Ψηφιακή Επεξεργασία και Ανάλυση Εικόνας
Ακαδημαϊκό Έτος 2020-2021
Εργαστηριακή Ασκήση - Μέρος Β΄

***Ονοματεπώνυμο**: Βασιλική Στάμου* ***Έτος**: 4ο* ***ΑΜ**: 1059543*

Θέμα 1: Αξιολόγηση της απόδοσης Συνελικτικών Νευρωνικών Δικτύων κατά την ανίχνευση και κατηγοριοποίηση αντικειμένων κυκλοφοριακής σκηνής

Συνοπτική Περιγραφή του θέματος: Αξιολόγηση της απόδοσης Συνελικτικών Νευρωνικών Δικτύων (squeezeDet – squeezeDetPlus) κατά την ανίχνευση και κατηγοριοποίηση αντικειμένων κυκλοφοριακής σκηνής σε ό,τι αφορά τις μετρικές Intersection over Union (IoU), True Positive (TP), False Positive (FP), False Negative (FN), Precision, Recall και Area Under Curve (AUC).

Περιβάλλον Υλοποίησης: Η εργασία υλοποιήθηκε σε λογισμικό Linux-Ubuntu , στο περιβάλλον Anaconda για Python 3.8.8. Ο κώδικας (του github) πάνω στο οποίο έγινε η υλοποίηση ήταν σε Python 2.7 οπότε κρίθηκε αναγκαίο να αλλάξω κάποια πράγματα για να γίνει συμβατός. Σας έχω επισυνάψει το αρχείο demo.py σε περίπτωση που θέλετε να το τρέξετε. Η όλη επεξεργασία που έγινε σε Python για τα ζητούμενα της επεξεργασίας βρίσκεται στο αρχείο demo.py , αυτό το αρχείο σας παραθέτω στο παράρτημα του κώδικα στο τέλος της αναφοράς με επαρκή σχόλια για εύκολη κατανόηση.

Σχολιασμός Αποτελεσμάτων :

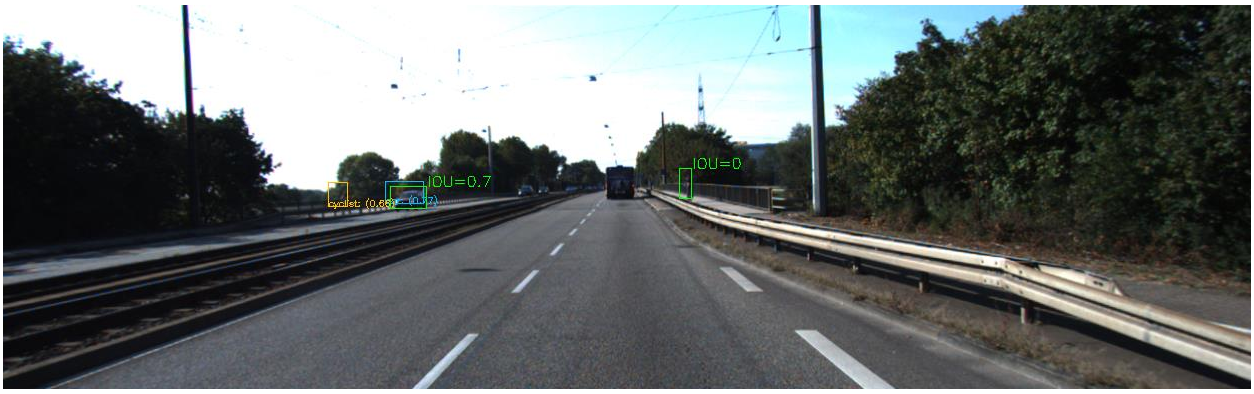
- Ενδεικτικά αποτελέσματα ανίχνευσης αντικειμένων που παράγει το SqueezeDet. Παρουσιάζεται η έξοδος (bounding boxes) του SqueezeDet συνδυαστικά με τα ground truth αντικείμενα (πράσινο χρώμα) ,το IOU για κάθε ανιχνευόμενο αντικείμενο και το IOU ανά frame.

Εικόνα 000000.png

```
iou for frame = 0.93
```

Εικόνα 000001.png

```
iou= [0.7, 0]

number of undetected objects =  1

iou for frame = 0.35
```

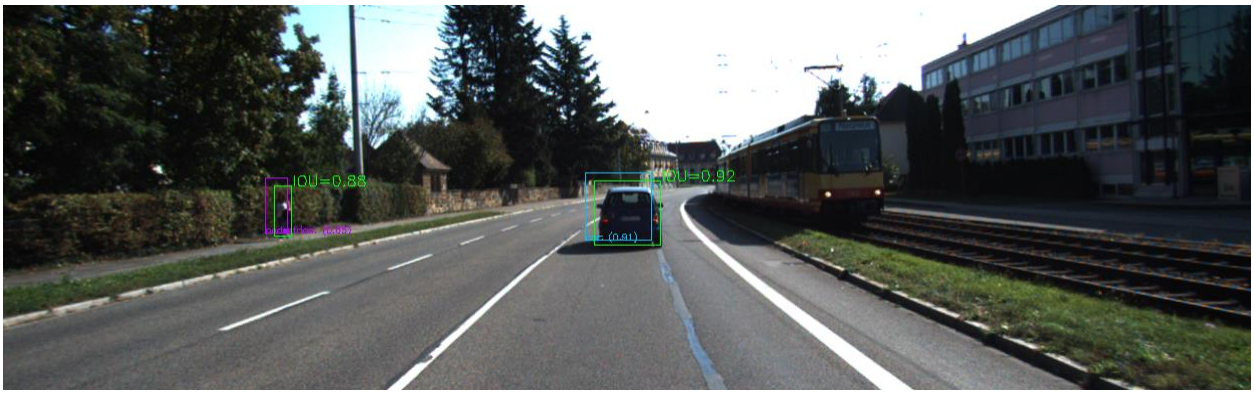


Εικόνα 000011.png

```
iou= [0.81, 0.97, 0.88, 0.89, 0.93, 0]

number of undetected objects =  1

iou for frame = 0.747
```

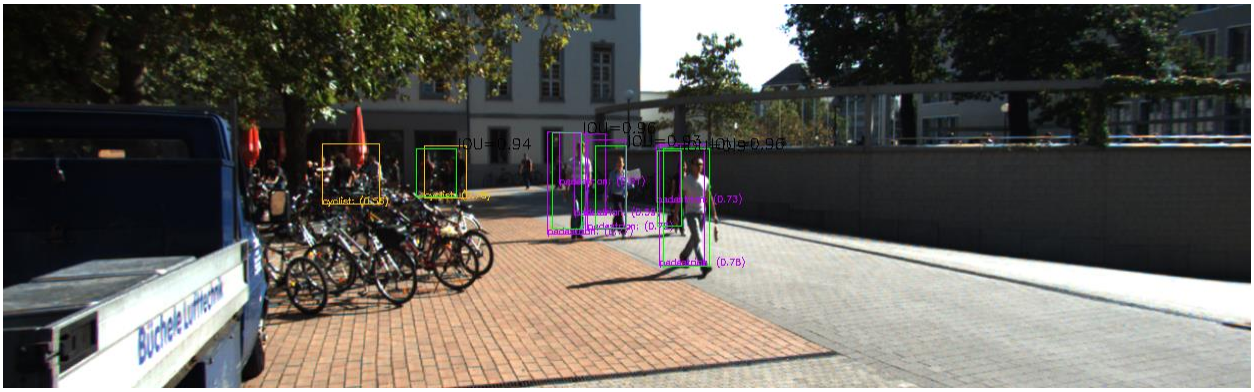
Εικόνα 000179.png

```
iou= [0.88, 0.92]

iou for frame = 0.9
```



Εικόνα 000189.png

```
iou= [0.94, 0.96, 0.93, 0.9, 0.96]

iou for frame = 0.938
```



Εικόνα 000195.png

```
iou= [0.92, 0.92]

iou for frame = 0.92
```

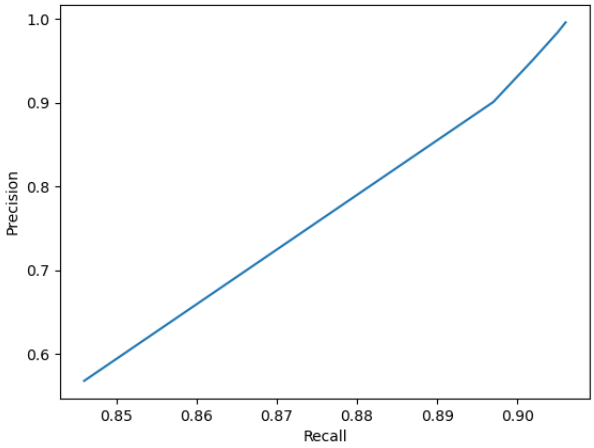
- Παρουσιάζεται παρακάτω το τελικό μέσο IOU του dataset ελέγχου για το SqueezeDet και το SqueezeDetPlus , επιλέχθηκαν 5 διαφορετικές τιμές IOU οι οποίες λειτουργούν ως κατώφλια

υπολογισμού των TP, FP, FN για το σύνολο των δεδομένων του dataset ελέγχου και για τα δύο δίκτυα. Επιπλέον ,και για τα δύο δίκτυα ,υπολογίστηκαν οι μητρικές Precision και Recall με βάση τον τελικό αριθμό TPs, FPs και FNs για κάθε κατώφλι. Τέλος κατασκευάστηκε η γραφική παράσταση Precision (-y) vs Recall (-x) και υπολογίστηκε το AUC με τον κανόνα του τραπεζίου και για τα δύο δίκτυα.

```
The five thresholds are =  [0.3, 0.5, 0.7, 0.8, 0.9]
```

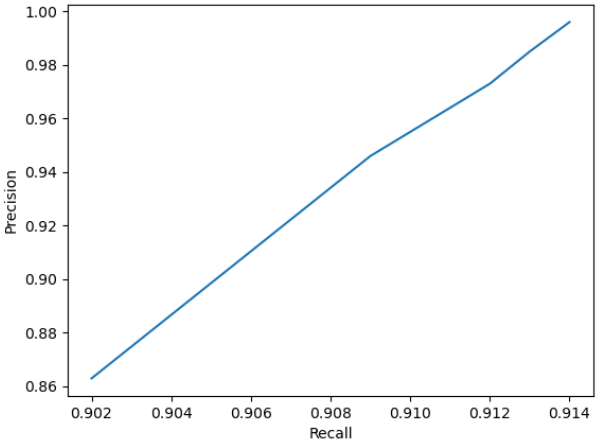
- **SqueezeDet :**

```
iou_dataset= 0.817

TP = [807, 797, 771, 730, 460]

FP = [3, 13, 39, 80, 350]

FN = [84, 84, 84, 84, 84]

Precision = [0.996, 0.984, 0.952, 0.901, 0.568]

Recall = [0.906, 0.905, 0.902, 0.897, 0.846]

AUC= 0.046

For model: squeezeDet
```



- **SqueezeDetPlus :**

```
iou_dataset= 0.864

TP = [814, 805, 795, 773, 705]

FP = [3, 12, 22, 44, 112]

FN = [77, 77, 77, 77, 77]

Precision = [0.996, 0.985, 0.973, 0.946, 0.863]

Recall = [0.914, 0.913, 0.912, 0.909, 0.902]

AUC= 0.011

For model: squeezeDet+
```



Αξιολόγηση απόδοσης δικτύων με βάση το IOU του dataset ελέγχου: Παρατηρούμε πως το squeezeDet έδωσε περίπου $\mathbf{IOU_{squeezeDet}=0.82}$, το squeezeDetPlus έδωσε καλύτερα αποτελέσματα , περίπου $\mathbf{IOU_{squeezeDetPlus}=0.86}$

Αξιολόγηση απόδοσης δικτύων με βάση το AUC για πέντε διαφορετικά κατώφλια: Παρατηρούμε πως το squeezeDet έδωσε περίπου $\mathbf{AUC_{squeezeDet}=0.05}$, το squeezeDetPlus έδωσε φτωχότερα αποτελέσματα , περίπου $\mathbf{AUC_{squeezeDetPlus}=0.01}$.Αξίζει να σημειωθεί πως και τα δύο AUC δίνουν κακά αποτελέσματα , αυτό δεν σημαίνει πως τα δύο αυτά δεν είναι καλά. Όπως παρατηρείται δύο από τα πέντε κατώφλια επιλέχθηκαν πολύ μικρά με σκοπό να φανεί ο καθοριστικό ρόλος της τιμής των thershold για την αξιολόγηση των μοντέλων μας , και εκεί αποδίδεται η κακή απόδοση AUC μετρικής.

Precision είναι η ικανότητα ενός κατηγοριοποιητή να αναγνωρίζει μόνο σχετικά αντικείμενα. Είναι το ποσοστό των σωστών θετικών προβλέψεων και δίνεται από τον λόγο TP /(TP+FP) ή TP/(all detections) .

Recall είναι μια μετρική που μετρά την ικανότητα ενός ταξινομητή να βρίσκει όλες τις σχετικές περιπτώσεις (δηλαδή όλα τα ground truths). Είναι το ποσοστό των πραγματικών θετικών (TP) που εντοπίζονται μεταξύ όλων των ground-truths.

PR Curve ή αλλιώς Precision – Recall Curve για μοντέλο με τέλειες ικανότητες απεικονίζεται σας σημείο στο (1,1), για επιδέξιο μοντέλο αντιπροσωπεύεται από μια καμπύλη που σκύβει προς τη συντεταγμένη (1,1), για ένα φτωχό μοντέλο θα είναι μια οριζόντια γραμμή .

AUC, μετρά ολόκληρη την δισδιάστατη περιοχή κάτω από ολόκληρη την καμπύλη Precision – Recall, όσο υψηλότερη είναι η καμπύλη στον άξονα y τόσο καλύτερη είναι η απόδοση του μοντέλου. Το AUC είναι scale-invariant, μετρά πόσο καλά κατατάσσονται οι προβλέψεις παρά τις απόλυτες τιμές τους. Επίσης το AUC είναι classification-threshold invariant, μετρά την ποιότητα των προβλέψεων του μοντέλου ανεξάρτητα από το threshold ταξινόμησης που επιλέγεται.

Κώδικας

```
# Author: Vasiliki Stamou 9/21

"""Implementation for the project

If you want to run squeezeDet for the whole dateset you give this command:

python ./src/demo.py --mode=dataset

If you want to run squeezeDetPlus for the whole dataset you give this command:

python ./src/demo.py --mode=dataset --
checkpoint=./data/model_checkpoints/squeezeDetPlus/model.ckpt-95000 --
out_dir=./data/out_squeezeDetPlus/ --demo_net=squeezeDet+

If you want to run squeezeDet for a single picture you give tis command:

python ./src/demo.py --mode=image --input_path=./data/KITTI/training/image_2/000195.png --
label_path=./data/KITTI/training/label_2/000195.txt

If you want to run squeezeDetPlus for a single picture you give thiw command:

python ./src/demo.py --mode=image --
checkpoint=./data/model_checkpoints/squeezeDetPlus/model.ckpt-95000 --
input_path=./data/KITTI/training/image_2/000000.png --
label_path=./data/KITTI/training/label_2/000000.txt --out_dir=./data/out_squeezeDetPlus/ --
demo_net=squeezeDet+

"""



from __future__ import absolute_import

from __future__ import division

from __future__ import print_function
```

```
import matplotlib.pyplot as plt
import cv2
import time
import sys
import os
import glob

import numpy as np
import tensorflow.compat.v1 as tf

from config import *
from train import _draw_box
from nets import *
from utils.util import iou,bbox_transform
from sklearn import metrics




FLAGS = tf.app.flags.FLAGS

tf.app.flags.DEFINE_string(
    'mode', 'image', """'image' or 'dataset'.""")
tf.app.flags.DEFINE_string(
    'checkpoint', './data/model_checkpoints/squeezeDet/model.ckpt-87000',
    """Path to the model parameter file.""")
tf.app.flags.DEFINE_string(
    'input_path', './data/sample.png',
    """Input image  to be detected. Can process glob input such as """
```

```
    """./data/00000*.png.""")
tf.app.flags.DEFINE_string(
    'label_path','./data/KITTI/training/label_2/000000.txt',"""Path for labels.""")
tf.app.flags.DEFINE_string(
    'out_dir', './data/out/', """Directory to dump output image .""")
tf.app.flags.DEFINE_string(
    'demo_net', 'squeezeDet', """Neural net architecture.""")
tf.app.flags.DEFINE_string(
    'image_dataset', './data/KITTI/training/image_2', """Image Dataset.""")
tf.app.flags.DEFINE_string(
    'label_dataset', './data/KITTI/training/label_2', """Label Dataset.""")


count=0
count_total=0




TP = [0] * 5
FP = [0] * 5
FN = [0] * 5

thr = []
thr.append(0.30)
thr.append(0.50)
thr.append(0.70)
thr.append(0.80)
thr.append(0.90)
```

```
def image_demo():
  """Detect image."""

  assert FLAGS.demo_net == 'squeezeDet' or FLAGS.demo_net == 'squeezeDet+', \
      'Selected nueral net architecture not supported: {}'.format(FLAGS.demo_net)

  with tf.Graph().as_default():
    # Load model
    if FLAGS.demo_net == 'squeezeDet':
      print("You chose sqeezeDet model")
      mc = kitti_squeezeDet_config()
      mc.BATCH_SIZE = 1
      # model parameters will be restored from checkpoint
      mc.LOAD_PRETRAINED_MODEL = False
      model = SqueezeDet(mc, FLAGS.gpu)
    elif FLAGS.demo_net == 'squeezeDet+':
      print("You chose sqeezeDet+ model")
      mc = kitti_squeezeDetPlus_config()
      mc.BATCH_SIZE = 1
      mc.LOAD_PRETRAINED_MODEL = False
      model = SqueezeDetPlus(mc, FLAGS.gpu)

    saver = tf.train.Saver(model.model_params)

    with tf.Session(config=tf.ConfigProto(allow_soft_placement=True)) as sess:
      saver.restore(sess, FLAGS.checkpoint)

      for f in glob.iglob(FLAGS.input_path):
        im = cv2.imread(f)
```

```
im = im.astype(np.float32, copy=False)
im = cv2.resize(im, (mc.IMAGE_WIDTH, mc.IMAGE_HEIGHT))
input_image = im - mc.BGR_MEANS

# Detect
det_boxes, det_probs, det_class = sess.run(
   [model.det_boxes, model.det_probs, model.det_class],
   feed_dict={model.image_input:[input_image]})

# Filter
final_boxes, final_probs, final_class = model.filter_prediction(
   det_boxes[0], det_probs[0], det_class[0])

keep_idx    = [idx for idx in range(len(final_probs)) \
               if final_probs[idx] > mc.PLOT_PROB_THRESH]
final_boxes = [final_boxes[idx] for idx in keep_idx]
final_probs = [final_probs[idx] for idx in keep_idx]
final_class = [final_class[idx] for idx in keep_idx]

cls2clr = {
   'car': (255, 191, 0),
   'cyclist': (0, 191, 255),
   'pedestrian':(255, 0, 191)
}

# Draw boxes
_draw_box(
   im, final_boxes,
   [mc.CLASS_NAMES[idx]+': (%.2f)'% prob \
```

```
        for idx, prob in zip(final_class, final_probs)],
    cdict=cls2clr,
)



# Get ground truth values from .txt
mylines = []
with open (FLAGS.label_path, 'rt') as myfile:
 for myline in myfile:
  mylines.append(myline.split())



# Keep objects of interest (We only care about Car, Pedestrian and Cyclist)
for i in range(len(mylines)):
 if mylines[i][0]=='DontCare' :
  del mylines[i][:]
 elif mylines[i][0]=='Truck' :
  del mylines[i][:]
 elif mylines[i][0]=='Misc' :
  del mylines[i][:]
 elif mylines[i][0]=='Van' :
  del mylines[i][:]
 elif mylines[i][0]=='Tram' :
  del mylines[i][:]
 else:
  print("\nObject of interest")
```

```
lst=[x for x in mylines if x]
print("\nlst=",lst)


# Our Grount Truth List
GroundTruthList = [ lst[i][4:8] for i in range(len(lst)) ]
GroundTruthList = [[int(float(j)) for j in i] for i in GroundTruthList]
print("\nGroundTruthList=",GroundTruthList)
print("\nLength of GroundTruthList= " ,len(GroundTruthList))



# Draw Ground Truth Box
c =(0,255,0)
for i in range(len(GroundTruthList)):
 cv2.rectangle(im, (GroundTruthList[i][0], GroundTruthList[i][1]), (GroundTruthList[i][2], GroundTruthList[i][3]), c, 1)



# Transform Predected Box to match Ground Truth
out_box = [ bbox_transform(final_boxes[i]) for i in range(len(final_boxes)) ]



#Sort GroundTruthList and out_box lists by the first element of each sublist
GroundTruthList=sorted(GroundTruthList, key=lambda x: x[0])
print("\nSorted GroundTruthList=",GroundTruthList)
out_box=sorted(out_box, key=lambda x: x[0])
```

```
print("\nSorted out_box=",out_box)



#In case predicted boxes are more than ground truth boxes ie. 0000003.png
val=500
if len(GroundTruthList) < len(out_box):
 size_diiference = len(out_box) - len(GroundTruthList)
 print("\nsize diff=",size_diiference)
 c=1
 k=[]*len(GroundTruthList)
 for i in range(len(GroundTruthList)):
  for j in range(len(out_box)):
   m = abs( GroundTruthList[i][0] - out_box[j][0])
   if val > m :
    val = m
    pos=j
 k.append(out_box[pos])
 val=500
 out_box = [x[:] for x in k]
 print("\nNew out_box=",out_box)
 print("\nNew out_box length=",len(out_box))



#In case model didn't predict groundtruth object
print("\nGroundTruthList length=",len(GroundTruthList))
print("\nout_box length=",len(out_box))
if len(GroundTruthList) == len(out_box):
 appr_difference=150
```

```
qq=[]
for i in range(len(out_box)):
 w = abs( out_box[i][0] - GroundTruthList[i][0])
 if appr_difference > w :
  pos=i
  qq.append(out_box[pos])
out_box = [x[:] for x in qq]
print("\nout_box length=",len(out_box))
print("\nout_box =",out_box)



# Calculate IOU per detected object and display it
if len(GroundTruthList) == len(out_box):
 IOU = [ iou(out_box[i],GroundTruthList[i]) for i in range(len(GroundTruthList)) ]
 IOU = [ round(IOU[i],2)  for i in range(len(IOU)) ]
 print("\niou=",IOU)
else :
 k = len(GroundTruthList) - len(out_box)
 diff=1000
 new_groundtruth=[]*len(out_box)
 for i in range(len(out_box)):
  for j in range(len(GroundTruthList)):
   m = abs( out_box[i][0] - GroundTruthList[j][0])
   if diff > m :
    diff = m
    pos=j
  new_groundtruth.append(GroundTruthList[pos])
  diff=1000
 print("\nnew_groundtruth =",new_groundtruth)
```

```
IOU = [ iou(out_box[i],new_groundtruth[i]) for i in range(len(out_box)) ]
for i in range(k):
 IOU.append(0)
IOU = [ round(IOU[i],2)  for i in range(len(IOU)) ]
print("\niou=",IOU)
print("\nnumber of undetected objects = ",k)



# Calculate IOU per frame and print it
IOU_per_frame = sum(IOU)/len(IOU)
print("\niou for frame =", round(IOU_per_frame, 3))



#Display IOU per detected object on picture
for i in range(len(IOU)):
 font = cv2.FONT_HERSHEY_SIMPLEX
 cv2.putText(im, "IOU=" +str(IOU[i]), (GroundTruthList[i][2]+1, GroundTruthList[i][1]), font, 0.5, c, 1)

global count
count = IOU_per_frame + count
print("\ncount=",count)



## Evaluation ##
global thr
global TP
global FP
global FN
```

```
        print("\nThe five thresholds are = ",thr)
        for i in range(len(IOU)):
         for j in range(len(thr)):
          if IOU[i] >= thr[j]:
           TP[j] = TP[j] + 1
          elif IOU[i] < thr[j] and IOU[i] != 0.00 :
           FP[j] = FP[j] + 1
          else:
           FN[j] = FN[j] + 1
        print("\nTP =",TP)
        print("\nFP =",FP)
        print("\nFN =",FN)




        file_name = os.path.split(f)[1]
        print("\nFilename=",file_name)
        out_file_name = os.path.join(FLAGS.out_dir, 'out_'+file_name)
        cv2.imwrite(out_file_name, im)
        print ('\nImage detection output saved to {}'.format(out_file_name))
        return count,TP,FP,FN


def main(argv=None):
  if not tf.gfile.Exists(FLAGS.out_dir):
    tf.gfile.MakeDirs(FLAGS.out_dir)
  if FLAGS.mode == 'image':
   image_demo()
```

```
#This is for the whole dataset
else:
 # Directory to be scanned
 dataset_images=[]
 obj = os.scandir(FLAGS.image_dataset)
 for entry in obj :
  dataset_images.append(entry.name)
  dataset_images.sort()
 print("dataset_images=",dataset_images)
 obj.close()


 # Directory to be scanned
 dataset_labels=[]
 obj = os.scandir(FLAGS.label_dataset)
 for entry in obj :
  dataset_labels.append(entry.name)
  dataset_labels.sort()
 print("dataset_labels=",dataset_labels)
 obj.close()

 i=0
 while i <=199:

  FLAGS.label_path= dataset_labels[i]
  print("FLAGS.label_path=",FLAGS.label_path)

  FLAGS.input_path= dataset_images[i]
```

```
 print("FLAGS.input_path=",FLAGS.input_path)

 print("FLAGS.image_dataset",FLAGS.image_dataset)
 print("FLAGS.label_dataset",FLAGS.label_dataset)


 FLAGS.input_path = os.path.join(FLAGS.image_dataset+ '/' +FLAGS.input_path)
 print("FLAGS.input_path=",FLAGS.input_path)

 FLAGS.label_path = os.path.join(FLAGS.label_dataset+ '/' +FLAGS.label_path)
 print("FLAGS.label_path=",FLAGS.label_path)
 image_demo()
 print("\ncount=",count)
 i=i+1
iou_dataset = count/i
print("\niou_dataset=",round(iou_dataset,3))
print("\nTP =",TP)
print("\nFP =",FP)
print("\nFN =",FN)
Precision = []
Recall = []
for i in range(len(thr)):
 Precision.append(round(TP[i]/(TP[i]+FP[i]),3))
 Recall.append(round(TP[i]/(TP[i]+FN[i]),3))
print("\nPrecision =",Precision)
print("\nRecall =",Recall)
plt.plot(Recall, Precision)
plt.ylabel('Precision')
plt.xlabel('Recall')
```

```
    plt.show()
    auc=metrics.auc(Recall, Precision)
    print("\nAUC=",round(auc,3))
    print("\nFor model:",FLAGS.demo_net)

if __name__ == '__main__':
    tf.app.run()
```