1. Setting Up:

To Install Darknet's YOLOv3 and use it to train your own custom Neural Network, you'll have to first install a couple dozen libraries, drivers and other auxiliary programs.

Some of the steps in the following tutorials overlap. It's ok, Ubuntu recognizes when you try to install duplicates, so if you're not sure if you've already installed something then install it again.

1.1. Install CUDA 10.0 and cuDNN 7.5.0 on Ubuntu 18.04 LTS

1.2. Install OpenCV 3.4.4

We suggest following this tutorial to the letter. There are newer versions of OpenCV, but we had a lot of issues with installing them so we settled for 3.4.4 which is not really outdated anyway.

1.3. Install TensorFlow-1.14.0 for GPU

By now you should already be using a virtual environment[1] for all the python packages. Tensorflow should be installed in the same virtual environment.

1.4. Install Keras

According to the documentation[2], Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. The main point of Keras is fast prototyping with Neural Networks.

1.5. Download and Install Darknet with CUDA

Darknet is an open-source Neural Network framework written in C & CUDA[3]. It uses the YOLO architecture and it's pretty good[4] at Object Detection. Essentially it's the best choice for real-time object detection because it's incredibly fast.

Potential issues:

a) Error making with cuda enabled
b) Error when running darknet to detect objects: Cuda Out of Memory

---

[1] How Virtual Environments work
[2] Keras homepage
[3] Darknet
[4] Object Detection - Comparison between prominent architectures

2. Training

Now that you've installed darknet and have a completed dataset,it's time to train.
This part is relatively simple. Follow the steps here.

After you're finished with the above steps you should have the following files ready:

a) data/obj.names: contains the names of your classes.
b) data/obj.data: contains the meta-data of your dataset.
c) Yolo-obj.cfg: contains the configuration of the yolo model suited for your dataset.
d) Darknet53.conv.74: contains the initial, pre-trained weights for your model.
e) data/Train.txt & data/Test.txt: those will be created by yolo-mark and will contain names of the images you want to train your model with.
f) data/obj/folder: the folder that contains all your images alongside the annotation .txt file created by yolo-mark.

Now you can begin training:

$ ./darknet detector train data/obj.data yolo-custom.cfg yolo.weights

You should wait for about 2000~ iterations per class or until the Average Loss is about ~0.1.
It's also possible to change how often the weights are saved in the backup/ folder. Go to line 136 of examples/detector.c and change this condition:

```
if(i%10000==0 || (i < 1000 && i%100 == 0))
```

3. Testing the Model

You should find your custom weights in the backup/ folder. The weights are saved after 100 or 1000 iterations.
E.g.: backup/yolov3-tiny-obj_1000.weights are the weights you get after 1000 iterations.

Some notes before you begin:

- You should probably have a few images separated from the training dataset as a "validation dataset" so that you can check your results on those.
- Testing your model on the training dataset only serves to verify that there were no errors during training, as the model is fitted to these images (therefore the error should be minimal).
- The results can only be as good as the annotations. Remember that the model is just mathematics. It doesn't know what a traffic cone is, is only knows what you annotated. Some possible pitfalls:
    a) Forgot to annotate some images
    b) Only annotated part of an image
    c) The bounding boxes were too loose or too small
- The bigger the dataset the better. Generally speaking, in a sophisticated deep neural network (such as darknet) that uses batch normalization and other techniques, too much data will not really cause overfitting.
- However, overfitting could be an issue when you have a small dataset but train for too long. For darknet, train for about ~2000 iterations / class
- Also, remember to train with the *-map* option if you have a validation set so if you over-train, you'll know.

3.1. Testing on images

Simply run:

./darknet detector test data/obj.data yolo-custom.cfg yolo-obj_final.weights

After loading, it will ask you for the filepath to any image you wish to process.

3.2. Testing on video

You can download and cut a video from youtube directly from here.

Similarly, run:

./darknet detector demo data/obj.data yolov3-tiny-obj.cfg backup/yolov3-tiny-obj_5000.weights data/ytVideo.mp4