```python
#The code below is for training the model for object recognition of the oscilloscope
#The code is adapted from Glenn Prince's work of object recognition
#Website of the original address:
#https://www.codeproject.com/Articles/5270246/Custom-Model-Object-Detection-with-OpenCV-and-Imag
#We use the YOLO3 detection method and tools from openCV, imageAI

#import libraries

import cv2 as cv
from imageai.Detection import ObjectDetection as od
import numpy as np
import requests as req
import os as os
```

```python
#Import the model trainer from the ImageAI toolbox
from imageai.Detection.Custom import DetectionModelTrainer

trainer = DetectionModelTrainer()
trainer.setModelTypeAsYOLOv3()
```

```python
#Set the directory of the training data
trainer.setDataDirectory(data_directory="DATA")
#Set the name of the desired object and set batch size
#Larger batch size usually requires longer training time but the result would be accurate
trainer.setTrainConfig(object_names_array=["oscilloscope"], batch_size=10, num_experiments=20)

trainer.trainModel()
```

```python
#After getting the training model, we can choose some of them to evaluate
#The loss function decreases for each model, so the model corresponding to later epoches usually has better result
#We can see the increase of recognition accuracy or choose the best model using the following code aftter evaluation
model05 = trainer.evaluateModel(model_path="DATA/models/detection_model-ex-005--loss-0039.274.h5",
                      json_path="DATA/json/detection_config.json", iou_threshold=0.5,
                      object_threshold=0.3, nms_threshold=0.5)
model10 = trainer.evaluateModel(model_path="DATA/models/detection_model-ex-011--loss-0032.434.h5",
                      json_path="DATA/json/detection_config.json", iou_threshold=0.5,
                      object_threshold=0.3, nms_threshold=0.5)
model15 = trainer.evaluateModel(model_path="DATA/models/detection_model-ex-015--loss-0030.568.h5",
                      json_path="DATA/json/detection_config.json", iou_threshold=0.5,
                      object_threshold=0.3, nms_threshold=0.5)
model20 = trainer.evaluateModel(model_path="DATA/models/detection_model-ex-023--loss-0026.242.h5",
                      json_path="DATA/json/detection_config.json", iou_threshold=0.5,
                      object_threshold=0.3, nms_threshold=0.5)

print('---------------------------------------------------------')
print('Iteration 05:', model05[0]['average_precision']['oscilloscope'])
print('Iteration 10:', model10[0]['average_precision']['oscilloscope'])
print('Iteration 15:', model15[0]['average_precision']['oscilloscope'])
print('Iteration 20:', model20[0]['average_precision']['oscilloscope'])
print('---------------------------------------------------------')
```

```python
#The code below tests the model and visualize the object recognition of previous trained model
#Import library and set the detection model path
from imageai.Detection.Custom import CustomObjectDetection

detector = CustomObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath("./DATA/models/detection_model-ex-023--loss-0026.242.h5")
detector.setJsonPath("./DATA/json/detection_config.json")
detector.loadModel()
```

```python
#A helper function that shows the image as request
def showImage(img):
    #resize the image to fit the screen (sometimes the image gets too big)
    img = cv.resize(img, (1440,960))
    window_name = 'image'
    cv.imshow(window_name, img)

    cv.waitKey(0)
    cv.destroyAllWindows()
```

```python
#Set the path we want to test the model on
#The function below randomly choose an image from the data directory
#It will show all of the possible identification that has probability above the threshold
#Increase the threshold decreases the numbers of detection shown on the screen
import random
oscImages = os.listdir("DATA/oscilloscope")
randomFile = oscImages[random.randint(0, len(oscImages) - 1)]

detectedImage, detections = detector.detectObjectsFromImage(output_type="array", input_image="DATA/oscilloscope/271.
#convertedImage = cv.cvtColor(detectedImage, cv.COLOR_RGB2BGR)
showImage(detectedImage)

for eachObject in detections:
    print(eachObject["name"] , " : ", eachObject["percentage_probability"], " : ", eachObject["box_points"] )
    print("--------------------------------")
```