

# 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 mask_rcnn_new
```

**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:

```
custom.py > ...
1  import os
2  import sys
3  import json
4  import datetime
5  import numpy as np
6  import skimage.draw
7  import cv2
8  from mrcnn.visualize import display_instances
9  import matplotlib.pyplot as plt
10 import imgaug
11
12 # Root directory of the project
13 ROOT_DIR = "C:\\Users\\ForAI\\OneDrive\\Desktop\\DL task\\maskrcnn_implementation\\Mask-R-CNN-using-Tensorflow2"
14
15 # Import Mask RCNN
16 sys.path.append(ROOT_DIR) # To find local version of the library
17 from mrcnn.config import Config
18 from mrcnn import model as modellib, utils
19
20 # Path to trained weights file
21 COCO_WEIGHTS_PATH = os.path.join(ROOT_DIR, "mask_rcnn_coco.h5")
22
23 # Directory to save logs and model checkpoints, if not provided
24 # through the command line argument --logs
25 DEFAULT_LOGS_DIR = os.path.join(ROOT_DIR, "logs")
26
27
```

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

```
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):

```
custom.py M
custom.py > CustomDataset > load_custom
59 class CustomDataset(utils.Dataset):
61     def load_custom(self, dataset_dir, subset):

106
107     # The VIA tool saves images in the JSON even if they don't have any
108     # annotations. Skip unannotated images.
109     annotations = [a for a in annotations if a['regions']]
110
111     # Add images
112     for a in annotations:
113         # print(a)
114         # Get the x, y coordinaets of points of the polygons that make up
115         # the outline of each object instance. There are stores in the
116         # shape_attributes (see json format above)
117         polygons = [r['shape_attributes'] for r in a['regions']]
118         objects = [s['region_attributes']['names'] for s in a['regions']]
119         print("objects:",objects)
120         name_dict = {"hardhat": 1,"safetyvest": 2}
121
122         # key = tuple(name_dict)
123         num_ids = [name_dict[a] for a in objects]
124
```

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')
```

use for no image augmentations

uses image augmentations for better  
model generalization (recommended)

```
# 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  
#             ]  
#             ))
```

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

```
config.py U •
mrcnn > config.py > Config
17 class Config(object):
18     # see model.compute_backbone_shapes
19     COMPUTE_BACKBONE_SHAPE = None
20
21     # The strides of each layer of the FPN Pyramid. These values
22     # are based on a Resnet101 backbone.
23     BACKBONE_STRIDES = [4, 8, 16, 32, 64]
24
25     # Size of the fully-connected layers in the classification graph
26     FPN_CLASSIF_FC_LAYERS_SIZE = 1024
27
28     # Size of the top-down layers used to build the feature pyramid
29     TOP_DOWN_PYRAMID_SIZE = 256
30
31     # Number of classification classes (including background)
32     NUM_CLASSES = 3 # Override in sub-classes
33
34     # Length of square anchor side in pixels
35     RPN_ANCHOR_SCALES = (32, 64, 128, 256, 512)
36
37     # Ratios of anchors at each cell (width/height)
38     # A value of 1 represents a square anchor, and 0.5 is a wide anchor
39     RPN_ANCHOR_RATIOS = [0.5, 1, 1, 2]
40
41     # Anchor stride
42     # If 1 then anchors are created for each cell in the backbone feature map.
43     # If 2, then anchors are created for every other cell, and so on.
44     RPN_ANCHOR_STRIDE = 1
45
46     # Non-max suppression threshold to filter RPN proposals.
47     # You can increase this during training to generate more proposals.
48     RPN_NMS_THRESHOLD = 0.7
49
50     # How many anchors per image to use for RPN training
51     RPN_TRAIN_ANCHORS_PER_IMAGE = 256
52
53
54
```

change the number of classes according to your dataset

adjust for higher accuracy

```
config.py U •
mrcnn > config.py > Config
17 class Config(object):
18
19     IMAGES_PER_GPU = 1
20
21     # Number of training steps per epoch
22     # This doesn't need to match the size of the training set. Tensorboard
23     # updates are saved at the end of each epoch, so setting this to a
24     # smaller number means getting more frequent TensorBoard updates.
25     # Validation stats are also calculated at each epoch end and they
26     # might take a while, so don't set this too small to avoid spending
27     # a lot of time on validation stats.
28     STEPS_PER_EPOCH = 1000
29
30     # Number of validation steps to run at the end of every training epoch.
31     # A bigger number improves accuracy of validation stats, but slows
32     # down the training.
33     VALIDATION_STEPS = 10
34
35     # Backbone network architecture
36     # Supported values are: resnet50, resnet101.
37     BACKBONE = "resnet101"
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
```

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



### Results:



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))
```

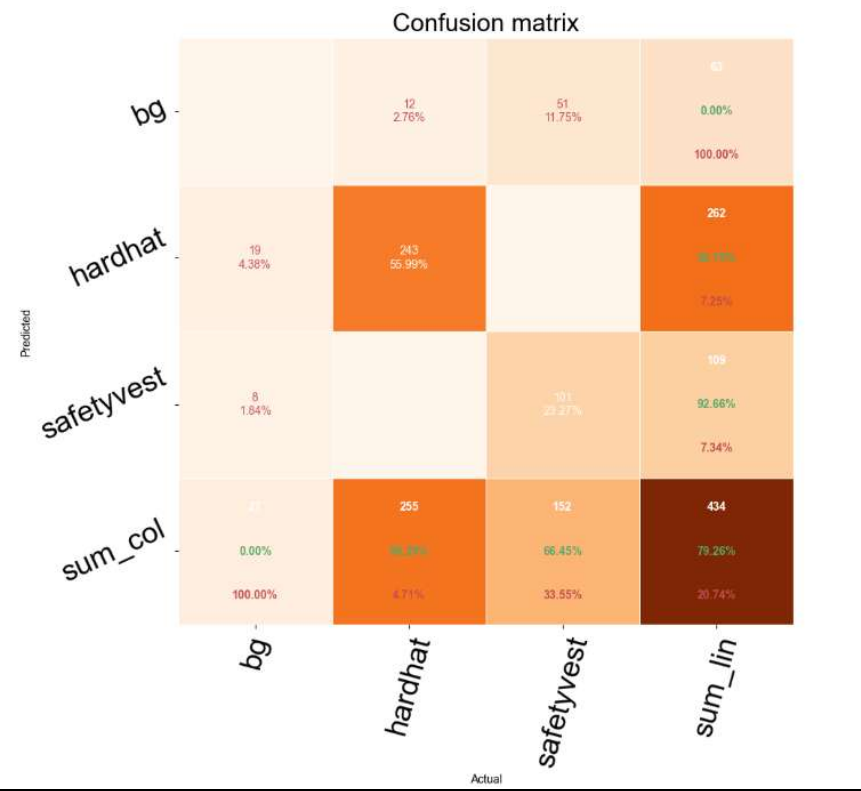
✓ 1m 6.7s

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

