Darknet and YOLO

Darknet is a framework to train neural networks, it is open source and written in C/CUDA and serves as the basis for YOLO.
Classifying whether an image is that of a cat or a dog is one problem, detecting the cats and the dogs in your image and their locations is a different problem. While the first problem can be solved by using neural networks as classifiers, effectively determining which class an image belongs to, amongst a selection, the second problem is quite different and requires a different approach. YOLO is a powerful neural net that does exactly that: it will tell you what is in your image giving the bounding box around the detected objects.

Darknet_ros

This package helps you to detect and classify various objects in a frame.
This is for Ardupilot Users.

Installations

First we need a plugin for direct connection of ros with gazebo.

```
sudo apt install ros-melodic-gazebo-ros ros-melodic-gazebo-plugins
```

This following repo contains some world files and it's important to have some world files to be there in your catkin_ws to use with ros.

```
cd ~/catkin_ws/src
git clone https://github.com/Intelligent-Quads/iq_sim.git
cd ~/catkin_ws
catkin build
Source ~/.zshrc
```

Now to launch gazebo+ros,

```
roslaunch iq_sim runway.launch
```

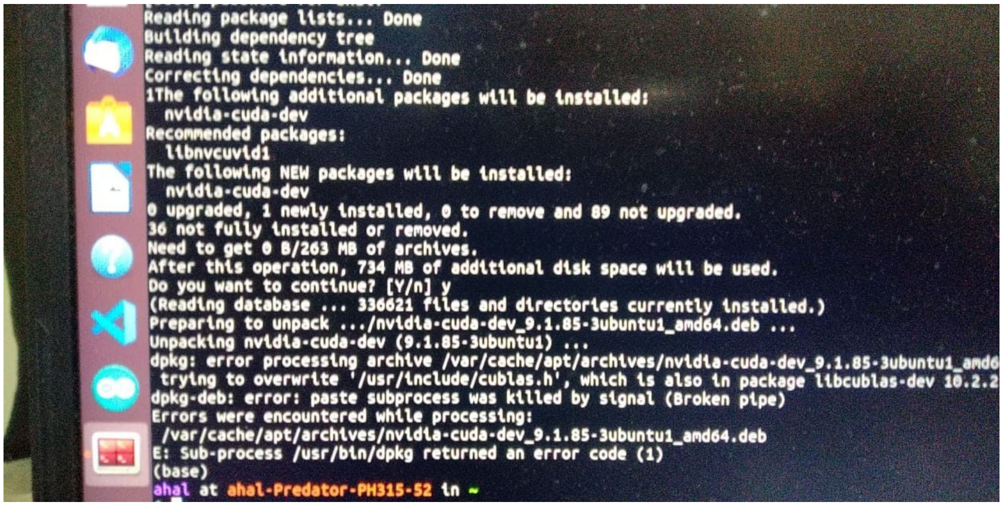
Make changes in your environment and save changes.

Now before installing darknet_ros, you need to install cuda package to enable darknet to work on gpu because the network is computationally expensive and needs a gpu for its full potential.

https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html
Don't forget to do the post installation steps too.
https://docs.nvidia.com/deeplearning/sdk/cudnn-install/index.html

Go to this site for cudnn installation which is an effective library for deep neural networks.

If you encounter with errors like broken packages,

Eg: 

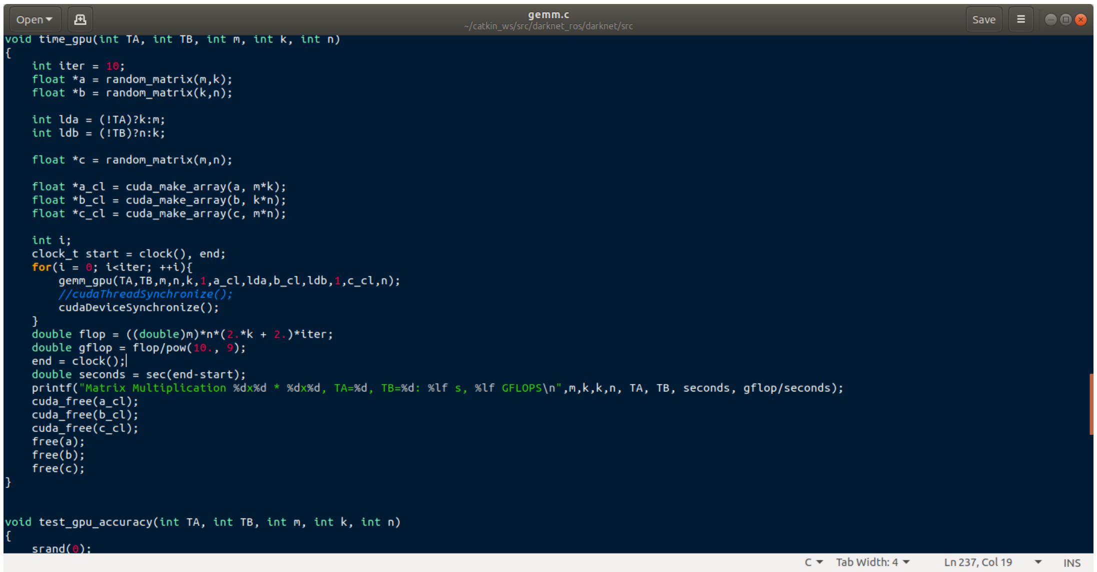
```
Reading package lists... Done
Building dependency tree
Reading state information... Done
Correcting dependencies... Done
1The following additional packages will be installed:
  nvidia-cuda-dev
Recommended packages:
  libnvcuvid1
The following NEW packages will be installed:
  nvidia-cuda-dev
0 upgraded, 1 newly installed, 0 to remove and 89 not upgraded.
36 not fully installed or removed.
Need to get 0 B/263 MB of archives.
After this operation, 734 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
(Reading database ... 336621 files and directories currently installed.)
Preparing to unpack .../nvidia-cuda-dev_9.1.85-3ubuntu1_amd64.deb ...
Unpacking nvidia-cuda-dev (9.1.85-3ubuntu1) ...
dpkg: error processing archive /var/cache/apt/archives/nvidia-cuda-dev_9.1.85-3ubuntu1_amd6
 trying to overwrite '/usr/include/cublas.h', which is also in package libcublas-dev 10.2.2
dpkg-deb: error: paste subprocess was killed by signal (Broken pipe)
Errors were encountered while processing:
 /var/cache/apt/archives/nvidia-cuda-dev_9.1.85-3ubuntu1_amd64.deb
E: Sub-process /usr/bin/dpkg returned an error code (1)
(base)
ahal at ahal-Predator-PH315-52 in ~
```

Uninstall cuda with
dpkg -l | grep cuda- | awk '{print $2}' | xargs -n1 sudo dpkg --purge --force-all

If you are getting warnings like this
dpkg: warning: while removing cuda-nvtx-10-2, directory '/usr/local/cuda-10.2/targets/x86_64-linux/include' not empty so not removed
dpkg: warning: while removing cuda-nvtx-10-2, directory '/usr/local/cuda-10.2/targets/x86_64-linux/lib' not empty so not removed
Then
Rm -rf your cuda folder, it will fix this.

Now clone the darknet_ros repo in your workspace
cd ~/catkin_ws/src
git clone --recursive https://github.com/leggedrobotics/darknet_ros.git

Now go to catkin_ws/src/darknet_ros/darknet/src , open gemm.c and change
/cudaThreadSynchronize(); to
cudaDeviceSynchronize();



```
void time_gpu(int TA, int TB, int m, int k, int n)
{
    int iter = 10;
    float *a = random_matrix(m,k);
    float *b = random_matrix(k,n);

    int lda = (!TA)?k:m;
    int ldb = (!TB)?n:k;

    float *c = random_matrix(m,n);

    float *a_cl = cuda_make_array(a, m*k);
    float *b_cl = cuda_make_array(b, k*n);
    float *c_cl = cuda_make_array(c, m*n);

    int i;
    clock_t start = clock(), end;
    for(i = 0; i<iter; ++i){
        gemm_gpu(TA,TB,m,n,k,1,a_cl,lda,b_cl,ldb,1,c_cl,n);
        //cudaThreadSynchronize();
        cudaDeviceSynchronize();
    }
    double flop = ((double)m)*n*(2.*k + 2.)*iter;
    double gflop = flop/pow(10., 9);
    end = clock();
    double seconds = sec(end-start);
    printf("Matrix Multiplication %dx%d * %dx%d, TA=%d, TB=%d: %lf s, %lf GFLOPS\n",m,k,k,n, TA, TB, seconds, gflop/seconds);
    cuda_free(a_cl);
    cuda_free(b_cl);
    cuda_free(c_cl);
    free(a);
    free(b);
    free(c);
}


void test_gpu_accuracy(int TA, int TB, int m, int k, int n)
{
    srand(0);
```

catkin build -DCMAKE_BUILD_TYPE=Release
-DCMAKE_C_COMPILER=/usr/bin/gcc-6

In the file ros.yaml specifies ros parameters. You can find this file under darknet_ros/darknet_ros/config. You will need to change the image topic from /camera/rgb/image_raw to /webcam/image_raw.

The file darknet_ros.launch will launch the darknet/yolo ros node. You can find this file under darknet_ros/darknet_ros/launch

in this file you can choose which version of yolo you would like to run by changing

<arg name="network_param_file"   default="$(find darknet_ros)/config/yolov2-tiny.yaml"/>

the options are as follows

- yolov1: Not recommended. this model is old
- yolov2: more accurate, and faster.
- yolov3: about as fast as v2, but more accurate. Yolo v3 has a high GPU ram requirement to train and run. If your graphics card does not have enough ram, use yolo v2
- tiny-yolo: Very fast yolo model. Would recommend for applications where speed is most important. Works very well on Nvidia Jetson

Now roslaunch darknet_ros darknet_ros.launch will launch the node.

Tips:

I am attaching a model file iris drone with a camera with detectable objects.

```
        <sensor name='camera' type='camera'>
        <pose frame=''>0 0 0 -1.57 -1.57 0</pose>
```

Make changes in the 4th entry to adjust your camera.

Now to get the live video feed of camera in your drone, type this in a new terminal,

```
 rosrun image_view image_view image:=/webcam/image_raw
```

Darknet_ros uses unique topics to transfer data.
Rostopic list after running darknet_ros will reveal all topics.

Eg: C++

```
#include <darknet_ros_msgs/BoundingBoxes.h>

void detection_cb(const darknet_ros_msgs::BoundingBoxes::ConstPtr& msg)
{
        for( int i=0; i < msg->bounding_boxes.size(); i++)
        {
                ROS_INFO("%s detected", msg->bounding_boxes[i].Class.c_str());
        }
}
ros::Subscriber sub = n.subscribe("/darknet_ros/bounding_boxes", 1, detection_cb);
```

Python

```
from darknet_ros_msgs.msg import BoundingBoxes
        mid=data.bounding_boxes[i].Class
rospy.Subscriber("/darknet_ros/bounding_boxes", BoundingBoxes, callback)
```

For finding out the type of object it detected.