax layer, thus extending the network for predictions instead of creating a new model

[3] "Deep Learning for Object Detection: A Comprehensive Review." 11 Sep. 2017, https://towardsdatascience.com/deep-learning-for-object-detection-a-comprehensive-review-73930816d8d9. Accessed 21 Apr. 2018.

TensorFlow:

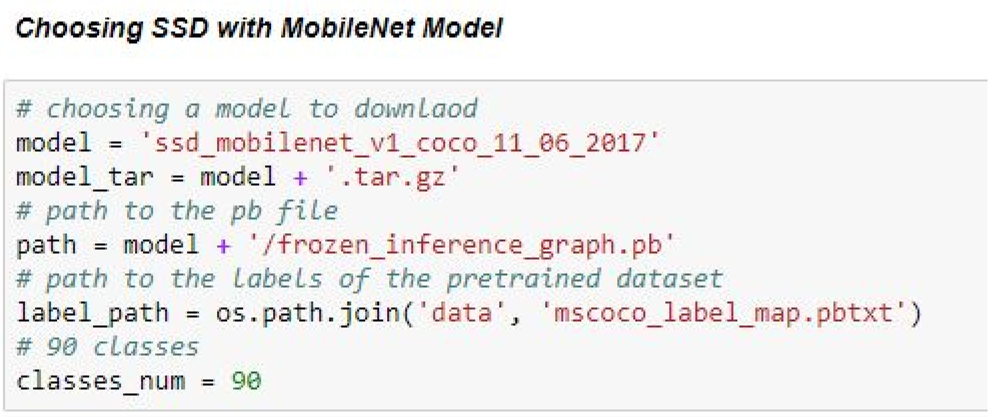
TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains. [4]

OpenCV:

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. The C++ API provides a class '*videocapture*' for capturing video from cameras or for reading video files and image sequences. It is basically used to access the Webcam of our computer to capture real-time videos. The frames from the videos can be read by creating an object of the VideoCapture class. This object which handles everything related to opening and closing of the webcam.

CODE WITH DOCUMENTATION:

The tar file for the model we are using is SSD with MobileNet that is downloaded from the official TensorFlow website. The labels and their respective weights are computed and stored in a Protobuf text file.

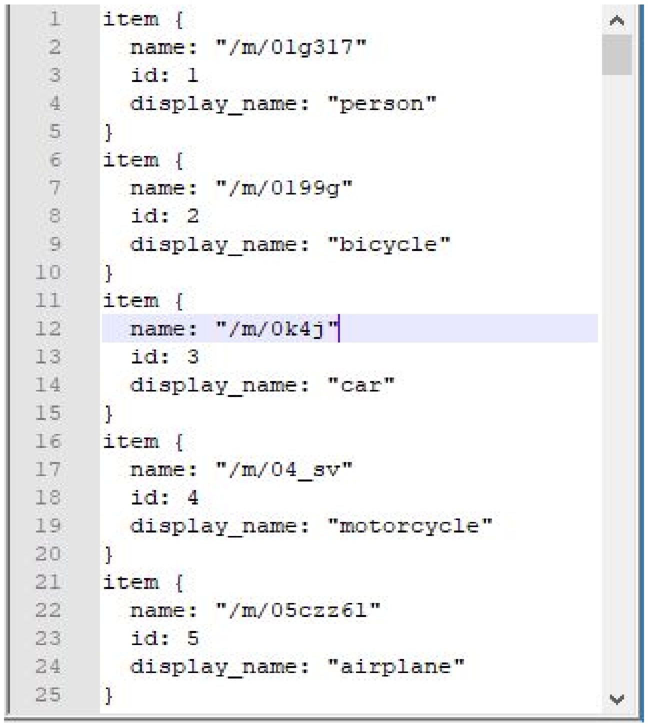


Choosing SSD with MobileNet Model

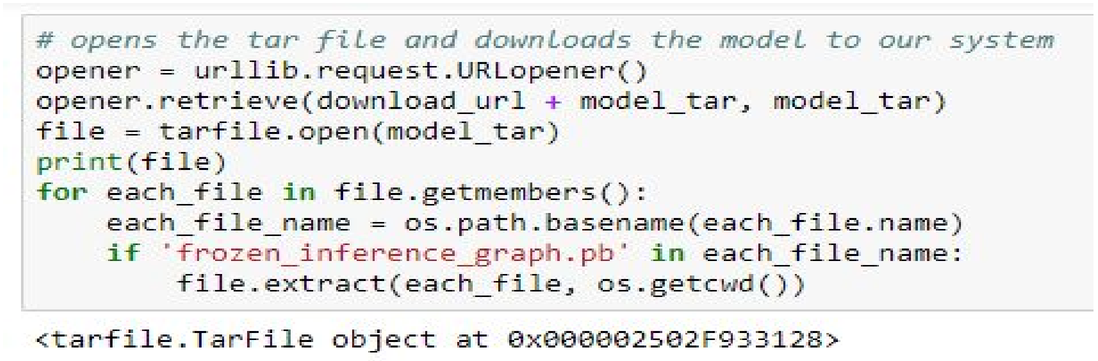
```
# choosing a model to downlaod
model = 'ssd_mobilenet_v1_coco_11_06_2017'
model_tar = model + '.tar.gz'
# path to the pb file
path = model + '/frozen_inference_graph.pb'
# path to the labels of the pretrained dataset
label_path = os.path.join('data', 'mscoco_label_map.pbtxt')
# 90 classes
classes_num = 90
```

[4] "TensorFlow.org." https://www.tensorflow.org/. Accessed 21 Apr. 2018.

A snapshot of the protobuf file (mscoco_label_map.pbtxt)



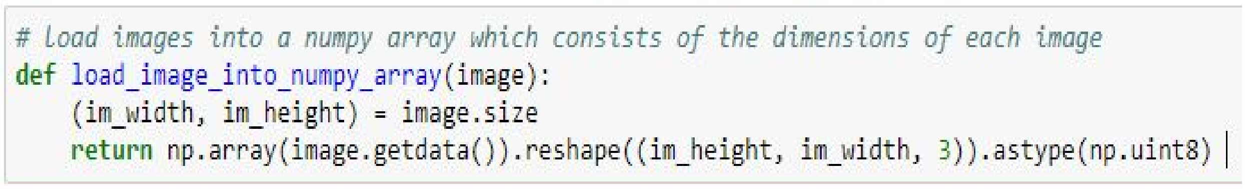
```
item {
  name: "/m/01g317"
  id: 1
  display_name: "person"
}
item {
  name: "/m/0199g"
  id: 2
  display_name: "bicycle"
}
item {
  name: "/m/0k4j"
  id: 3
  display_name: "car"
}
item {
  name: "/m/04_sv"
  id: 4
  display_name: "motorcycle"
}
item {
  name: "/m/05czz6l"
  id: 5
  display_name: "airplane"
}
```



```
# opens the tar file and downloads the model to our system
opener = urllib.request.URLopener()
opener.retrieve(download_url + model_tar, model_tar)
file = tarfile.open(model_tar)
print(file)
for each_file in file.getmembers():
    each_file_name = os.path.basename(each_file.name)
    if 'frozen_inference_graph.pb' in each_file_name:
        file.extract(each_file, os.getcwd())
```

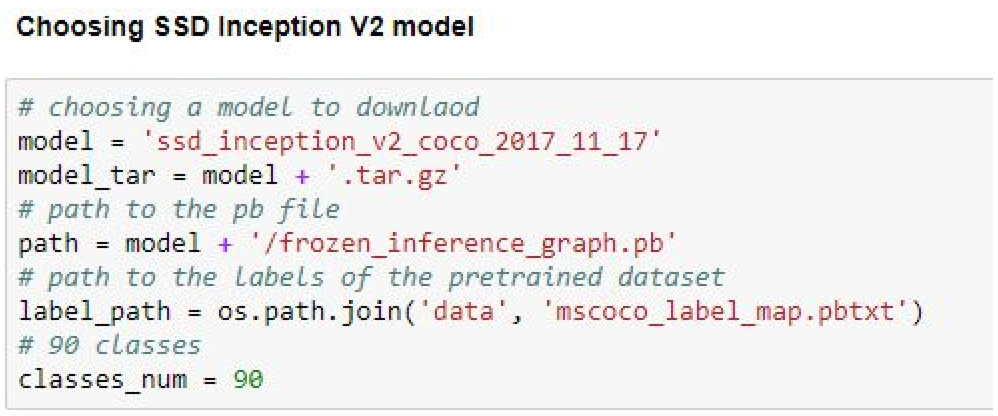
```
<tarfile.TarFile object at 0x000002502F933128>
```

We are loading the dimensions of each image into a numpy array. The dimensions of the image are the height, the width and the RGB intensity at various points in the image.



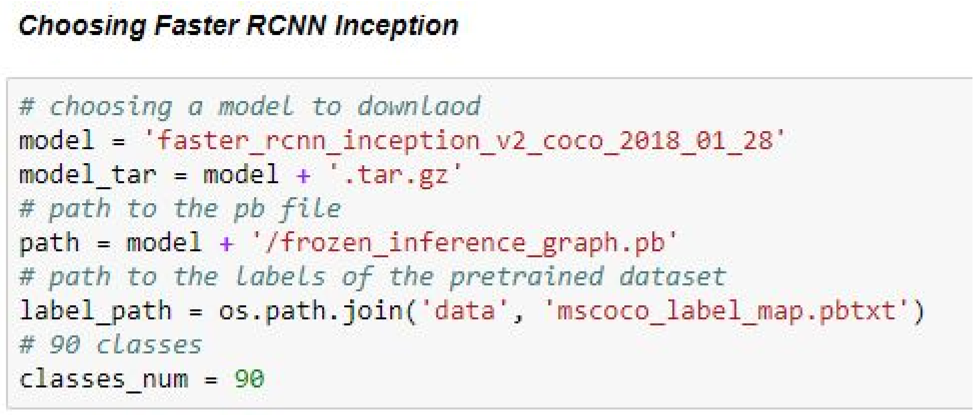
```
# Load images into a numpy array which consists of the dimensions of each image
def load_image_into_numpy_array(image):
    (im_width, im_height) = image.size
    return np.array(image.getdata()).reshape((im_height, im_width, 3)).astype(np.uint8)
```

Similarly the Object Detection API is implemented using the Faster RCNN Inception Model to compare it with the SSD MobileNet model, and SSD Inception V2 model.



Choosing SSD Inception V2 model

```
# choosing a model to downlaod
model = 'ssd_inception_v2_coco_2017_11_17'
model_tar = model + '.tar.gz'
# path to the pb file
path = model + '/frozen_inference_graph.pb'
# path to the labels of the pretrained dataset
label_path = os.path.join('data', 'mscoco_label_map.pbtxt')
# 90 classes
classes_num = 90
```

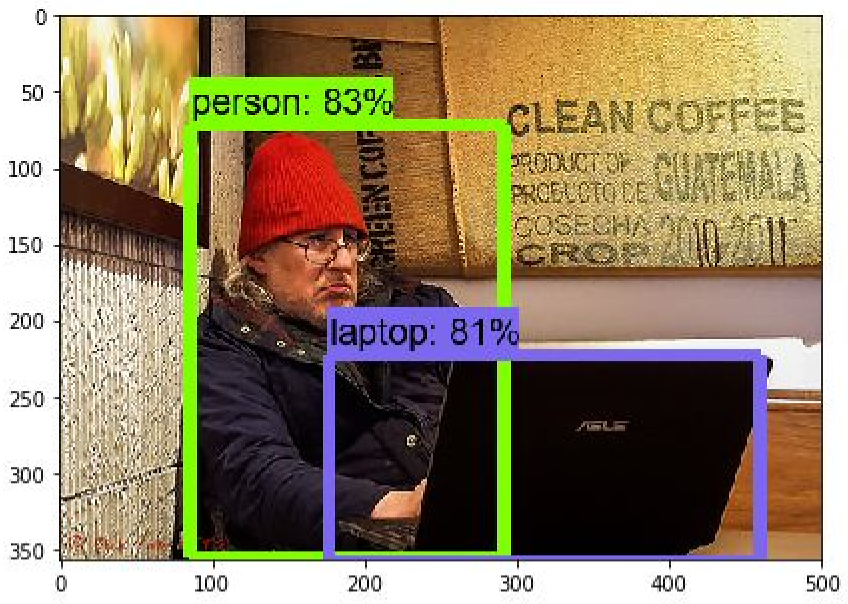


Choosing Faster RCNN Inception

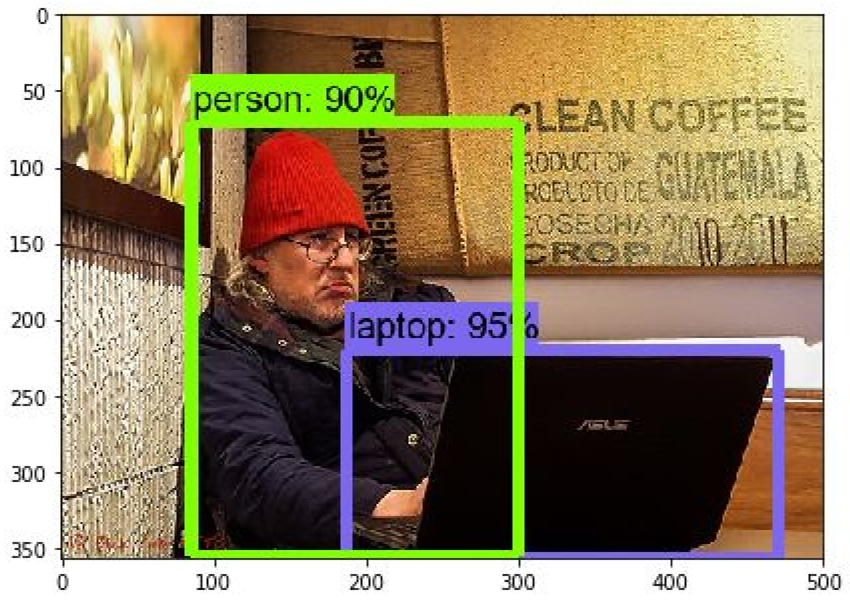
```
# choosing a model to downlaod
model = 'faster_rcnn_inception_v2_coco_2018_01_28'
model_tar = model + '.tar.gz'
# path to the pb file
path = model + '/frozen_inference_graph.pb'
# path to the labels of the pretrained dataset
label_path = os.path.join('data', 'mscoco_label_map.pbtxt')
# 90 classes
classes_num = 90
```

RESULTS:

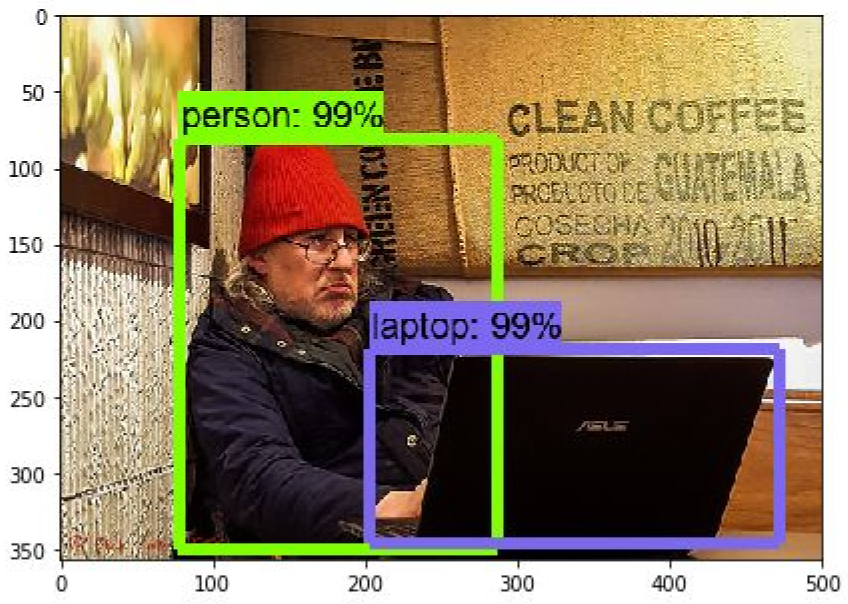
For SSD with MobileNet, the accuracy of object detection in image is 83% for person and 81% for laptop. This model worked fast though had the least accuracy.



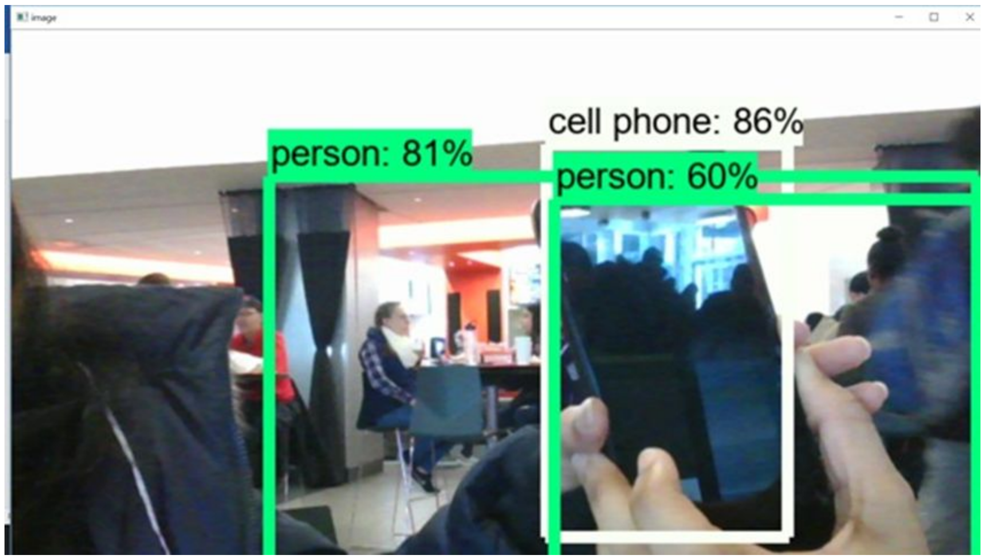
For SSD Inception V2 model, the accuracy of the objects detected in the images is 90% for the person and 95% for the laptop.



For Faster RCNN Inception Model, the accuracy of object detection in image is 99% for person and 99% for laptop



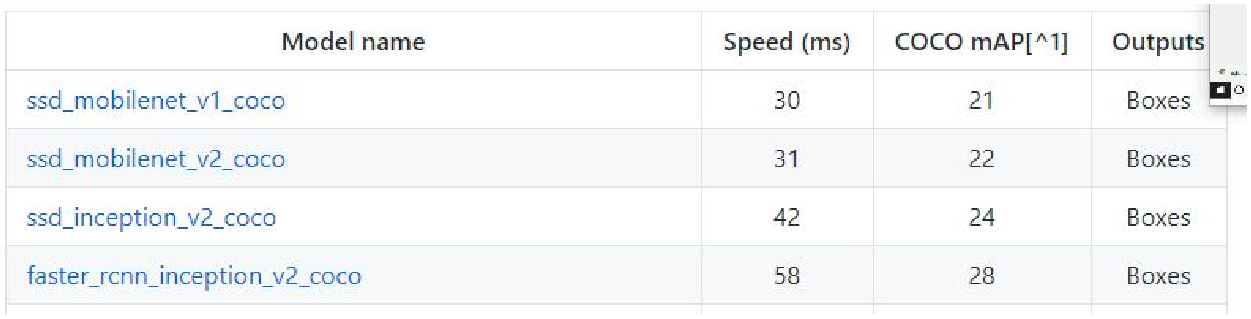
In addition, we were also successful in accessing the webcam of our system using OpenCV to detect real-time objects. The model used here is SSD with MobileNets as it produces much faster results as compared to the other two models.



DISCUSSION:

The principal difference between the two models is that Faster RCNN Inception V2 is optimized for accuracy, while the MobileNets are optimized to be small and efficient, at the cost of some accuracy. The SSD with MobileNets detects objects in only a single shot with just two components in its architecture namely, Feature Extraction and Detection Generator, while the Faster R-CNN consists of three components- Feature Extraction, Proposal Generation and Box Classifier.

In the Faster R-CNN Inception model, a region proposal network is used to generate regions of interest and then either fully-connected layers or position-sensitive convolutional layers to classify those regions. SSD does the two in a “single shot,” simultaneously predicting the bounding box and the class as it processes the image.



Model name	Speed (ms)	COCO mAP[^1]	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v2_coco	31	22	Boxes
ssd_inception_v2_coco	42	24	Boxes
faster_rcnn_inception_v2_coco	58	28	Boxes

The higher the mAp (minimum average precision), the better the model. Based on the observations, SSD with MobileNets provided much better results in terms of speed but the Faster RCNN Inception Model provided a higher accuracy with some compromise on the speed.

REFERENCES:

1. https://github.com/tensorflow/models/tree/master/research/object_detection
2. https://codelabs.developers.google.com/codelabs/tensorflow-for-poets/#0
3. https://hackernoon.com/creating-insanely-fast-image-classifiers-with-mobilenet-in-tensorflow-f030ce0a2991
4. https://pdfs.semanticscholar.org/3ac3/6574d1b1af029974cef6d6709feee2502194.pdf
5. https://towardsdatascience.com/deep-learning-for-object-detection-a-comprehensive-review-73930816d8d9