Patryk Laskowski

GitHub

YOLOv4 gym

This is fully automated training environment for YOLO v4.

TODO

- [x] Numerate the notebook
- [ ] Complete **7.2. Object detection** section
- [x] Add folder tree
- [ ] Add *add training examples* feature of darknet
- [ ] Add test (with print)
- [ ] Complete **Re-run training** section
- [ ] Complete **Overview** section (#How to use)
- [ ] Limitations

Overview

a) What is it?

Notebook is complete environment for Yolo v4 training using GPU. The goal of this project was to create solution that automate all the preparation process thereby simplifying genraterion of new customized Yolo v4 weights.

Whole process starting from dataset downloading throug environment structure preparation and Darknet customization is taken care of without any need of user interaction. Taking advance of free GPU, served by Google Colaboratory after few hours you may have ready to use customized object detection model.

There is a great open-images-dataset project full of annotated images that is used in this pipeline. Another crucial part is Darknet framework with actual Yolo v4 model. It's github source code is here.

b) How to use?

Try with basic setup

Basically there are only two required steps in order to achieve first results: (1) GPU runtime activation and (2) Google Drive mounting (runtime activation have to always be the first one). However in this case you will obtain working but shabby results. If you done those two, just run all the cells and wait for model's weights as well as performance chart to be saved on Your Google Drive. Simple as that!

Customize it
If you want to have tailor made weights resulting with better performance, there is also third (3) "Customize YOLO v4 objective" step. There are fours variables to adjust: `classes`, `size`, `n_train`, `n_validation`. These describe model objective with classes you want model to detect, size of model input (size = input image height = input image width) and maximum number of train and validation examples. Description on how certain settings affect the performance are directly in step's section.

Don't worry about runtime disconnection
Finally there is one more step worth to consider: (4) "Prevent idle disconnection". This is substep of "Train" section that helps to not worry about runtime interrutption when wheights are optimizing. This is optional but recommended.

All the dataflow is designed in such way that there

c) Table of contents

d) Example final contents of directories in a tree-like format

`/mydrive` is a symbolic link to mounted Google Drive root directory
[`/content/gdrive/MyDrive`](/content/gdrive/MyDrive)

```
/mydrive
└── yolov4
    └── Vehicle_Human_hand_Banana
        ├── backup
        │   ├── yolov4-custom_1000.weights
        │   ├── ...
        │   └── yolov4-custom_last.weights
        ├── chart.png
        ├── data
        │   ├── train
        │   │   ├── 00c166195da83904.jpg
        │   │   ├── 00c166195da83904.txt
        │   │   └── ...
        │   ├── train.txt
        │   ├── validation
        │   │   ├── 04833bdaa8c68594.jpg
        │   │   ├── 04833bdaa8c68594.txt
        │   │   └── ...
        │   ├── validation.txt
        │   ├── yolov4-custom.data
        │   └── yolov4-custom.names
        └── yolov4-custom.cfg
```

e) How it works?

GPU and Google Drive
In the first place this notebook takes advantage of free GPU in the cloud service by Google Colab. This is why this environment has been chosen. This approach is device independent making solution universal. Having in mid that runtime is temporary there is need to have some secure data storage thus Google Drive Mount is second of required steps.
Data preparation (automated)
When GPU runtime is on and drive is connected all the magic is happening - based on YOLO configuration, dataset is downloaded using OIDv4 ToolKit. Next step include bounding box annotation conversion to YOLO format. After that new directory is created on drive where prepared data is moved.
Darknet preparation (automated)
Following is Darknet framework preparation. Oryginal solution is downloaded from github than adjusted and built. Darknet is running to test it's build, then, if all went smooth, some necessery files are created such as `.cfg`, `.name`, `.data`, etc. Finally pre-trained weights (`.weights`) file is downloaded to speed up training process - this technique is called transfer learning.
Training (automated)
When all things are set it is time for weights optimization (training). Process takes many hours (depends on task complexity), during which performance visualization is generated (by Darknet framework). To not loose this `chart.png` (which is helpful to determine best weights) there is "parallel" thread running that saves the chart on google drive. Threfore you may take a look at model performence even during training.
Finally
Weights are the most important - they get determine model accuracy. `.weights` files are saved every 1000 epochs in Google Drive on path `/mydrive/yolov4/<project dir>/backup`. They are ready to use which is presented in sections "Evaluate" and "Re-run training".

Before You start

This notebook is prepared in such way, that only up to 4 steps require your action:

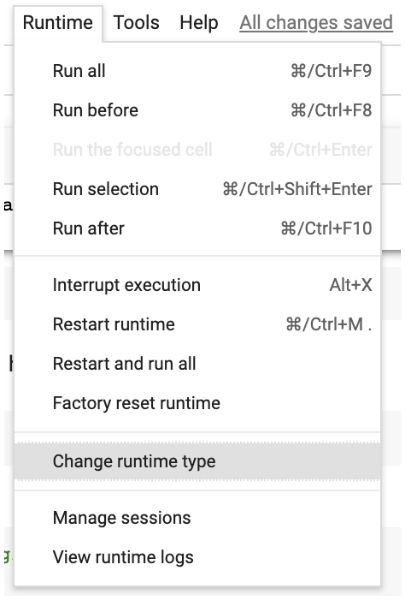
1. **Activate GPU runtime**
   - Darknet framework (with YOLO v4) is configurated to be automatically run with use of GPU. This speeds up trainig drastically (comaring to CPU).
2. **Customize YOLO v4 objective** (optionally - default values will work)
   - This is high level input and output customization. Choosing new classes for object detection is as easy as add new, or replace existing value to list `classes`.
3. **Mount Google Drive**
   - Required action since colab runtime does not last forever. This prevent eventual data loss or doing peparation from scratch again in case of disconnection.
4. **Prevent idle disconnection** (optionally)
   - Colaboratory runtime is limited. You have to fake some user activity to prevent automatic idle disconnection. If ommited, training time might be interrupted soon.

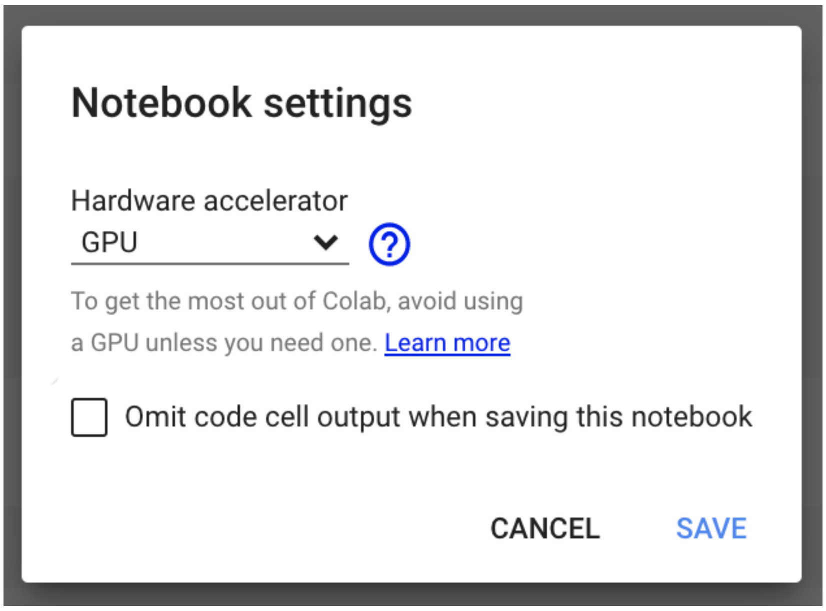
The rest configurations are meant to be done automatically.

1) Activate GPU runtime

This is very first step defining runtime type in which code will be executing.

Notebook setting to run automatically require GPU runtime.





Notebook settings

Hardware accelerator
GPU

To get the most out of Colab, avoid using a GPU unless you need one. Learn more

☐ Omit code cell output when saving this notebook

2) Mount Google Drive

This prevents data loss when runtime disconnect.

```
# root directory
%cd /content

import os

if not os.path.exists('/content/gdrive'):
  from google.colab import drive
  drive.mount('/content/gdrive')
```

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

```
# create symbolic link: now "/content/gdrive/MyDrive" equals "/mydrive"
!ln -s /content/gdrive/MyDrive/ /mydrive
```

3) Customize YOLO v4 objective

- **classes** : list of classes you want the model to detect. Maybe any of 600 object classes from Open Images Dataset.
- **size** : height ad width of expected input. Any multiple of 32 (default is 416)
- **n_train** : Max number of train examples each class. The more the better.
- **n_validation** : Max number of validation examples each class. The more the better.

[ ] ↳ *3 cells hidden*

4) Prepare dataset

OIDv4 ToolKit enables image download from Open Images Dataset.

4.1. Setup OIDv4 ToolKit environment

```
%cd /content
# Download git repository
!git clone https://github.com/patryklaskowski/OIDv4_ToolKit.git
%cd OIDv4_ToolKit
# Install requirements
!python3 -m pip install -r requirements.txt
```

```
    /content
    Cloning into 'OIDv4_ToolKit'...
    remote: Enumerating objects: 447, done.
    remote: Counting objects: 100% (447/447), done.
    remote: Compressing objects: 100% (253/253), done.
```

```
remote: Total 447 (delta 171), reused 447 (delta 171), pack-reused 0
Receiving objects: 100% (447/447), 34.07 MiB | 42.34 MiB/s, done.
Resolving deltas: 100% (171/171), done.
/content/OIDv4_ToolKit
Requirement already satisfied: pandas in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-package
Collecting awscli
  Downloading https://files.pythonhosted.org/packages/df/6a/0d77c582f0c1ef35e
     |████████████████████████████████| 3.5MB 8.1MB/s
Requirement already satisfied: urllib3 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: opencv-python in /usr/local/lib/python3.6/dist
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/pytho
Requirement already satisfied: PyYAML<5.4,>=3.10; python_version != "3.4" in
Collecting s3transfer<0.4.0,>=0.3.0
  Downloading https://files.pythonhosted.org/packages/69/79/e6afb3d8b0b4e96ce
     |████████████████████████████████| 71kB 10.0MB/s
Collecting botocore==1.19.45
  Downloading https://files.pythonhosted.org/packages/40/c3/1cbe252d7d3674901
     |████████████████████████████████| 7.2MB 47.8MB/s
Collecting rsa<=4.5.0,>=3.1.2; python_version != "3.4"
  Downloading https://files.pythonhosted.org/packages/26/f8/8127fdda0294f0441
Collecting colorama<0.4.4,>=0.2.5; python_version != "3.4"
  Downloading https://files.pythonhosted.org/packages/c9/dc/45cdef1b4d119eb96
Collecting docutils<0.16,>=0.10
  Downloading https://files.pythonhosted.org/packages/22/cd/a6aa959dca619918c
     |████████████████████████████████| 552kB 54.4MB/s
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-pack
Collecting jmespath<1.0.0,>=0.7.1
  Downloading https://files.pythonhosted.org/packages/07/cb/5f001272b6faeb23c
Requirement already satisfied: pyasn1>=0.1.3 in /usr/local/lib/python3.6/dist
ERROR: datascience 0.10.6 has requirement folium==0.2.1, but you'll have foli
ERROR: botocore 1.19.45 has requirement urllib3<1.27,>=1.25.4; python_version
Installing collected packages: jmespath, botocore, s3transfer, rsa, colorama,
  Found existing installation: rsa 4.6
    Uninstalling rsa-4.6:
      Successfully uninstalled rsa-4.6
  Found existing installation: docutils 0.16
    Uninstalling docutils-0.16:
      Successfully uninstalled docutils-0.16
Successfully installed awscli-1.18.205 botocore-1.19.45 colorama-0.4.3 docuti
```

4.2. Download multiple classes in a common folder

```
os.environ['CLASSES'] = ' '.join(classes)
os.environ['N_TRAIN'] = str(n_train)
os.environ['N_VALIDATION'] = str(n_validation)

# See the global variables
!echo -e "CLASSES: " "$CLASSES" \
"\n number of train instances per class: " "$N_TRAIN"\
"\n number of validation instances per class: " "$N_VALIDATION"
```

```
CLASSES:  Vehicle Human_hand Banana
 number of train instances per class:  10
 number of validation instances per class:  5
```

```
%cd /content/OIDv4_ToolKit
# train dataset
!python3 main.py downloader -y --classes $CLASSES --type_csv train --limit $N_TRAIN
# The data set will be saved on path /content/OIDv4_ToolKit/OID/Dataset/train/<clas

# validation dataset
!python3 main.py downloader -y --classes $CLASSES --type_csv validation --limit $N_
# The data set will be saved on path /content/OIDv4_ToolKit/OID/Dataset/validation/
```

```
    /content/OIDv4_ToolKit

                   ___   _____  ______            _    _
                 .'   `.|_   _||_   _ `.         | |  | |
                /  .-.  \ | |    | | `. \ _   __ | |__| |_
                | |   | | | |    | |  | |[ \ [  ]|____   _|
                \  `-'  /_| |_  _| |_.' / \ \/ /     _| |_
                 `.___.'|_____||______.'   \__/     |_____|



                 _____                    _                 _
                (____ \                  | |               | |
                 _   \ \ ___  _ _ _ ____ | | ___   ____  _ | | ____  ____
                | |   | / _ \| | | |  _ \| |/ _ \ / _  |/ || |/ _  )/ ___)
                | |__/ / |_| | | | | | | | | |_| ( ( | ( (_| ( (/ /| |
                |_____/ \___/ \____|_| |_|_|\___/ \_||_|\____|\____)_|


        [INFO] | Downloading ['Vehicle', 'Human hand', 'Banana'] together.
       [ERROR] | Missing the class-descriptions-boxable.csv file.
    [DOWNLOAD] | Automatic download.
    ...145%, 0 MB, 50954 KB/s, 0 seconds passed
    [DOWNLOAD] | File class-descriptions-boxable.csv downloaded into OID/csv_fold
       [ERROR] | Missing the train-annotations-bbox.csv file.
    [DOWNLOAD] | Automatic download.
    ...100%, 1138 MB, 40214 KB/s, 28 seconds passed
    [DOWNLOAD] | File train-annotations-bbox.csv downloaded into OID/csv_folder/t

    Vehicle
        [INFO] | Downloading train images.
        [INFO] | [INFO] Found 15736 online images for train.
        [INFO] | Limiting to 10 images.
        [INFO] | Download of 10 images in train.
    100% 10/10 [00:09<00:00,  1.05it/s]
        [INFO] | Done!
        [INFO] | Creating labels for Vehicle of train.
        [INFO] | Labels creation completed.

    Human hand
        [INFO] | Downloading train images.
        [INFO] | [INFO] Found 22093 online images for train.
        [INFO] | Limiting to 10 images.
        [INFO] | Download of 10 images in train.
    100% 10/10 [00:09<00:00,  1.06it/s]
        [INFO] | Done!
        [INFO] | Creating labels for Human hand of train.
        [INFO] | Labels creation completed.

    Banana
        [INFO] | Downloading train images.
```

```
          [INFO] | [INFO] Found 723 online images for train.
          [INFO] | Limiting to 10 images.
          [INFO] | Download of 10 images in train.
      100% 10/10 [00:09<00:00,  1.08it/s]
          [INFO] | Done!
          [INFO] | Creating labels for Banana of train.
          [INFO] | Labels creation completed.
```

4.3. Correct directory names

When class is build of more than one word, by default directory name join these words with " " instead of "_" f.e.:

`classes = ['Vehicle', 'Human_hand', 'Banana']` by default as multiclass saved on path: `./OIDv4_ToolKit/OID/Dataset/train/Vehicle_Human hand_Banana`

In this step directory will be renamed by replacing " " with "_" f.e.:

`Vehicle_Human hand_Banana` ---> `Vehicle_Human_hand_Banana`

```
toolkit_dataset_path = '/content/OIDv4_ToolKit/OID/Dataset'
os.chdir(toolkit_dataset_path)
dirs = [dir for dir in os.listdir(toolkit_dataset_path) if os.path.isdir(dir)]

for dir in dirs:
  path = os.path.join(toolkit_dataset_path, dir)
  for dirname in os.listdir(path):
    if ' ' in dirname:
      new_dirname = dirname.replace(' ', '_')
      print('%10s: %s ---> %s' % (dir, dirname, new_dirname))
      os.rename(os.path.join(path, dirname), os.path.join(path, new_dirname))
```

```
    validation: Vehicle_Human hand_Banana ---> Vehicle_Human_hand_Banana
         train: Vehicle_Human hand_Banana ---> Vehicle_Human_hand_Banana
```

4.4. Convert annotations

```
%cd /content/OIDv4_ToolKit

# Pepare classes.txt file
with open('classes.txt', 'w') as f:
  for cls in classes:
    f.write(f'{cls}\n')

!cat classes.txt
```

```
    /content/OIDv4_ToolKit
    Vehicle
    Human_hand
    Banana
```

```
# This creates single txt file for each image with normalized annotations
%cd /content/OIDv4_ToolKit

!python3 convert_annotations.py
```

```
    /content/OIDv4_ToolKit
    > Currently in subdirectory: validation
    > Converting annotations for class: Vehicle_Human_hand_Banana
    100% 15/15 [00:00<00:00, 50.71it/s]
    > Currently in subdirectory: train
    > Converting annotations for class: Vehicle_Human_hand_Banana
    100% 30/30 [00:01<00:00, 18.03it/s]
```

4.4.1. Delete old unnecesary `Label` directories

```
# Delete unnecessary folders with old labels
os.environ['PROJECT_DIR'] = '_'.join(classes)
!echo "$PROJECT_DIR"

!rm -r OID/Dataset/train/"$PROJECT_DIR"/Label
!rm -r OID/Dataset/validation/"$PROJECT_DIR"/Label
```

```
    Vehicle_Human_hand_Banana
```

4.5. Copy images to Google Drive

```
def create_path(path):
  '''Creates path if does not exist.'''
  if not os.path.exists(path):
    os.mkdir(path)


mydrive = '/mydrive' # symbolic link of "/content/gdrive/MyDrive"
yolov4_dir = 'yolov4'
project_dir = '_'.join(classes)
data_dir = 'data'
backup_dir = 'backup'

# Make sure yolov4 folder exists on path '/mydrive'
yolov4_path = os.path.join(mydrive, yolov4_dir)
print(f'%15s : %s' % ('yolov4_path', yolov4_path))
create_path(yolov4_path)

# Make sure <project_dir> folder exists on path '/mydrive/yolov4'
project_path = os.path.join(yolov4_path, project_dir)
print(f'%15s : %s' % ('project_path', project_path))
create_path(project_path)

# Make sure data folder exists on path /mydrive/yolov4/<project_dir>/data
data_path = os.path.join(project_path, data_dir)
print(f'%15s : %s' % ('data_path', data_path))
print(f'%15s : %s' % ('data_path', data_path))
create_path(data_path)


# Make sure backup folder exists on path /mydrive/yolov4/<project_dir>/backup
backup_path = os.path.join(project_path, backup_dir)
print(f'%15s : %s' % ('backup_path', backup_path))
create_path(backup_path)
```

```
        yolov4_path : /mydrive/yolov4
       project_path : /mydrive/yolov4/Vehicle_Human_hand_Banana
          data_path : /mydrive/yolov4/Vehicle_Human_hand_Banana/data
        backup_path : /mydrive/yolov4/Vehicle_Human_hand_Banana/backup
```

```
# Move downloaded images to mounted Google Drive

os.environ['PROJECT_PATH'] = project_path

!echo $PROJECT_PATH

!cp -r /content/OIDv4_ToolKit/OID/Dataset/train/"$PROJECT_DIR" "$PROJECT_PATH"/data
!cp -r /content/OIDv4_ToolKit/OID/Dataset/validation/"$PROJECT_DIR" "$PROJECT_PATH'
```

```
    /mydrive/yolov4/Vehicle_Human_hand_Banana
```

5) Prepare Daknet

Darknet is an open source neural network framework written in C and CUDA.

5.1. Download

```
%cd /content
!git clone https://github.com/patryklaskowski/darknet
```

```
    /content
    Cloning into 'darknet'...
    remote: Enumerating objects: 14240, done.
    remote: Total 14240 (delta 0), reused 0 (delta 0), pack-reused 14240
    Receiving objects: 100% (14240/14240), 12.88 MiB | 24.25 MiB/s, done.
    Resolving deltas: 100% (9704/9704), done.
```

5.2. Configure `Makefile` for GPU and OpenCv

```
# verify CUDA
!/usr/local/cuda/bin/nvcc --version
```

```
    nvcc: NVIDIA (R) Cuda compiler driver
    Copyright (c) 2005-2019 NVIDIA Corporation
    Built on Sun_Jul_28_19:07:16_PDT_2019
    Cuda compilation tools, release 10.1, V10.1.243
```

```
# Change Makefile to have GPU and OPENCV enabled
%cd /content/darknet

!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
```

```
    /content/darknet
```

5.3. Build up

```
# build darknet
!make
```

```
    mkdir -p ./obj/
    mkdir -p backup
    chmod +x *.sh
    g++ -std=c++11 -std=c++11 -Iinclude/ -I3rdparty/stb/include -DOPENCV `pkg-con
    ./src/image_opencv.cpp: In function ‘void draw_detections_cv_v3(void**, detec
    ./src/image_opencv.cpp:926:23: warning: variable ‘rgb’ set but not used [-Wun
                     float rgb[3];
                           ^~~
    ./src/image_opencv.cpp: In function ‘void draw_train_loss(char*, void**, int,
    ./src/image_opencv.cpp:1127:13: warning: this ‘if’ clause does not guard... [
                 if (iteration_old == 0)
                 ^~
    ./src/image_opencv.cpp:1130:10: note: ...this statement, but the latter is mi
              if (iteration_old != 0){
              ^~
    ./src/image_opencv.cpp: In function ‘void cv_draw_object(image, float*, int,
    ./src/image_opencv.cpp:1424:14: warning: unused variable ‘buff’ [-Wunused-var
             char buff[100];
                  ^~~~
    ./src/image_opencv.cpp:1400:9: warning: unused variable ‘it_tb_res’ [-Wunused
         int it_tb_res = cv::createTrackbar(it_trackbar_name, window_name, &it_tr
             ^~~~~~~~~
    ./src/image_opencv.cpp:1404:9: warning: unused variable ‘lr_tb_res’ [-Wunused
         int lr_tb_res = cv::createTrackbar(lr_trackbar_name, window_name, &lr_tr
             ^~~~~~~~~
    ./src/image_opencv.cpp:1408:9: warning: unused variable ‘cl_tb_res’ [-Wunused
         int cl_tb_res = cv::createTrackbar(cl_trackbar_name, window_name, &cl_tr
             ^~~~~~~~~
    ./src/image_opencv.cpp:1411:9: warning: unused variable ‘bo_tb_res’ [-Wunused
         int bo_tb_res = cv::createTrackbar(bo_trackbar_name, window_name, boxonl
             ^~~~~~~~~
    g++ -std=c++11 -std=c++11 -Iinclude/ -I3rdparty/stb/include -DOPENCV `pkg-con
    In file included from ./src/http_stream.cpp:580:0:
    ./src/httplib.h:129:0: warning: "INVALID_SOCKET" redefined
     #define INVALID_SOCKET (-1)

    ./src/http_stream.cpp:73:0: note: this is the location of the previous defini
     #define INVALID_SOCKET -1
```

```
./src/http_stream.cpp: In member function ‘bool JSON_sender::write(const char
./src/http_stream.cpp:249:21: warning: unused variable ‘n’ [-Wunused-variable
                 int n = _write(client, outputbuf, outlen);
                     ^
./src/http_stream.cpp: In member function ‘bool MJPG_sender::write(const cv::
./src/http_stream.cpp:507:113: warning: format ‘%zu’ expects argument of type
                 sprintf(head, "--mjpegstream\r\nContent-Type: image/jpeg\r\n

./src/http_stream.cpp: In function ‘void set_track_id(detection*, int, float,
./src/http_stream.cpp:845:27: warning: comparison between signed and unsigned
         for (int i = 0; i < v.size(); ++i) {
                         ~~^~~~~~~~~~
./src/http_stream.cpp:853:33: warning: comparison between signed and unsigned
     for (int old_id = 0; old_id < old_dets.size(); ++old_id) {
                          ~~~~~~~^~~~~~~~~~~~~~~~~
./src/http_stream.cpp:873:31: warning: comparison between signed and unsigned
     for (int index = 0; index < new_dets_num*old_dets.size(); ++index) {
                         ~~~~~~^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
./src/http_stream.cpp:908:28: warning: comparison between signed and unsigned
```

5.4. Check

```
# Helper function
import cv2
import matplotlib.pyplot as plt
%matplotlib inline

def imshow(path):
  '''Show image from path.'''
  img = cv2.imread(path)
  plt.figure(figsize=(7, 7))
  plt.title(path)
  plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
  plt.axis("off")
  plt.show()


imshow('data/dog.jpg')
```

data/dog.jpg

5.4.1. Download pre-trained YOLO v4 weights

Trained on COCO dataset containing 80 classes.

```
# Download pre-trained weights
!wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optima

    --2020-12-30 18:41:16--  https://github.com/AlexeyAB/darknet/releases/downloa
    Resolving github.com (github.com)... 192.30.255.112
    Connecting to github.com (github.com)|192.30.255.112|:443... connected.
    HTTP request sent, awaiting response... 302 Found
    Location: https://github-production-release-asset-2e65be.s3.amazonaws.com/753
    --2020-12-30 18:41:16--  https://github-production-release-asset-2e65be.s3.am
    Resolving github-production-release-asset-2e65be.s3.amazonaws.com (github-pro
    Connecting to github-production-release-asset-2e65be.s3.amazonaws.com (github
    HTTP request sent, awaiting response... 200 OK
    Length: 257717640 (246M) [application/octet-stream]
    Saving to: 'yolov4.weights'

    yolov4.weights      100%[===================>] 245.78M  48.2MB/s    in 5.5s

    2020-12-30 18:41:22 (44.6 MB/s) - 'yolov4.weights' saved [257717640/257717640
```

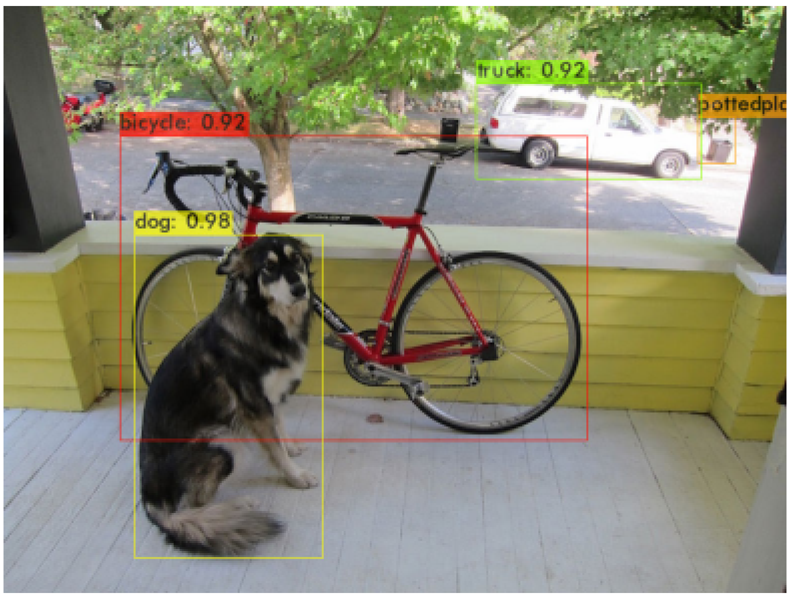
5.4.2. Predict

```
# Make prediction
!./darknet detector test cfg/coco.data cfg/yolov4.cfg yolov4.weights data/dog.jpg -

     CUDA-version: 10010 (10010), cuDNN: 7.6.5, CUDNN_HALF=1, GPU count: 1
     CUDNN_HALF=1
     OpenCV version: 3.2.0
     0 : compute_capability = 750, cudnn_half = 1, GPU: Tesla T4
    net.optimized_memory = 0
    mini_batch = 1, batch = 8, time_steps = 1, train = 0
       layer   filters  size/strd(dil)      input                output
       0 conv     32       3 x 3/ 1    608 x 608 x   3 ->  608 x 608 x  32 0.639
       1 conv     64       3 x 3/ 2    608 x 608 x  32 ->  304 x 304 x  64 3.407
       2 conv     64       1 x 1/ 1    304 x 304 x  64 ->  304 x 304 x  64 0.757
       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
       5 conv     32       1 x 1/ 1    304 x 304 x  64 ->  304 x 304 x  32 0.379
       6 conv     64       3 x 3/ 1    304 x 304 x  32 ->  304 x 304 x  64 3.407
       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
       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
      11 conv    128       3 x 3/ 2    304 x 304 x  64 ->  152 x 152 x 128 3.407
      12 conv     64       1 x 1/ 1    152 x 152 x 128 ->  152 x 152 x  64 0.379
      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
      15 conv     64       1 x 1/ 1    152 x 152 x  64 ->  152 x 152 x  64 0.189
      16 conv     64       3 x 3/ 1    152 x 152 x  64 ->  152 x 152 x  64 1.703
```

```
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
19 conv     64       3 x 3/ 1    152 x 152 x  64 ->  152 x 152 x  64 1.703
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
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
24 conv    256       3 x 3/ 2    152 x 152 x 128 ->   76 x  76 x 256 3.407
25 conv    128       1 x 1/ 1     76 x  76 x 256 ->   76 x  76 x 128 0.379
26 route  24                                     ->   76 x  76 x 256
27 conv    128       1 x 1/ 1     76 x  76 x 256 ->   76 x  76 x 128 0.379
28 conv    128       1 x 1/ 1     76 x  76 x 128 ->   76 x  76 x 128 0.189
29 conv    128       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 128 1.703
30 Shortcut Layer: 27,  wt = 0, wn = 0, outputs:  76 x  76 x 128 0.001 BF
31 conv    128       1 x 1/ 1     76 x  76 x 128 ->   76 x  76 x 128 0.189
32 conv    128       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 128 1.703
33 Shortcut Layer: 30,  wt = 0, wn = 0, outputs:  76 x  76 x 128 0.001 BF
34 conv    128       1 x 1/ 1     76 x  76 x 128 ->   76 x  76 x 128 0.189
35 conv    128       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 128 1.703
36 Shortcut Layer: 33,  wt = 0, wn = 0, outputs:  76 x  76 x 128 0.001 BF
37 conv    128       1 x 1/ 1     76 x  76 x 128 ->   76 x  76 x 128 0.189
38 conv    128       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 128 1.703
39 Shortcut Layer: 36,  wt = 0, wn = 0, outputs:  76 x  76 x 128 0.001 BF
40 conv    128       1 x 1/ 1     76 x  76 x 128 ->   76 x  76 x 128 0.189
41 conv    128       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 128 1.703
42 Shortcut Layer: 39,  wt = 0, wn = 0, outputs:  76 x  76 x 128 0.001 BF
43 conv    128       1 x 1/ 1     76 x  76 x 128 ->   76 x  76 x 128 0.189
44 conv    128       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 128 1.703
45 Shortcut Layer: 42,  wt = 0, wn = 0, outputs:  76 x  76 x 128 0.001 BF
46 conv    128       1 x 1/ 1     76 x  76 x 128 ->   76 x  76 x 128 0.189
47 conv    128       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 128 1.703
48 Shortcut Layer: 45,  wt = 0, wn = 0, outputs:  76 x  76 x 128 0.001 BF
49 conv    128       1 x 1/ 1     76 x  76 x 128 ->   76 x  76 x 128 0.189
50 conv    128       3 x 3/ 1     76 x  76 x 128 ->   76 x  76 x 128 1.703
51 Shortcut Layer: 48,  wt = 0, wn = 0, outputs:  76 x  76 x 128 0.001 BF
```

```
imshow('predictions.jpg')
```

predictions.jpg



5.5. `.cfg` file configuration

Custom object detection cfg oryginal setup [instruction]

width & height

- any value multiple of 32 (416 is standard). Increase to imporove results e.g. 640. Remember that increasing the size slows down training.

max_batches

- (# of classes) * 2000 (but no less than 6000)

steps

- (80% of max_batches), (90% of max_batches)

filters

- (# of classes + 5) * 3

random

- (optional) random = 0 to speed up training but slightly reduce accuracy of model. Help to save memory if you run into any memory issues

Example for 2 classes:

- [x] width=128
- [x] height=128
- [x] max_batches = 6000
- [x] steps=4800,5400
- [x] classes=2
- [x] filters=21
- [x] random=0

5.5.1. Copy `.cfg` file from Darknet directory to Your Google Drive project path.

```
os.environ['CFG_FILE'] = os.path.join(project_path, 'yolov4-custom.cfg')
!echo -e $CFG_FILE

!cp /content/darknet/cfg/yolov4-custom.cfg "$CFG_FILE"
# !head -n 24 "$CFG_FILE"
```

```
    /mydrive/yolov4/Vehicle_Human_hand_Banana/yolov4-custom.cfg
```

5.5.2. Make changes in `.cfg` file

```
os.environ['SIZE'] = str(size)
```

```
!echo "SIZE: ""$SIZE"

max_batches = len(classes) * 2000 if len(classes) * 2000 >= 6000 else 6000
os.environ['MAX_BATCHES'] = str(max_batches)
!echo "MAX_BATCHES: ""$MAX_BATCHES"

steps = [str(int(x*y)) for x, y in zip([max_batches, max_batches], [0.8, 0.9])]
os.environ['STEPS'] = ','.join(steps)
!echo "STEPS: ""$STEPS"

os.environ['N_CLASSES'] = str(len(classes))
!echo "N_CLASSES: ""$N_CLASSES"

os.environ['FILTERS'] = str((len(classes) + 5) * 3)
!echo "FILTERS: ""$FILTERS"
```

```
    SIZE: 256
    MAX_BATCHES: 6000
    STEPS: 4800,5400
    N_CLASSES: 3
    FILTERS: 24
```

```
# Height and width (any multiple of 32, where 416 px is standard)
!sed -i "s/width=608/width=""$SIZE""/" "$CFG_FILE"
!sed -i "s/height=608/height=""$SIZE""/" "$CFG_FILE"

# Max batches = (# of classes) * 2000 (but no less than 6000)
!sed -i "s/max_batches = 500500/max_batches = ""$MAX_BATCHES""/" "$CFG_FILE"

# Steps = (80% of max_batches), (90% of max_batches)
!sed -i "s/steps=400000,450000/steps=""$STEPS""/" "$CFG_FILE"

# Number of classes
!sed -i "s/classes=80/classes=""$N_CLASSES""/" "$CFG_FILE"

# Filters = (# of classes + 5) * 3
!sed -i "s/filters=255/filters=""$FILTERS""/" "$CFG_FILE"

# Random
!sed -i "s/random=1/random=0/" "$CFG_FILE"

!head -n 24 "$CFG_FILE"
```

```
    [net]
    # Testing
    #batch=1
    #subdivisions=1
    # Training
    batch=64
    subdivisions=16
    width=256
    height=256
    channels=3
    momentum=0.949
    decay=0.0005
    angle=0
```

```
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches = 6000
policy=steps
steps=4800,5400
scales=.1,.1
```

5.6. `.names` file configuration

```
os.environ['NAMES_FILE'] = os.path.join(data_path, 'yolov4-custom.names')
!echo -e "$NAMES_FILE"

# Remember: in .names file ORDER matters but not exact names
!cat /content/OIDv4_ToolKit/classes.txt > "$NAMES_FILE"
!cat "$NAMES_FILE"
```

```
/mydrive/yolov4/Vehicle_Human_hand_Banana/data/yolov4-custom.names
Vehicle
Human_hand
Banana
```

5.7. `.data` file configuration

[ ] ↳ *1 cell hidden*

5.8. Generate train.txt and validation.txt

It's time to generate `train.txt` and `validation.txt` which paths has been provided in `.data` file.

Both `tain.txt` / `validation.txt` contain absolute paths to train/validation images.

```
# generate_train.py

#
# Creates train.txt file where all train images paths are listed.
# Save path: /mydrive/yolov4/<project_dir>/data/train.txt
#

import os

# Path to directory with images to train on
# /mydrive/yolov4/<project_dir>/data/train/
train_path = os.path.join(data_path, 'train')
```

```
image_files = []
for filename in os.listdir(train_path):
  if filename.endswith('.jpg'):
    image_files.append(os.path.join(train_path, filename))

# /mydrive/yolov4/<project_dir>/data/
os.chdir(data_path)
print(data_path)
with open("train.txt", "w") as file:
  for image in image_files:
    file.write(f'{image}\n')

print(f'Found {len(image_files)} train images total ({len(image_files)/len(classes)
```

```
/mydrive/yolov4/Vehicle_Human_hand_Banana/data
Found 30 train images total (10.0 per class).
```

```
!head -n 5 $PROJECT_PATH/data/train.txt
```

```
/mydrive/yolov4/Vehicle_Human_hand_Banana/data/train/f8773f11ed5604da.jpg
/mydrive/yolov4/Vehicle_Human_hand_Banana/data/train/ee7fab74a6efcbe6.jpg
/mydrive/yolov4/Vehicle_Human_hand_Banana/data/train/0c6bf0305bf365a2.jpg
/mydrive/yolov4/Vehicle_Human_hand_Banana/data/train/010490795874c6dc.jpg
/mydrive/yolov4/Vehicle_Human_hand_Banana/data/train/7aff32eacb705c36.jpg
```

```
# generate_validation.py

#
# Creates validation.txt file where all validation images paths are listed.
# Save path: /mydrive/yolov4/<project_dir>/data/validation.txt
#

import os

# Path to directory with images to validate on
# /mydrive/yolov4/<project_dir>/data/validation/
validation_path = os.path.join(data_path, 'validation')

image_files = []
for filename in os.listdir(validation_path):
  if filename.endswith('.jpg'):
    image_files.append(os.path.join(validation_path, filename))

# /mydrive/yolov4/<project_dir>/data/
os.chdir(data_path)
print(data_path)
with open("validation.txt", "w") as file:
  for image in image_files:
    file.write(f'{image}\n')

print(f'Found {len(image_files)} train images total ({len(image_files)/len(classes)
```

```
/mydrive/yolov4/Vehicle_Human_hand_Banana/data
Found 15 train images total (5.0 per class).
```

```
!head -n 5 $PROJECT_PATH/data/validation.txt

    /mydrive/yolov4/Vehicle_Human_hand_Banana/data/validation/04833bdaa8c68594.jp
    /mydrive/yolov4/Vehicle_Human_hand_Banana/data/validation/7fa25536f608af03.jp
    /mydrive/yolov4/Vehicle_Human_hand_Banana/data/validation/e86b1d0bf7235885.jp
    /mydrive/yolov4/Vehicle_Human_hand_Banana/data/validation/b63576d39182fadb.jp
    /mydrive/yolov4/Vehicle_Human_hand_Banana/data/validation/cc94d1513871c552.jp
```

5.9. Download weights for training

```
%cd /content/darknet
!wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal

    /content/darknet
    --2020-12-30 18:41:35--  https://github.com/AlexeyAB/darknet/releases/downloa
    Resolving github.com (github.com)... 192.30.255.113
    Connecting to github.com (github.com)|192.30.255.113|:443... connected.
    HTTP request sent, awaiting response... 302 Found
    Location: https://github-production-release-asset-2e65be.s3.amazonaws.com/753
    --2020-12-30 18:41:35--  https://github-production-release-asset-2e65be.s3.am
    Resolving github-production-release-asset-2e65be.s3.amazonaws.com (github-pro
    Connecting to github-production-release-asset-2e65be.s3.amazonaws.com (github
    HTTP request sent, awaiting response... 200 OK
    Length: 170038676 (162M) [application/octet-stream]
    Saving to: ‘yolov4.conv.137’

    yolov4.conv.137     100%[===================>] 162.16M  44.6MB/s    in 4.1s

    2020-12-30 18:41:40 (39.7 MB/s) - ‘yolov4.conv.137’ saved [170038676/17003867
```

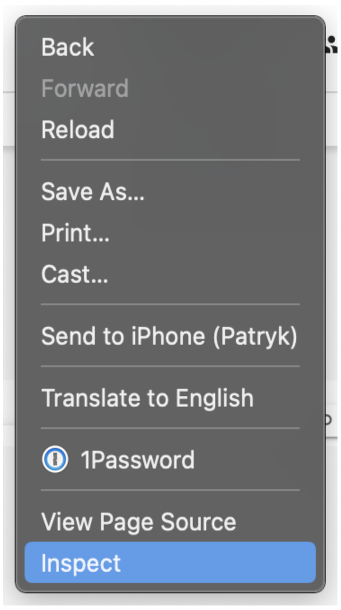
6) Train

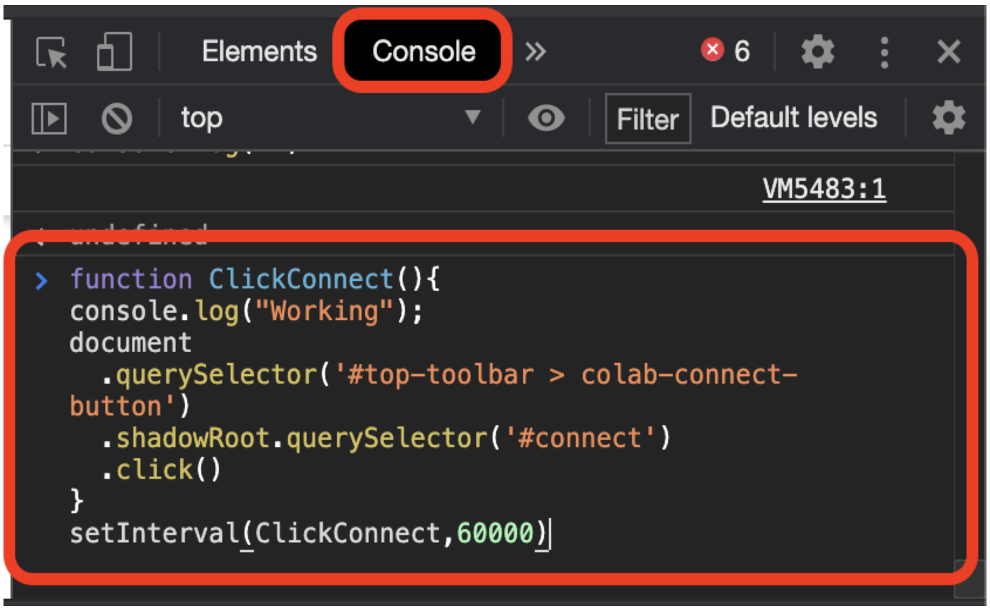
6.1. Prevent idle disconnection

Google Colab check your activity. If You are idle for about 90 minutes, kicks You off the runtime.

To simulate activity copy and paste in browser console below javascript code.

```
function ClickConnect(){
console.log("Working");
document
  .querySelector('#top-toolbar > colab-connect-button')
  .shadowRoot.querySelector('#connect')
  .click()
}
setInterval(ClickConnect,60000)
```





```
> function ClickConnect(){
  console.log("Working");
  document
    .querySelector('#top-toolbar > colab-connect-
  button')
    .shadowRoot.querySelector('#connect')
    .click()
  }
  setInterval(ClickConnect,60000)
```

6.2. Prepare `backup_chart()` function

Additional funtion that will prevent chart loss with visualuisation of training performance in case the runtime will be shomehow disconnected.

Use Thread to continously save/overwrite chart to Google Drive project backup directory every 60 seconds (default).

```
import time
from shutil import copyfile

def backup_chart(path, event, chart='/content/darknet/chart.png', wait=60):
  '''Function meant to be running using Thread object.

  Make sure that every 60 seconds (default) the chart presenting train performance
  is copied (overwritted is exist) into new path.
  This prevent data loss when runtime is interrupted.'''

  print('\n>>> backup_chart START.')
  while not event.is_set():
    if os.path.exists(chart):
      print(f'> Chart path "{chart}" found, copying to "{path}" ...')
      copyfile(chart, path)
    else:
      print(f'> Chart path "{chart}" not found...')
    print(f'> Thread is now waiting {wait} second(s)...')
    interrupted = event.wait(wait)
    if interrupted:
      print(f'> Thread waiting has been interrupted...')
  print('\n>>> backup_chart FINISH.')
```

6.3. START!

```
from threading import Thread, Event

backup_chart_path = os.path.join(project_path, 'chart.png')

# Event allows simple communication between threads
event = Event()
print(f'event.is_set(): {event.is_set()}')

# Run Thread
t = Thread(target=backup_chart, name='backup_chart', kwargs={'path': backup_chart_
t.start()

# Train
%cd /content/darknet
!./darknet detector train "$DATA_FILE" "$CFG_FILE" yolov4.conv.137 -dont_show -map
```

Streaming output truncated to the last 5000 lines.

```
 total_bbox = 4382283, rewritten_bbox = 1.844586 %
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.89078
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.89673
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.91456
 total_bbox = 4382334, rewritten_bbox = 1.844565 %
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.92279
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.87469
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.90110
 total_bbox = 4382377, rewritten_bbox = 1.844547 %
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.89638
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.91272
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.80638
 total_bbox = 4382406, rewritten_bbox = 1.844535 %
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.95242
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.91549
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.87841
 total_bbox = 4382423, rewritten_bbox = 1.844528 %
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.70782
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.90044
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.94104
 total_bbox = 4382442, rewritten_bbox = 1.844520 %
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.82575
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.90474
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.90188
 total_bbox = 4382537, rewritten_bbox = 1.844480 %
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.84189
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.88086
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.92975
 total_bbox = 4382606, rewritten_bbox = 1.844473 %
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.87850
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.94148
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.95652
 total_bbox = 4382632, rewritten_bbox = 1.844462 %
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.88909
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.85547
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.89039
 total_bbox = 4382644, rewritten_bbox = 1.844457 %
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.66273
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.88727
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.93229
 total_bbox = 4382665, rewritten_bbox = 1.844471 %
```

```
    v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.91085
    v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.94235
    v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.93104
     total_bbox = 4382688, rewritten_bbox = 1.844462 %
    v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.72373
    v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.98038
    v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.96082
     total_bbox = 4382702, rewritten_bbox = 1.844479 %
    v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.87506
    v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.94011
    v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.89491
     total_bbox = 4382728, rewritten_bbox = 1.844468 %
    v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.84586
    v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.93602
    v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.94508
     total_bbox = 4382754, rewritten_bbox = 1.844480 %
    v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.75979
```

6.4. **CAUTION!** Interrupt the thread

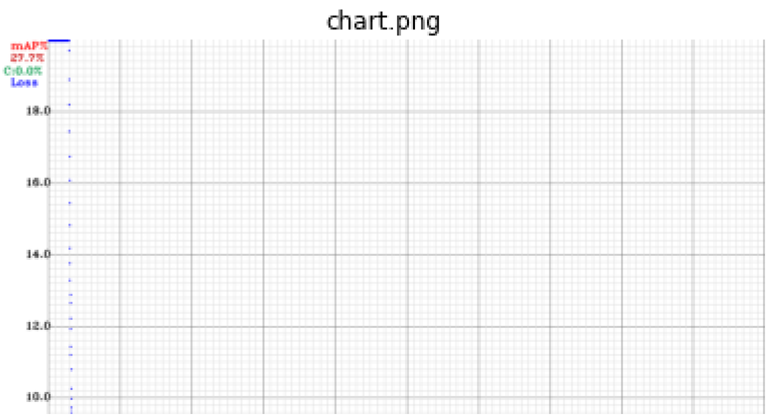
If You interrupted execution or training time finished or by whatever reason training has been stopped, **remember to shut down** thread that is running in background.

```
# Interrupt

print(f'Is Thread alive?: {t.is_alive()}')
event.set()
t.join()
print(f'Is Thread alive?: {t.is_alive()}')
print('Done.')
```

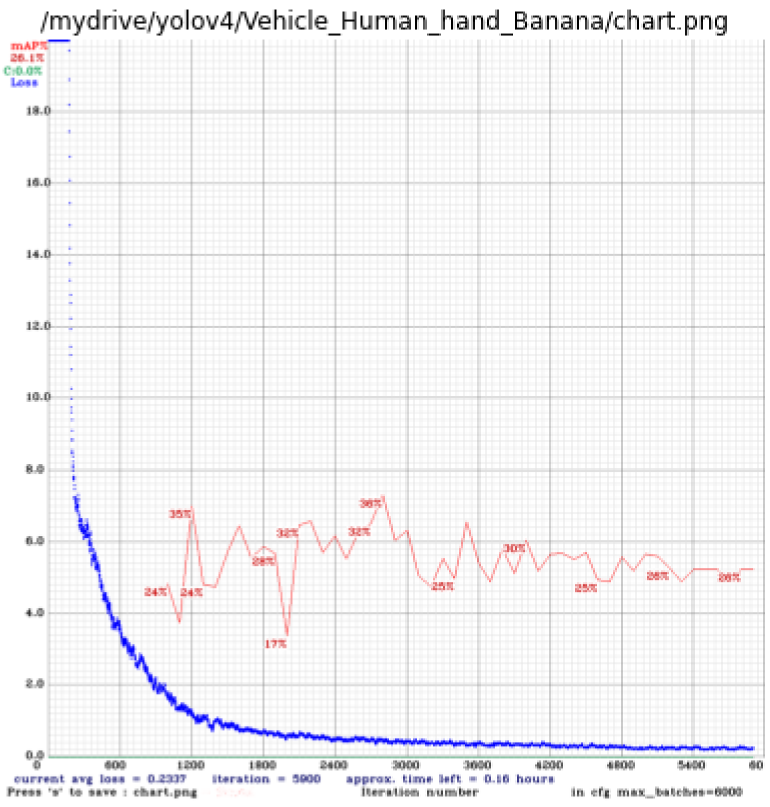
```
    Is Thread alive?: True
    > Thread waiting has been interrupted...

    >>> backup_chart FINISH.
    Is Thread alive?: False
    Done.
```

6.6. Plot training performance

```
imshow('chart.png')
```



```
imshow(backup_chart_path)
```



```
assert False
```

7) Evaluate

7.1. mAP (Mean Average Precision)

```
# mean average precision (mAP)
!./darknet detector map "$DATA_FILE" "$CFG_FILE" "$PROJECT_PATH"/backup/yolov4-cus
```

```
      CUDA-version: 10010 (10010), cuDNN: 7.6.5, CUDNN_HALF=1, GPU count: 1
      OpenCV version: 3.2.0
      0 : compute_capability = 750, cudnn_half = 1, GPU: Tesla T4
        layer   filters  size/strd(dil)      input                output
```

```
   0 conv     32       3 x 3/ 1    256 x 256 x   3 ->  256 x 256 x  32 0.113
   1 conv     64       3 x 3/ 2    256 x 256 x  32 ->  128 x 128 x  64 0.604
   2 conv     64       1 x 1/ 1    128 x 128 x  64 ->  128 x 128 x  64 0.134
   3 route  1                                      ->  128 x 128 x  64
   4 conv     64       1 x 1/ 1    128 x 128 x  64 ->  128 x 128 x  64 0.134
   5 conv     32       1 x 1/ 1    128 x 128 x  64 ->  128 x 128 x  32 0.067
   6 conv     64       3 x 3/ 1    128 x 128 x  32 ->  128 x 128 x  64 0.604
   7 Shortcut Layer: 4,  wt = 0, wn = 0, outputs: 128 x 128 x  64 0.001 BF
   8 conv     64       1 x 1/ 1    128 x 128 x  64 ->  128 x 128 x  64 0.134
   9 route  8 2                                    ->  128 x 128 x 128
  10 conv     64       1 x 1/ 1    128 x 128 x 128 ->  128 x 128 x  64 0.268
  11 conv    128       3 x 3/ 2    128 x 128 x  64 ->   64 x  64 x 128 0.604
  12 conv     64       1 x 1/ 1     64 x  64 x 128 ->   64 x  64 x  64 0.067
  13 route  11                                     ->   64 x  64 x 128
  14 conv     64       1 x 1/ 1     64 x  64 x 128 ->   64 x  64 x  64 0.067
  15 conv     64       1 x 1/ 1     64 x  64 x  64 ->   64 x  64 x  64 0.034
  16 conv     64       3 x 3/ 1     64 x  64 x  64 ->   64 x  64 x  64 0.302
  17 Shortcut Layer: 14,  wt = 0, wn = 0, outputs:  64 x  64 x  64 0.000 BF
  18 conv     64       1 x 1/ 1     64 x  64 x  64 ->   64 x  64 x  64 0.034
  19 conv     64       3 x 3/ 1     64 x  64 x  64 ->   64 x  64 x  64 0.302
  20 Shortcut Layer: 17,  wt = 0, wn = 0, outputs:  64 x  64 x  64 0.000 BF
  21 conv     64       1 x 1/ 1     64 x  64 x  64 ->   64 x  64 x  64 0.034
  22 route  21 12                                  ->   64 x  64 x 128
  23 conv    128       1 x 1/ 1     64 x  64 x 128 ->   64 x  64 x 128 0.134
  24 conv    256       3 x 3/ 2     64 x  64 x 128 ->   32 x  32 x 256 0.604
  25 conv    128       1 x 1/ 1     32 x  32 x 256 ->   32 x  32 x 128 0.067
  26 route  24                                     ->   32 x  32 x 256
  27 conv    128       1 x 1/ 1     32 x  32 x 256 ->   32 x  32 x 128 0.067
  28 conv    128       1 x 1/ 1     32 x  32 x 128 ->   32 x  32 x 128 0.034
  29 conv    128       3 x 3/ 1     32 x  32 x 128 ->   32 x  32 x 128 0.302
  30 Shortcut Layer: 27,  wt = 0, wn = 0, outputs:  32 x  32 x 128 0.000 BF
  31 conv    128       1 x 1/ 1     32 x  32 x 128 ->   32 x  32 x 128 0.034
  32 conv    128       3 x 3/ 1     32 x  32 x 128 ->   32 x  32 x 128 0.302
  33 Shortcut Layer: 30,  wt = 0, wn = 0, outputs:  32 x  32 x 128 0.000 BF
  34 conv    128       1 x 1/ 1     32 x  32 x 128 ->   32 x  32 x 128 0.034
  35 conv    128       3 x 3/ 1     32 x  32 x 128 ->   32 x  32 x 128 0.302
  36 Shortcut Layer: 33,  wt = 0, wn = 0, outputs:  32 x  32 x 128 0.000 BF
  37 conv    128       1 x 1/ 1     32 x  32 x 128 ->   32 x  32 x 128 0.034
  38 conv    128       3 x 3/ 1     32 x  32 x 128 ->   32 x  32 x 128 0.302
  39 Shortcut Layer: 36,  wt = 0, wn = 0, outputs:  32 x  32 x 128 0.000 BF
  40 conv    128       1 x 1/ 1     32 x  32 x 128 ->   32 x  32 x 128 0.034
  41 conv    128       3 x 3/ 1     32 x  32 x 128 ->   32 x  32 x 128 0.302
  42 Shortcut Layer: 39,  wt = 0, wn = 0, outputs:  32 x  32 x 128 0.000 BF
  43 conv    128       1 x 1/ 1     32 x  32 x 128 ->   32 x  32 x 128 0.034
  44 conv    128       3 x 3/ 1     32 x  32 x 128 ->   32 x  32 x 128 0.302
  45 Shortcut Layer: 42,  wt = 0, wn = 0, outputs:  32 x  32 x 128 0.000 BF
  46 conv    128       1 x 1/ 1     32 x  32 x 128 ->   32 x  32 x 128 0.034
  47 conv    128       3 x 3/ 1     32 x  32 x 128 ->   32 x  32 x 128 0.302
  48 Shortcut Layer: 45,  wt = 0, wn = 0, outputs:  32 x  32 x 128 0.000 BF
  49 conv    128       1 x 1/ 1     32 x  32 x 128 ->   32 x  32 x 128 0.034
  50 conv    128       3 x 3/ 1     32 x  32 x 128 ->   32 x  32 x 128 0.302
  51 Shortcut Layer: 48,  wt = 0, wn = 0, outputs:  32 x  32 x 128 0.000 BF
  52 conv    128       1 x 1/ 1     32 x  32 x 128 ->   32 x  32 x 128 0.034
  53 route  52 25                                  ->   32 x  32 x 256
  54 conv    256       1 x 1/ 1     32 x  32 x 256 ->   32 x  32 x 256 0.134
```

```
# see results
!cat result.txt | grep "mean average precision"
```

```
    calculation mAP (mean average precision)...
    mean average precision (mAP@0.50) = 0.276667, or 27.67 %
```

7.2. Object detection

RUN Object detection

```
# need to set our custom cfg to test mode
%cd cfg
!sed -i 's/batch=64/batch=1/' yolov4-obj.cfg
!sed -i 's/subdivisions=16/subdivisions=1/' yolov4-obj.cfg
%cd ..
```

```
!./darknet detector test data/obj.data cfg/yolov4-obj.cfg /mydrive/yolov4/backup/yolov4-o
```

```
#
#
# TODO
#
#
```

```
assert False
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-44-a871fdc9ebee> in <module>()
----> 1 assert False

AssertionError:
```

SEARCH STACK OVERFLOW

Re-run training

Re-run training from last checkpoint e.g. `yolov4-custom_last.weights`.
Weights backup has been done automatically during model training. `.weights` files were saved to mounted Google Drive on path:
`./yolov4/Vehicle_Human_hand/backup`

At that time you may have to rerun some required code.
Below script is enough to restart training from any checkpoint, assuming that runtime has been disconnected and directory in Google Colab has been initialized correctly.

```
  # Python variables
  classes = []

  # Python functions
  # backup_chart()

  # Environmental variables

  # Darknet

  # .cfg
  # done in google drive

  # .names
  # done in google drive

  # train.txt
  # done in google drive

  # validation.txt
  # done in google drive

  # .weights
  # done in google drive




#
#
# TODO
#
#



  backup_chart_path = os.path.join(project_path, 'chart_last.png')

  event = Event()
  print(event.is_set())
  t = Thread(target=backup_chart, name='backup_chart', kwargs={'path': backup_chart_path, '
  t.start()


  # Train
  !./darknet detector train "$DATA_FILE" "$CFG_FILE" "$PROJECT_PATH"/backup/yolov4-custom_l
```

```
# backup_chart_path = os.path.join(project_path, 'chart_last.png')

# event = Event()
# print(event.is_set())
# t = Thread(target=backup_chart, name='backup_chart', kwargs={'path': backup_chart
# t.start()

# # Train
# !./darknet detector train "$DATA_FILE" "$CFG_FILE" "$PROJECT_PATH"/backup/yolov4-
```

CAUTION! Interrupt the `backup_chart()` thread

```
# Interrupt

print(f'Is Thread alive?: {t.is_alive()}')
event.set()
t.join()
print(f'Is Thread alive?: {t.is_alive()}')
print('Done.')
```

```
# # Interrupt

# print(f'Is Thread alive?: {t.is_alive()}')
# event.set()
# t.join()
# print(f'Is Thread alive?: {t.is_alive()}')
# print('Done.')
```

Patryk Laskowski

GitHub