Tutorial by Nooh Ayub
(MaskRCNN For Safety-Object-Detection using Tensorflow)

Objectives:

• Understand Pre-trained model

• Train this pretrained model for your custom dataset

• Evaluating the model on test dataset.

Pretrained Weights:

The COCO weights will be downloaded automatically by the name "mask_rcnn_coco.h5"

Dataset:

- We have used a construction-safety-object-detection dataset downloaded from Kaggle for this experiment. (The dataset is provided in the repository)

- We will be training our model on this dataset

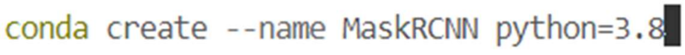
Supported Backbones:

- Resnet101
- Resnet50

Here’s a step by step breakdown:

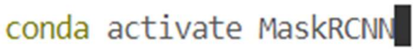
Step 1 — Create a directory named “Mask-R-CNN” and inside this directory create a virtual environment:

(a) create a conda environment with python 3.8:



```
conda create --name MaskRCNN python=3.8
```

(b) now activate this environment:

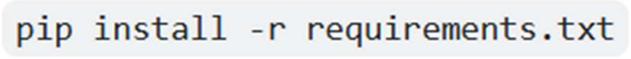


```
conda activate MaskRCNN
```

Step 2 — Now clone the following github repository and go inside that cloned folder using the subsequent command:

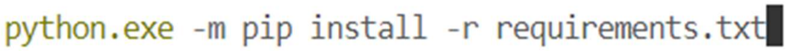
git clone https://github.com/nooh007/Mask-R-CNN-using-Tensorflow2.git

Step 3 — Next install the requirements.txt file by using the following command:



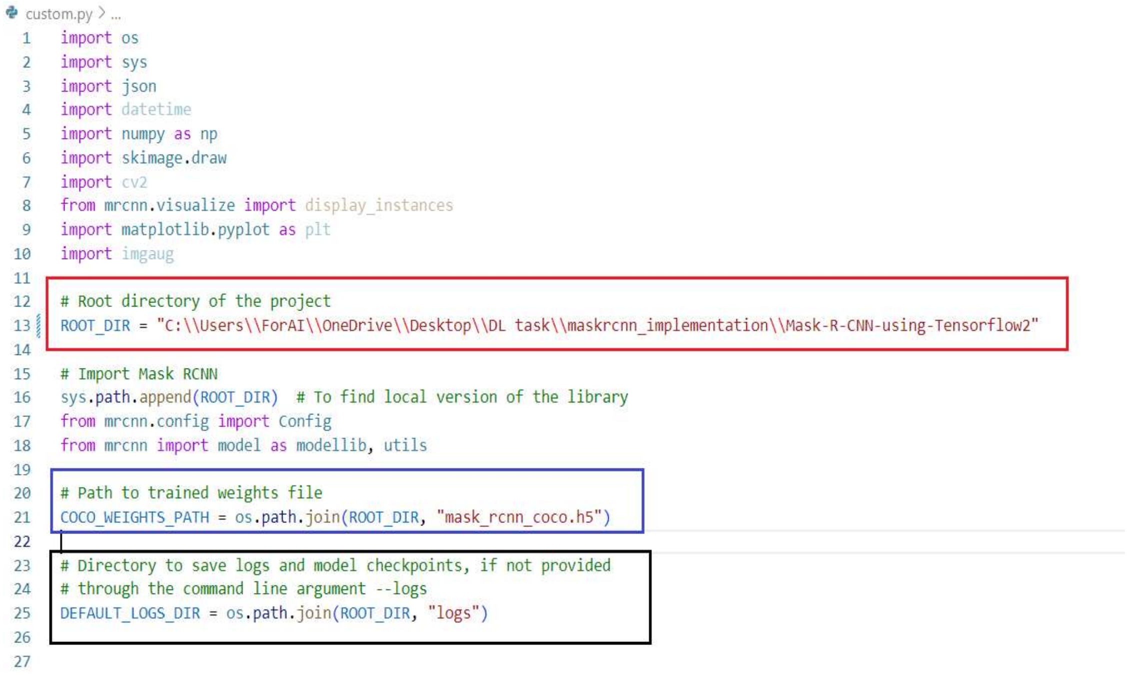
```
pip install -r requirements.txt
```

Or (if error occurs then use the following):



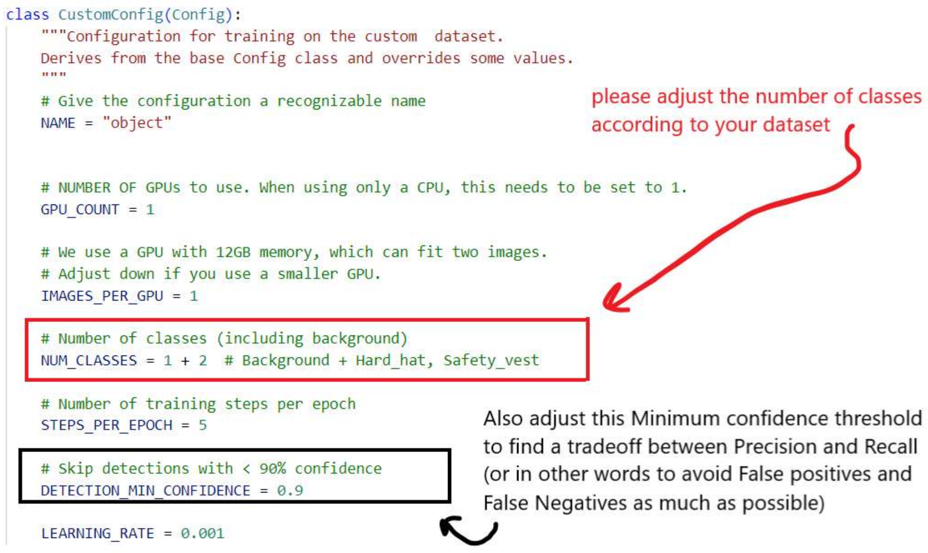
```
python.exe -m pip install -r requirements.txt
```

Step 4— Make sure to specify the correct paths (In custom.py) for the Root Directory, COCO weights and the logs Directory as shown below:



```python
import os
import sys
import json
import datetime
import numpy as np
import skimage.draw
import cv2
from mrcnn.visualize import display_instances
import matplotlib.pyplot as plt
import imgaug

# Root directory of the project
ROOT_DIR = "C:\\Users\\ForAI\\OneDrive\\Desktop\\DL task\\maskrcnn_implementation\\Mask-R-CNN-using-Tensorflow2"

# Import Mask RCNN
sys.path.append(ROOT_DIR)  # To find local version of the library
from mrcnn.config import Config
from mrcnn import model as modellib, utils

# Path to trained weights file
COCO_WEIGHTS_PATH = os.path.join(ROOT_DIR, "mask_rcnn_coco.h5")

# Directory to save logs and model checkpoints, if not provided
# through the command line argument --logs
DEFAULT_LOGS_DIR = os.path.join(ROOT_DIR, "logs")

```

Step 5—Next Tune the Following hyperparameters in custom.py:

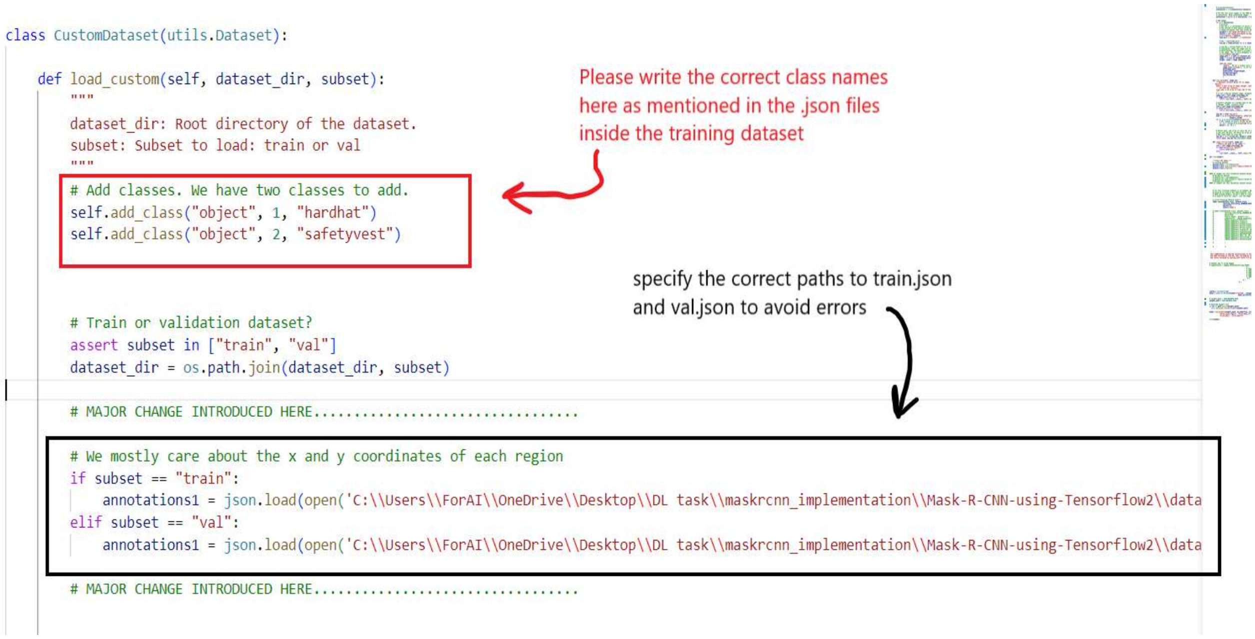


```python
class CustomConfig(Config):
    """Configuration for training on the custom  dataset.
    Derives from the base Config class and overrides some values.
    """
    # Give the configuration a recognizable name
    NAME = "object"


    # NUMBER OF GPUs to use. When using only a CPU, this needs to be set to 1.
    GPU_COUNT = 1

    # We use a GPU with 12GB memory, which can fit two images.
    # Adjust down if you use a smaller GPU.
    IMAGES_PER_GPU = 1

    # Number of classes (including background)
    NUM_CLASSES = 1 + 2  # Background + Hard_hat, Safety_vest

    # Number of training steps per epoch
    STEPS_PER_EPOCH = 5

    # Skip detections with < 90% confidence
    DETECTION_MIN_CONFIDENCE = 0.9

    LEARNING_RATE = 0.001
```

please adjust the number of classes according to your dataset

Also adjust this Minimum confidence threshold to find a tradeoff between Precision and Recall (or in other words to avoid False positives and False Negatives as much as possible)

Step 6—Specifying the Correct Class names and paths to .json files (custom.py):

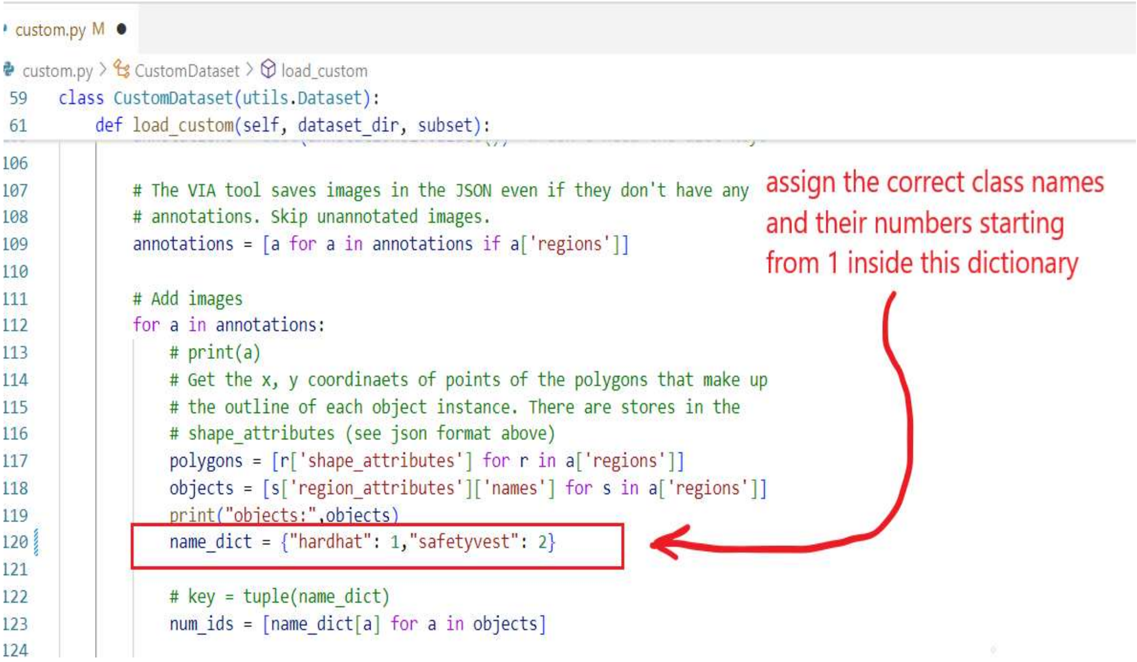


```
class CustomDataset(utils.Dataset):

    def load_custom(self, dataset_dir, subset):
        """
        dataset_dir: Root directory of the dataset.
        subset: Subset to load: train or val
        """
        # Add classes. We have two classes to add.
        self.add_class("object", 1, "hardhat")
        self.add_class("object", 2, "safetyvest")


        # Train or validation dataset?
        assert subset in ["train", "val"]
        dataset_dir = os.path.join(dataset_dir, subset)

        # MAJOR CHANGE INTRODUCED HERE...............................

        # We mostly care about the x and y coordinates of each region
        if subset == "train":
            annotations1 = json.load(open('C:\\Users\\ForAI\\OneDrive\\Desktop\\DL task\\maskrcnn_implementation\\Mask-R-CNN-using-Tensorflow2\\data
        elif subset == "val":
            annotations1 = json.load(open('C:\\Users\\ForAI\\OneDrive\\Desktop\\DL task\\maskrcnn_implementation\\Mask-R-CNN-using-Tensorflow2\\data

        # MAJOR CHANGE INTRODUCED HERE...............................
```

Please write the correct class names here as mentioned in the .json files inside the training dataset

specify the correct paths to train.json and val.json to avoid errors

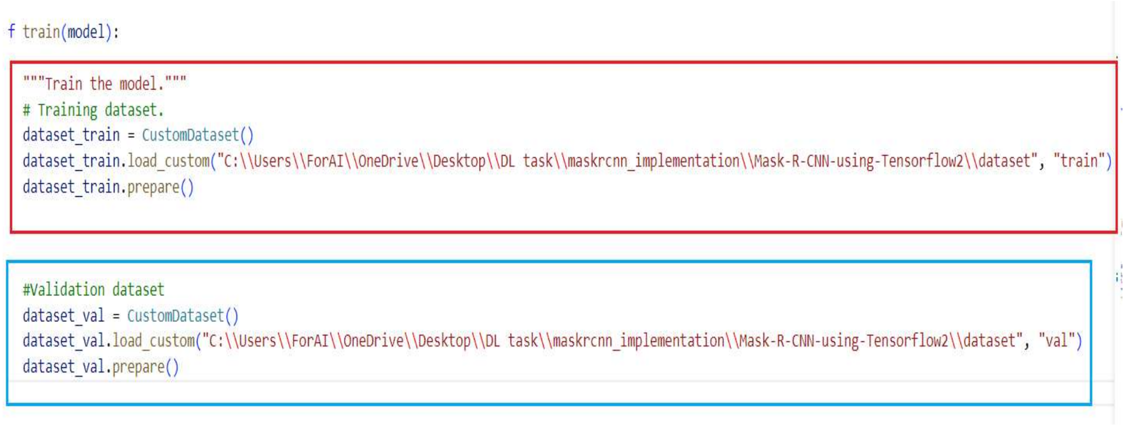
Step 7— Edit the correct Class Labels and numbers in the following dictionary as shown below (custom.py):



```
class CustomDataset(utils.Dataset):
    def load_custom(self, dataset_dir, subset):

        # The VIA tool saves images in the JSON even if they don't have any
        # annotations. Skip unannotated images.
        annotations = [a for a in annotations if a['regions']]

        # Add images
        for a in annotations:
            # print(a)
            # Get the x, y coordinaets of points of the polygons that make up
            # the outline of each object instance. There are stores in the
            # shape_attributes (see json format above)
            polygons = [r['shape_attributes'] for r in a['regions']]
            objects = [s['region_attributes']['names'] for s in a['regions']]
            print("objects:",objects)
            name_dict = {"hardhat": 1,"safetyvest": 2}

            # key = tuple(name_dict)
            num_ids = [name_dict[a] for a in objects]
```

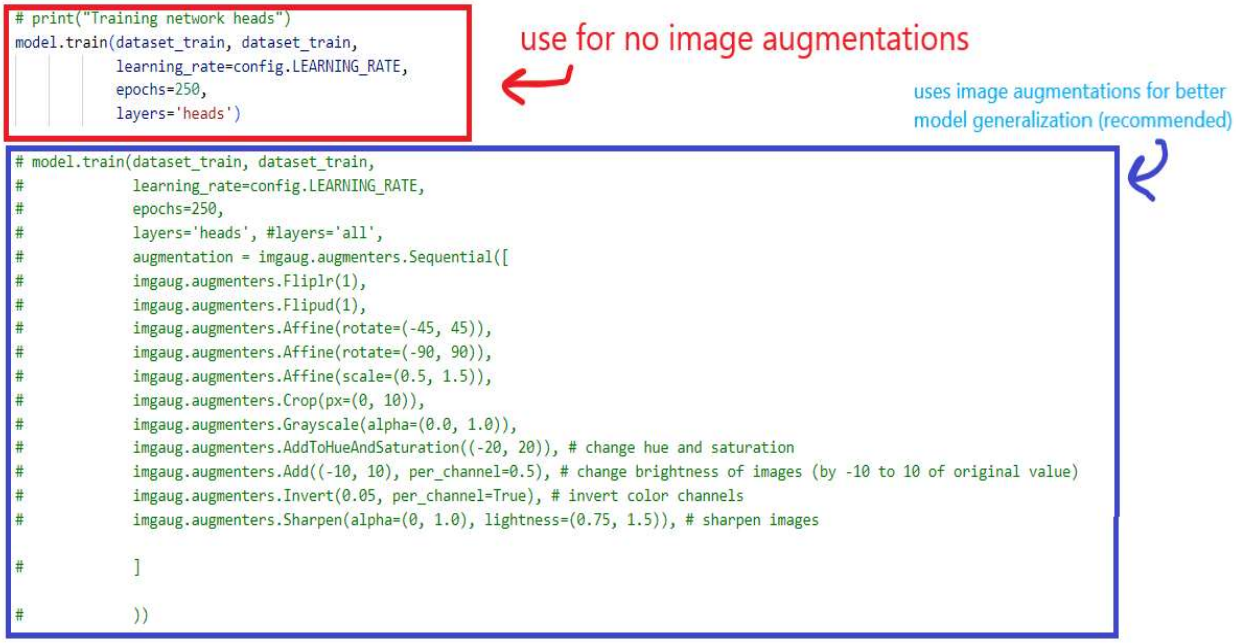
assign the correct class names and their numbers starting from 1 inside this dictionary

Step 8—Specify the correct path to train and val folders (custom.py) inside your dataset folder as shown below:



```
f train(model):
    """Train the model."""
    # Training dataset.
    dataset_train = CustomDataset()
    dataset_train.load_custom("C:\\Users\\ForAI\\OneDrive\\Desktop\\DL task\\maskrcnn_implementation\\Mask-R-CNN-using-Tensorflow2\\dataset", "train")
    dataset_train.prepare()

    #Validation dataset
    dataset_val = CustomDataset()
    dataset_val.load_custom("C:\\Users\\ForAI\\OneDrive\\Desktop\\DL task\\maskrcnn_implementation\\Mask-R-CNN-using-Tensorflow2\\dataset", "val")
    dataset_val.prepare()
```

Step 9— Use Image Augmentations for better generalization by uncommenting only one as shown below:

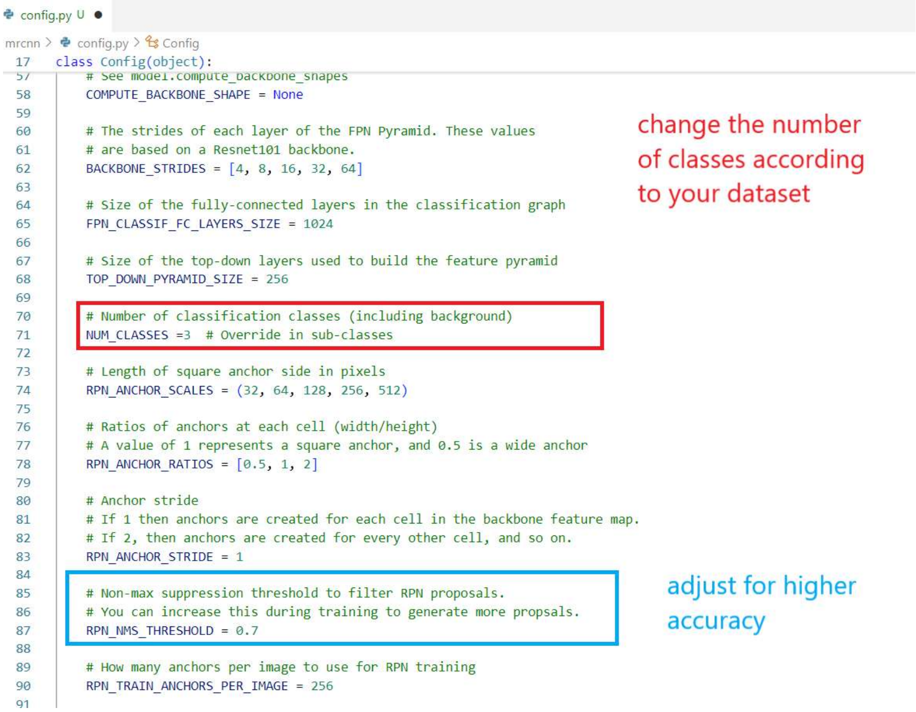


```
# print("Training network heads")
model.train(dataset_train, dataset_train,
            learning_rate=config.LEARNING_RATE,
            epochs=250,
            layers='heads')
# model.train(dataset_train, dataset_train,
#             learning_rate=config.LEARNING_RATE,
#             epochs=250,
#             layers='heads', #layers='all',
#             augmentation = imgaug.augmenters.Sequential([
#             imgaug.augmenters.Fliplr(1),
#             imgaug.augmenters.Flipud(1),
#             imgaug.augmenters.Affine(rotate=(-45, 45)),
#             imgaug.augmenters.Affine(rotate=(-90, 90)),
#             imgaug.augmenters.Affine(scale=(0.5, 1.5)),
#             imgaug.augmenters.Crop(px=(0, 10)),
#             imgaug.augmenters.Grayscale(alpha=(0.0, 1.0)),
#             imgaug.augmenters.AddToHueAndSaturation((-20, 20)), # change hue and saturation
#             imgaug.augmenters.Add((-10, 10), per_channel=0.5), # change brightness of images (by -10 to 10 of original value)
#             imgaug.augmenters.Invert(0.05, per_channel=True), # invert color channels
#             imgaug.augmenters.Sharpen(alpha=(0, 1.0), lightness=(0.75, 1.5)), # sharpen images

#             ]

#             ))
```

use for no image augmentations

uses image augmentations for better model generalization (recommended)

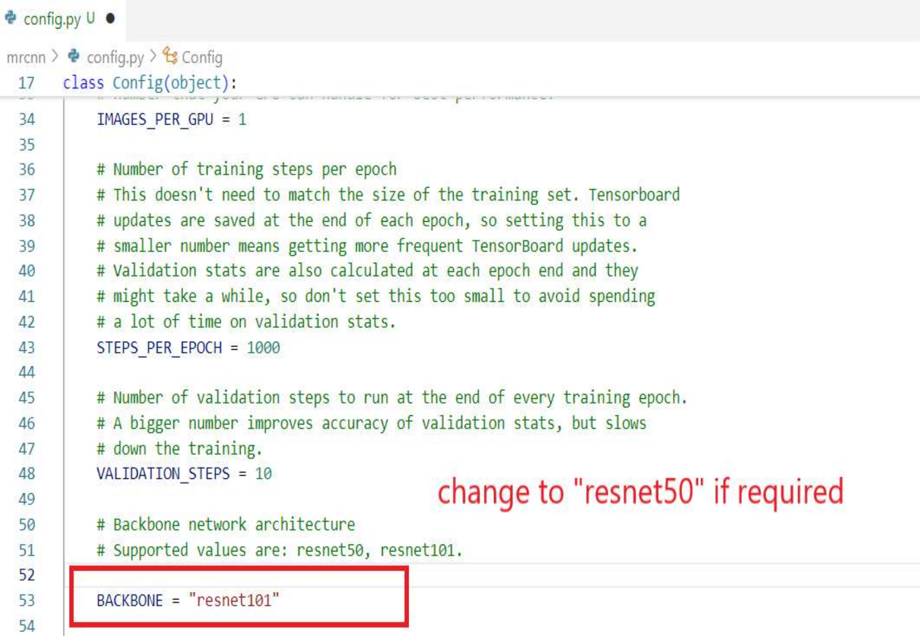
Step 10— Next we will adjust some hyperparameters and class names in config.py:



```python
class Config(object):
    # See model.compute_backbone_shapes
    COMPUTE_BACKBONE_SHAPE = None

    # The strides of each layer of the FPN Pyramid. These values
    # are based on a Resnet101 backbone.
    BACKBONE_STRIDES = [4, 8, 16, 32, 64]

    # Size of the fully-connected layers in the classification graph
    FPN_CLASSIF_FC_LAYERS_SIZE = 1024

    # Size of the top-down layers used to build the feature pyramid
    TOP_DOWN_PYRAMID_SIZE = 256

    # Number of classification classes (including background)
    NUM_CLASSES =3  # Override in sub-classes

    # Length of square anchor side in pixels
    RPN_ANCHOR_SCALES = (32, 64, 128, 256, 512)

    # Ratios of anchors at each cell (width/height)
    # A value of 1 represents a square anchor, and 0.5 is a wide anchor
    RPN_ANCHOR_RATIOS = [0.5, 1, 2]

    # Anchor stride
    # If 1 then anchors are created for each cell in the backbone feature map.
    # If 2, then anchors are created for every other cell, and so on.
    RPN_ANCHOR_STRIDE = 1

    # Non-max suppression threshold to filter RPN proposals.
    # You can increase this during training to generate more propsals.
    RPN_NMS_THRESHOLD = 0.7

    # How many anchors per image to use for RPN training
    RPN_TRAIN_ANCHORS_PER_IMAGE = 256
```

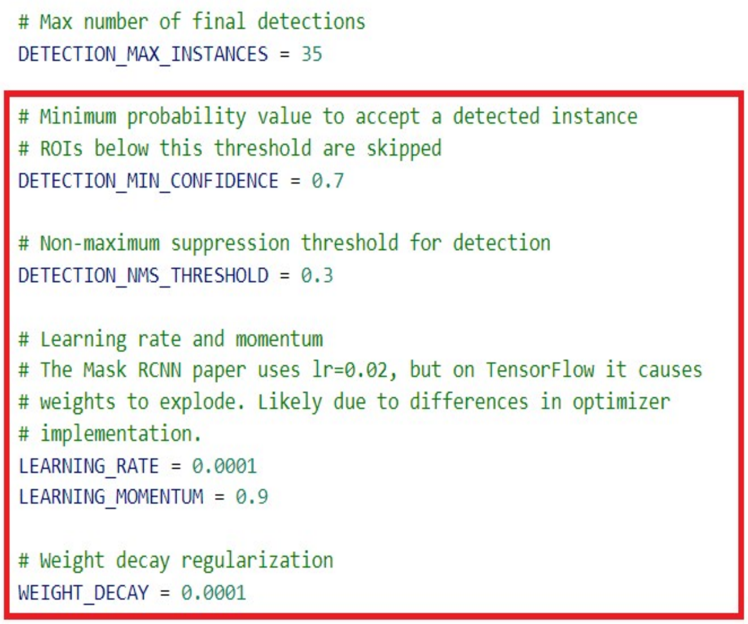
change the number of classes according to your dataset

adjust for higher accuracy



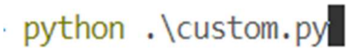
```python
class Config(object):
    # Number that your GPU can handle for best performance.
    IMAGES_PER_GPU = 1

    # Number of training steps per epoch
    # This doesn't need to match the size of the training set. Tensorboard
    # updates are saved at the end of each epoch, so setting this to a
    # smaller number means getting more frequent TensorBoard updates.
    # Validation stats are also calculated at each epoch end and they
    # might take a while, so don't set this too small to avoid spending
    # a lot of time on validation stats.
    STEPS_PER_EPOCH = 1000

    # Number of validation steps to run at the end of every training epoch.
    # A bigger number improves accuracy of validation stats, but slows
    # down the training.
    VALIDATION_STEPS = 10

    # Backbone network architecture
    # Supported values are: resnet50, resnet101.

    BACKBONE = "resnet101"
```

change to "resnet50" if required



```
# Max number of final detections
DETECTION_MAX_INSTANCES = 35

# Minimum probability value to accept a detected instance
# ROIs below this threshold are skipped
DETECTION_MIN_CONFIDENCE = 0.7

# Non-maximum suppression threshold for detection
DETECTION_NMS_THRESHOLD = 0.3

# Learning rate and momentum
# The Mask RCNN paper uses lr=0.02, but on TensorFlow it causes
# weights to explode. Likely due to differences in optimizer
# implementation.
LEARNING_RATE = 0.0001
LEARNING_MOMENTUM = 0.9

# Weight decay regularization
WEIGHT_DECAY = 0.0001
```

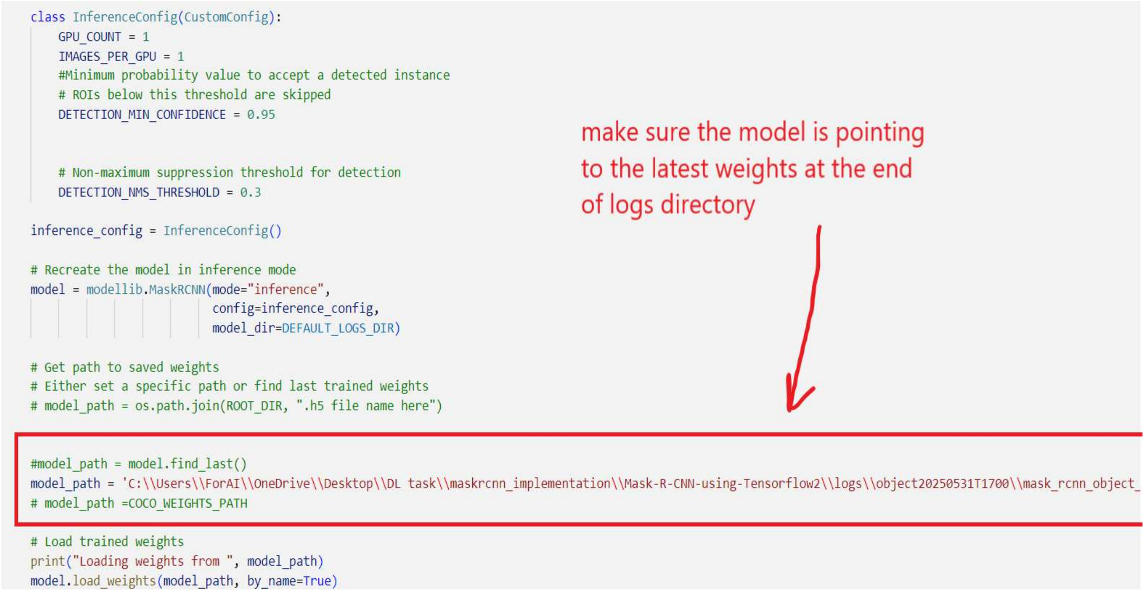
Step 11— Now run the training script with the following command:



```
python .\custom.py
```

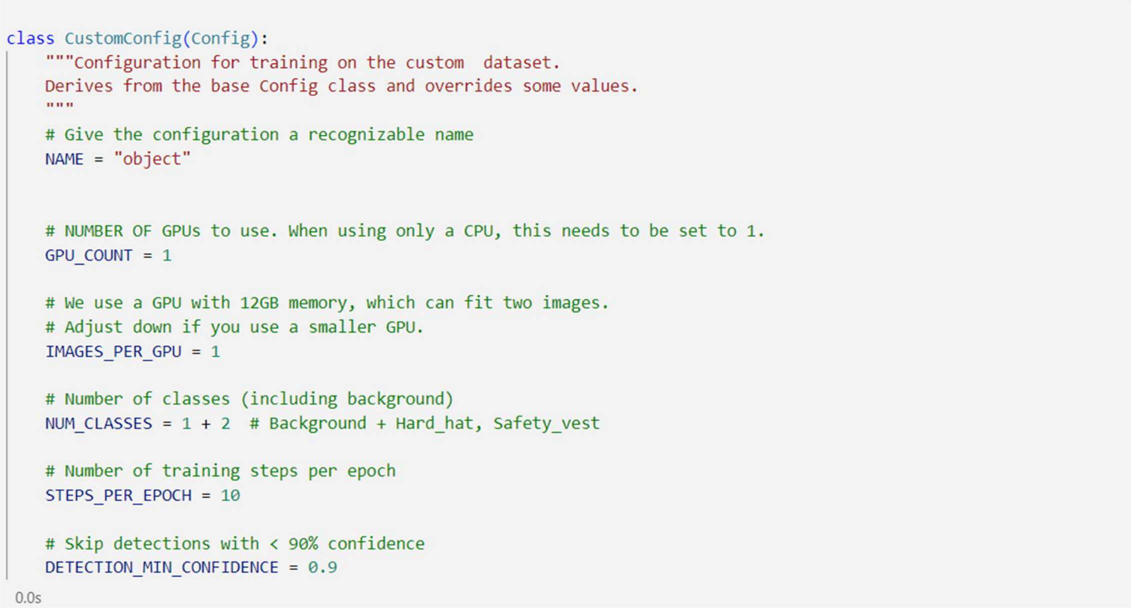
Step 12—Now execute the “test_model.ipynb” jupyter notebook to test our model:

(a) change the model path in Inference class:



```
class InferenceConfig(CustomConfig):
    GPU_COUNT = 1
    IMAGES_PER_GPU = 1
    #Minimum probability value to accept a detected instance
    # ROIs below this threshold are skipped
    DETECTION_MIN_CONFIDENCE = 0.95


    # Non-maximum suppression threshold for detection
    DETECTION_NMS_THRESHOLD = 0.3

inference_config = InferenceConfig()

# Recreate the model in inference mode
model = modellib.MaskRCNN(mode="inference",
                          config=inference_config,
                          model_dir=DEFAULT_LOGS_DIR)

# Get path to saved weights
# Either set a specific path or find last trained weights
# model_path = os.path.join(ROOT_DIR, ".h5 file name here")

#model_path = model.find_last()
model_path = 'C:\\Users\\ForAI\\OneDrive\\Desktop\\DL task\\maskrcnn_implementation\\Mask-R-CNN-using-Tensorflow2\\logs\\object20250531T1700\\mask_rcnn_object_
# model_path =COCO_WEIGHTS_PATH

# Load trained weights
print("Loading weights from ", model_path)
model.load_weights(model_path, by_name=True)
```

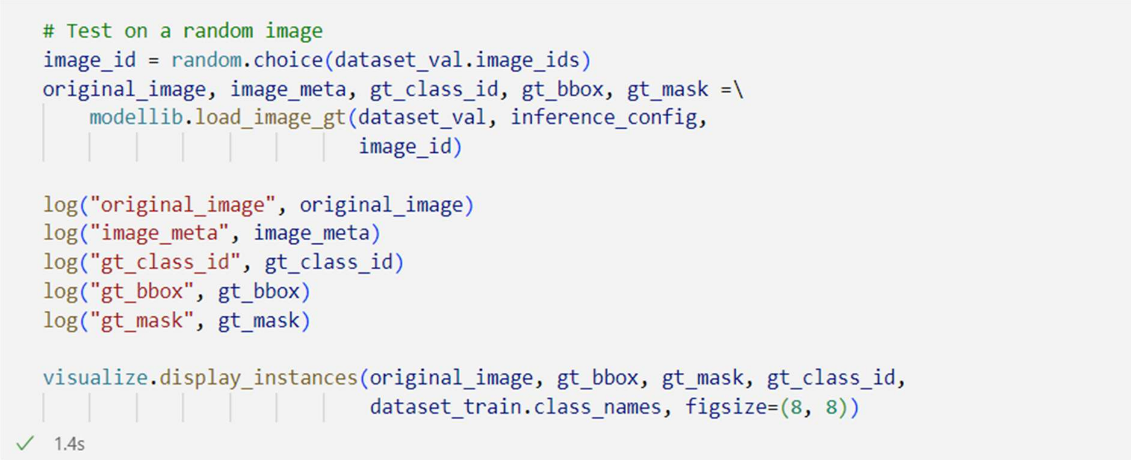
make sure the model is pointing to the latest weights at the end of logs directory *(b)* *adjust the number of classes:*



```
class CustomConfig(Config):
    """Configuration for training on the custom  dataset.
    Derives from the base Config class and overrides some values.
    """
    # Give the configuration a recognizable name
    NAME = "object"


    # NUMBER OF GPUs to use. When using only a CPU, this needs to be set to 1.
    GPU_COUNT = 1

    # We use a GPU with 12GB memory, which can fit two images.
    # Adjust down if you use a smaller GPU.
    IMAGES_PER_GPU = 1

    # Number of classes (including background)
    NUM_CLASSES = 1 + 2  # Background + Hard_hat, Safety_vest

    # Number of training steps per epoch
    STEPS_PER_EPOCH = 10

    # Skip detections with < 90% confidence
    DETECTION_MIN_CONFIDENCE = 0.9
```

0.0s

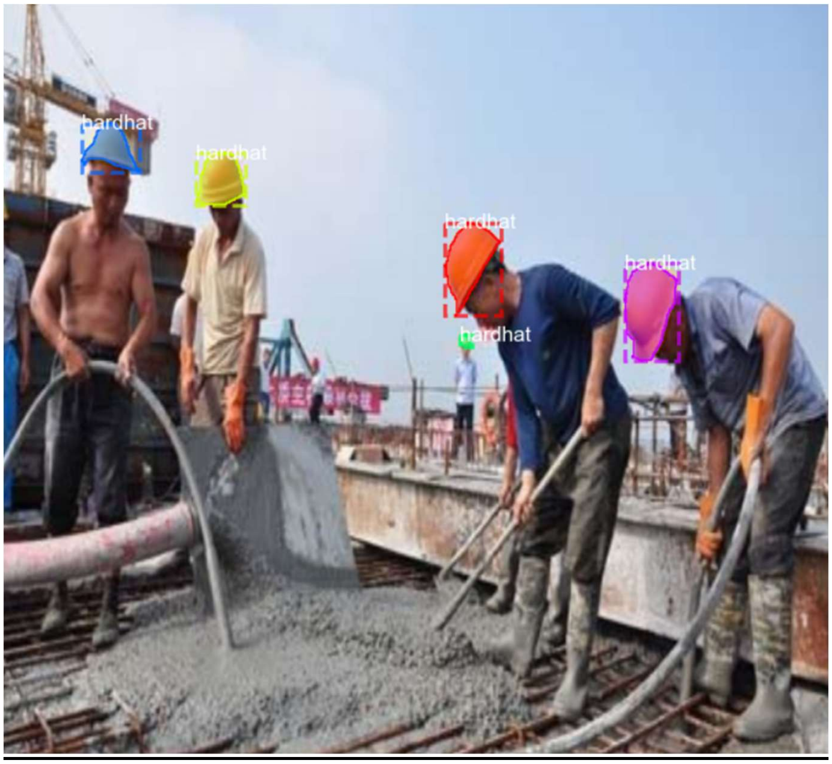
(c) *Test on a random image:*



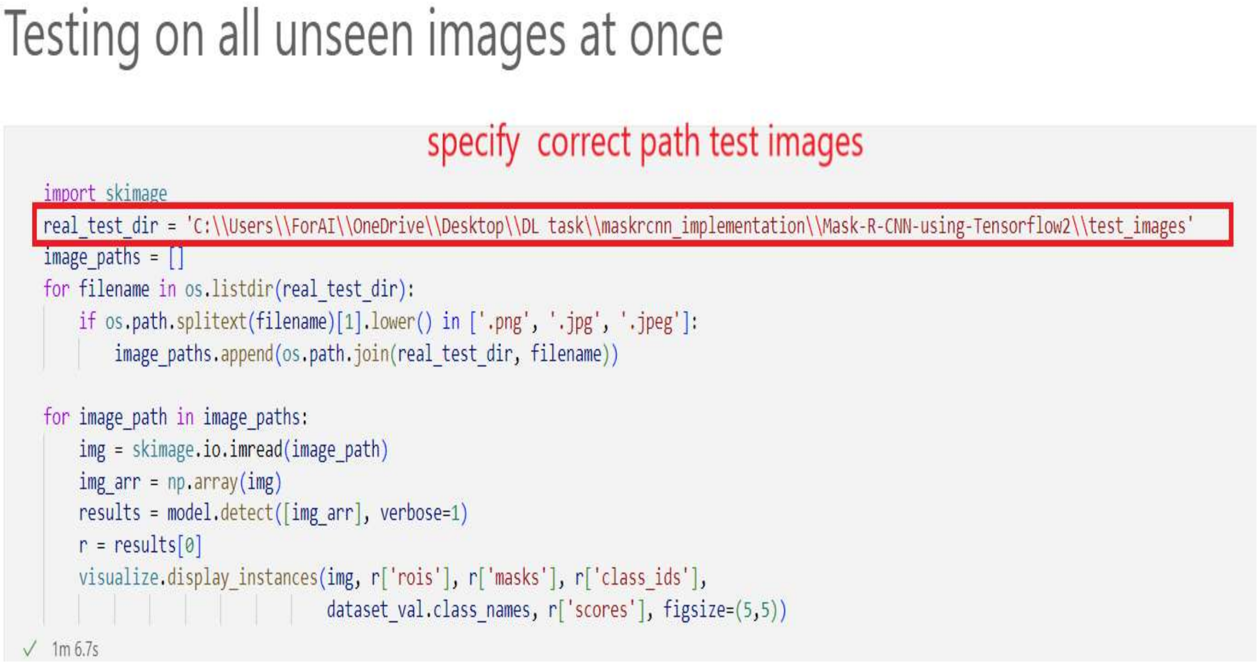
```
# Test on a random image
image_id = random.choice(dataset_val.image_ids)
original_image, image_meta, gt_class_id, gt_bbox, gt_mask =\
    modellib.load_image_gt(dataset_val, inference_config,
                           image_id)

log("original_image", original_image)
log("image_meta", image_meta)
log("gt_class_id", gt_class_id)
log("gt_bbox", gt_bbox)
log("gt_mask", gt_mask)

visualize.display_instances(original_image, gt_bbox, gt_mask, gt_class_id,
                            dataset_train.class_names, figsize=(8, 8))
```

✓ 1.4s

<u>Results:</u>



Step 13—Test your model on Unseen images:



Testing on all unseen images at once

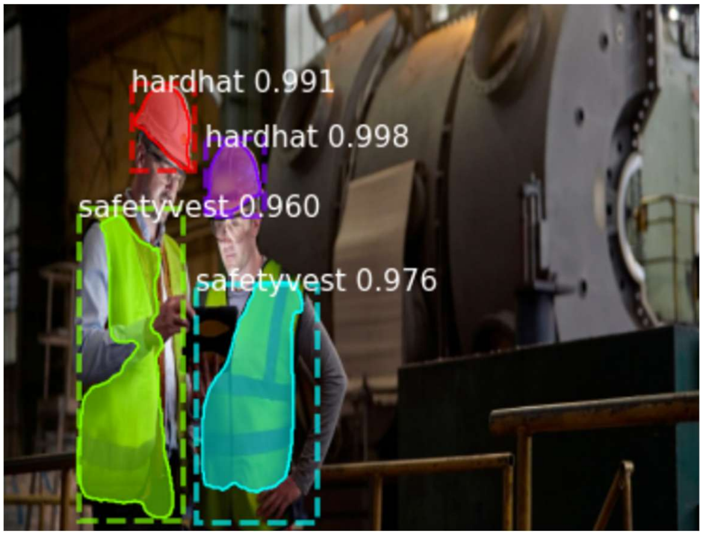
specify correct path test images

```
import skimage
real_test_dir = 'C:\\Users\\ForAI\\OneDrive\\Desktop\\DL task\\maskrcnn_implementation\\Mask-R-CNN-using-Tensorflow2\\test_images'
image_paths = []
for filename in os.listdir(real_test_dir):
    if os.path.splitext(filename)[1].lower() in ['.png', '.jpg', '.jpeg']:
        image_paths.append(os.path.join(real_test_dir, filename))

for image_path in image_paths:
    img = skimage.io.imread(image_path)
    img_arr = np.array(img)
    results = model.detect([img_arr], verbose=1)
    r = results[0]
    visualize.display_instances(img, r['rois'], r['masks'], r['class_ids'],
                                dataset_val.class_names, r['scores'], figsize=(5,5))
```

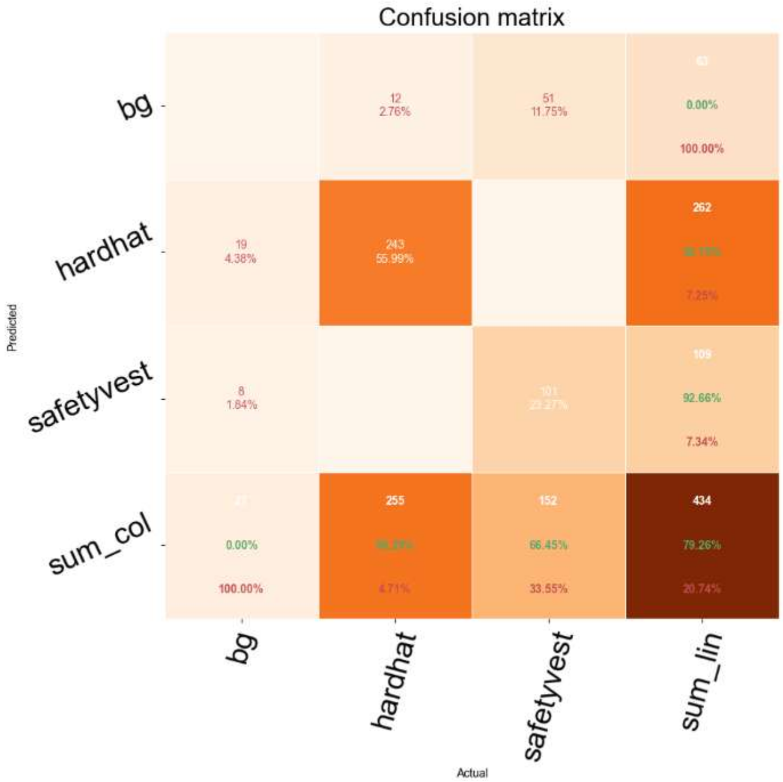
✓ 1m 6.7s

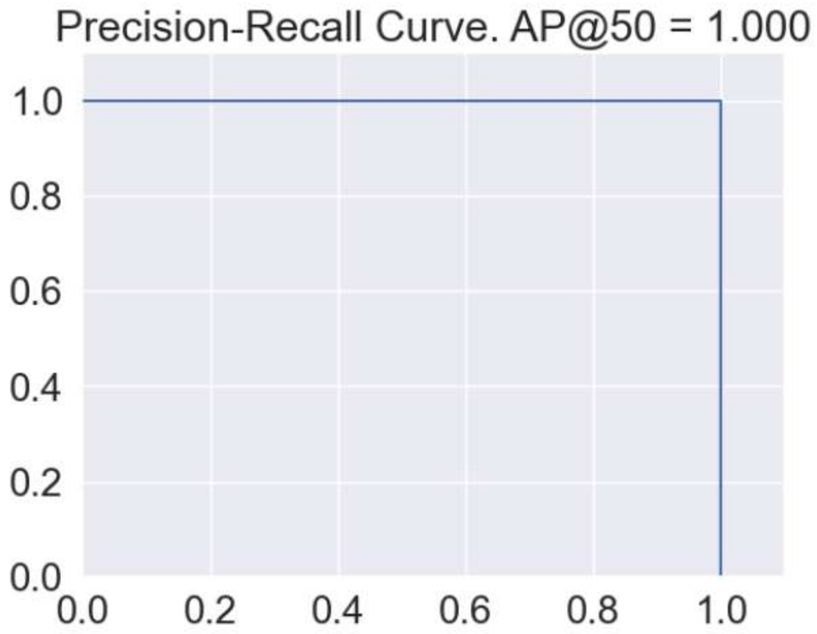
Step 14—Verify the accuracy of your model by Test results and plots:

(a) *Test results:*



(b) *Confusion Matrix and Precision-Recall curve:*




Precision-Recall Curve. AP@50 = 1.000
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0