Project 3: Detecting Private Objects in Photos Using YOLO

Simeon Babatunde

YOLO Setup and Testing

The information needed to setup the YOLO platform was found on the website of darknet. It was accessed via ***https://pjreddie.com/darknet/yolo/***. The screenshots below demonstrates successful compilation and testing of YOLO.

- Download the YOLO from github and compile.

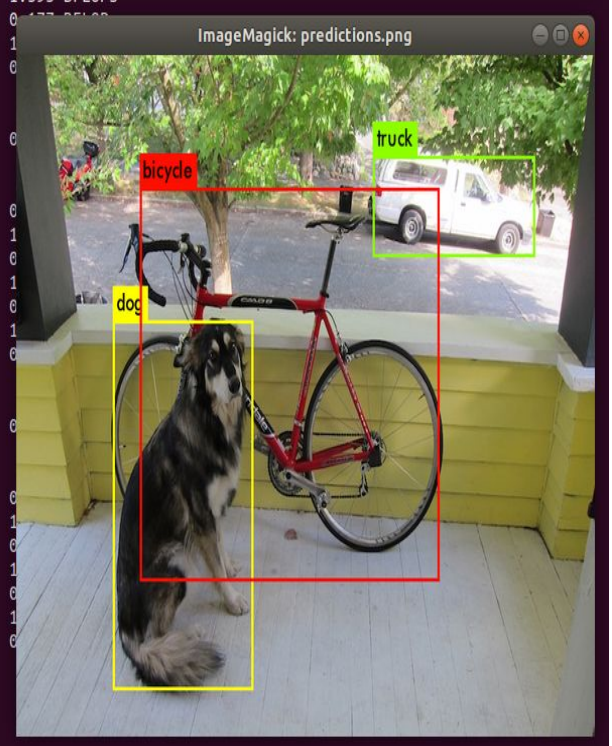
```
File  Edit  View  Search  Terminal  Help
simeon@simeon-HP-ENVY-Sleekbook-6:~/Documents$ git clone https://github.com/pjreddie/darknet
Cloning into 'darknet'...
remote: Counting objects: 5752, done.
remote: Total 5752 (delta 0), reused 0 (delta 0), pack-reused 5752
Receiving objects: 100% (5752/5752), 5.92 MiB | 1.19 MiB/s, done.
Resolving deltas: 100% (3872/3872), done.
simeon@simeon-HP-ENVY-Sleekbook-6:~/Documents$ cd darknet/
simeon@simeon-HP-ENVY-Sleekbook-6:~/Documents/darknet$ ls
cfg   examples  LICENSE       LICENSE.gen  LICENSE.meta  LICENSE.v1  python     scripts
data  include   LICENSE.fuck  LICENSE.gpl  LICENSE.mit   Makefile    README.md  src
simeon@simeon-HP-ENVY-Sleekbook-6:~/Documents/darknet$ make
mkdir -p obj
mkdir -p results
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/gemm.c -o obj/gemm.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/utils.c -o obj/utils.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/cuda.c -o obj/cuda.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/deconvolutional_layer.c
utional_layer.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/convolutional_layer.c -
nal_layer.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/list.c -o obj/list.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/image.c -o obj/image.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/activations.c -o obj/ac
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/im2col.c -o obj/im2col.
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/col2im.c -o obj/col2im.
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/blas.c -o obj/blas.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/crop_layer.c -o obj/cro
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/dropout_layer.c -o obj/
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/maxpool_layer.c -o obj/
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/softmax_layer.c -o obj/
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/data.c -o obj/data.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/matrix.c -o obj/matrix.
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/network.c -o obj/networ
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/connected_layer.c -o ob
r.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/cost_layer.c -o obj/cos
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/parser.c -o obj/parser.
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/option_list.c -o obj/op
```

- Download pre-trained weight file.

```
File  Edit  View  Search  Terminal  Help
simeon@simeon-HP-ENVY-Sleekbook-6:~/Documents/darknet$ wget https://pjreddie.com/media/files/yolov3.weights
--2018-04-07 16:57:26--  https://pjreddie.com/media/files/yolov3.weights
Resolving pjreddie.com (pjreddie.com)... 128.208.3.39
Connecting to pjreddie.com (pjreddie.com)|128.208.3.39|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 248007048 (237M) [application/octet-stream]
Saving to: ‘yolov3.weights’

yolov3.weights                      100%[===================================================================>] 236.52M  5.54MB/s    in 66s

2018-04-07 16:58:32 (3.60 MB/s) - ‘yolov3.weights’ saved [248007048/248007048]

simeon@simeon-HP-ENVY-Sleekbook-6:~/Documents/darknet$ ls
cfg      data      include       libdarknet.so  LICENSE.fuck  LICENSE.gpl   LICENSE.mit  Makefile  python     results  src
darknet  examples  libdarknet.a  LICENSE        LICENSE.gen   LICENSE.meta  LICENSE.v1   obj       README.md  scripts  yolov3.weights
simeon@simeon-HP-ENVY-Sleekbook-6:~/Documents/darknet$
```

- Run the detector on a sample image data

```
File  Edit  View  Search  Terminal  Help
   76 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024  1.595 BFLOPs
   77 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512  0.177 BFLOPs
   78 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024  1.595 BFLOPs
   79 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512  0
   80 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024  1
   81 conv    255  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 255  0
   82 detection
   83 route  79
   84 conv    256  1 x 1 / 1    13 x  13 x 512   ->    13 x  13 x 256  0
   85 upsample            2x    13 x  13 x 256   ->    26 x  26 x 256
   86 route  85 61
   87 conv    256  1 x 1 / 1    26 x  26 x 768   ->    26 x  26 x 256  0
   88 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512  1
   89 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256  0
   90 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512  1
   91 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256  0
   92 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512  1
   93 conv    255  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 255  0
   94 detection
   95 route  91
   96 conv    128  1 x 1 / 1    26 x  26 x 256   ->    26 x  26 x 128  0
   97 upsample            2x    26 x  26 x 128   ->    52 x  52 x 128
   98 route  97 36
   99 conv    128  1 x 1 / 1    52 x  52 x 384   ->    52 x  52 x 128  0
  100 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256  1
  101 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128  0
  102 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256  1
  103 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128  0
  104 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256  1
  105 conv    255  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 255  0
  106 detection
Loading weights from yolov3.weights...Done!
data/dog.jpg: Predicted in 11.621729 seconds.
dog: 99%
truck: 92%
bicycle: 99%
simeon@simeon-HP-ENVY-Sleekbook-6:~/Documents/darknet$ display predictions.png
```



Top 10 Private and Public Objects Computation

The following is a detailed explanation of the method used in computing the top 10 private and public objects in the set of images provided:

- Download and place the dataset in the darknet directory. Then, I created a python program to execute the detection commands and analyze the response in order to deduce the top 10 private and public objects.
- In this implementation, top private and public objects are the objects the highest number of occurence.
- In the implementation consist of a function called **detect_with_yolo(workingdir, imgpath, objtype)** This function accepts three arguments namely, the current working directory (darknet), the path to the image directory and string that describe the directory being accessed.
- The function implementation contains a **Popen()** method call which is used to execute the commands that run darknet with yolo weights and configurations. It also takes in the working directory and the image directory. The response from the **stdout** was parsed in order to extract the predicted objects. An ordered dictionary was used to keep track of the predicted objects. It was also manipulated to sort the objects in descending order based on the number of occurence. The screenshots below shows the output of python program:

```
[21:24:18] sbabatu@joey1:~/Documents/yolo/darknet [22]
[21:24:19] sbabatu@joey1:~/Documents/yolo/darknet [22] ./yolopredict.py
     Private Object        |   No of Occurence
##########################+#####################
         person            |          145
           cup             |           17
           dog             |           10
          chair            |           9
       wine glass          |           7
       cell phone          |           5
       diningtable         |           5
          book             |           4
           cat             |           4
         bottle            |           4
          knife            |           3
       sports ball         |           3
         bicycle           |           2
        tvmonitor          |           2
           car             |           2
           tie             |           2
          fork             |           2
         handbag           |           2
         carrot            |           1
          horse            |           1
        backpack           |           1
        microwave          |           1
          bench            |           1
      refrigerator         |           1
        sandwich           |           1
          vase             |           1
          sink             |           1
         remote            |           1
          sofa             |           1
           bed             |           1
        suitcase           |           1
```

```
      Public Object        |   No of Occurence
##########################+#####################
         person            |          97
         bottle            |          9
          chair            |          6
          knife            |          5
           cup             |          4
          book             |          4
          train            |          4
           car             |          4
         giraffe           |          4
          oven             |          3
          cake             |          3
           bus             |          2
       diningtable         |          2
           cat             |          2
         bicycle           |          1
          apple            |          1
         carrot            |          1
        keyboard           |          1
        tvmonitor          |          1
          boat             |          1
        backpack           |          1
           tie             |          1
        umbrella           |          1
          vase             |          1
         handbag           |          1
        broccoli           |          1
         hot dog           |          1
           bed             |          1
          truck            |          1


[22:00:27] sbabatu@joey1:~/Documents/yolo/darknet [23]
```

Table Depicting Top 10 Private and Public Objects

S/N	Private Objects	Public Objects
1	person	person
2	cup	bottle
3	dog	chair
4	chair	knife
5	wine glass	cup
6	cell phone	book
7	diningtable	train
8	book	car
9	cat	giraffe
10	bottle	oven

Observations about top 10 Private and Public Objects

Some of the identified observations include:

- Both top 10 private and public objects have 5 objects in common namely: person, cup, chair, book and bottle.
- Person appears to the most prominent object in both private and public photos
- About 90% of the top private objects are things that can be found in the home, while the public objects also contains some items that doesn't belong to the home

APPENDIX

```
#!/usr/bin/env python

from subprocess import Popen, PIPE
import os
from texttable import Texttable
from collections import OrderedDict
from operator import itemgetter

currentwd = os.getcwd()
private = currentwd + '/dataset/private'
public = currentwd + '/dataset/public'

def detect_with_yolo(workingdir, imgpath, objtype):
    mydict = {}

    #i = 0
    for image in os.listdir(imgpath):
        if(image.endswith(('.png', '.jpg', '.jpeg'))):
            p = Popen(['./darknet', 'detect', 'cfg/yolov3.cfg',
'yolov3.weights',
            imgpath + '/' + image], cwd = workingdir, stdout = PIPE,
stderr = PIPE)
            stdout, stderr = p.communicate()
            response = stdout.decode("utf-8")

            # Split the whole string by newline and extract the lines
with the predictions
            for item in response.split("\n"):
                if "%" in item:
                    obj = item.split(":")[0].strip()
                    if not obj in mydict:
                        mydict[obj] = 1
                    else:
                        mydict[obj] += 1
            # We just wanted to run for the first 10 images
            #i = i + 1
            #if(i == 10):
            #    break
```

```
    # Lets print a table
    tab = Texttable()
    header = [objtype +' Object', 'No of Occurence']
    tab.header(header)
    tab.set_cols_width([25,20])
    tab.set_cols_align(['c','c'])
    tab.set_cols_valign(['m','m'])
    tab.set_deco(tab.HEADER | tab.VLINES)
    tab.set_chars(['-','|','+','#'])

    # LEts Sort the Dictionary
    ordered = OrderedDict(sorted(mydict.items(), key=itemgetter(1),
reverse=True))
    for result in ordered:
        row = [result,ordered[result]]
        tab.add_row(row)

    # Lets draw the table
    s = tab.draw()    print s
    print "\n\n"

# Call the function for private images and public images

detect_with_yolo(currentwd, private, "Private")
detect_with_yolo(currentwd, public, "Public")
```