Running a YOLOv4 Object Detector with Darknet in the Cloud! (GPU ENABLED)

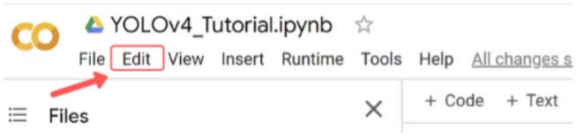
This tutorial will help you build YOLOv4 easily in the cloud with GPU enabled so that you can run object detections in milliseconds!

Step 1: Enabling GPU within your notebook

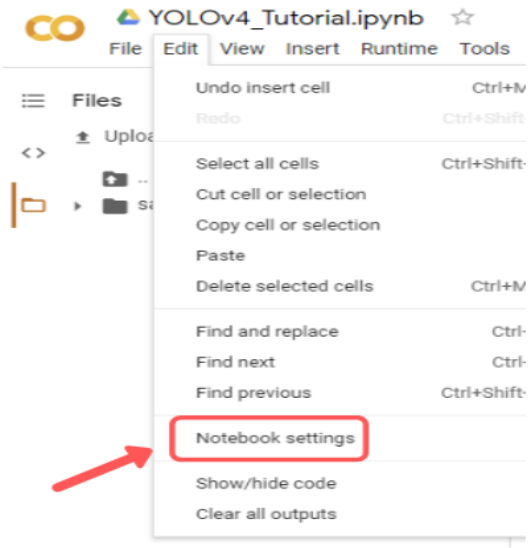
You will want to enable GPU acceleration within your Colab notebook so that your YOLOv4 system will be able to process detections over 100 times faster than CPU.

Steps:

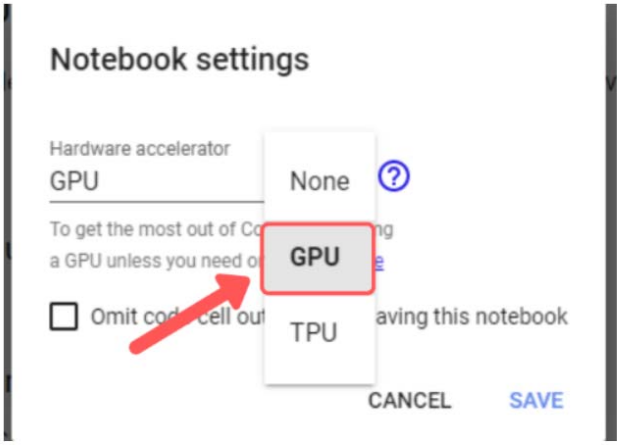
i) Click **Edit** at top left of your notebook



ii) Click **Notebook Settings** within dropdown



iii) Under 'Hardware Accelerator' select **GPU** and then hit **Save**



Your notebook should now have GPU enabled!

Step 2: Cloning and Building Darknet

The following cells will clone darknet from AlexeyAB's famous repository, adjust the Makefile to enable OPENCV and GPU for darknet and then build darknet.

Do not worry about any warnings when you run the '!make' cell!

---

[1]
```
!git clone https://github.com/AlexeyAB/darknet
 # github에서 darknet open source를 가져와서 colab의 클루우드 컴퓨터에 다운받는다.
```

```
Cloning into 'darknet'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 14751 (delta 0), reused 3 (delta 0), pack-reused 14748
Receiving objects: 100% (14751/14751), 13.28 MiB | 16.83 MiB/s, done.
Resolving deltas: 100% (10029/10029), done.
```

[2]
```
%cd darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile          # Makefile 내의 OPENCV=0이라는 문장을 찾아서 OPENCV=1로 변환
!sed -i 's/GPU=0/GPU=1/' Makefile                # Makefile 내의 GPU=0이라는 문장을 찾아서 GPU=1로 변환
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile            # Makefile 내의 CUDNN=0이라는 문장을 찾아서 CUDNN=1로 변환
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile  # Makefile 내의 CUDNN_HALF=0이라는 문장을 찾아서 CUDNN_HALF=1로 변환
```

```
/content/darknet
```

[3]
```
!/usr/local/cuda/bin/nvcc --version          # CUDA의 버젼을 확인 한다.
```

```
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2020 NVIDIA Corporation
Built on Wed_Jul_22_19:09:09_PDT_2020
Cuda compilation tools, release 11.0, V11.0.221
Build cuda_11.0_bu.TC445_37.28845127_0
```

[4]
```
!make                                        # make를 통해 yolov4를 설치한다.
```

```
./src/blas_kernels.cu(1086): warning: variable "out_index" was declared but never referenced

./src/blas_kernels.cu(1130): warning: variable "step" was set but never used

./src/blas_kernels.cu(1736): warning: variable "stage_id" was declared but never referenced

./src/blas_kernels.cu(1086): warning: variable "out_index" was declared but never referenced

./src/blas_kernels.cu(1130): warning: variable "step" was set but never used

./src/blas_kernels.cu(1736): warning: variable "stage_id" was declared but never referenced

./src/blas_kernels.cu(1086): warning: variable "out_index" was declared but never referenced

./src/blas_kernels.cu(1130): warning: variable "step" was set but never used

./src/blas_kernels.cu(1736): warning: variable "stage_id" was declared but never referenced

./src/blas_kernels.cu(1086): warning: variable "out_index" was declared but never referenced

./src/blas_kernels.cu(1130): warning: variable "step" was set but never used
```

Step 3: Download pre-trained YOLOv4 weights

YOLOv4 has been trained already on the coco dataset which has 80 classes that it can predict. We will grab these pretrained weights so that we can run YOLOv4 on these pretrained classes and get detections.

```
[5] !wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.weights
    # github에서 미리 학습된 모델 가중치를 제공 받는다.
```

```
--2021-04-07 07:18:28--  https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.weights
Resolving github.com (github.com)... 52.192.72.89
Connecting to github.com (github.com)|52.192.72.89|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-releases.githubusercontent.com/75388965/ba4b6380-889c-11ea-9751-f994f5961796?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credе
--2021-04-07 07:18:28--  https://github-releases.githubusercontent.com/75388965/ba4b6380-889c-11ea-9751-f994f5961796?X-Amz-Algorithm=AWS4-HMAC-SHA
Resolving github-releases.githubusercontent.com (github-releases.githubusercontent.com)... 185.199.111.154, 185.199.108.154, 185.199.109.154, ...
Connecting to github-releases.githubusercontent.com (github-releases.githubusercontent.com)|185.199.111.154|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 257717640 (246M) [application/octet-stream]
Saving to: 'yolov4.weights'

yolov4.weights      100%[===================>] 245.78M  22.8MB/s    in 11s

2021-04-07 07:18:40 (22.0 MB/s) - 'yolov4.weights' saved [257717640/257717640]
```

Step 4: Define Helper Functions

These three functions are helper functions that will allow you to show the image in your Colab Notebook after running your detections, as well as upload and download images to and from your Cloud VM.

```
[ ] def imShow(path):
      import cv2                                  # cv2 open source를 가져온다.
      import matplotlib.pyplot as plt             # 이미지를 그리기위해 라이브러리를 가져온다.
      %matplotlib inline

      image = cv2.imread(path)                    # 경로를 이용하여 이미지를 불러온다.
      height, width = image.shape[:2]             # 이미지의 높이와 너비를 나타낸다.
      resized_image = cv2.resize(image,(3*width, 3*height), interpolation = cv2.INTER_CUBIC) # 경로에 저장된 사진크기를 3배 크게 해준다.

      fig = plt.gcf()                             # Figure 객체를 구한다.
      fig.set_size_inches(18, 10)                 # 객체 크기는 18,10으로 나타낸다.
      plt.axis("off")                               # 축을 제거 시킨다.
      plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))   # resizd_image와 BGR컬러를 사용하여 이미지를 나타내준다.
      plt.show()

    def upload():
      from google.colab import files              # 로컬 컴퓨터에서 코랩 서버 컴퓨터로 파일을 전송
      uploaded = files.upload()
      for name, data in uploaded.items():
        with open(name, 'wb') as f:               # 객체를 저장해 준다.
          f.write(data)
          print ('saved file', name)

    def download(path):
      from google.colab import files              # 코랩 서버 컴퓨터에서 로컬 컴퓨터로 전송
      files.download(path)
```

Step 5: Run Your Detections with Darknet and YOLOv4!

Darknet is now built and ready to run detections using YOLOv4 in the cloud! You can find out which sorts of classes the pre-trained YOLOv4 weights can detect by clicking here. COCO CLASSES

The object detector can be run using the following command

```
!./darknet detector test <path to .data file> <path to config> <path to weights> <path to image>
```

Darknet comes with a few images already installed in the darknet/data/ folder.

Note: After running detections OpenCV can't open the image instantly in the cloud so we must run:

```
imShow('predictions.jpg')
```

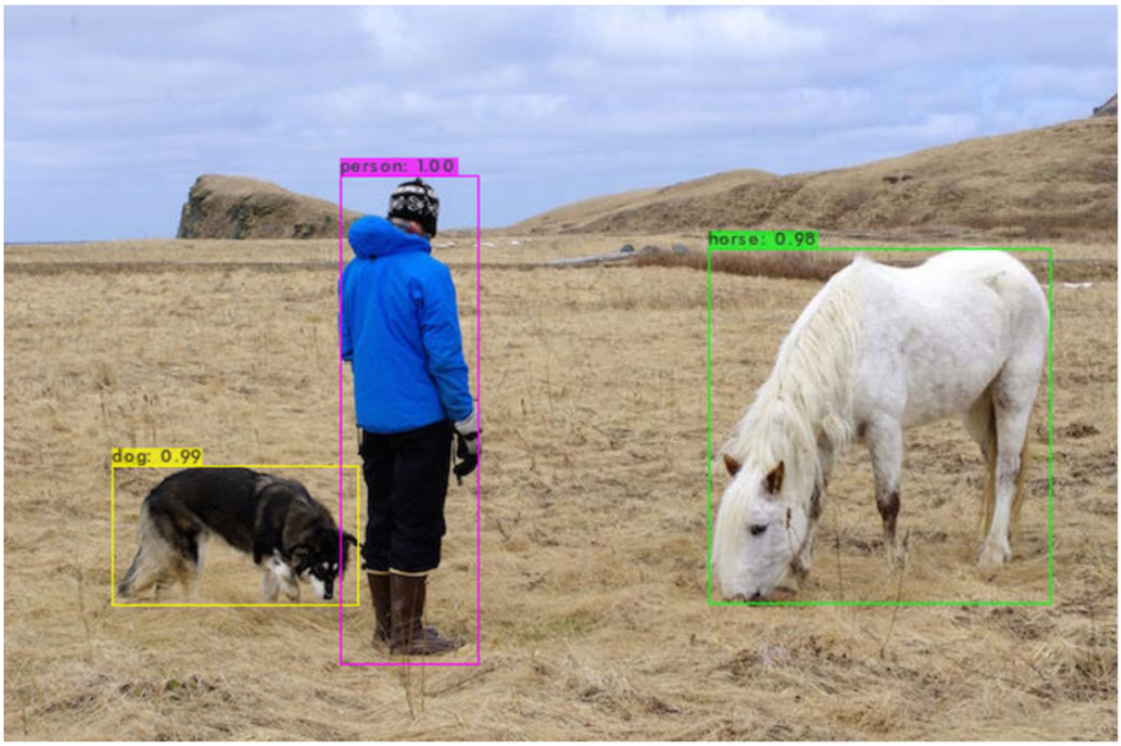
This will output the image with the detections shown. The most recent detections are always saved to 'predictions.jpg'

Try out the examples below for yourself!

```
!./darknet detector test cfg/coco.data cfg/yolov4.cfg yolov4.weights data/person.jpg
# open source darknet과 가중치 yolov4를 실행시켜 테스트 이미지를 통해 객체 탐지 확인
```

```
 CUDA-version: 11000 (11020), cuDNN: 7.6.5, CUDNN_HALF=1, GPU count: 1
 CUDNN_HALF=1
 OpenCV version: 3.2.0
 0 : compute_capability = 370, cudnn_half = 0, GPU: Tesla K80
net.optimized_memory = 0
mini_batch = 1, batch = 8, time_steps = 1, train = 0
   layer   filters  size/strd(dil)      input                output
   0 Create CUDA-stream - 0
 Create cudnn-handle 0
conv     32       3 x 3/ 1    608 x 608 x   3 ->  608 x 608 x  32 0.639 BF
   1 conv     64       3 x 3/ 2    608 x 608 x  32 ->  304 x 304 x  64 3.407 BF
   2 conv     64       1 x 1/ 1    304 x 304 x  64 ->  304 x 304 x  64 0.757 BF
   3 route  1                                     ->  304 x 304 x  64
   4 conv     64       1 x 1/ 1    304 x 304 x  64 ->  304 x 304 x  64 0.757 BF
   5 conv     32       1 x 1/ 1    304 x 304 x  64 ->  304 x 304 x  32 0.379 BF
   6 conv     64       3 x 3/ 1    304 x 304 x  32 ->  304 x 304 x  64 3.407 BF
   7 Shortcut Layer: 4,  wt = 0, wn = 0, outputs: 304 x 304 x  64 0.006 BF
   8 conv     64       1 x 1/ 1    304 x 304 x  64 ->  304 x 304 x  64 0.757 BF
   9 route  8 2                                   ->  304 x 304 x 128
  10 conv     64       1 x 1/ 1    304 x 304 x 128 ->  304 x 304 x  64 1.514 BF
  11 conv    128       3 x 3/ 2    304 x 304 x  64 ->  152 x 152 x 128 3.407 BF
  12 conv     64       1 x 1/ 1    152 x 152 x 128 ->  152 x 152 x  64 0.379 BF
  13 route  11                                    ->  152 x 152 x 128
  14 conv     64       1 x 1/ 1    152 x 152 x 128 ->  152 x 152 x  64 0.379 BF
  15 conv     64       1 x 1/ 1    152 x 152 x  64 ->  152 x 152 x  64 0.189 BF
  16 conv     64       3 x 3/ 1    152 x 152 x  64 ->  152 x 152 x  64 1.703 BF
  17 Shortcut Layer: 14,  wt = 0, wn = 0, outputs: 152 x 152 x  64 0.001 BF
  18 conv     64       1 x 1/ 1    152 x 152 x  64 ->  152 x 152 x  64 0.189 BF
  19 conv     64       3 x 3/ 1    152 x 152 x  64 ->  152 x 152 x  64 1.703 BF
  20 Shortcut Layer: 17,  wt = 0, wn = 0, outputs: 152 x 152 x  64 0.001 BF
  21 conv     64       1 x 1/ 1    152 x 152 x  64 ->  152 x 152 x  64 0.189 BF
  22 route  21 12                                ->  152 x 152 x 128
  23 conv    128       1 x 1/ 1    152 x 152 x 128 ->  152 x 152 x 128 0.757 BF
  24 conv    256       3 x 3/ 2    152 x 152 x 128 ->   76 x  76 x 256 3.407 BF
```

```
imShow('predictions.jpg')
# 모델에 예시를 들어 확인해본다.
```



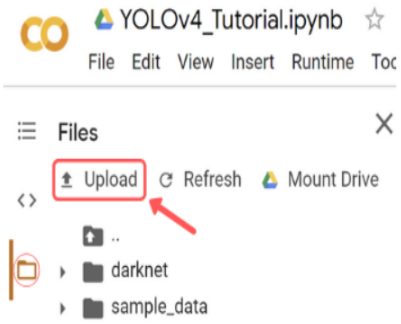
Step 6: Uploading Local or Google Drive Files to Use

You may want to run detections on more than just the images within the darknet/data/ folder. This step will show you how to upload local or Google Drive files to the cloud VM and run detections on them!

Method 1: Local Files

To upload local files just use our helper function by running 'upload()' as seen below. Click **Choose Files** and select the image from your local machine that you want to upload to the cloud VM.

If this function doesn't work for you then click the **Upload** button in the File Explorer on the left side of your notebook.



The image should save to the root directory of the cloud VM so that you can access it from the darknet command by running.

```
!./darknet detector test cfg/coco.data cfg/yolov4.cfg yolov4.weights ../<your image name>
```

```
# upload 함수를 사용하여 파일을 코랩 서버로 가져온다.
%cd ..
upload()
%cd darknet
```

```
/content
파일 선택 차지기과제1.jpg
- 차지기과제1.jpg(image/jpeg) - 558827 bytes, last modified: 2021. 4. 7. - 100% done
Saving 차지기과제1.jpg to 차지기과제1.jpg
saved file 차지기과제1.jpg
/content/darknet
```

```
# open source darknet과 가중치 yolov4를 실행시켜 파일에서 가져온 이미지를 통해 객체 탐지 확인
!./darknet detector test cfg/coco.data cfg/yolov4.cfg yolov4.weights ../차지기과제1.jpg
imShow('predictions.jpg')
```

```
Loading weights from yolov4.weights...
 seen 64, trained: 32032 K-images (500 Kilo-batches_64)
Done! Loaded 162 layers from weights-file
 Detection layer: 139 - type = 28
 Detection layer: 150 - type = 28
 Detection layer: 161 - type = 28
../차지기과제1.jpg: Predicted in 162.457000 milli-seconds.
car: 98%
person: 93%
person: 34%
person: 48%
person: 48%
car: 81%
truck: 25%
traffic light: 33%
bus: 58%
truck: 50%
car: 85%
traffic light: 96%
traffic light: 33%
traffic light: 32%
person: 50%
Unable to init server: Could not connect: Connection refused

(predictions:1152): Gtk-WARNING **: 07:25:49.777: cannot open display:
```

```
# upload 함수를 사용하여 파일을 코랩 서버로 가져온다.
%cd ..
upload()
%cd darknet
```

```
/content
파일 선택 차지기과제2.jpg
```

- **차지기과제2.jpg**(image/jpeg) - 602619 bytes, last modified: 2021. 4. 7. - 100% done

```
Saving 차지기과제2.jpg to 차지기과제2.jpg
saved file 차지기과제2.jpg
/content/darknet
```

```
# open source darknet과 가중치 yolov4를 실행시켜 파일에서 가져온 이미지를 통해 객체 탐지 확인
!./darknet detector test cfg/coco.data cfg/yolov4.cfg yolov4.weights ../차지기과제2.jpg
imShow('predictions.jpg')
```

```
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.05
nms_kind: greedynms (1), beta = 0.600000
Total BFLOPS 128.459
avg_outputs = 1068395
 Allocate additional workspace_size = 6.65 MB
Loading weights from yolov4.weights...
 seen 64, trained: 32032 K-images (500 Kilo-batches_64)
Done! Loaded 162 layers from weights-file
 Detection layer: 139 - type = 28
 Detection layer: 150 - type = 28
 Detection layer: 161 - type = 28
../차지기과제2.jpg: Predicted in 165.670000 milli-seconds.
person: 54%
person: 91%
bench: 47%
person: 34%
car: 40%
```

Method 2: Google Drive

Images can also be uploaded from your Google Drive and easily have YOLOv4 detections run on them.

You will want to run the below cell to mount your google drive into the cloud VM so that you can access its contents. It is that easy!

NOTE: We will be creating a symbolic link between '/content/gdrive/My\ Drive/' and '/mydrive.

This means we are just creating a shortcut '/mydrive' to map to the contents within the folder '/content/gdrive/My\ Drive/'.

The reason for this is that sometime having the space in 'My Drive' folder path can cause issues when running certain commands. This symbolic link will stop this from happening!

Now you can run YOLOv4 with images from Google Drive using the darknet command:

```
!./darknet detector test cfg/coco.data cfg/yolov4.cfg yolov4.weights /mydrive/<path to image>
```

I recommend saving images within a folder called 'images' at the root level of your Google Drive.

```
[14] %cd ..
     from google.colab import drive  # 구글 드라이브와 연동시켜준다.
     drive.mount('/content/gdrive')
```

```
/content
Mounted at /content/gdrive
```

```
[15] !ln -s /content/gdrive/My# Drive/ /mydrive # 드라이브 접근위치를 Mydrive로 접근 가능하게 해준다.
     !ls /mydrive                               # ls를 통해 드라이브 내 파일을 확인한다.
```

```
차지기  'Colab Notebooks'
```

```
%cd darknet
```

```
[Errno 20] Not a directory: 'darknet'
/content/darknet
```

```
# 구글 드라이브에 이미지 폴더를 만들고 넣어둔 이미지를 통해 객체를 탐지한다.
!./darknet detector test cfg/coco.data cfg/yolov4.cfg yolov4.weights /mydrive/image/highway.png
imShow('predictions.jpg')
```

```
 157 conv   1024       3 x 3/ 1     19 x  19 x 512 ->   19 x  19 x1024 3.407 BF
 158 conv    512       1 x 1/ 1     19 x  19 x1024 ->   19 x  19 x 512 0.379 BF
 159 conv   1024       3 x 3/ 1     19 x  19 x 512 ->   19 x  19 x1024 3.407 BF
 160 conv    255       1 x 1/ 1     19 x  19 x1024 ->   19 x  19 x 255 0.189 BF
 161 yolo
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norn: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.05
nms_kind: greedynms (1), beta = 0.600000
Total BFLOPS 128.459
avg_outputs = 1068395
 Allocate additional workspace_size = 6.65 MB
Loading weights from yolov4.weights...
 seen 64, trained: 32032 K-images (500 Kilo-batches_64)
Done! Loaded 162 layers from weights-file
 Detection layer: 139 - type = 28
 Detection layer: 150 - type = 28
 Detection layer: 161 - type = 28
/mydrive/image/highway.png: Predicted in 165.728000 milli-seconds.
car: 54%
car: 49%
car: 31%
car: 84%
car: 38%
```

Download Files to Local Machine or Google Drive from Cloud VM

You can also easily download images from your cloud VM to save to your local machine or Google Drive.

Method 1: Local Machine

You can do it easily by using our helper function 'download()' or by right clicking the image in the File Explorer on the left side of your notebook and hitting **Download**. Files will be saved to your *Downloads* folder.

This is useful if you want to download the '**predictions.jpg**' images that the object detector outputs.

Method 2: Google Drive

A simple copy command can copy over any file to your Google Drive as it is already mounted. (you must run the mount command above if you have not already)

```
!cp <file to download> <destination to save file>
```

See example of each below!

```
[23] # 코랩에서 작업한 파일을 로컬 컴퓨터로 다운
     download('predictions.jpg')  # download 함수를 사용하여 파일의 경로와 파일명을 적어준다.
```

```
[25] !cp predictions.jpg /mydrive/image/detection1.jpg  # 구글 드라이브에 저장
```

✓ 0초 오후 4:44에 완료됨

predictions.jpg

Step 7: Running YOLOv4 on Video in the Cloud!

You can also run YOLOv4 object detections on video in your Google Colab VM. Follow the cells below to see how to run videos from your local machine or from your Google Drive!

Local Machine Video

Here is how to upload video from local machine, run detector and then download video showing detections.

```
upload()  # video upload
```

파일 선택 test.mp4

- **test.mp4**(video/mp4) - 2463213 bytes, last modified: 2021. 4. 7. - 100% done

```
Saving test.mp4 to test.mp4
saved file test.mp4
```

```
[28] !./darknet detector demo cfg/coco.data cfg/yolov4.cfg yolov4.weights -dont_show test.mp4 -i 0 -out_filename results.avi
     # open source, coco.data, 가중치를 가져오고 test.mp4를 upload한다.
```

```
bicycle: 33%
person: 100%
person: 97%
person: 97%
person: 97%
person: 97%
person: 97%
person: 97%
```

```
[29] download('results.avi') # 결과 영상을 다시 다운로드 해준다.
```

Google Drive Video

Here is how to run detector on video stored in Google Drive and save video straight to Google drive as well!

Note: You will have to change the paths to where your video is stored within your Google Drive and to where you want the resulting video stored.

I have a videos folder in the home directory of my Google Drive.

```
!./darknet detector demo cfg/coco.data cfg/yolov4.cfg yolov4.weights -dont_show /mydrive/videos/test.mp4 -i 0 -out_filename /mydrive/videos/results.avi
# 드라이브에 videos를 저장하고 사용
```

Step 8: Customize YOLOv4 with the different command line flags.

Darknet and YOLOv4 have a lot of command line flags you can add to your '!./darknet detector ...' to allow it to be customizeable and flexible.

I will show a few examples of these flags that you can take advantage of! Feel free to mix and match them together to customize your detections in any way you want.

Threshold Flag

There is a flag '-thresh' you can use to add a threshold for confidences on the detections. Only detections with a confidence level above the threshold you set will be returned.

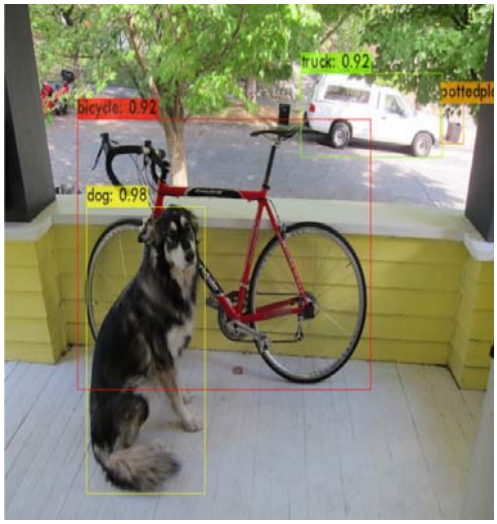
In the example below we run darknet with YOLOv4 without a threshold on the test image dog.jpg. The image returns four detections, the lowest confidence being on the pottedplant with 33%.

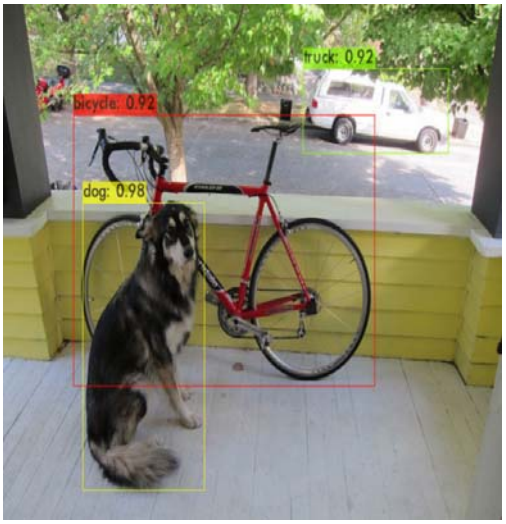
If we add the '-thresh 0.5' flag this will only output three detections as now pottedplant falls below the threshold and is ignored.

Check it out below!

```
[ ] !./darknet detector test cfg/coco.data cfg/yolov4.cfg yolov4.weights data/dog.jpg
    imShow('predictions.jpg')
```

```
[ ] # 정확도를 올리기 위해 threshold를 바꿔주어 진행한다.
    !./darknet detector test cfg/coco.data cfg/yolov4.cfg yolov4.weights data/dog.jpg -thresh 0.5
    imShow('predictions.jpg')
```





Output Bounding Box Coordinates

You can output bounding box coordinates for each detection with the flag '-ext_output'. This external outputs flag will give you a few extra details about each detection within an image.

Check it out below!

```
[33] # -ext_ouput을 통해 탐지 결과에서 Bounding BOX 좌표를 얻어 온다.
     !./darknet detector test cfg/coco.data cfg/yolov4.cfg yolov4.weights data/person.jpg -ext_output
     imShow('predictions.jpg')
```

```
dog: 99%        (left_x:   62   top_y:  265   width:  142   height:   80)
person: 100%    (left_x:  194   top_y:   98   width:   78   height:  281)
horse: 98%      (left_x:  406   top_y:  140   width:  196   height:  204)
Unable to init server: Could not connect: Connection refused

(predictions:1884): Gtk-WARNING **: 08:15:47.653: cannot open display:
```