**Object Detection with ImageAI in Python**

Introduction

**Object detection** is a technology that falls under the broader domain of [Computer Vision](https://en.wikipedia.org/wiki/Computer_vision). It deals with **identifying** and **tracking objects** present in images and videos. **Object detection** has multiple applications such as face detection, vehicle detection, pedestrian counting, self-driving cars, security systems, etc.

The two major objectives of **object detection** include:

* To **identify all objects** present in an image
* Filter out the object of attention

In this article, you will see how to perform object detection in Python with the help of the [ImageAI](https://imageai.readthedocs.io/en/latest/detection/index.html" \t "_blank) library.

### Deep Learning for Object Detection

Deep learning techniques have been proven state of the art for various object detection problems. The following are some of the commonly used deep learning approaches for **object detection**:

* **ImageAI**
* [Single Shot Detectors](https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/single-shot-detectors.html)
* [YOLO (You only look once)](https://appsilon.com/object-detection-yolo-algorithm/)
* [Region-based Convolutional Neural Networks](https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/single-shot-detectors.html)

In the rest of this article, we will see what exactly **ImageAI** is and how to use it to perform **object detection**.

### ImageAI

ImageAI is a Python library built to empower developers to build applications and systems with self-contained deep learning and **Computer Vision** capabilities using a few lines of straight forward code. **ImageAI** contains a Python implementation of almost all of the state-of-the-art deep learning algorithms like [RetinaNet](https://arxiv.org/abs/1708.02002" \t "_blank), [YOLOv3](https://pjreddie.com/darknet/yolo/), and TinyYOLOv3.

ImageAI makes use of several APIs that work offline - it has object detection, video detection, and object tracking APIs that can be called without internet access. **ImageAI** makes use of a pre-trained model and can easily be customized.

The ObjectDetection class of the **ImageAI** library contains functions to perform object detection on any image or set of images, using pre-trained models. With **ImageAI**, you can detect and recognize 80 different kinds of common, everyday objects.

#### Setting up your Environment

In this part of the tutorial, we will work through the installation of **ImageAI**.

To use ImageAI you need to install a few dependencies. The first step is to have Python installed on your computer. Download and install Python 3 from the [official Python website](https://www.python.org/).

Once you have Python installed on your computer, install the following dependencies using pip:

**TensorFlow**

$ pip install tensorflow

**OpenCV**

$ pip install opencv-python

**Keras**

$ pip install keras

**ImageAI**

$ pip install imageAI

Now download the [TinyYOLOv3](https://github.com/OlafenwaMoses/ImageAI/releases/download/1.0/yolo-tiny.h5) model file that contains the classification model that will be used for object detection.

Performing Object Detection with ImageAI

Now let's see how to actually use the ImageAI library. I'll explain step by step how you can build your first object detection model with ImageAI.

**Step 1**

Our first task here is to create the necessary **folders**. For this tutorial we need the following folders:

* **Object detection**: root folder
* **models**: stores pre-trained model
* **input**: stores image file on which we want to perform object detection
* **output**: stores image file with detected objects

After you have created your folders, your Object detection folder should have the following **sub-folders**:

├── input  
├── models  
└── output  
3 directories, 0 files

**Step 2**

Open your preferred text editor for writing Python code and create a new file detector.py.

**Step 3**

Import ObjectDetection class from the ImageAI library.

from imageai.Detection import ObjectDetection

**Step 4**

Now that you have imported imageAI library and the ObjectDetection class , the next thing is to create an instance of the class ObjectDetection, as shown here:

detector = ObjectDetection()

**Step 5**

Let's specify the path from our input image, output image, and model.

model\_path = "./models/yolo-tiny.h5"  
input\_path = "./input/test45.jpg"  
output\_path = "./output/newimage.jpg"

**Step 6**

After instantiating the ObjectDetection class we can now call various functions from the class. The class contains the following functions to call **pre-trained models**: setModelTypeAsRetinaNet(), setModelTypeAsYOLOv3(), and setModelTypeAsTinyYOLOv3().  
For the purpose of this tutorial, I'll be using the pre-trained TinyYOLOv3 model, and hence we will use the setModelTypeAsTinyYOLOv3() function to load our model.

detector.setModelTypeAsTinyYOLOv3()

**Step 7**

Next, we are going to call the function setModelPath(). This function accepts a string which contains the path to the pre-trained model:

detector.setModelPath(model\_path)

**Step 8**

This step calls the function loadModel() from the detector instance. It loads the model from the path specified above using the setModelPath() class method.

detector.loadModel()

**Step 9**

To detect objects in the image, we need to call the detectObjectsFromImage   
function using the detector object that we created in the previous section.  
This function requires two arguments: input\_image and output\_image\_path.   
input\_image is the path where the image we are detecting is located, while the   
output\_image\_path parameter is the path to store the image with detected objects. This function returns a **dictionary** which contains the names and percentage   
probabilities of all the objects detected in the image.

detection = detector.detectObjectsFromImage(input\_image=input\_path, output\_image\_path=output\_path)

**Step 10**

The dictionary items can be accessed by traversing through each item in the dictionary.

for eachItem in detection:  
 print(eachItem["name"] , " : ", eachItem["percentage\_probability"])

**Complete Code for Object Detection**

Here is the complete code for the image detection:

from imageai.Detection import ObjectDetection  
detector = ObjectDetection()  
model\_path = "./models/yolo-tiny.h5"  
input\_path = "./input/test45.jpg"  
output\_path = "./output/newimage.jpg"  
detector.setModelTypeAsTinyYOLOv3()  
detector.setModelPath(model\_path)  
detector.loadModel()  
detection = detector.detectObjectsFromImage(input\_image=input\_path, output\_image\_path=output\_path)  
for eachItem in detection:  
 print(eachItem["name"] , " : ", eachItem["percentage\_probability"])

In the output, you can see the name of each detected object along with its percentage probability as shown below:  
**Output**

car : 54.72719073295593  
car : 58.94589424133301  
car : 62.59384751319885  
car : 74.07448291778564  
car : 91.10507369041443  
car : 97.26507663726807  
car : 97.55765795707703  
person : 53.6459743976593  
person : 56.59831762313843  
person : 72.28181958198547

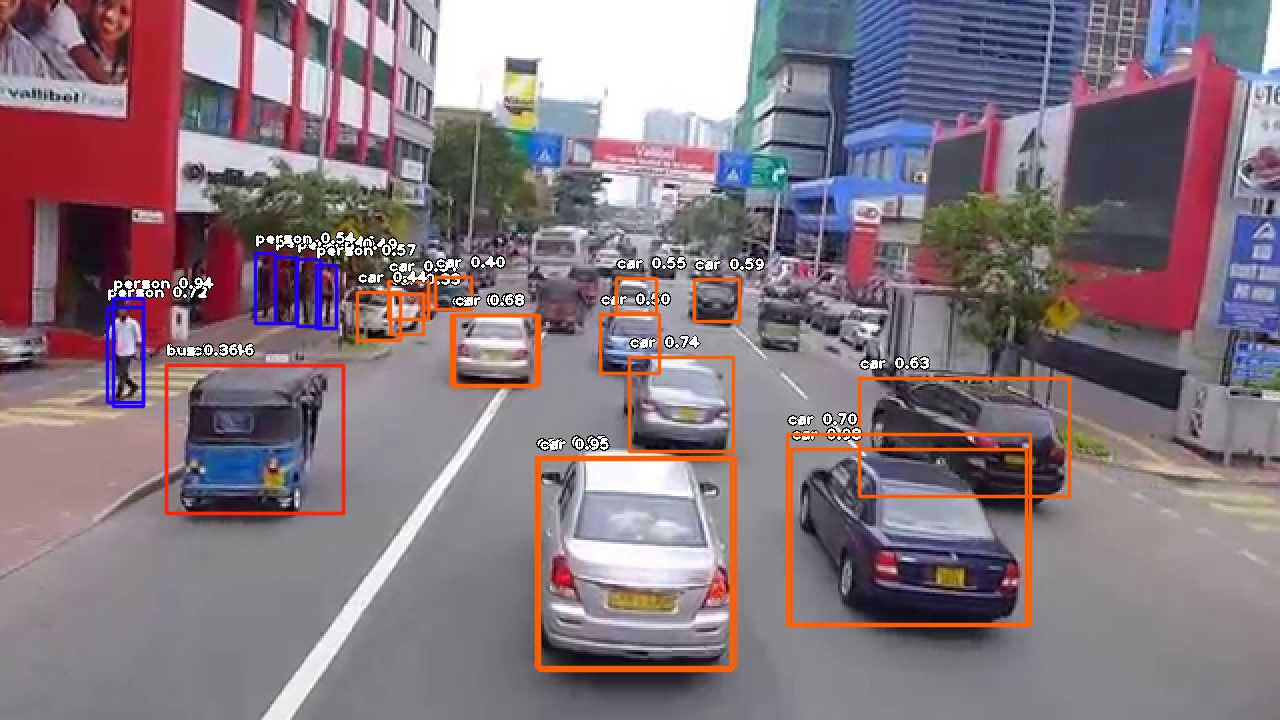
**Original image:**

The original image, i.e. "test45", looked like this:



**Image with Object Detection:**

After the object detection, the resulting image looks like this:



You can see that ImageAI has successfully identified cars and persons in the image.

Conclusion

**Object detection** is one of the most common **computer vision tasks**. This article explains how to perform object detection in Python using the ImageAI library with the help of an example.

References

* <https://imageai.readthedocs.io/en/latest/detection/index.html>