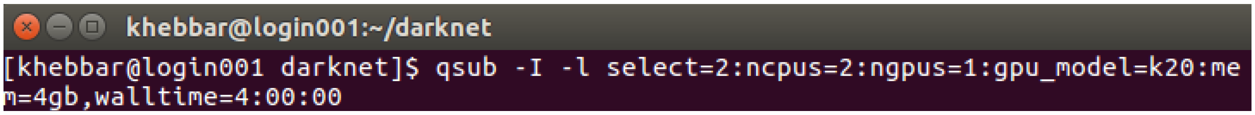
Detecting Private Objects in Photos Using YOLO

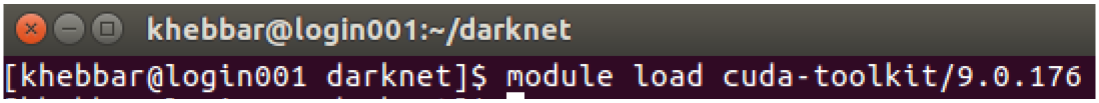
❖ Install Darknet with GPU

- Made use of Palmetto to run YOLO: Created an interactive job with 2 core CPU and a K20 GPU.



```
[khebbar@login001 darknet]$ qsub -I -l select=2:ncpus=2:ngpus=1:gpu_model=k20:me
m=4gb,walltime=4:00:00
```

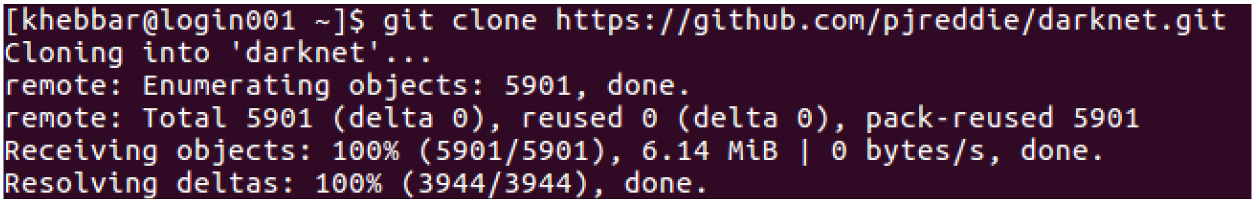
- The cuda module is loaded in palmetto:



```
[khebbar@login001 darknet]$ module load cuda-toolkit/9.0.176
```

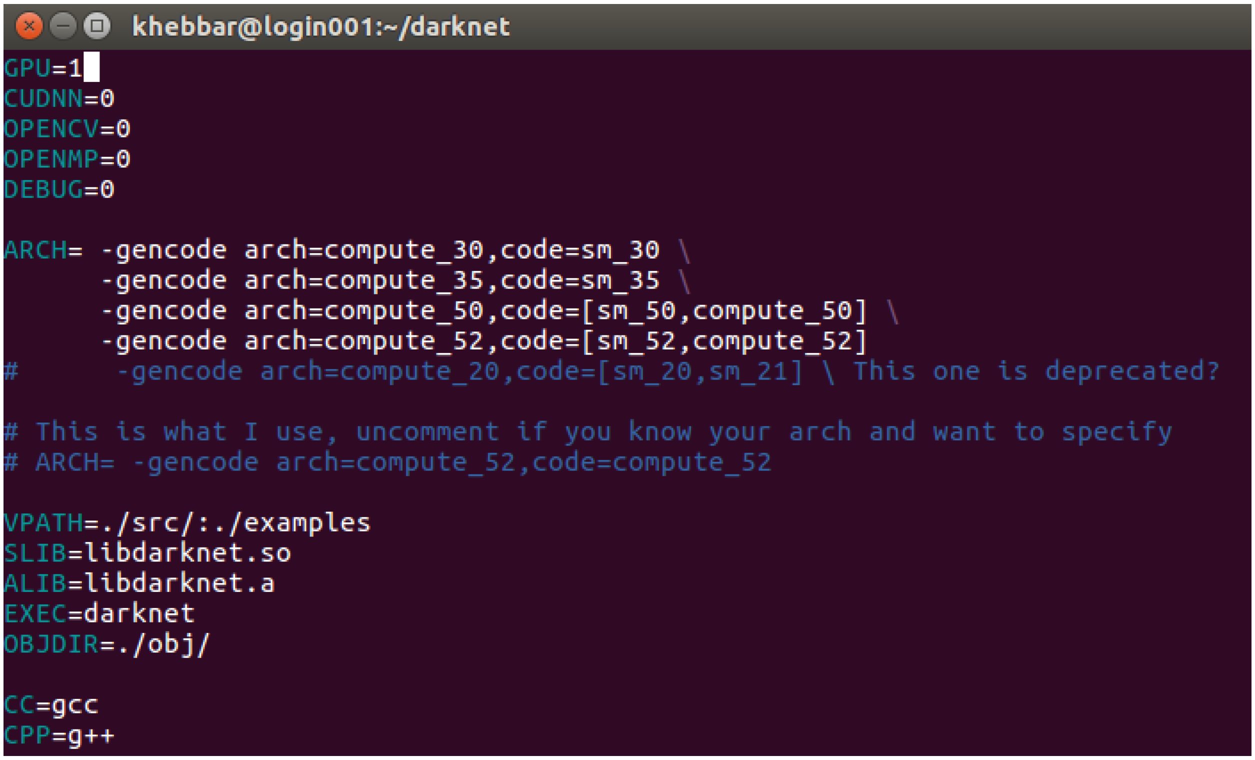
- Following steps are then taken to install darknet successfully

1. The github repository is first cloned:



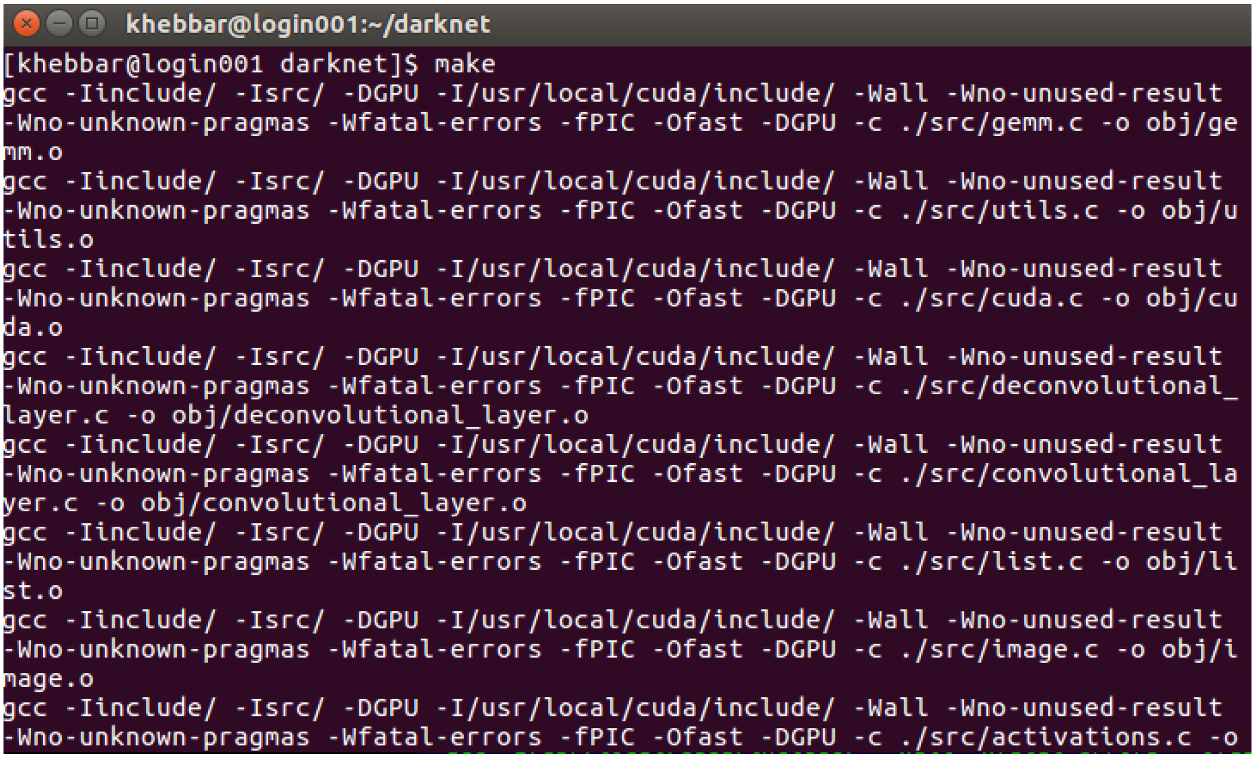
```
[khebbar@login001 ~]$ git clone https://github.com/pjreddie/darknet.git
Cloning into 'darknet'...
remote: Enumerating objects: 5901, done.
remote: Total 5901 (delta 0), reused 0 (delta 0), pack-reused 5901
Receiving objects: 100% (5901/5901), 6.14 MiB | 0 bytes/s, done.
Resolving deltas: 100% (3944/3944), done.
```

2. We then cd into darknet directory and alter the Makefile to make use of the gpu as follows:



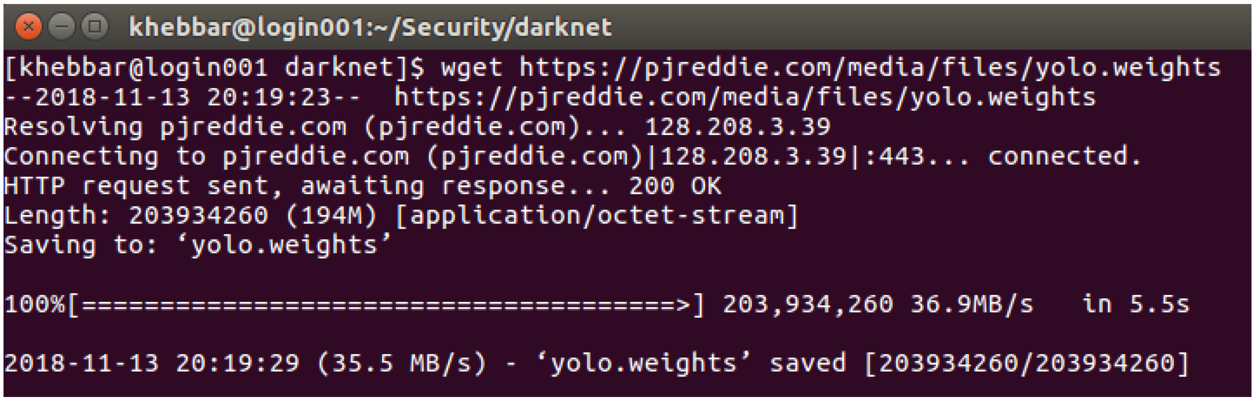
```
GPU=1
CUDNN=0
OPENCV=0
OPENMP=0
DEBUG=0

ARCH= -gencode arch=compute_30,code=sm_30 \
      -gencode arch=compute_35,code=sm_35 \
      -gencode arch=compute_50,code=[sm_50,compute_50] \
      -gencode arch=compute_52,code=[sm_52,compute_52]
#      -gencode arch=compute_20,code=[sm_20,sm_21] \ This one is deprecated?

# This is what I use, uncomment if you know your arch and want to specify
# ARCH= -gencode arch=compute_52,code=compute_52

VPATH=./src/:./examples
SLIB=libdarknet.so
ALIB=libdarknet.a
EXEC=darknet
OBJDIR=./obj/

CC=gcc
CPP=g++
```

3. We then make to install darknet:



```
[khebbar@login001 darknet]$ make
gcc -Iinclude/ -Isrc/ -DGPU -I/usr/local/cuda/include/ -Wall -Wno-unused-result
-Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -DGPU -c ./src/gemm.c -o obj/ge
mm.o
gcc -Iinclude/ -Isrc/ -DGPU -I/usr/local/cuda/include/ -Wall -Wno-unused-result
-Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -DGPU -c ./src/utils.c -o obj/u
tils.o
gcc -Iinclude/ -Isrc/ -DGPU -I/usr/local/cuda/include/ -Wall -Wno-unused-result
-Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -DGPU -c ./src/cuda.c -o obj/cu
da.o
gcc -Iinclude/ -Isrc/ -DGPU -I/usr/local/cuda/include/ -Wall -Wno-unused-result
-Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -DGPU -c ./src/deconvolutional_
layer.c -o obj/deconvolutional_layer.o
gcc -Iinclude/ -Isrc/ -DGPU -I/usr/local/cuda/include/ -Wall -Wno-unused-result
-Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -DGPU -c ./src/convolutional_la
yer.c -o obj/convolutional_layer.o
gcc -Iinclude/ -Isrc/ -DGPU -I/usr/local/cuda/include/ -Wall -Wno-unused-result
-Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -DGPU -c ./src/list.c -o obj/li
st.o
gcc -Iinclude/ -Isrc/ -DGPU -I/usr/local/cuda/include/ -Wall -Wno-unused-result
-Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -DGPU -c ./src/image.c -o obj/i
mage.o
gcc -Iinclude/ -Isrc/ -DGPU -I/usr/local/cuda/include/ -Wall -Wno-unused-result
-Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -DGPU -c ./src/activations.c -o
```

- The COCO dataset is then downloaded:

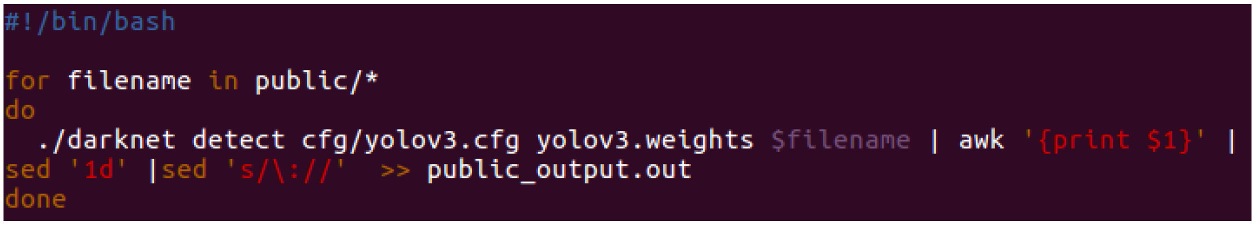


```
[khebbar@login001 darknet]$ wget https://pjreddie.com/media/files/yolo.weights
--2018-11-13 20:19:23--  https://pjreddie.com/media/files/yolo.weights
Resolving pjreddie.com (pjreddie.com)... 128.208.3.39
Connecting to pjreddie.com (pjreddie.com)|128.208.3.39|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 203934260 (194M) [application/octet-stream]
Saving to: ‘yolo.weights’

100%[======================================>] 203,934,260 36.9MB/s   in 5.5s

2018-11-13 20:19:29 (35.5 MB/s) - ‘yolo.weights’ saved [203934260/203934260]
```

- The public and private data was downloaded from: https://drive.google.com/open?id=1gZTtZMSNXsOumbrx4wF13bPc7fqwtBCq

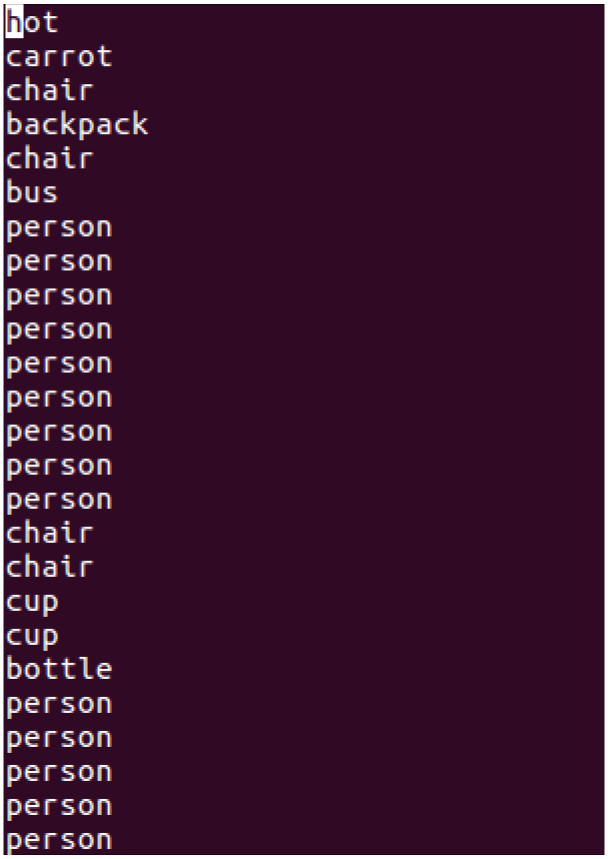
➢ PUBLIC DATA:

- The following bash script was executed on the public dataset to loop through each image, and redirect the output to a file called public_output.out:



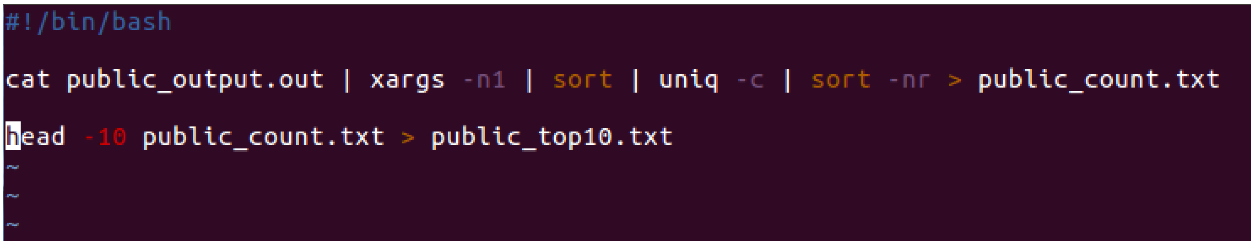
```
#!/bin/bash

for filename in public/*
do
  ./darknet detect cfg/yolov3.cfg yolov3.weights $filename | awk '{print $1}' |
sed '1d' |sed 's/\://'  >> public_output.out
done
```

- The output of the above script is as follows:



```
hot
carrot
chair
backpack
chair
bus
person
person
person
person
person
person
person
person
person
chair
chair
cup
cup
bottle
person
person
person
person
person
```

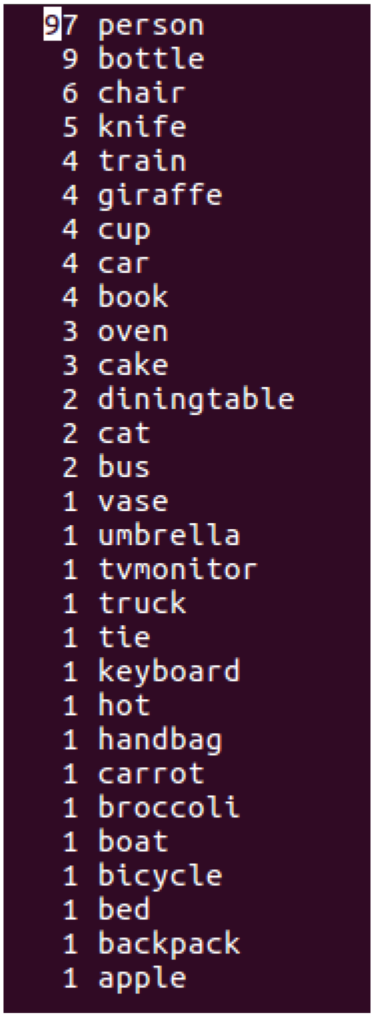
- The following shell script was written to count the number of occurrences of each object and redirect to public_count.txt file and list the top 10 objects in the public dataset and redirect the output of that to public_top10.txt file.



```
#!/bin/bash

cat public_output.out | xargs -n1 | sort | uniq -c | sort -nr > public_count.txt

head -10 public_count.txt > public_top10.txt
~
~
~
```

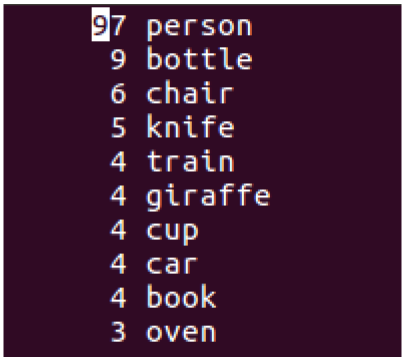
- The output of the above shell script is as follows:

1. Public_count.txt:



```
     97 person
      9 bottle
      6 chair
      5 knife
      4 train
      4 giraffe
      4 cup
      4 car
      4 book
      3 oven
      3 cake
      2 diningtable
      2 cat
      2 bus
      1 vase
      1 umbrella
      1 tvmonitor
      1 truck
      1 tie
      1 keyboard
      1 hot
      1 handbag
      1 carrot
      1 broccoli
      1 boat
      1 bicycle
      1 bed
      1 backpack
      1 apple
```

2. Public_top10.txt:

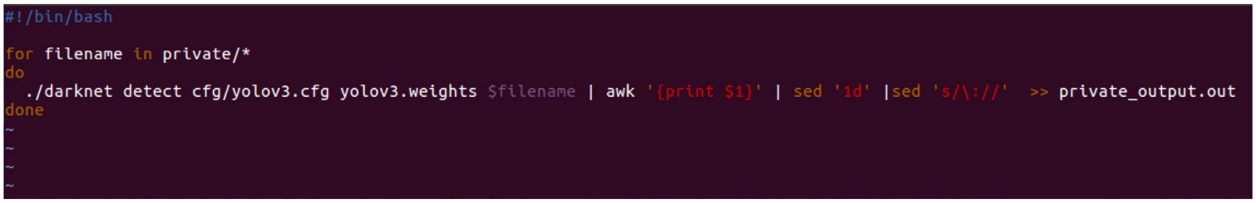


```
     97 person
      9 bottle
      6 chair
      5 knife
      4 train
      4 giraffe
      4 cup
      4 car
      4 book
      3 oven
```

OBJECT	**COUNT**
Person	97
Bottle	9
Chair	6
Knife	5
Train	4
Giraffe	4
Cup	4
Car	4
Book	4
Oven	3

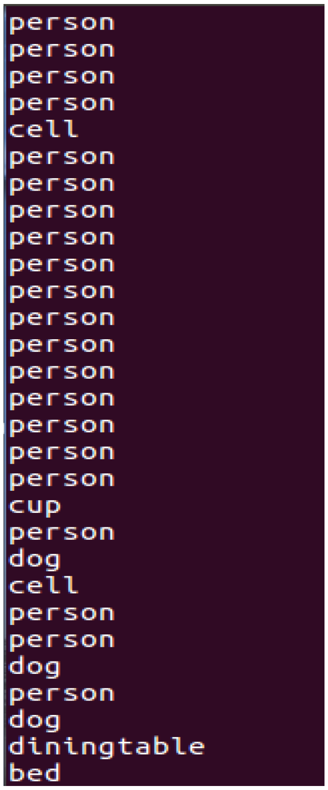
➢ PRIVATE DATA:

- The following bash script was executed on the private dataset to loop through each image, and redirect the output to a file called private_output.out:



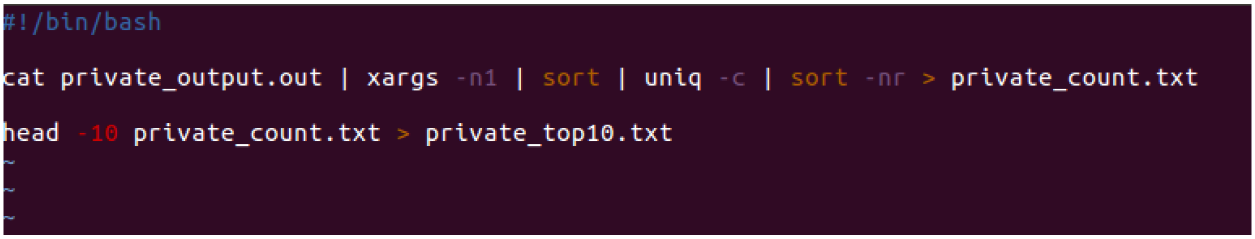
```
#!/bin/bash

for filename in private/*
do
  ./darknet detect cfg/yolov3.cfg yolov3.weights $filename | awk '{print $1}' | sed '1d' |sed 's/\://'  >> private_output.out
done
~
~
~
~
```

- The output of the above script is as follows:



```
person
person
person
person
cell
person
person
person
person
person
person
person
person
person
person
person
person
person
cup
person
dog
cell
person
person
dog
person
dog
diningtable
bed
```

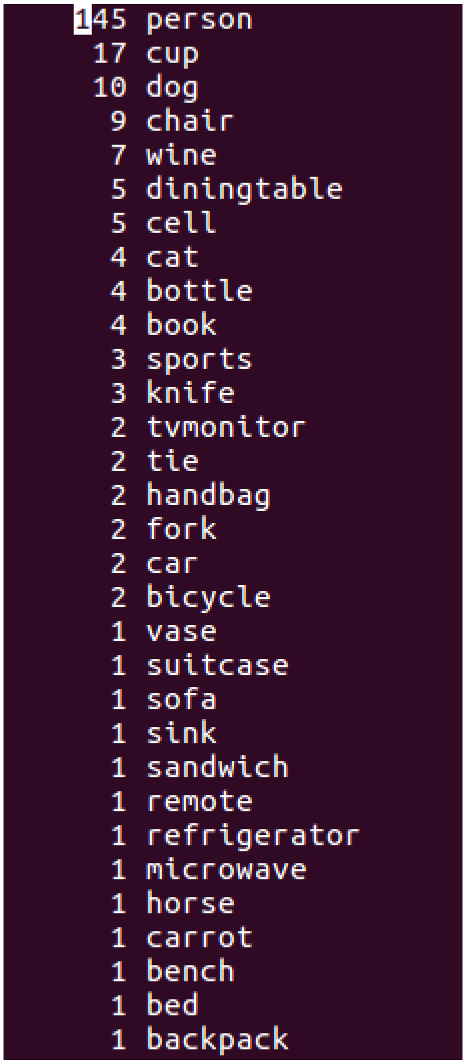
- The following shell script was written to count the number of occurrences of each object and redirect to private_count.txt file and list the top 10 objects in the private dataset and redirect the output of that to private_top10.txt file.



```
#!/bin/bash

cat private_output.out | xargs -n1 | sort | uniq -c | sort -nr > private_count.txt

head -10 private_count.txt > private_top10.txt
~
~
~
```

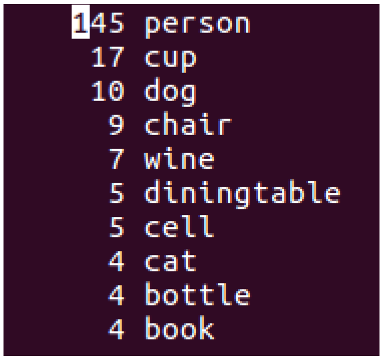
- The output of the above shell script is as follows:

1. Private_count.txt:



```
    145 person
     17 cup
     10 dog
      9 chair
      7 wine
      5 diningtable
      5 cell
      4 cat
      4 bottle
      4 book
      3 sports
      3 knife
      2 tvmonitor
      2 tie
      2 handbag
      2 fork
      2 car
      2 bicycle
      1 vase
      1 suitcase
      1 sofa
      1 sink
      1 sandwich
      1 remote
      1 refrigerator
      1 microwave
      1 horse
      1 carrot
      1 bench
      1 bed
      1 backpack
```

- Private_top10.txt:



```
    145 person
     17 cup
     10 dog
      9 chair
      7 wine
      5 diningtable
      5 cell
      4 cat
      4 bottle
      4 book
```

OBJECT	**COUNT**
Person	145
Cup	17
Dog	10
Chair	9
Wine	7
Dining table	5
Cell	5
Cat	4
Bottle	4
Book	4

❖ Observation about Top 10 public and private objects:

- Higher count of people in Private dataset
- Pets such as dogs and cats are made private
- Cup, wine, chair, dining table considered as indoor parts of a house are all considered as private
- Wine considered to be an alcoholic substance and hence might be classified as a private image