CO352

Computer Graphics Mini Project

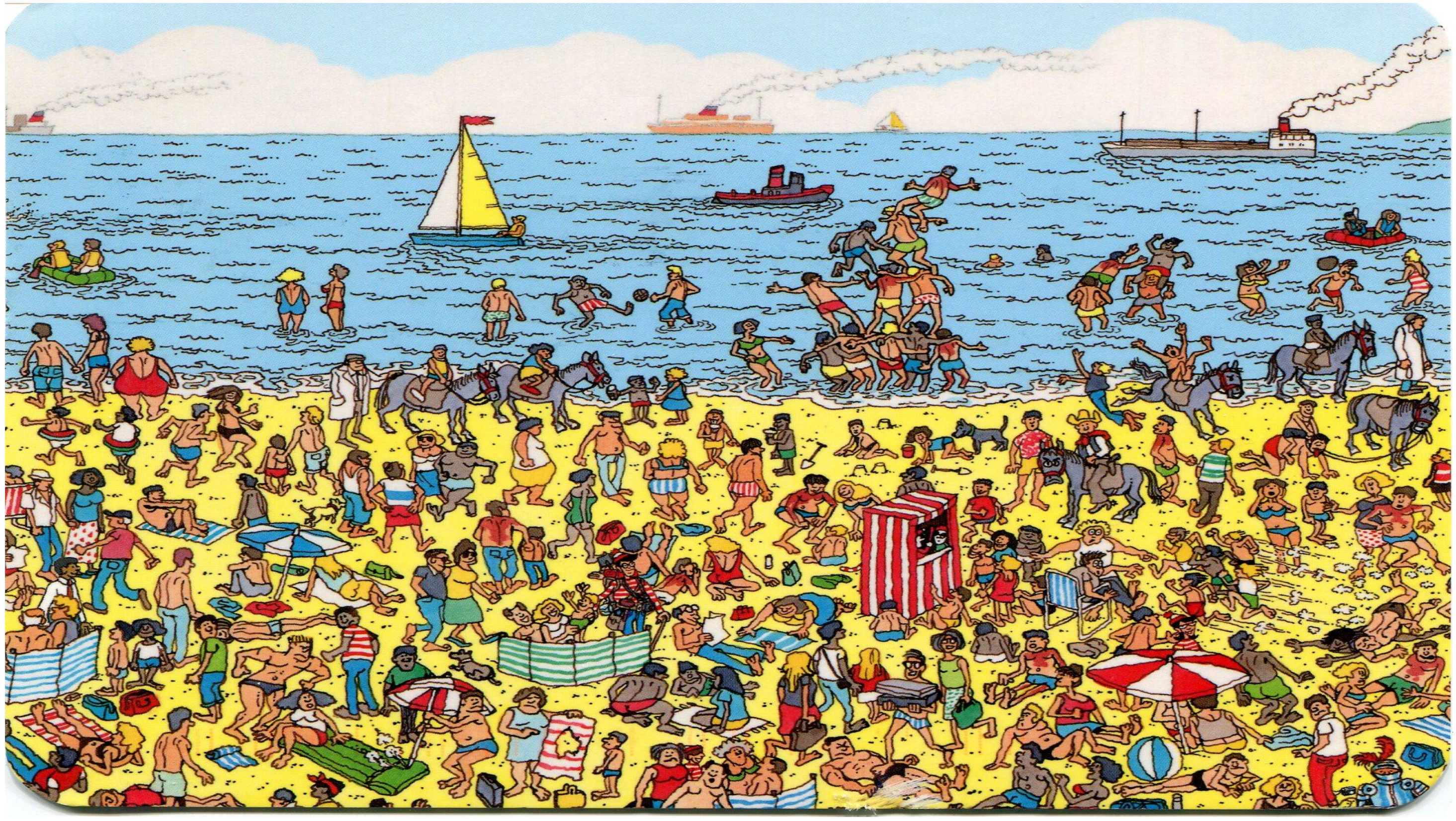
Solution to the Waldo Puzzle using Object Detection in Tensorflow

What is the Waldo Puzzle?

The Waldo Puzzle consists of a picture or an image with a lot of different characters and objects. The objective is to locate the position of a certain character named **Waldo** in the image.

The difficulty of the puzzle is usually high,and it generally takes a considerable amount of time to find him using the naked eye.

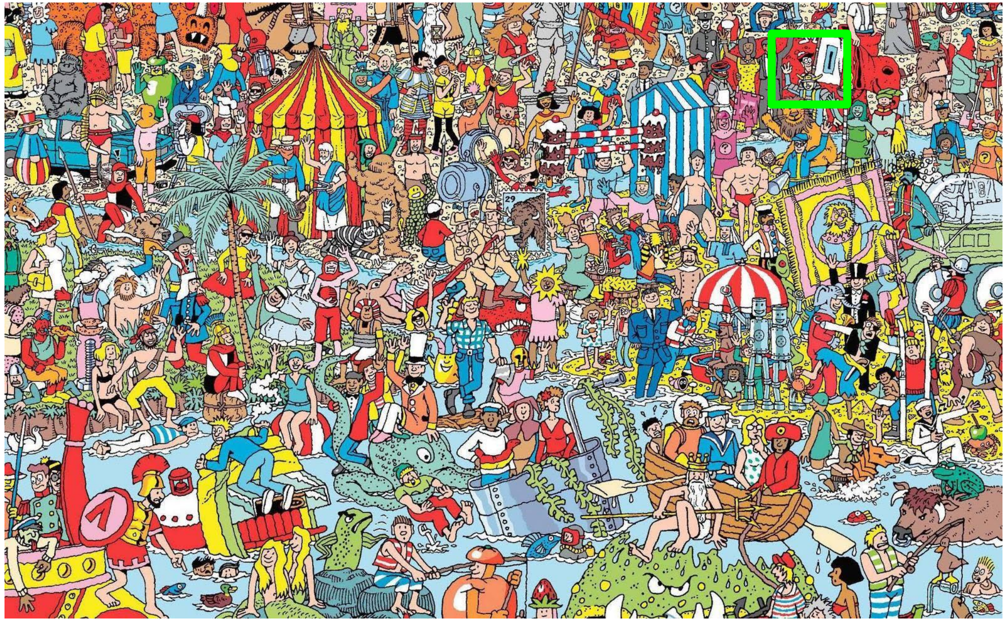
An example of the Waldo Puzzle



Project Aim and Objectives

- **Main Aim**: To find the position of Waldo in the Waldo puzzle.
- **Methods**:
  1. Template Matching
  2. Machine Learning
- **Tools Used:**
  1. OpenCV
  2. Tensorflow

Template Matching using OpenCV



Machine Learning Approach

- Tensorflow Object Detection API to locate Waldo’s position in the parent image.
- Model Used : RCNN with Inception V2 model
- Steps Involved:
  1. **Preparing the Dataset** by creating a set of labelled training images
  2. **Fetching and configuring the model** to use with Tensorflow Object Detection API
  3. **Training the model** on our dataset
  4. **Testing the model on evaluation images**

Retraining/Transfer Learning

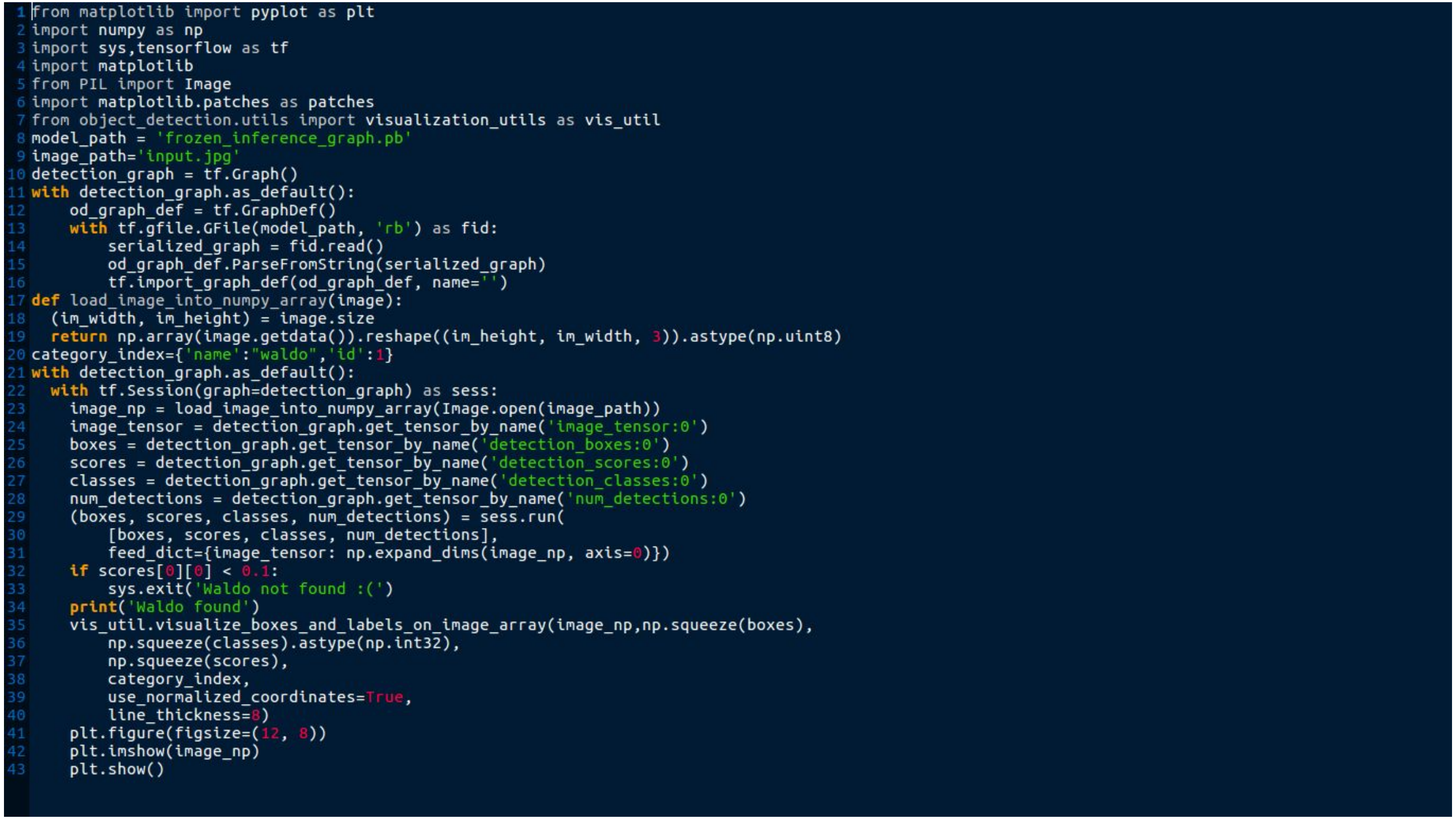
- The model could be trained from scratch, but this process would probably take weeks. Instead, we used a method called **transfer learning.**
- **Transfer learning** involves taking a model usually trained to solve some general problem, and retraining it to solve ours. The idea behind transfer learning is that instead of training our model from scratch, we can use the knowledge obtained in the pre-trained model and transfer it to our new model.
- This saves us **a lot of time** so that the time spent for training can be invested into obtaining only the knowledge specific to our problem.

What we did: We collected 15 Waldo puzzle images along with respective positions of Waldo in a csv file. We used RCNN with Inception V2 model already trained to detect pets(COCO dataset) . This will be used to train the model to detect Waldo.

Training the model

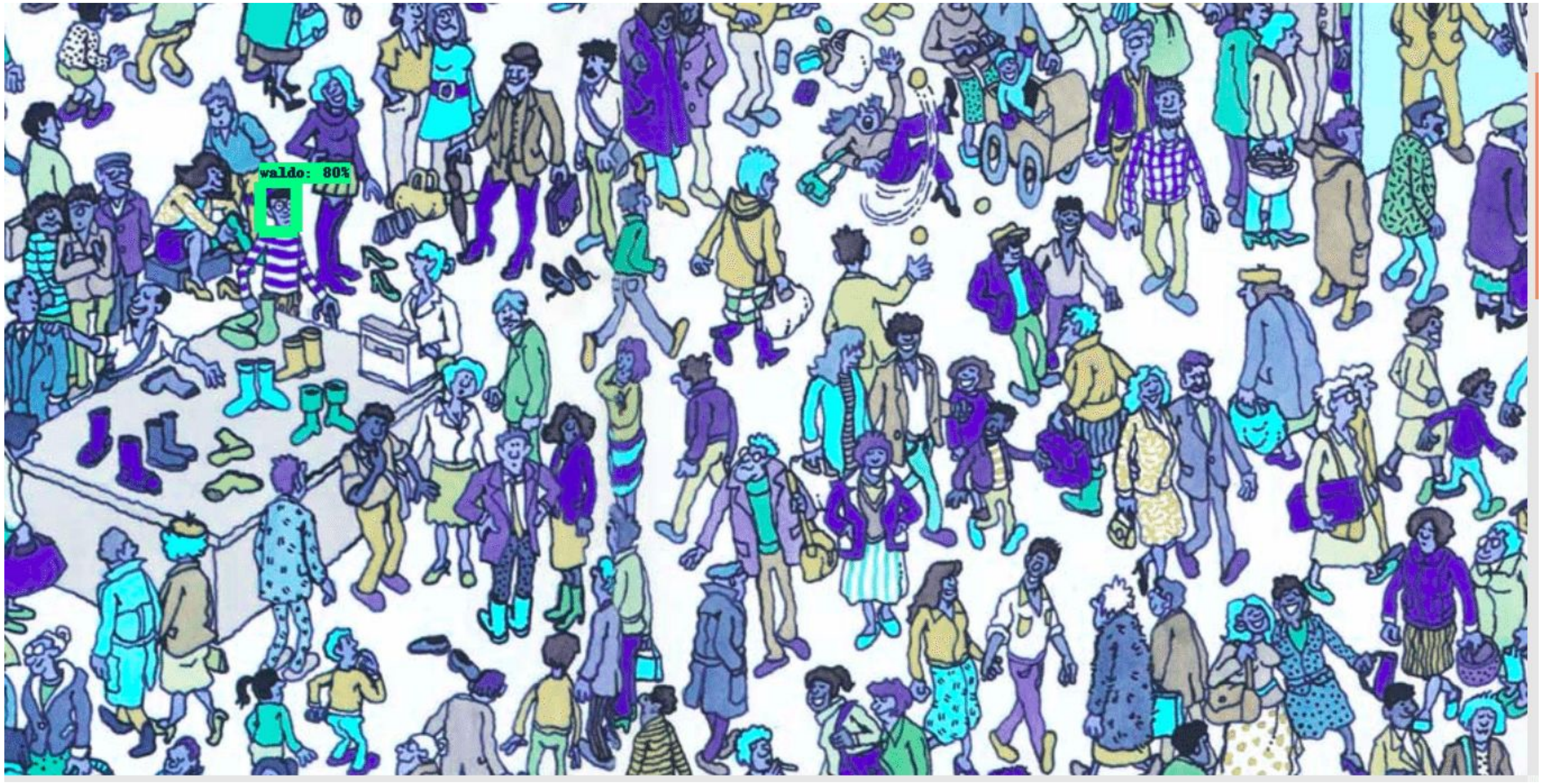
- The process of training the model involves providing an algorithm with training data to learn from.
- The learning algorithm finds patterns in the training data that map the input data attributes to the target(Waldo) , and it outputs a model that captures these patterns.
- While training, the most important information to look for is **loss**. It’s a summation of the errors made for each example in training or validation sets. We want it to be as low as possible, meaning that if it’s slowly decreasing, that means that our model is learning.

Source Code



```python
from matplotlib import pyplot as plt
import numpy as np
import sys,tensorflow as tf
import matplotlib
from PIL import Image
import matplotlib.patches as patches
from object_detection.utils import visualization_utils as vis_util
model_path = 'frozen_inference_graph.pb'
image_path='input.jpg'
detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(model_path, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')
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)
category_index={'name':"waldo",'id':1}
with detection_graph.as_default():
  with tf.Session(graph=detection_graph) as sess:
    image_np = load_image_into_numpy_array(Image.open(image_path))
    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: np.expand_dims(image_np, axis=0)})
    if scores[0][0] < 0.1:
        sys.exit('Waldo not found :(')
    print('Waldo found')
    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=(12, 8))
    plt.imshow(image_np)
    plt.show()
```

Output



Applications

The applications of object detection are numerous. Some of them are:

- **People Counting**
- **Vehicle detection**
- **Manufacturing Industry**
- **Online image classifying**
- **Self driving cars**

Advantages and Disadvantages

Advantages	Disadvantages
1. Finding Waldo using machine learning saves us considerable amount of time compared to finding it with the naked eye. 2. The concept of object detection can be extended to a variety of applications.	1. When Waldo is very large in the image, our model fails to find him. 2. The process of training can be time-consuming especially if there are a variety of objects to be identified.

Conclusion

- The model we trained worked well for all testing data(images). It managed to find Waldo in the evaluation images and did pretty great on some extra random examples from the internet.

- It failed to find Waldo where he was really large, which should be even easier to solve as opposed to finding him where he’s really small. This probably was because our model overfit our training data as a result of using only a handful of training images.

THANK YOU!