Capstone Project
License Plate Detection

Team Members:
Fouad Majd Alkadri
Hadi Abou Daya

1. Introduction

In this project, we'll be developing a web-based application that will be applying Computer Vision and use one of the applications of Computer Vision which is object detection by detecting the plate number for cars. These detected objects will be using the trained model and that will be developed and deployed throughout the project. In addition, it will be able to use optical character recognition to extract the characters and compare the detected license plate numbers of cars and the database that will be used for registering or adding information related to the license plate number, so for instance, determining if the given vehicle has access to the place it's entering, or maybe some other tasks. Finally, this project could be extendable by adding more features to the database that will be adding more values to the project and that would make the model able to do more tasks that help the user of the web-based application.

2. Proposed Approach and Tools

The Approach of this project is that we do an Automatic license plate number recognition project. And in this project has the following features:

- Detecting the license plate number of cars.
- Applying Analysis to the detected license plate numbers if the license plate number either expired or not.
- Applying Analysis to Know the type of the car by detecting the license plate number.
- Applying Analysis if the detected license plate number of the car is already registered to a specific company or the compound building.
- Applying Analysis to the detected license plate number and knowing the person who's driving the car.

2.1. Tools

The tool that will be used are as follows:

- **Google Colab:** is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser and is especially well suited to machine learning, data analysis, and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs.
- **GitHub:** is an Internet hosting service for software development and version control using Git.
- **Yolo7:** it's a tool that will help us for real-time detection of an object.
- **DeepSORT:** A tracking algorithm that allows the model to know that the object it is seeing in the current frame is the same as the last one.
- **DataSpell:** is an IDE for data science and Machine learning with intelligent Jupyter notebooks, interactive Python scripts, and lots of other built-in tools.
- **Wandb:** tool used to track, compare, and visualize machine learning model.

3. The Dataset available

For our project, we are going to use an external existing license plate number of images dataset for cars to shorten our development time and increase the accuracy of our project, most of our datasets will be taken from the **Roboflow** website.

4. Dataset Specification

The Dataset: https://universe.roboflow.com/augmented-startups/vehicle-registration-plates-trudk/dataset/2.

Dataset Description: The following dataset contains the images of cars and the coordinates of the license plate number within the images or the labeled license plate number of the car.

The shape of the Dataset:

- **Rows:** 21959
- **Columns:** 8

Features:

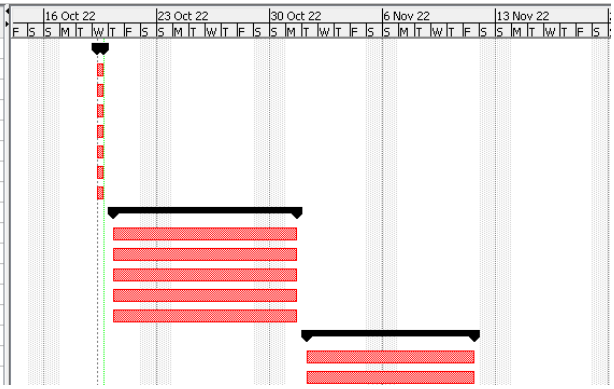
- **(Filename)** file path that contains many files that contain the images.
- **(width)** the width of the image frame.
- **(height)** the height of the image frame.
- **(xmin)** x-axis start-point coordinate in the image frame of the license plate number from the image.
- **(xmax)** x-axis endpoint coordinate in the image frame of the license plate number from the image.
- **(ymin)** y-axis start-point coordinate in the image frame of the license plate number from the image.
- **(ymax)** y-axis endpoint coordinate in the image frame of the license plate number from the image.

Dataset benefit:

- It contains huge amounts of images, so that would lead to high accuracy and promising results.
- The dataset is already labeled, so would save us a lot of time.

5. Work Plan

Name	Duration	Start	Finish
Proposal	1 day?	**10/19/22 8:00 AM**	**10/19/22 5:00 PM**
Introduction	1 day?	10/19/22 8:00 AM	10/19/22 5:00 PM
Objectives	1 day?	10/19/22 8:00 AM	10/19/22 5:00 PM
The Dataset avaliable	1 day?	10/19/22 8:00 AM	10/19/22 5:00 PM
Dataset Specification	1 day?	10/19/22 8:00 AM	10/19/22 5:00 PM
Tools to be Used	1 day?	10/19/22 8:00 AM	10/19/22 5:00 PM
Work Plan	1 day?	10/19/22 8:00 AM	10/19/22 5:00 PM
References	1 day?	10/19/22 8:00 AM	10/19/22 5:00 PM
Progress Report	8 days?	**10/20/22 8:00 AM**	**10/31/22 5:00 PM**
Problem definition	8 days?	10/20/22 8:00 AM	10/31/22 5:00 PM
Model used	8 days?	10/20/22 8:00 AM	10/31/22 5:00 PM
EDA	8 days?	10/20/22 8:00 AM	10/31/22 5:00 PM
Model Matrices	8 days?	10/20/22 8:00 AM	10/31/22 5:00 PM
Evaluating the Model	8 days?	10/20/22 8:00 AM	10/31/22 5:00 PM
Final Report	9 days?	**11/1/22 8:00 AM**	**11/11/22 5:00 PM**
Presentation	9 days?	11/1/22 8:00 AM	11/11/22 5:00 PM
Report	9 days?	11/1/22 8:00 AM	11/11/22 5:00 PM



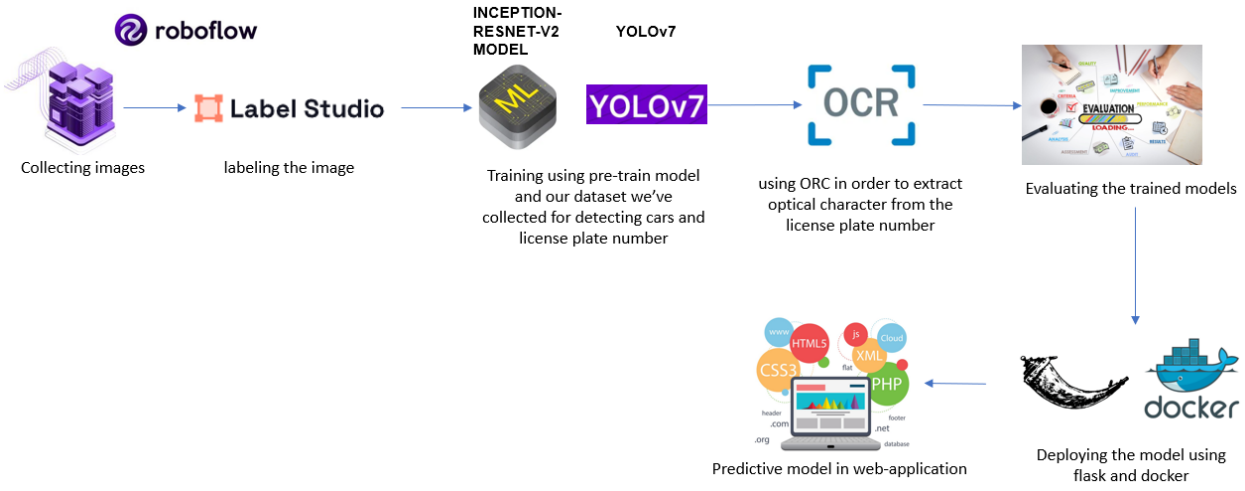
6. The Problem

The problem that we are trying to solve is that we want to apply multiple trained models that will be detecting multiple objects in real-time, and these are the following:

- Detecting multiple car objects.
- Detecting the license plate number of the car.
- Detecting the optical character to extract the characters from the license plate number.

These detection models require a lot of datasets for training, so in that case, it would take a lot of time and effort for collecting a dataset and label each one of them. And These results that we've to find a solution to shorten our time consumption for training multiple models, so in that case, we're going to use pre-trained models to detect the cars and the optical characters of the license plate number while detecting license plate number of the car, we're going to use the provided dataset as mentioned above and train it.

7. Project Architecture



8. Applied Tool & Models

The following tools and models that we're going to use as follows:

- **Google colab** for visualization purposes.
- **DataSpell** to train the model on the GPU offline without getting interrupted or disconnected while training the model.
- **Yolov7** will be used to train the datasets for detecting multiple classes which are the following:
  - Cars
  - license plate number
  - Optical character recognition.
- **GitHub** will be used for gathering useful code resources.
- **DeepSort** is a tracking algorithm that will be used for enhancing and improving the frame rate of real-time detection after training the model in **Yolov7.**
- **Roboflow** will be used to import the datasets that will be already labeled and Split into train and testing datasets.
- **Wandb:** tool used to track, compare, and visualize machine learning model.
- **OCR Model** will be used for extracting the characters of the license plate number using optical characters recognition from the images. And has multiple pre-trained models as follows:
  - **Paddle OCR model** we will be using this model**.**
  - **Keras OCR model**
  - Tesseract model.
  - Easy OCR model.

9. Tool & Model Comparison

In this section, we'll be comparing different models and tools and we will explain why we would choose that specific tool and model based on the related projects of other people who had experience with these tools and models.

Tool comparison 1:

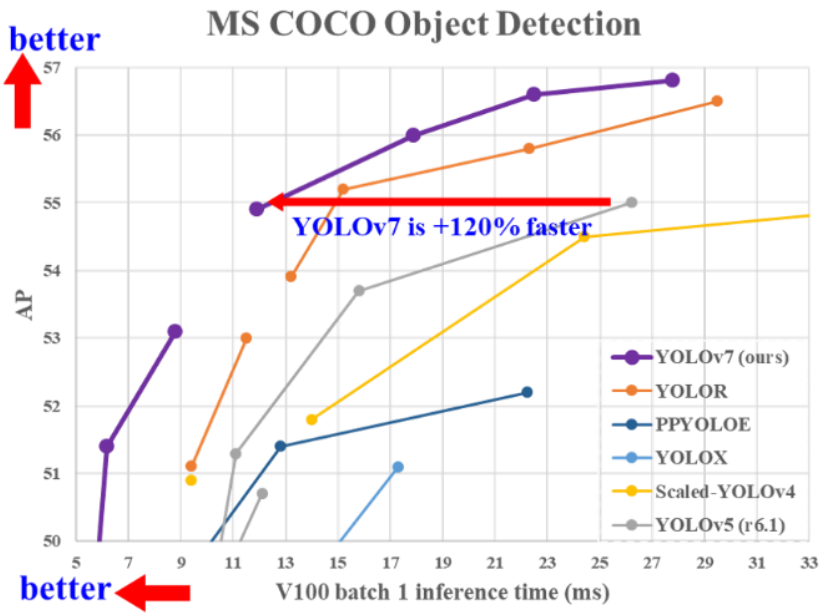
These two tools will be used since **Google colab** cannot handle the training process because of high GPU memory usage and it requires **12GB** to train on **YOLOv7** and it might crash during the training process, so to solve this problem we will be using an offline tool that works for training and doing the other visualization part on the **Google colab**.

Model Comparison 2:

The following model could be applied for real-time object detection to our project are the following:

- YOLO (2016).
- SSD (2016).
- RetinaNet (2017).
- YOLOv3 (2018).
- YOLOv4 (2020).
- YOLOR (2021).
- **YOLOv7 (2022).**

The model that we're going to use is **YOLOv7** which is the most recent version and algorithm used for real-time object detection model that has been established until now. And based on the performance that has been tested compared to other different versions of **YOLO's** it's performing 120% faster than other versions as shown in the figures below.



According to the resources, the mean average precision (**MAP**) of **RetinaNet** reached 82.89%, but the frames per second (**FPS**) is only one-third of **YOLOv3**, which makes it difficult to achieve real-time performance. **SSD** does not perform as well on the indicators of MAP and FPS. Although the **MAP** of **YOLOv3** is slightly lower than the others (80.69%). And these results that the **YOLOv3** is having better performance than **SSD** and **RetinaNet** algorithms. Finally, **YOLOv7** is having the leading accuracy with higher **FPS** and real-time performance compared to other versions of **YOLO's**.

Model Comparison 3:

The following model could be applied to the optical character recognition of our project are the following:

- Tesseract
- Keras-OCR
- EasyOCR
- **PaddleOCR**

The model that we're going to use is the **Paddle-OCR model** since it works both for detecting the optical character in either high-quality or low-quality images while other models for instance **Easy-OCR** it does the same as the **Keras-ORC model** in either high-quality or low-quality images, but with lower accuracy. Finally, the last model is the **Tesseract model** which can only extract optical characters in high-quality images while in low-quality images it cannot extract optical characters, but it can achieve very high accuracy as shown in below the 2 figures.

	Actual Value	Tesseract Prediction	Keras-OCR Prediction	EasyOCR Prediction
Number Plate High Quality	HR26DK8337	HR260K8337	HR26DK8337	HRZ6DK8337
Number Plate Low Quality	MH14GN9239	Spaces	MHL4GH9239	9239
Handwritten Low Quality	AMIT ASHISH	Spaces	ADIT ASHISH	AdIT ASHISH
Handwritten High Quality	LAKSHMINIVAS TOURIST HOME	LAKSHMINIVAS TOURIST HOME	LAKSHMINIVAS TOURIST HOM	LAKSHMINIVAS TOURIST HOME
Image with text High Quality	Albert Einstein	Albert Einstein	Albert Einstein	A|ber t Einsteim
Image with text Low Quality	Kotak Mahindra Bank	Kotak Mahindra Bank	Kotak Mahindra Bank	Kotak Mahindra Bank
Reciept High Quality	Order #19866	Order #19866	Order #119666	Order #19866
Reciept Low Quality	Amoxicillin 500mg	Spaces	Amoxicillin 500mg	Amoxicillin 500mg

	Tesseract	PaddleOCR	PaddleOCR (Puntucation fixed)	PaddleOCR (White-spaces fixed)
Number of Errors	**95**	387	274	164
Accuracy	**0.98**	0.92	0.96	0.97

	Tesseract	PaddleOCR CPU	PaddleOCR GPU
Speed	3.83 s	4.12 s	**2.07 s**
English Model Size	23 MB	**2 MB**	**2 MB**

And based on the resources the model **Paddle-OCR model** performs 46% in a fast manner in 2.07 seconds while the **Tesseract** model spends 3.83 seconds. And in addition to that, the difference in the size of the English model is very high in the **Tesseract model** compared to the **Paddle-OCR model.**

10. Experiment

10.1 Visualization:

In this section, we'll visualize the dataset that we have and apply some analysis on the dataset as follows:

Google Colab Notebook Link:

1. **Getting details about our dataset and knowing the number of rows and columns of our dataset.**

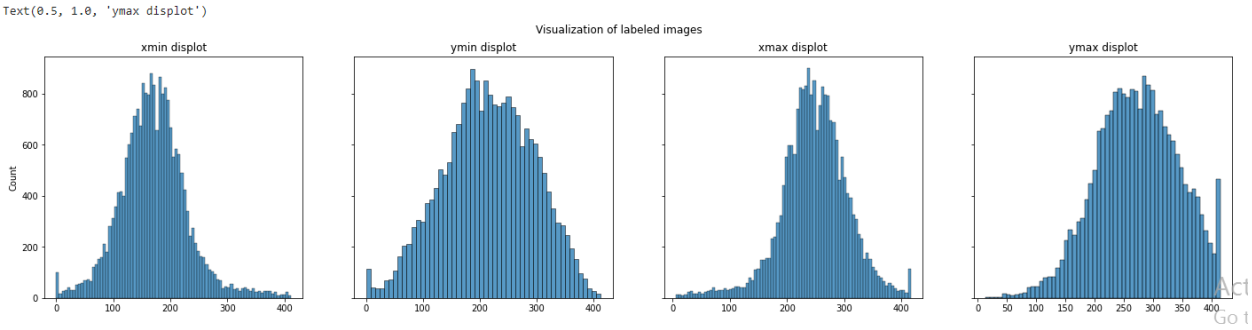
```
[7]  1 df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 21959 entries, 0 to 1839
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   filename  21941 non-null  object
 1   width     21941 non-null  float64
 2   height    21941 non-null  float64
 3   class     21941 non-null  object
 4   xmin      21941 non-null  float64
 5   ymin      21941 non-null  float64
 6   xmax      21941 non-null  float64
 7   ymax      21941 non-null  float64
dtypes: float64(6), object(2)
memory usage: 1.5+ MB
```

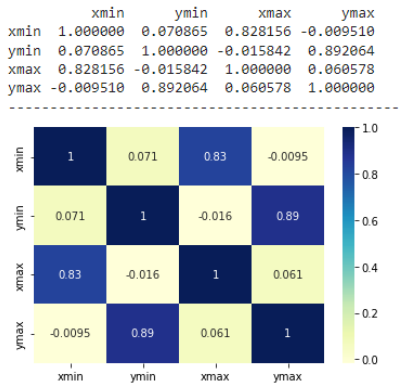
```
[8]  1 print("- The Following Dataset contains:",df.shape[0],"Images with",df.shape[1],"Features")

- The Following Dataset contains: 21959 Images with 8 Features
```

2. **Visualizing the distribution of data of the bounding box of plate numbers values that are (xmin, xmax, ymin, ymax) or labeled images as shown below:**



3. **Visualizing the correlation of data of the bounding box of plate numbers values that are (xmin, xmax, ymin, ymax) or labeled images as shown below:**

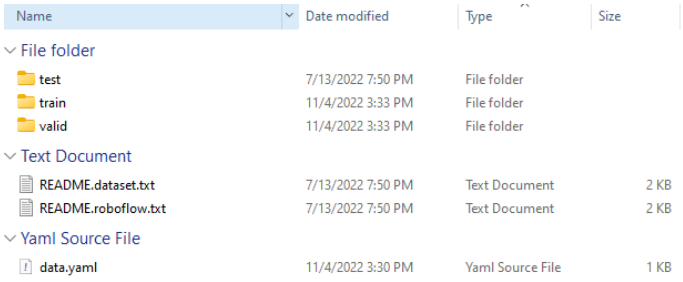


10.2 Training

In this section, we'll show the complete step for training our model on our dataset to achieve one of the approaches by detecting the license plate number in real time.
Script Code: [Google Colab Notebook Link:]

The following step for training our dataset as follows:

1. **The first step is that we convert our dataset format into YOLO format to be able to train our dataset.**



2. **Making sure that we are using an appropriate GPU since it requires a high Graphics memory card of approximately more than 12GB memory card for training the YOLOv7 model since it's a heavy model, so we have to make sure that we have selected an appropriate GPU card and make sure that it works fine as shown below in the figure**

```
!nvidia-smi
✓ 0.2s

Fri Nov  4 23:51:25 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 526.47       Driver Version: 526.47       CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name            TCC/WDDM | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ... WDDM  | 00000000:01:00.0  On |                  N/A |
| N/A   66C    P3    27W /  N/A |   2371MiB /  6144MiB |     34%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A      8224    C+G   ...ser\Application\brave.exe    N/A      |
|    0   N/A  N/A     13940      C   ...les\LGHUB\lghub_agent.exe    N/A      |
|    0   N/A  N/A     16404      C   ...thon\Python310\python.exe    N/A      |
|    0   N/A  N/A     21656    C+G   ...root\Office16\WINWORD.EXE    N/A      |
+-----------------------------------------------------------------------------+
```

```
torch.cuda.is_available()
✓ 0.1s

True
```

```
torch.cuda.device_count(), torch.cuda.current_device()
✓ 0.1s

(1, 0)
```

```
torch.cuda.get_device_name(0)
✓ 0.9s

'NVIDIA GeForce RTX 2060'
```

```
torch.backends.cudnn.enabled
✓ 0.5s

True
```

```
torch.backends.cudnn.version()
✓ 0.1s

8302
```

3. **Download the [YOLOv7] model from GitHub and its dependencies.**

Model	Test Size	AP^{test}	$AP_{50}{}^{test}$	$AP_{75}{}^{test}$	batch 1 fps	batch 32 average time
YOLOv7	640	51.4%	69.7%	55.9%	161 *fps*	2.8 *ms*
YOLOv7-X	640	53.1%	71.2%	57.8%	114 *fps*	4.3 *ms*
YOLOv7-W6	1280	54.9%	72.6%	60.1%	84 *fps*	7.6 *ms*
YOLOv7-E6	1280	56.0%	73.5%	61.2%	56 *fps*	12.3 *ms*
YOLOv7-D6	1280	56.6%	74.0%	61.8%	44 *fps*	15.0 *ms*
YOLOv7-E6E	1280	56.8%	74.4%	62.1%	36 *fps*	18.7 *ms*

```
# Download YOLOv7 repository and install requirements
%cd /Users/fouad/Work/Courses/ZAKA_ML_Track/Capstone_Project/LP
!git clone https://github.com/augmentedstartups/yolov7.git
%cd yolov7
!pip install -r requirements.txt
✓
```

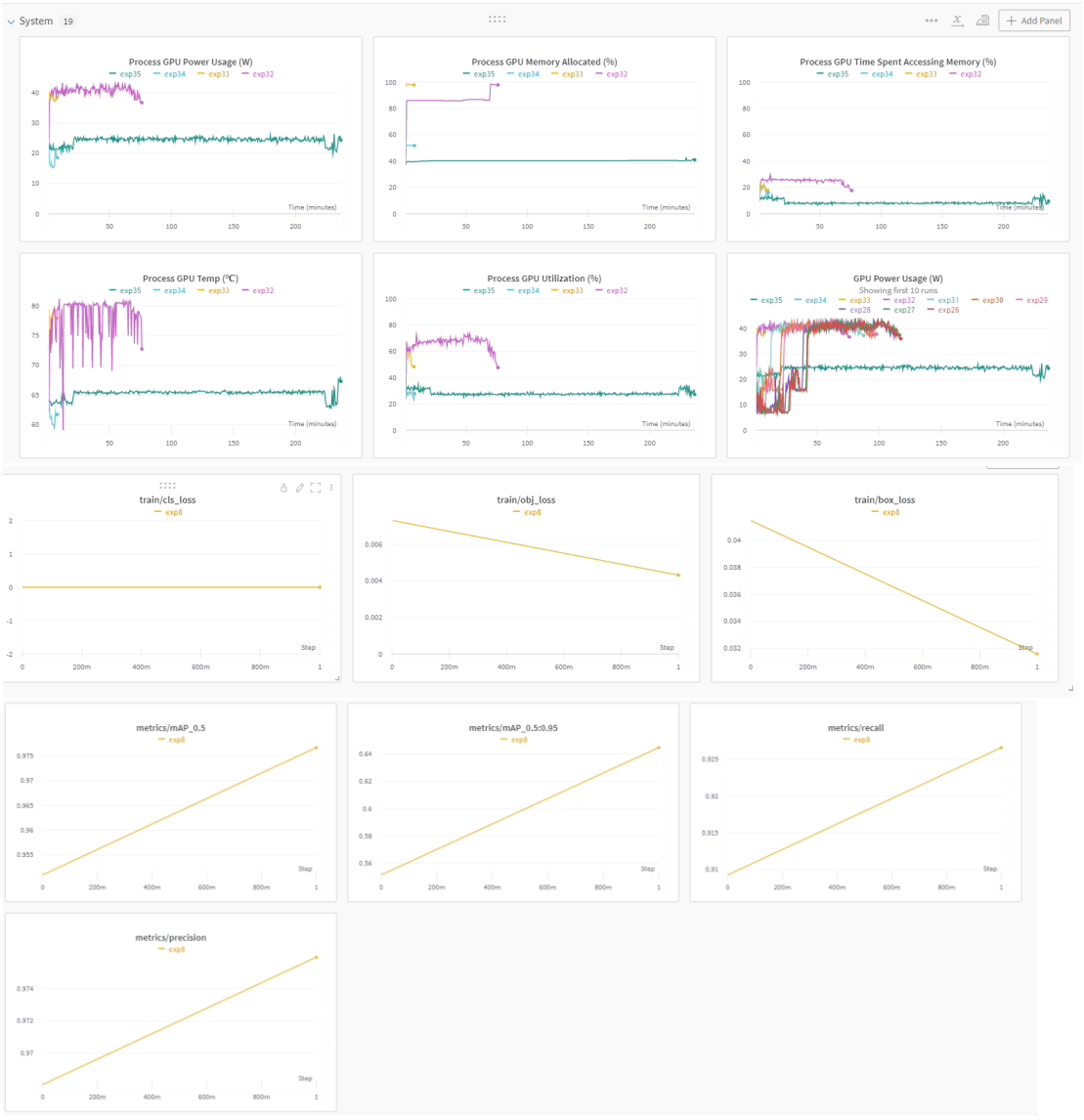
4. **After that, we Download the yolov7x.pt model that will be used for training our model as shown below in the figure:**

```
%%bash
wget https://github.com/WongKinYiu/yolov7/releases/download/v0.1/yolov7x.pt
```

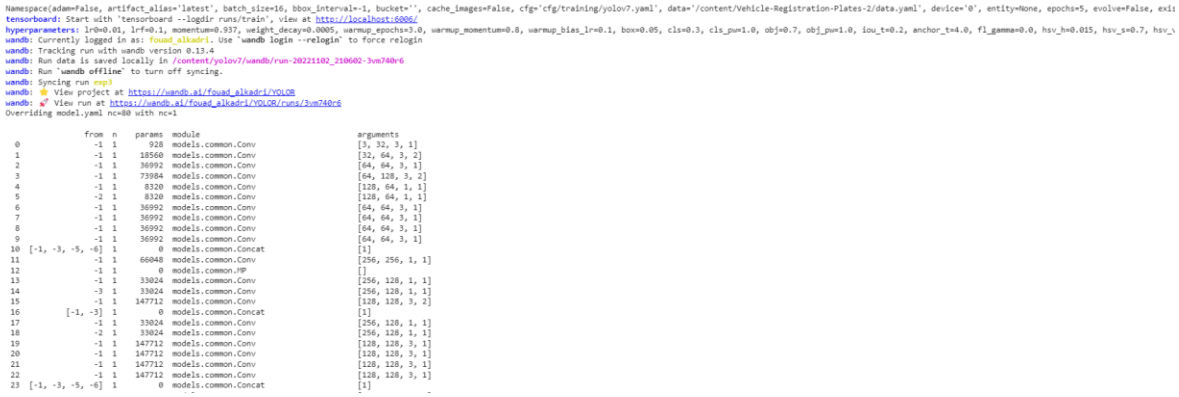
5. **Use the Wandb tool for visualizing how the training process is going on during the training based on the given dataset we are training on it and ensuring that our model is training in the right path. After that, we start training the YOLO7 model by defining the batch size, the weight of the specified model that we have downloaded (yolov7x.pt), and defining the GPU we're using for training the model as shown below in the figure:**

```
!python train.py --batch 8 --cfg cfg/training/yolov7.yaml --epochs 5 --data {dataset.location}/data.yaml --weights 'yolov7x.pt' --device 0
```

6. **Using the Wandb tool.**



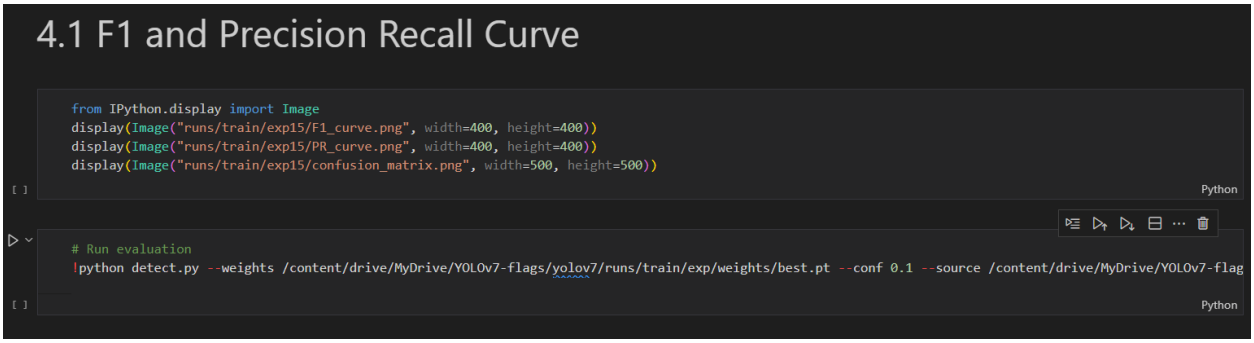
7. **Training the model.**



8. **Keep tracking the training process and monitoring the performance of our training by using the Wandb tool.**

10.3 Evaluation:

1. **Evaluating our trained model after we have finished our training process.**



10.4 Issues Related:

Throughout these processes and steps that we have done we had some related issues as follows:

- Limit Resources
  - Not enough GPU memory since we had of range (8 GB - 10 GB).
  - Sometimes it crashes because it gets over the limit of what we had, and the training got interrupted.
  - Time-consuming since the model it takes days to train the model.

11. References

Blog. Converter App Blog. (n.d.). Retrieved November 5, 2022, from https://converter.app/blog/paddleocr-engine-example-and-benchmark.

Boesch, G. (2022, September 25). *Object detection in 2022: The Definitive Guide*. viso.ai. Retrieved October 30, 2022, from https://viso.ai/deep-learning/object-detection/.

Deshwalmahesh. (n.d.). *Deshwalmahesh/yolov7-deepsort-tracking: Modular and ready to deploy code to detect and track videos using Yolo-V7 and DeepSORT*. GitHub. Retrieved October 19, 2022, from https://github.com/deshwalmahesh/yolov7-deepsort-tracking.

Google. (n.d.). Google colab. Retrieved October 19, Google. (n.d.) 2022,fromhttps://research.google.com/colaboratory/faq.html.

Sami, T. (2022, February 12). *"tesseract" vs "Keras-ocr" vs "easyocr"*. Medium. Retrieved October 30, 2022, from https://medium.com/mlearning-ai/tesseract-vs-keras-ocr-vs-easyocr-ec8500b9455b.

Startups, A. (2022, June 27). *Vehicle registration plates object detection dataset (V2, LICENSEPLATEDATASETV1) by augmented startups*. Roboflow. Retrieved October 30, 2022, from https://universe.roboflow.com/augmented-startups/vehicle-registration-plates-trudk/dataset/2.

Tan, L., Huangfu, T., Wu, L., & Chen, W. (2021, November 22). *Comparison of RetinaNet, SSD, and Yolo V3 for real-time pill identification - BMC Medical Informatics and decision making*. BioMed Central. Retrieved October 30, 2022, from https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-021-01691-8.

Tuan, A. (2022, June 20). *Tutorial: OCR with PADDLEOCR (PP-OCR)*. Medium. Retrieved October 30, 2022, from https://medium.com/@anhtuan_40207/tutorial-ocr-with-paddleocr-pp-ocr-9a4342e4d7f.

WongKinYiu. (n.d.). *Wongkinyiu/Yolov7: Implementation of paper - yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. GitHub. Retrieved October 19, 2022, from https://github.com/WongKinYiu/yolov7.