

**COMPUTER VISION RESEARCH**  
**Objects Recognition in Video Sequences**  
**Evaluation and Test through Three State of the Art (SoA)**

**Joe Wild**

**Abstract**

MMTracking (an open source video perception toolbox by PyTorch – MMLab) [1], EVA (vanilla ViT) [2], Deep Learning in :1/ On stage (YOLO models, RetinaNet, and SSDs) or, 2/ Two stages (R-CNN), ho wait! Non-Maximum suppression is great too! [3]

So many choices, so many tools, but first: **What's the goal?**

Time – Money - Persistence (Motivation), are main elements to take into consideration in almost all the domains, as a Team ,or/ and, as a “Self-Worker” working remotely or not, motivation decreases with difficulties over time, especially if a CEO, or a superior is waiting for prompt results without understanding the complexity of the project.

Let's try to understand all the real processes behind the scenes with one task in Computer Vision.

At first sight, it looks very easy to orchestrate an Object recognition model for instance; Nowadays Internet give us the illusion of the: “Everything done on the internet can be done without effort at Home”. OK, hence... What's the price? Let's test!

We want to develop a computer vision system taking an input video (real time or not) and able to turn it back with recognition of some objects previously labeled.

Let's try different Algorithms and approaches. What will be the challenges?

Computer Science is a domain that evolves quickly. In January 2023, Ultralytics released a new YOLO, the YOLOv8. Ready to test? ✓

What doesn't run equivalently while being close? DETection TRansformer (DETR)? Introduced in 2020 by FAIR (Facebook AI Research) it's a sequence-to-sequence model (uses Encoder & Decoder). [4]

Now, what about the RT-DETR? For Real-Time Detection. Validated for test2 ✓

Now, you've probably heard something named Detectron2. [5]

[“Originally succeeding the Detectron library released in 2018, Detectron2 transitioned from the Caffe2 to PyTorch framework, enhancing its flexibility and ease of use. Regular updates have introduced features like panoptic segmentation and keypoint detection, reflecting Facebook's commitment to open-source computer vision research.”]  
[1]

Seems to be good and versatile. OK, this will be our third element. Selected as Test3 ✓  
Now as we said, \_“What’s the purpose of our trip?” -----> “Testing”

Test:

With a Dell Laptop Alienware on premises (VsCode, Jupyter, Anaconda...), and servers.

Media for Test: <https://www.youtube.com/watch?v=3MmSeu0pksc>

We are going to track some classes elements such as [Animal-Building-Human-Vehicle]

---

## I. The “Runing Test”

### Which ones? (“Harder, Better, Faster, Stronger...”)

**Objectives:** Release an output relatively reliable, Fastly, without crash, accurate (Precision, Recall, F1 Score, Accuracy, IOU, MAE-Mean absolute Error, Receiver Operating Characteristic (ROC) Curve, ... [6]) and not too much expensive in resources and in money. In an Economic Society the feasibility in time depends greatly on this last one. This must also be considered.

#### COMPUTER

NVIDIA-SMI 555.99										Driver Version: 555.99										CUDA Version: 12.5									
GPU		Name		Driver-Model										Bus-Id		Disp.A		Volatile		Uncorr. ECC									
Fan		Temp		Perf		Pwr:Usage/Cap												Memory-Usage		GPU-Util		Compute M.							
																						MIG M.							
0		NVIDIA		GeForce RTX 3070		...		WDDM		00000000:01:00.0		Off						N/A											
N/A		42C		P3		24W / 50W				0MiB / 8192MiB						0%		Default											
																		N/A											
Processes:																													
GPU		GI		CI		PID		Type		Process name										GPU Memory									
		ID		ID																Usage									
No running processes found																													

### Phase Tests Presentation

#### Steps

#### 1/ YoloV8 (You Only Look Once)

Architecture developed by Ultralytics for Objects detection on a Custom Dataset [7] with Roboflow (Easy tools for computer Vision). In a Collab.research.google environment \* [8].

(\*“Colab is a hosted Jupyter Notebook service that requires no setup to use and provides free

of charge access to computing resources, including GPUs and TPUs. Colab is especially well suited to machine learning, data science, and education”)

“YOLOv8 (medium) has a 50.2 mAP score at 1.83 milliseconds on the COCO dataset and A100

TensorRT

Read more at: <https://viso.ai/deep-learning/yolov8-guide/>”

**2/ RT-DETR** (Real-Time Detection Transformer), developed by Baidu, uses Vision Transformers (ViT). High performance on CUDA Architecture with TensorRT. Primarily uses the AdamW optimizer for training DL models effectively, especially Vision transformer like RT-DETR. RT-DETR Large in Python language into VsCode. [9]

**3/ DETECTRON2**, trained with a Custom dataset. In a Colab research Google + Vscode. (drive for the content). Built by Facebook AI Research (FAIR) (Meta) [10]

---

## II. Evaluation Criteria

### Tests Implementation, Adaptation & Review

#### 1/ YOLOV8

**Step1** \_ Starting the search, the existence of Roboflow, a computer vision developer framework, turns out to be a precious source of information.

The "YOLOv8: How to Train for Object Detection on a Custom Dataset" video on YouTube provides a useful thread to follow.

After creating a Roboflow account, we follow the procedure for setting up a workspace and a new object detection project. Everything is controlled through the platform.

We are invited to import YouTube video links or other media sources and then create a dataset. To experiment with the algorithm, I tried manual labeling by myself, and labeled 92 images or 4 classes:

## Classes

[Add Classes](#)[Modify Classes](#)

### ☐ Lock Annotation Classes

Locking classes will prevent new classes from being added to the project from any source. This includes in the Annotation Tool, using Label Assist, and uploading new annotation classes.

Color	Class Name
-------	------------



Animal



Building



Human



Vehicle

Unfortunately, the Precision of this first training wasn't good enough to be kept. (64% Precision).

We must try the Automatic labeling. Here is the result:  
Object Detection Test2 » Dataset Health Check

Generated on June 11, 2024 at 9:30 am. [Regenerate](#)

Images

106

0 missing annotations  
0 null examples

Annotations

506

4.8 per image (average)  
Across 4 classes

Average Image Size

0.23 mp

from 0.23 mp  
to 0.23 mp

Median Image Ratio

640x360

4:3 wide

Class Balance

Manage Classes

Rebalance Splits

all train valid test

Train Valid Test

Building

284

Vehicle

96

Human

67

Animal

59

under represented

under represented

object-detection-test2-cfacj/1

More Metrics

Visualize

Model Type: Roboflow 3.0 Object Detection (Fast)

Checkpoint: COCO

mAP

81.5%

Precision

77.4%

Recall

77.0%

## Step2\_ Setting up the Python environment:

Following the “train-yolov8-object-detection-on-custom-dataset.ipynb” notebook in the google collab research [\[Link\]](#), we checked the access of our GPU and Install YOLOv8 with the collected packages.

Ultralytics YOLOv8.0.196 Python-3.10.12 torch-2.3.0+cu121 CUDA:0 (Tesla T4, 15102MiB)  
Setup complete (2 CPUs, 12.7 GB RAM, 30.2/78.2 GB disk)

It's mandatory to verify if WSL (Windows Linux Ubuntu) is needed and if applicable install it correctly.

```
Installing: Ubuntu
Ubuntu has been installed.
```

✓

Now, we can continue following the Steps with our custom dataset made previously.

- Before you start
- Install YOLOv8
- CLI Basics
- Inference with Pre-trained COCO Model
- Roboflow Universe
- Preparing a custom dataset
- Custom Training
- Validate Custom Model
- Inference with Custom Model

Just to note that at this point in the study, we are coming across links like: [\[This\]](#), making it very difficult to stay focused on our current task. (Cloning notebooks can quickly turn into an obsession).

To summarize, at the end of the overall process, we have our video with the four recognized object classes. Both processes work on Roboflow and VSCode, although there are more issues and wasted time due to the environment interactions compared to an online framework.

[Search Google Lab](#)

**Link of the OUTPUT VIDEO: [Final](#) [<https://youtu.be/II4ubNbRf9s>]**

### **Pros and Cons Reflection:**

YOLO8v on Roboflow:

- + Very easy / Don't require a lot of resources/
- \$249 per month, billed monthly

YOLO8V deployment in Python on Vscod:

- + "Relatively" easy
- Time / Resources / Compatibility between the different modules & environments

YOLOV8 deployment last test on Jupyter notebook / Jupyter Lab:

- + Visually clearer
- Time / Resources/

## **2/ RE-DETR L**

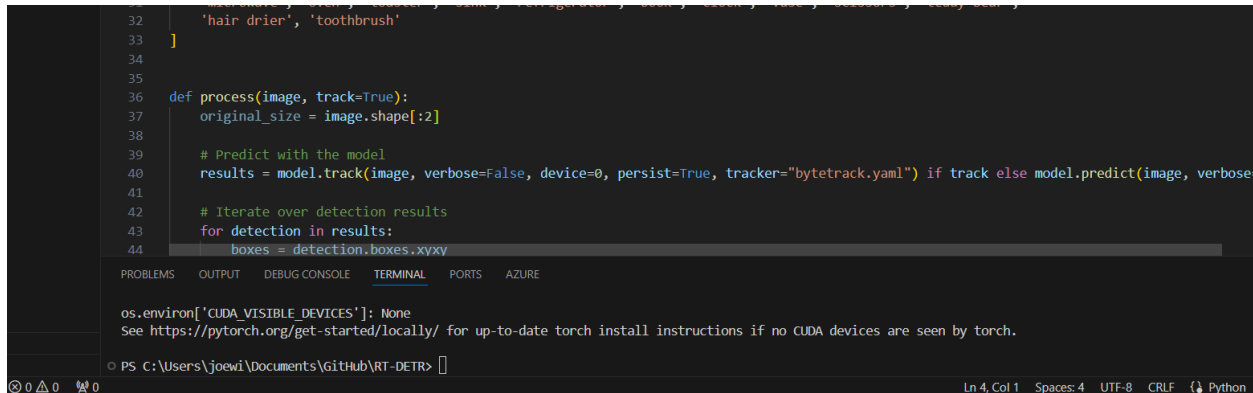
**Step1** \_ Looking for the good code [\[GitHub\]](#) \_ "Pick, Clone and Thanks".

[GitHub - bharath5673/RT-DETR: Ultralytics RT-DETR (Realtime Detection Transformer)]

First Error:

The environmental problem seems to be a recurrence and must be imperatively resolved.

“Wasted of Time”: around 4 hours :/



```
32     'hair drier', 'toothbrush'
33 ]
34
35
36 def process(image, track=True):
37     original_size = image.shape[:2]
38
39     # Predict with the model
40     results = model.track(image, verbose=False, device=0, persist=True, tracker="bytetrack.yaml") if track else model.predict(image, verbose=
41
42     # Iterate over detection results
43     for detection in results:
44         boxes = detection.bboxes.xyxy
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

os.environ['CUDA\_VISIBLE\_DEVICES']: None  
See https://pytorch.org/get-started/locally/ for up-to-date torch install instructions if no CUDA devices are seen by torch.

PS C:\Users\joewi\Documents\GitHub\RT-DETR>

Ln 4, Col 1 Spaces: 4 UTF-8 CRLF Python

Finally, starting from scratch ended up solving the library and environment problem between Vscode and Anaconda.

**Step2**\_Trying to play with the dataset on Coco to find the same classes as the first test. Unfortunately, the time for this test has largely expired.

Let's move on.

**Link of the OUTPUT VIDEO:** [Final \[https://youtu.be/wi8aM9dG0kE\]](https://youtu.be/wi8aM9dG0kE)

## Pros and Cons Reflection:

RE-DETR L:

- A thorough review of every line of code during installation is indispensable; (my fault)
- It's crucial to be cautious with the environments and all the associated lines of code.
- + The code is intuitive and runs very efficiently, living up to its name and reputation.
- + Time (Real-Time Detection Transformer). Output with object recognition in '45seconds.

[Transformer's information](#)

## 3/ DETECTRON2

**Step1** \_ Instruction and codes

[Instruction](https://www.youtube.com/@ComputerVisionEngineer) by Felipe: <https://www.youtube.com/@ComputerVisionEngineer>

Implementation:

notebook “Train\_Detectron2\_Object\_Detector\_Custom\_Data.ipynb” in Google research (under MIT License). Connection to Google drive. Adding the necessary files into the drive.

Taking care of being on the T4 GPU Run time and to be on the good folder into the drive:

```
from google.colab import drive  
  
drive.mount('/content/drive')
```

Modification of drive.mount

## Step 2\_ libraries

```
!pip install torch==2.0.0  
!pip install torchvision==0.15.1  
!pip install opencv-python==4.6.0.66  
!pip install matplotlib==3.5.3  
!pip install git+https://github.com/facebookresearch/detectron
```

Following the process...

Primary issue in the package Installation. We start all over again. It works.

Restart Session. “Pip Detectron2” only works with Linux. Unfortunately, we’re working on Anaconda. Hence, we changed for: conda install detectron2.

Anaconda and detectron2 are not in a good relationship even when founding this kind of link:

[\[Link\]](#) we are going in C++ ...

At the end, after more than 5 hours. No real solution are found with the installation of Detectron2

Example of Error msg:

```
!python train.py --device gpu --learning-rate 0.00001 --iterations 6000
```

It seems to be because the environment isn’t appropriate, again, and it doesn’t find the required module.

After what >

Thanks to the website forum Stake overflow and this source: <https://dev.to/reckon762/how-to-install-detectron2-on-windows-3hiln> ,

we don’t give up and found:

“Open a terminal or command prompt. Create a new environment called detectron2-env with the following command:

```
conda create --name detectron2-env python==3.9 -y
```

Activate the environment with the following command:

**Linux** conda activate detectron2-env

**Windows** activate detectron2-env

Install the dependencies with the following commands:

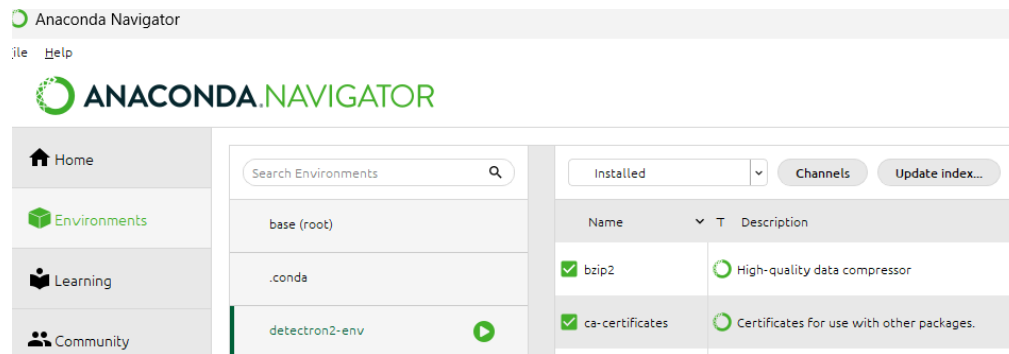
```
pip3 install torch torchvision torchaudio
```

```
git clone https://github.com/facebookresearch/detectron2.git
```

```
python -m pip install -e detectron2
```

After much research, a new bug comes:

<https://github.com/facebookresearch/detectron2/issues/5008>



It seems there will be no end...

The tests on this project aim to evaluate the implementation of these different models. As a team composed of just myself, it is clear that I do not have enough expertise to work as an expert who would probably solve everything in less time."

### Pros and Cons Reflection:

Detectron2:

- Looks to be not well-adapted to this configuration (Windows 11, Anaconda, Python).

Perhaps powerful but not capable of fitting the test and evaluation criteria.

<https://github.com/facebookresearch/detectron2/issues/810#issuecomment-596194293>

<https://stackoverflow.com/questions/75357936/how-to-install-detectron2>

**DETECTRON2** More manageable under Linux /**CUDA** / **MIT LICENCE**

---

## III. Conclusion

It's been a long journey, full of trials and work, but to conclude with the greatest possible objectivity, online services and platforms offer a huge toy box for those who want to create without coding. But at what cost? This highlights the fact that we're living in a World of "Work & code, or Pay".

The First Test with Yolov8 and the help of Roboflow was a piece of cake. Positive in Time, easy to understand and to train. We don't have access to the back of the cards but at the end we have a template which can be used with other media.

Yolov8 is a little slower in Python environments, but we have the upper hand on the codes, and



therefore more independence and autonomy.

For, RT-DETR due to a good contributor [\[Github\]](#), the code worked well. Despite a misinterpretation in the operational process, everything was put back in place. More difficulties training the model with our custom dataset. [Horse and Building are missing]; It would have just taken longer.

Regarding the third test, the Detectron2, arduous to say since a battle between os, environments and libraries seems to have been fought. It's a pity, working on google.search fits well with the fun aspect of learning. The answer will probably be further down the line.

To conclude, the framework around Computer Vision in this project, precisely for objects recognition, is that is a vast adventure where the final configuration is determined by the needs and preferences of the user. We simply ask ourselves who we want to be: A User using SaaS or the Creator of this kind of service...

[11]

## REFERENCES

- [1] <https://github.com/open-mmlab/mtracking>
- [2] <https://arxiv.org/abs/2211.07636>
- [3] [https://builtin.com/machine-learning/non-maximum-suppression#:~:text=Non%2Dmaximum%20suppression%20\(NMS\)%20is%20a%20post%2Dprocessing,Image%3A%20Shutterstock%20%2F%20Built%20In](https://builtin.com/machine-learning/non-maximum-suppression#:~:text=Non%2Dmaximum%20suppression%20(NMS)%20is%20a%20post%2Dprocessing,Image%3A%20Shutterstock%20%2F%20Built%20In)
- [4] <https://medium.com/@faheemrustamy/detection-transformer-detr-vs-yolo-for-object-detection-baeb3c50bc3>
- [5] <https://www.modelbit.com/model-hub/detectron-2-model-guide#:~:text=Release%20and%20Development,flexibility%20and%20ease%20of%20use.>
- [6] <https://viso.ai/computer-vision/model-performance/>
- [7] <https://www.youtube.com/watch?v=wuZtUMeiKWY&list=PLZCA39VpuaZZJ-aS7B7pZVrD9AtRx8-7u&index=15/>
- [8] <https://research.google.com/colaboratory/faq.html>
- [9] <https://docs.ultralytics.com/models/rtdetr/#overview>
- [10] <https://ai.meta.com/tools/detectron2/>
- [11] [Nyckel](#)