

How to train YOLOv3 using Darknet on Colab notebook

Để train được cho Yolo nhận diện các đối tượng đặc thù theo yêu cầu, chúng ta sẽ cần làm các bước lớn sau.

1. Tải source code Darknet – Yolo về máy tính, chỉnh tham số và tiến hành biên dịch (make) source code đó ra file thực thi tùy theo hệ điều hành (window thì là exe, macos với linux thì file bash thì phải, tóm lại là file chạy được)

2. Chuẩn bị dữ liệu train: Hình ảnh của đối tượng bạn định train. Ví dụ như bài này là súng ngắn.

3. Gán nhãn cho dữ liệu: Cụ thể là với từng ảnh trong dữ liệu, chúng ta sẽ gán nhãn cho máy biết đâu là đối tượng cần nhận dạng bằng cách vẽ một hình chữ nhật xung quanh đối tượng đó. Cái này có tool nhé.

4. Tạo các file cần thiết để phục vụ quá trình train, chỉnh sửa tham số train trong file cấu hình Yolo.

5. Chạy lệnh train và ngồi uống cafe đợi.

6. Tận hưởng thành quả bằng cách detect thử một ảnh sample.

A. Cài đặt Darknet on Colab và test thử kết quả

Theo hướng dẫn sau:

<https://colab.research.google.com/drive/1WsyEMeIgl02sWei0KmDtnfJywkYuERqQ>

B. Cách train Yolo để detect các object đặc thù trên Colab

Bước 1. Chuẩn bị dữ liệu train

Bước 2. Gán nhãn cho dữ liệu

Có 02 công cụ:

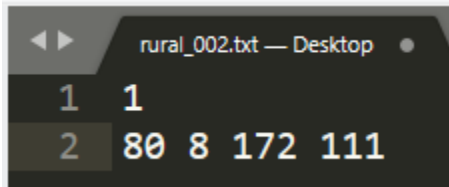
*Nếu sử dụng công cụ **Bbox-Label-Tool**:*

Chú ý chọn để cho format là Yolo chứ ko phải là Pascal VOC nhé.

The format of annotations generated by BBox-Label-Tool is:
--

```
class_number
box1_x1 box1_y1 box1_width box1_height
box2_x1 box2_y1 box2_width box2_height
```

ví dụ:

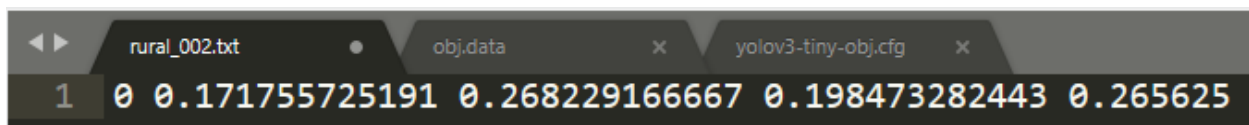


```
rural_002.txt — Desktop
1 1
2 80 8 172 111
```

After conversion, the format of annotations converted by [scripts/convert.py](#) is:

```
class_number box1_x1_ratio box1_y1_ratio box1_width_ratio box1_height_ratio
class_number box2_x1_ratio box2_y1_ratio box2_width_ratio box2_height_ratio
```

ví dụ:



```
rural_002.txt  obj.data  yolov3-tiny-obj.cfg
1 0 0.171755725191 0.268229166667 0.198473282443 0.265625
```

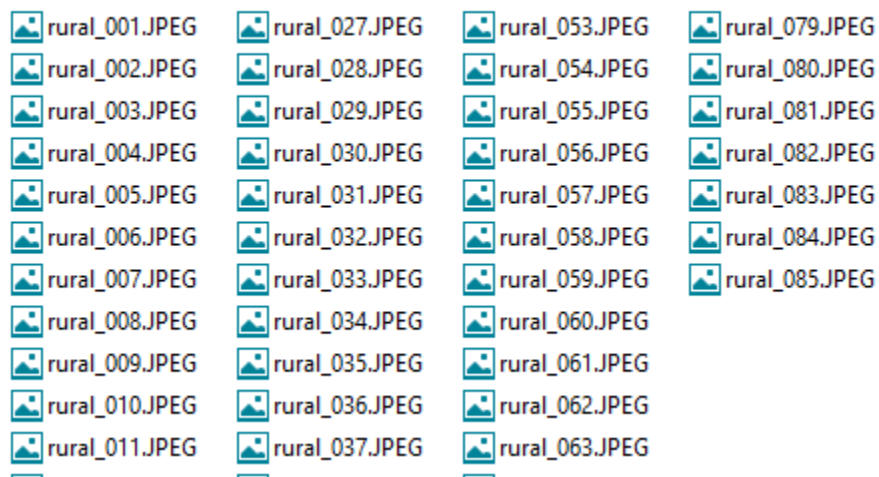
Sau khi thực hiện gán nhãn chúng ta có kết quả sau:

Trong thư mục images:

stopsign
yieldsign

Trong thư mục stopsign

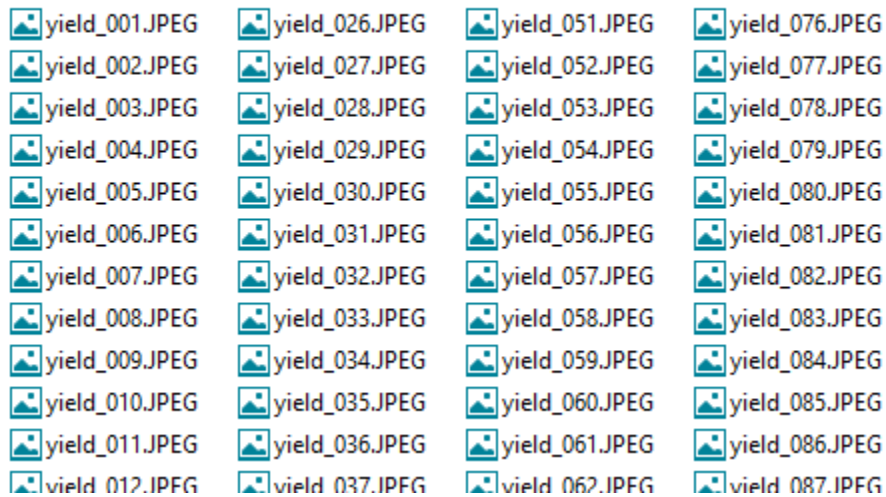
loads ▶ data ▶ images ▶ stopsign



rural_001.JPG	rural_027.JPG	rural_053.JPG	rural_079.JPG
rural_002.JPG	rural_028.JPG	rural_054.JPG	rural_080.JPG
rural_003.JPG	rural_029.JPG	rural_055.JPG	rural_081.JPG
rural_004.JPG	rural_030.JPG	rural_056.JPG	rural_082.JPG
rural_005.JPG	rural_031.JPG	rural_057.JPG	rural_083.JPG
rural_006.JPG	rural_032.JPG	rural_058.JPG	rural_084.JPG
rural_007.JPG	rural_033.JPG	rural_059.JPG	rural_085.JPG
rural_008.JPG	rural_034.JPG	rural_060.JPG	
rural_009.JPG	rural_035.JPG	rural_061.JPG	
rural_010.JPG	rural_036.JPG	rural_062.JPG	
rural_011.JPG	rural_037.JPG	rural_063.JPG	

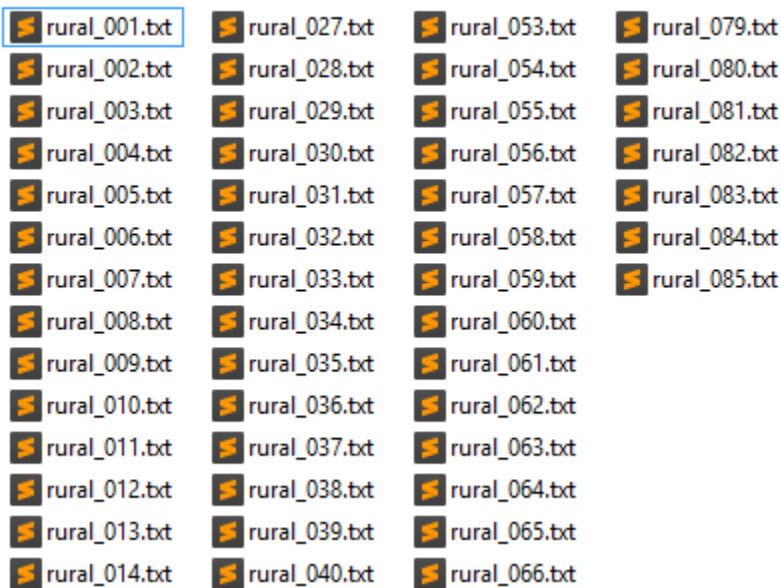
Trong thư mục yieldsign

ploads ▶ data ▶ images ▶ yieldsign

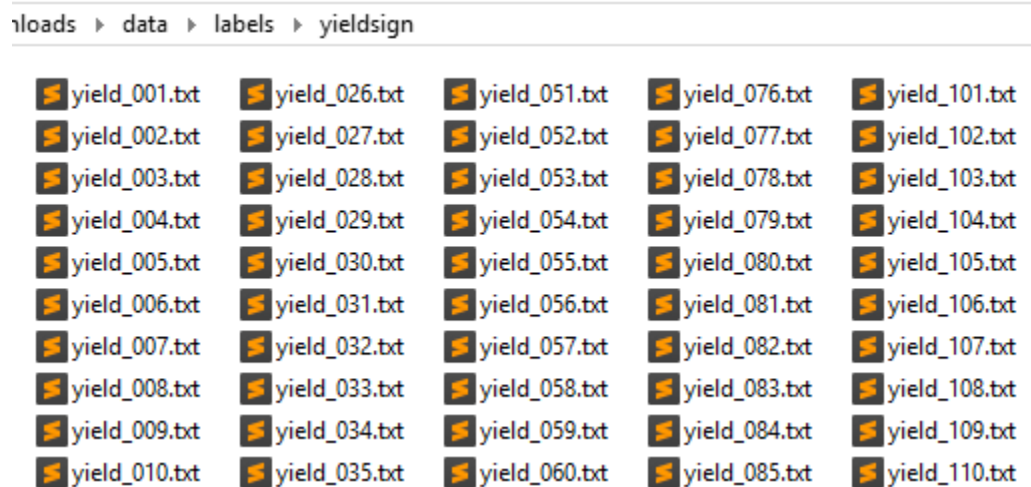


Tương ứng trong thư mục labels cũng có 02 thư mục như thư mục images:

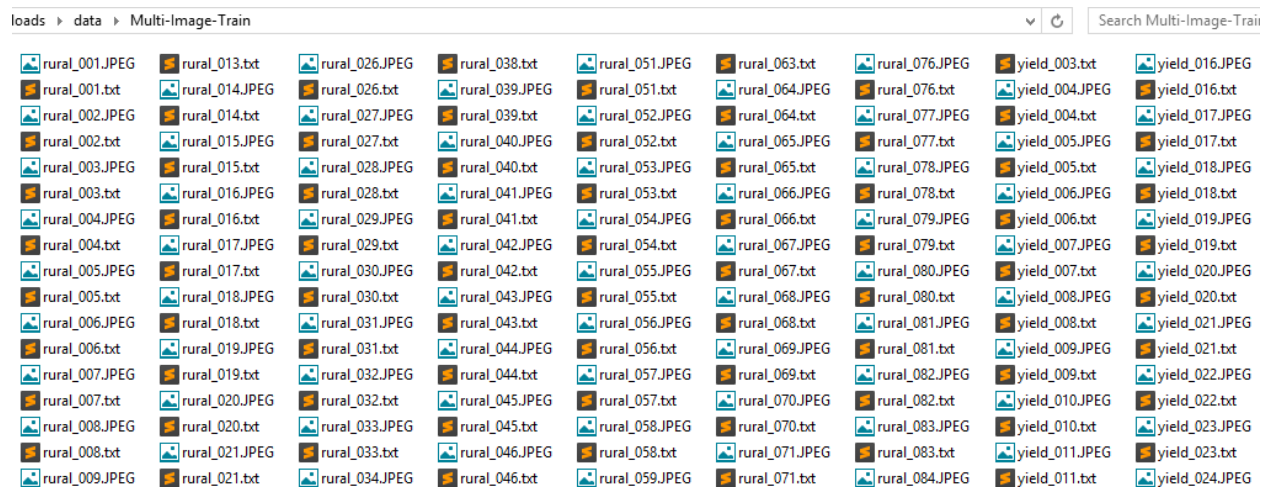
ploads ▶ data ▶ labels ▶ stopsign



Thư mục yieldsign



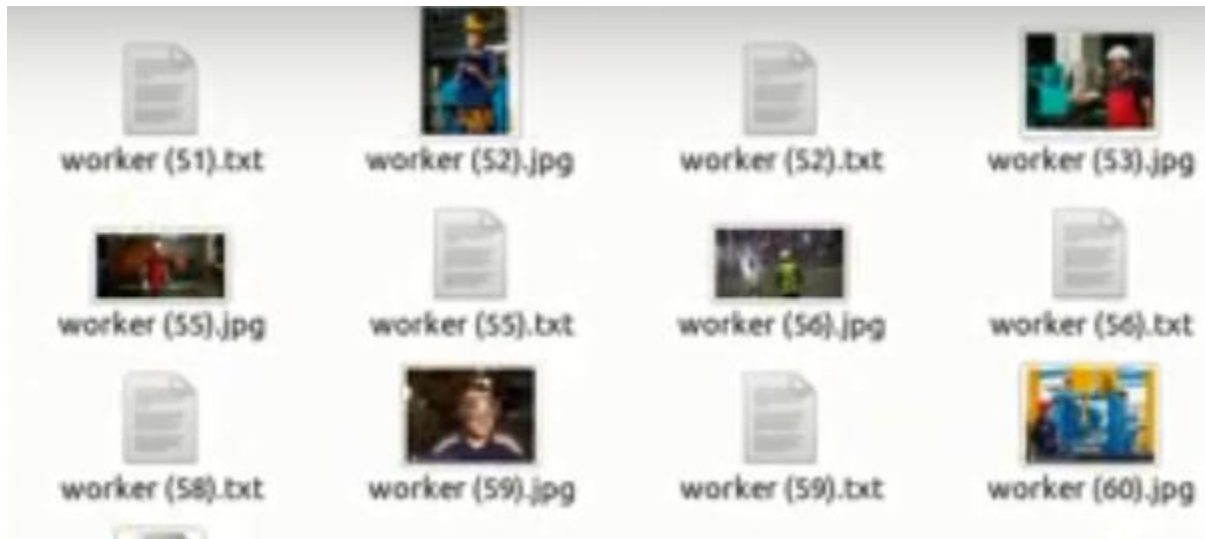
Để chuẩn bị dữ liệu (tách ra tập train.txt và test.txt) chúng ta cần tạo một thư mục chép tất cả hình ảnh, label vào:



Nếu sử dụng công cụ **LabelImg**:

Đối với công cụ **LabelImg** (chúng ta không cần mất thời gian như trên nhé). Chép tất cả hình ảnh vào một thư mục (dù bạn có nhiều nhãn: *stopsign, yieldsign, ...*).

Khi sử dụng công cụ này sẽ tạo ra thư mục như hình trên (kaka...)



Hướng dẫn chi tiết: <https://github.com/tzutalin/labelImg>.

Bước 4: Tạo file train.txt và test.txt

Khi chúng ta có thư mục như trên, tiến hành phân chia thành 02 tập train.txt và test.txt để chuẩn bị huấn luyện.

Đây là code để tách ra 02 file được chạy trên colab:

```
[ ] import glob, os

# Current directory

current_dir = os.getcwd()+"/Multi-Image-Train"
print(current_dir)

/ content/gdrive/My Drive/app/Darknet_colab/Multi-Image-Train

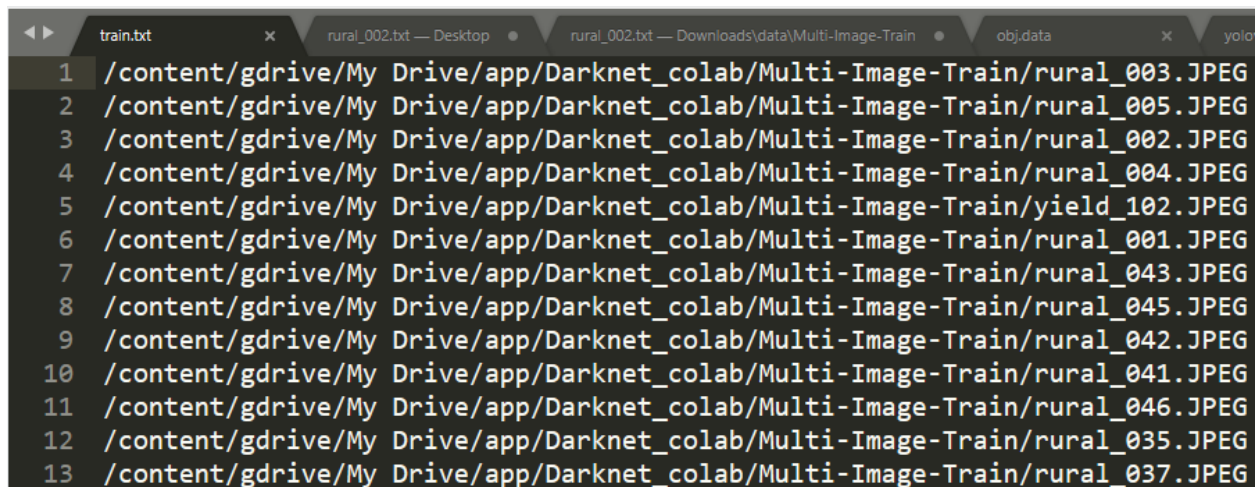
[ ] # Percentage of images to be used for the test set
percentage_test = 10;

# Create and/or truncate train.txt and test.txt
file_train = open('train.txt', 'w')
file_test = open('test.txt', 'w')

# Populate train.txt and test.txt
counter = 1
index_test = round(100 / percentage_test)
for pathAndFilename in glob.iglob(os.path.join(current_dir, "*.JPEG")):
    print(pathAndFilename)
    title, ext = os.path.splitext(os.path.basename(pathAndFilename))

    if counter == index_test:
        counter = 1
        file_test.write(current_dir + "/" + title + '.JPEG' + "\n")
    else:
        file_train.write(current_dir + "/" + title + '.JPEG' + "\n")
        counter = counter + 1
```

Ví dụ nội dung file train.txt



```
1 /content/gdrive/My Drive/app/Darknet_colab/Multi-Image-Train/rural_003.JPEG
2 /content/gdrive/My Drive/app/Darknet_colab/Multi-Image-Train/rural_005.JPEG
3 /content/gdrive/My Drive/app/Darknet_colab/Multi-Image-Train/rural_002.JPEG
4 /content/gdrive/My Drive/app/Darknet_colab/Multi-Image-Train/rural_004.JPEG
5 /content/gdrive/My Drive/app/Darknet_colab/Multi-Image-Train/yield_102.JPEG
6 /content/gdrive/My Drive/app/Darknet_colab/Multi-Image-Train/rural_001.JPEG
7 /content/gdrive/My Drive/app/Darknet_colab/Multi-Image-Train/rural_043.JPEG
8 /content/gdrive/My Drive/app/Darknet_colab/Multi-Image-Train/rural_045.JPEG
9 /content/gdrive/My Drive/app/Darknet_colab/Multi-Image-Train/rural_042.JPEG
10 /content/gdrive/My Drive/app/Darknet_colab/Multi-Image-Train/rural_041.JPEG
11 /content/gdrive/My Drive/app/Darknet_colab/Multi-Image-Train/rural_046.JPEG
12 /content/gdrive/My Drive/app/Darknet_colab/Multi-Image-Train/rural_035.JPEG
13 /content/gdrive/My Drive/app/Darknet_colab/Multi-Image-Train/rural_037.JPEG
```

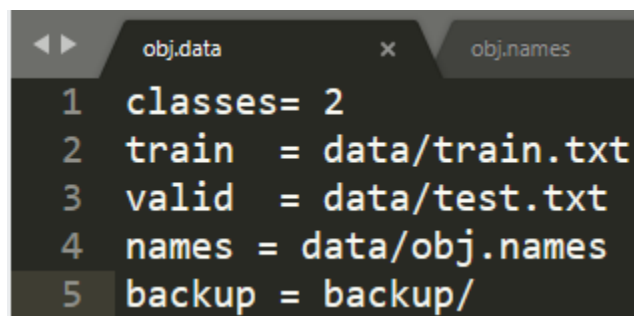
Bước 5. Chuẩn bị các file cần thiết phục vụ quá trình train dữ liệu

Chú ý: mục đích sử dụng **tiny-yolo** (to detect your custom objects)

Copy file **train.txt** đã tạo ở trên in the directory **darknet/data**

Copy file **test.txt** đã tạo ở trên in the directory **darknet/data**

Create file **obj.data** in the directory **darknet/data**



```
1 classes= 2
2 train = data/train.txt
3 valid = data/test.txt
4 names = data/obj.names
5 backup = backup/
```

Trong đó:

classes = 2 # Số lượng class, ở đây chỉ có 2 đối tượng lên classes=2

train = data/train.txt # trỏ đến file train của ta thôi

test = data/test.txt # trỏ đến file test của ta

names = data/obj.names # trỏ đến file names làm bên trên

backup = backup/ # là đường dẫn sẽ lưu các file weights trong quá trình train

Create file **obj.names** in the directory **darknet/data**



yolov3-tiny.weights

Download default weights file for yolov3-tiny: <https://pjreddie.com/media/files/yolov3-tiny.weights>

Copy **yolov3-tiny.weights** vào thư mục **darknet/**

yolov3-tiny.conv.15

Tạo file **yolov3-tiny.conv.15** sử dụng lệnh trên colab

```
!./darknet partial cfg/yolov3-tiny.cfg yolov3-tiny.weights yolov3-tiny.conv.15 15
```

```
[ ] !./darknet partial cfg/yolov3-tiny.cfg yolov3-tiny.weights yolov3-tiny.conv.15 15
```

layer	filters	size	input	output
0 conv	16	3 x 3 / 1	416 x 416 x 3	416 x 416 x 16 0.150 BF
1 max		2 x 2 / 2	416 x 416 x 16	208 x 208 x 16 0.003 BF
2 conv	32	3 x 3 / 1	208 x 208 x 16	208 x 208 x 32 0.399 BF
3 max		2 x 2 / 2	208 x 208 x 32	104 x 104 x 32 0.001 BF
4 conv	64	3 x 3 / 1	104 x 104 x 32	104 x 104 x 64 0.399 BF
5 max		2 x 2 / 2	104 x 104 x 64	52 x 52 x 64 0.001 BF
6 conv	128	3 x 3 / 1	52 x 52 x 64	52 x 52 x 128 0.399 BF
7 max		2 x 2 / 2	52 x 52 x 128	26 x 26 x 128 0.000 BF
8 conv	256	3 x 3 / 1	26 x 26 x 128	26 x 26 x 256 0.399 BF
9 max		2 x 2 / 2	26 x 26 x 256	13 x 13 x 256 0.000 BF
10 conv	512	3 x 3 / 1	13 x 13 x 256	13 x 13 x 512 0.399 BF
11 max		2 x 2 / 1	13 x 13 x 512	13 x 13 x 512 0.000 BF
12 conv	1024	3 x 3 / 1	13 x 13 x 512	13 x 13 x 1024 1.595 BF
13 conv	256	1 x 1 / 1	13 x 13 x 1024	13 x 13 x 256 0.089 BF
14 conv	512	3 x 3 / 1	13 x 13 x 256	13 x 13 x 512 0.399 BF
15 conv	255	1 x 1 / 1	13 x 13 x 512	13 x 13 x 255 0.044 BF
16 yolo				
17 route	13			
18 conv	128	1 x 1 / 1	13 x 13 x 256	13 x 13 x 128 0.011 BF
19 upsample		2x	13 x 13 x 128	26 x 26 x 128
20 route	19 8			
21 conv	256	3 x 3 / 1	26 x 26 x 384	26 x 26 x 256 1.196 BF
22 conv	255	1 x 1 / 1	26 x 26 x 256	26 x 26 x 255 0.088 BF
23 yolo				

Total BFLOPS 5.571
Loading weights from yolov3-tiny.weights...
seen 64
Done!
Saving weights to yolov3-tiny.conv.15

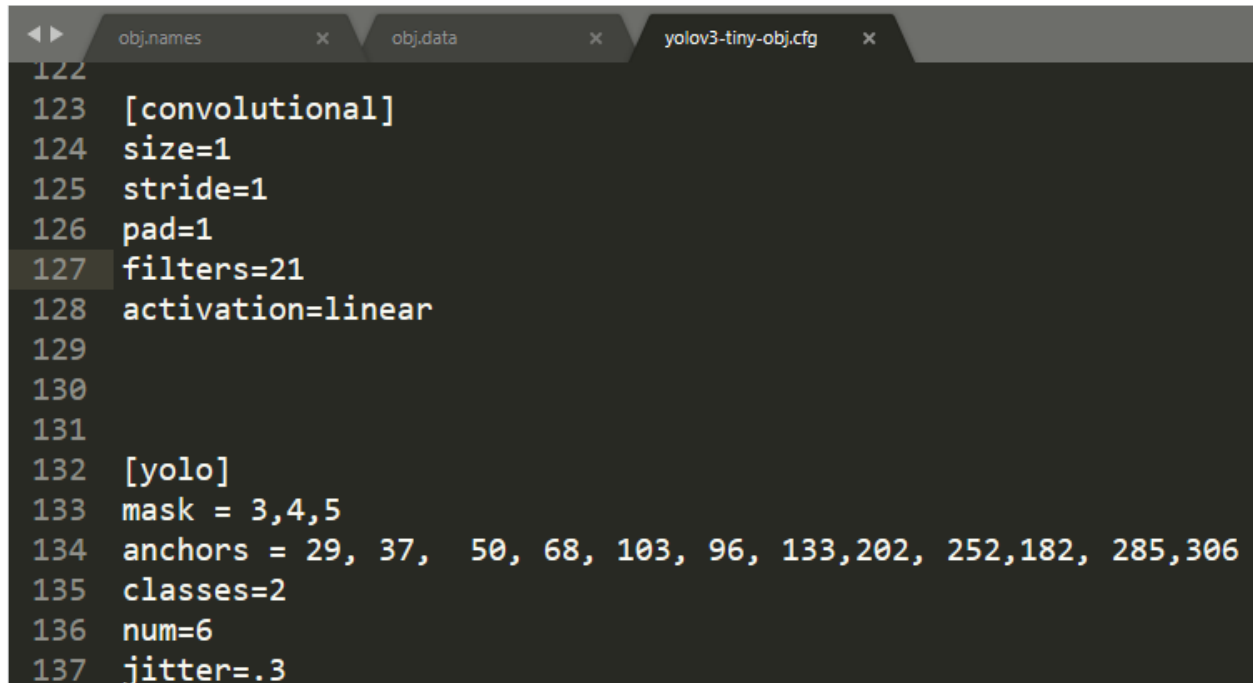
Và file **yolov3-tiny.conv.15** cũng nằm trong thư mục **darknet/**

yolov3-tiny-obj.cfg

Tạo file: yolov3-tiny-obj.cfg nằm trong thư mục **darknet/**

từ file (copy): **darknet/cfg/yolov3-tiny_obj.cfg** instead of **yolov3.cfg**

chỉnh sửa nội dung file **yolov3-tiny-obj.cfg**

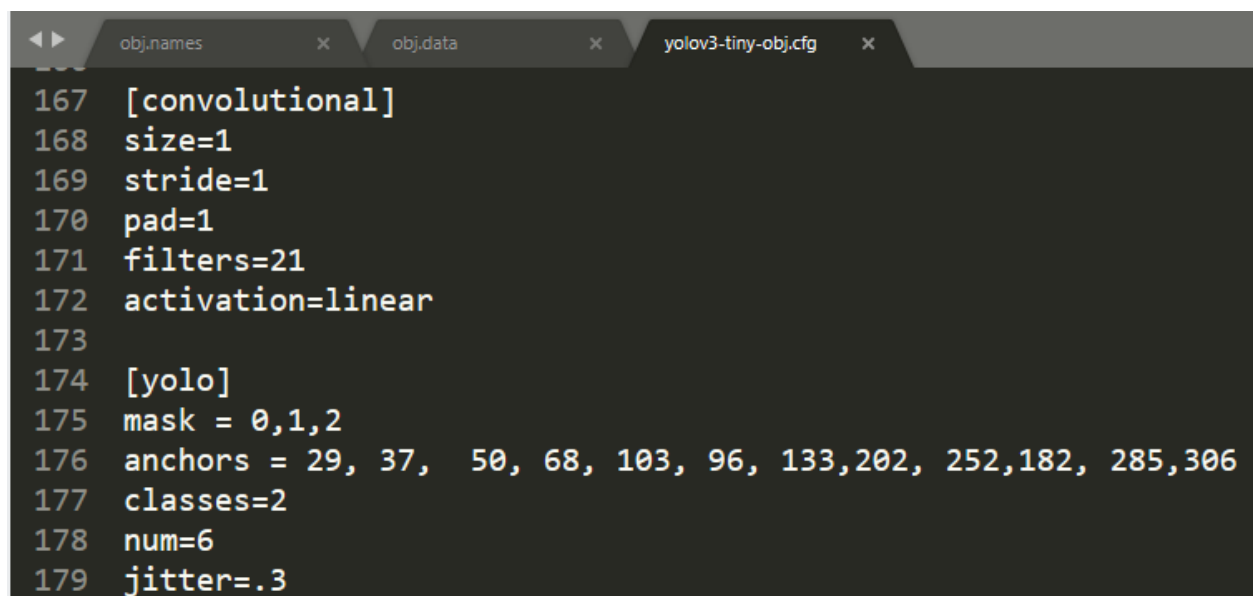


```
122
123 [convolutional]
124 size=1
125 stride=1
126 pad=1
127 filters=21
128 activation=linear
129
130
131
132 [yolo]
133 mask = 3,4,5
134 anchors = 29, 37, 50, 68, 103, 96, 133,202, 252,182, 285,306
135 classes=2
136 num=6
137 jitter=.3
```

So if classes=1 then should be filters=18. If classes=2 then write filters=21.

$\text{filters} = (\text{classes} + \text{coords} + 1) * \text{number of mask}$

$\text{filters} = (2 + 4 + 1) * 3 = 21$



```
167 [convolutional]
168 size=1
169 stride=1
170 pad=1
171 filters=21
172 activation=linear
173
174 [yolo]
175 mask = 0,1,2
176 anchors = 29, 37, 50, 68, 103, 96, 133,202, 252,182, 285,306
177 classes=2
178 num=6
179 jitter=.3
```

Thay đổi anchors:

Sử dụng lệnh:

```
!./darknet detector calc_anchors data/obj.data -num_of_clusters 6 -width 416 -height 416
```

Kết quả: 29, 37, 50, 68, 103, 96, 133, 202, 252, 182, 285, 306

Copy bỏ vào mục anchors trong file trên

Bước 6: Tiến hành train model

chạy lệnh sau để biến darknet thành file executable nhé:

```
chmod +x darknet
```

Rồi, cuối cùng chạy lệnh để train nào:











```
!./darknet detector train data/obj.data yolov3-tiny-obj.cfg yolov3-tiny.conv.15 -  
dont_show
```

Trong quá trình train, bạn để ý 2 tham số loss và avg loss, nếu thấy nó bão hòa và không thay đổi nhiều nữa thì có thể stop lại quá trình train.

```
!./darknet detector train data/obj.data yolov3-tiny-obj.cfg yolov3-tiny.conv.15 -dont_show
```

```
libpng warning: iCCP: known incorrect sRGB profile  
libpng warning: iCCP: known incorrect sRGB profile  
try to allocate additional workspace_size = 148.84 MB  
CUDA allocate done!  
Loaded: 0.327542 seconds  
  
5631: 0.127809, 0.142688 avg loss, 0.001000 rate, 0.635193 seconds, 360384 images  
Loaded: 0.092049 seconds  
libpng warning: iCCP: known incorrect sRGB profile  
  
5632: 0.126533, 0.141072 avg loss, 0.001000 rate, 0.674721 seconds, 360448 images  
Loaded: 0.147152 seconds  
  
5633: 0.138498, 0.140815 avg loss, 0.001000 rate, 0.653022 seconds, 360512 images  
Loaded: 0.154401 seconds  
  
5634: 0.182356, 0.144969 avg loss, 0.001000 rate, 0.614240 seconds, 360576 images  
Loaded: 0.117552 seconds  
libpng warning: iCCP: known incorrect sRGB profile  
  
5635: 0.122162, 0.142688 avg loss, 0.001000 rate, 0.646577 seconds, 360640 images  
Loaded: 0.469067 seconds  
  
5636: 0.136402, 0.142060 avg loss, 0.001000 rate, 0.637161 seconds, 360704 images  
Loaded: 0.234547 seconds  
^C
```

Kết quả các file weights được lưu trong quá trình train

My Drive > ... > darknet > backup ▾		
Name		Owner
 yolov3-tiny-obj_1000.weights		me
 yolov3-tiny-obj_2000.weights		me
 yolov3-tiny-obj_3000.weights		me
 yolov3-tiny-obj_5000.weights		me
 yolov3-tiny-obj_last.weights		me

Bước 7. Kiểm thử quá trình train bằng cách detect thử 1 ảnh.

```
!./darknet detector test data/obj.data yolov3-tiny-obj.cfg backup/yolov3-tiny-obj_5000.weights data/test.JPEG -i 0 -thresh 0.15
```

```
!./darknet detector test data/obj.data yolov3-tiny-obj.cfg backup/yolov3-tiny-obj_5000.weights data/test.JPEG -i 0 -thresh 0.15
```

layer	filters	size	input	output
0 conv	16	3 x 3 / 1	416 x 416 x 3	416 x 416 x 16 0.150 BF
1 max		2 x 2 / 2	416 x 416 x 16	208 x 208 x 16 0.003 BF
2 conv	32	3 x 3 / 1	208 x 208 x 16	208 x 208 x 32 0.399 BF
3 max		2 x 2 / 2	208 x 208 x 32	104 x 104 x 32 0.001 BF
4 conv	64	3 x 3 / 1	104 x 104 x 32	104 x 104 x 64 0.399 BF
5 max		2 x 2 / 2	104 x 104 x 64	52 x 52 x 64 0.001 BF
6 conv	128	3 x 3 / 1	52 x 52 x 64	52 x 52 x 128 0.399 BF
7 max		2 x 2 / 2	52 x 52 x 128	26 x 26 x 128 0.000 BF
8 conv	256	3 x 3 / 1	26 x 26 x 128	26 x 26 x 256 0.399 BF
9 max		2 x 2 / 2	26 x 26 x 256	13 x 13 x 256 0.000 BF
10 conv	512	3 x 3 / 1	13 x 13 x 256	13 x 13 x 512 0.399 BF
11 max		2 x 2 / 1	13 x 13 x 512	13 x 13 x 512 0.000 BF
12 conv	1024	3 x 3 / 1	13 x 13 x 512	13 x 13 x 1024 1.595 BF
13 conv	256	1 x 1 / 1	13 x 13 x 1024	13 x 13 x 256 0.089 BF
14 conv	512	3 x 3 / 1	13 x 13 x 256	13 x 13 x 512 0.399 BF
15 conv	21	1 x 1 / 1	13 x 13 x 512	13 x 13 x 21 0.004 BF
16 yolo				
17 route	13			
18 conv	128	1 x 1 / 1	13 x 13 x 256	13 x 13 x 128 0.011 BF
19 upsample		2x	13 x 13 x 128	26 x 26 x 128
20 route	19 8			
21 conv	256	3 x 3 / 1	26 x 26 x 384	26 x 26 x 256 1.196 BF
22 conv	21	1 x 1 / 1	26 x 26 x 256	26 x 26 x 21 0.007 BF
23 yolo				

Total BFLOPS 5.449
Allocate additional workspace_size = 52.43 MB
Loading weights from backup/yolov3-tiny-obj_5000.weights...
seen 64
Done!
data/test.JPEG: Predicted in 6.001000 milli-seconds.
yieldsign: 99%
Unable to init server: Could not connect: Connection refused

(predictions:29334): Gtk-WARNING **: 09:28:14.034: cannot open display:

Định nghĩa hàm imshow() để hiển thị kết quả:

```
#download files
def imshow(path):
    import cv2
    import matplotlib.pyplot as plt
    %matplotlib inline

    image = cv2.imread(path)
    height, width = image.shape[:2]
    resized_image = cv2.resize(image, (3*width, 3*height), interpolation = cv2.INTER_CUBIC)

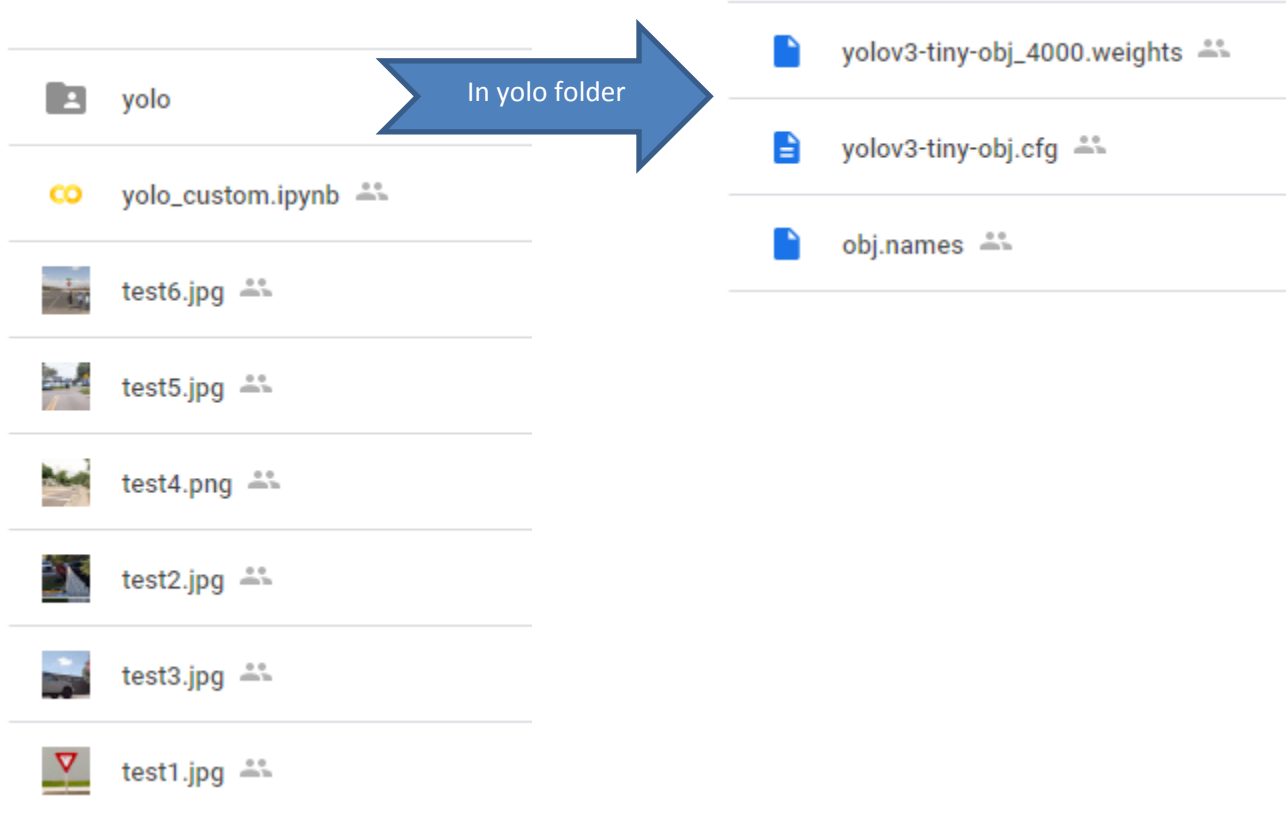
    fig = plt.gcf()
    fig.set_size_inches(18, 10)
    plt.axis("off")
    #plt.rcParams['figure.figsize'] = [10, 5]
    plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
    plt.show()
```

```
[ ] # Show the result using the helper imgShow()
    imshow('predictions.jpg')
```



YOLO object detection with OpenCV

Cấu trúc thư mục:



```
[ ] # import the necessary packages
import numpy as np
import argparse
import time
import cv2
import os
```

```
[ ] # load the COCO class labels our YOLO model was trained on
labelsPath = os.path.sep.join(['yolo', "obj.names"])
LABELS = open(labelsPath).read().strip().split("\n")
```

```
[ ] print(LABELS)
```

```
↳ ['stopsign', 'yieldsign']
```

```
[ ] # initialize a list of colors to represent each possible class label
    np.random.seed(42)
    COLORS = np.random.randint(0, 255, size=(len(LABELS), 3),
                                dtype="uint8")

    # derive the paths to the YOLO weights and model configuration
    weightsPath = os.path.sep.join(["yolo", "yolov3-tiny-obj_4000.weights"])
    configPath = os.path.sep.join(["yolo", "yolov3-tiny-obj.cfg"])

    # load our YOLO object detector trained on COCO dataset (80 classes)
    print("[INFO] loading YOLO from disk...")
    net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)
```

➞ [INFO] loading YOLO from disk...

```
[ ] #download files
def imShow1(path):
    import cv2
    import matplotlib.pyplot as plt
    %matplotlib inline

    image = path
    height, width = image.shape[:2]
    resized_image = cv2.resize(image, (3*width, 3*height), interpolation = cv2.INTER_CUBIC)

    fig = plt.gcf()
    fig.set_size_inches(18, 10)
    plt.axis("off")
    #plt.rcParams['figure.figsize'] = [10, 5]
    plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
    plt.show()
```

[]

```
def nhandang(image):
    (H, W) = image.shape[:2]
    # determine only the *output* layer names that we need from YOLO
    ln = net.getLayerNames()
    ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]

    # construct a blob from the input image and then perform a forward
    # pass of the YOLO object detector, giving us our bounding boxes and
    # associated probabilities
    blob = cv2.dnn.blobFromImage(image, 1 / 255.0, (416, 416),
        swapRB=True, crop=False)
    net.setInput(blob)
    start = time.time()
    layerOutputs = net.forward(ln)
    end = time.time()

    # show timing information on YOLO
    print("[INFO] YOLO took {:.6f} seconds".format(end - start))

    # initialize our lists of detected bounding boxes, confidences, and
    # class IDs, respectively
    boxes = []
    confidences = []
    classIDs = []

    # loop over each of the layer outputs
    for output in layerOutputs:
        # loop over each of the detections
        for detection in output:
            # extract the class ID and confidence (i.e., probability) of
            # the current object detection
            scores = detection[5:]
            classID = np.argmax(scores)
            confidence = scores[classID]

            # filter out weak predictions by ensuring the detected
            # probability is greater than the minimum probability
            if confidence > 0.5:
                # scale the bounding box coordinates back relative to the
                # size of the image, keeping in mind that YOLO actually
                # returns the center (x, y)-coordinates of the bounding
                # box followed by the boxes' width and height
                box = detection[0:4] * np.array([W, H, W, H])
                (centerX, centerY, width, height) = box.astype("int")

                # use the center (x, y)-coordinates to derive the top and
                # and left corner of the bounding box
                x = int(centerX - (width / 2))
                y = int(centerY - (height / 2))

                # update our list of bounding box coordinates, confidences,
                # and class IDs
                boxes.append([x, y, int(width), int(height)])
                confidences.append(float(confidence))
                classIDs.append(classID)

    # apply non-maxima suppression to suppress weak, overlapping bounding
    # boxes
    idxs = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.3)

    # ensure at least one detection exists
    if len(idxs) > 0:
        # loop over the indexes we are keeping
        for i in idxs.flatten():
            # extract the bounding box coordinates
            (x, y) = (boxes[i][0], boxes[i][1])
            (w, h) = (boxes[i][2], boxes[i][3])

            # draw a bounding box rectangle and label on the image
            color = [int(c) for c in COLORS[classIDs[i]]]
            cv2.rectangle(image, (x, y), (x + w, y + h), color, 2)
            text = "{}: {:.4f}".format(LABELS[classIDs[i]], confidences[i])
            cv2.putText(image, text, (x, y - 5), cv2.FONT_HERSHEY_SIMPLEX,
                0.5, color, 2)
    imShow1(image)
```

```
[ ] # load our input image and grab its spatial dimensions
image = cv2.imread("test1.jpg")
nhandang(image)
```

📄 [INFO] YOLO took 0.081916 seconds



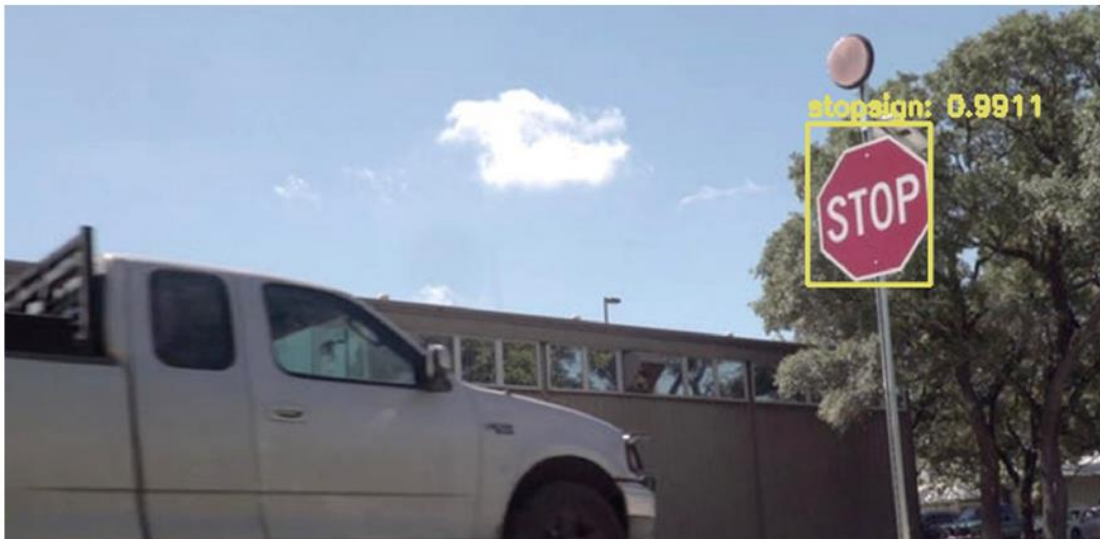
```
[ ] # load our input image and grab its spatial dimensions
image = cv2.imread("test2.jpg")
nhandang(image)
```

📄 [INFO] YOLO took 0.082914 seconds




```
[ ] # load our input image and grab its spatial dimensions
image = cv2.imread("test3.jpg")
nhandang(image)
```

➡ [INFO] YOLO took 0.126001 seconds



```
[ ] # load our input image and grab its spatial dimensions
image = cv2.imread("test4.png")
nhandang(image)
```

➡ [INFO] YOLO took 0.114337 seconds



https://medium.com/@manivannan_data/how-to-train-multiple-objects-in-yolov2-using-your-own-dataset-2b4fee898f17