YOLOv3 Custom Object Detection with Transfer Learning

This guide will help you to perform object detection for your own custom data by applying Transfer Learning using YOLOv3. So let's begin.

Step 1: Prepare dataset.

a) Create a dataset of the object for which you want to perform its detection. You may scrape images from **Google Images** to download data or use any other source for your data.

b) Clean your dataset by removing unwanted/irrelevant images. Also, make sure all the images are of the format **.jpg.**

Once you are done with the above two steps, you're good to go for the next step.

Step 2: Data Annotation.

We need a data annotation tool to label our images. Although there are plenty of annotation tools available, we will go ahead with **LabelImg.**

a) Clone or download ZIP for LabelImg from the following GitHub repository:

```
git clone https://github.com/tzutalin/labelImg.git
```

b) Open LabelImg and configure the settings as shown in the figure below.

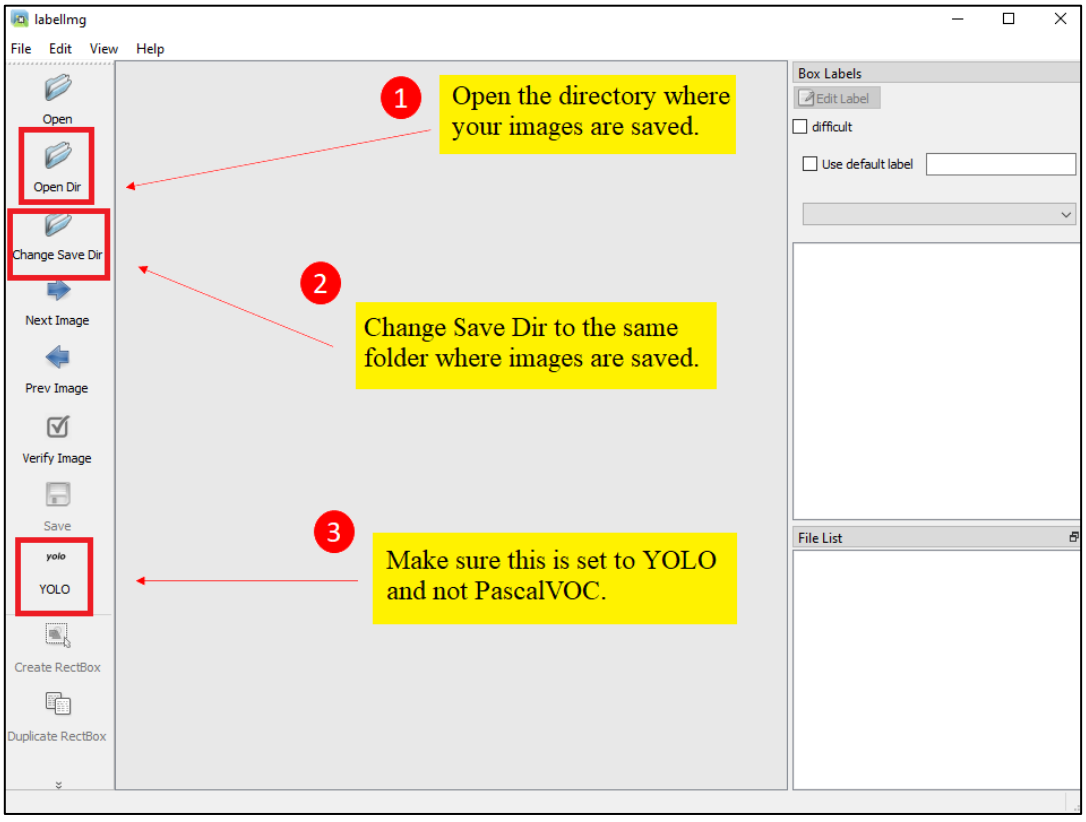


Fig-1: Configure LabelImg

c) You are now good to label your images.

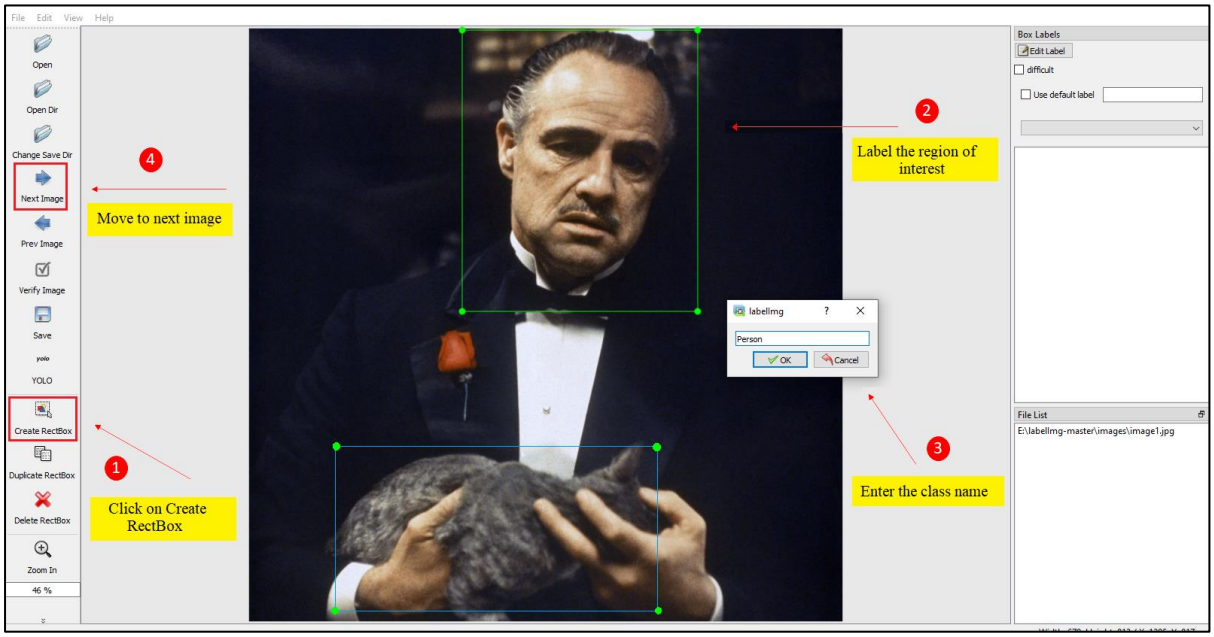


Fig-2: Data Labelling

d) A **classes.txt** file is generated that includes the list of all classes you have annotated in your dataset. For every image file annotated, a corresponding **.txt** file is also generated that includes the metadata.

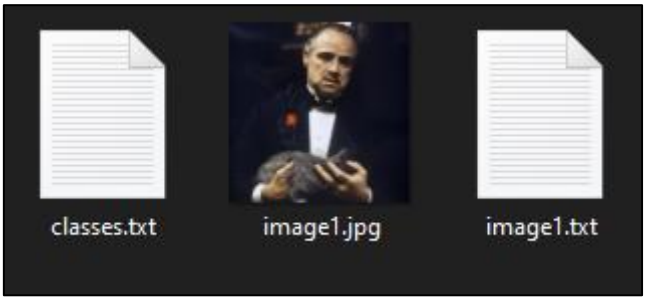


Fig-3: Files in directory

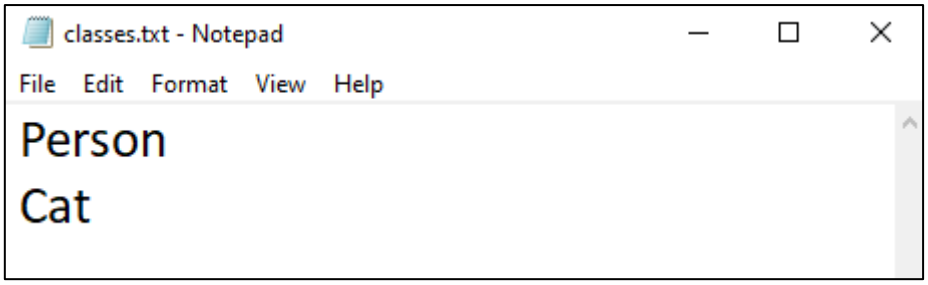


Fig-4: classes.txt

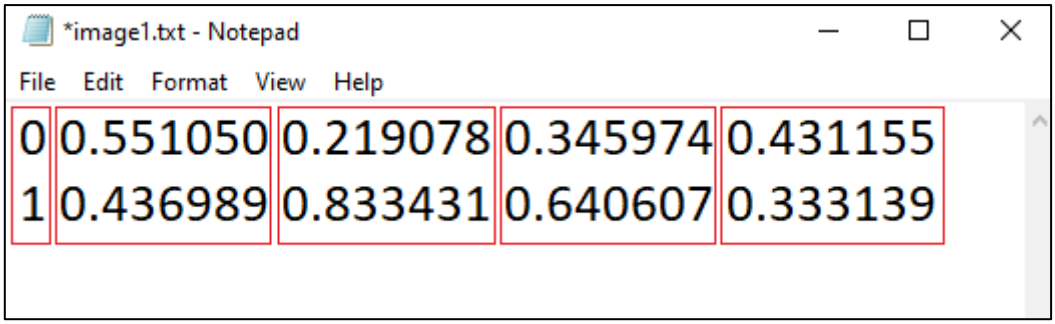


Fig-5: image1.txt (metadata)

The metadata includes the following –
object_id center_x center_y width height

object_id represents the number corresponding to the object category which we listed in 'classes.txt' earlier.

center_x and **center_y** represent the center point of the bounding box. But they are normalized to range between 0 and 1 by dividing by the width and height of the image.

width and **height** represents the width and height of the bounding box. Again normalized to the range 0 to 1 dividing by the original width and height of the image.

Step 3: Training the model.

Once you have labelled all your data, you may go ahead with the actual training process of the model. Remember, *"Garbage in, garbage out"*. Make sure you have a healthy dataset size and have correctly labelled the objects if you want good accuracy.

a. Upload dataset on Google Drive.

(i) Create a ZIP file of your dataset that contains all the images ***.jpg**, annotations ***.txt** and **classes.txt** files and name it as **images.zip**.

(ii) Sign in to your Google account and open Google Drive. Create a new folder named as **yolov3** and upload the images.zip folder inside it.

b. Setting up Google Colab.

We'll be doing our model training on Google Colab as it provides free GPU access as well as an environment to install all the required dependencies with ease.

(i) Clone or download ZIP from the following GitHub repository on your local machine.

```
git clone https://github.com/NSTiwari/YOLOv3-Custom Object-Detection.git
```

(ii) Open Google Colab and upload the **YOLOv3_Custom_Object_Detection.ipynb** file from the downloaded repository.

c. Model Training.

We are now ready to train our model. Run the notebook cells one-by-one as follows.

(i) Check if NVIDIA GPU is enabled for training.

```
# Check if NVIDIA GPU is enabled
!nvidia-smi
```

```
Tue Dec  1 16:59:58 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 455.38       Driver Version: 418.67       CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla P4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   37C    P8     7W /  75W |      0MiB /  7611MiB |      0%      Default |
|                               |                      |                 ERR! |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

(ii) Mount your Google Drive on Google Colab.

This is to provide access to the dataset **images.zip** stored on your Google Drive. Click on the link highlighted in blue and copy the authorization code that appears in a new tab. Paste it in the box below as shown.

```
from google.colab import drive
drive.mount('/content/gdrive')
!ln -s /content/gdrive/My\ Drive/ /mydrive
!ls /mydrive
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=

Enter your authorization code:
```

(iii) Clone, configure and compile Darknet.

```
# Clone
!git clone https://github.com/AlexeyAB/darknet
```

```
Cloning into 'darknet'...
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 14518 (delta 0), reused 3 (delta 0), pack-reused 14500
Receiving objects: 100% (14518/14518), 13.24 MiB | 12.38 MiB/s, done.
Resolving deltas: 100% (9865/9865), done.
```

```
# Configure
%cd darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
```

```
[ ]  # Compile
     !make
```

(iv) Configure yolov3.cfg file.

This cell creates a copy of **yolov3.cfg** file and names it as **yolov3_training.cfg** so that all the changes and configurations that we will make for our custom model would be reflected in the copy and not the original file. It's a good practice to keep a backup of the original .cfg file if in case anything goes wrong.

```
[ ]  # Make a copy of yolov3.cfg
     !cp cfg/yolov3.cfg cfg/yolov3_training.cfg
```

The cell below edits the **yolov3_training.cfg** file as follows:

On lines 610, 696 and 783, the no. of classes are changed from 80 to 2 since we are working on 2 classes (Person and Cat).

On lines 603, 689 and 776, the no. of filters are changed from 255 to 21.

```
[ ]  # Change lines in yolov3.cfg file
     !sed -i 's/batch=1/batch=64/' cfg/yolov3_training.cfg
     !sed -i 's/subdivisions=1/subdivisions=16/' cfg/yolov3_training.cfg
     !sed -i 's/max_batches = 500200/max_batches = 4000/' cfg/yolov3_training.cfg
     !sed -i '610 s@classes=80@classes=2@' cfg/yolov3_training.cfg
     !sed -i '696 s@classes=80@classes=2@' cfg/yolov3_training.cfg
     !sed -i '783 s@classes=80@classes=2@' cfg/yolov3_training.cfg
     !sed -i '603 s@filters=255@filters=21@' cfg/yolov3_training.cfg
     !sed -i '689 s@filters=255@filters=21@' cfg/yolov3_training.cfg
     !sed -i '776 s@filters=255@filters=21@' cfg/yolov3_training.cfg
```

Note: If your custom object detection model includes 'n' no. of classes, then **max_batches = 2000 * n** and **filters = (n + 5) * 3**

For n = 1: max_batches = 2000, classes=1, filters=18
For n = 2: max_batches = 4000, classes=2, filters=21
For n = 3: max_batches = 8000, classes=3, filters=24

(v) Create .names and .data files.

This cell creates **obj.names** and **obj.data** files inside the **darknet/data/obj** directory. These files include metadata such as class names and no. of classes required for training.

```
[ ] !echo -e 'Person\nCat' > data/obj.names
    !echo -e 'classes= 2\ntrain  = data/train.txt\nvalid  = data/test.txt\nnames = data/obj.names\nbackup = /mydrive/yolov3' > data/obj.data
```

Note: If your custom object detection model includes 'n' no. of classes, then edit the above cell accordingly by specifying the class names and no. of classes.

(vi) Save yolov3_training.cfg and obj.names files in Google Drive.

```
[ ]  !cp cfg/yolov3_training.cfg /mydrive/yolov3/yolov3_testing.cfg
     !cp data/obj.names /mydrive/yolov3/classes.txt
```

(vii) Unzip the images dataset.

The cell below unzips the **images.zip** file stored on your Google Drive, into the **darknet/data/obj** directory.

```
     !mkdir data/obj
     !unzip /mydrive/yolov3/images.zip -d data/obj
```

(viii) Create train.txt file.

The cell below creates a **train.txt** file that includes the location of all the images ***.jpg** stored inside **darknet/data/obj** directory. In other words, images will be fetched from the location specified in this file during training.

```
[ ]  import glob
     images_list = glob.glob("data/obj/*.jpg")
     with open("data/train.txt", "w") as f:
         f.write("\n".join(images_list))
```

(ix) Download pre-trained weights for the convolutional layers file.

We are applying Transfer Learning by adding our own layers to an already trained model. So, we download the pre-trained weights called **darknet53.conv.74**. Thus, our custom model will be trained using these pre-trained weights instead of randomly initialized weights which in turn will save a lot of time and computations while training our own model.

```
[ ]  !wget https://pjreddie.com/media/files/darknet53.conv.74
```

(x) Start training.

We are now set to train our model. Run the cell below to start training.

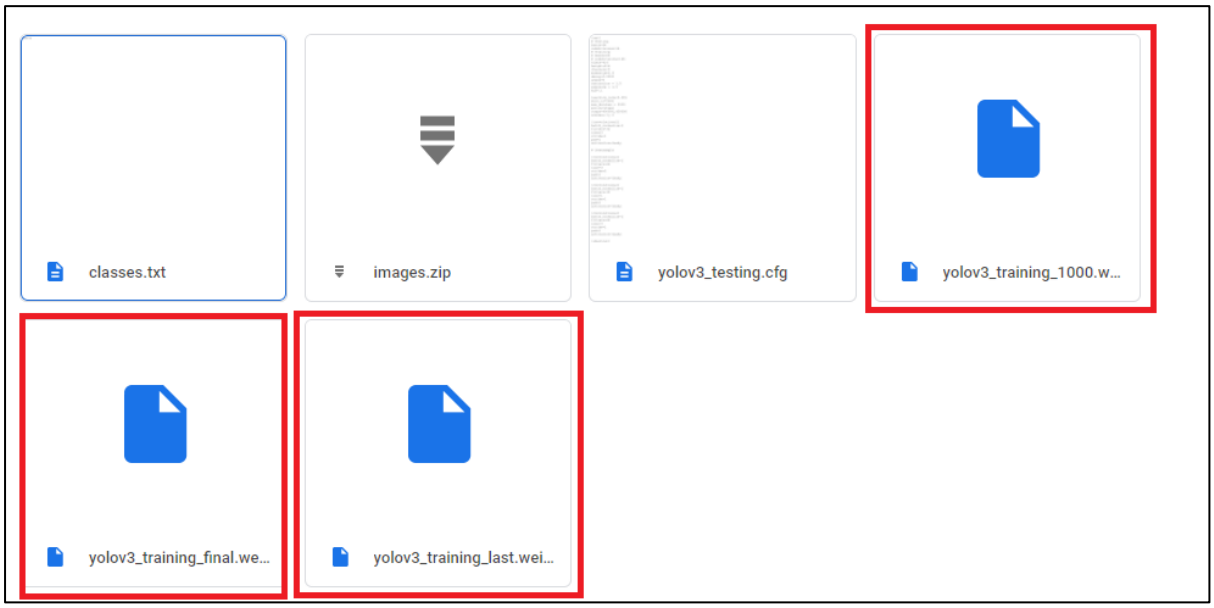
```
[ ] !./darknet detector train data/obj.data cfg/yolov3_training.cfg darknet53.conv.74 -dont_show
    # Uncomment below and comment above to re-start your training from last saved weights
    #!./darknet detector train data/obj.data cfg/yolov3_training.cfg /mydrive/yolov3/yolov3_training_last.weights -dont_show
```

The model will take some time to train depending upon your dataset size and the no. of classes. Meanwhile the model is training, go and have a coffee or instead go out for a walk. It should take about 7-8 hours for training 2 classes each of 400 training sample size, so you can estimate the approximate time required for training your own custom model depending upon your dataset size and no. of classes.

In case the training stopped for some reasons i.e. because of network or power failure or non-availability of GPU resource allocation or for any other reason, don't worry. You can continue the training process from the last saved weights. Simply comment the first line and uncomment the last line in the above cell and rerun it.

Step 4: Testing the model.

Once the model is trained completely, atleast three files will be downloaded inside the **yolov3** folder on your Google Drive, depending upon the model size; as shown in the figure below.

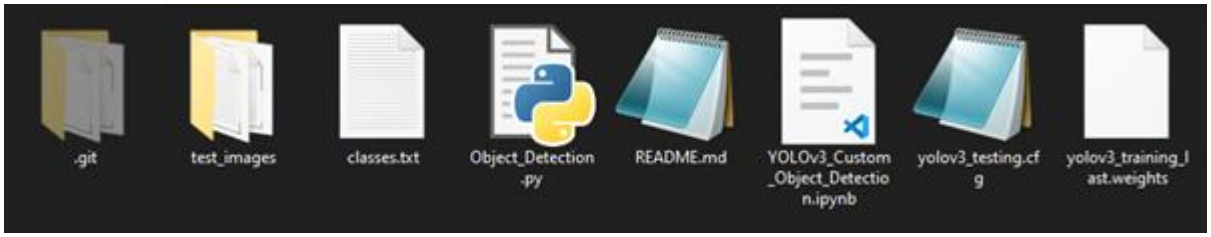


(i) Download **yolov3_training_last.weights, classes.txt** and **yolov3_testing.cfg** and save inside the **YOLOv3_Custom_Object_Detection** repository you downloaded in Step 3b.

(ii) Create a new folder called **test_images** inside the **YOLOv3_Custom_Object_Detection** repository and save some images inside it which you would like to test the model on.

(iii) Open the **Object_Detection.py** file and edit Line 17 by replacing <your_test_image> with the name of image file you want to test.

The overall directory should look like this.



Before you test the model, make sure you have installed OpenCV on your machine. If not, run the following two commands on command prompt.

```
pip install opencv-python
pip install opencv-contrib-python
```

You are now ready to test the model. Open command prompt and navigate to the **YOLOv3_Custom_Object_Detection** directory and run the following command.

```
python Object_Detection.py
```

Congratulations, you have successfully created your own custom object detection model.