**Object Detection with Yolov3**

20/12/2019

**Content**

Overview

Implementation

Object detection with pre-trained model

Custom object detector

Additional details

**Overview**

Y0lo, family of models are a series of end-to-end deep learning models designed for fast object detection, developed by [Joseph Redmon](https://pjreddie.com/). In other words, it is an object detector. (object locator, object recognizer)

 Yolo model has several advantages over classifier-based systems. It looks at the whole image at test time, so its predictions are informed by global context in the image. It also makes predictions with a single network evaluation unlike systems like [R-CNN](https://github.com/rbgirshick/rcnn) which require thousands for a single image. This makes it extremely fast, more than 1000x faster than R-CNN and 100x faster than [Fast R-CNN](https://github.com/rbgirshick/fast-rcnn).

This report is suitable for whoever want to get started with Yolov3. You can think about it as a guide or a cheat sheet that contain suggested way to install Yolov3 for working on both CPU, GPU, using a pre-trained model to detect image and video and training own model to detect custom image and video.

<https://pjreddie.com/darknet/yolo/>

**Implementation**

1. Darknet
   1. Darknet official (<https://pjreddie.com/darknet/>)

Support: GPU (Nvida), CPU

Advantage: Fast (written in C with CUDA)

Disadvantage: Difficult to answers for errors. It’s only works with Linux. Slow when run on CPU.

1.2Darknet from other source (<https://github.com/AlexeyAB/darknet>)

Support: GPU(Nvida), CPU

Advantage: Fork of Darknet to support vary OS. It is an excellent source to find tips about YOLO.

Disadvantage: Hard to install on Windows.   
 Slow when run on CPU.

1. Darkflow (<https://github.com/thtrieu/darkflow/>)

Support: GPU(Nvida), CPU

Advantage: It’s the adaptation of darknet to Tensorflow.

It’s compatible with Linux, Windows and Mac.

Using CPU-only Darkflow is several times faster than the original Darkent.

Disadvantage: The installation is complex, especially on Windows.

Not updated to YOLOv3.

1. OpenCV

Support: CPU

Advantage: Just install OpenCV with at least version 3.4.2.

Python support.

Disadvantage: Not support GPU, make it can’t perform high speed.

**1.1 Darknet official**

<https://pjreddie.com/darknet/install/>

git clone https://github.com/pjreddie/darknet

cd darknet

make

mkdir -p obj

If you have any errors, try to fix them? If everything seems to have compiled correctly, try running it!

./darknet

You should get the output:

usage: .darknet<function>

**Compiling With CUDA**

Darknet on the CPU is fast but it's like 500 times faster on GPU! You'll have to have an [Nvidia GPU](https://developer.nvidia.com/cuda-gpus) and you'll have to install [CUDA](https://developer.nvidia.com/cuda-downloads). Once you have CUDA installed, change the first line of the Makefile in the base directory to read:

GPU=1

Now you can make the project and CUDA will be enabled. By default, it will run the network on the 0th graphics card in your system (if you installed CUDA correctly you can list your graphics cards using nvidia-smi). If you want to change what card Darknet uses you can give it the optional command line flag -i <index>, like:

./darknet -i 1 imagenet test cfg/alexnet.cfg alexnet.weights

If you compiled using CUDA but want to do CPU computation for whatever reason you can use -nogpu to use the CPU instead:

./darknet -nogpu imagenet test cfg/alexnet.cfg alexnet.weights

alexnet.weights

**Compiling with OpenCV**

By default, Darknet uses [stb\_image.h](https://github.com/nothings/stb/blob/master/stb_image.h) for image loading. If you want more support for weird formats (like CMYK jpegs, thanks Obama) you can use [OpenCV](http://opencv.org/) instead! OpenCV also allows you to view images and detections without having to save them to disk.

First install OpenCV. If you do this from source, it will be long and complex so try to get a package manager to do it for you.

Next, change the 2nd line of the Makefile to read:

OPENCV=1

You're done! To try it out, first re-make the project. Then use the imtest routine to test image loading and displaying:

./darknet imtest data/eagle.jpg

**1.2Darknet from other source**

Recommended sources

**1.2.1** [**https://github.com/AlexeyAB/darknet**](https://github.com/AlexeyAB/darknet) (Support Windows and Linux)

+ <https://www.youtube.com/watch?v=NSLcXiwyFYE> (How to install on Windows)

The list below shows main contents on this site.

* [Requirements (and how to install dependecies)](https://github.com/AlexeyAB/darknet#requirements)
* [Pre-trained models](https://github.com/AlexeyAB/darknet#pre-trained-models)
* [Explanations in issues](https://github.com/AlexeyAB/darknet/issues?q=is%3Aopen+is%3Aissue+label%3AExplanations)
* [Yolo v3 in other frameworks (TensorRT, TensorFlow, PyTorch, OpenVINO, OpenCV-dnn, TVM,...)](https://github.com/AlexeyAB/darknet#yolo-v3-in-other-frameworks)
* [Datasets](https://github.com/AlexeyAB/darknet#datasets)

1. [Improvements in this repository](https://github.com/AlexeyAB/darknet#improvements-in-this-repository)
2. [How to use](https://github.com/AlexeyAB/darknet#how-to-use-on-the-command-line)
3. How to compile on Linux
   * [Using cmake](https://github.com/AlexeyAB/darknet#how-to-compile-on-linux-using-cmake)
   * [Using make](https://github.com/AlexeyAB/darknet#how-to-compile-on-linux-using-make)
4. How to compile on Windows
   * [Using CMake-GUI](https://github.com/AlexeyAB/darknet#how-to-compile-on-windows-using-cmake-gui)
   * [Using vcpkg](https://github.com/AlexeyAB/darknet#how-to-compile-on-windows-using-vcpkg)
   * [Legacy way](https://github.com/AlexeyAB/darknet#how-to-compile-on-windows-legacy-way)
5. [How to train (Pascal VOC Data)](https://github.com/AlexeyAB/darknet#how-to-train-pascal-voc-data)
6. [How to train with multi-GPU:](https://github.com/AlexeyAB/darknet#how-to-train-with-multi-gpu)
7. [How to train (to detect your custom objects)](https://github.com/AlexeyAB/darknet#how-to-train-to-detect-your-custom-objects)
8. [How to train tiny-yolo (to detect your custom objects)](https://github.com/AlexeyAB/darknet#how-to-train-tiny-yolo-to-detect-your-custom-objects)
9. [When should I stop training](https://github.com/AlexeyAB/darknet#when-should-i-stop-training)
10. [How to calculate mAP on PascalVOC 2007](https://github.com/AlexeyAB/darknet#how-to-calculate-map-on-pascalvoc-2007)
11. [How to improve object detection](https://github.com/AlexeyAB/darknet#how-to-improve-object-detection)
12. [How to mark bounded boxes of objects and create annotation files](https://github.com/AlexeyAB/darknet#how-to-mark-bounded-boxes-of-objects-and-create-annotation-files)
13. [How to use Yolo as DLL and SO libraries](https://github.com/AlexeyAB/darknet#how-to-use-yolo-as-dll-and-so-libraries)

**1.2.2** [**https://www.npmjs.com/package/darknet**](https://www.npmjs.com/package/darknet) (Yolov3 in JavaScript)

Support Linux, Mac, Windows (Linux sub-system).

**1.2.3** [**https://jkjung-avt.github.io/yolov3/**](https://jkjung-avt.github.io/yolov3/) (Yolov3 on Jetson TX2)

**Prerequisite**

1.Cuda

2.<https://www.jetsonhacks.com/2017/04/05/build-opencv-nvidia-jetson-tx2/>

**Step-by-step**

1.Clone some darknet source code from GitHub. Such as 1.1, 1.2 etc.

$ cd ~/project

$ git clone <https://github.com/pjreddie/darknet> yolov3

$ cd yolov3

2. Modify the first few lines of Makefile as follows. Note that CUDA architecture of TX2 is “62”.

GPU=1

CUDNN=1

OPENCV=1

......

ARCH= -gencode arch=compute\_53,code=[sm\_53,compute\_53] \

-gencode arch=compute\_62,code=[sm\_62,compute\_62]

3.Build the code

$ make

**Object detection with pre-trained model**

* 1. **OpenCV**

<https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/>

<https://github.com/iArunava/YOLOv3-Object-Detection-with-OpenCV>

* 1. **Darknet**

<https://github.com/pjreddie/darknet/wiki/YOLO:-Real-Time-Object-Detection>

**Image**

./darknet yolo test cfg/yolo.cfg <path>/yolo.weights <image>

**Multiple images**

./darknet yolo test cfg/yolo.cfg yolo.weights

**Real-time Detection on a Webcam**

To run this demo you will need to compile [Darknet with CUDA and OpenCV](http://pjreddie.com/darknet/install/#cuda). You will also need to [pick a YOLO config](http://pjreddie.com/darknet/yolo/#models) file and have the appropriate weights file. Then run the command:

./darknet yolo demo cfg/yolo.cfg yolo.weights

* 1. **Darknet (AlexeyAB)**

<https://github.com/AlexeyAB/darknet#how-to-use-on-the-command-line>

**Custom object detector**

**Darknet**

**Ref:**

<https://blog.francium.tech/custom-object-training-and-detection-with-yolov3-darknet-and-opencv-41542f2ff44e>

<https://www.learnopencv.com/training-yolov3-deep-learning-based-custom-object-detector/>

<https://github.com/ivangrov/YOLOv3-Series>

<https://towardsdatascience.com/training-yolo-for-object-detection-in-pytorch-with-your-custom-dataset-the-simple-way-1aa6f56cf7d9>

1. Gathering Data
   * OpenImageV5 <https://storage.googleapis.com/openimages/web/index.html>
   * Scraping <https://www.youtube.com/watch?v=yXD5_W0JPuw>

Code: <https://github.com/ivangrov/YOLOv3-Series/tree/master/%5BPart%203%5D%20Get%20Images>

1. Labeling image (Data annotation)

Label file is text file which each row contains information about a single bounding box in the image. So, each image will have a Label file with same name saving in the same directory.

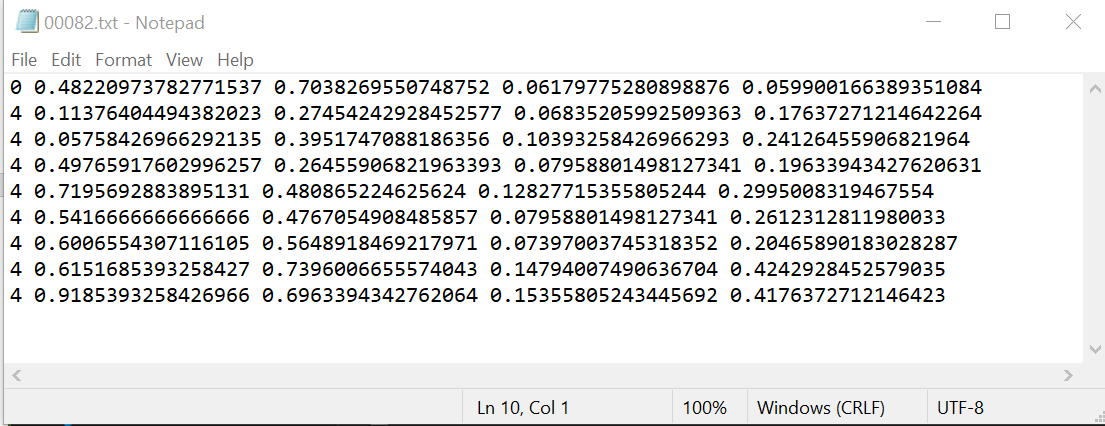


Image -> 00082.jpg

Label file -> 00082.txt

Both Image and Label file need to be saved in save directory.

<object-class-id> <center-x> <center-y> <width> <height>

x-coordinate (in pixels) of the center of the bounding box  
y-coordinate (in pixels) of the center of the bounding box  
w-width (in pixels) of the bounding box

h –height (in pixels) of the bounding box

W – width (in pixels) of the whole image  
H – height (in pixels) of the whole image

Then we compute the annotation values in the label files as follows:

center-x = x/W

center-y = y/H

width = w/W

height = h/H

Here are tools for help you labeling images.

* + LabelImg <https://github.com/tzutalin/labelImg>
  + Tutorial:<https://www.youtube.com/watch?v=kmgocZpidU8&list=PLZBN9cDu0MSk4IFFnTOIDihvhnHWhAa8W&index=4>   
    code: <https://github.com/ivangrov/YOLOv3-Series/blob/master/%5Bpart%204%5DOpenLabelling/run.py>
  + Yolo\_Label <https://github.com/developer0hye/Yolo_Label>

1. Download pre-trained model

For yolov3.cfg, yolov3-spp.cfg

cd <your darknet folder>

wget https://pjreddie.com/media/files/darknet53.conv.74 -O ~/darknet/darknet53.conv.74

For yolov3-tiny-prn.cfg , yolov3-tiny.cfg

Download

https://drive.google.com/file/d/18v36esoXCh-PsOKwyP2GWrpYDptDY8Zf/view?usp=sharing

to your darknet folder

1. Manage data file

Create <name>.data file in the <folderName> directory which should contain information like below.

classes=<number of image classes you want to detect>  
train=<folderName>/train.txt  
valid=<folderName>/test.txt  
names=<folderName>/<name>.names  
backup=backup/

**Train and Test sets**

We can then randomly split the annotated images into train and test sets in the ratio of 80:20

<folderName>/train.txt Each row in the file should have the location of train dataset.

<folderName>/test.txt Each row in the file should have the location of test dataset.

**<name>.names** file represents the names of all the classes orderly.

**Backup** is the location where newly trained weights would be saved.

1. Configurations

Create our own configuration file based on the demo configuration file, such as yolov3-voc.cfg, yolov3.cfg, yolov3-tiny.cfg. All the important training parameters are stored in these configuration file. You can see from **example** below.

* + Choose tiny-yolo.cfg as prototype.

Copy the tiny-yolo.cfg and save the file name as <name>-tiny-yolo.cfg

Following edits:

Line 3: set batch=24, this means we will be using 24 images for every training step

Line 4: set subdivisions=8, the batch will be divided by 8 to decrease GPU VRAM requirements.

Line 127: set filters=(classes + 5)\*3

Line 135: set classes=2, the number of categories we want to detect

Line 171: set filters=(classes + 5)\*3

Line 177: set classes=2, the number of categories we want to detect

* + Choose yolov3.cfg as prototype

Copy the yolo3.cfg and save the file name as <name>-yolo3.cfg

Following edits:

Line 3: set batch=24, this means we will be using 24 images for every training step

Line 4: set subdivisions=8, the batch will be divided by 8 to decrease GPU VRAM requirements.

Line 603: set filters=(classes + 5)\*3 in our case filters=21

Line 610: set classes=2, the number of categories we want to detect

Line 689: set filters=(classes + 5)\*3 in our case filters=21

Line 696: set classes=2, the number of categories we want to detect

Line 776: set filters=(classes + 5)\*3 in our case filters=21

Line 783: set classes=2, the number of categories we want to detect

If you get problem with the performance of your model. You might need to calculate new anchor value in your configuration file. From this command:

./darknet.exe detector calc\_anchors data/obj.data -num\_of\_clusters 9 -width 416 -height 416

For more advanced setting you should read from Reference above.

1. Training model

For yolov3.cfg, yolov3-spp.cfg

./darknet detector train data/obj.data yolo-obj.cfg darknet53.conv.74

For yolov3-tiny-prn.cfg , yolov3-tiny.cfg

darknet.exe detector train data/obj.data yolov3-tiny-obj.cfg yolov3-tiny.conv.15

After each 10000 iterations you can stop and later start training from this point. For example, after 10000 iterations you can stop training, and later just start training using:

./darknet detector train data/obj.data yolo-obj.cfg backup\yolo-obj\_10000.weights

(if iterations > 1000)

1. Use your trained model

./darknet detector test data/obj.data yolo-obj.cfg yolo-obj\_20000.weights

**Darknet/Alexey**

**Ref:** <https://github.com/AlexeyAB/darknet>

Recommended way. You just follow it since implementation. Read it from the start patiently and you will be fine.

**Additional detailed**

**How to improve object detection**

[**https://github.com/AlexeyAB/darknet#how-to-improve-object-detection**](https://github.com/AlexeyAB/darknet#how-to-improve-object-detection)

**How YOLOv3 work**

<https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088>

**What’s new in YOLOv3**

<https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b>

**mAP**

<https://wiseodd.github.io/techblog/2017/01/01/mle-vs-map/>

**IoU**

<https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>

#### **Command line Darknet/AlexeyAB**

On Linux use ./darknet instead of darknet.exe, like this: ./darknet detector test ./cfg/coco.data ./cfg/yolov3.cfg ./yolov3.weights

On Linux find executable file ./darknet in the root directory, while on Windows find it in the directory \build\darknet\x64

Yolo v3 COCO - **image**: darknet.exe detector test cfg/coco.data cfg/yolov3.cfg yolov3.weights -thresh 0.25

**Output coordinates** of objects: darknet.exe detector test cfg/coco.data yolov3.cfg yolov3.weights -ext\_output dog.jpg

Yolo v3 COCO - **video**: darknet.exe detector demo cfg/coco.data cfg/yolov3.cfg yolov3.weights -ext\_output test.mp4

Yolo v3 COCO - **WebCam 0**: darknet.exe detector demo cfg/coco.data cfg/yolov3.cfg yolov3.weights -c 0

Yolo v3 COCO for **net-videocam** - Smart WebCam: darknet.exe detector demo cfg/coco.data cfg/yolov3.cfg yolov3.weights http://192.168.0.80:8080/video?dummy=param.mjpg

Yolo v3 - **save result videofile res.avi**: darknet.exe detector demo cfg/coco.data cfg/yolov3.cfg yolov3.weights test.mp4 -out\_filename res.avi

Yolo v3 **Tiny** COCO - video: darknet.exe detector demo cfg/coco.data cfg/yolov3-tiny.cfg yolov3-tiny.weights test.mp4

**JSON and MJPEG server** that allows multiple connections from your soft or Web-browser ip-address:8070 and 8090: ./darknet detector demo ./cfg/coco.data ./cfg/yolov3.cfg ./yolov3.weights test50.mp4 -json\_port 8070 -mjpeg\_port 8090 -ext\_output

Yolo v3 Tiny **on GPU #1**: darknet.exe detector demo cfg/coco.data cfg/yolov3-tiny.cfg yolov3-tiny.weights -i 1 test.mp4

Alternative method Yolo v3 COCO - image: darknet.exe detect cfg/yolov3.cfg yolov3.weights -i 0 -thresh 0.25

Train on **Amazon EC2**, to see mAP & Loss-chart using URL like: http://ec2-35-160-228-91.us-west-2.compute.amazonaws.com:8090 in the Chrome/Firefox (**Darknet should be compiled with OpenCV**): ./darknet detector train cfg/coco.data yolov3.cfg darknet53.conv.74 -dont\_show -mjpeg\_port 8090 -map

186 MB Yolo9000 - image: darknet.exe detector test cfg/combine9k.data cfg/yolo9000.cfg yolo9000.weights

Remeber to put data/9k.tree and data/coco9k.map under the same folder of your app if you use the cpp api to build an app

To process a list of images data/train.txt and save results of detection to result.json file use: darknet.exe detector test cfg/coco.data cfg/yolov3.cfg yolov3.weights -ext\_output -dont\_show -out result.json < data/train.txt

To process a list of images data/train.txt and save results of detection to result.txt use:  
darknet.exe detector test cfg/coco.data cfg/yolov3.cfg yolov3.weights -dont\_show -ext\_output < data/train.txt > result.txt

Pseudo-lableing - to process a list of images data/new\_train.txt and save results of detection in Yolo training format for each image as label <image\_name>.txt (in this way you can increase the amount of training data) use: darknet.exe detector test cfg/coco.data cfg/yolov3.cfg yolov3.weights -thresh 0.25 -dont\_show -save\_labels < data/new\_train.txt

To calculate anchors: darknet.exe detector calc\_anchors data/obj.data -num\_of\_clusters 9 -width 416 -height 416

To check accuracy mAP@IoU=50: darknet.exe detector map data/obj.data yolo-obj.cfg backup\yolo-obj\_7000.weights

To check accuracy mAP@IoU=75: darknet.exe detector map data/obj.data yolo-obj.cfg backup\yolo-obj\_7000.weights -iou\_thresh 0.75

##### For using network video-camera mjpeg-stream with any Android smartphone

1. Download for Android phone mjpeg-stream soft: IP Webcam / Smart WebCam
   * Smart WebCam - preferably: <https://play.google.com/store/apps/details?id=com.acontech.android.SmartWebCam2>
   * IP Webcam: <https://play.google.com/store/apps/details?id=com.pas.webcam>
2. Connect your Android phone to computer by WiFi (through a WiFi-router) or USB
3. Start Smart WebCam on your phone
4. Replace the address below, on shown in the phone application (Smart WebCam) and launch:

Yolo v3 COCO-model: darknet.exe detector demo data/coco.data yolov3.cfg yolov3.weights http://192.168.0.80:8080/video?dummy=param.mjpg -i 0