UPEC
UNIVERSITÉ PARIS-EST CRÉTEIL

Master 2 :

Systèmes Distribués et technologies de la Data Science

DEEP LEARNING

Project Report:

Prepared by :

Attrassi Alaeddine

Suprvised by :

Nadiya Shvai

Amir Nakib

a.u : 2020/2021

I- Introduction

Artificial intelligence (AI) is a process of imitating human intelligence that relies on the creation and application of algorithms executed in a dynamic computing environment. Its purpose is to enable computers to think and act like human beings.

To achieve this, three components are needed:

- Computer systems
- Data with management systems
- Advanced AI algorithms (code)

To get as close as possible to human behavior, artificial intelligence needs a high amount of data and high processing capacity.

The goal of the project is to create a code in Google Colab doped with artificial intelligence (AI) that can detect faces with or without a mask, bikes, scooters..

In this project we will we will answer the following questions :

- What is a Draknet ?
- What is Yolo and what the advantages of YOLO ?
- What are the different steps of project realization ?
- Model training with yolov4
- Test the model
- Submit the result to kaggle

II- What is a Darknet

Darknet is an open source neural network framework written in C and CUDA, It is fast, easy to install, and supports CPU and GPU computation. Darknet is mainly for Object Detection, and have different architecture, features than other deep learning frameworks. It is faster than many other NN architectures and approaches

Here you can some find project using darknet framework:

Darknet: Open Source Neural Networks in C (pjreddie.com)

)

III- What is Yolo

Object detection is one of the classical problems in computer vision where you work to recognize *what* and *where* — specifically what objects are inside a given image and also where they are in the image. The problem of object detection is more complex than classification, which also can recognize objects but doesn't indicate where the object is located in the image. In addition, classification doesn't work on images containing more than one object.

YOLO is popular because it achieves high accuracy while also being able to run in real-time. The algorithm "only looks once" at the image in the sense that it requires only one forward propagation pass through the neural network to make predictions. After non-max suppression (which makes sure the object detection algorithm only detects each object once), it then outputs recognized objects together with the bounding boxes.

The advantages of YOLO :

With YOLO, a single CNN simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This model has a number of benefits over other object detection methods:

YOLO is extremely fast

YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance.

YOLO learns generalizable representations of objects so that when trained on natural images and tested on artwork, the algorithm outperforms other top detection methods.

Note :

Darknet architecture & YOLO is a specialized framework, and they are on top of their game in speed and accuracy. YOLO can run on CPU but you get 500 times more speed on GPU as it leverages CUDA and cuDNN.

IV- Different steps of project realization

We have followed these steps to make this project work, make sure that you have available GPU, and preinstalled Darknet framework, and openCV on your environnement.

We have used Google Colab to take advantage of the free GPU that they offer, here you can check how can we interact with this work environment

Bienvenue dans Colaboratory - Colaboratory (google.com)

1- Step 1 : Requirements :

As we know deep learning is an approch used to help 'machine learning', or we can define it as a methodology of machine learning based on neural networks.

we can have deep learning in many fields as Healthcare, computer vision, machine vision, speech recognition, natural language processing, audio recognition, social network filtering, ect … .

The core of the project is based on developing a system that can recognize object and count them, there existence in an a image, as a bike, scooter, pedestrian, the presence of mask or not, to do so, we have collected some data that refers to our needs as a man wearnig a mask, a biker, ect…

Than we have procced to the fragmentation of videos into images using openCV.

After that we have labeled the images, that mean we have created for each object exciting in the image a bounding box that contains, we have 6 classes (Bike, Scooter, pedestrian, mask, pas_de_masque, mal_mis). Then we got the labeling results in (.csv) using VGG annotator.

Example of a labeled image

1- Step 2 : Data cleaning and preparation

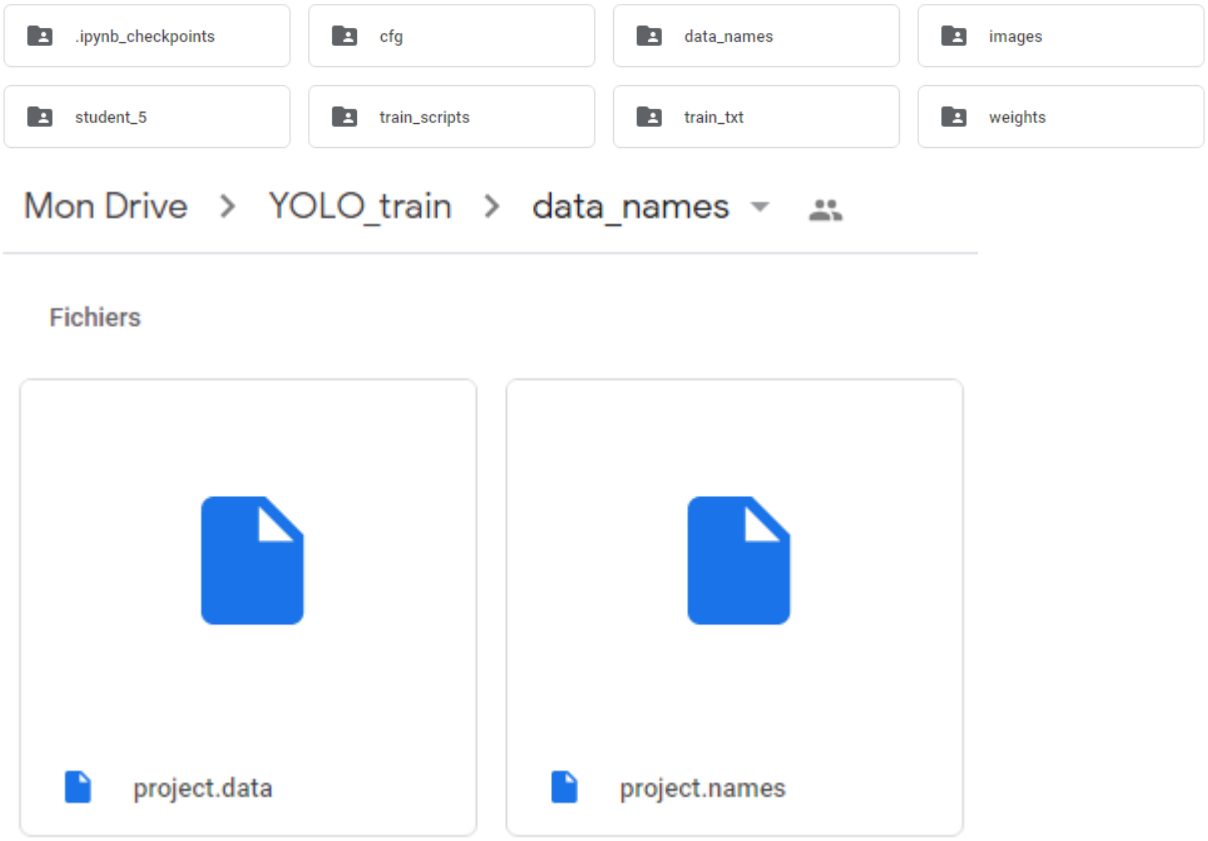
```
vgg_labels_path0 = "/content/drive/My Drive/yolo_5_9_13_30/student_5.csv"
vgg_labels_path1 = "/content/drive/My Drive/yolo_5_9_13_30/student_9.csv"
vgg_labels_path2 = "/content/drive/My Drive/yolo_5_9_13_30/student_13.csv"
vgg_labels_path3 = "/content/drive/My Drive/yolo_5_9_13_30/student_30.csv"
```

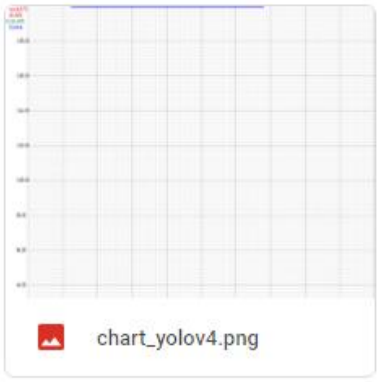
Labels of student (5, 9, 13, 30)

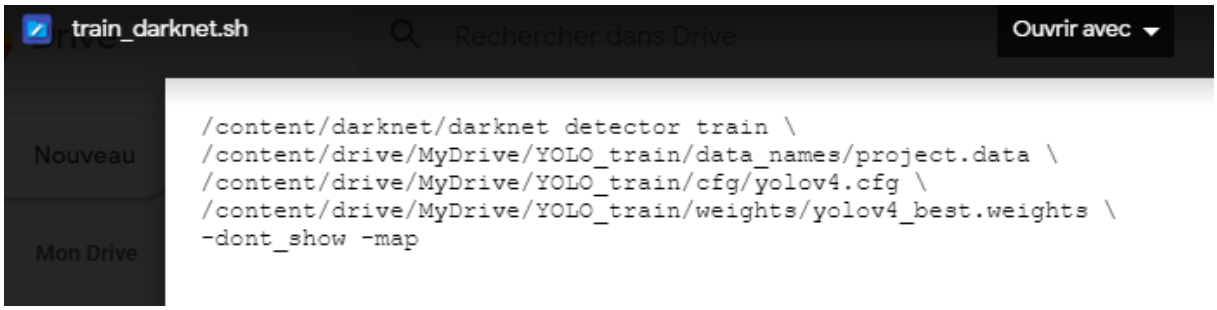
Above we define the path for the data, in this step, we worked on 776 images. we cleaned up the data by deleting the data bad labeled, and keeping only the well-labeled images. Then we save the coordinates of the bounding boxes of the images in files (.txt). Finally, we split the data taking 75% of the images for training and the remaining 15% for validation.

2- Algorithm

Before we start we should configure the envirenement so we need to do some changes in the makefile we start by activating the GPU and more, we configure the .cfg yolov4 we a have made some change like number of batchs number of class, each cnvulutional layer before the yolo layer we change to (number of class + 5) * 3 also we need to but the data names and the data, data names refer to the mapping class







We're going to define some functions that helps us to create labels and to process in data cleaning

```
[ ]  1 # here we define the path for the file that contains the informations about class and bounding boxes
     2 vgg_labels_path = "/content/drive/MyDrive/YOLO_train/student_5/student_5.csv"
```

```
[ ]  1 # we load the the data in dataframe
     2 df_labels = pd.read_csv(vgg_labels_path, sep=",")
```

```
[ ]  1 unique_filenames = df_labels.filename.unique()
```

```
[ ]  1 len(unique_filenames)
```

```
194
```

```python
# this function contains the value (x,y,width,height) for each bounding boxes
def read_region_shape_attributes(attributes):
    if isinstance(attributes, str):
        attributes = eval(attributes)

    xmin = attributes["x"]
    ymin = attributes["y"]

    width = attributes["width"]
    height = attributes["height"]

    xcenter = xmin + width/2
    ycenter = ymin + height/2

    return xcenter, ycenter, width, height
```

```python
# this function help us to get the heights and weights of an given image
def get_image_width_height(image_path):
    image = cv2.imread(image_path)
    image_shape = image.shape
    image_height, image_width = image_shape[0], image_shape[1]

    return image_width, image_height
```

```python
# here we gonna read the class attriibuted to each bounding boxes
def read_region_attributes(attributes):
    try:
        if isinstance(attributes, str):
            attributes = eval(attributes)

        class_name = attributes["class"]
# if there is no class_name we gonna attribute a -1, that we should clean after,
# the main goal is all the bounding boxes should have a class
    except:
        class_name = -1

    return class_name
```

Check existing classes

```
[ ]  1 df_labels["class_name"] = df_labels.region_attributes.apply(read_region_attributes)
```

```
[ ]  1 df_labels.class_name.unique()
```

```
array(['piéton', 'masque', 'pas_de_masque', -1, 'mal_mis', 'velo'],
      dtype=object)
```

```
[ ]  1 # here we select all the ligne that has (-1) in classe name
     2 df_labels[df_labels.class_name == -1]
```

```
[ ]  1 # here we regroup all bad filenames, the class that has -1 in calsss_name in a unique dataframe
     2 bad_filenames = df_labels[df_labels.class_name == -1].filename.unique()
```

```
[ ]  1 # and now we're gonig to clean the data, we taking all the data except the ones that has -1 in class_name
     2 df_labels_clean = df_labels[~df_labels.filename.isin(bad_filenames)]
```

```
[ ]  1 # and as we see that the data has been cleaned
     2 len(df_labels_clean.filename.unique())
```

```
176
```

```
[ ]  1 # here we define the diffrent class that we have in our project, diffrent names but same id to make it unique
     2 class_mapping = {
     3     "velo":0,
     4     'vÃ©lo':0,
     5     "piéton":1,
     6     "piÃ©ton":1,
     7     "pieton":1,
     8     "trottinette":2,
     9     "masque":3,
    10     "mal_mis":4,
    11     "pas_de_masque":5,
    12     "pas_de_masque ":5
    13 }
```

Prepare labels

```
[ ]  1 from shutil import copy
     2 import os
```

```
[ ]  1 all_filenames = df_labels_clean.filename.unique()
```

```
[ ]  1 source_image_folder = "/content/drive/MyDrive/YOLO_train/student_5"
```

```
[ ]  1 # define the ouptub folder where we going to store the
     2 output_folder = "/content/drive/MyDrive/YOLO_train/images"
```

Prepare labels

```
[ ]  1 from shutil import copy
     2 import os
```

```
[ ]  1 all_filenames = df_labels_clean.filename.unique()
```

```
[ ]  1 source_image_folder = "/content/drive/MyDrive/YOLO_train/student_5"
```

```
[ ]  1 # define the ouptub folder where we going to store the
     2 output_folder = "/content/drive/MyDrive/YOLO_train/images"
```

```
1 for filename in all_filenames:
2     df_slice = df_labels_clean[df_labels_clean.filename == filename]
3     img_path = os.path.join(source_image_folder, filename)
4
5     copy(img_path, output_folder)
6
7     img_width, img_height = get_image_width_height(img_path)
8
9     label_name = '.'.join(filename.split('.')[:-1]) + ".txt"
10    label_path = os.path.join(output_folder, label_name)
11
12    label_str = ""
13    if not df_slice.iloc[0].region_count == 0:
14        for _, row in df_slice.iterrows():
15            class_name = row["class_name"]
16            class_id = class_mapping[class_name]
17
18            xcenter, ycenter, width, height = read_region_shape_attributes(row["region_shape_attributes"])
19
```

```
xcenter, ycenter, width, height = read_region_shape_attributes(row["region_shape_attributes"])

xcenter_norm = float(xcenter) / img_width
ycenter_norm = float(ycenter) / img_height

width_norm = float(width) / img_width
height_norm = float(height) / img_height

bbox_line = "{} {:.3f} {:.3f} {:.3f} {:.3f}".format(class_id, xcenter_norm, ycenter_norm, width_norm, height_norm)
label_str += bbox_line
label_str += "\n"
```

```
30  open(label_path, "w") as output:
31 output.write(label_str)
```

Split files into train and val

```
from sklearn.model_selection import train_test_split
import os
```

```
# here we define the path that contains all the data
img_folder = "/content/drive/MyDrive/YOLO_train/images"
```

```
# with this line below we choose all the file that are in the path
# and don't have an extension (.txt) and we joining to the path
img_path_list = [os.path.join(img_folder, img_name) for img_name in os.listdir(img_folder) \
                 if not img_name.endswith(".txt") ]
```

```
len(img_path_list)
```

```
len(img_path_list)
```

```
176
```

```
# here we split the data into train data and test data
img_path_list_train, img_path_list_test = train_test_split(img_path_list, test_size=0.15, random_state=42)
```

```
# we define two paths one for test and one for train
output_train_txt_path = "/content/drive/MyDrive/YOLO_train/train_txt/train_20201001.txt"
output_test_txt_path = "/content/drive/MyDrive/YOLO_train/train_txt/test_20201001.txt"
```

```
# here we're going to store the file and his path in txt file so at the
# ouput we will have the path of each data (image for the train
with open(output_train_txt_path, "w") as output:
    output.write("\n".join(img_path_list_train))

with open(output_test_txt_path, "w") as output:
    output.write("\n".join(img_path_list_test))
```

```
# téléchargement de framework darknet
!git clone https://github.com/AlexeyAB/darknet
```

```
Cloning into 'darknet'...
remote: Enumerating objects: 14691, done.
remote: Total 14691 (delta 0), reused 0 (delta 0), pack-reused 14691
Receiving objects: 100% (14691/14691), 13.26 MiB | 24.46 MiB/s, done.
Resolving deltas: 100% (9995/9995), done.
```

```
# change makefile to have GPU and OPENCV enabled
%cd 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
```

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

```
[ ]  1 # compilation de framework (envirennoment darknet)
     2 !make
```

```
1 #téléchargement des poinds (weights) de départ necessaire à une premiére phase de l'entrainement
2 !wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.weights
```

```
1 #to train the model
2 !cd /content/drive/MyDrive/YOLO_train/train_scripts && bash train_darknet.sh
```

```
# script helper to generate the CSV file for kaggle
import cv2
import numpy as np
from os import listdir
from os.path import isfile, join
import pandas as pd
weights = "/content/drive/MyDrive/yolo_5_9_13_30/weights/yolov4_best.we
ights"
config_file = "/content/drive/MyDrive/yolo_5_9_13_30/cfg/yolov4.cfg"
dataset = "/content/drive/MyDrive/yolo_5_9_13_30/images_run"
# read pre-trained model and config file
net = cv2.dnn.readNetFromDarknet(config_file, weights)
net.setPreferableBackend(cv2.dnn.DNN_BACKEND_OPENCV)
# function to get the output layer names in the architecture
def get_output_layers(net):
    return [net.getLayerNames()[i[0] - 1] for i in net.getUnconnectedOu
tLayers()]
def run(frame):
```

```
    # create input blob
    blob = cv2.dnn.blobFromImage(frame, 0.00392, (512, 512), True, crop
=False)
    # set input blob for the network
    net.setInput(blob)
    # run inference through the network and gather predictions from out
put layers
    outs = net.forward(get_output_layers(net))
    # for each detetion from each output layer get the confidence, clas
s id,
    # bounding box params and ignore weak detections (confidence < 0.25
)
    count_list = [0,0,0,0,0,0] # [velo, pieton, trottinette, masque, ma
l_mis, pas_de_masque]
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > 0.25:
                count_list [class_id] += 1
    return count_list
num_img = [f for f in listdir(dataset) if isfile(join(dataset, f))]
R_Project_1 = {"filename": [], "pieton": [], "masque": [], "mal_mis": [
]}
R_Project_2 = {"filename": [], "trottinette": [], "velo": []}
R_Project_3 = {"filename": [], "pieton": [], "velo": []}
for i in num_img:
    img = cv2.imread(dataset + "/" + i)
    result = run(img)
    R_Project_1["filename"].append(i)
    R_Project_1["pieton"].append(result[1])
    R_Project_1["masque"].append(result[3])
    R_Project_1["mal_mis"].append(result[4])
    R_Project_2["filename"].append(i)
    R_Project_2["velo"].append(result[0])
    R_Project_2["trottinette"].append(result[2])
    R_Project_3["filename"].append(i)
    R_Project_3["velo"].append(result[0])
    R_Project_3["pieton"].append(result[1])
pd.DataFrame(R_Project_1).to_csv("/content/Project_1.csv", index=False)
pd.DataFrame(R_Project_2).to_csv("/content/Project_2.csv", index=False)
pd.DataFrame(R_Project_3).to_csv("/content/Project_3.csv", index=False)
```

Conclusion :

In this project, we were able to get a taste of the capabilities of YOLOv4 for custom objects. To create our own object alarm we covered the following steps:

- Create a personalized dataset from our own videos and we have labeled it.
- We did the data cleaninig and data preparation in the format indicated for training the model.Configuration de notre environnement GPU sur Google Colab.
- Installation of the Darknet YOLO v4 training environment.
- Download our custom dataset and configure directories for YOLO v4.
- Configure the custom YOLO v4 training configuration file to achieve optimal results with limited computational capacity.
- Train our custom YOLO v4 object detector from a pre-trained model.

We used the weights trained on our data and made inferences about test images never seen by the model. To conclude we can say that pre-trained networks are very powerful and very often more efficient than what we can create ourselves in a reasonable amount of time.

The transfer learning approach requires less data and fewer eras (less training time), so it's a win-win situation. To be more precise, transfer learning requires more training time per epoch, but requires fewer epochs to train a usable model..