



HOW TO TRAIN YOLOv4

OBJECT DETECTION
MODULE 5 – TRAINING YOLOv4

COURSE

YOLOv4 Training- Step 1

Change the Configuration Files

- In the darknet folder, we will go to file location **"build/darknet/x64/cfg/"**, in this folder we have to copy the contents of the **"yolo-custom.cfg"** file to a new file named as **"yolo-obj.cfg"**.
- In our custom object detection, we will have **two classes**. (*Mask and Without_mask*)
- Therefore, in the new "yolo-obj.cfg" file we make the following changes:
 - First, we will change the variable `max_batches=2000 x no_of_classes` , in our case **max_batches=4000**.
 - The steps variable will be equal to 80-90% of max_batches , in our **case steps=3200,3600**.
 - The height and width parameters will be in multiples of 32 , here in the file we will change it to **height=416** and **width=416** .
 - Now, we will have to change the classes parameter at the respective line number 970, 1058, 1146 to 2 , since we have only **2 classes** (mask, without_mask).
 - Now similarly, update the filters parameter to **filters=(classes + 5)*3** in the 3 convolutional layers before each yolo layer. In this case classes = 2 so, set the **filters = 21** in the lines 963, 1051, 1139 .

(Don't write filters = (classes + 5) x 3 in the .cfg file)

```
sign>  
sign>  
sign>  
sign>pip install labeling_
```

YOLOv4 Training- Step 2

Creating the obj.names files

- Now, after doing the above changes successfully, we will create another file name **obj.names** in the **build/darknet/x64/data/** folder .

This file will have the names of classes i.e. in the first line **Mask** and in the second line **Without_mask** .

```
sign>  
sign>  
sign>  
sign>  
sign>pip install labeling_
```



YOLOv4 Training- Step 3

Importing Images and Annotation Files

Now, we will create a folder inside **build/darknet/x64/data/** with the name **obj**.

Inside this folder(**build/darknet/x64/data/obj/**) we have to copy-paste all the dataset images and annotations for all images that we have created using Labellmg/Darknet software or custom downloaded datasets.



YOLOv4 Training- Step 4

Metadata Files for train.txt and test.txt files

- Now, we have to create two files **train.txt** and **test.txt**, these files will contain the address to every train and test images as these will be needed while training.

- To create this file only a python script is to be executed and the data will be distributed in training and testing data in the ratio of **80%** and **20%** respectively.

- In the python script, the **imgspath variable** consists of the path of the folder where images are stored. The **path variable** is the **path of images in the obj folder** relative to the darknet.exe file in the x64 folder. The train.txt and test.txt will have the **relative path of each image** of training and testing and so this path should be appended with each image file name and kept in .txt files.

- The python script should be kept and executed in the **build/darknet/x64/data** folder so that the **.txt files** directly get generated in that folder.

```
imgspath = 'build/darknet/x64/data/obj/'  
path = 'data/obj/'
```

- The above is the example of a file path to be set in the python script if any issue is there you can also check in your own system and change the path accordingly.

- To run the script we simply type

```
python file_separate.py
```

and press enter. And now we should have our list in the train and test text files.

```
sign>  
sign>  
sign>  
sign>pip install labeling_
```

YOLOv4 Training- Step 5.1

Metadata files for Training

Now, we need to create a file **obj.data** in the **build/darknet/x64/data** folder. This file will be as follows:

- classes=2
- train = data/train.txt
- test = data/test.txt
- names = data/obj.names
- backup = backup/

The above 5 lines need to be written in the **obj.data** file and stored in the aforementioned folder.

```
sign>  
sign>  
sign>  
sign>  
sign>pip install labeling_
```



YOLOv4 Training- Step 5.2

Metadata Files for train.txt and test.txt files

- Remember all paths are relative to the x64 folder as it contains the darknet.exe application file which will be used to perform training and detection.
- After every 100 iterations, a file will be created in the **build/darknet/x64/backup/** folder of the weights of the last 100 iterations. (**yolo-obj_last.weights**)
- After every **1000 iterations**, a file will be created **yolo-obj_xxxx.weights**. This file can be used to test for better accuracy. After 1000 iterations the training can be stopped and so, we will have the trained weights for those 1000 iterations. If needed we can also run for more iterations as per a person's choice.
- We are trying to achieve custom object detection and so we need the **yolo4.conv.137 weights** file to train our model. The link found in Downloads Section
- The file should be stored in the **build/darknet/x64/** folder.

```
sign>  
sign>  
sign>  
sign>pip install labeling_
```




YOLOv4 Training- Step 5.3

Command to Train YOLOv4

We need to go inside the x64 folder where darknet.exe is placed, the command is as follows:

```
darknet.exe train data/obj.data cfg/yolo-obj.cfg  
yolo4.conv.137
```

A black circular graphic representing a terminal window. It contains white text showing a command prompt 'eign>' followed by 'pip install labeling_'.

```
eign>  
eign>  
eign>  
eign>pip install labeling_
```

