**Setting up your YOLO/Darknet Repo for training and inference**

**Get Darknet and build the DLLs/EXEs:**

* First, checkout the RavvenNet modified repo for Darknet and save in a folder such as D:\yolo

<https://github.com/ravvenlabs/darknet>

* This Repo has:
  + DLLs and EXEs all pre-compiled with specific versions of tools
  + Specific scripts and projects that` have all been set up and tested so that you can evaluate the tools right from the start
  + Large files like data, weights, and openCV will be up in the pcloud and can be accessed with the links provided

**Download additional files that are too large for GIT**

* Download the weights files for yolo v2 tiny. More models can be found on the darknet page.
  + [yolov2-tiny.weights](http://u.pc.cd/BHX)
  + Place the weights in the “D:\yolo\darknet\build\darknet\x64\weights” directory
* Next, download this [test.mp4](http://u.pc.cd/59PctalK) file:
  + Place it in: “D:\yolo\darknet\build\darknet\x64\data”
* Download the following three tar files for the PASCAL VOC dataset by following these links:
  + [VOCtest\_06-Nov-2007.tar](http://u.pc.cd/YBSctalK)
  + [VOCtrainval\_06-Nov-2007.tar](http://u.pc.cd/WpK7)
  + [VOCtrainval\_11-May-2012.tar](http://u.pc.cd/U2l7)
* Extract all three to D:/Yolo/darknet/build/darknet/x64/data/ The contents should be now be in the folder: D:/Yolo/darknet/build/darknet/x64/data/VOCdevkit/
  + Inside will be 2 folders. One for 2007 and one for 2012
* Download pretrained weights for Darknet53 to work with for training. The link to follow is:
  + [darknet53.weights](http://u.pc.cd/7wM7)
  + Place this in D:/Yolo/darknet/build/darknet/x64/pretrained/
* To download the already trained yolov2-tiny-voc weights use this pcloud link:
  + [yolov2-tiny-voc.weights](http://u.pc.cd/ryT)
  + Place these in D:\yolo\darknet\build\darknet\x64\weights
  + This is for the mAP evaluation step on VOC data

**Before you start:**

* You will need to have installed cuda 10.2 and visual studio 2019 on your PC and have followed the setup instructions in the visual studio setup guide. You will also need to have installed OpenCV v4.2.0 and python3.7 through the provided anaconda installer. (**These steps are from the visual studio setup document)**

**What’s provided:**

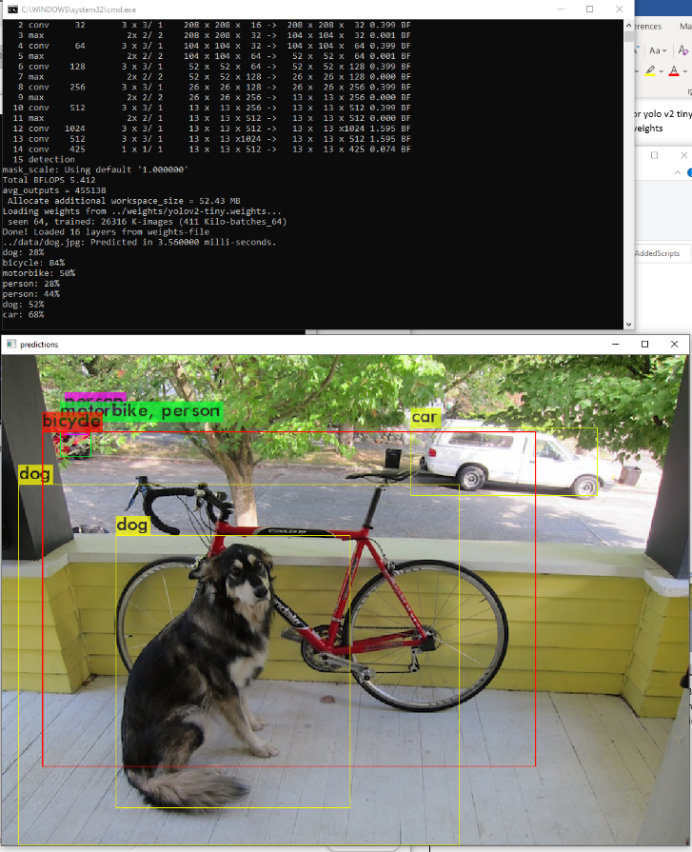
* The contents of this project allow for:
  + Training of a yolov2-tiny-voc model
  + Single image inference with a yolo-tiny-voc model using darknet (EXE)
  + Single image inference with a python wrapped darknet library (DLL)
  + Video inference with a python wrapped darknet library (DLL)
  + mAP testing scripts with 2 different models (EXE)
  + VS2019 Python projects for mixed debugging (DLL)
* There is no need to build the libraries and executables. But there is an option to do so.

**Verify that YOLO/Darknet work and the required environment is properly installed on your computer with the following scripts and examples**

**Download a model and do a basic prediction:**

* Run the dog\_test.cmd script from “D:\yolo\darknet\build\darknet\x64\ravvenAddedScripts\” to use darknet.exe and predict bounding boxes and classifications for a test image of a dog with Yolov2-tiny.

Here is an output you can expect:



**Using python wrapped Darknet for video and image inference:**

* As a temporary solution, install OpenCV 4.2.0.34 for python as well. Without this, you will not be able to import CV2 for darknet\_video.py. If you choose to build OpenCV instead of download binaries, this is not needed as you can link to the proper python interpreter with CMAKE
  + **Here are the commands to download OpenCV 4.2.0 for python with anaconda:** 
    - Activate anaconda: >
      * Open command prompt
      * cd to anaconda directory. Note: Directory may be different. Use your anaconda 3 installation directory.

cd D:\Anaconda3\Scripts

* + - * activate anaconda virtual environment

activate base

python -m pip install opencv-python==4.2.0.34

python -m pip install opencv-contrib-python==4.2.0.34

* Run the darknet.py script from “D:\yolo\darknet\build\darknet\x64\” to use the yolo\_gpu\_dll and predict bounding boxes and classifications for a test image. This can be done from the command line.

cd D:\yolo\darknet\build\darknet\x64

python darknet.py

Here is an output you can expect:

A screenshot of a computer

Description automatically generated

* CPU mode:
  + Running darknet.py does eval with the gpu based dll. Read the directions in darknet.py and follow one of the 3 steps within the darknet.py script to set an environment variable that will force cpu use only. This may be nice for benchmarking/comparisons in the future but isn’t really needed. ( I had used the environment variable FORCE\_CPU, and changing this requires a VS restart to take effect)
  + Output below:

A screenshot of a computer

Description automatically generated

**Darknet Video:**

* Once darknet.py works for at least gpu inference, run **darknet\_video.py** from the command line
* This uses darknet.py to do the same prediction on a video sequence. This script also uses opencv (cv2) which should be installed already. This is installed with either a “visual studio setup guide” custom build process or by downloading it as shown in this doc.
* Run darknet\_video.py from the command line. If the video pops up with bounding boxes and updates positions as the video plays, then everything is working as it should.

A screen shot of a computer

Description automatically generated

**Training a yolo model with float32 on gpu w/ PASCAL VOC:**

* This next set of instructions will set up the VOC dataset and show how to train a model
* **Data:**
* Download the three tar files for the PASCAL VOC dataset if you haven’t done so (3 links at the top of doc)
* Extract all three to D:/Yolo/darknet/build/darknet/x64/data/ The contents should be now be in the folder: D:/Yolo/darknet/build/darknet/x64/data/VOCdevkit/
  + Inside will be 2 folders. One for 2007 and one for 2012
* Labels need to be generated for VOC data. Run the provided file: **voc\_label\_edit\_windows.py** in the directory: D:/YOLO/darknet/build/darknet/x64/data/
* After running, labels are generated and the file train.txt contains the train and validation data from 2007 and 2012 all in one.
* **Configs for Pascal VOC:**
* Open the file D:/YOLO/darknet/build/darknet/x64/cfg/brian\_voc.data and replace the paths to the train and test data with what is on your machine. And make sure it matches the content below

A screenshot of a cell phone

Description automatically generated

* Next, **run this script:**
  + train\_voc\_v2.cmd which is in D:/Yolo/darknet/build/darknet/x64/ravvenAddedScripts/
* Continuing from a checkpoint just requires that you point to a weights file you’d like to start at. (Check the backup folder for these after training and just edit the script)
* Running the .cmd should start the training and it will look like this:

A screenshot of a computer

Description automatically generated

**Notes on training:**

* When you start training the first time, open some sort of resource monitor to make sure your GPU is in use and you are using it fully.
  + You can use nvidia-smi in CMD if you have it or just use Task manager and look at the performance tab.
  + If there is room for more data to be processed, you can increase your batch size in your cfg file.
    - D:/Yolo/darknet/build/darknet/x64/cfg/ yolov2-tiny-voc.cfg
  + To do this, edit your model config (.cfg) file and change your batch size in the net section at the top of the file
* This might take a couple days so you can download one that has already been trained for the next step (Check top of doc)

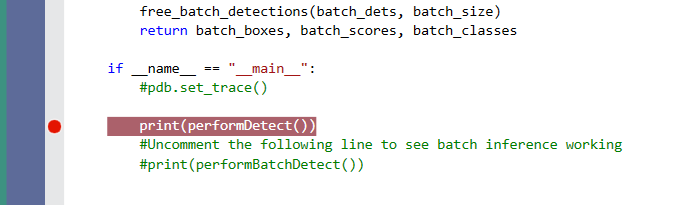
**Evaluate mAP:**

* Once you finish training yolov2-tiny-voc, you can run the yolov2 mAP evaluation script: yolov2-voc-map.cmd in ravvenAddedScripts
* There is a yolov3 version of this script too if you’d like to evaluate that.
* Results will be printed out in a text section when complete

**Mixed debug mode for python and C++:**

* If you would like to debug the DLLs with mixed C and python, there is a project set up for that. Go to “.darknet\build\darknet\x64\mixed\_debug\_darknet\” and open **mixed\_debug\_darknet.sln**
* Place a breakpoint atthe main functionwhich is where the python code calls a C ++ library function in the **YOLO\_GPU\_DLL.dll**

(Breakpoint placed BY CLICKING ON THE GRAY VERTICAL BAR)



* From here, step through the code with F11. You will now be able to step into the C code when you hit a library function.
  + Some library functions to step into are:
    - Load\_net\_main()
    - Detect()
* When inside, you can look at python and C variables in the watch lists
  + Normal operation will take longer in visual studio during debugging

**Resources:**

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

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

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