GALGOTIAS UNIVERSITY
(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Tensor flow Object Detection

A Report for the Evaluation 3 of Project 2

Submitted by

Krishna Kant Nath Tiwari
(1613101341 / 16SCSE101746)

in partial fulfillment for the award of the degree of

Bachelor of Technology

IN

Computer Science of Engineering with specialization in Computer Network & Cyber Security

School of Computing Science & Engineering

Under the Supervision of

Mr. S. Rakesh Kumar, Assistant Professor

May & 2020

Table of Contents

Abstract

Efficient and accurate object detection has been an important topic in theadvancement of computer vision systems. With the advent of deep learning techniques, the accuracy for object detection has increased drastically. The project aims to incorporate state-of-the-art technique for object detection with the goal of achieving high accuracy with a real-time performance. A major challenge in many of the object detection systems is the dependency on other computer vision techniques for helping the deep learning based approach, which leads to slow and non-optimal performance. In this project, we use a completely deep learning based approach to solve the problem of object detection in an end-to-end fashion. The network is trained on the most challenging publicly available dataset (PASCAL VOC), on which a object detection challenge is conducted annually. The resulting system is fast and accurate, thus aiding those applications which require object detection.

1 Introduction

1.1 Problem Statement

Many problems in computer vision were saturating on their accuracy before a decade. However, with the rise of deep learning techniques, the accuracy of these problems drastically improved. One of the major problem was that of image classification, which is defined as predicting the class of the image. A slightly complicated problem is that of image localization, where the image contains a single object and the system should predict the class of the location of the object in the image (a bounding box around the object). The more complicated problem (this project), of object detection involves both classification and localization. In this case, the input to the system will be a image, and the output will be a bounding box corresponding to all the objects in the image, along with the class of object in each box. An overview of all these problems is depicted in Fig. 1.

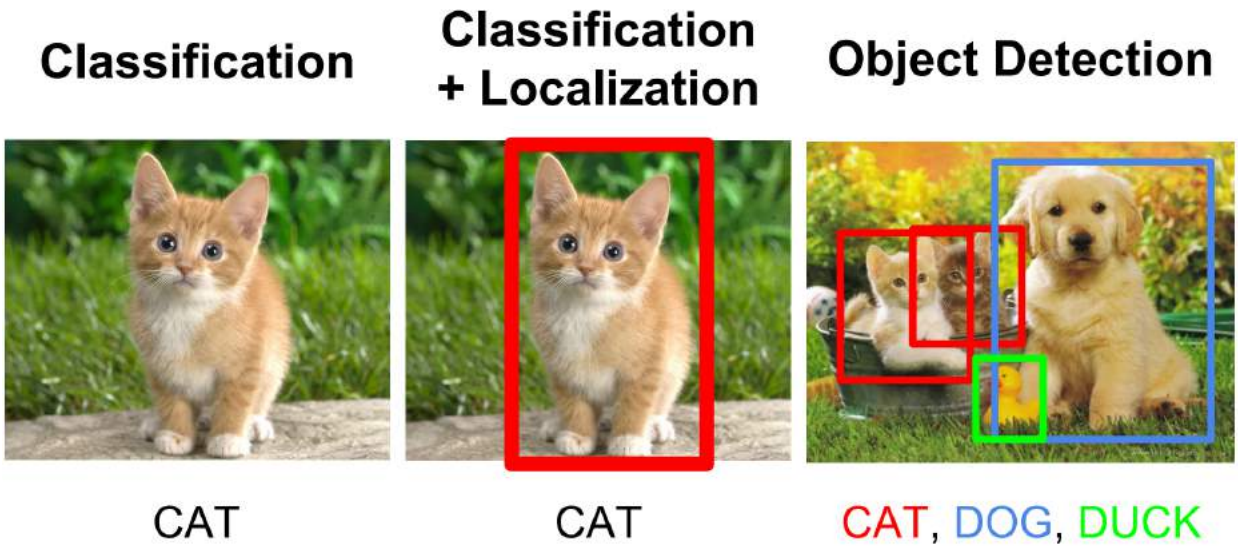
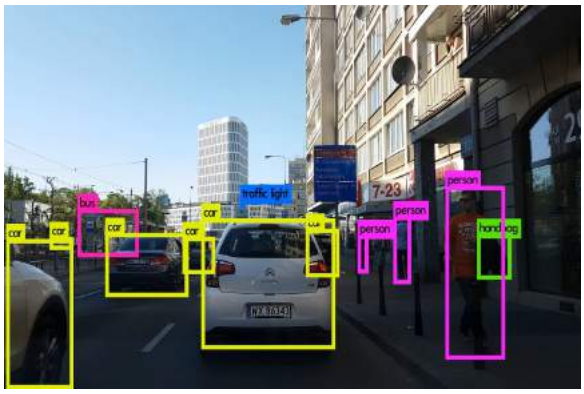


Figure 1: Computer Vision Tasks

1.2 Applications

A well known application of object detection is face detection, that is used in almost all the mobile cameras. A more generalized (multi-class) application can be used in autonomous driving where a variety of objects need to be detected. Also it has a important role to play in surveillance systems. These systems can be integrated with other tasks such as pose estimation where the first stage in the pipeline is to detect the object, and then the second stage will be to estimate pose in the detected region. It can be used for tracking objects and thus can be used in robotics and medical applications. Thus this problem serves a multitude of applications.

(a) Surveillance



(b) Autonomous vehicles

Figure 2: Applications of object detections

1.3 Challenges

The major challenge in this problem is that of the variable dimension of the output which is caused due to the variable number of objects that can be present in any given input image. Any general machine learning task requires a fixed dimension of input and output for the model to be trained. Another important obstacle for widespread adoption of object detection systems is the requirement of real-time (¿30fps) while being accurate in detection. The more complex the model is, the more time it requires for inference; and the less complex the model is, the less is the accuracy. This trade-off between accuracy and performance needs to be chosen as per the application. The problem involves classification as well as regression, leading the model to be learnt simultaneously. This adds to the complexity of the problem.

Existing System

There has been a lot of work in object detection using traditional computer vision techniques (sliding windows, deformable part models). However, they lack the accuracy of deep learning based techniques. Among the deep learning based techniques, two broad class of methods are prevalent: two stage detection (RCNN [1], Fast RCNN [2], Faster RCNN [3]) and unified detection (Yolo [4], SSD [5]). The major concepts involved in these techniques have been explained below.

2.1 Bounding Box

The bounding box is a rectangle drawn on the image which tightly fits the object in the image. A bounding box exists for every instance of every object in the image. For the box, 4 numbers (center_x, center_y, width, height) are predicted. This can be trained using a distance measure between predicted and ground truth bounding box. The distance measure is a jaccard distance which computes intersection over union between the predicted and ground truth boxes as shown in Fig. 3.

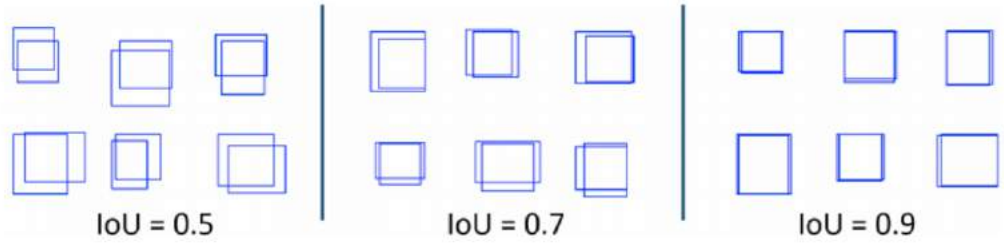


Figure 3: Jaccard distance

2.2 Classification + Regression

The bounding box is predicted using regression and the class within the bounding box is predicted using classification. The overview of the architecture is shown in Fig. 4

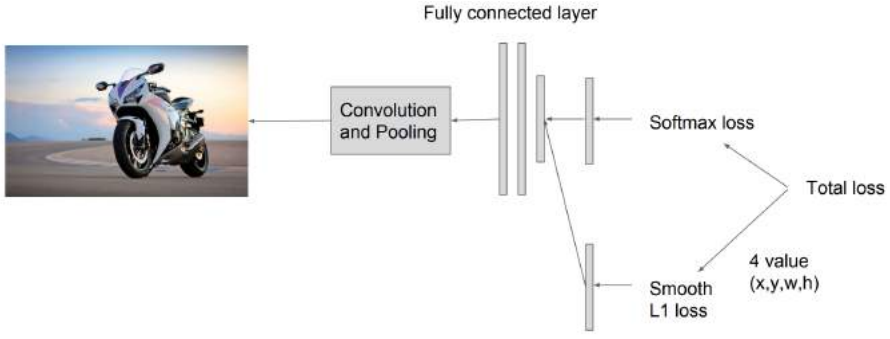
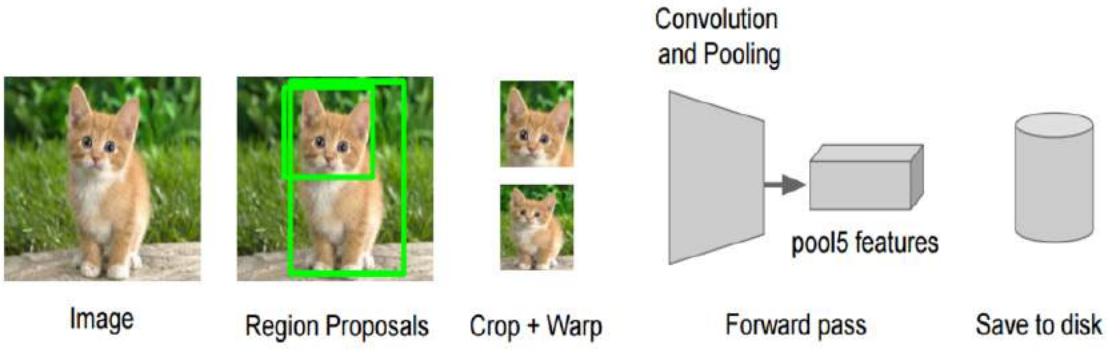


Figure 4: Architecture overview

2.3 Two-stage Method

In this case, the proposals are extracted using some other computer vision technique and then resized to fixed input for the classification network, which acts as a feature extractor. Then an SVM is trained to classify between object and background (one SVM for each class). Also a bounding box regressor is trained that outputs some some correction (offsets) for proposal boxes. The overall idea is shown in Fig. 5 These methods are very accurate but are computationally intensive (low fps).



(a) Stage 1

(b) Stage 2

Figure 5: Two stage method

2.4 Unified Method

The difference here is that instead of producing proposals, pre-define a set of boxes to look for objects. Using convolutional feature maps from later layers of the network, run another network over these feature maps to predict class scores and bounding box offsets. The broad idea is depicted in Fig. 6. The steps are mentioned below:

1. Train a CNN with regression and classification objective.
2. Gather activation from later layers to infer classification and location with a fully connected or convolutional layers.

3. During training, use jaccard distance to relate predictions with the ground truth.

4. During inference, use non-maxima suppression to filter multiple boxes around the same object.

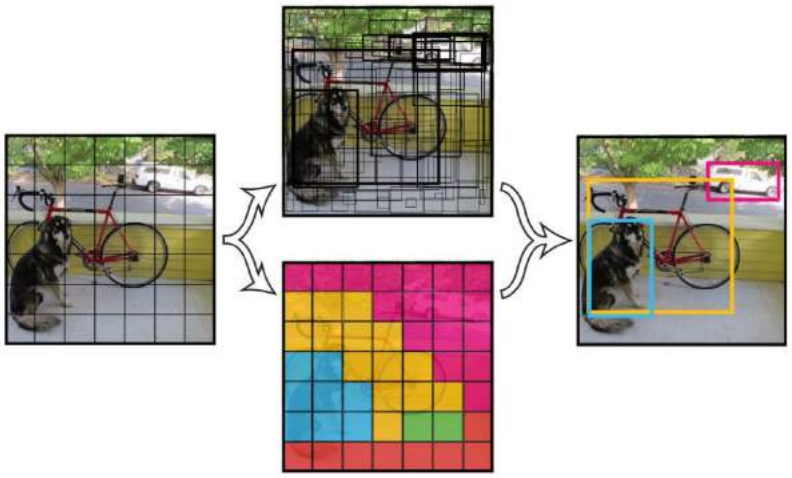


Figure 6: Unified Method

The major techniques that follow this strategy are: SSD (uses different activation maps (multiple-scales) for prediction of classes and bounding boxes) and Yolo (uses a single activation map for prediction of classes and bounding boxes). Using multiple scales helps to achieve a higher mAP(mean average precision) by being able to detect objects with different sizes on the image better. Thus the technique used in this project is SSD.

Proposed System and Approach

The network used in this project is based on Single shot detection (SSD) [5]. The architecture is shown in Fig. 7.

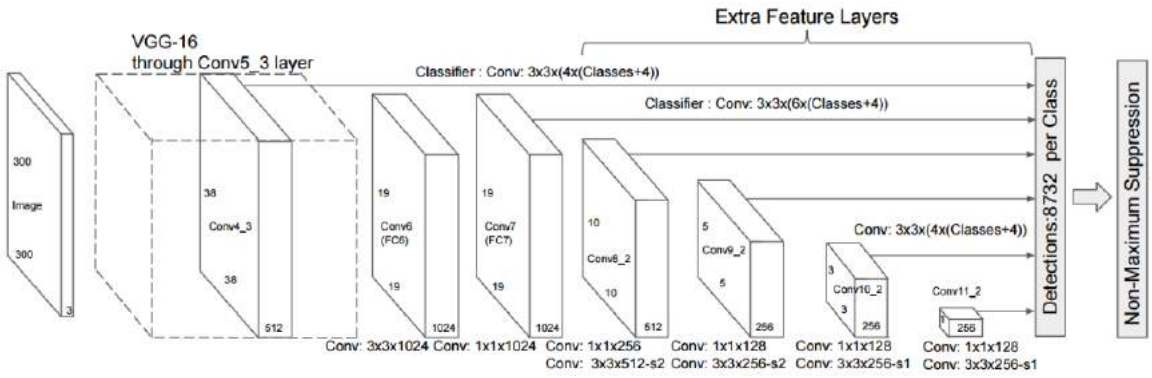


Figure 7: SSD Architecture

The SSD normally starts with a VGG [6] model, which is converted to a fully convolutional network. Then we attach some extra convolutional layers, that help to handle bigger objects. The output at the VGG network is a 38x38 feature map (conv4_3). The added layers produce 19x19, 10x10, 5x5, 3x3, 1x1 feature maps. All these feature maps are used for predicting bounding boxes at various scales (later layers responsible for larger objects).

Thus the overall idea of SSD is shown in Fig. 8. Some of the activations are passed to the sub-network that acts as a classifier and a localizer.

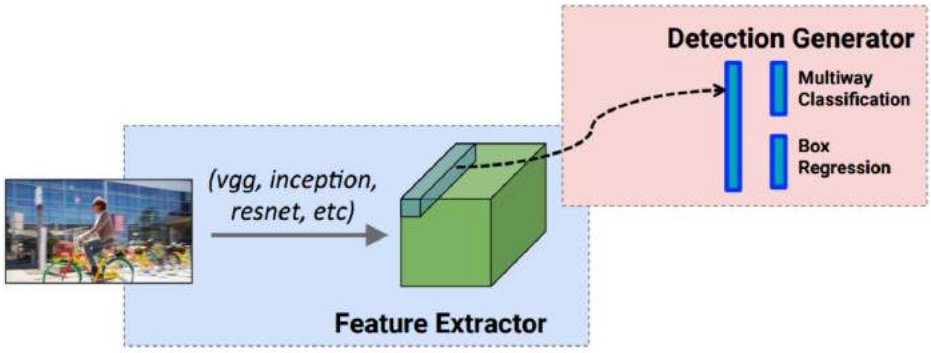


Figure 8: SSD Overall Idea

Anchors (collection of boxes overlaid on image at different spatial locations, scales and aspect ratios) act as reference points on ground truth images as shown in Fig. 9.

A model is trained to make two predictions for each anchor:

- A discrete class
- A continuous offset by which the anchor needs to be shifted to fit the ground-truth bounding box

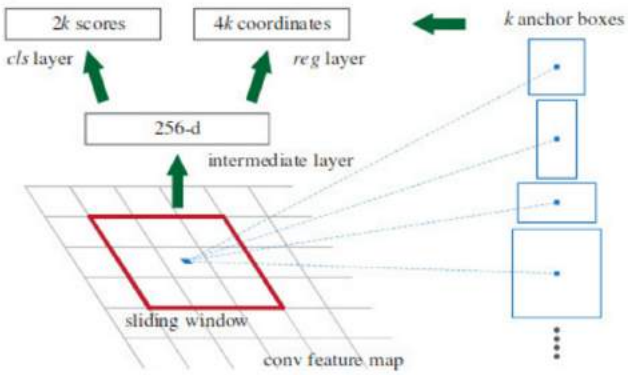


Figure 9: Anchors

During training SSD matches ground truth annotations with anchors. Each element of the feature map (cell) has a number of anchors associated with it. Any anchor with an IoU (jaccard distance) greater than 0.5 is considered a match. Consider the case as shown in Fig. 10, where the cat has two anchors matched and the dog has one anchor matched. Note that both have been matched on different feature maps.

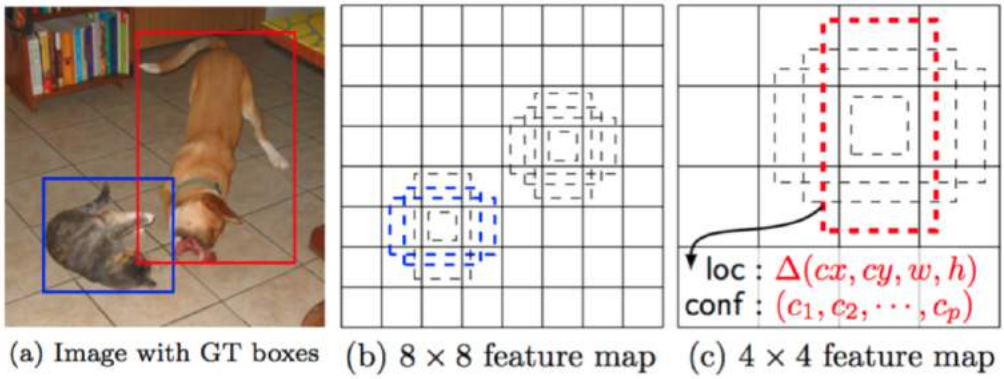


Figure 10: Matches

The loss function used is the multi-box classification and regression loss. The classification loss used is the softmax cross entropy and, for regression the smooth L1 loss is used.

During prediction, non-maxima suppression is used to filter multiple boxes per object that may be matched as shown in Fig. 11.

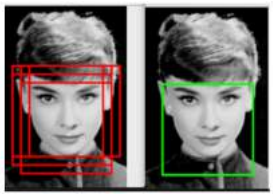


Figure 11: Non-maxima suppression

Implementation

4.1 Dataset

For the purpose of this project, the publicly available PASCAL VOC dataset will be used. It consists of 10k annotated images with 20 object classes with 25k object annotations (xml format). These images are downloaded from flickr. This dataset is used in the PASCAL VOC Challenge which runs every year since 2006.

Figure 12: Dataset

4.2 Implementation Details

The project is implemented in python 3. Tensorflow was used for training the deep network and OpenCV was used for image pre-processing.

The system specifications on which the model is trained and evaluated are mentioned as follows: CPU - Intel Core i7-7700 3.60 GHz, RAM - 32 Gb, GPU - Nvidia Titan Xp.

4.2.1 Pre-processing

The annotated data is provided in xml format, which is read and stored into a pickle file along with the images so that reading can be faster. Also the images are resized to a fixed size.

4.2.2 Network

The entire network architecture is shown in Fig. 13. The model consists of the base network derived from VGG net and then the modified convolutional layers for fine-tuning and then the classifier and localizer networks. This creates a deep network which is trained end-to-end on the dataset.

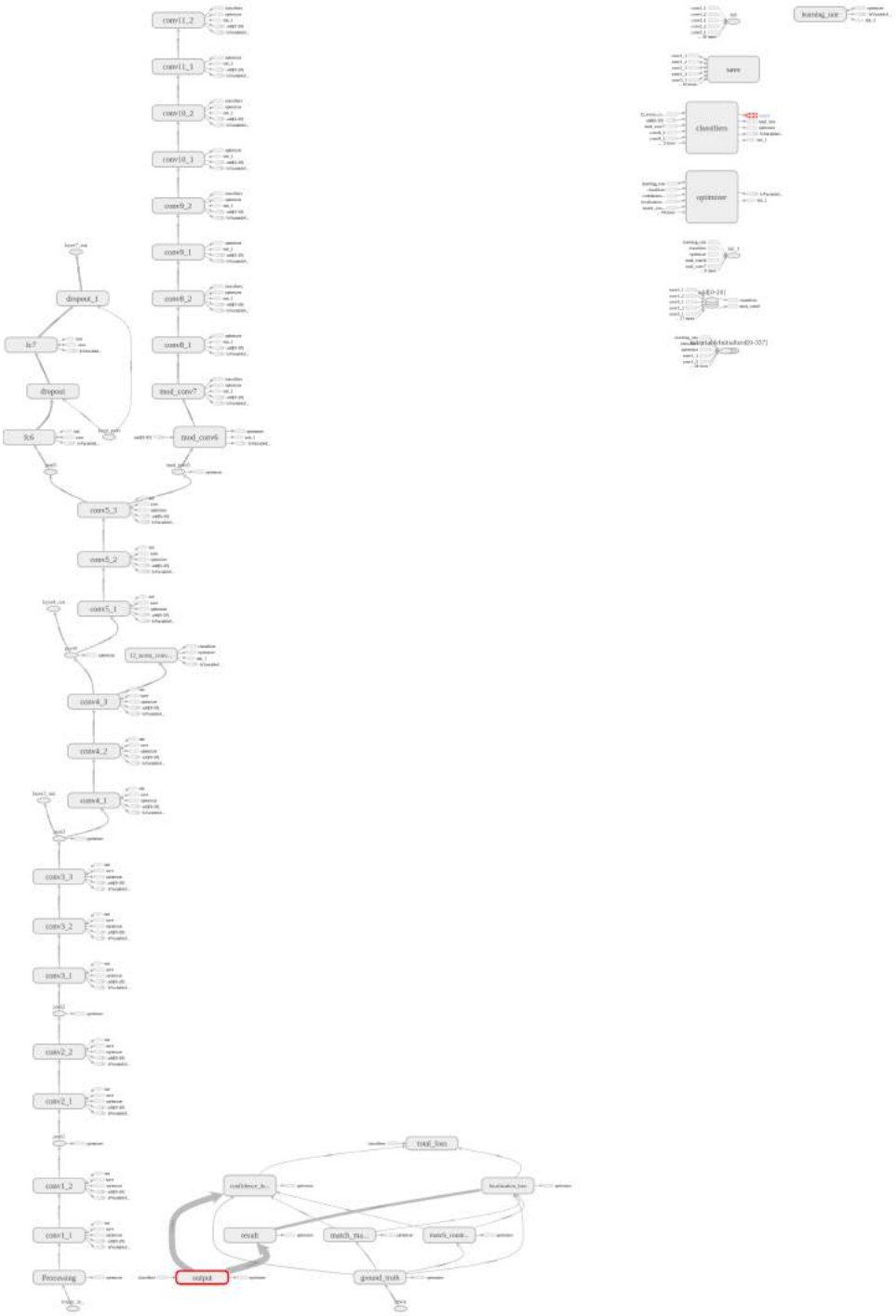


Figure 13: Network in Tensorboard

4.3 Qualitative Analysis

The results from the PASCAL VOC dataset are shown in Table 1.

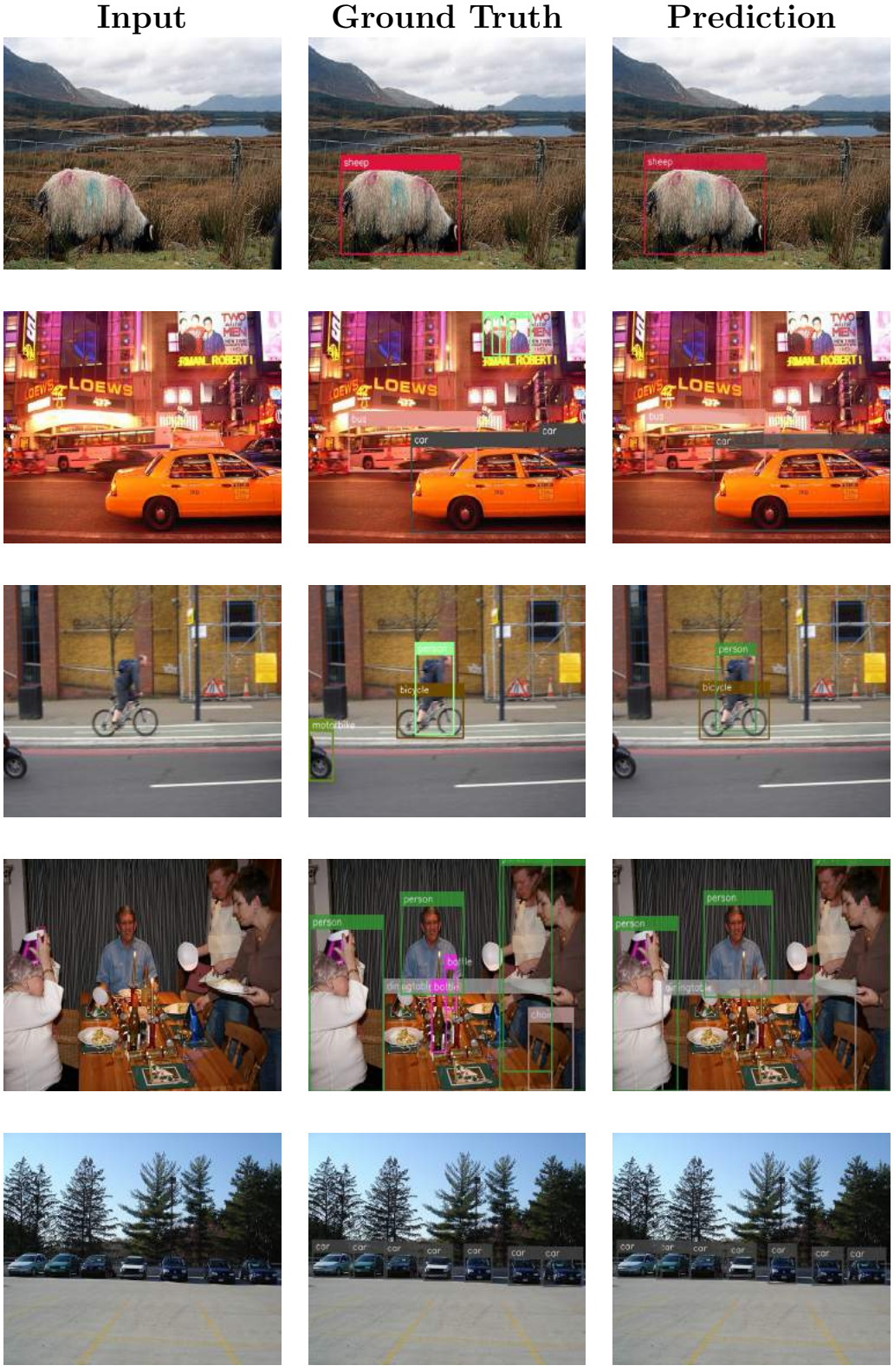


Table 1: Detection results on PASCAL VOC dataset.

The results on custom dataset are shown in Table 2.

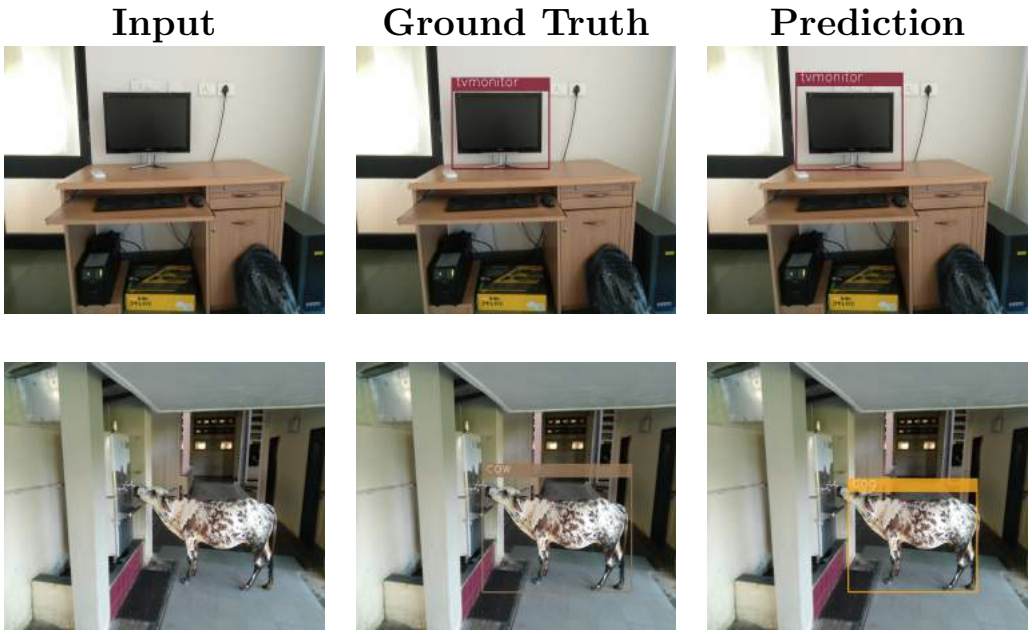


Table 2: Detection results on custom dataset.

The system handles illumination variations thus providing a robust detection. In Fig. 14 the same person is standing in the shade and then in the sunny environment.

(a) High illumination (b) Low illumination

Figure 14: Detection robust to illumination variation

However, occlusion creates a problem for detection. As shown in Fig. 15, the occluded birds are not detected correctly.

Also larger object dominated when present along with small objects as found in Fig. 16. This could be the reason for the average precision of smaller objects to be less when compared to larger objects. This has been reported in the next section.

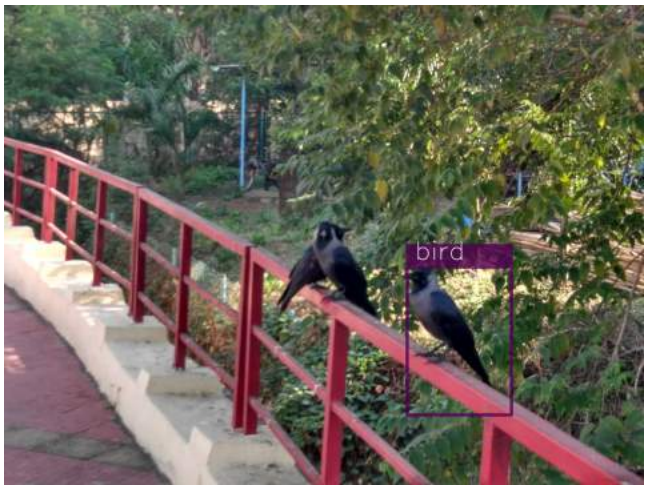


Figure 15: Occlusion

(a) Only small object in image

(b) Small and large object in image

Figure 16: Domination of larger object in detection

4.4 Quantitative Analysis

The evaluation metric used is mean average precision (mAP). For a given class, precision-recall curve is computed. Recall is defined as the proportion of all positive examples ranked above a given rank. Precision is the proportion of all examples above that rank which are from the positive class. The AP summarizes the shape of the precision-recall curve, and is defined as the mean precision at a set of eleven equally spaced recall levels [0, 0.1, ... 1]. Thus to obtain a high score, high precision is desired at all levels of recall. This measure is better than area under curve (AUC) because it gives importance to the sensitivity.

The detections were assigned to ground truth objects and judged to be true/false positives by measuring bounding box overlap. To be considered a correct detection, the area of overlap between the predicted bounding box and ground truth bounding box must exceed a threshold. The output of the detections assigned to ground truth objects satisfying the overlap criterion were ranked in order of (decreasing) confidence output. Multiple detections of the same object in an image were considered false detections, i.e. 5 detections of a single object counted as 1 true positive and 4 false positives. If no prediction is made for an image

then it is considered a false negative.

The average precision for all the object categories are reported in Table 3. The mAP for the PASCAL VOC dataset was found to be 0.633. The current state-of-the-art best mAP value is reported to be 0.739.

Class	**Average Precision**
Motorbike	0.724
Bottle	0.272
Bird	0.635
Cat	0.909
Aeroplane	0.727
Chair	0.360
Person	0.542
Diningtable	0.534
Boat	0.544
Train	0.909
Sofa	0.710
Bicycle	0.636
Bus	0.726
Horse	0.726
Tvmonitor	0.633
Cow	0.632
Pottedplant	0.359
Car	0.634
Dog	0.818
Sheep	0.633

Table 3: Average precision for all classes

Output

```
import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
import tensorflow as tf
import zipfile

from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image

from utils import label_map_util
from utils import visualization_utils as vis_util

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
MODEL_NAME = 'ssd_mobilenet_v1_coco_11_06_2017'
MODEL_FILE = MODEL_NAME + '.tar.gz'
DOWNLOAD_BASE = 'http://download.tensorflow.org/models/object_detection/'
PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'

PATH_TO_LABELS = os.path.join('data', 'mscoco_label_map.pbtxt')

NUM_CLASSES = 90

if not os.path.exists(MODEL_NAME + '/frozen_inference_graph.pb'):
        print ('Downloading the model')
        opener = urllib.request.URLopener()
        opener.retrieve(DOWNLOAD_BASE + MODEL_FILE, MODEL_FILE)
        tar_file = tarfile.open(MODEL_FILE)
        for file in tar_file.getmembers():
          file_name = os.path.basename(file.name)
          if 'frozen_inference_graph.pb' in file_name:
            tar_file.extract(file, os.getcwd())
        print ('Coco Model Download complete!!!')
```

```
else:
        print ('COCO Model already exists')

detection_graph = tf.Graph()
with detection_graph.as_default():
  od_graph_def = tf.GraphDef()
  with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
    serialized_graph = fid.read()
    od_graph_def.ParseFromString(serialized_graph)
    tf.import_graph_def(od_graph_def, name='')

label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES, use_display_name=True)
category_index = label_map_util.create_category_index(categories)
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)

PATH_TO_TEST_IMAGES_DIR = 'C:/Users/MY
LENOVO/PycharmProjects/test/object_recognition_detection/test_images/'
TEST_IMAGE_PATHS = [ os.path.join(PATH_TO_TEST_IMAGES_DIR,
'image{}.jpg'.format(i)) for i in range(1, 3) ]

IMAGE_SIZE = (12, 8)

with detection_graph.as_default():
  with tf.Session(graph=detection_graph) as sess:
    for image_path in TEST_IMAGE_PATHS:
      image = Image.open(image_path)

      image_np = load_image_into_numpy_array(image)
      image_np_expanded = np.expand_dims(image_np, axis=0)
      image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')
      boxes = detection_graph.get_tensor_by_name('detection_boxes:0')

      scores = detection_graph.get_tensor_by_name('detection_scores:0')
      classes = detection_graph.get_tensor_by_name('detection_classes:0')
```

```
num_detections = detection_graph.get_tensor_by_name('num_detections:0')

(boxes, scores, classes, num_detections) = sess.run(
    [boxes, scores, classes, num_detections],
    feed_dict={image_tensor: image_np_expanded})
vis_util.visualize_boxes_and_labels_on_image_array(
    image_np,
    np.squeeze(boxes),
    np.squeeze(classes).astype(np.int32),
    np.squeeze(scores),
    category_index,
    use_normalized_coordinates=True,
    line_thickness=8)
plt.figure(figsize=IMAGE_SIZE)
plt.imshow(image_np)
plt.show()
```

Conclusion

An accurate and efficient object detection system has been developed which achieves comparable metrics with the existing state-of-the-art system. This project uses recent techniques in the field of computer vision and deep learning. Custom dataset was created using labelling and the evaluation was consistent. This can be used in real-time applications which require object detection for pre-processing in their pipeline. An important scope would be to train the system on a video sequence for usage in tracking applications. Addition of a temporally consistent network would enable smooth detection and more optimal than per-frame detection.