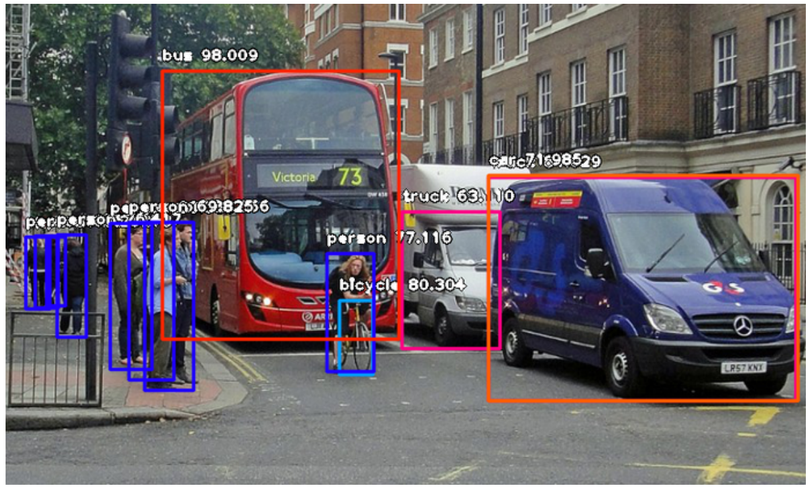
Link to the YouTube:https://www.youtube.com/watch?v=uAjLCq23A98

Object detection
Introduction

- **Problem:** Object detection is a computer vision technique to find and classify instances in images or videos. Despite significant progress in computer vision, object detection is still a complex process and comes with its own set of challenges.A lot of different images appear on the Internet every day and it is necessary to recognize objects on them for classification and easier search of the image of interest



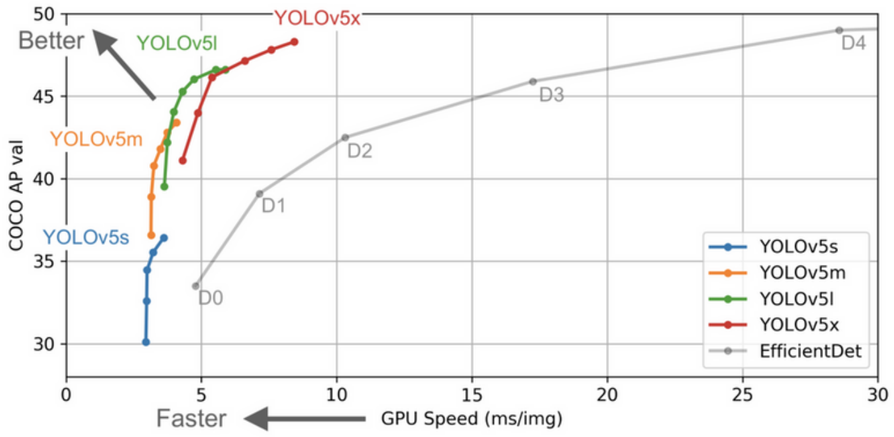
- **Literature review with links (another solutions):** In my project I used the YOLO model(You only look once), but other models can be used for this task, for example R-CNN(Region-Based Convolutional Neural Network): https://pyimagesearch.com/2020/07/13/r-cnn-object-detection-with-keras-tensorflow-and-deep-learning/ , SSD(Single Shot Detector): https://towardsdatascience.com/review-ssd-single-shot-detector-object-detection-851a94607d11 ,

- **Current work (description of the work)-**The work was done in Google colab

1. Import libraries(PyTorch-open source machine learning framework,developed by Facebook,was used because we need computer vision,numpy-to work with numbers ,os-operating system functions)
2. Connecting to Google drive
3. In Google drive, create a folder and put the images there
4. Loading the pretrained neural network model-YOLOv5, it has already been trained to detect objects of multiple classes, in this case the model is trained for 80 classes
5. Pass the path to the folder with images to the variable and output a list of paths to images
6. The folder with images has been transferred to the model
7. Using the print method, we output the size of the images and the identified objects in each image
8. Using the show method, we output images, each object is taken into the box and the probability that the object belongs to a certain class is indicated
9. Using the save method, we can get images in a separate window
10. With the help of pandas, you can output a table with more detailed information about the recognized objects on each image and output the total number of images.I have output detailed information for the eight image

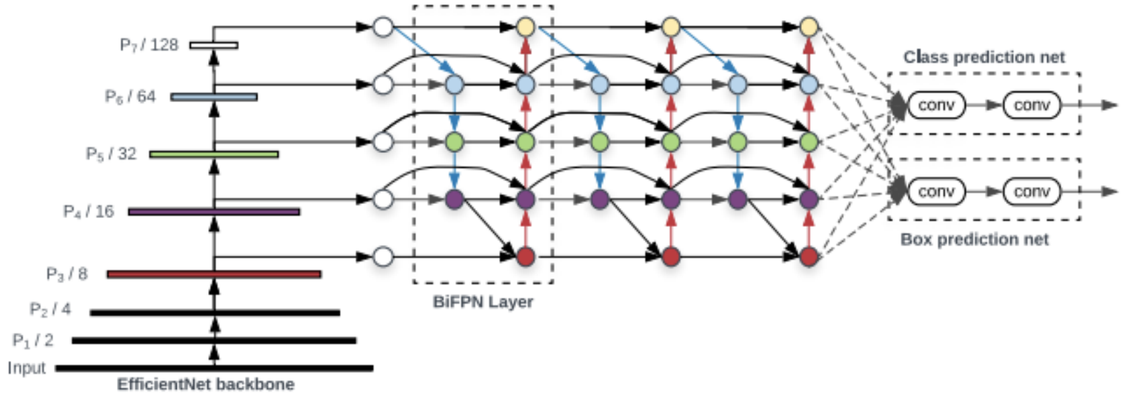
Data and Methods

- **Information about the data:** As input, I used a pack with images created on Googledrive.This is very convenient because sometimes you will need to use several folders in one project and in order not to upload data to Google colab every time, you can connect to Google drive.The images are given in various formats (png,jpeg, jfif), but since the YOLOv5 model was used, they all passed detection in approximately the same way
- **Description of the ML models you used with some theory:** YOLO(You Only Look Once) is one of the most popular computer vision algorithms today.The main task is to detect objects (classes) in the image and output a bounded rectangle representing the coordinates of the object in the image.For each object, the object detection algorithm assigns a confidence value that reflects the degree of confidence in this detection.The finished model is capable of detecting objects such as people, cars, bicycles, dogs, cats, planes, boats, etc.. YOLOv5 comes in four main versions: small (s), medium (m), large (l), and extra large (x), each offering progressively higher accuracy rates. Each variant also takes a different amount of time to train. A small version of "yolov5s" was used in my project.



- **Description of the ML models you used with some theory:**In the diagram, the goal is to create an object detector model that is very productive (Y-axis) relative to the output time (X-axis). Preliminary results show that YOLOv5 copes with this task excellently compared to other modern methods(EfficientDet). The YOLO model was the first object detector that combined the prediction procedure of bounding rectangles with class labels in an end-to-end differentiable network.

The anatomy of an object detector



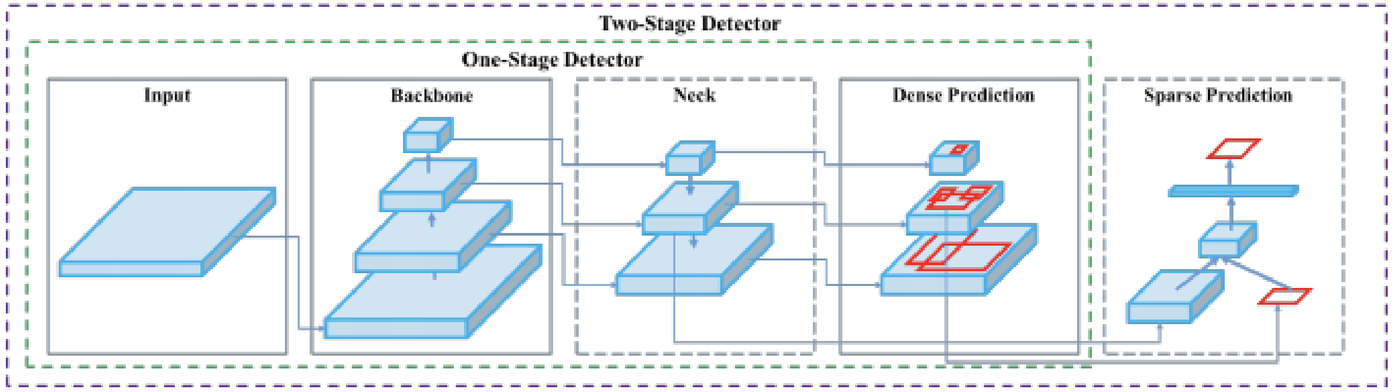
The YOLO network consists of three main pieces:

Backbone: A convolutional neural network that aggregates and forms image features at different granularities.

Neck: A series of layers to mix and combine image features to pass them forward to prediction.

Head: Consumes features from the neck and takes box and class prediction steps.

Another picture of the object detection process



Results

Detailed information about 8th image

Number of detected images: 17

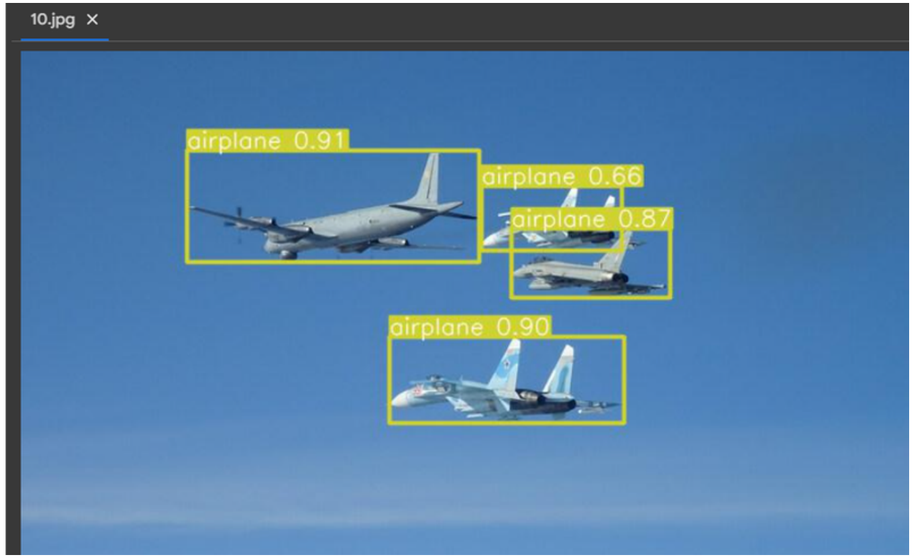
	xmin	ymin	xmax	ymax	confidence	class	name
0	1730.609009	485.357758	1819.512573	558.858643	0.852630	2	car
1	818.426514	444.862244	1059.987427	888.799988	0.843809	0	person
2	735.683899	653.422363	1167.870117	907.678772	0.832776	1	bicycle
3	0.000000	598.006042	257.066956	852.013245	0.826634	2	car
4	1804.530518	489.499054	1920.000000	577.027649	0.815207	2	car
5	1037.632812	473.939575	1094.436035	618.417297	0.804795	0	person
6	1553.923096	476.256409	1592.811035	507.040466	0.802731	2	car
7	1162.001587	491.913483	1341.015259	665.209045	0.782341	2	car
8	388.105530	530.759644	1101.890747	795.245972	0.779867	2	car
9	1449.771484	489.333435	1509.114136	537.146851	0.773699	2	car
10	1386.439819	482.691833	1424.116577	521.744751	0.751858	2	car
11	214.659180	34.243469	312.143890	307.394073	0.700036	9	traffic light
12	713.381531	319.632568	756.044556	408.786987	0.584455	9	traffic light
13	1170.776001	280.940918	1227.555542	404.373962	0.496484	9	traffic light
14	160.146790	595.133789	243.901199	787.460815	0.475392	12	parking meter
15	1275.491699	482.364899	1351.940918	581.363037	0.420414	2	car
16	42.650791	594.617126	249.827774	842.981567	0.323884	12	parking meter
17	52.236305	85.413116	162.333313	319.770447	0.311944	9	traffic light
18	633.279602	293.832489	675.266357	421.886810	0.302389	9	traffic light
19	1210.249756	324.015930	1260.591797	405.222839	0.286106	9	traffic light

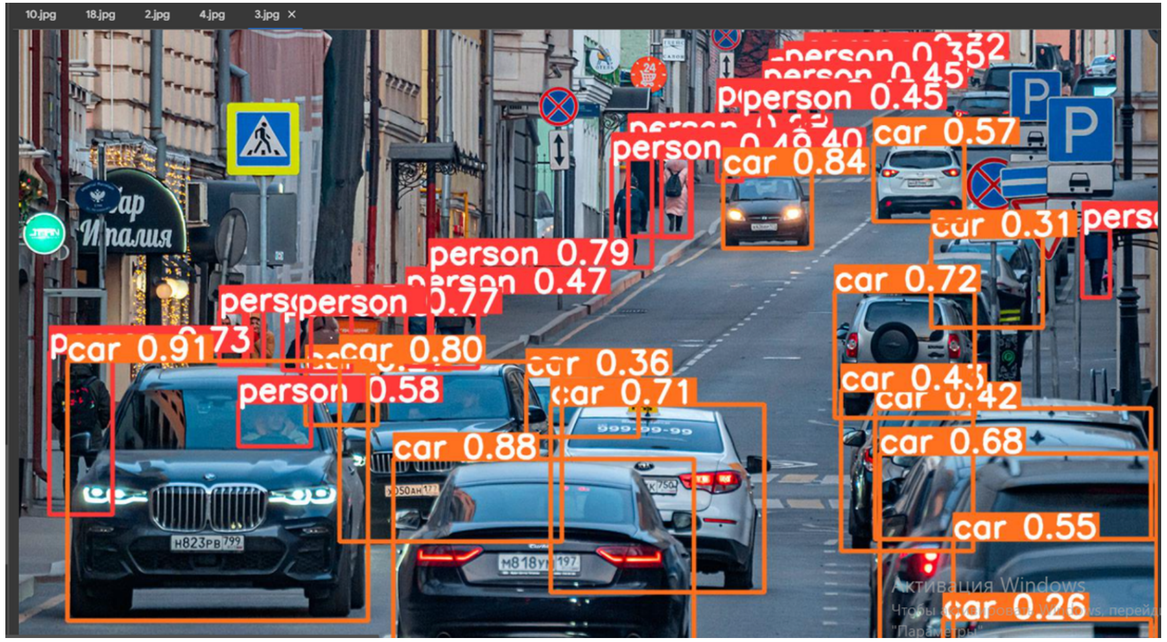
Information about detected objects in images

```
%time
results=model(img_files)
results
```

```
CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 8.34 µs
YOLOv5 <class 'models.common.Detections'> instance
image 1/17: 6000x4000 9 cars, 1 stop sign
image 2/17: 917x1279 17 persons, 1 dog, 3 backpacks, 2 handbags
image 3/17: 512x869 1 person, 2 buss
image 4/17: 900x1440 18 persons, 15 cars
image 5/17: 1050x1680 13 persons, 7 cars, 1 traffic light
image 6/17: 178x283 3 cars
image 7/17: 168x300 1 car
image 8/17: 1800x2880 5 persons, 22 cars
image 9/17: 1080x1920 2 persons, 1 bicycle, 9 cars, 6 traffic lights, 2 parking meters
image 10/17: 440x774 4 airplanes
image 11/17: 1080x1920 3 persons, 2 trains
image 12/17: 471x640 2 birds
image 13/17: 450x800 1 cat, 1 dog
image 14/17: 415x740 2 birds
image 15/17: 1667x2500 5 persons, 1 car, 1 truck
image 16/17: 1197x1920 19 persons, 1 backpack, 3 suitcases
image 17/17: 2448x3264 5 boats
Speed: 564.7ms pre-process, 555.1ms inference, 2.4ms NMS per image at shape (17, 3, 640, 640)
```

Some output images





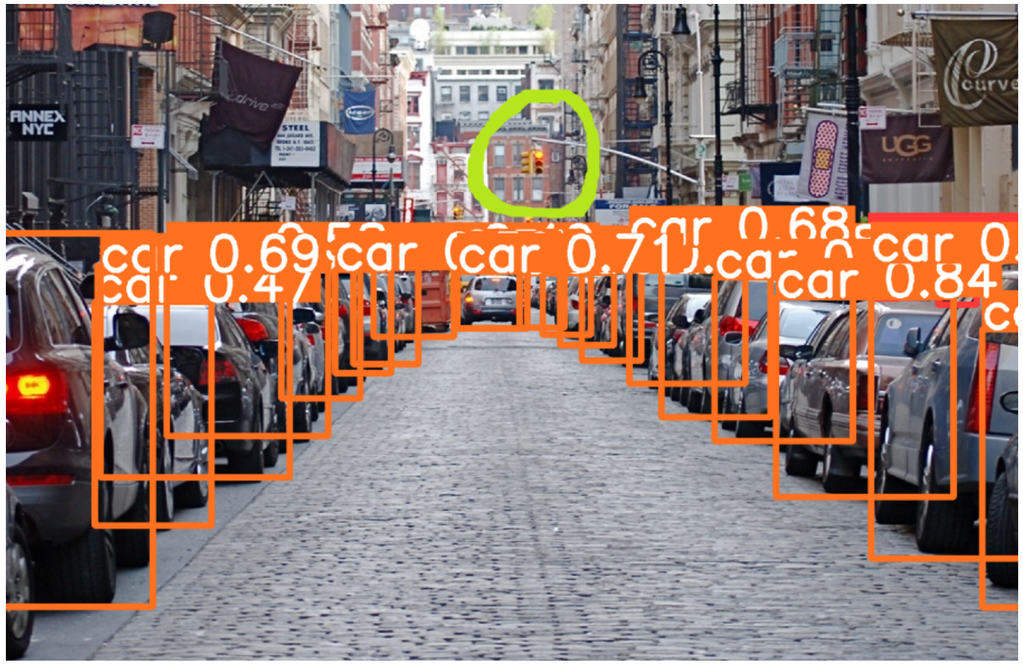
Discussion

- **Critical review of results**

Advantages:Speed,processes frames quickly.Detects objects with high probability.Fairly accurate localization

Disadvantages:Comparatively low recall and more localization error compared to Faster R_CNN.Struggles to detect close objects because each grid can propose only 2 bounding boxes.Struggles to detect small objects

For example:Didn't find the traffic light here, but found it in another image



- **Next steps**:Possible to add a large number of images and look at the speed and accuracy of the model.Efficiently train the object detection algorithm YOLOv5 on your own custom dataset. Possible to connect the camera and detect objects in real time.Real-time detection can be applied in face recognition, self-driving cars or robotics

- **Changing the stack:**YOLO is faster than Retina Net.It works significantly faster than other detection methods with comparable performance (hence the name "You only look once").But YOLO is less accurate and therefore suitable for tasks where real-time performance is a priority. In addition, you can easily find a compromise between speed and accuracy by simply resizing the model without having to retrain the model.

Sources

The practical guide for Object Detection with YOLOv5 algorithm:
:https://towardsdatascience.com/the-practical-guide-for-object-detection-with-yolov5-algorithm-74c04aac4843#:~:text=To%20achieve%20a%20robust%20YOLOv5,to%20reduce%20false%2Dpositives%20errors

What is YOLOv5:
https://blog.roboflow.com/yolov5-improvements-and-evaluation/

A Gentle Introduction to Object Recognition With Deep Learning:
https://machinelearningmastery.com/object-recognition-with-deep-learning/

Object Detection Algorithms and Libraries:
https://neptune.ai/blog/object-detection-algorithms-and-libraries#:~:text=%E2%80%93%20RetinaNet%20is%20currently%20one%20of,and%20accurate%20results%20for%20images.

What is YOLO algorithm? | *Deep Learning Tutorial 31*:https://www.youtube.com/watch?v=ag3DLKsl2vk&t=3s