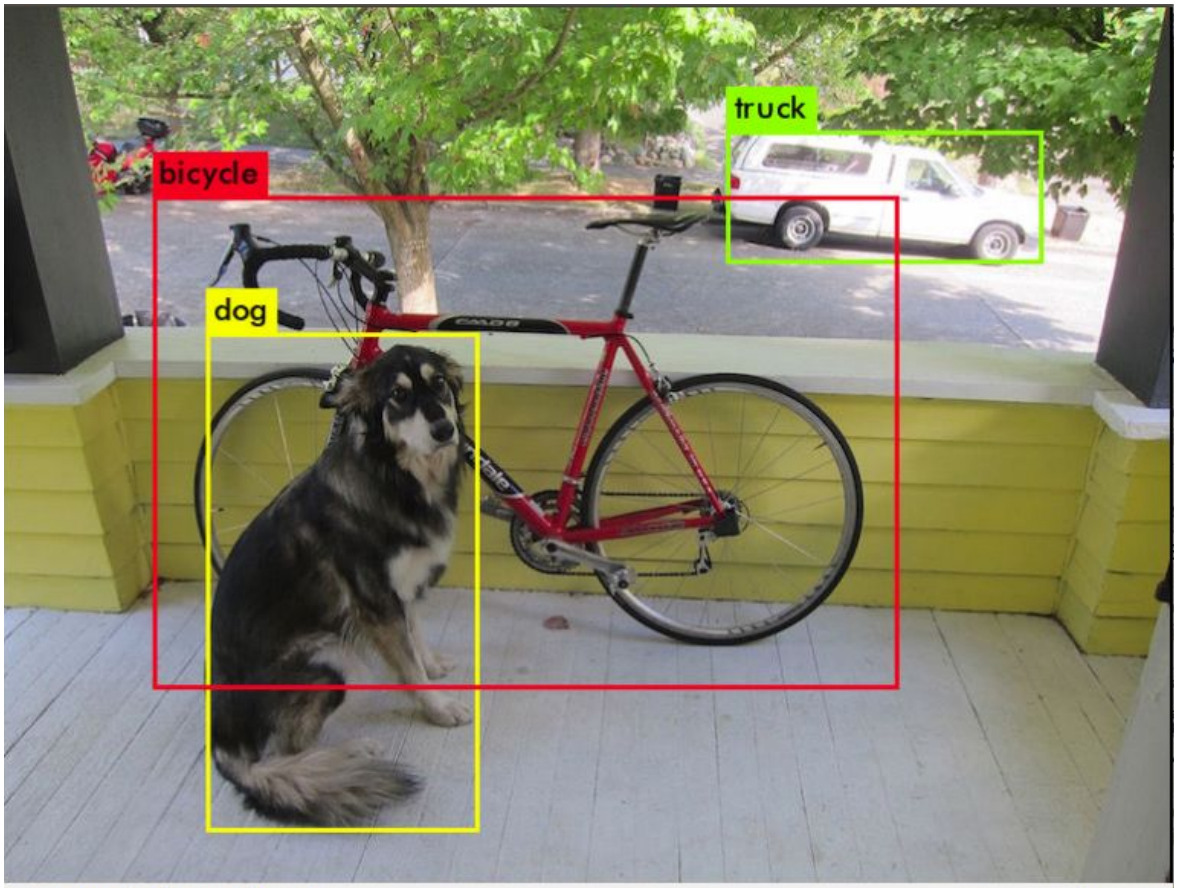
Antonin Bousquier
Ibrahim Elouard

CPE Lyon - 5 ETI - Robotique

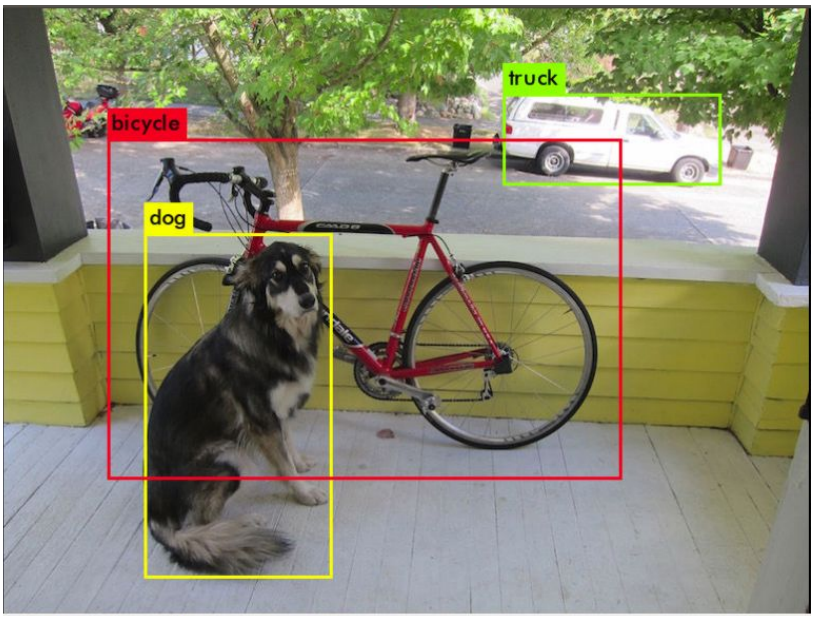
27/11/2018

TP VISION : Classification d'images par Deeplearning. Mise en oeuvre de Yolo/Darknet

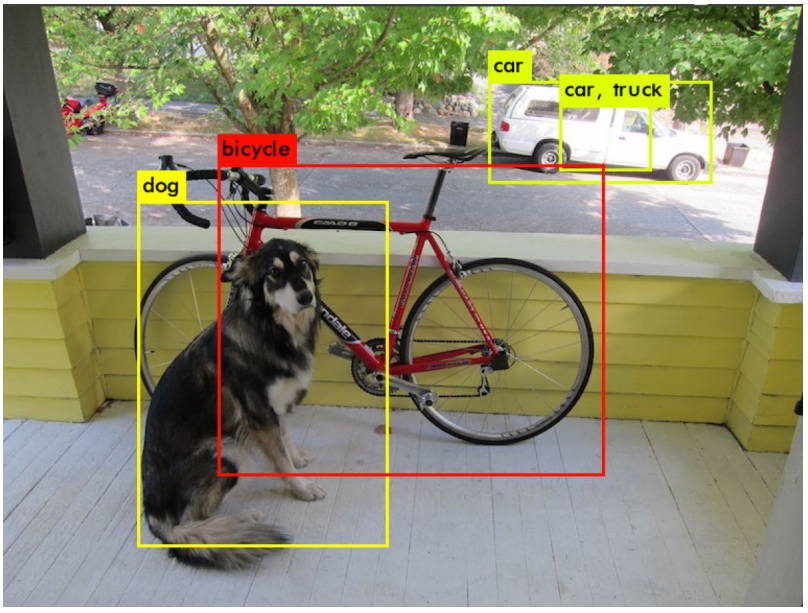


Prise en main local de darknet:

On suit donc le tutoriel pour apprendre à utiliser YOLO, en testant les 2 modèles préentrainés : yolov3 et yolov3-tiny. La version tiny est moins couteuse en ressources et moins longue mais elle est moins précise et obtient de moins bons résultats.



Analyse de l'image par le modèle préentrainé yolov3

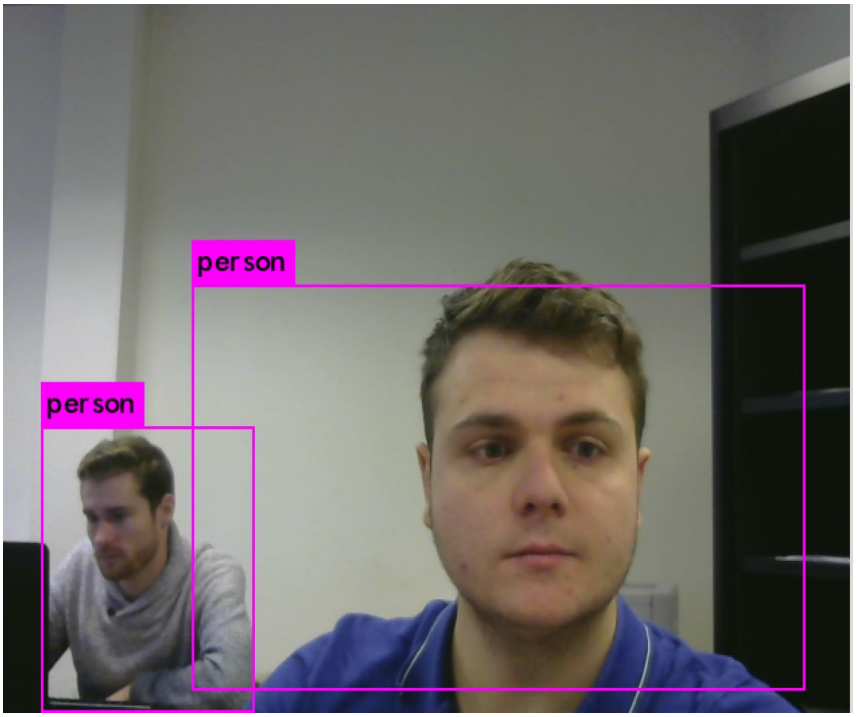


Analyse de l'image par le modèle préentrainé yolov3-tiny

On voit donc que la reconnaissance du vélo et du chien qui sont au premier plan fonctionne de la même manière, en revanche pour le 4x4 un peu caché en arrière plan la version tiny a

du mal à l'identifier (notamment à cause de la forte luminosité). On peut modifier le thresh hold selon la précision que l'on veut et la taille des objets sur l'image :
./darknet detect cfg/yolov3.cfg yolov3.weights data/dog.jpg -thresh 0

Pour l'analyse en temps réel on doit utiliser le modèle tiny, car nos ordinateurs ne sont pas assez puissants. On obtient bien une identification en temps réel du retour webcam :



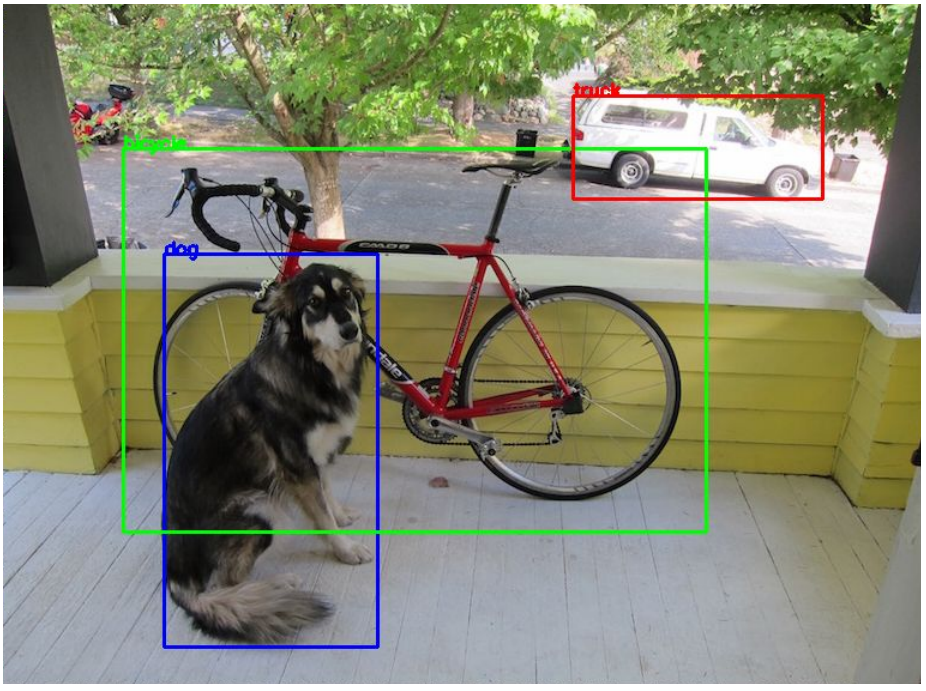
Analyse du retour caméra par le modèle préentrainé yolov3-tiny

Interface Python:

On va recréer le code permettant d'obtenir le même retour que l'image donnée dans le sujet : des cadres de couleurs différents pour chaque objet ainsi que la possibilité d'afficher qu'un seul type d'identification. On doit d'abord changer tout les chemins d'accès sur le code donnée sur cpe-campus, puis on y ajoute le code suivant :

```
        r = detect(net, meta,
"/fs03/share/users/antonin.bousquier/home/Vision/TP3/YOLO/darknet/data/chiens.jpg")
        img=
cv2.imread("/fs03/share/users/antonin.bousquier/home/Vision/TP3/YOLO/darknet/data/chi
ens.jpg",1)
        font = cv2.FONT_HERSHEY_SIMPLEX
        j=0
        for i in r:
        rgb=(0,0,0)
        if j==0:
        rgb=(255,0,0)
        elif j==1:
        rgb=(0,255,0)
```

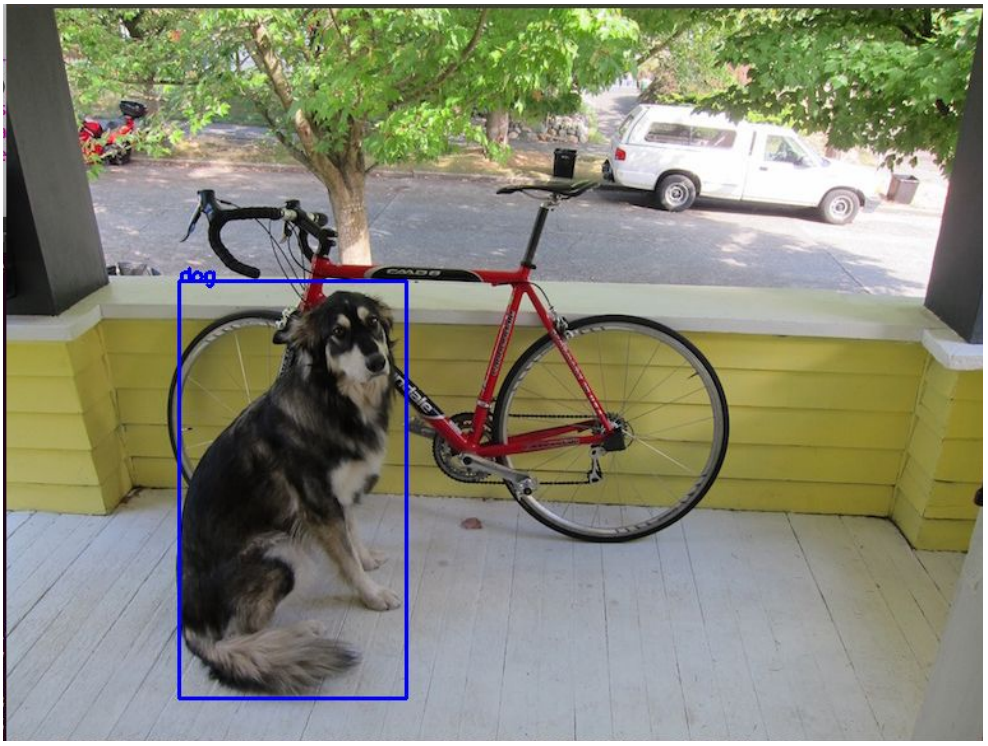
```
  elif j==2:
  rgb=(0,0,255)
  elif j==3:
  rgb=(255,255,255)
  elif j==4:
  rgb=(0,0,0)
  elif j==5:
  rgb=(255,255,0)
  else:
  rgb=(255,0,255)
name=i[0]
#if name == "dog":
x=int(i[2][0])
y=int(i[2][1])
w=int(i[2][2])
h=int(i[2][3])
cv2.rectangle(img,(x+w/2,y+h/2),(x-w/2,y-h/2),rgb,2)
cv2.putText(img,name,(x-w/2,y-h/2), font, 0.5,rgb,2,0)
 j=j+1
  cv2.imshow('test', img)
  print r
  while True:
  if cv2.waitKey(1) & 0xFF == 27:
            break
```



Affichage du résultat du code darknet.py selon le protocole du sujet

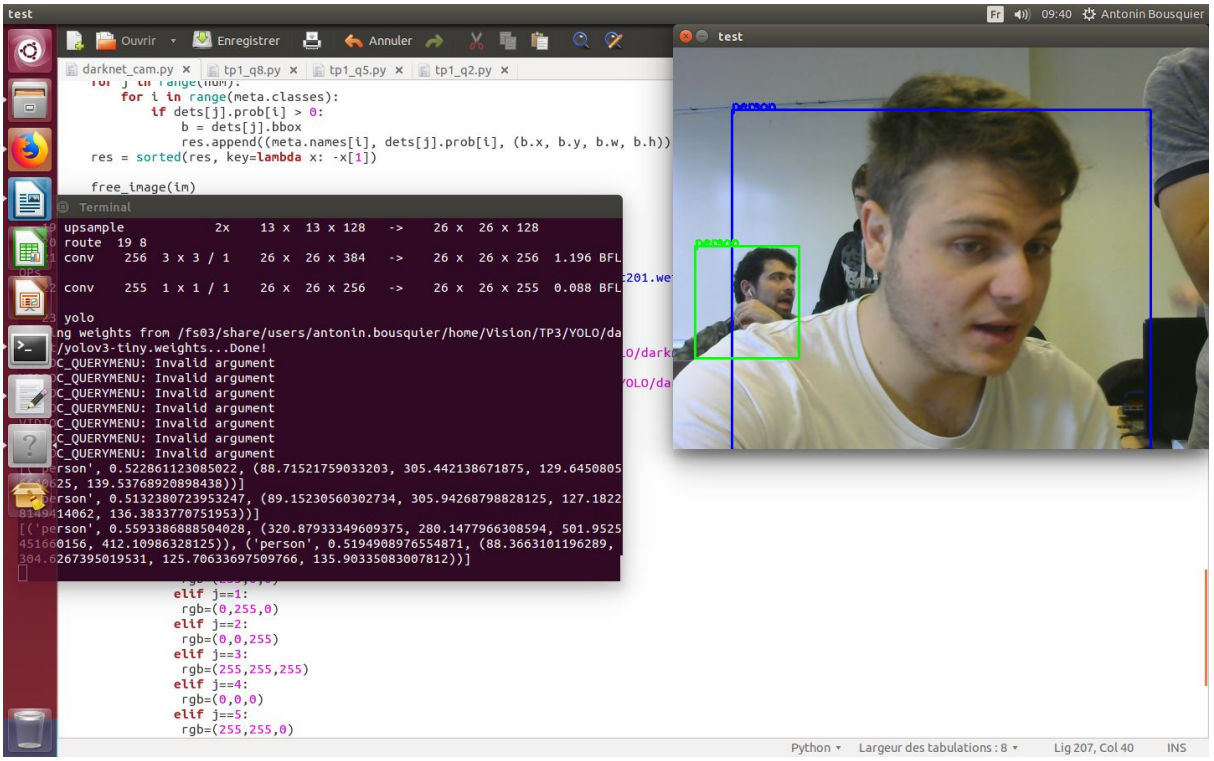
On peut également afficher qu'une seule classe (en décommentant la ligne commentée sur le code python : *if name == "dog":*), on peut également créer une liste regroupant les classes

que l'on souhaite identifier (les animaux par exemple) en ajoutant tous les animaux que l'on souhaite puis vérifier si name est présent dans liste. Exemple ici pour la classe "dog" :



Affichage du résultat du code darknet.py selon le protocole du sujet (avec une seule classe)

On va maintenant adapter le code pour gérer une flux vidéo (avec un nombre de FPS assez faible), pour cela il faut modifier notre code, tout d'abord faire la liaison avec la caméra grace à la commande *cap = cv2.VideoCapture(1)* on va ensuite grâce à une boucle while éxécuter en boucle l'analyse de l'image envoyer par la caméra, en revanche pour que l'analyse soit possible il nous faut modifier son format : *img=array_to_image(frame)* on reprend ensuite simplement le code pour l'image seule :

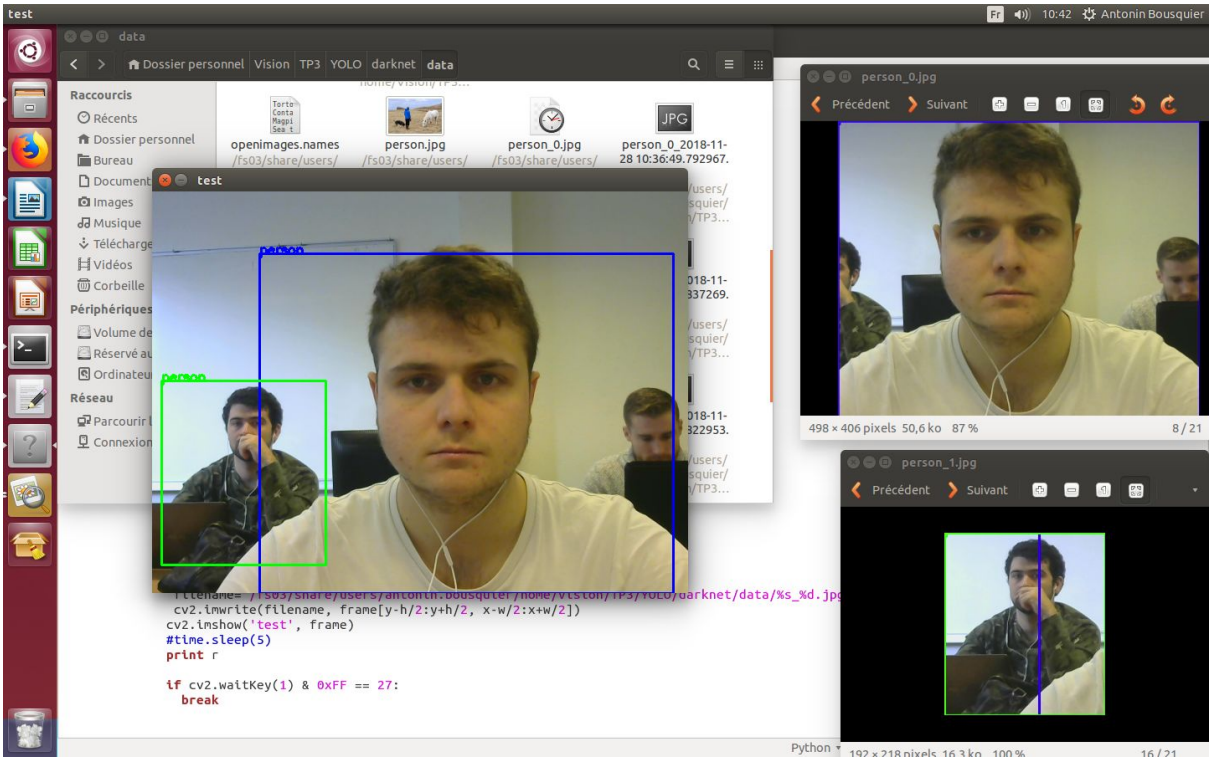


Affichage du résultat du code darknet_cam.py avec le retour caméra

On va maintenant sauvegarder les différentes parties à chaque lecture du flux vidéo, dans un fichier jpg grâce au code suivant :

```
filename="/fs03/share/users/antonin.bousquier/home/Vision/TP3/YOLO/darknet/data/%s_%d.jpg" % (name, j-1)
    cv2.imwrite(filename, frame[y-h/2:y+h/2, x-w/2:x+w/2])
```

On obtient ainsi les fichiers sous le bon format :

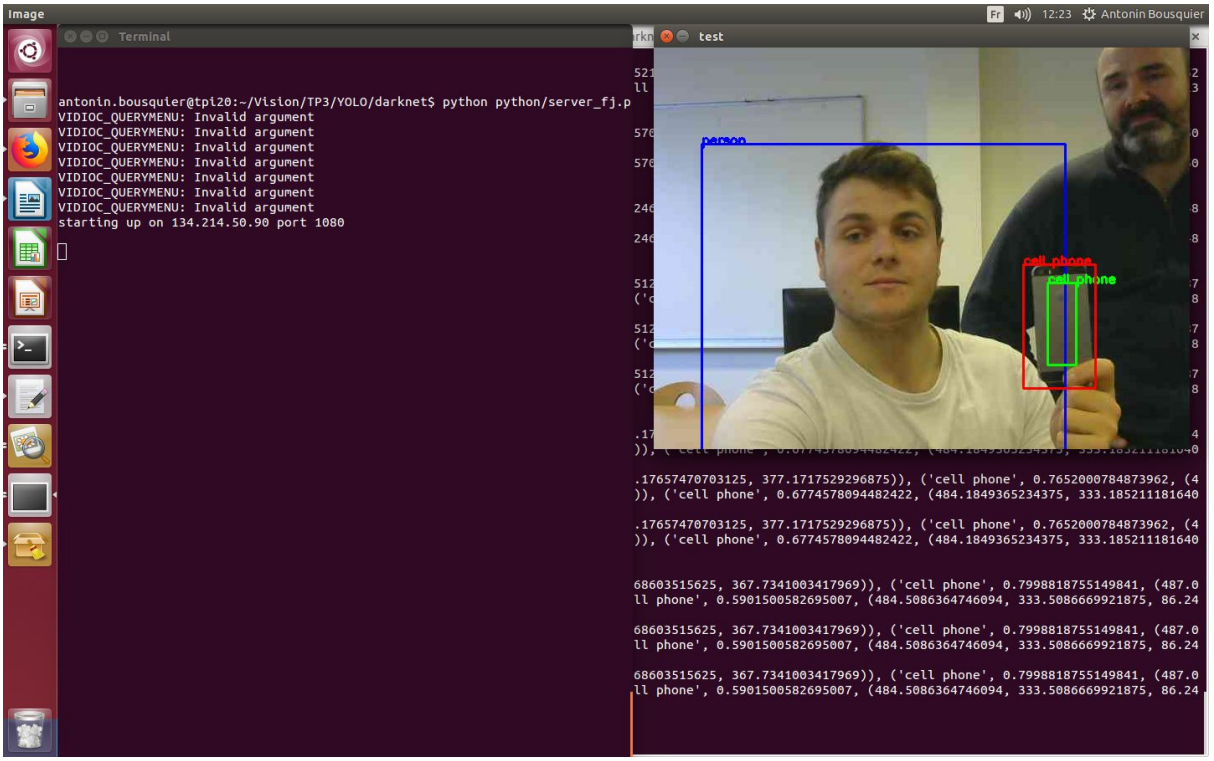


Enregistrement des résultats du code darknet_cam.py avec le retour caméra

On va maintenant tenter de lancer notre script sur le serveur possédant un GPU, ce qui permettra une vitesse de calcul et donc une analyse de l'image plus rapide (et donc une augmentation du nombre de FPS).

Pour cela nous devons envoyer le flux de notre webcam en UDP à la machine de calcul grâce au script server_fj.py qu'on adapt à notre cas (/dev/video1, carte réseau cpe).
Coté machine de calcul, nous récupérons notre code darknet grâce à la commande scp.

On modifie ensuite le script client.py pour qu'au lieu de nous afficher le retour caméra, il fasse en même temps l'analyse en temps réel et affiche le même retour que le code de la partie précédente (client_server.py dans le fichier python):



Analyse de notre retour caméra envoyé à la machine de calcul et analyse via le code client_serv.py qui affiche ici le résultat